

Moving Target Defense (MTD): A Software Defined Networking (SDN) Approach

Dijiang Huang

Secure Networking And Computing (SNAC) Research Group
School of Computing Informatics and Decision Systems Engineering
Ir. A. Fulton School of Engineering
Arizona State University

5/24/2018

ICC 2018

About the tutorial speaker:

Dijiang Huang

Associate Professor

Secure Networking And Computing research group, Arizona State University

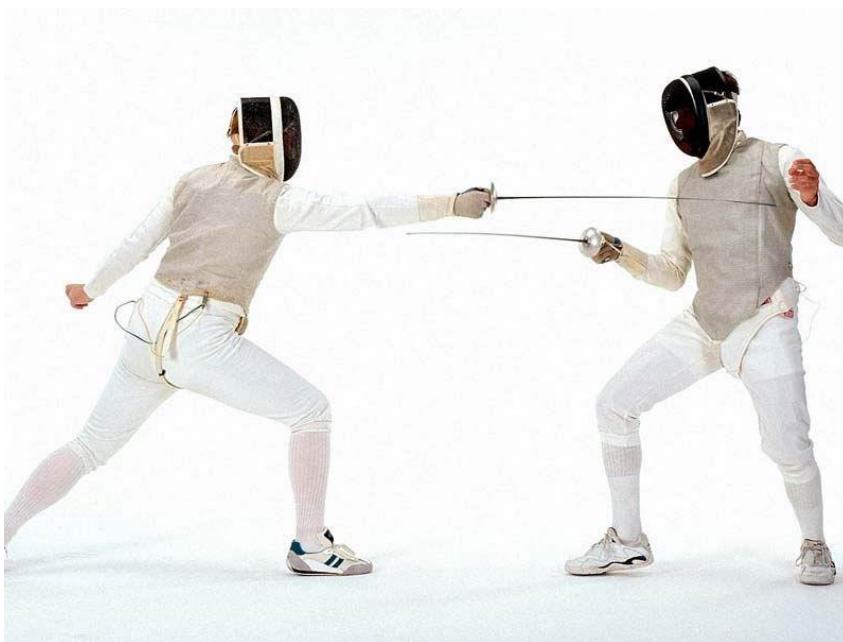
- Research area: computer network security, mobile computing, applied cryptography.
- Research Sponsored by NSF, DoD (ARO, ONR, Navy, NRL), and Industry (HP, China Mobile)
- Research areas and sponsored research.



Agenda

- 1. Moving Target Defense (MTD)**
 - What is MTD?
 - MTD Case Study
2. SDN-Based MTD Approaches
3. Q&A and Discussion

What is MTD?



Shell Game



Static Target



Moving Target



VS

5th ACM Workshop on Moving Target Defense (MTD 2018)

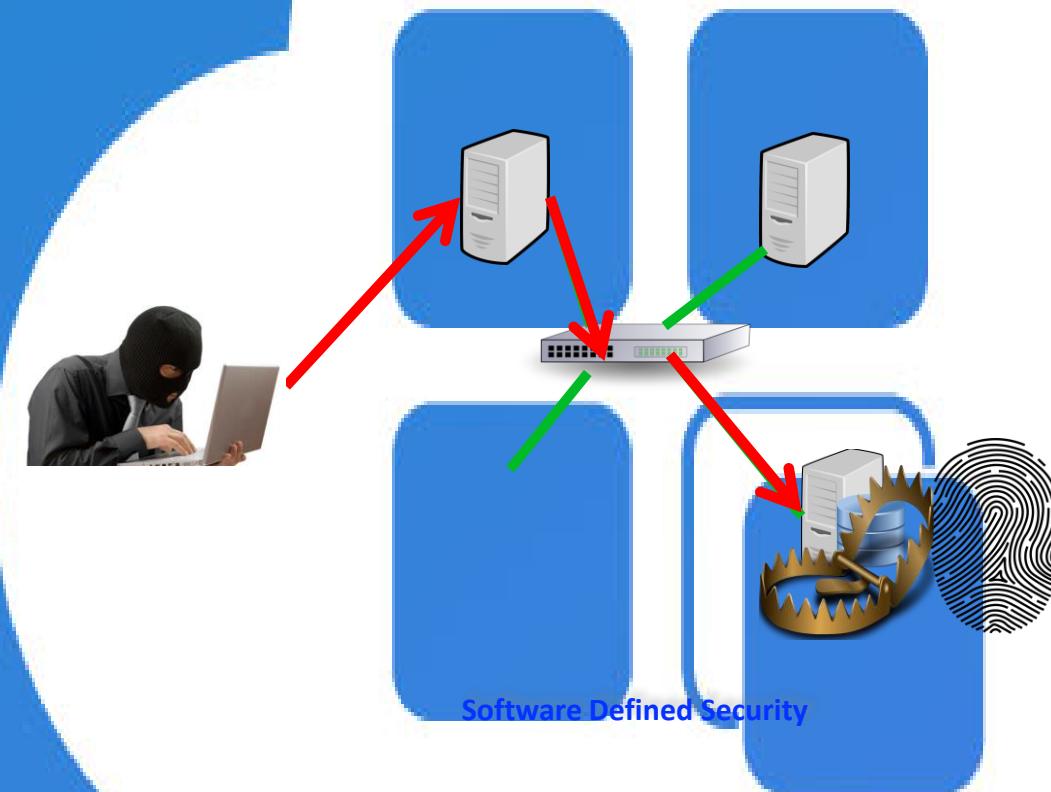
In conjunction with the 25th ACM Conference on Computer and Communications Security (ACM CCS 2018)

October 15-19, 2018, Toronto, Canada



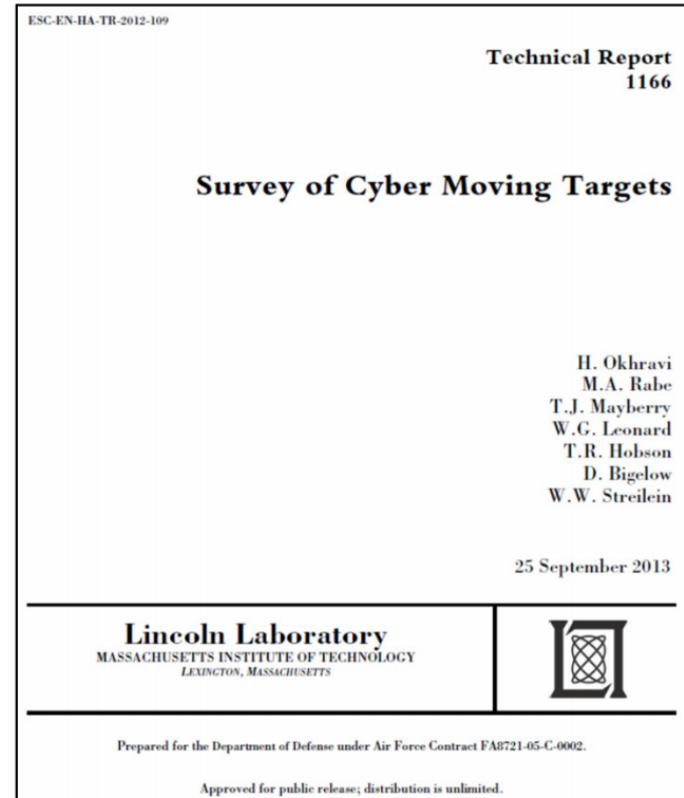
- The idea of moving-target defense (MTD) is to impose the same asymmetric disadvantage on attackers by **making systems dynamic and therefore harder to explore and predict**. With a constantly **changing** system and its **ever-adapting attack surface**, attackers will have to deal with significant **uncertainty** just like defenders do today. The ultimate goal of MTD is to **increase the attackers' workload** so as to level the cybersecurity playing field for defenders and attackers – ultimately tilting it in favor of the defender.

A network defense example of moving target defense



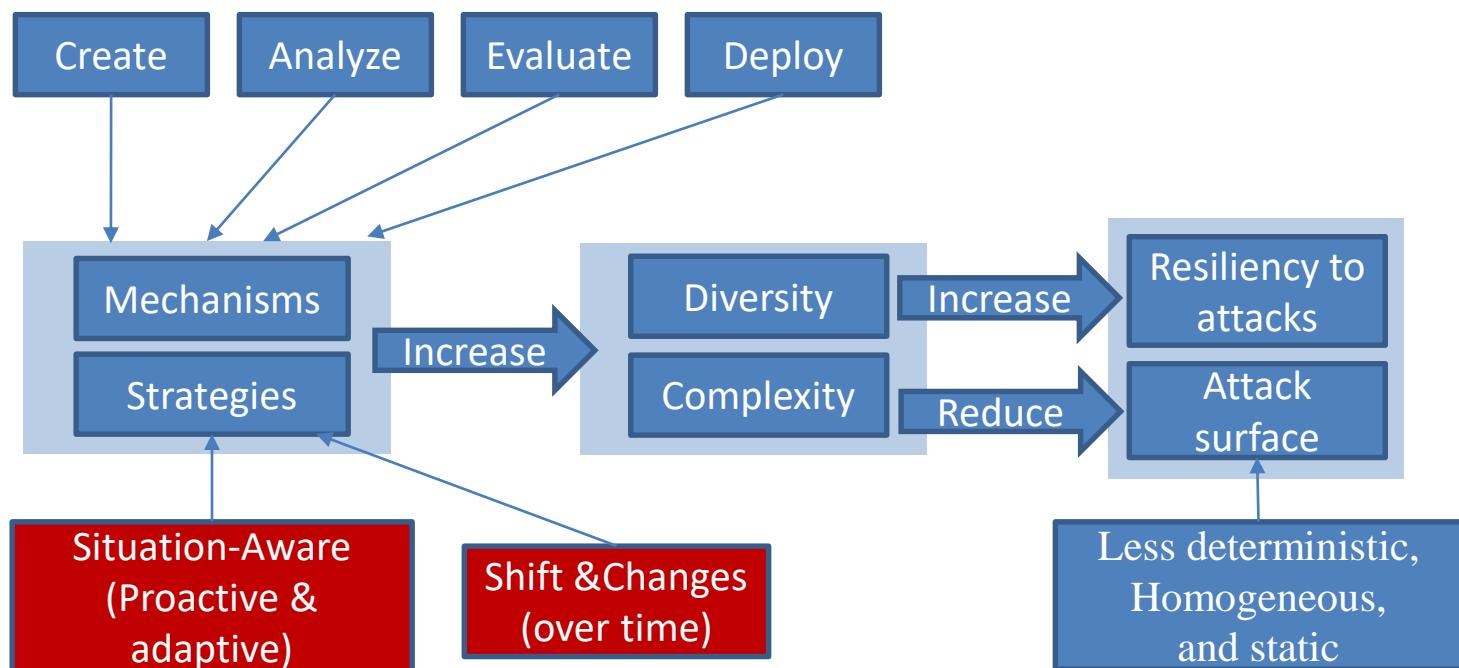
Background

- 2013 Survey:
 - 39 MTDs organized across 5 categories
- Taxonomy
 - Dynamic Platforms
 - Dynamic Runtime Environments
 - Dynamic Software
 - Dynamic Data
 - Dynamic Networks



Okhravi, Hamed, M. A. Rabe, T. J. Mayberry, W. G. Leonard, T. R. Hobson, D. Bigelow, and W. W. Streilein. *Survey of cyber moving target techniques*. No. MIT/LL-TR-1166. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2013.

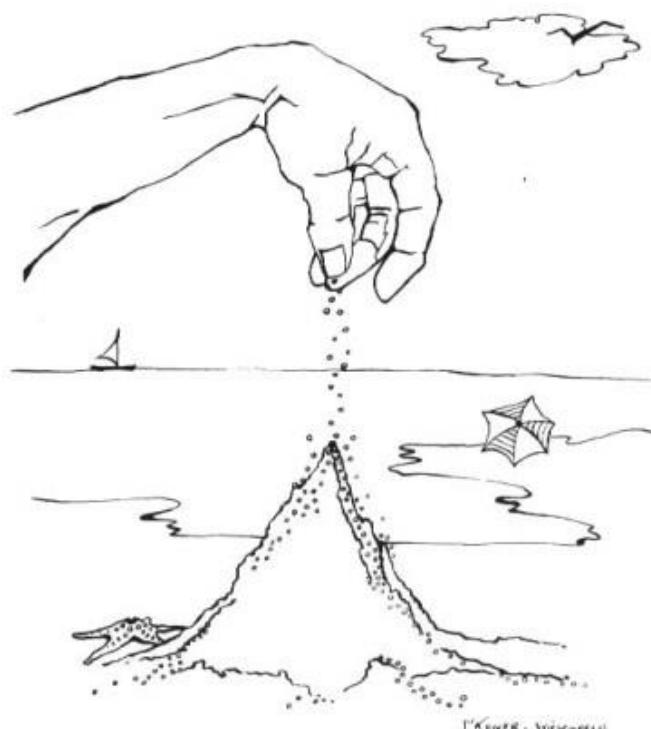
Research for Moving Target Defense (MTD)



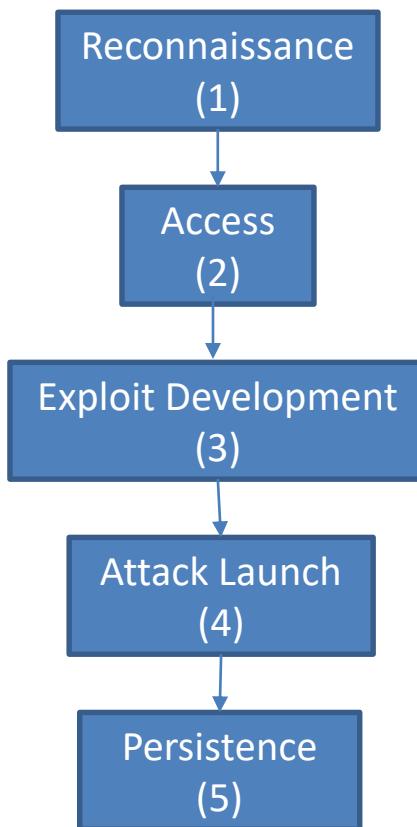
https://www.nitrd.gov/SUBCOMMITTEE/csia/Fed_Cybersecurity_RD_Strategic_Plan_2011.pdf

MTD Challenge

- The challenge is to demonstrate that MTD introduced complexity is indeed a benefit and not a liability.



MTD Areas: Cyber Kill Chain Model



- The attacker collects useful information about the target.
- The attacker tries to connect or communicate with the target to identify its properties (versions, vulnerabilities, configurations, etc.).
- The attacker develops an exploit for a vulnerability in the system in order to gain a foothold or escalate his privilege.
- The attacker delivers the exploit to the target. This can be through a network connection, using phishing-like attacks, or using a more sophisticated supply chain or gap jumping attack (e.g., infected USB drive).
- The attacker installs additional backdoors or access channels to keep his persistence access to the system.

MTD Categories

- System-based MTD
 - Software-based
 - Application, OS, Data
 - Hardware-based: processor, FPGA
- Network-based MTD
 - MAC layer: changing MAC address
 - IP layer: IP randomization
 - TCP (Traffic) layer: changing network protocol
 - Session layer

Software-based MTD

- Goals
 - Prevent unwanted modification
 - Protect software against analysis
- Types
 1. Dynamic Runtime Environment: Address Space Layout Randomization (ASLR), Instruction Set Randomization,
 2. Dynamic software: In-place code randomization, Compiler-based Software Diversity
 3. Dynamic Data

Network-based MTD

- Network reconnaissance is the first step for attackers to collect network and host information and prepare for future targeted attacks.
- **Goal:** reduce attack surface and enhance defense surface, e.g., make the scanning results expire soon or give the attacker a different view of the target system
- **Examples:** IP randomization, Port randomization

How to analyze an MTD solution?

- Specific threat model
- Technical details
- Cost, intrusiveness, usability, and benefit
- Weakness and Improvements

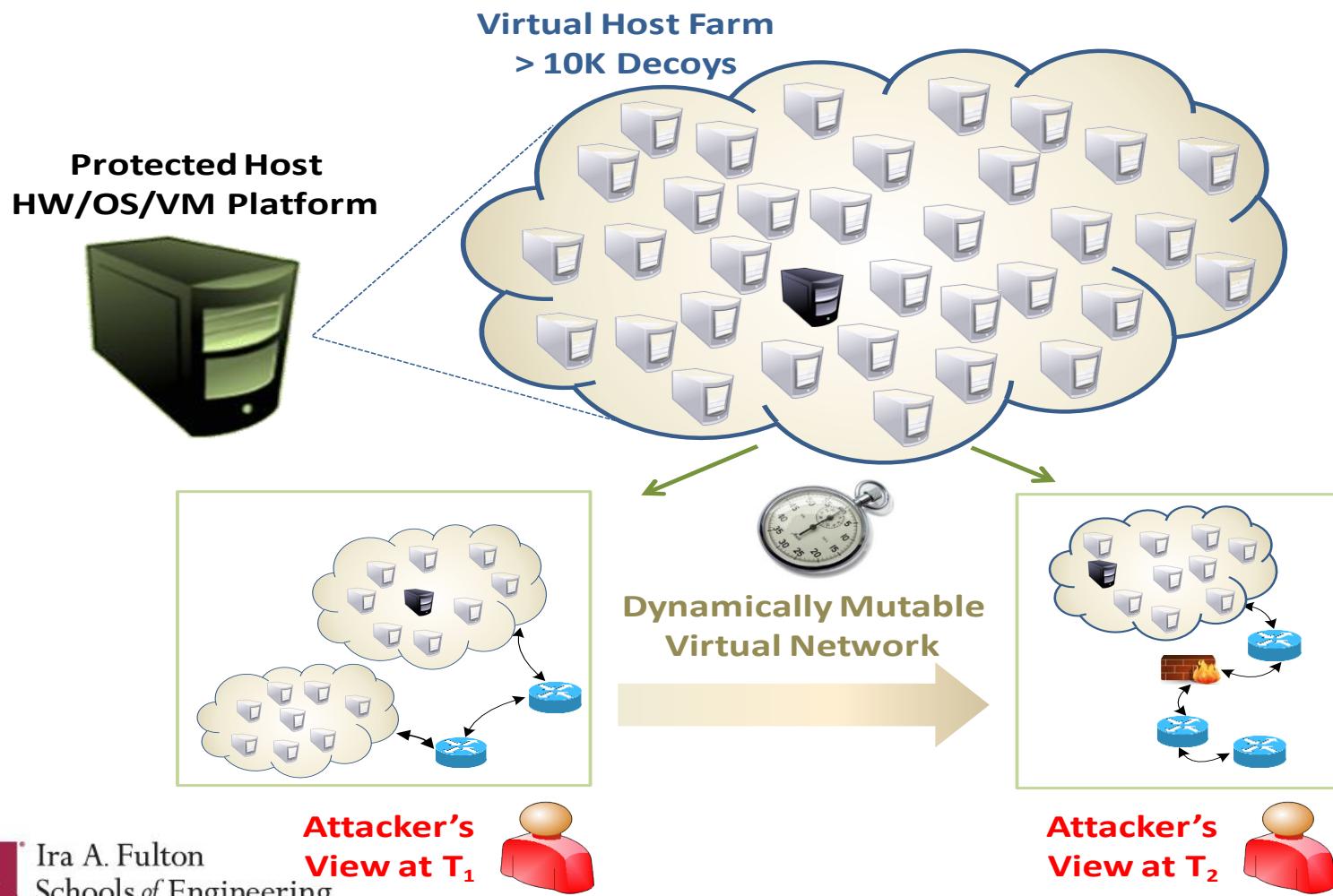
MTD Case Study

- How to defeat port scanning attacks?
 - IP/port randomization

Threat Model

- Data leakage attacks, e.g., steal crypto keys from memory
- Denial of Service attacks, i.e., exhaust or manipulate resources in the systems
- Injection attacks
 - Code injection: buffer overflow, ROP, SQL injection
 - Control injection: return-oriented programming (ROP)
- Spoofing attack, e.g., man-in-the-middle
- Authentication exploitation: cross-site scripting (XSS)
- Scanning, e.g., port scanning, IP scanning for targeted attack
- Physical attack: malicious processor
-

Dynamic Virtualized Network Topology



Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

VM-based Dynamic Virtualized Network

- 3-level decoys
 - VM level: KVM
 - OS level: OpenVZ/LXC
 - Process level: Honeyd
- Dynamic Network Topology
 - Centralized controller in the hypervisor



Two Challenges in Network-based MTD

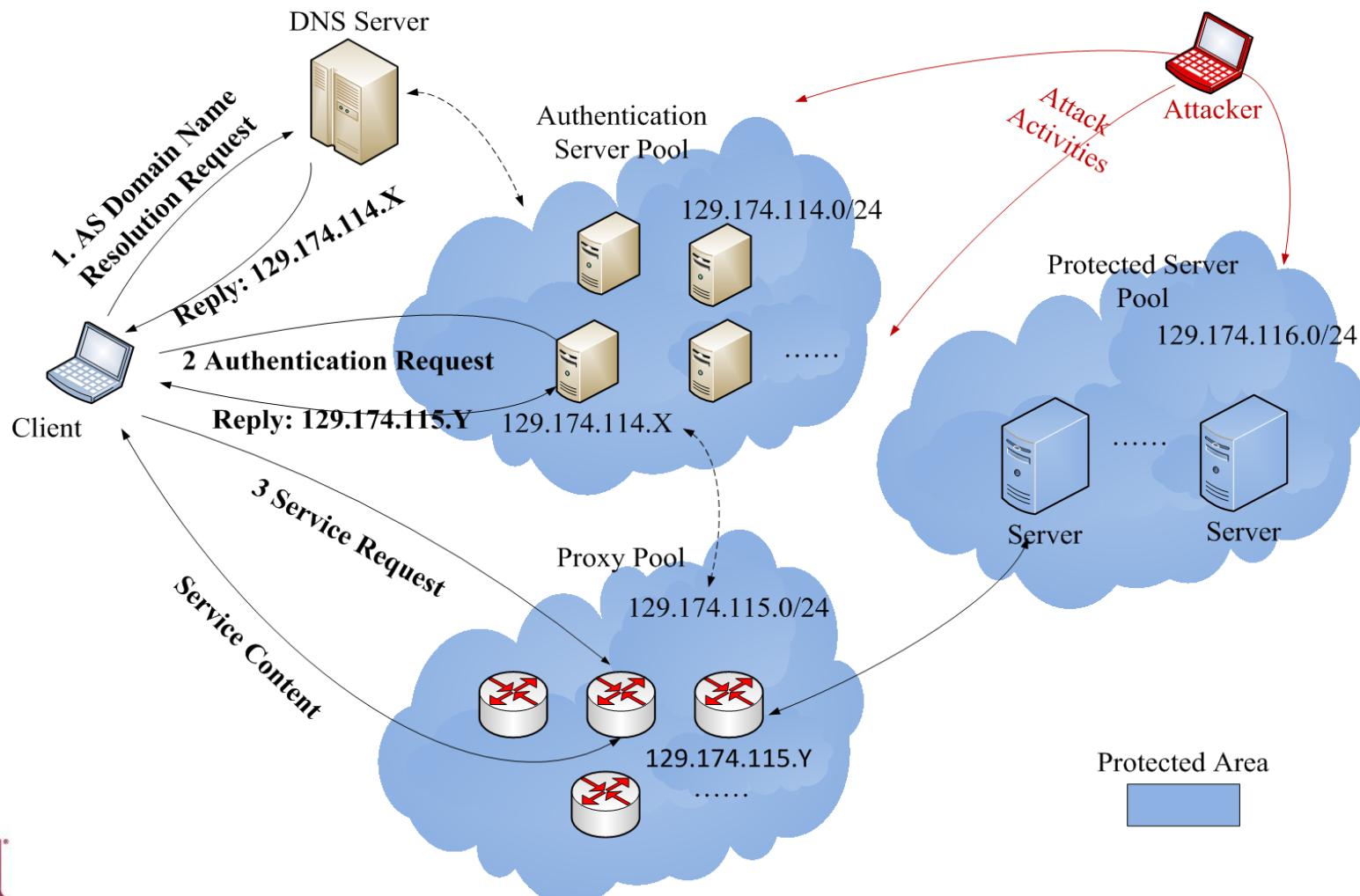
1. Service availability

- Authenticated clients should always know the new IP address/port number.
- When the IP and Port changes, the connection still maintained, minimizing service downtime.

2. Service Security

- Only the authenticated users can access the service.
- How to mitigate insider attacks?

Authentication Framework

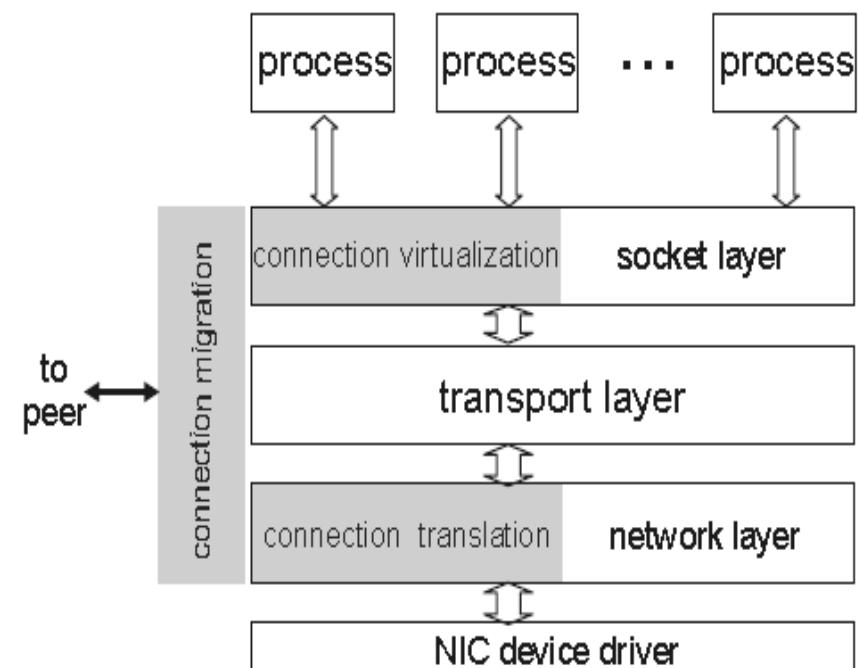


Live VM Migration

- VMware vMotion
 - Three execution states for moving a VM without service interruption
 - Disk state: shared storage such as SAN and NAS
 - Memory state: trace phase => pre-copy phase => switchover phase
 - Network State: Virtual switch, virtual NIC
 - **Require source and destination hosts on the same subnet.**
 - **Migration should be fast to prevent network connection timeout.**

Seamless TCP Connection Migration

- Keep end-to-end transport connection alive through separating **transport** endpoint identification from **network** endpoint identification.
- Three components
 - Connection virtualization
 - Connection translation
 - Connection migration



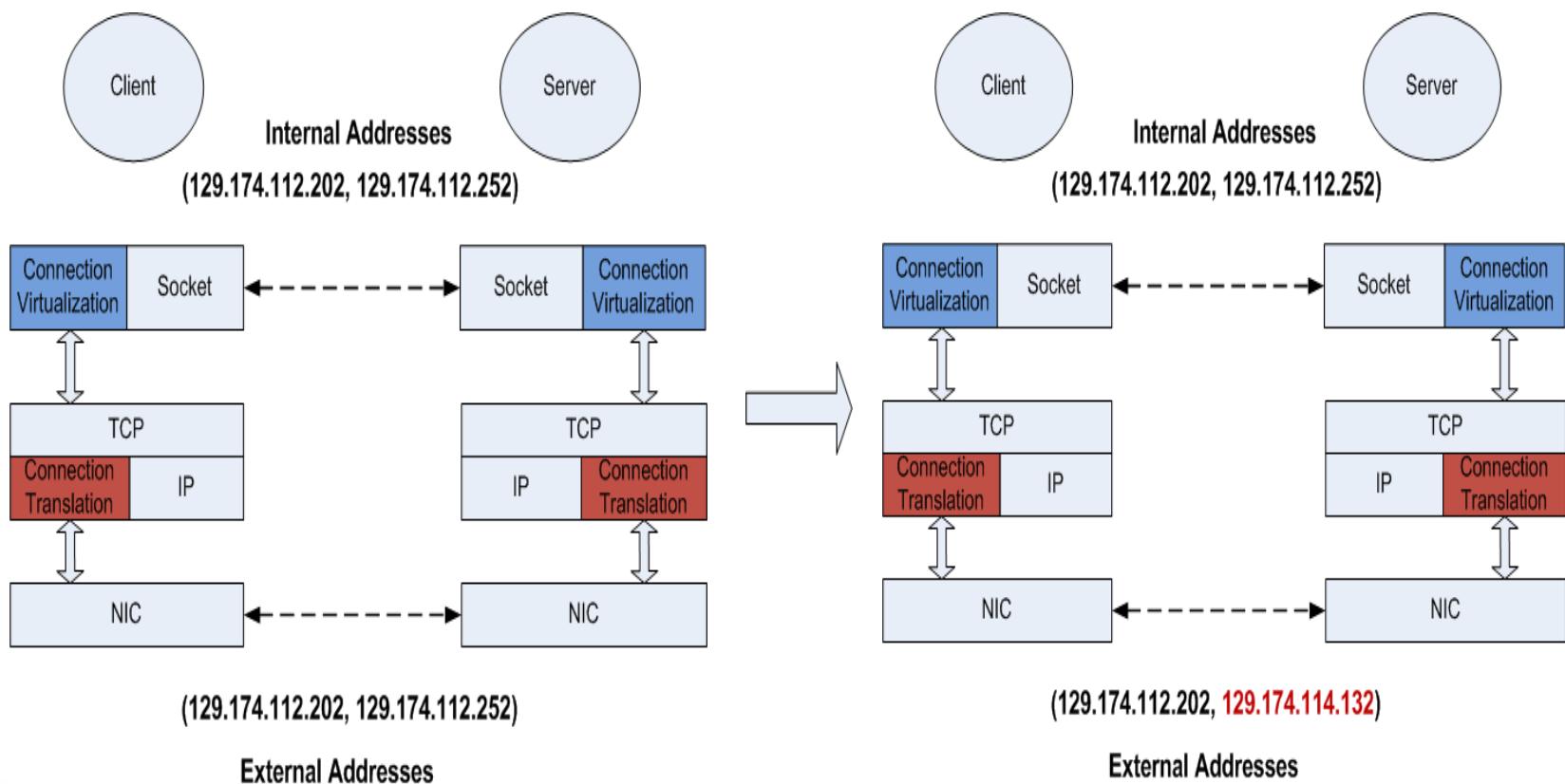
Connection Virtualization

- Internal address for applications;
 - IP and Ports
 - never changes for one connection
- External address for communications
 - IP and Ports
 - may change according to MTD requirements
- A map to translate between Internal address and External address

Connection Translation

At beginning,
internal address == external addresses

Server **changes** its IP address



Network Migration

- After the server changes its IP address and port, it will inform the client to update the internal-external address mapping.
- Migration Steps: protected by a shared secret key
 - Suspend a connection
 - Keep connection alive
 - Resume a connection
 - Update internal-external endpoints mappings
 - Server sends UPDATE packet
 - Client sends UPDATE_ACK packet
- Both endpoints need to know the same internal address pair.

Implementation

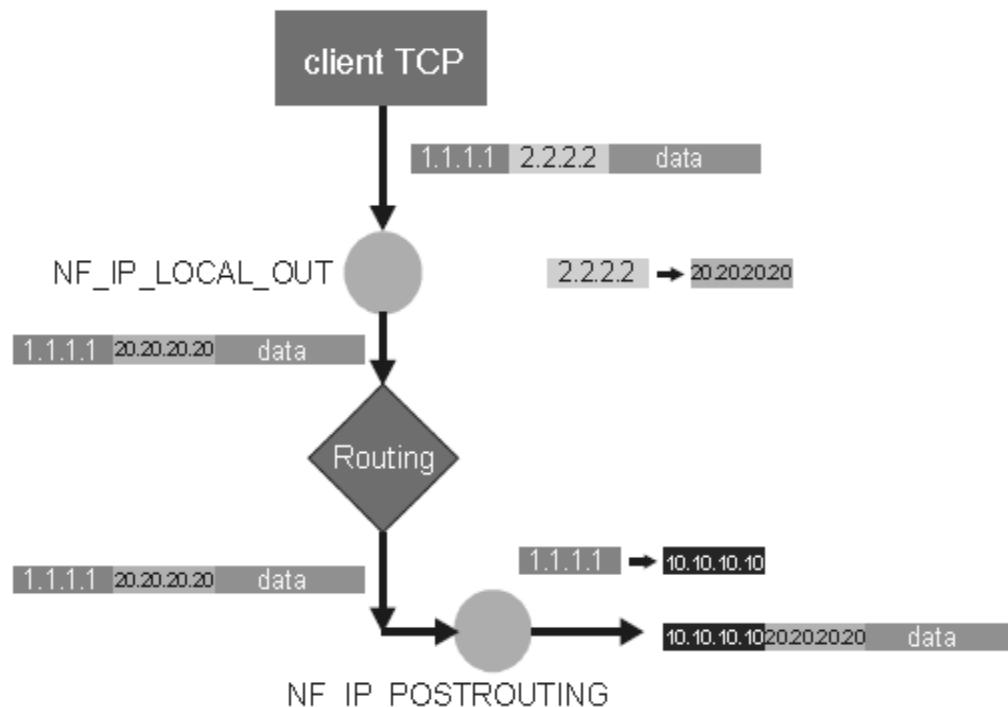
- All in a kernel module in Linux
- Support both client and server mobility
- Connection Virtualization
 - Intercept socket system calls
- Connection Translation
 - Instrument Netfilter hooks
- Connection Migration
 - Migration daemon

Intercept System Calls

- Overwrite the function pointers in the system call table
- Intercept
 - Accept()
 - Connect()
 - Close()
 - Getsockname()
 - Getpeername()

Instrument Netfilter hooks

- For outgoing traffic
 - NF_IP_LOCAL_OUT for destination address translation
 - NF_IP_POSTROUTING for source address translation
- For incoming traffic
 - NF_IP_PREROUTING for destination address translation
 - NF_IP_LOCAL_IN for source address translation

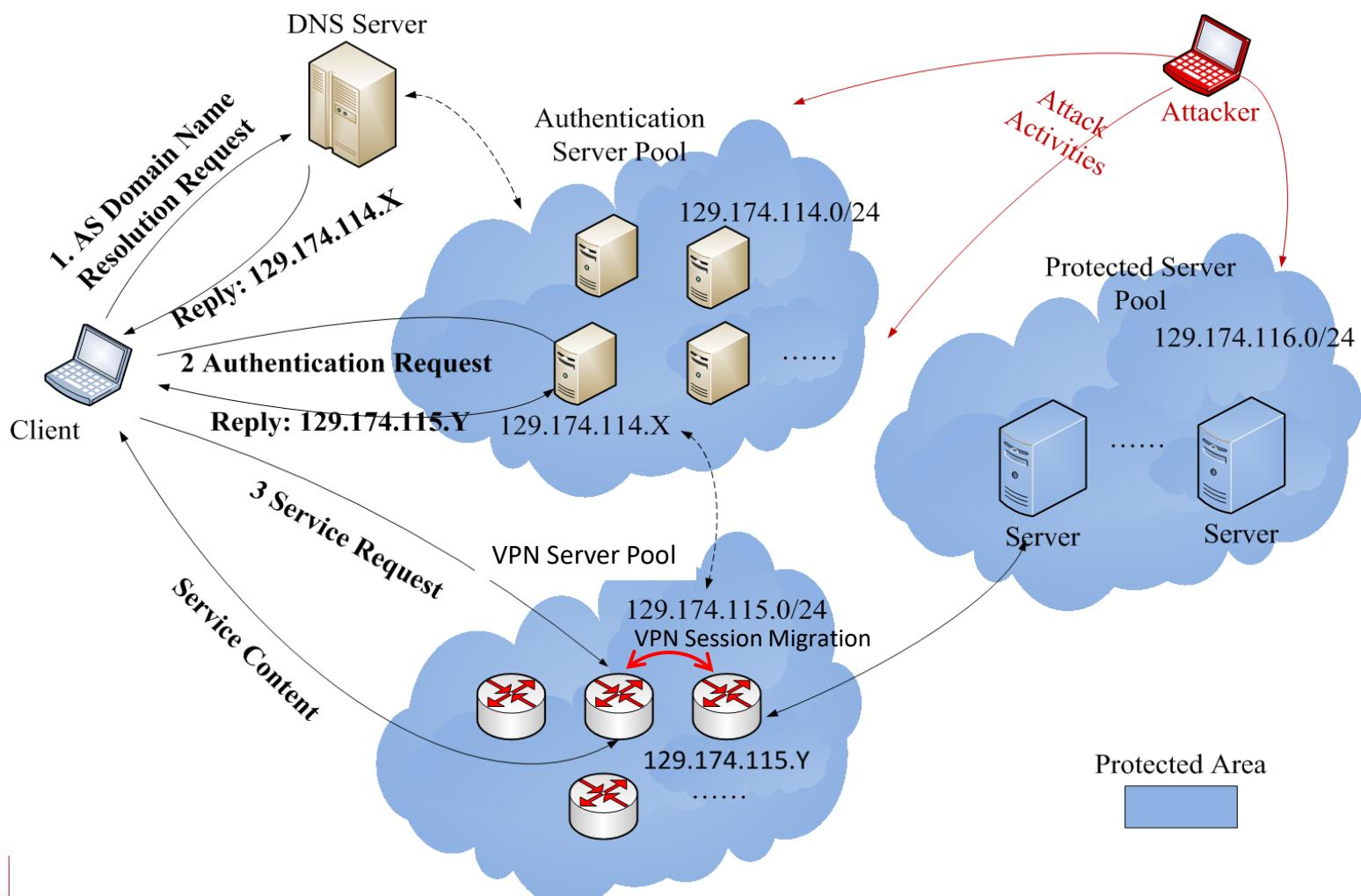


Migration Daemon

- A Kernel thread as a server process
- Initiate the suspension after receiving a suspend event from an Access Policy Manager (APM)
 - Active the connection migration helper
- Restore the connection after receiving a resumption event from APM
 - Exchange UPDATE and UPDATE_ACK packets to update the internal to external address mapping

MobiVPN: Towards VPN Migration

Mobile VPN Scenarios



Fast & Lightweight VPN Session Resumption

- In a traditional VPN (like OpenVPN), both the VPN client and server **are expected to maintain the same physical IP address** during the entire VPN session.
- In mobile environment, mobile clients will most likely **obtain new IP addresses** after a handover.
- VPN server's IP address can change too in MTD environments (IP hopping).

Fast & Lightweight VPN Session Resumption

- OpenVPN ties each VPN session with the IP address of the VPN client, and uses that IP as the **identifier** of the VPN session.
- Change of the IP address of the client → inability to locate the VPN session the VPN server → A new VPN session has to be renegotiated.

Delays Due to Network Handover

- OpenVPN:

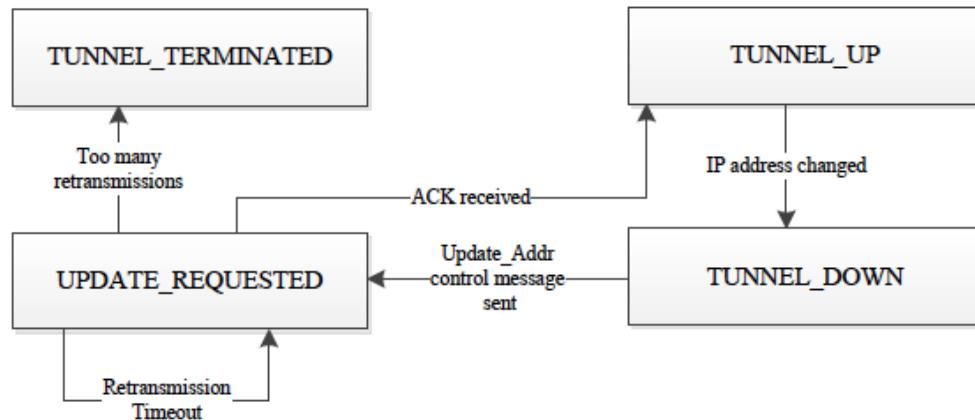
Delay	Network Handover (L2/L3)	VPN session termination	VPN renegotiation
Causes	Joining Network, authentication, DHCP etc	Inactivity Timeout (60s recommended) (Controlled by VPN server)	Full TLS handshake, pushing VPN configuration, pushing routing options etc.

- MobiVPN

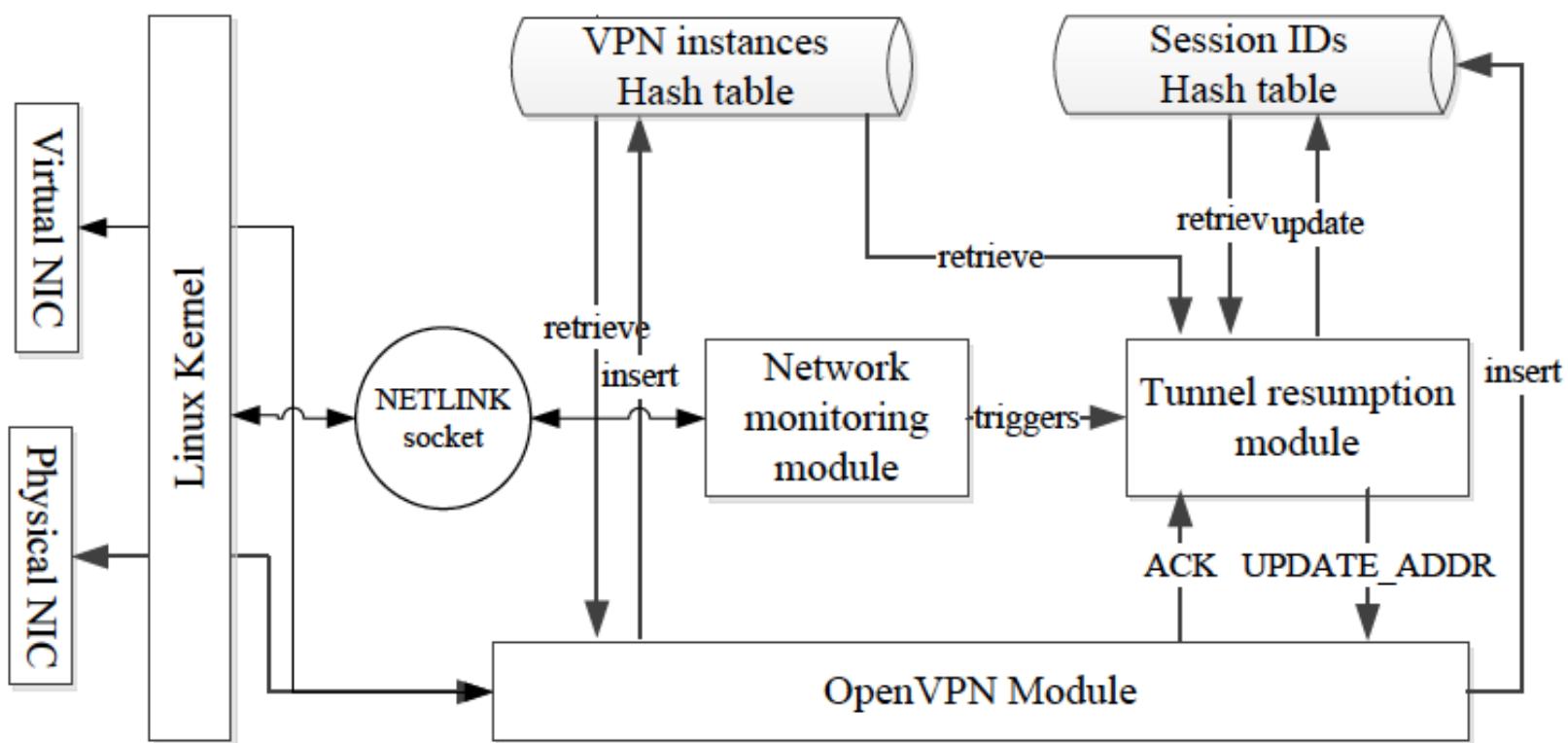
Delay	Network Handover (L2/L3)	VPN resumption
Causes	Joining Network, authentication, DHCP etc	Light-weight VPN resumption

Effect of VPN server's IP address Change

- OpenVPN:
 - The VPN server is **Passive**. Connected clients have to timeout first, before they can renegotiate new VPN sessions.
- MobiVPN
 - The VPN server is **Active**. It can uses our light-weight VPN resumption protocol to **update** all connecting VPN clients with its new IP address **eliminating** the need to renegotiate new VPN sessions.



System Design

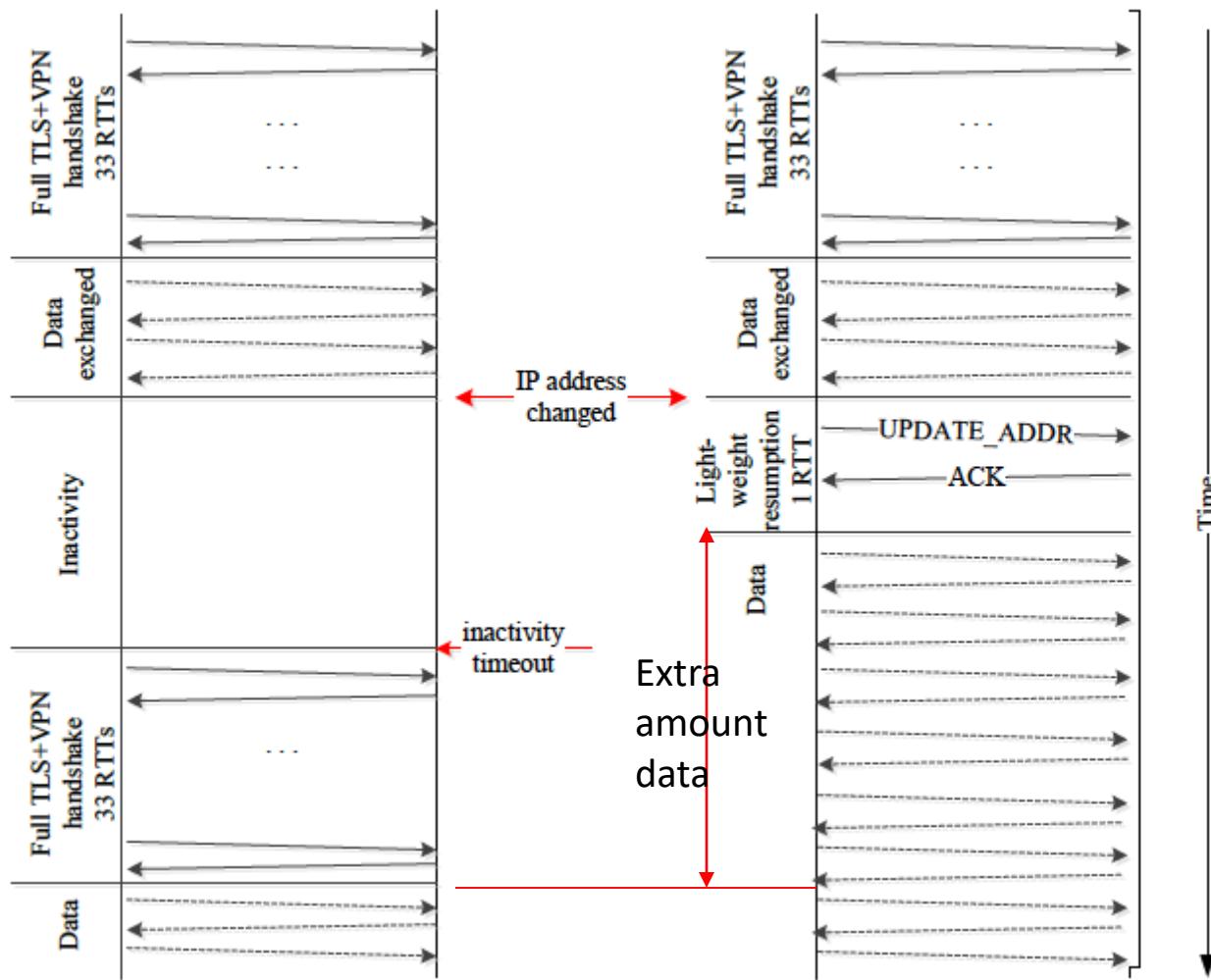


MobiVPN Hash Tables

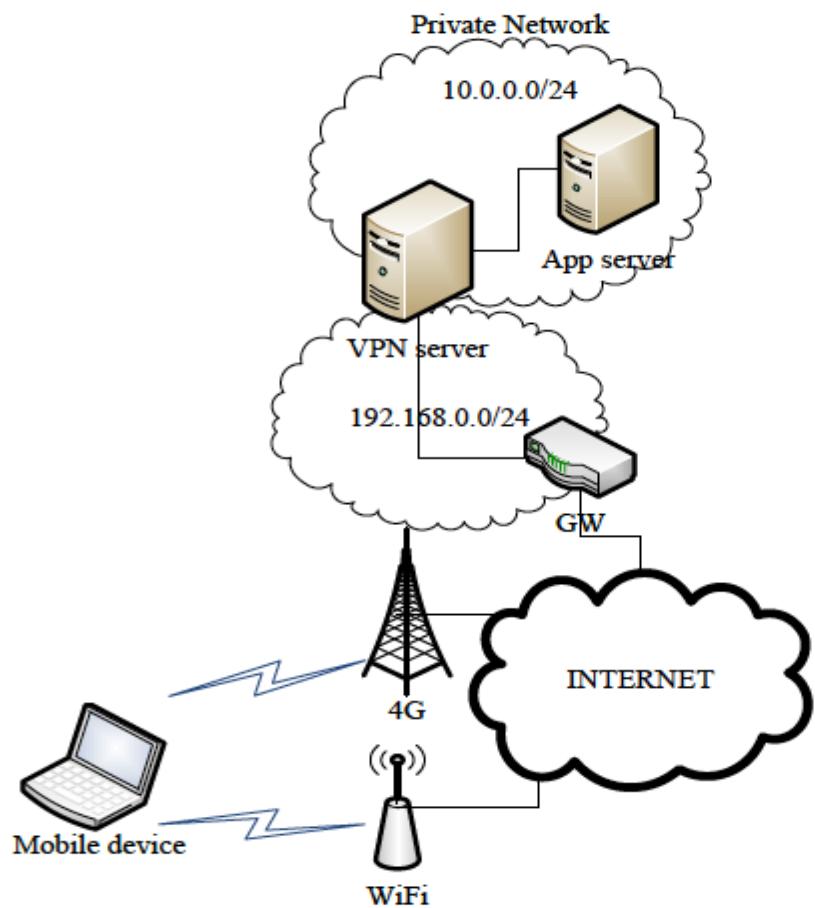
IP	VPN Instance
192.168.100.2	struct VPN_instance { .. }
192.168.100.10	struct VPN_instance { ... }
....

Session ID	IP
XYZ123	192.168.100.2
ABC456	192.168.100.10
....

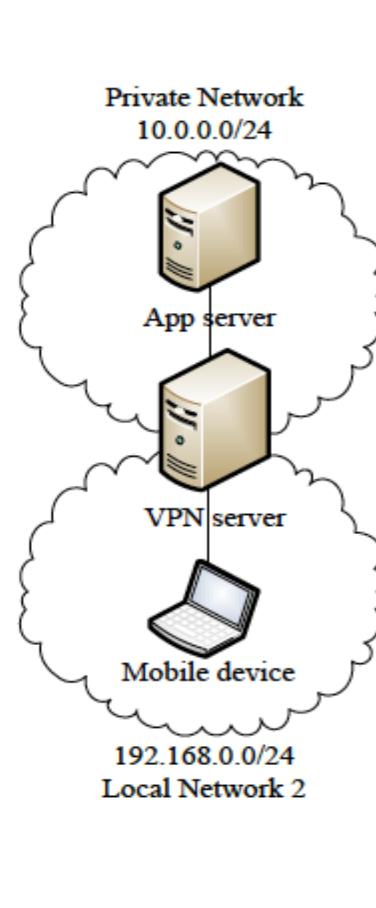
OpenVPN vs. MobiVPN



Evaluation - Testbed



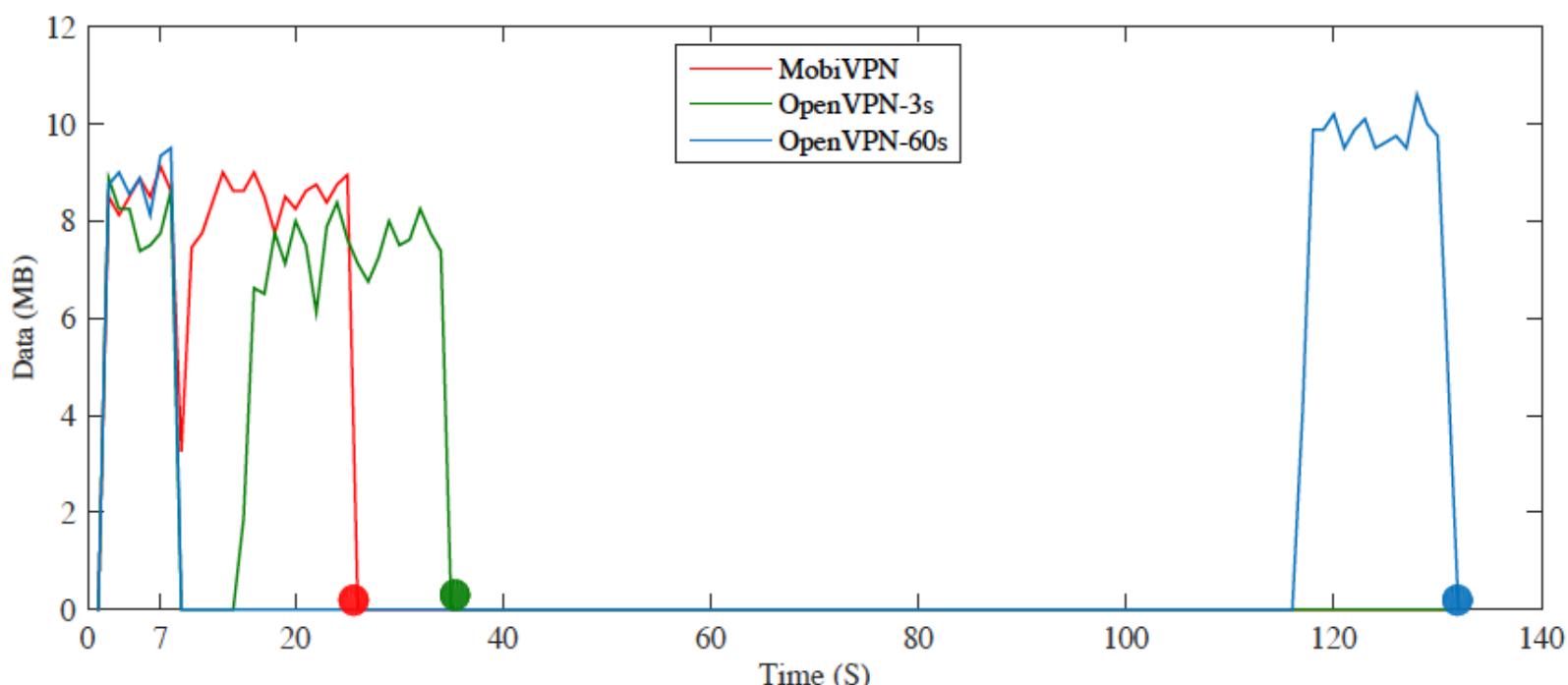
Distant Setup



Local Setup

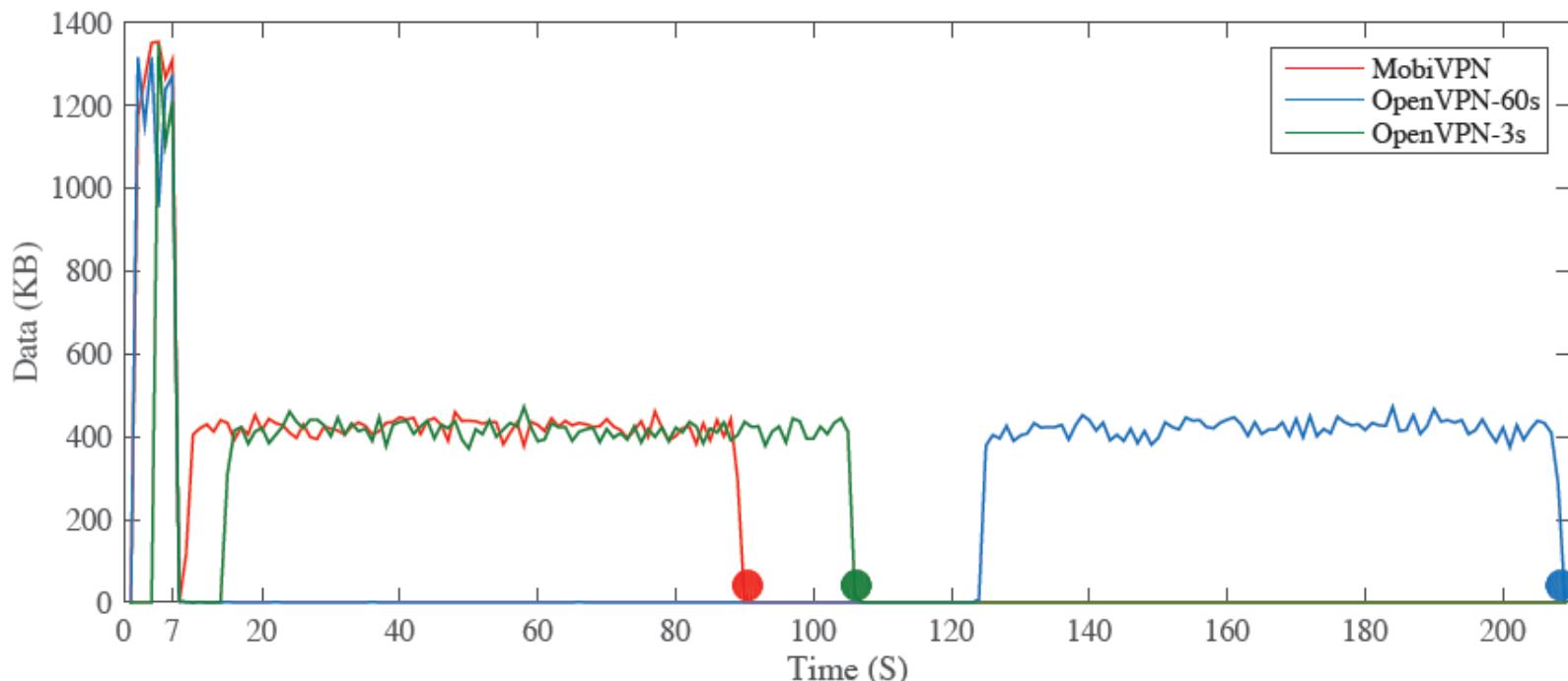
Effect of Fast VPN Resumption on Data Transfer - Local Testbed

- iperf is used to transmit 200MB of data between mobile device and application server through the VPN tunnel.
- IP of physical NIC was changed after 7 seconds.



Effect of Fast VPN Resumption on Data Transfer – Distant Testbed

- iperf is used to transmit 40MB of data between mobile device and application server through the VPN tunnel.
- After 7 seconds, the Wifi interface was turned off, and the 4G (USB tethering) interface was turned on.



Performance Measurements When The VPN Client Changes its IP Address

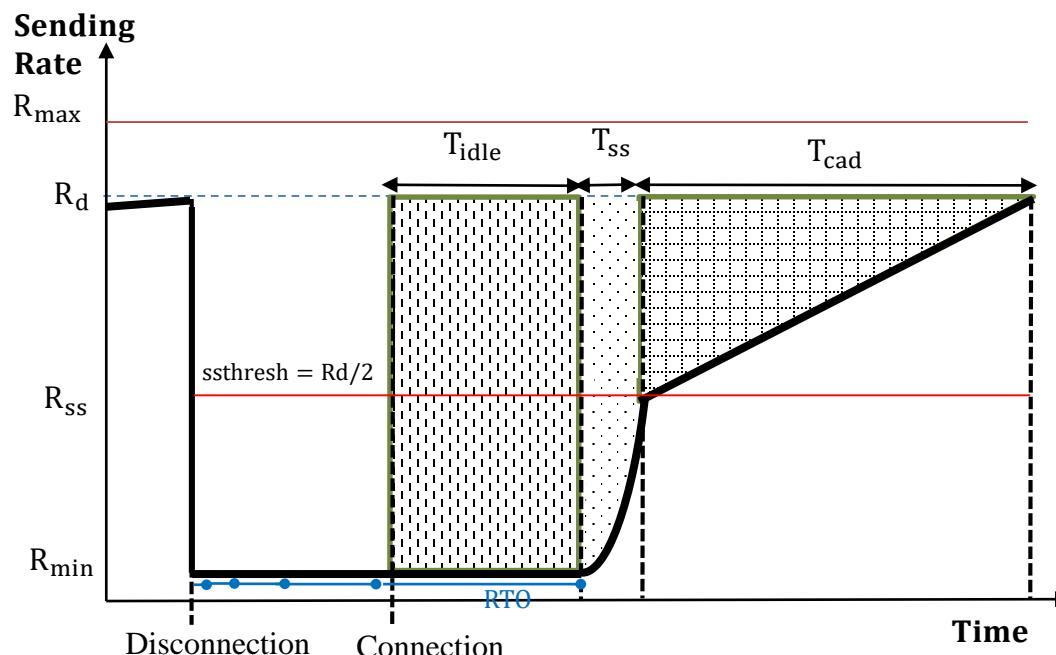
Testbed	Measured Metric	MobiVPN	OpenVPN -3s	% Decrease by MobiVPN	OpenVPN -60s	% Decrease by MobiVPN
Local	VPN Unavailability Time	214 ms	4.498 s	95.24%	61.711 s	99.65%
	VPN Session Resumption Time	8 ms	2.409 s	99.67%	2.412 s	99.67%
	Total Time To Resume VPN	222 ms	6.907 s	96.79%	64.123 s	99.65%
	Data Transfer Time (200MB)	24 s	33.1 s	27.49%	130.1 s	81.55%
Distant	VPN Unavailability Time	512 ms	4.710 s	89.13%	61.927 s	99.17%
	VPN Session Resumption Time	46 ms	3.462 s	98.67%	3.447 s	98.67%
	Total Time To Resume VPN	558 ms	8.172 s	93.17%	65.374 s	99.15%
	Data Transfer Time (40MB)	87 s	104 s	16.35%	207 s	57.97%

Persistence and Fast Resumption of TCP-based Applications

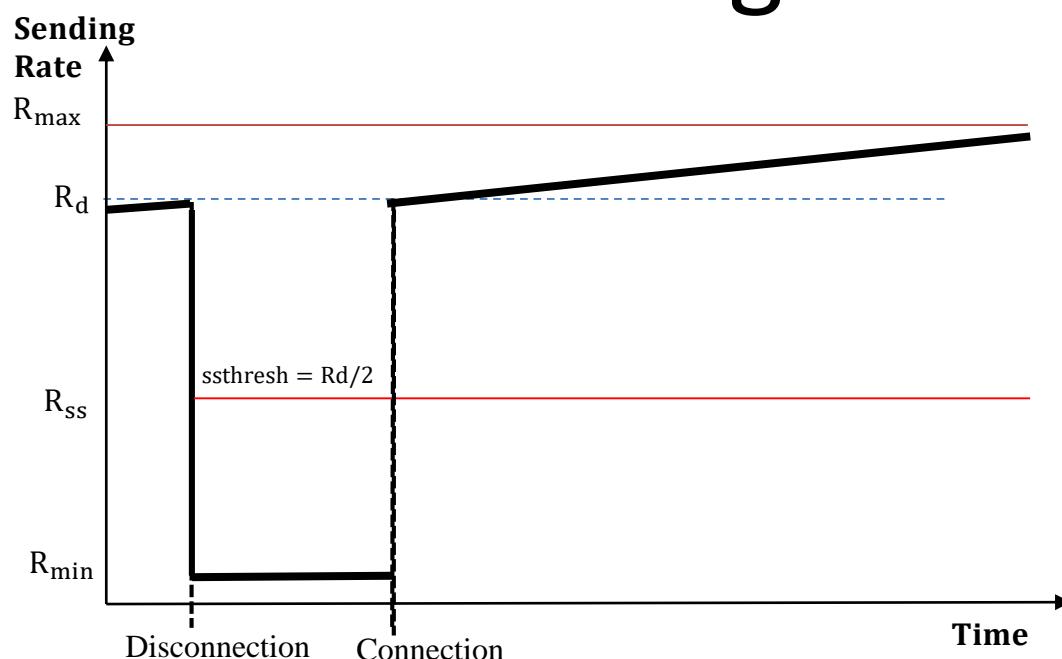
Problem Statement (Goal)

- Application sessions in the mobile node must be kept alive by hiding the VPN tunnel breakage and re-establishment from the applications.

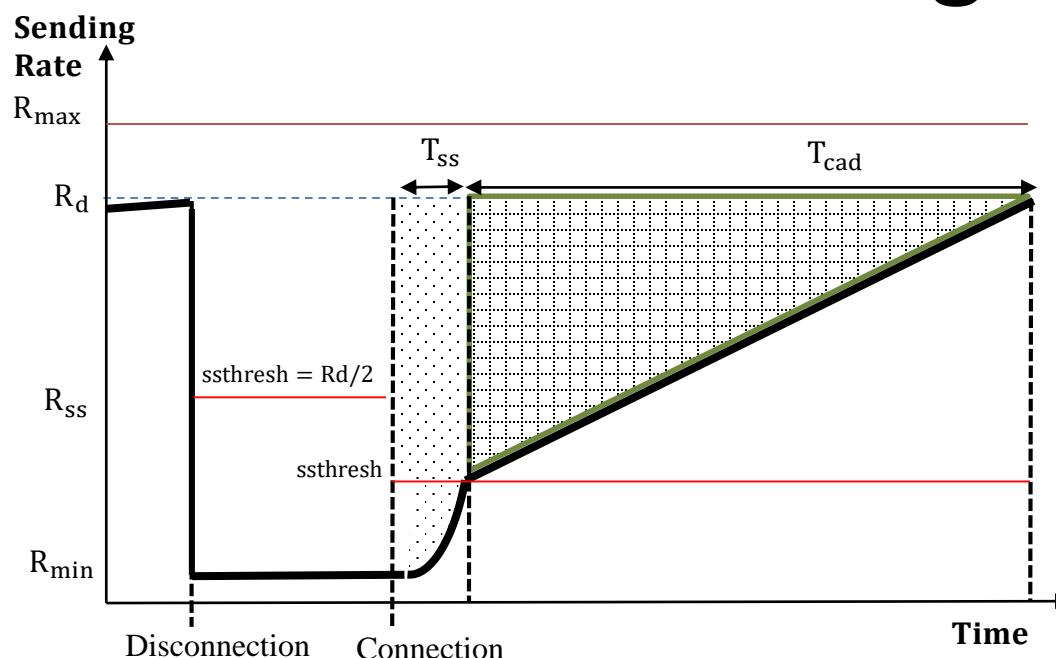
Effect of mobility on TCP Reno sending rate



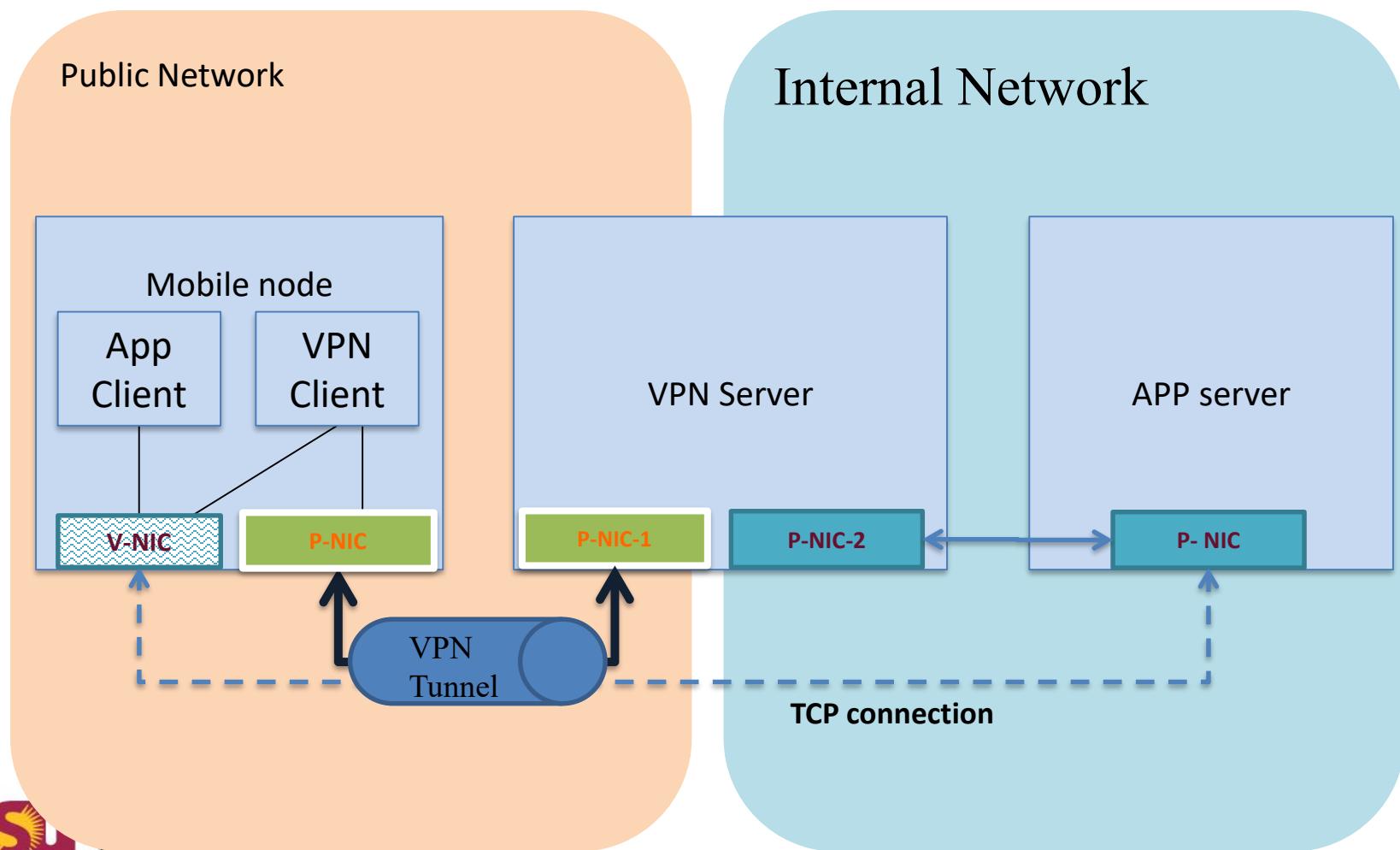
TCP sending rate – MobiVPN with buffering



TCP sending rate – MobiVPN without buffering



Our Approach



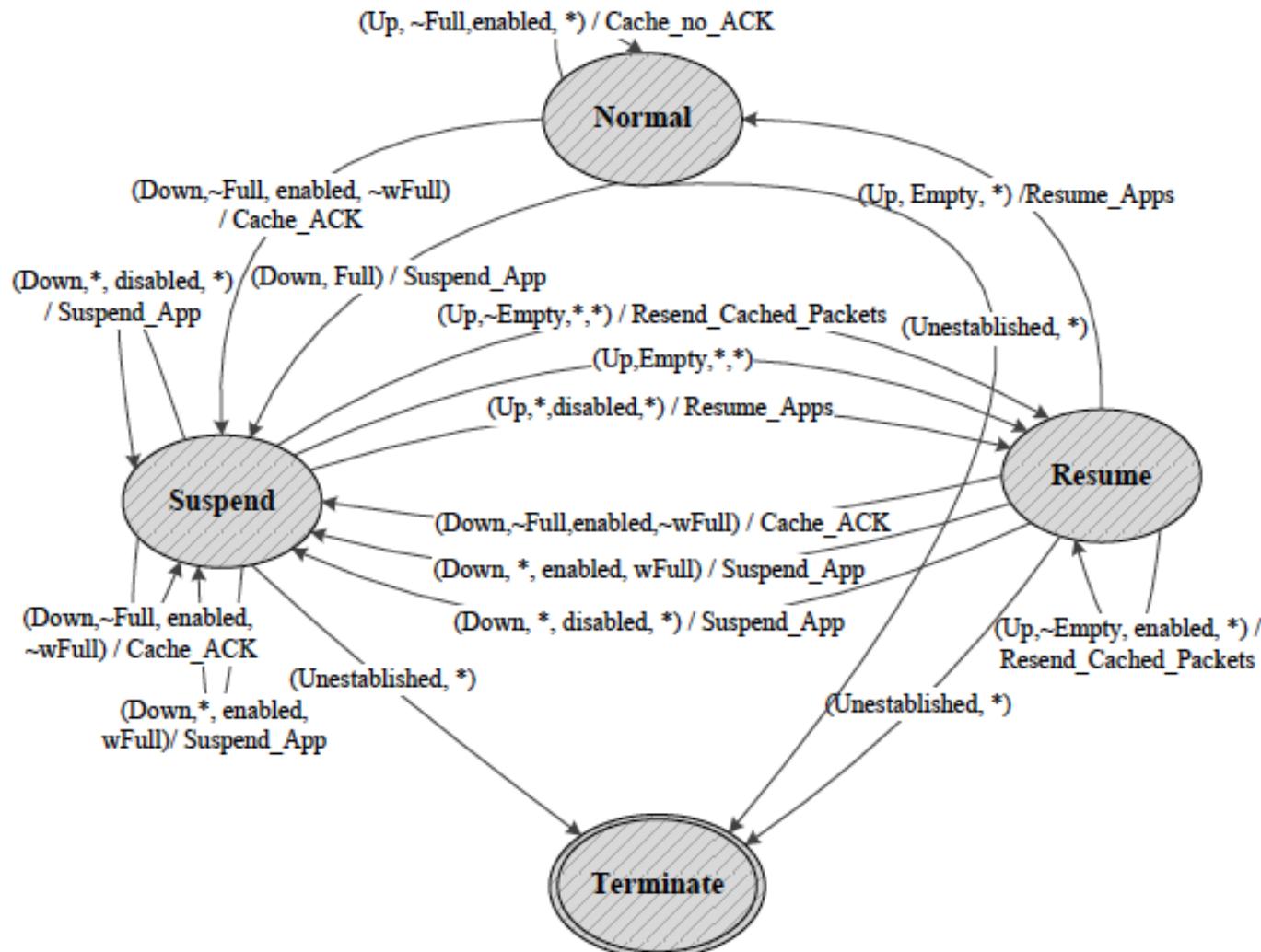
MobiVPN System Model

- *Modeled as a Finite State Transducer with these states:*
- **Normal state:** the VPN tunnel is healthy and the applications' TCP sessions behave as normal.
- **Suspend state:** when VPN tunnel fails due to network disconnection.
 - MobiVPN suspend non-buffered flows.
 - For buffered flows, MobiVPN buffers and acknowledges packets from the applications,
 - and eventually suspends the application sessions when the buffer is full by sending a Zero-Window packet,
 - This causes TCP to suspend its timers preventing it from terminating the TCP socket.
 - It also prevents TCP sockets from dropping the congestion window, avoiding slow-start upon reconnection.

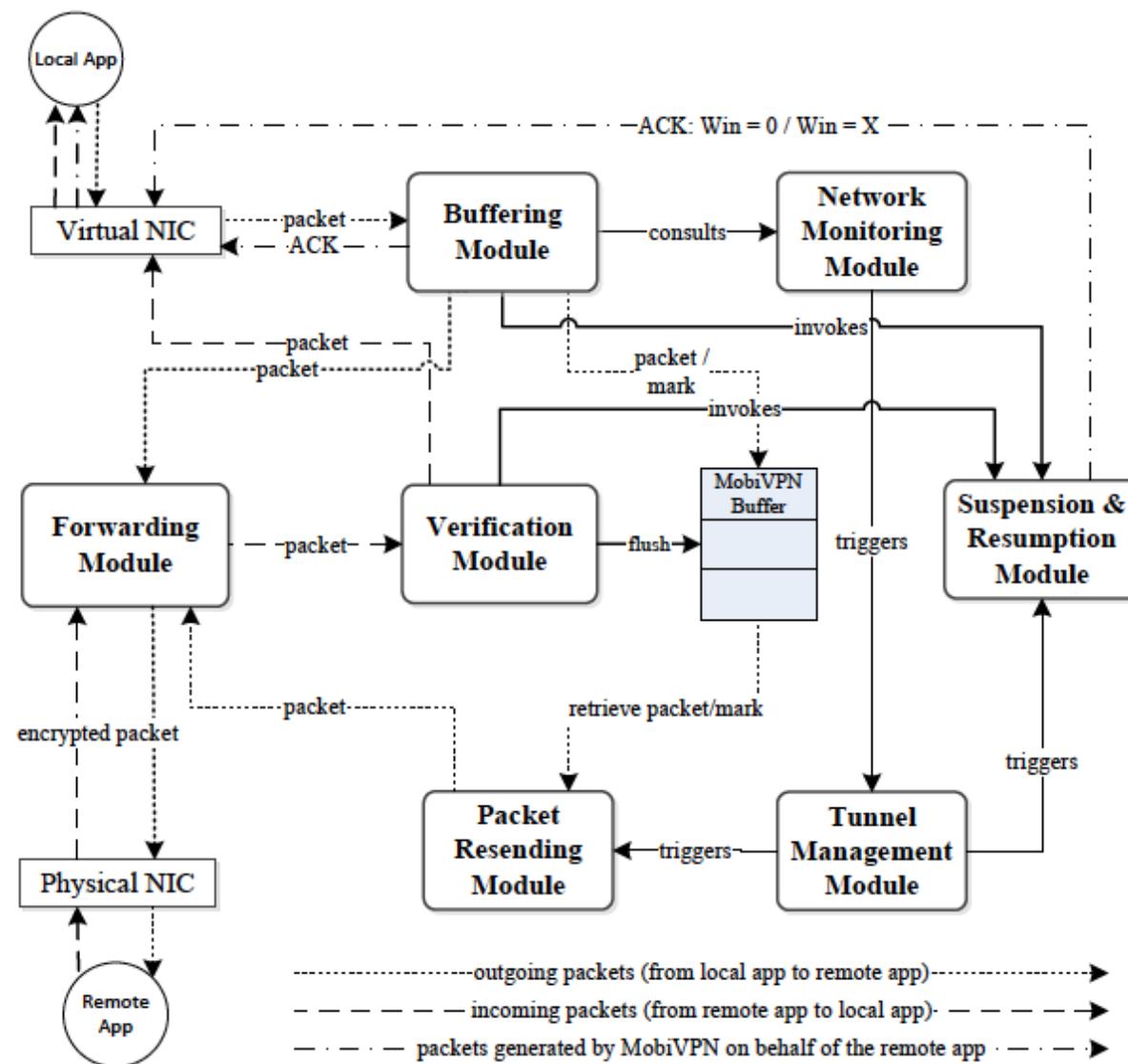
MobiVPN System Model

- ***Resume state:*** when the VPN tunnel is restored.
 - Non-buffered flows are resumed using Triple-ACK by the Suspension & Resumption (S&R) Module.
 - The packet resending module sends out buffered packets to the intended recipients until the buffer is cleared .
 - The verification module responsible for intercepting the packets received from the other end and flush the acknowledged packets from the buffer, while forwarding any incoming data to the local app.
 - The S&R module is invoked to resume the flows whose all buffered packets are acknowledged.

State Transition in MobiVPN



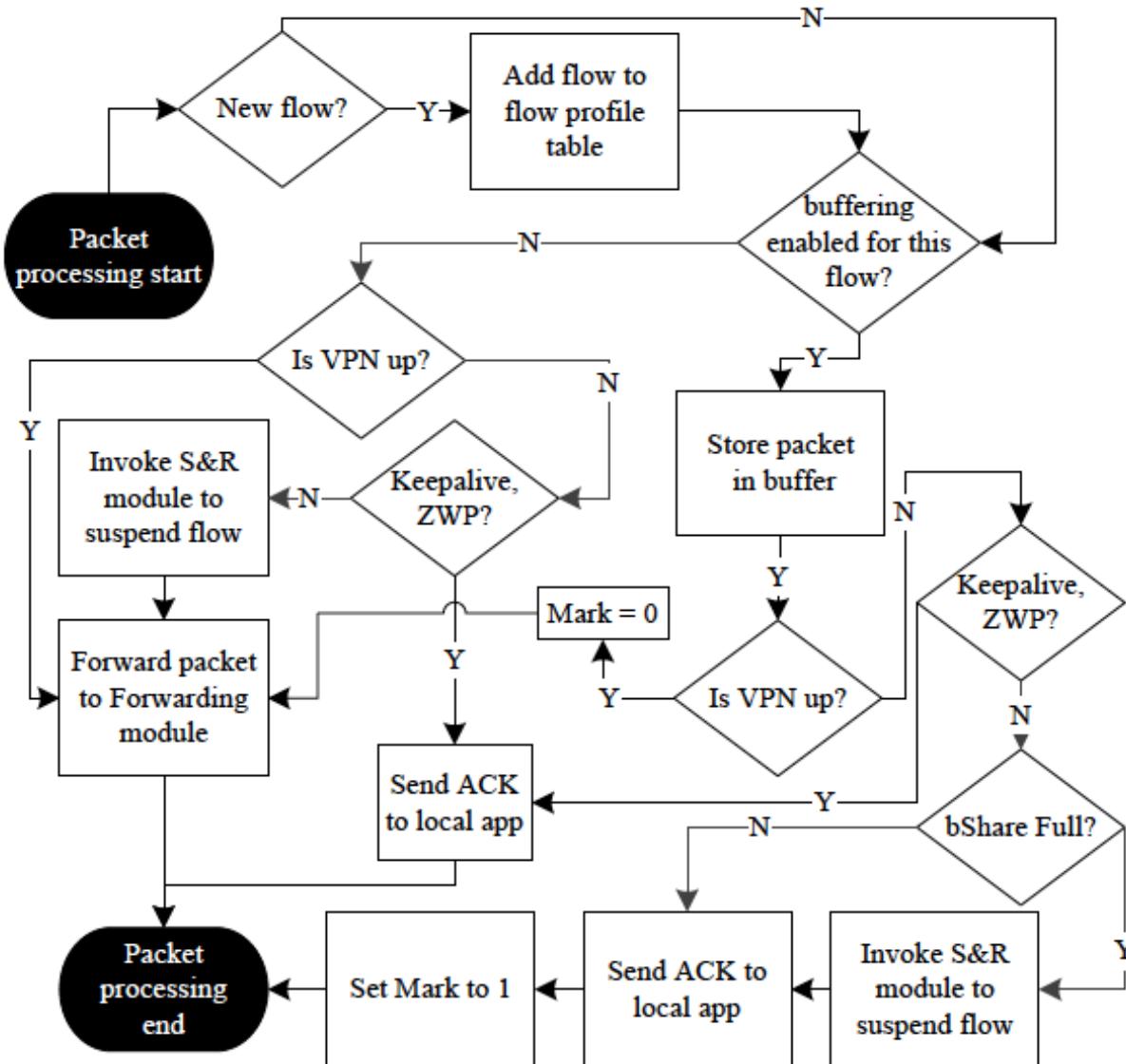
Modules Relations in MobiVPN



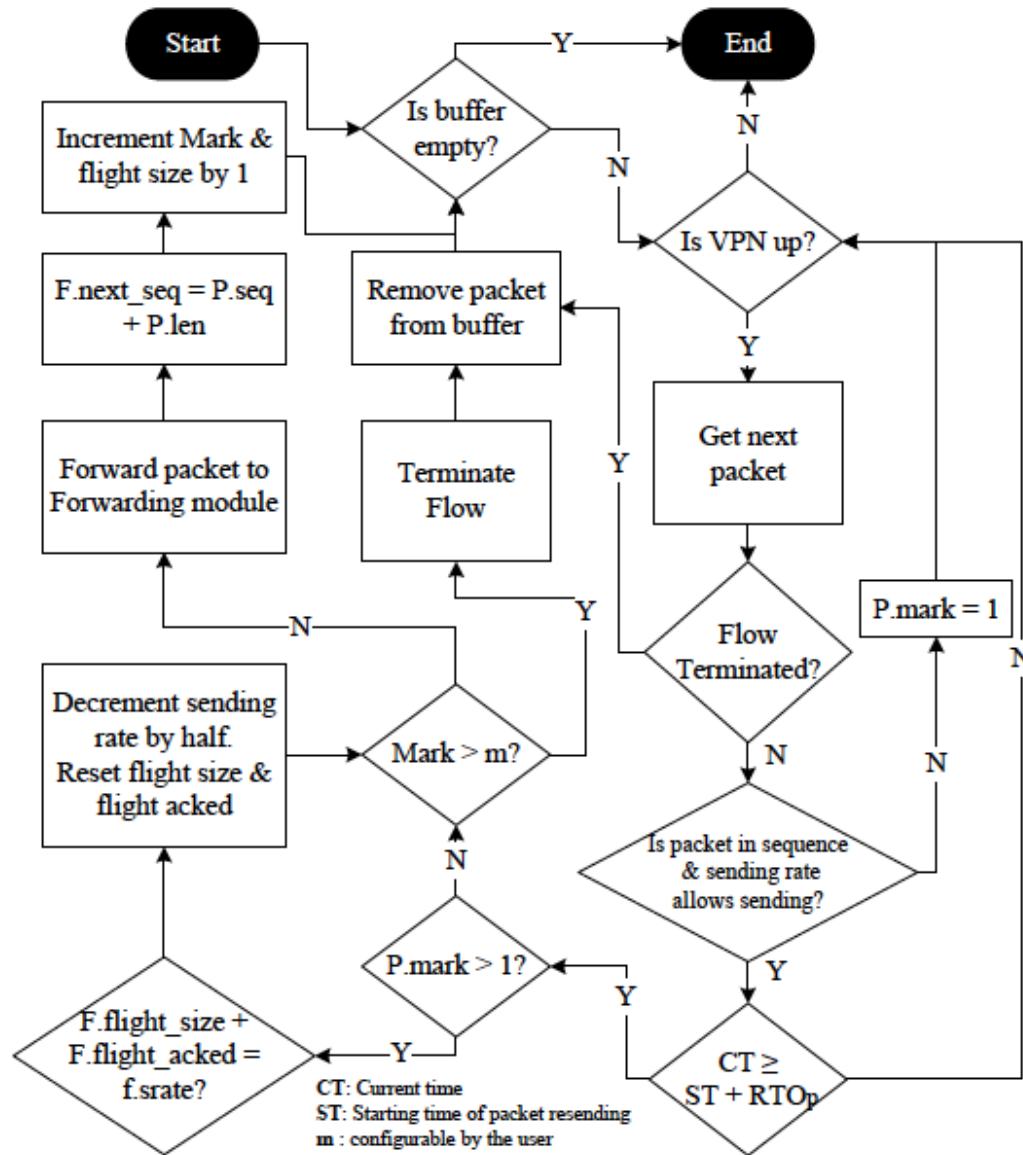
MobiVPN also does:

- **Suspend Phase:**
 - Respond to keep alive messages.
 - Respond to window probe messages.
- **Resume Phase:**
 - Make sure the ACKs are synchronized.

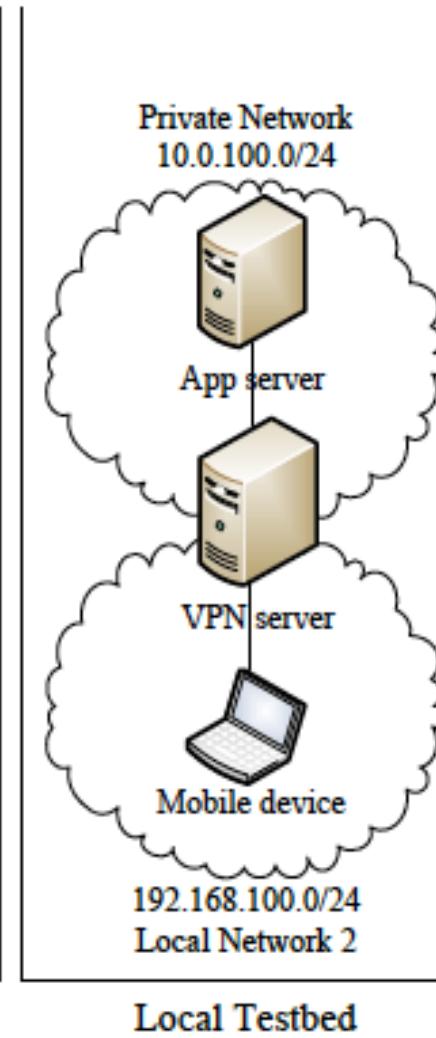
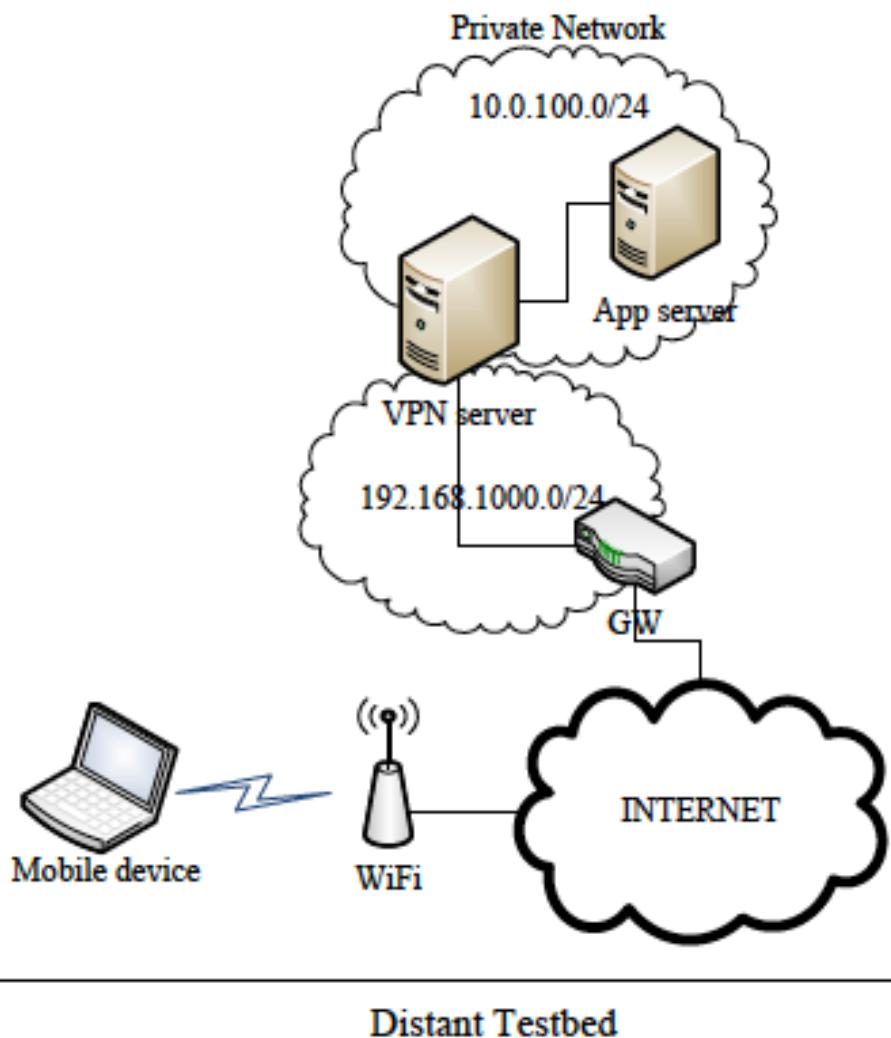
Buffering Module



Packet Resending Module



Evaluation- Testbed



Persistency Evaluation

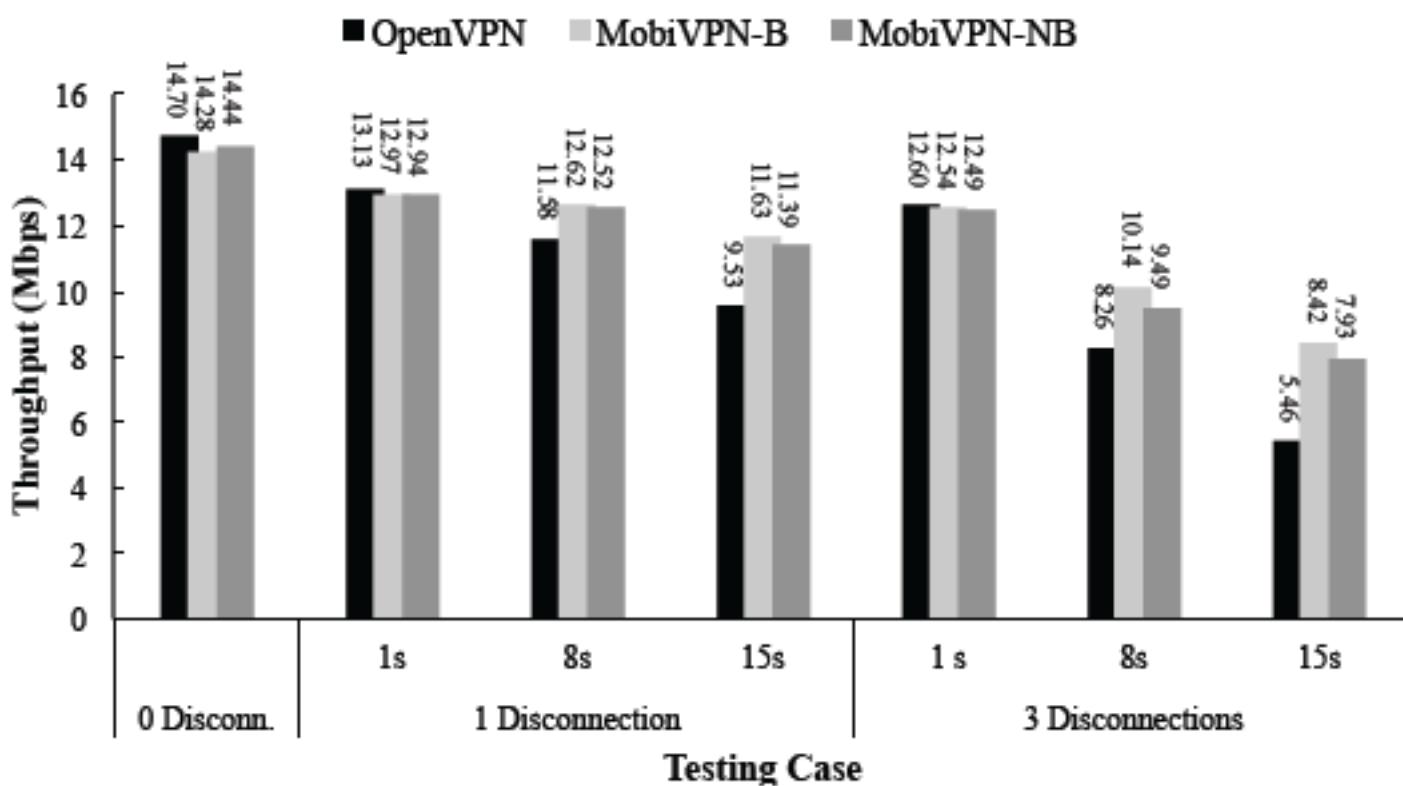
- Transmitted a 1GB text file using a simple file transfer program that uses a TCP socket:
`socket(AF_INET,SOCK_STREAM,0)`
- Disconnected the network interface for X seconds after a few seconds from the beginning of file transfer.

Transfer direction	Server's TCP settings	Disconnection length	Transfer Completed?	
			OpenVPN	MobiVPN
S → C	<code>tcp_retries2 = 6</code>	10 seconds	✓	✓
		45 seconds	✗	✓
C → S	<code>tcp_keepalive_time=10s</code>	10 seconds	✓	✓
	<code>tcp_keepalive_intvl=5s</code>	45 seconds	✗	✓
	<code>tcp_keepalive_probes=3</code>			

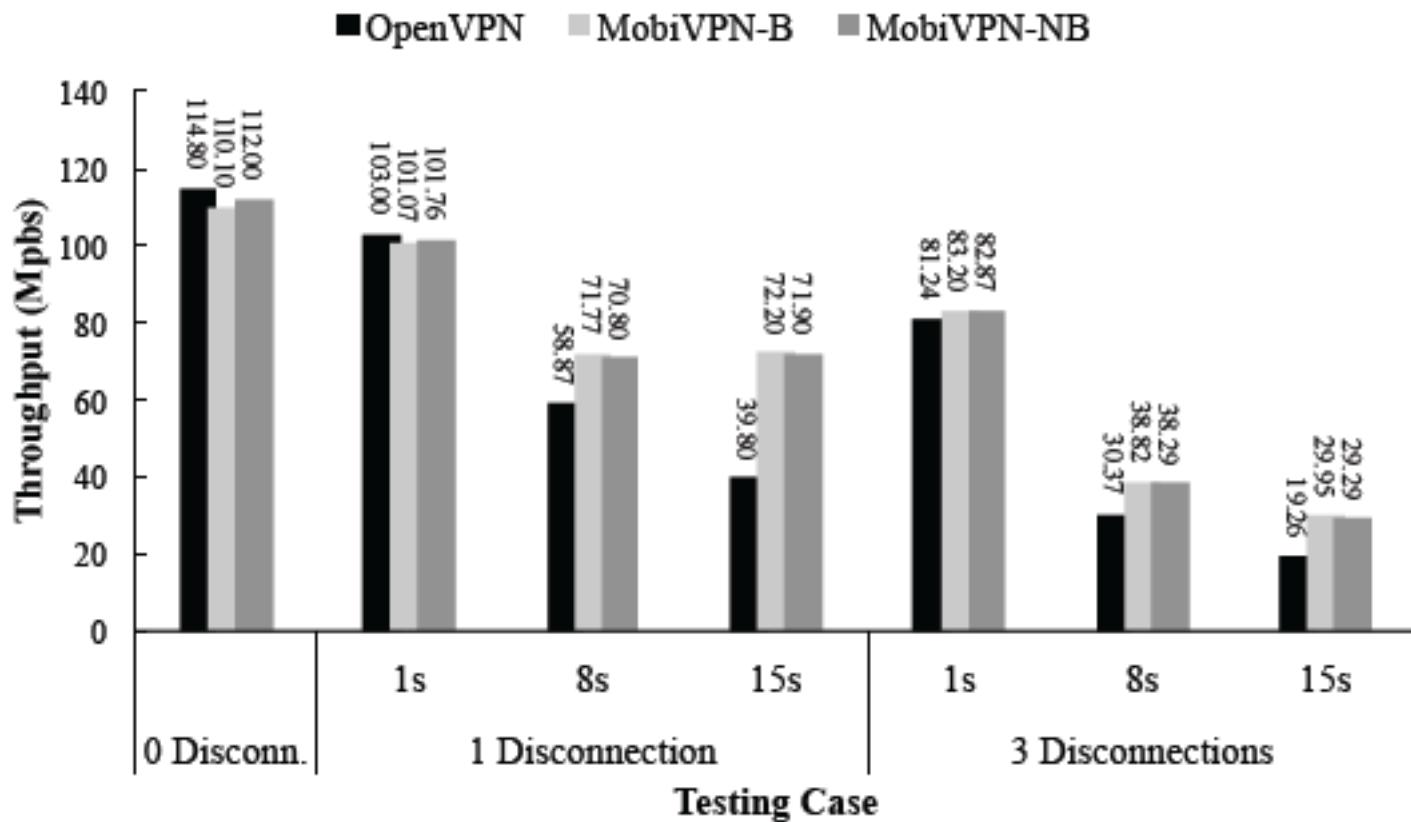
Performance Evaluation

- We used *iperf* to measure the average throughput of OpenVPN vs. MobiVPN by streaming 200 MB (in local testbed) and 100MB (in distant testbed) of data from the client to the application server using several scenarios varying the number of disconnections and their length.
- No network switching was performed in order to eliminate the factor of the VPN tunnel resumption which we discussed earlier.

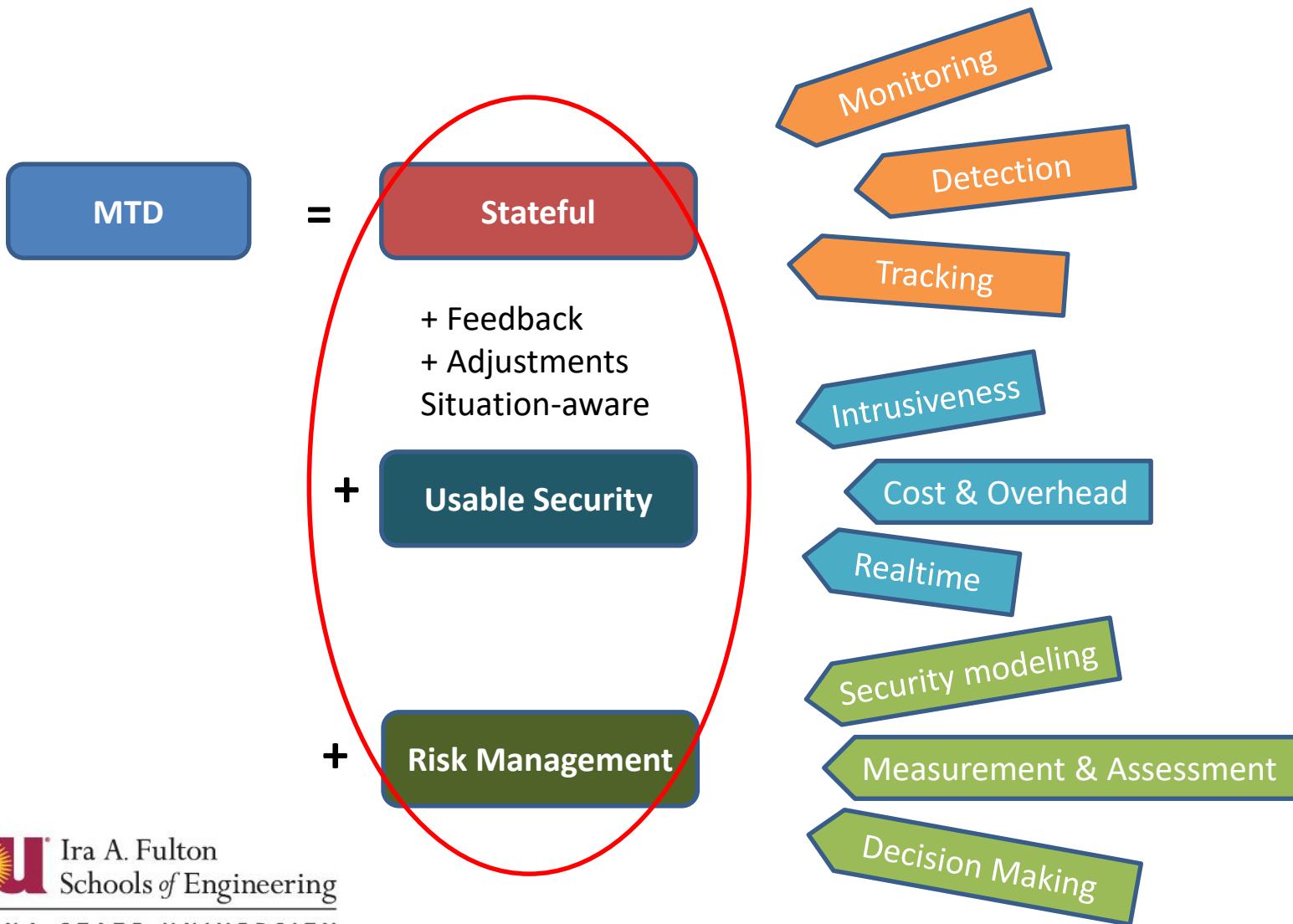
Results – Distant testbed



Results – Local testbed

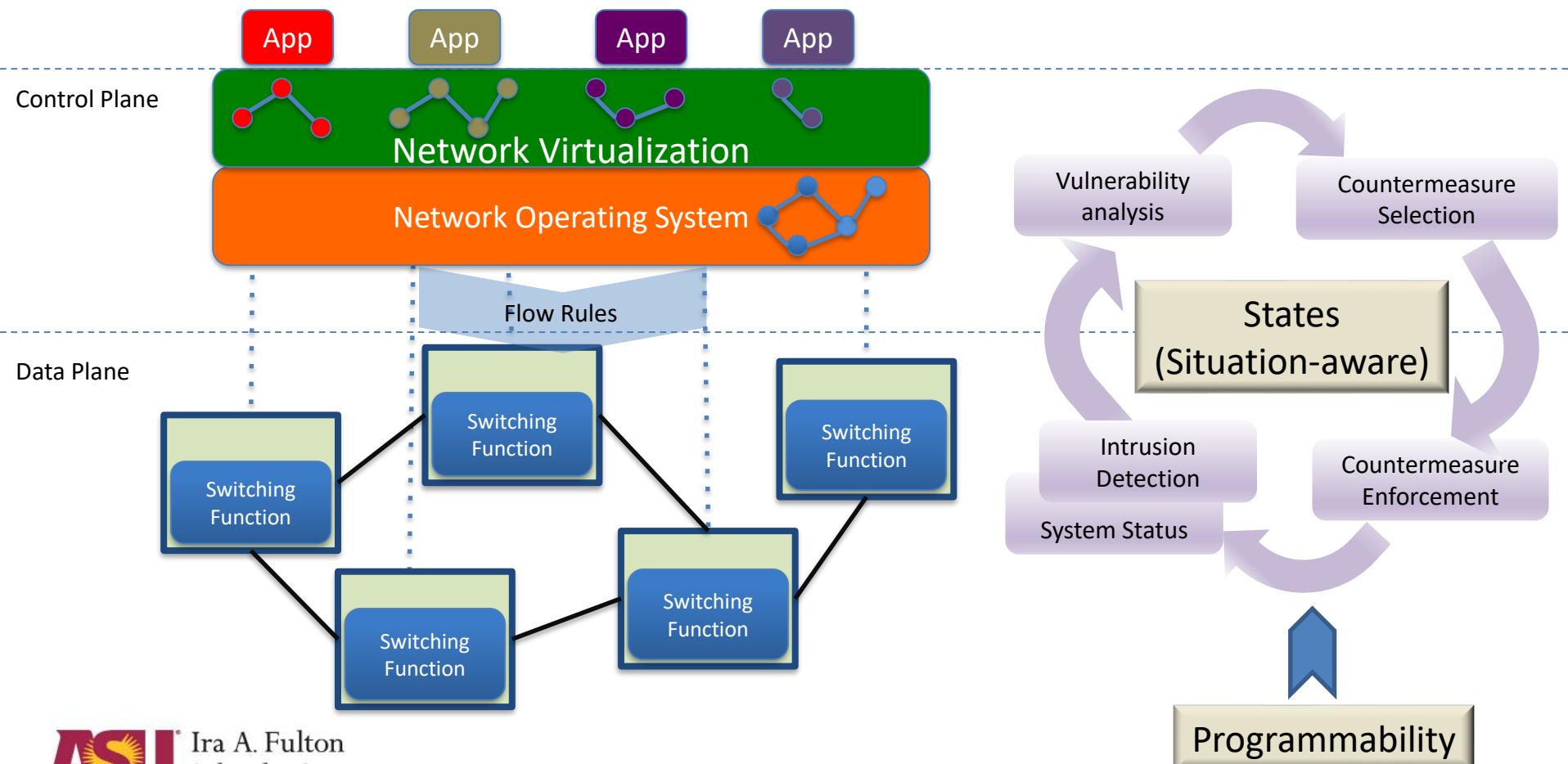


Summary of MTD



Why SDN?

Abstract SDN Model and MTD



Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

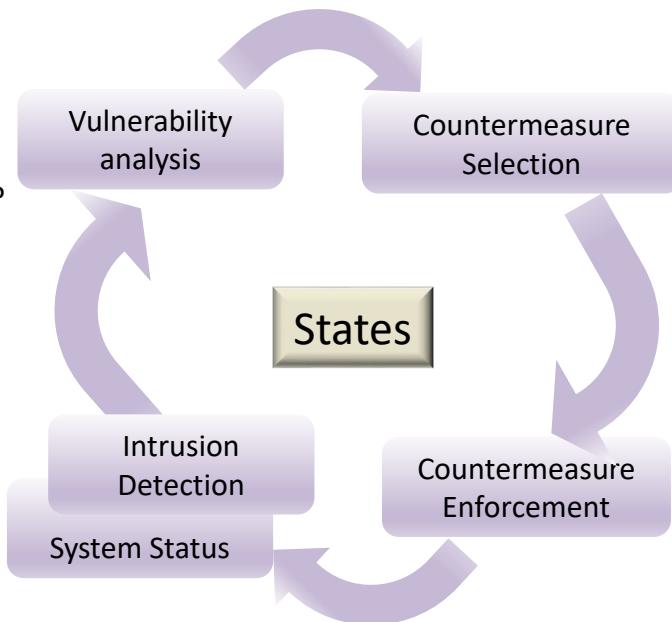
Challenges

Scalable security analysis Tools

- Healthiness analysis
- Attack projection
- Is attack graph/tree a good candidate?

What information needs to be collected?

- MAC, IP, Transport, App?
- APT, Zero-day attacks



Comprehensive considerations

- Quarantine/isolation, mitigation/flow rules, reconfiguration
- Cost of countermeasure deployment
- Intrusiveness to good users
- QoS considerations
- Deployment frequency

Will SWs take over the role of FW?

- Distributed FW.
- How to maintain states?
- Will this violate the principle of the isolation of control and data planes?

Agenda

1. Moving Target Defense (MTD)
2. **SDN-Based MTD Approaches**
 - SDN Basis
 - A Case Study: An Intelligent SDN-based MTD Approach
 - Research challenges, opportunities, and future directions
3. Q&A and Discussion

SDN Basis

- **Concept of SDN**
- OpenFlow – a SDN Implementation
- Open vSwitch

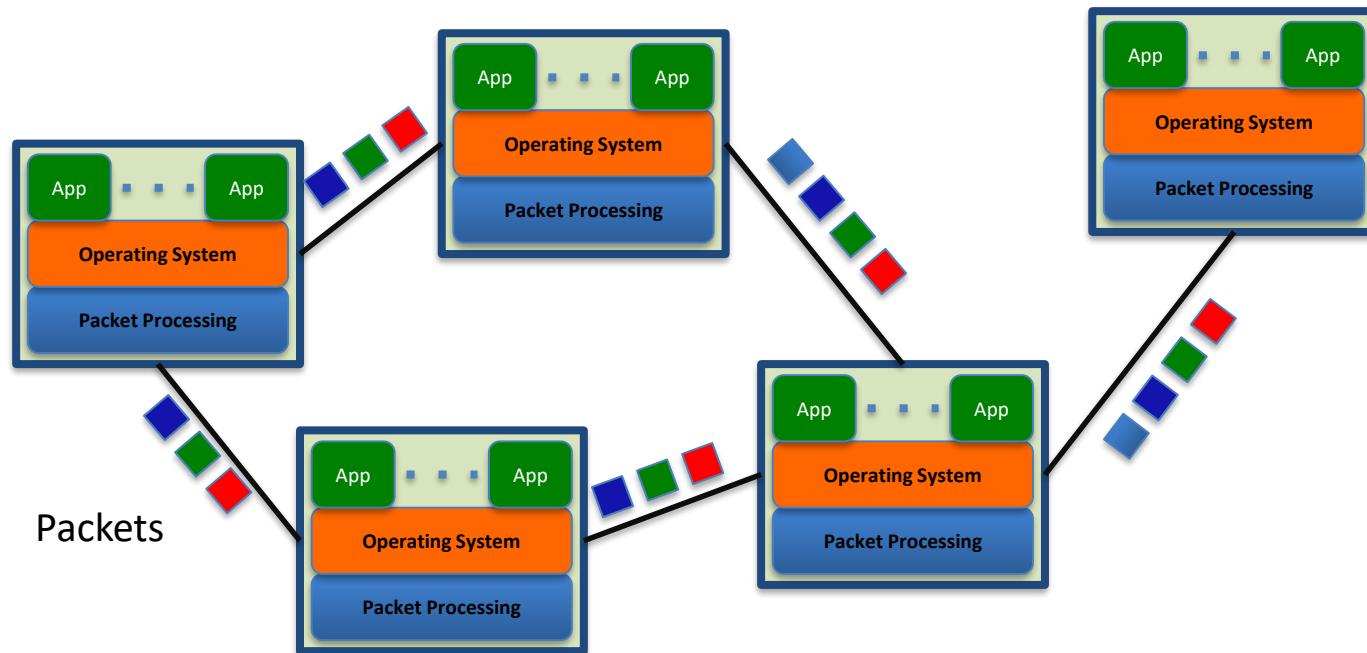


Why do we need SDN?

- Do you think a Network system simple?
- Networks used to be simple: Ethernet, IP, TCP....
- New **control** requirements led to great complexity
 - Isolation → VLANs, ACLs
 - Traffic engineering → MPLS, ECMP,Weights
 - Packet processing → Firewalls, NATs, middleboxes
 - Payload analysis → Deep packet inspection (DPI)
 -
- Mechanisms designed and deployed independently
 - Complicated “control plane” design, primitive functionality

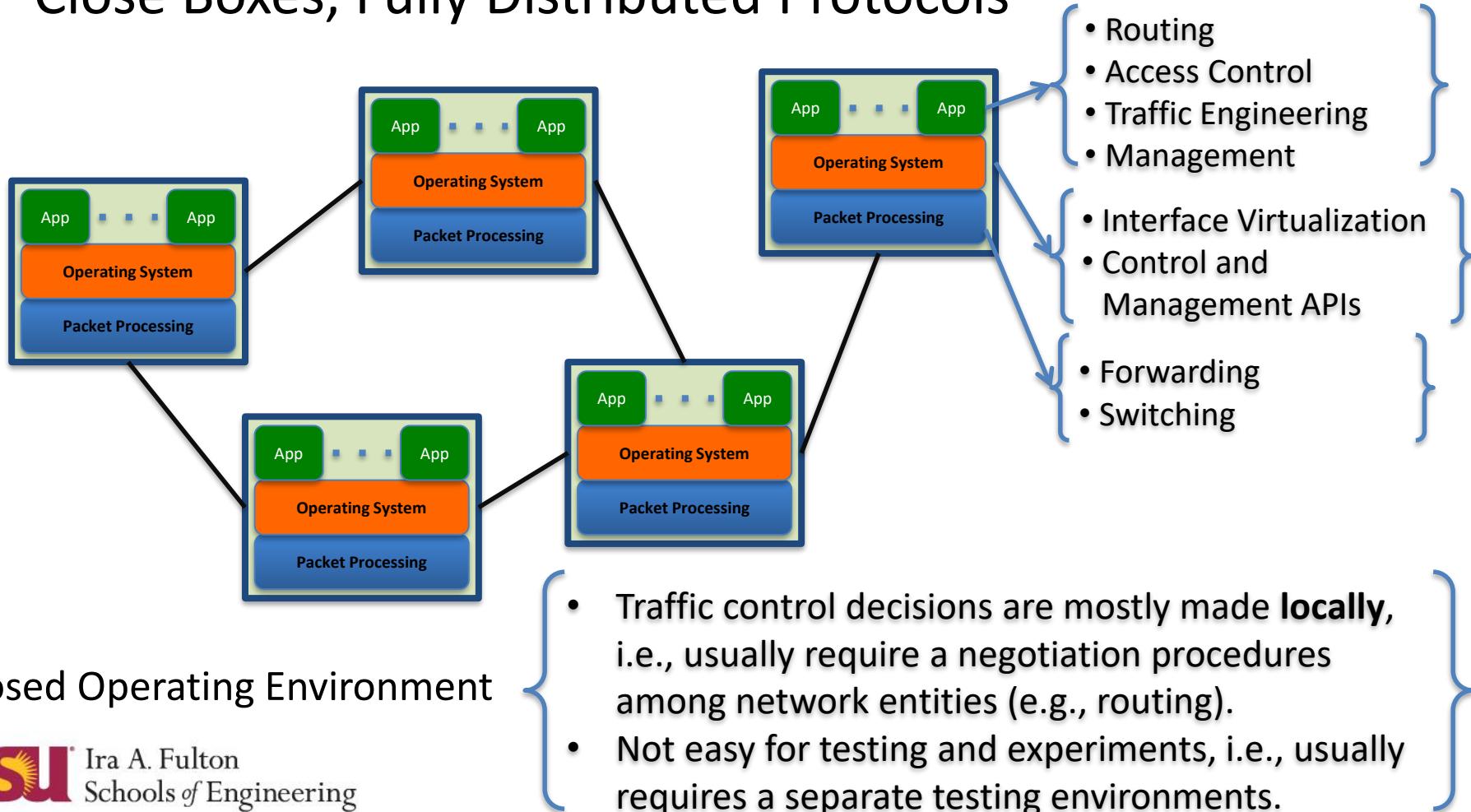
Data processing and Control in Distributed Systems

Data processing and controls are **not separated**. Algorithms, e.g., routing, is performed in a distributed environment. Each router acts **individually**, much like our current social networks.



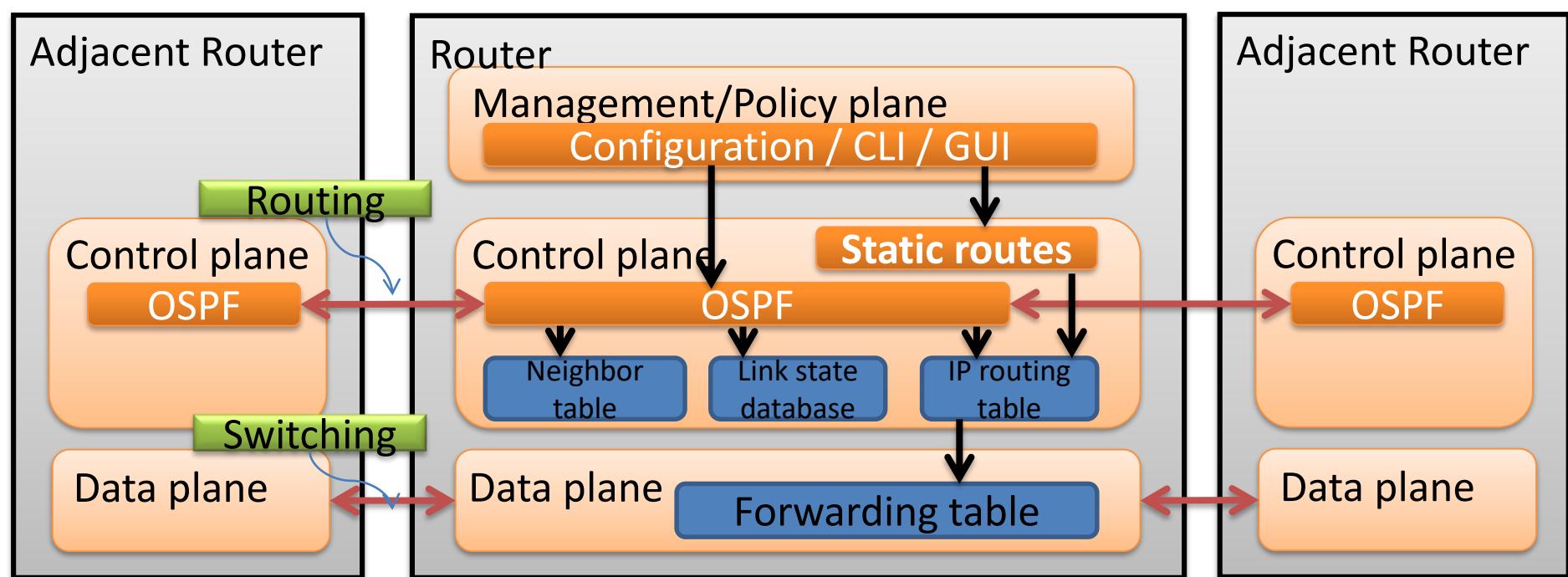
Distributed Networking Systems

- Close Boxes, Fully Distributed Protocols



Traditional network node: Router

- Router can be partitioned into control and data plane
 - Management plane/ configuration
 - Control plane / Decision: OSPF (Open Shortest Path First)
 - Data plane / Forwarding



Two “planes”

- Two fundamental terms to begin understanding the SDN

Processing Plane	Where it runs	How fast these processes run	Type of processes performed
Control Plane	Switch CPU (smart but slow)	In the order of thousands of packets per second	Routing protocols (i.e. OSPF, IS-IS, BGP), Spanning Tree, SYSLOG, AAA (Authentication Authorization Accounting), NDE (Netflow Data Export), CLI (Command Line interface), SNMP
Data Plane	Dedicated Hardware ASIC (fast but dumb)	Millions or Billions of packets per second	Layer 2 switching, Layer 3 (IPv4 IPv6) switching, MPLS forwarding, VRF Forwarding, QOS (Quality of Service) Marking, Classification, Policing, Netflow flow collection, Security Access Control Lists

SDN definitions

- SDN is an approach to building computer networks that separates and abstracts elements of these systems –
from Wikipedia
 - In SDN, not all processing happens inside the same device.
- In the Software Defined Networking architecture, the control and data planes are **decoupled**, network intelligence and state are **logically centralized**, and the underlying network infrastructure is **abstracted** from the applications.
– from ONF White Paper

How to do it?

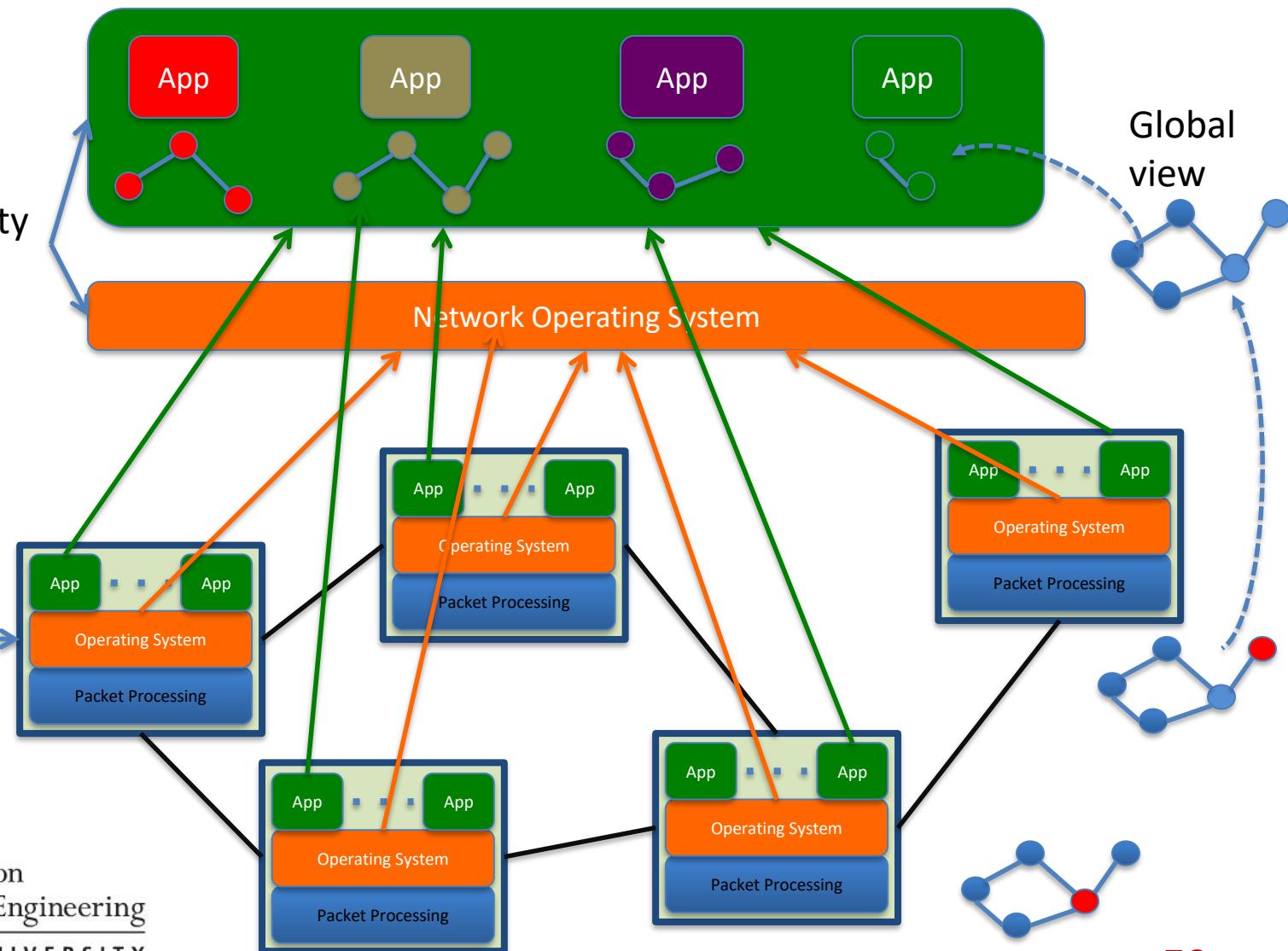
- How to get a simpler, more systematic design for the so complicate network control mechanisms?
- The power of Abstraction
 - “Modularity based on abstraction is the way things get done.” – Barbara Liskov
- **Abstractions → Interfaces → Modularity**

Abstraction Details

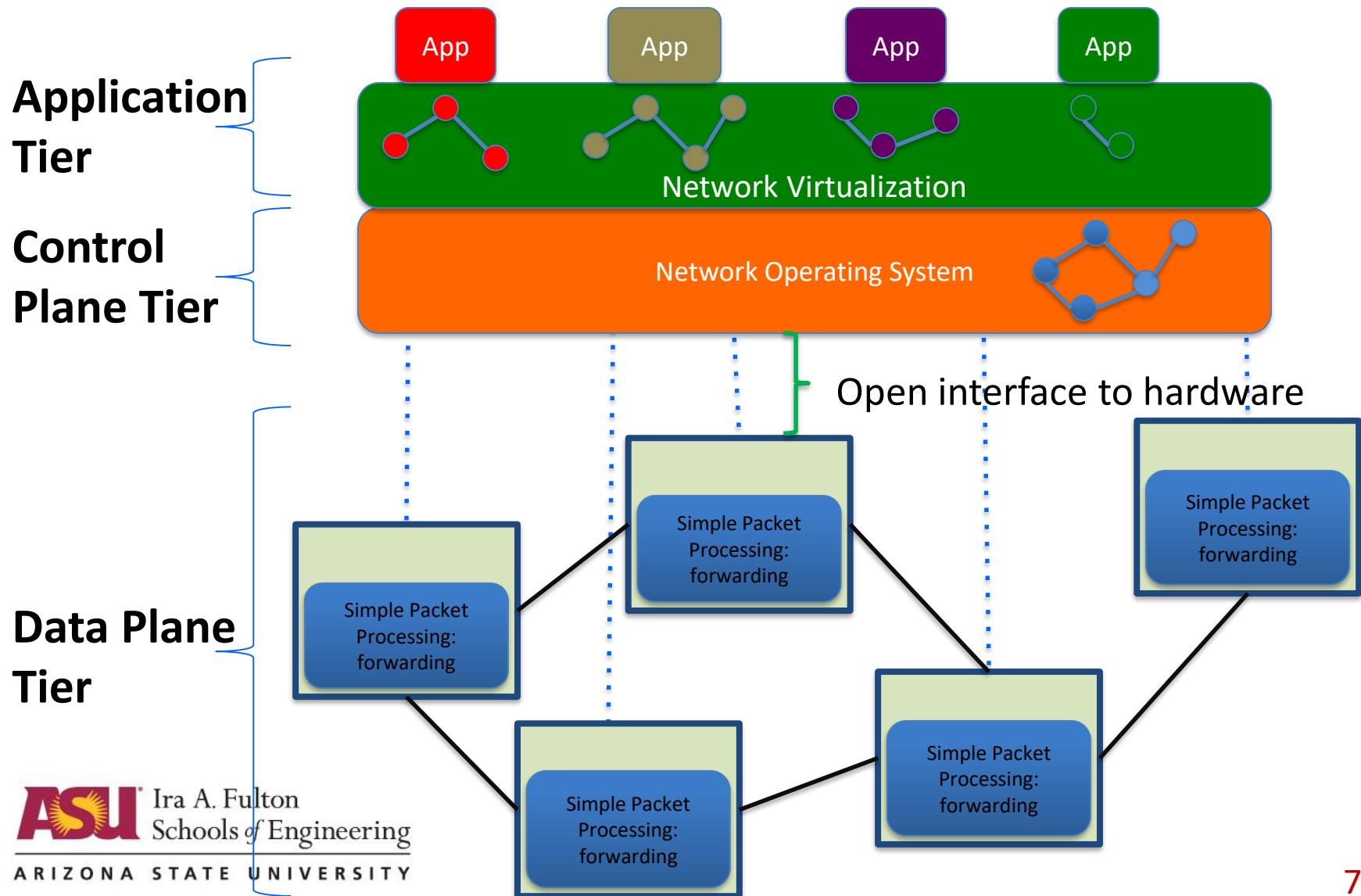
- Abs#1: ***Forwarding abstraction***
 - OpenFlow is current proposal for forwarding standard
 - Configuration in terms of **flow entries**: <header, action>
- Abs#2: ***Network state abstraction***
 - **Global network view** abstraction
 - **Network OS** (controllers) queries network devices to form “view” and sends commands to them to control forwarding
- Abs#3: ***Specification abstraction***
 - Abstract view of network: simple model with only enough to specify goals.
 - **Network virtualization**: map abstract configuration to physical configuration

From traditional networking to SDN

- Programmability
- Virtualization
- Networking Resource Isolation
- Fine-grained access control



Abstract SDN Model



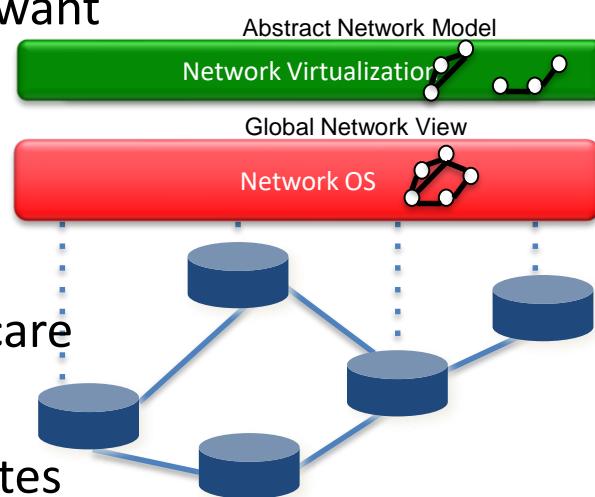


Clean Separation of Concerns

- **Control program:** express goals on abstract view
 - Driven by **Operator Requirements**
- **Virtualization Layer:** abstract view \leftrightarrow global view
 - Driven by **Specification Abstraction** for particular task
- **NOS:** global view \leftrightarrow physical switches
 - API: driven by **Network State Abstraction**
 - Switch interface: driven by **Forwarding Abstraction**

How to process the SDN requests?

- Write a simple program to configure a simple model
 - Configuration merely a way to specify what you want
- Examples
 - ACLs: who can talk to who
 - Isolation: who can hear my broadcasts
 - Routing: only specify routing to the degree you care
 - Some flows over satellite, others over landline
 - TE: specify in terms of quality of service, not routes
- Virtualization layer “compiles” these requirements
 - Produces suitable configuration of actual network devices
- NOS then transmits these settings to physical boxes



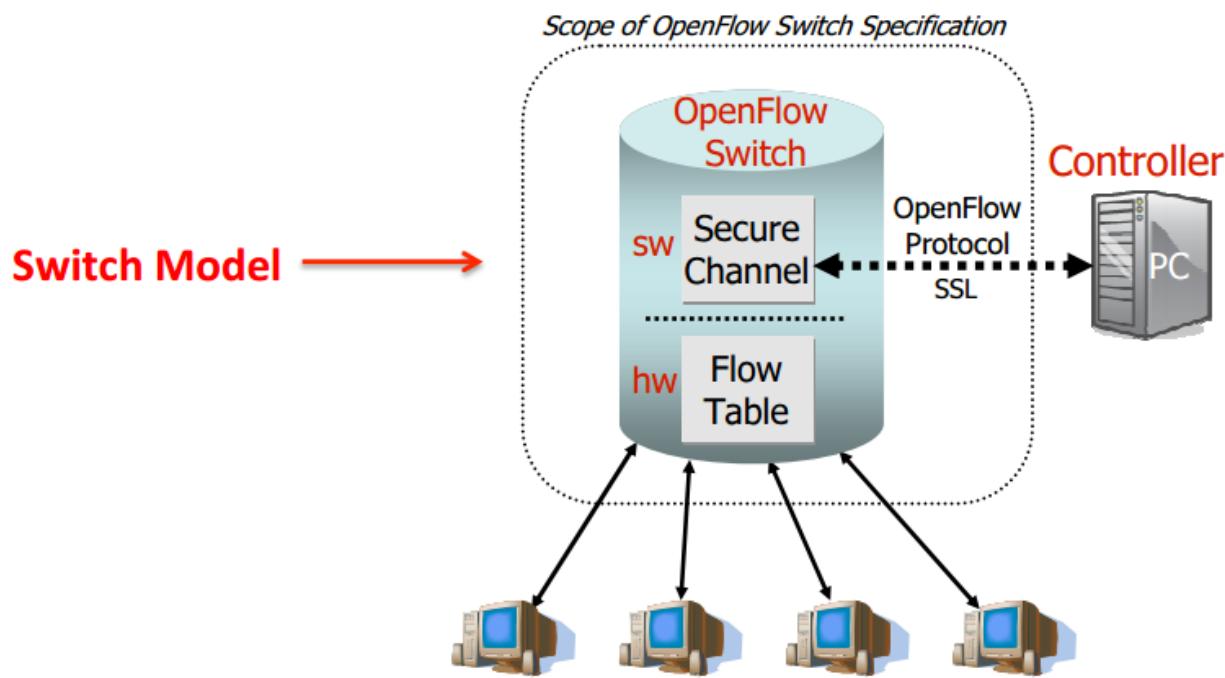
SDN Basis

- Concept of SDN
- **OpenFlow – a SDN Implementation**
- Open vSwitch



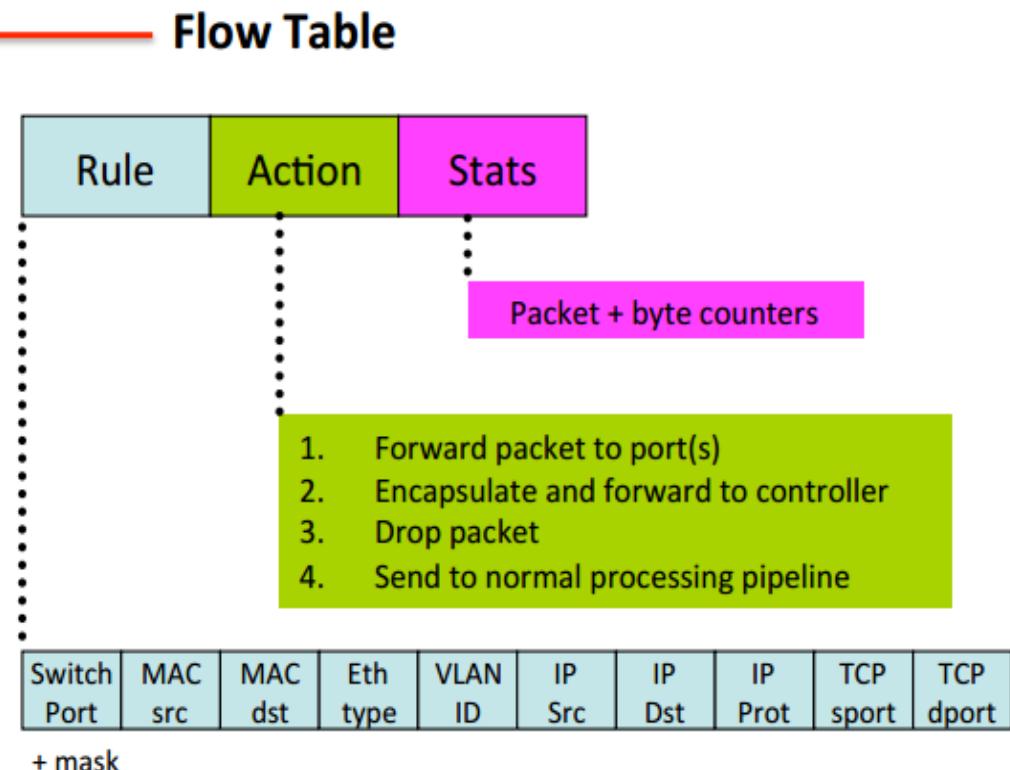
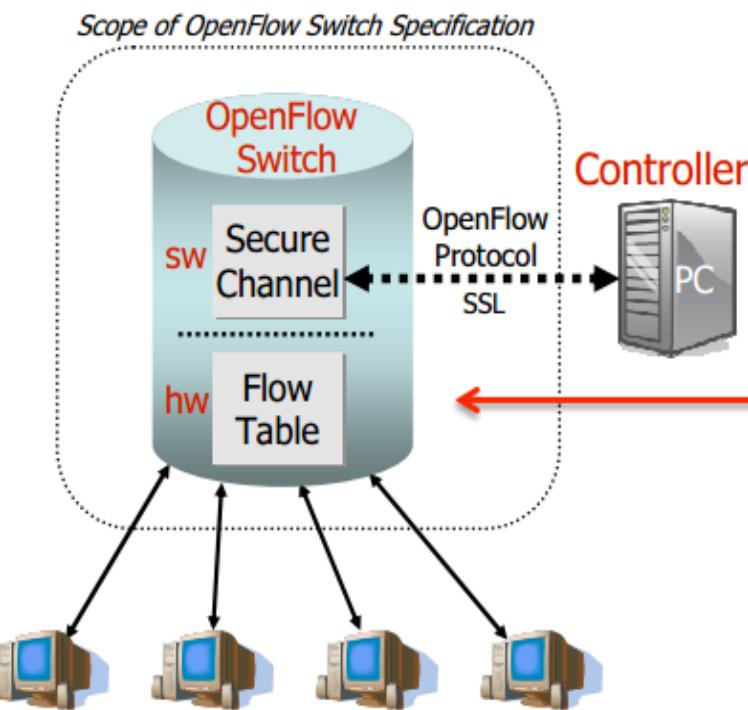
OpenFlow

- OpenFlow is a Layer 2 communications protocol that gives access to the forwarding plane of a network switch or router over the network.

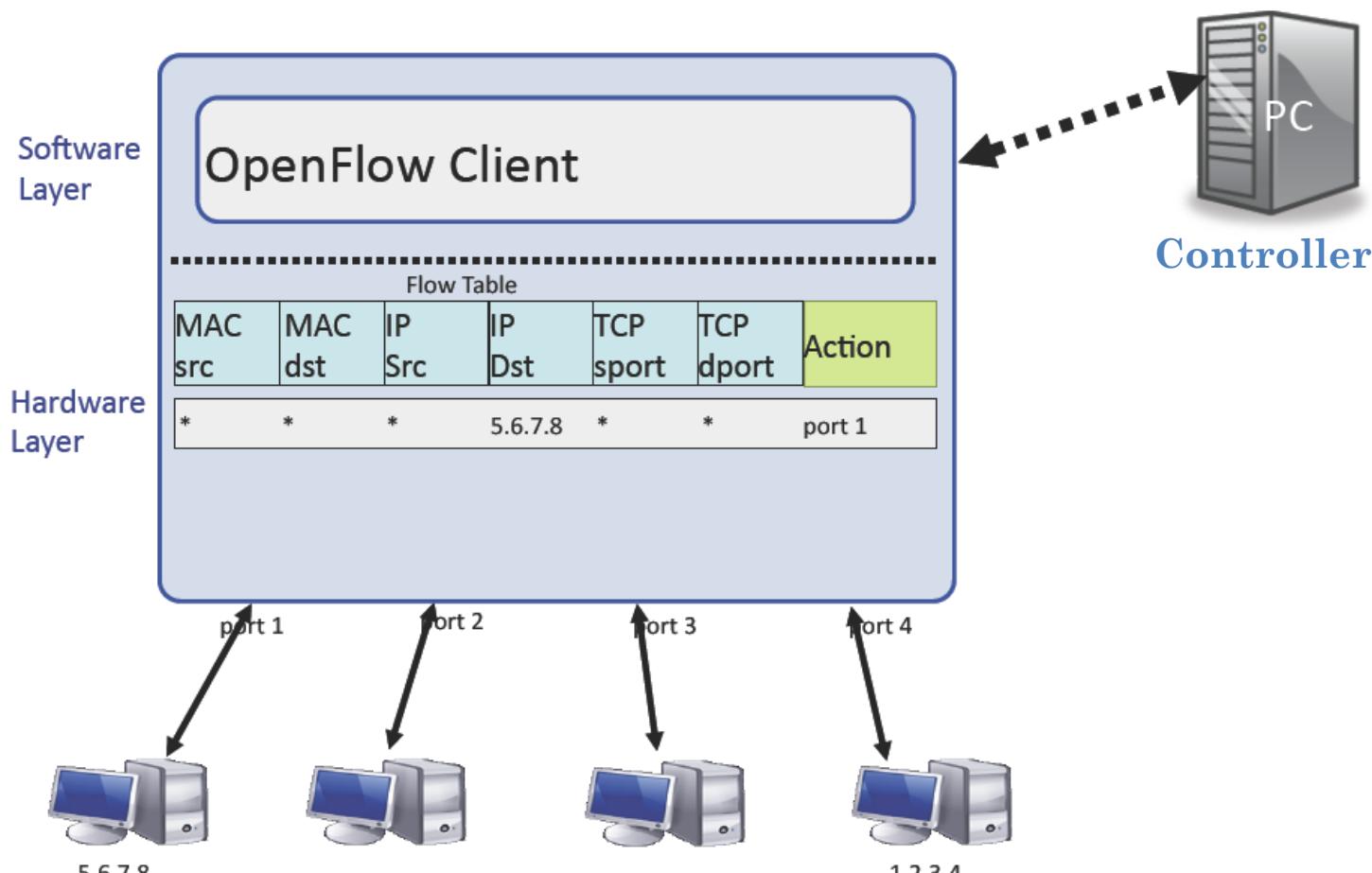


OpenFlow Switch

- Controller **manages** the traffic (network flows) by **manipulating** the **flow table** at switches.
 - Instructions are stored in flow tables.
- When packet arrives at switch, **match** the **header fields** with flow entries in a flow table.
- If any entry matches, performs indicated **actions** and update the **counters**.
- If Does not match, Switch asks controller by sending a message with the packet header.



OpenFlow Flow Table Abstraction



Flow entry examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f...	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	*	22 drop

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching (Efficient !!!)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

OpenFlow Table: Basic Actions

- **All**: To all interfaces except incoming interface.
- **Controller**: Encapsulate and send to controller.
- **Local**: send to its local networking stack.
- **Table**: Perform actions in the next flow table (table chaining or multiple table instructions).
- **In_port**: Send back to input port.
- **Normal**: Forward using traditional Ethernet.
- **Flood**: Send along minimum spanning tree except the incoming interface.

OpenFlow Table: Basic Stats

Per Table	Per Flow	Per Port	Per Queue
Active Entries	Received Packets	Received Packets	Transmit Packets
Packet Lookups	Received Bytes	Transmitted Packets	Transmit Bytes
Packet Matches	Duration (Secs)	Received Bytes	Transmit overrun errors
	Duration (nanosecs)	Transmitted Bytes	
		Receive Drops	
		Transmit Drops	
		Receive Errors	
		Transmit Errors	
		Receive Frame Alignment Errors	
		Receive Overrun errors	
		Receive CRC Errors	
		Collisions	

- Provide counter for incoming flows or packets.
- Information on counter can be retrieved to control plane.
- Can be used to monitor network traffic.

Additional Feature to Rules and Stats

OpenFlow Version	Match fields	Statistics	# Matches		# Instructions		# Actions		# Ports	
			Req	Opt	Req	Opt	Req	Opt	Req	Opt
v 1.0	Ingress Port	Per table statistics	18	2	1	0	2	11	6	2
	Ethernet: src, dst, type, VLAN	Per flow statistics								
	IPv4: src, dst, proto, ToS	Per port statistics								
	TCP/UDP: src port, dst port	Per queue statistics								
v 1.1	Metadata, SCTP, VLAN tagging	Group statistics	23	2	0	0	3	28	5	3
	MPLS: label, traffic class	Action bucket statistics								
v 1.2	OpenFlow Extensible Match (OXM)		14	18	2	3	2	49	5	3
	IPv6: src, dst, flow label, ICMPv6									
v 1.3	PBB, IPv6 Extension Headers	Per-flow meter	14	26	2	4	2	56	5	3
		Per-flow meter band								
v 1.4	—	—	14	27	2	4	2	57	5	3
		Optical port properties								

Controller Plane Software

- [**POX**](#): (Python) Out of Date.
- [**IRIS**](#): (Java) Scalability and High Availability
- [**MUL**](#): (C) MUL, is an openflow (SDN) controller. It has a C based multi-threaded infrastructure at its core.
- [**NOX**](#): (C++/Python) **NOX was the first OpenFlow controller.**
- [**Jaxon**](#): (Java) Jaxon is a NOX-dependent Java-based OpenFlow Controller.
- [**Trema**](#): (C/Ruby) Trema is a full-stack framework for developing OpenFlow controllers in Ruby and C.
- [**Beacon**](#): (Java) Beacon supports both event-based and threaded operation.
- [**Floodlight**](#): (Java) It was forked from the Beacon controller, originally developed by David Erickson at Stanford.
- [**ONOS**](#): (Java/OSGi) ONF controller, supports both configuration and real-time control of the network.
- And many more.

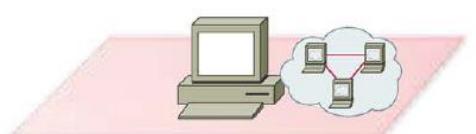
Basic OpenFlow Recap

SDN Concept:

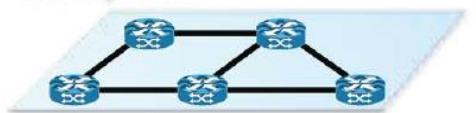
Management plane (Application Plane)



Control plane



Data plane



OpenFlow:

- Support different applications: routing, load balancers, monitoring, security, etc.
- Programmable: Modify and interact with the network model in control Plane.
- Global view of the entire network (the network model).
- Centralized per flow based control.
- Distributed system that creates a consistent, up-to-date network view (real time).
 - Runs on servers (controllers) in the network.
- Uses an open protocol to:
 - Get state information **from** switch.
 - Give control directives **to** switch.

Data and Control plane communicate via **secure Channel**

- Packet forwarding according to instruction stored in flow Tables.
- Provide statistic on network traffic to controller.
- Hardware: (Dump) Switches.

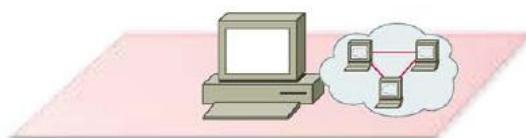
OpenFlow: More Details

SDN Concept

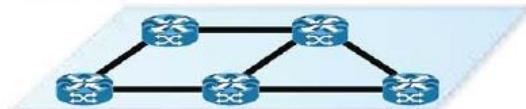
Management plane (Application Plane)



Control plane



Data plane



Different layers in OpenFlow

Network Applications

Routing, load balancers, security, etc.

Programming Languages

Language-based Virtualization

Northbound Interface

Make decisions and instructions

Network Operating System

Network Hypervisor

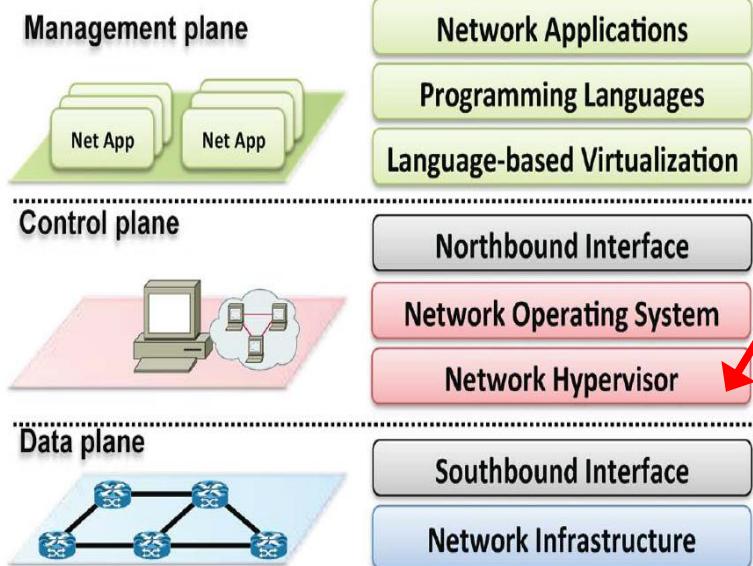
Southbound Interface

Firmware handling instructions from control plane (e.g Open Vswitch) via flow tables.

Network Infrastructure

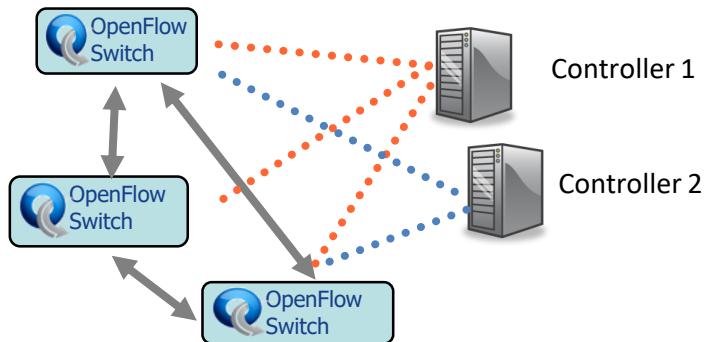
Hardware (switches)

Network Hypervisor (Virtualization)

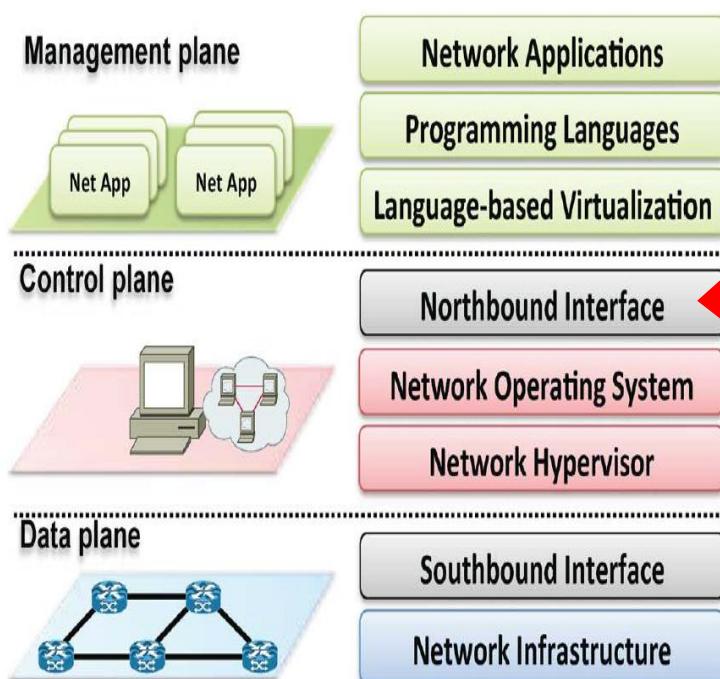


- Hide complexity (Dump it down)
 - Present only the necessary information and avoid too many details.
- Network operators “Delegate” control of subsets of network hardware and/or traffic to other network operators or users
- Multiple controllers can talk to the same set of switches.
- Allow experiments to be run on the network in isolation of each other and production traffic.
- Virtualized network model (topology, routing, etc.).

Multiple Controllers scenario is possible

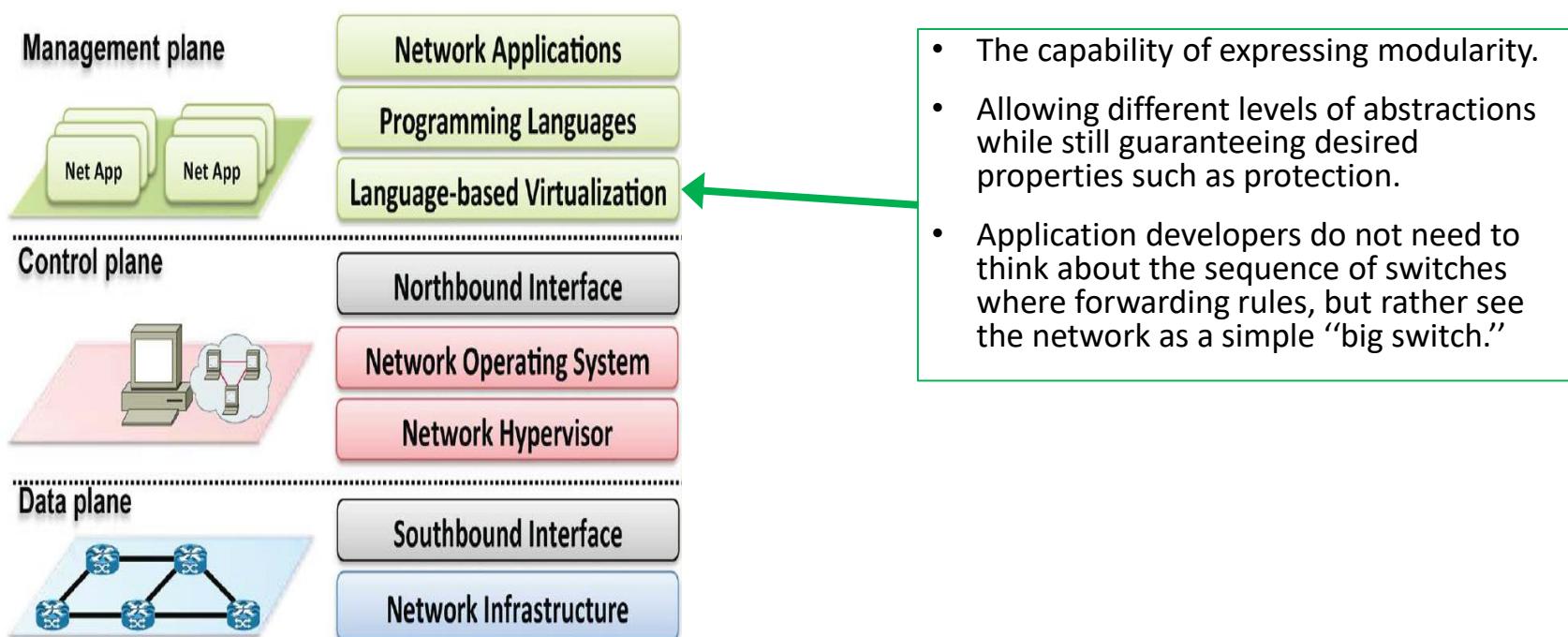


Northbound Interface

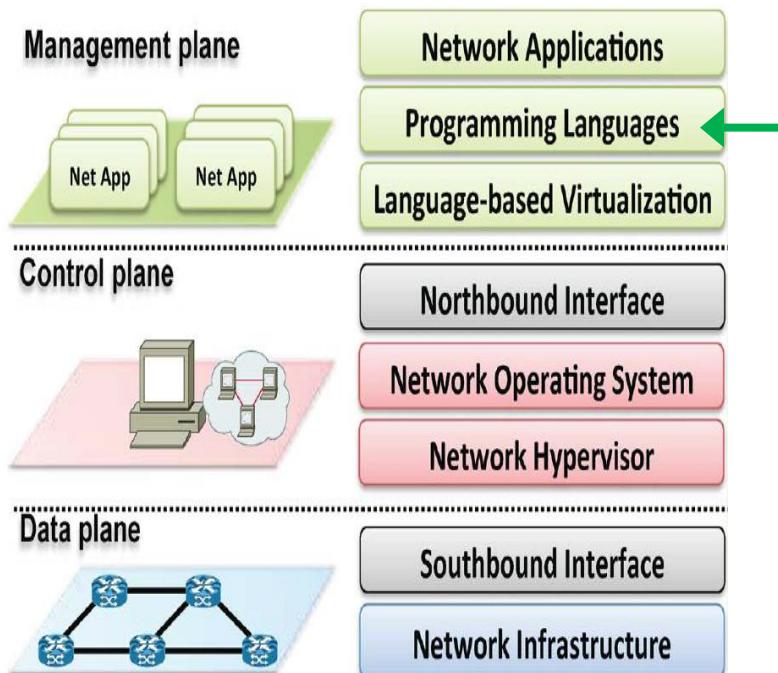


- API (interface) to management plane or applications.
- Open issue.
- No Standardization.
- Software based ecosystem.
- Considered new theme in SDN as 2015.

Language-based Virtualization

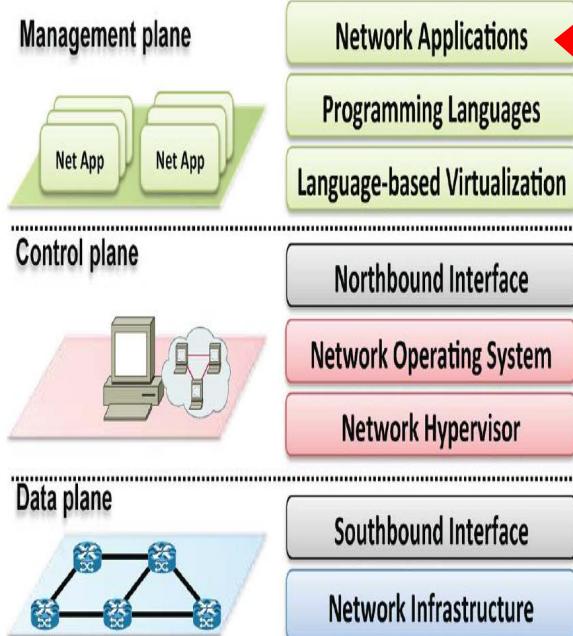


Programming Language



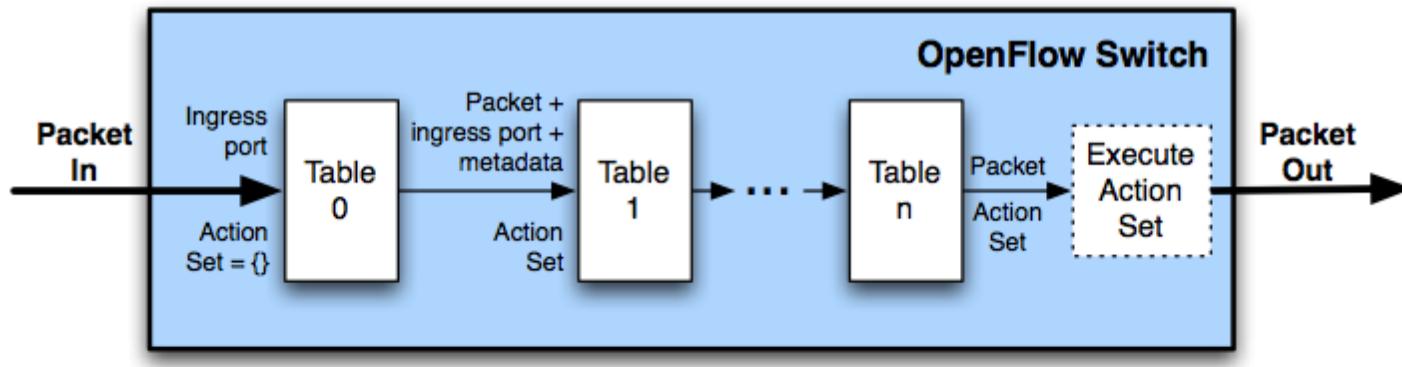
- Programming language, abstraction, and interfaces to implement SDN.
- Ensure multiple tasks of a single application do not interfere with others.
- Checking conflicted rules.
- Provide higher level programming interface to avoid low level instructions and configuration.
- Special abstraction for management requirements (e.g monitoring).
- Regular expressions.
- Etc.

Network Applications: Software for *Data Center Networking*

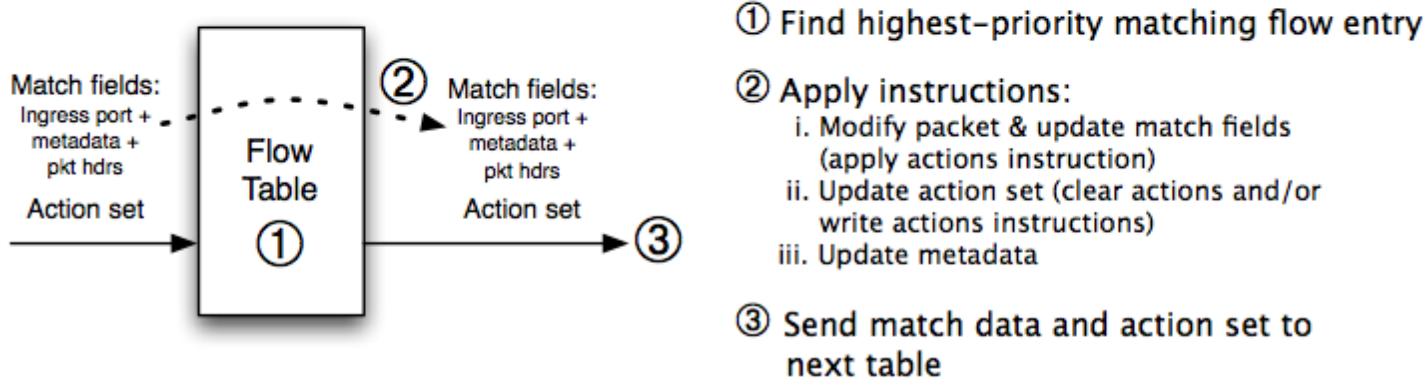


- **Big Data Apps:** Optimize network Utilization.
- **CloudNaaS:** Networking primitives for cloud apps, NOX controller.
- **FlowComb:** Predict Apps workload, uses NOX.
- **FlowDiff:** Detects Operational Problems, FlowVisor Controller.
- **LIME:** Live Network migration, FloodLight Controller.
- **NetGraph:** Graph Queries for network management, uses its own controller.
- **OpenTCP:** Dynamic and programmable TCP adaptation, uses its own controller.
- All of them employ OpenFlow to communicate with switches, except *OpenTCP*.

OpenFlow matching process



(a) Packets are matched against multiple tables in the pipeline



(b) Per-table packet processing

ONF, “OpenFlow Switch Specification v.1.3.0”, June, 2012

SDN Basis

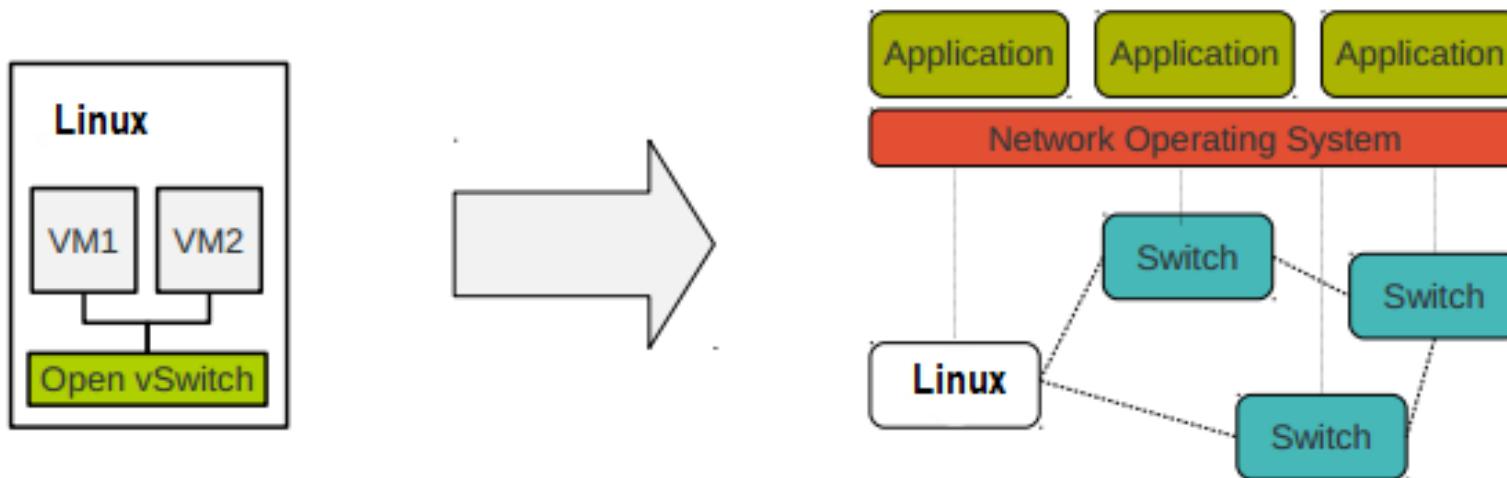
- Concept of SDN
- OpenFlow – a SDN Implementation
- **Open vSwitch**

What is Open vSwitch

- Open vSwitch (OVS) is an OpenFlow-based multilayer software switch licensed under the open source Apache 2 license (User Space) and GPL (Kernel).
- OVS is a **virtual switch** for hypervisors providing network connectivity to VMs.
- It exposes **standard control and visibility interfaces** to the virtual networking layer.
- It was designed to support distribution **across multiple physical servers**.
- OVS supports multiple Linux-based virtualization technologies including Xen/XenServer, KVM, and VirtualBox.

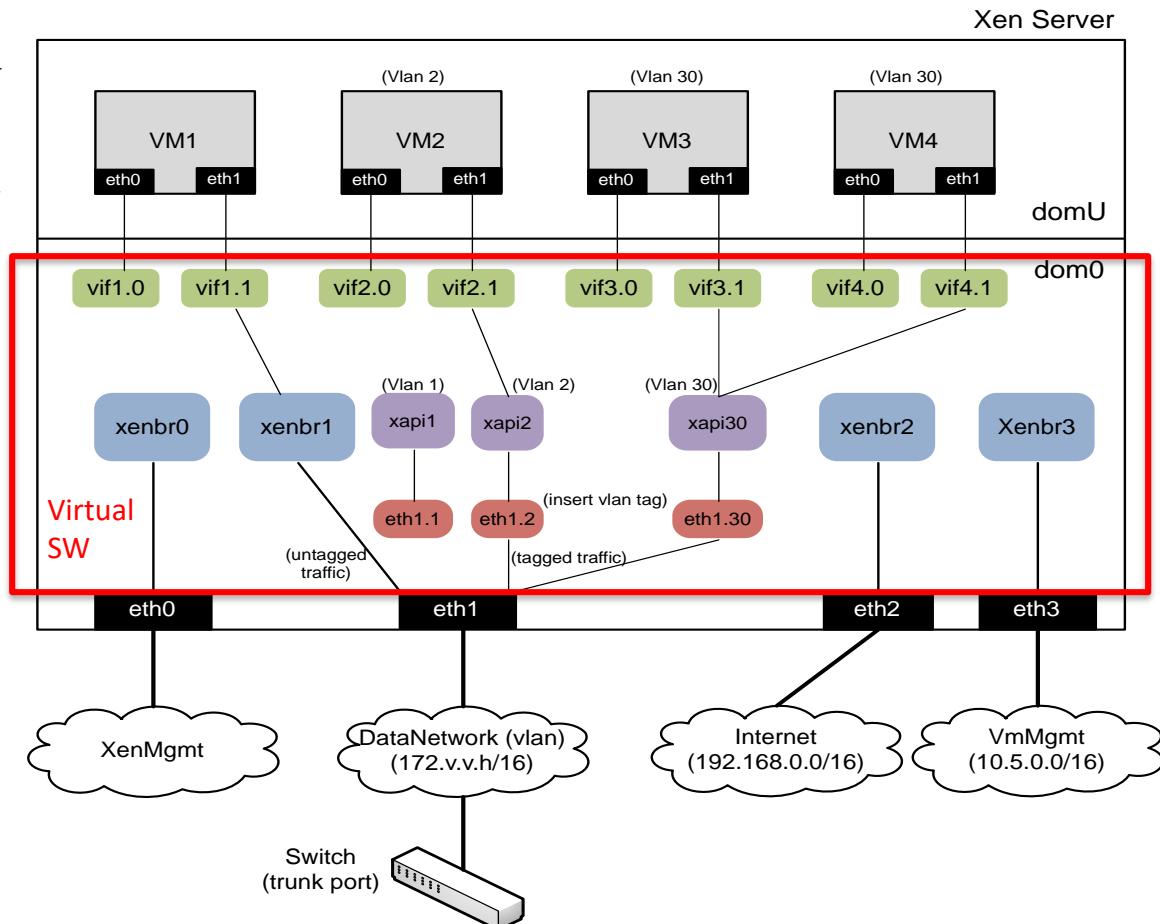
Why Open vSwitch

- Open vSwitch enables Linux to become part of a SDN architecture.



Xen & Virtual Software Networking

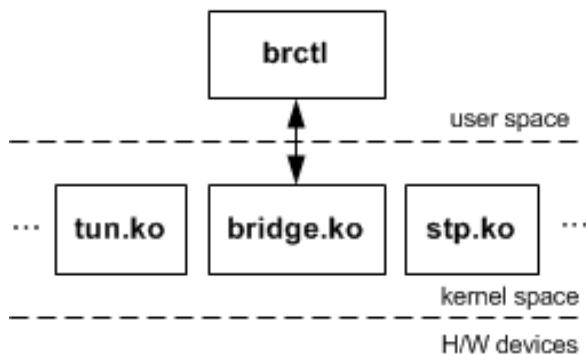
- The old version of Citrix XenServer (before v5.6 FP1) using simple Linux Bridge.
- Many hypervisor based virtualization also apply Linux Bridge model, such as KVM, libvirt.
- All of bridging work are done by ‘brctl’ .
- Provide simple L2 switching functions.



Linux Bridge

- The Linux bridge code implements a subset of the ANSI/IEEE 802.1d standard.
- The original Linux bridging was first done in Linux 2.2. The code for bridging has been integrated into 2.4 and 2.6 kernel series.
- Bridging functions are handled by bridge.ko kernel module and managed by user space management tool brctl command.

Linux Bridge Command

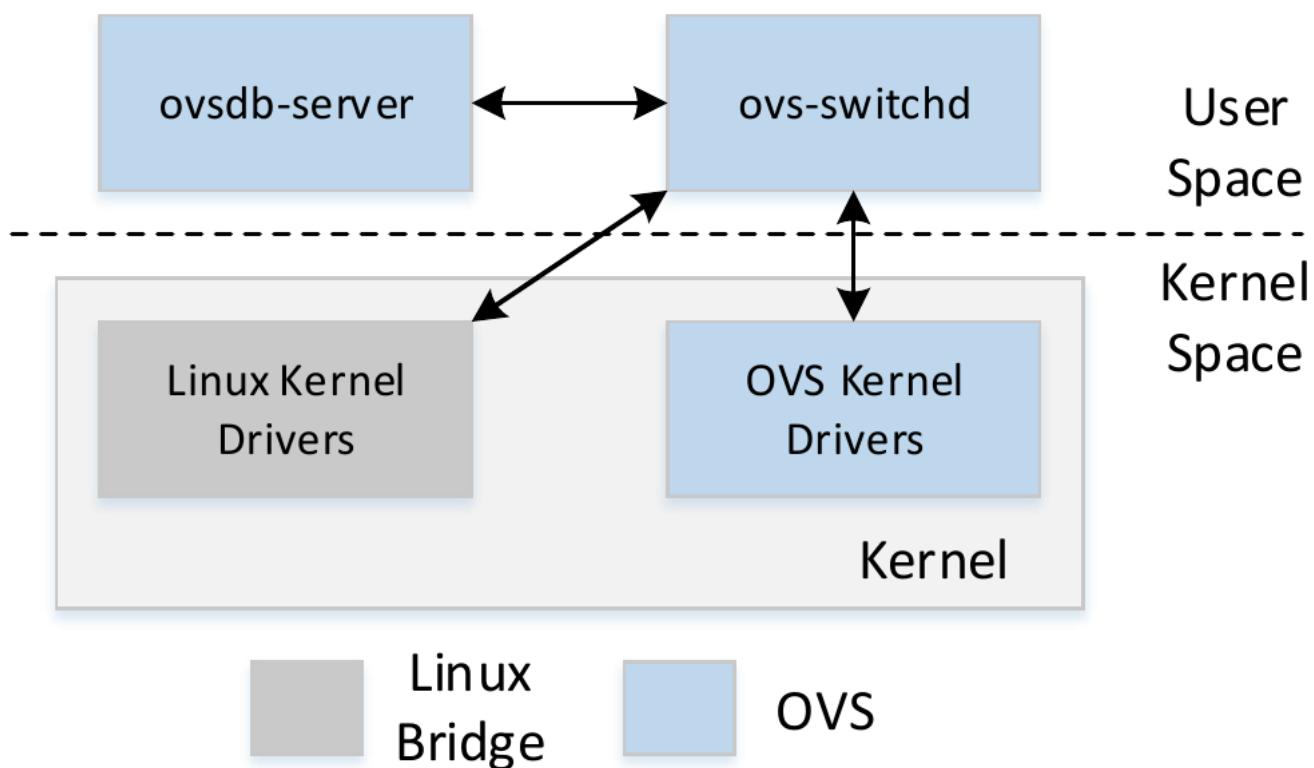


# brctl	# commands:	
	addbr	<bridge>
	delbr	<bridge>
	addif	<bridge> <device>
	delif	<bridge> <device>
	setageing	<bridge> <time>
	setbridgeprio	<bridge> <prio>
	setfd	<bridge> <time>
	sethello	<bridge> <time>
	setmaxage	<bridge> <time>
	setpathcost	<bridge> <port> <cost>
	setportprio	<bridge> <port> <prio>
	show	
	showmacs	<bridge>
	showstp	<bridge>
	stp	<bridge> <state>

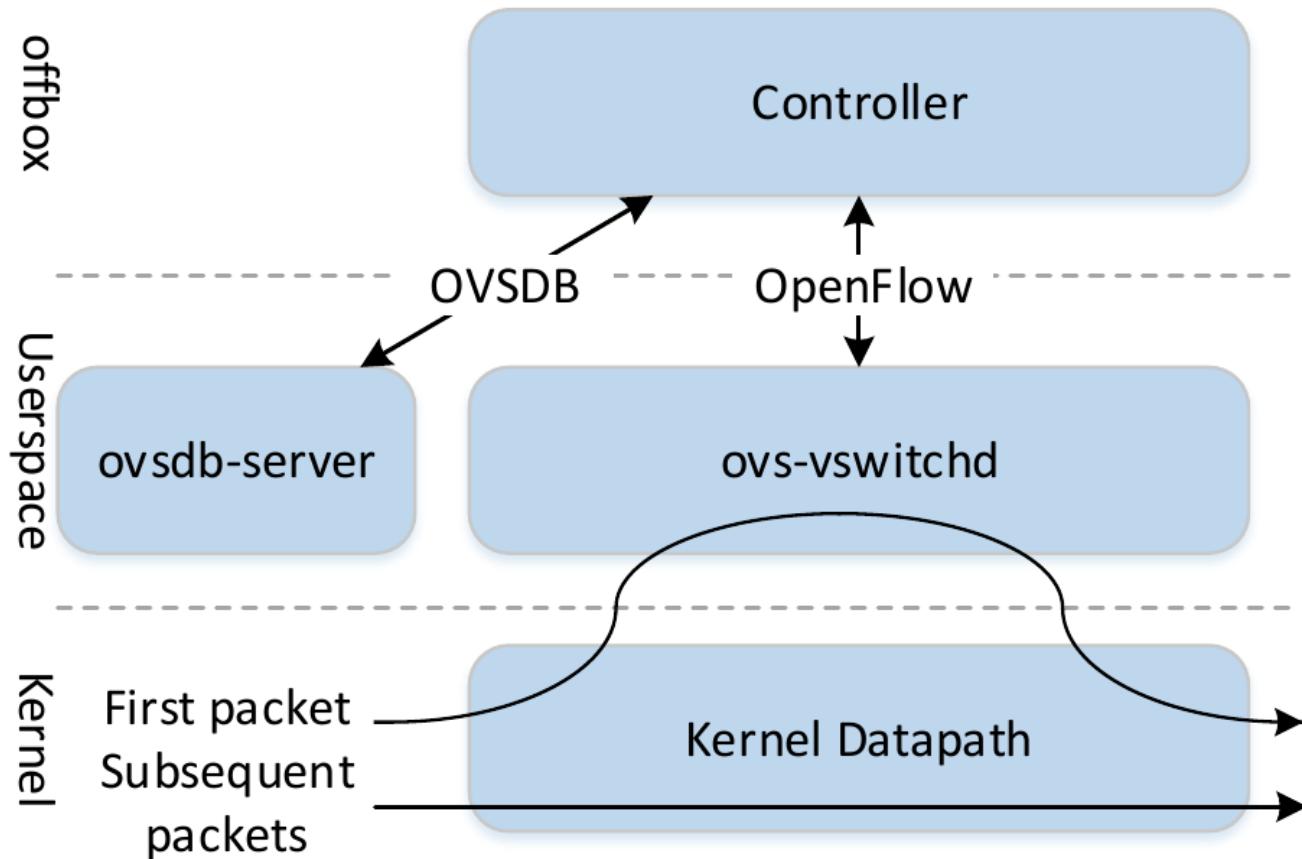
The source code of bridge.ko in Linux kernel 2.6.38:

<http://lxr.linux.no/#linux+v2.6.38/net/bridge>

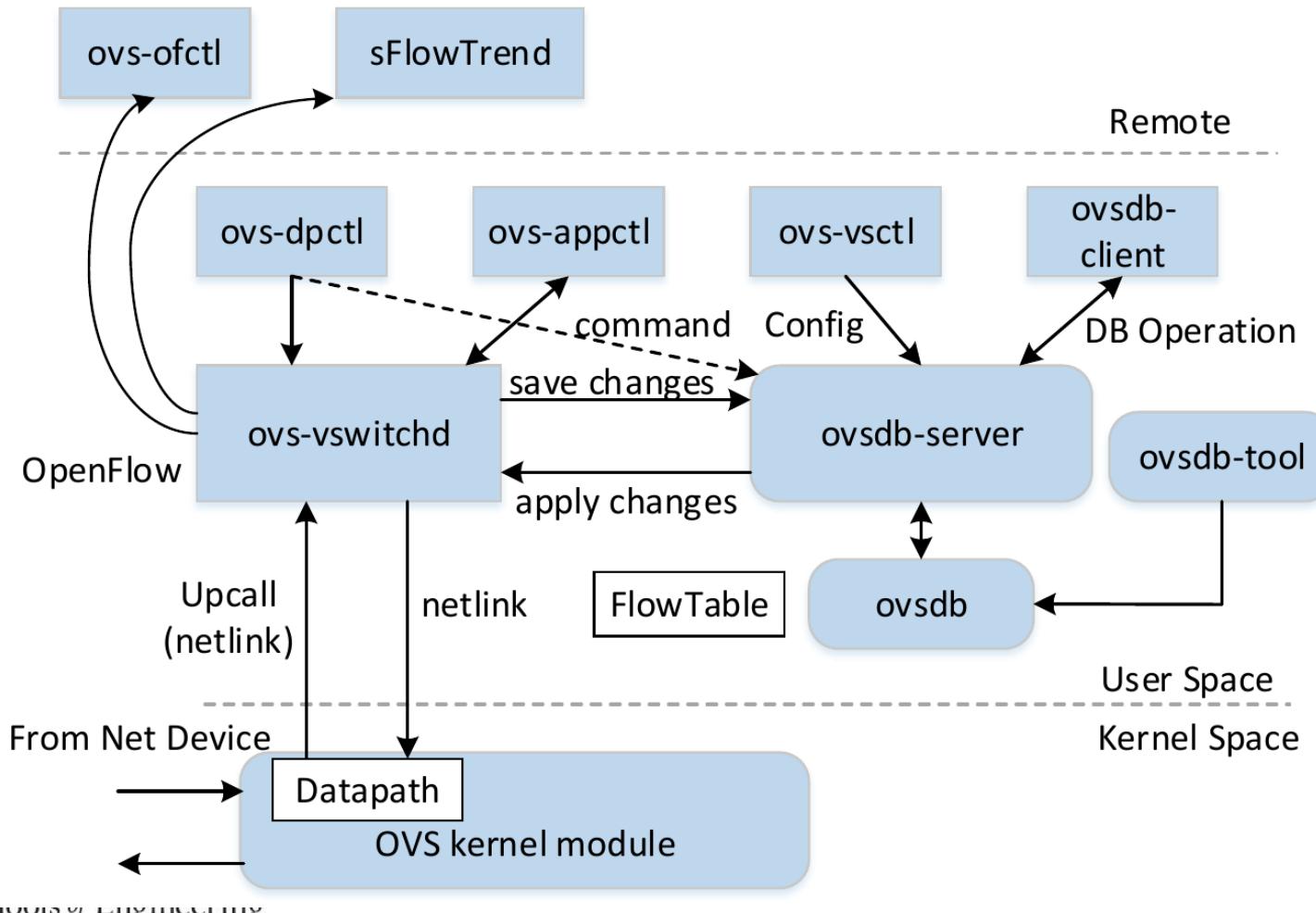
Linux Bridge and OVS



OVS Packet Processing



Main Components of OVS



Open vSwitch's Features

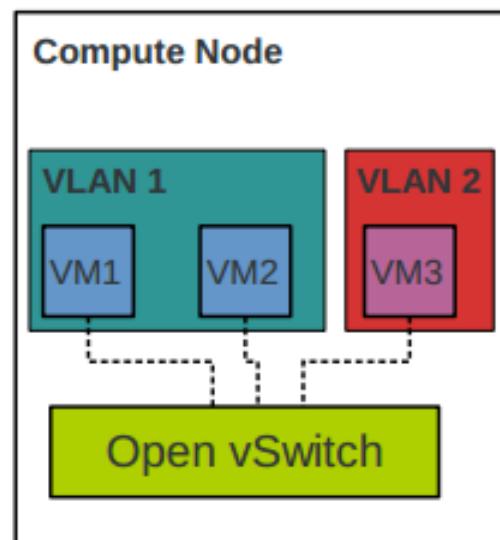
- Visibility:
 - NetFlow, sFlow, Mirroring (SPAN/RSPAN/ERSPAN)
- Control:
 - Centralized control through OpenFlow
 - Missed flows go to central controller
 - Fine-grained ACL and QoS (Quality of Service) policies
 - L2-L4 matching and actions to forward, drop, modify, and queue
- Forwarding:
 - LACP (Link Aggregate Control Protocol)
 - Port bonding
 - Standard 802.1Q VLAN model with trunk and access ports
 - GRE, GRE over IPSEC, Ethernet-over-GRE and CAPWAP tunneling
- Compatibility layer for Linux bridging code
- High-performance forwarding using a Linux kernel module

Feature – security/L2 segregation

- VLAN isolation enforces VLAN membership of a VM without the knowledge of the guest it self.

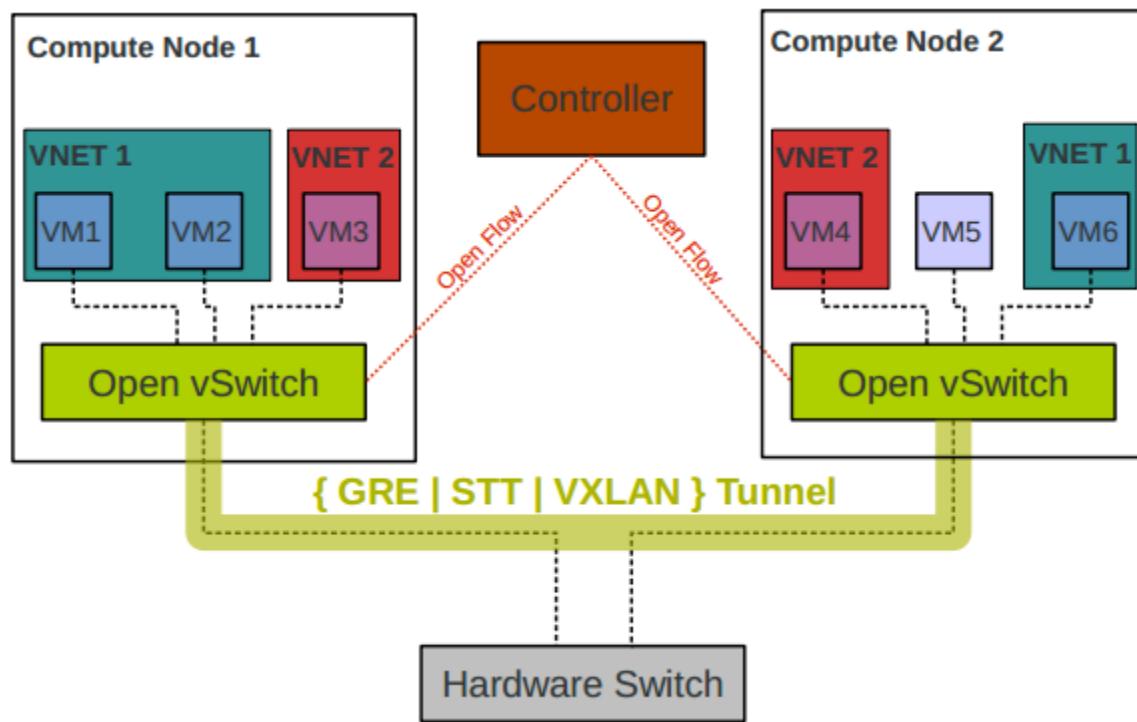
```
# ovs-vsctl add-port ovsbr port2 tag=10
```

Any limit for VLAN ID?

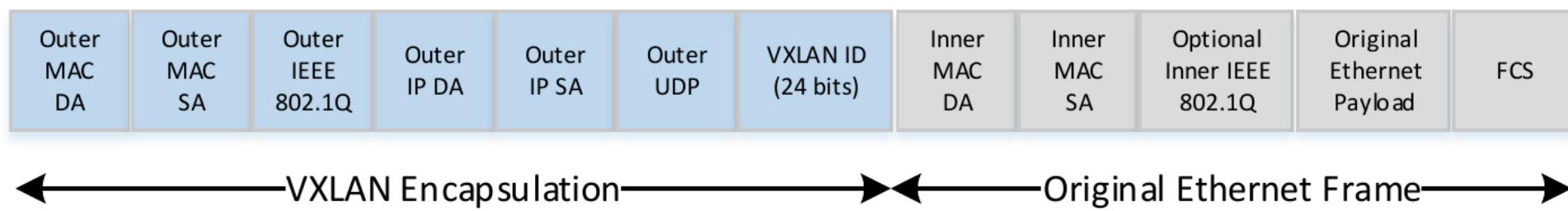
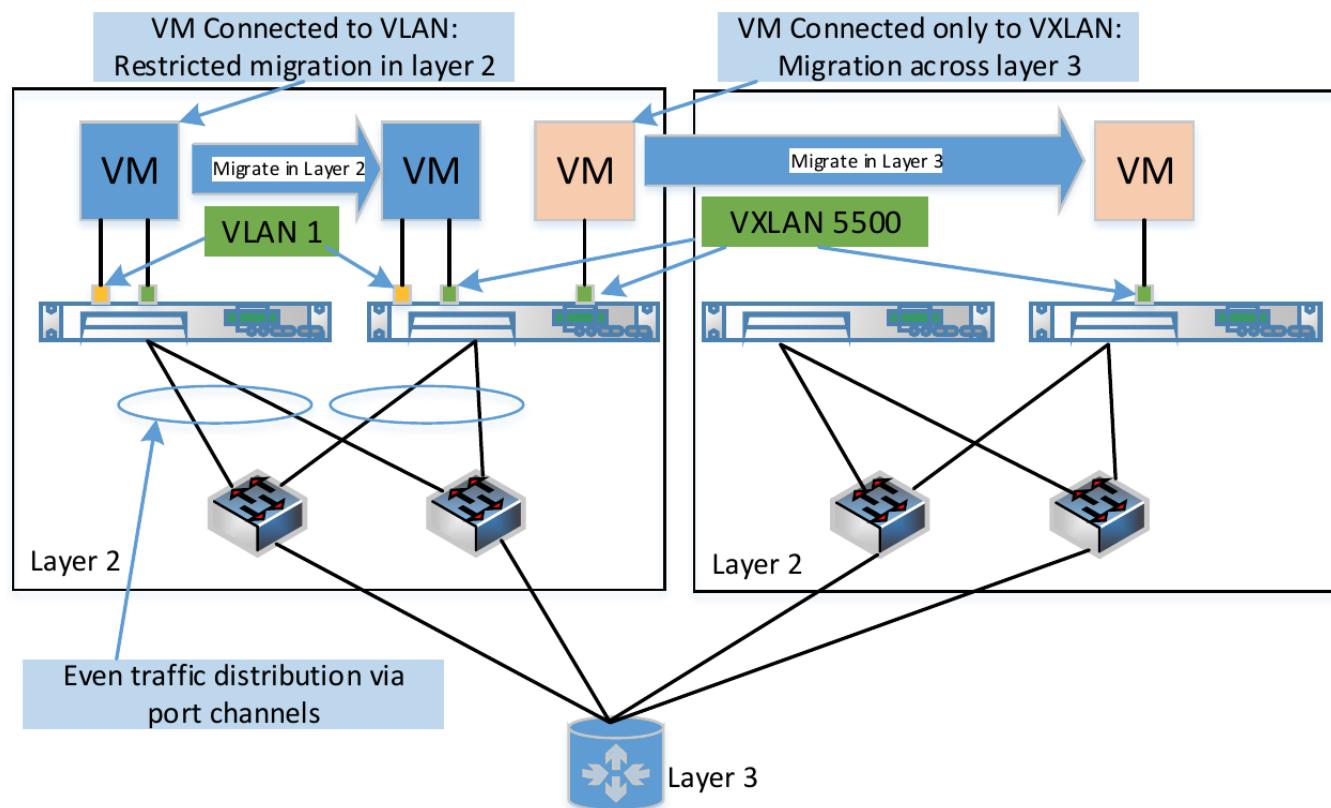


Feature – Tunneling

- Tunneling provides isolation and reduces dependencies on the physical network.



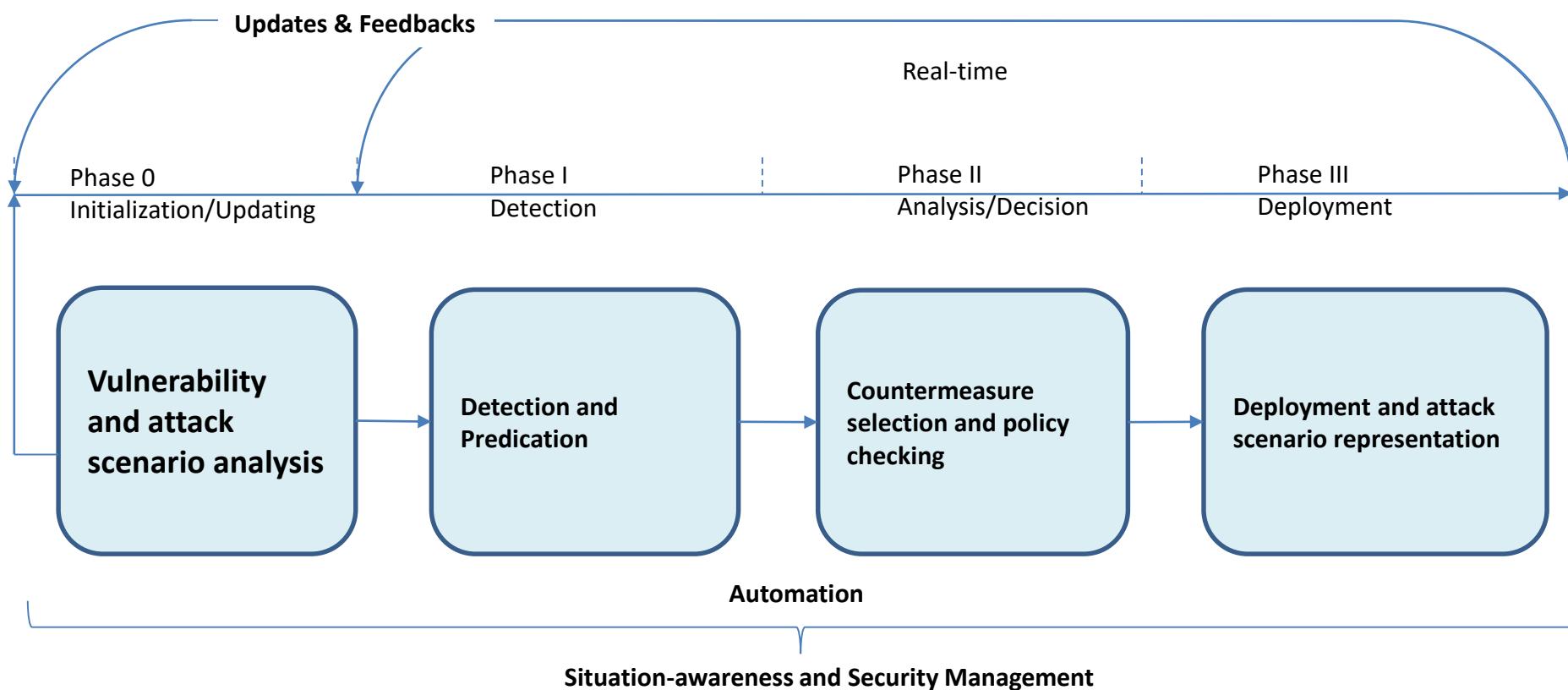
VXLAN Port migration and packet format



Agenda

1. Moving Target Defense (MTD)
2. **SDN-Based MTD Approaches**
 - SDN Basis
 - **A Case Study: An Intelligent SDN-based MTD Approach**
 - Research challenges, opportunities, and future directions
3. Q&A and Discussion

A case Study: An Intelligent Software Defined Security Architecture



A datacenter security scenario: VMs are fundamental cloud components

- They are dynamic and collocated on physical cloud servers.
- They share resources (computing, communication and storage).
- They are gentlemen when they are controlled by a centralized authority and free for malicious codes.
- They are evil when attackers deploy attacks based on them – lateral movements of attacks.

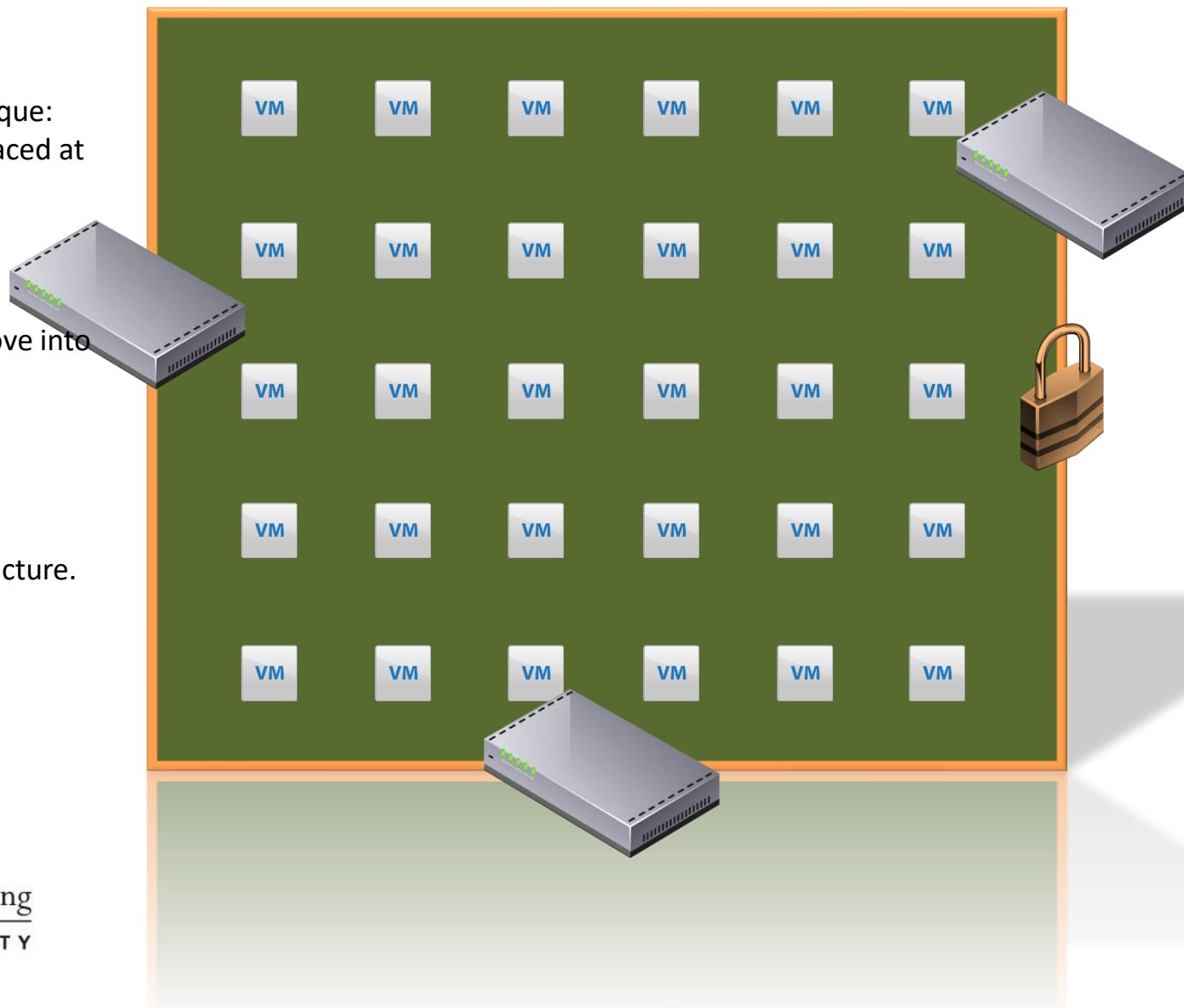


Protect the boundary

- Establishing the security perimeters is the traditional fundamental security technique: physical devices statically placed at boundaries

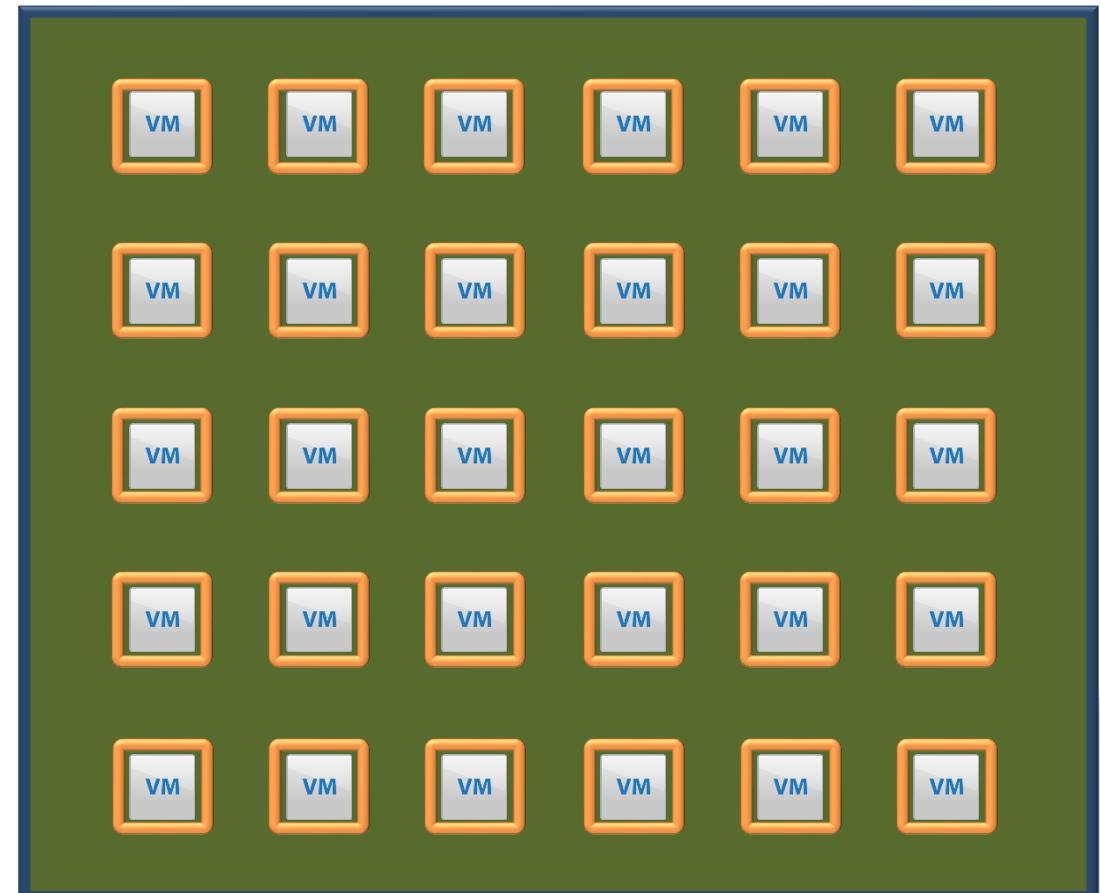
In cloud:

- Move into Software AND move into virtual fabric
- BIG MOVE FROM securing virtualized infrastructure to virtualizing security infrastructure.



Micro-segmented Protection

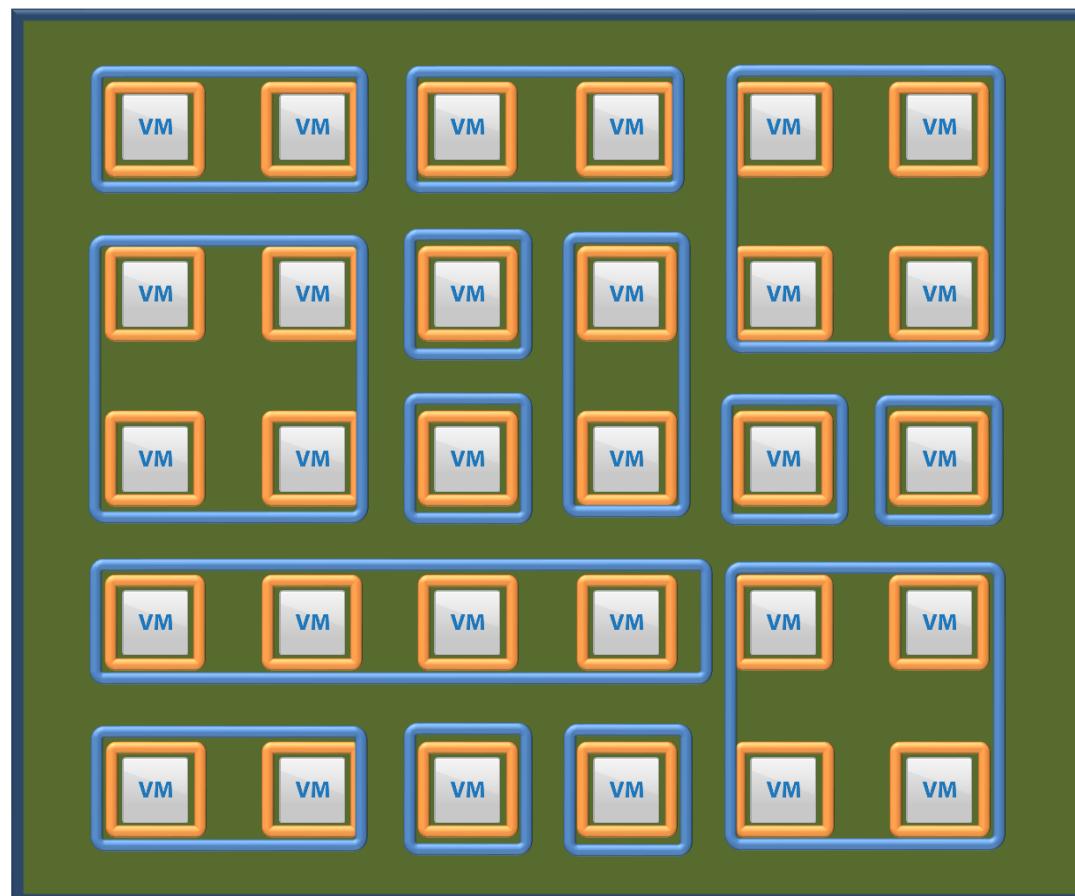
- Protect the data stored and processed in each individual vm.
- Monitor and audit processes and executions in each individual vm.
- Prevent data loss for each vm.



Protect the VM Content and safeguard VM activities.

Secure the virtual local networking

- Boundaries of VMs can be constructed based on the composing Apps and working relations.
- Local networks provide secure isolations and prevent traditional layer-2 attacks.
- The virtual network construction can be dynamic and configurable through a software-based approach.



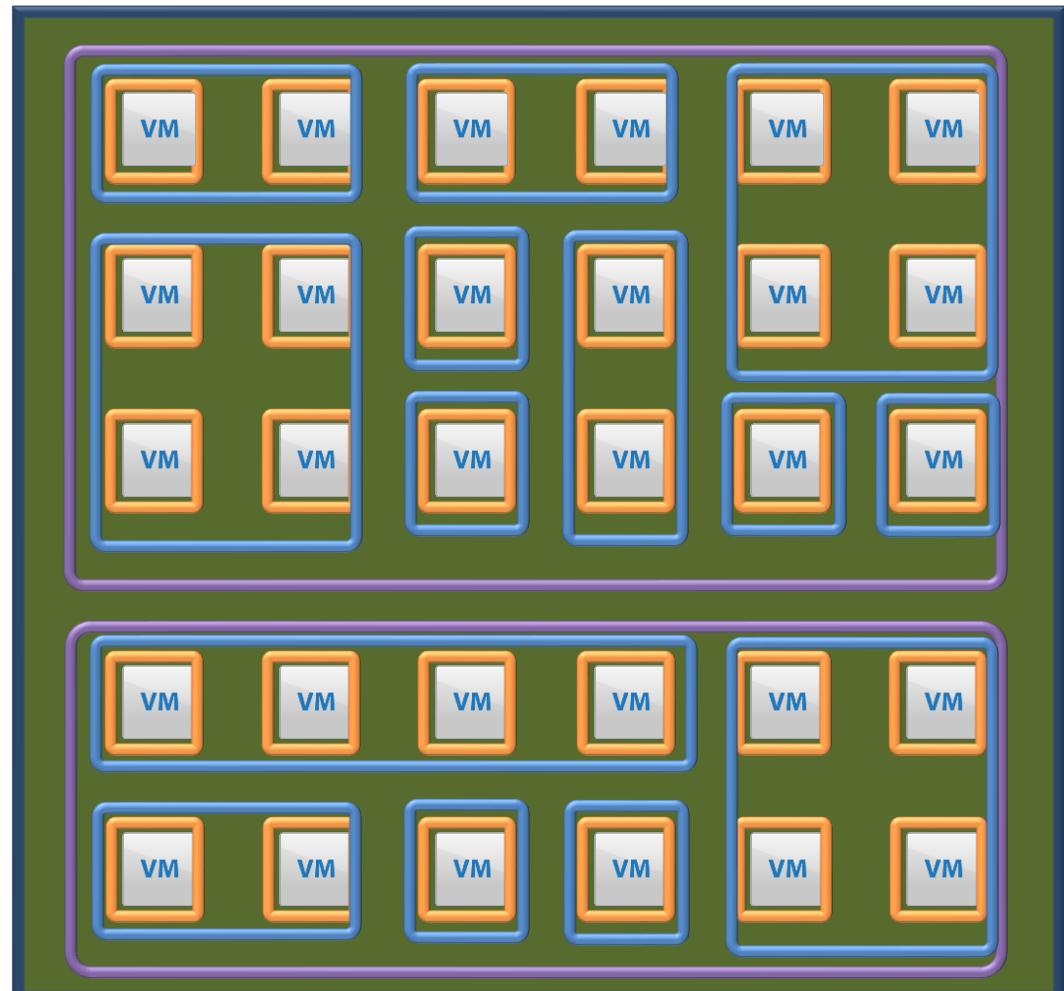
Protect the Logical Apps

Secure the Internetworking

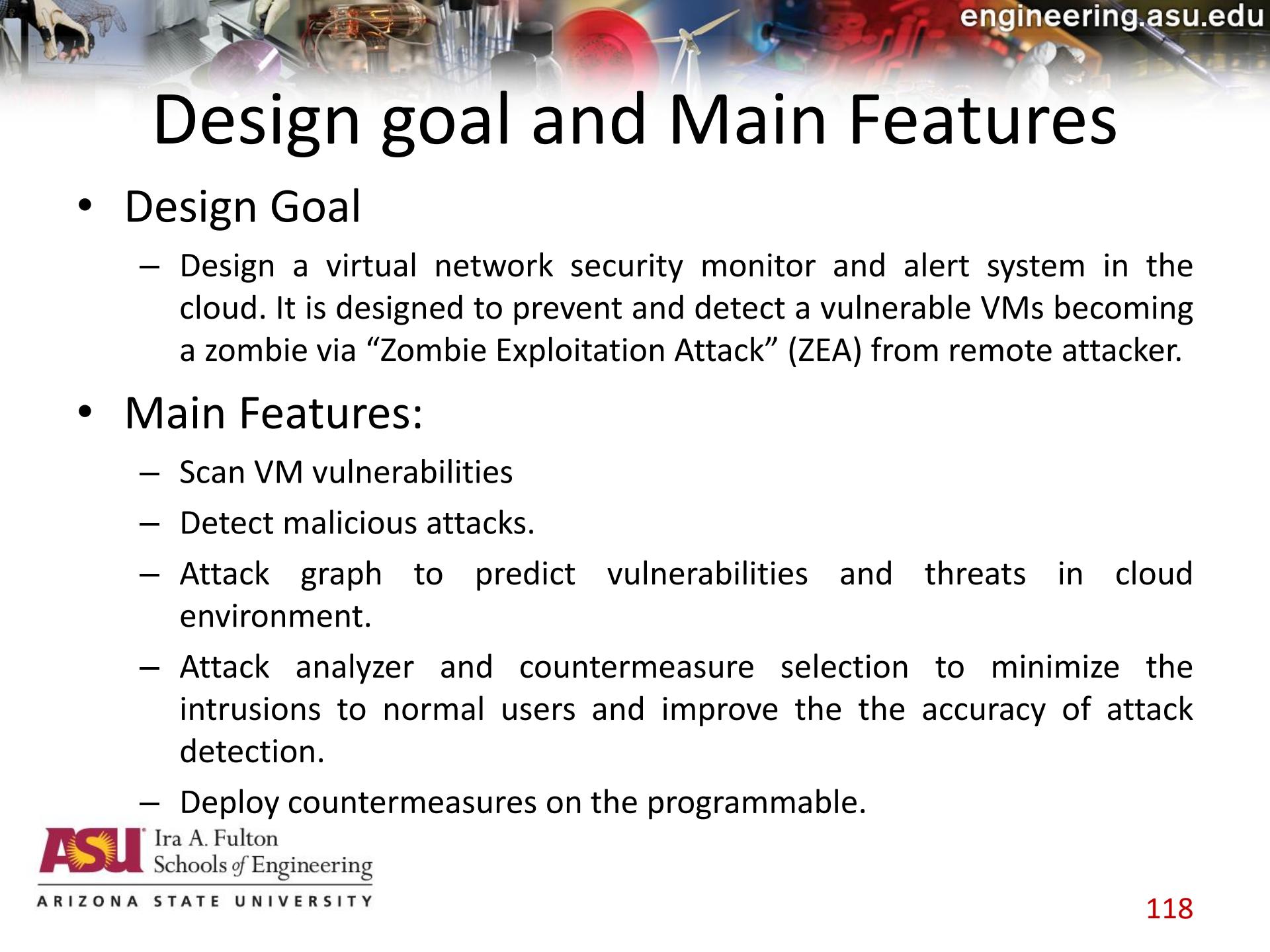
- Inter-networking can be established based on overlay networking techniques.
- Security policies can be enforced by changing the network topology and firewall rules established among virtual networks and VMs.

Research challenges to answer the following questions:

- Why we need to partition the entire system into multiple segments and establish secure firewalls among them?
- When we need to isolate VMs or virtual networks?
- How to reconstruct the virtual networking systems.



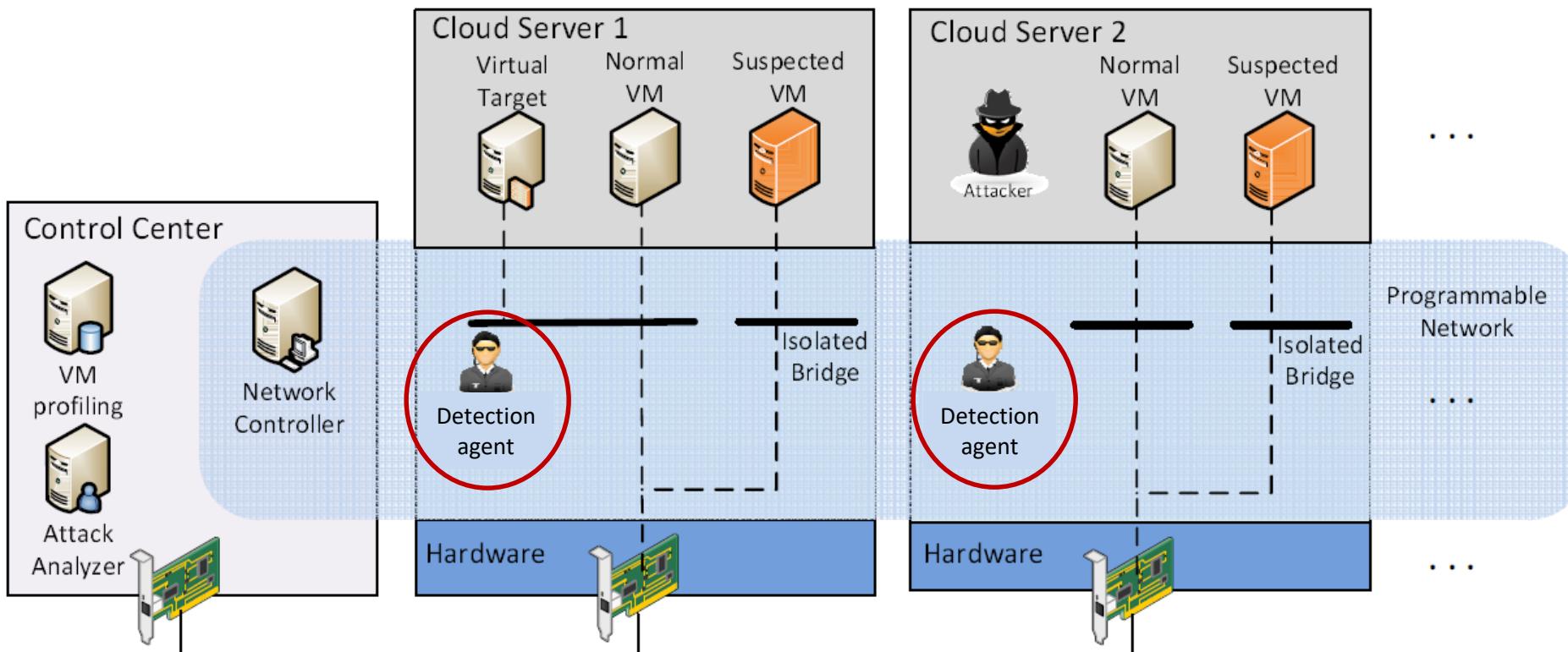
Protect the Logical Datacenter



Design goal and Main Features

- Design Goal
 - Design a virtual network security monitor and alert system in the cloud. It is designed to prevent and detect a vulnerable VMs becoming a zombie via “Zombie Exploitation Attack” (ZEA) from remote attacker.
- Main Features:
 - Scan VM vulnerabilities
 - Detect malicious attacks.
 - Attack graph to predict vulnerabilities and threats in cloud environment.
 - Attack analyzer and countermeasure selection to minimize the intrusions to normal users and improve the the accuracy of attack detection.
 - Deploy countermeasures on the programmable.

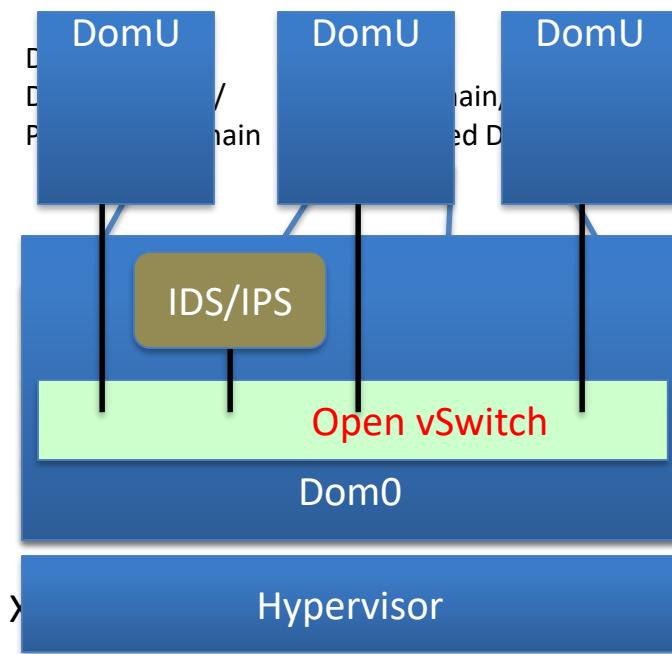
Virtual System Setup Example



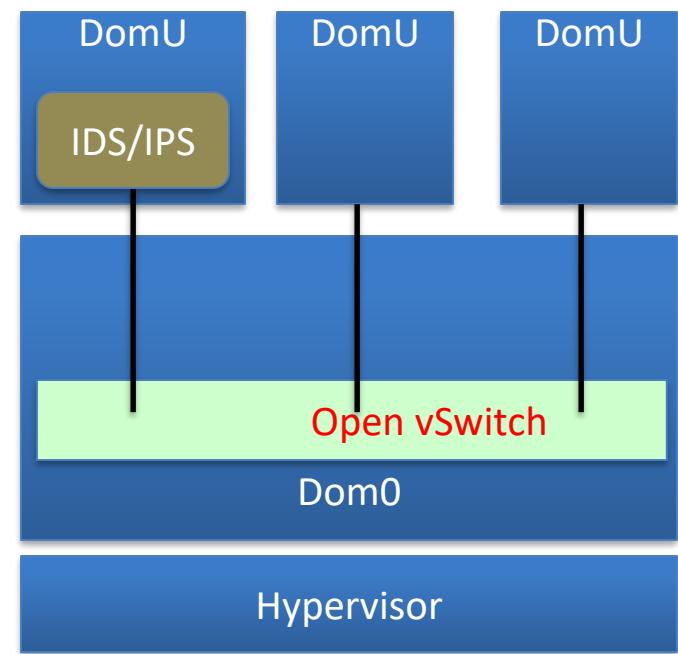
Major Components

- **Detection Agent:** Distributed attack detection agent installed on each cloud server (based on XEN terminologies)
 - Dom0 IDS/IPS
 - DomU proxy-based IDS/IPS
 - DomU mirroring-based IDS/IPS
- **Control Center**
 - *Attack Analyzer*: receives alerts from detection agent, assesses the severity, and generates possible countermeasures.
 - *Attack Graph Constructor*: constructs the attack graph based on the vulnerabilities and alerts.
 - *Network Controller*: applies the countermeasure in the programmable network (e.g., change network topology or redirect the flow in the software switch)

Detection Agent Placement

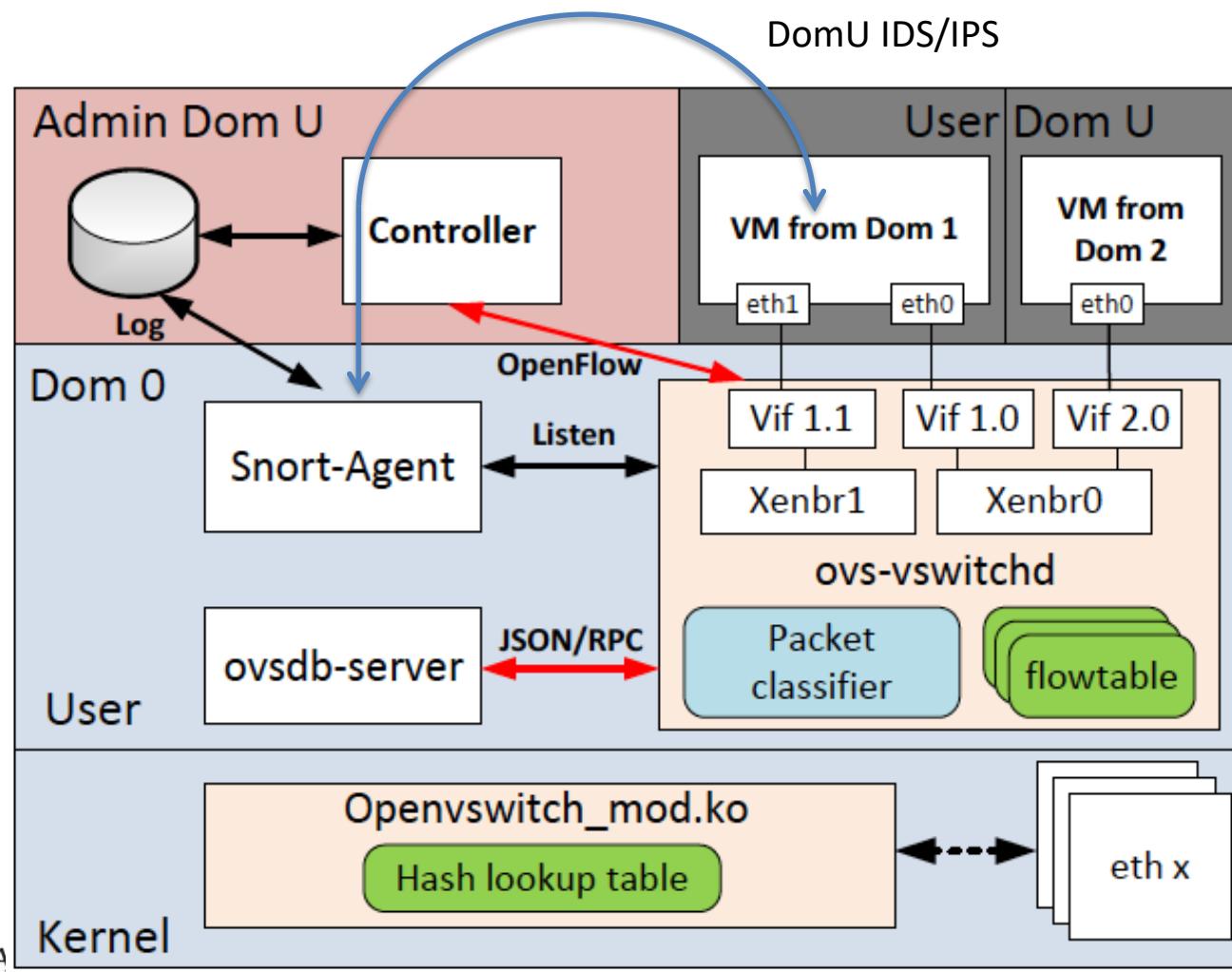


Privileged Domain Placement



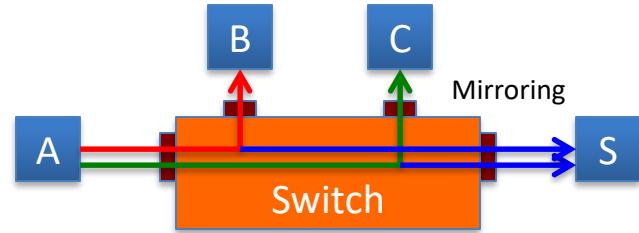
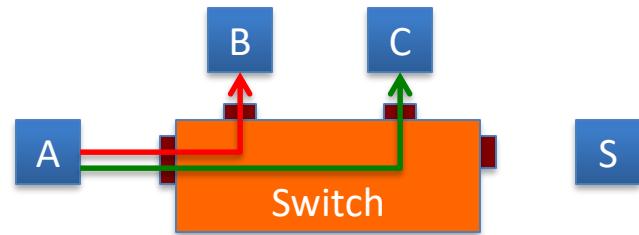
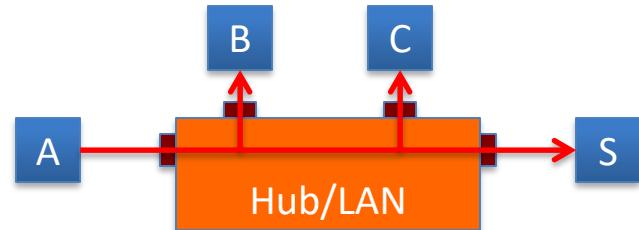
unprivileged Domain Placement

Dom0 and DomU IDS/IPS

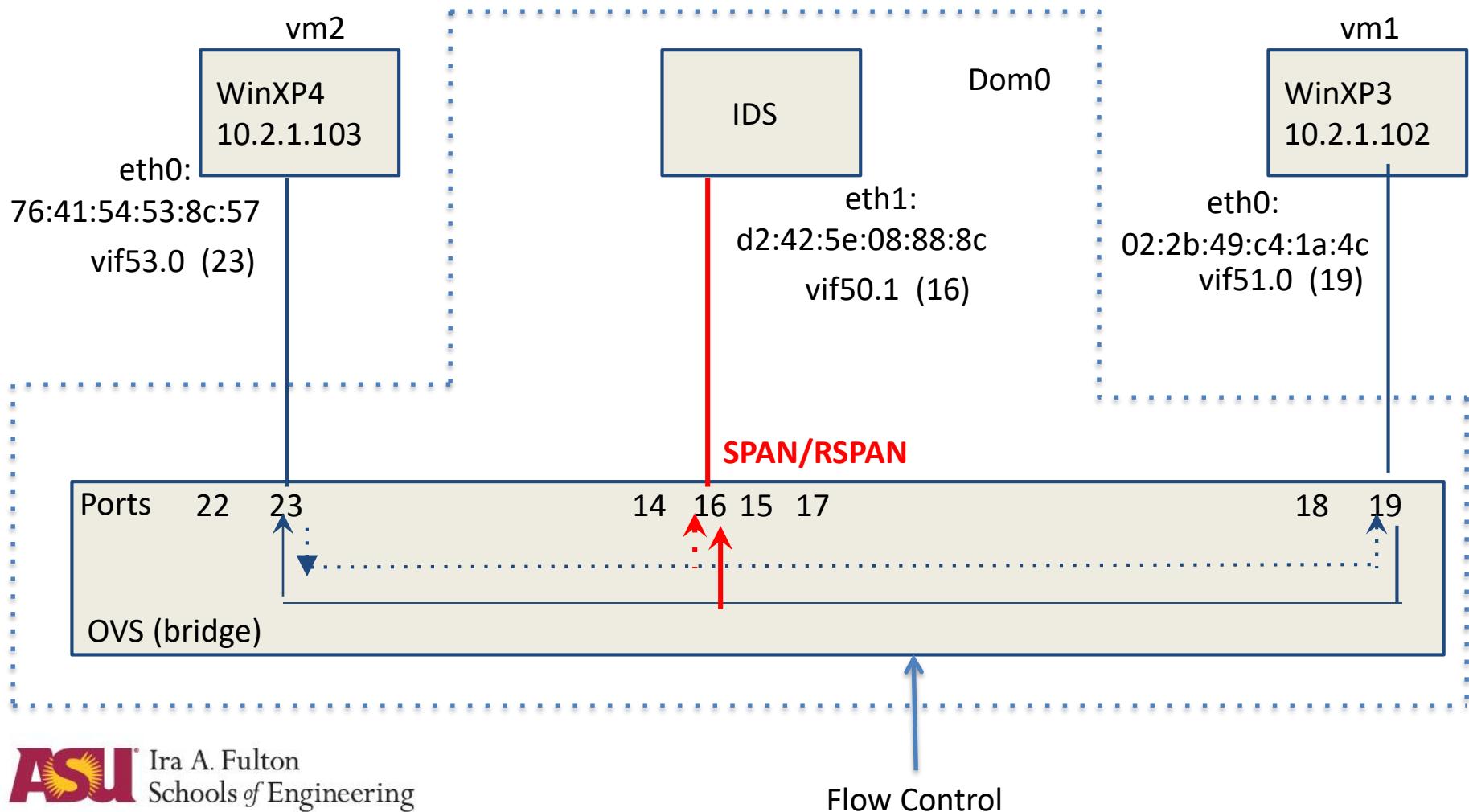


LAN technique in: Port Mirroring

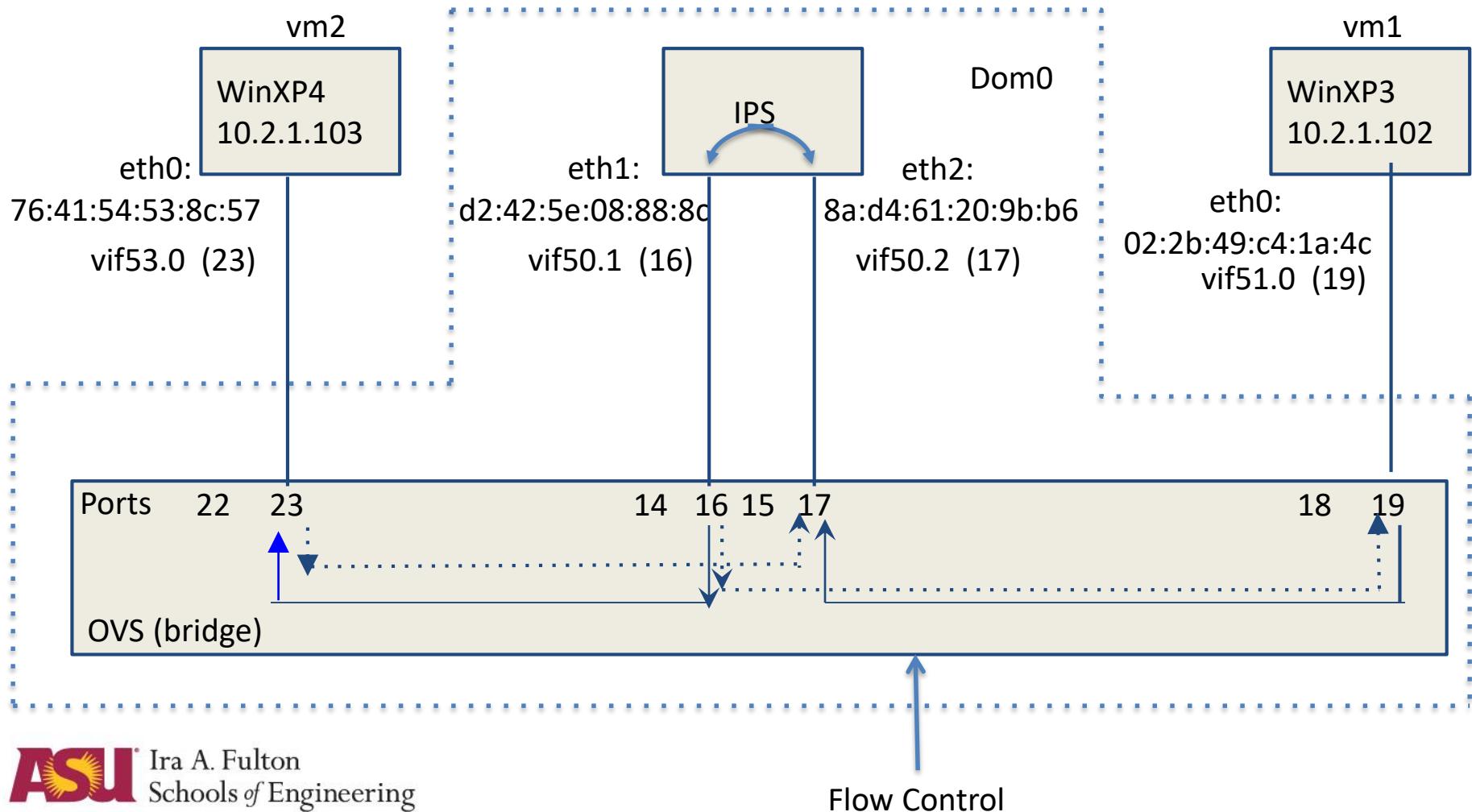
- In the past, with a shared hub, packets are replicated on all ports, plug a “sniffer” into any data port will see all traffic.
- With a LAN switch, traffic travels point-to-point only; packets no longer replicated; network visibility is lost
- SPAN/RSPAN/ERSPAN (or mirroring) port was invented to replicate packets of a single port or a single VLAN for monitoring.



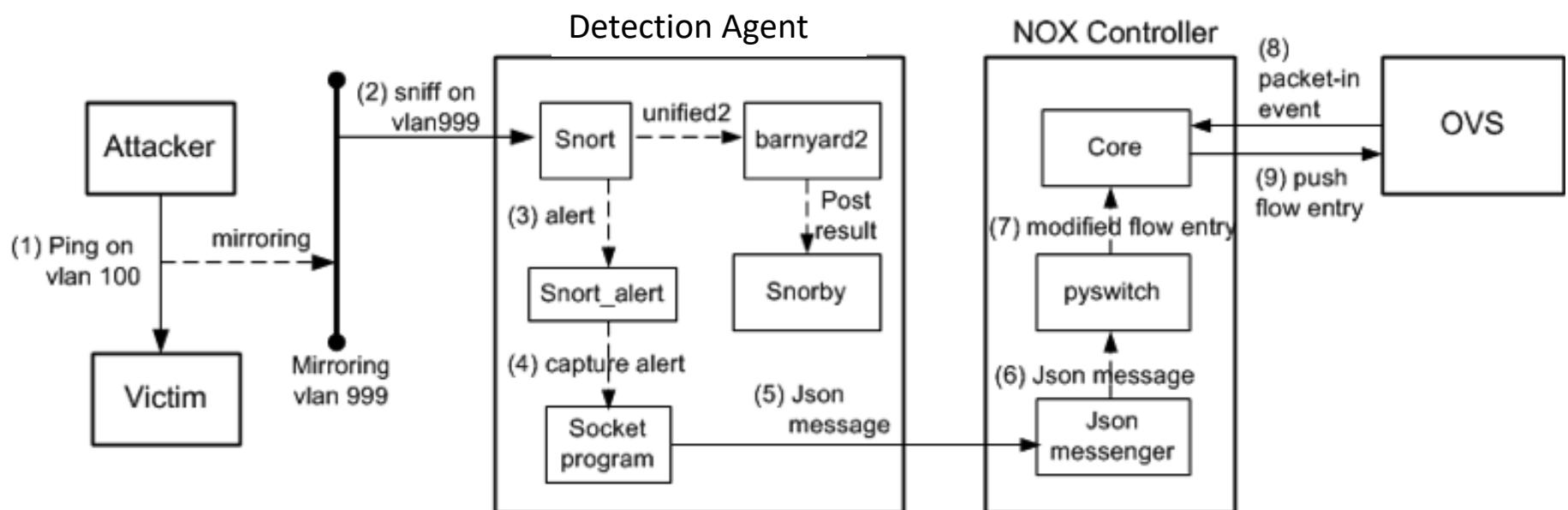
IDS Virtual Networking



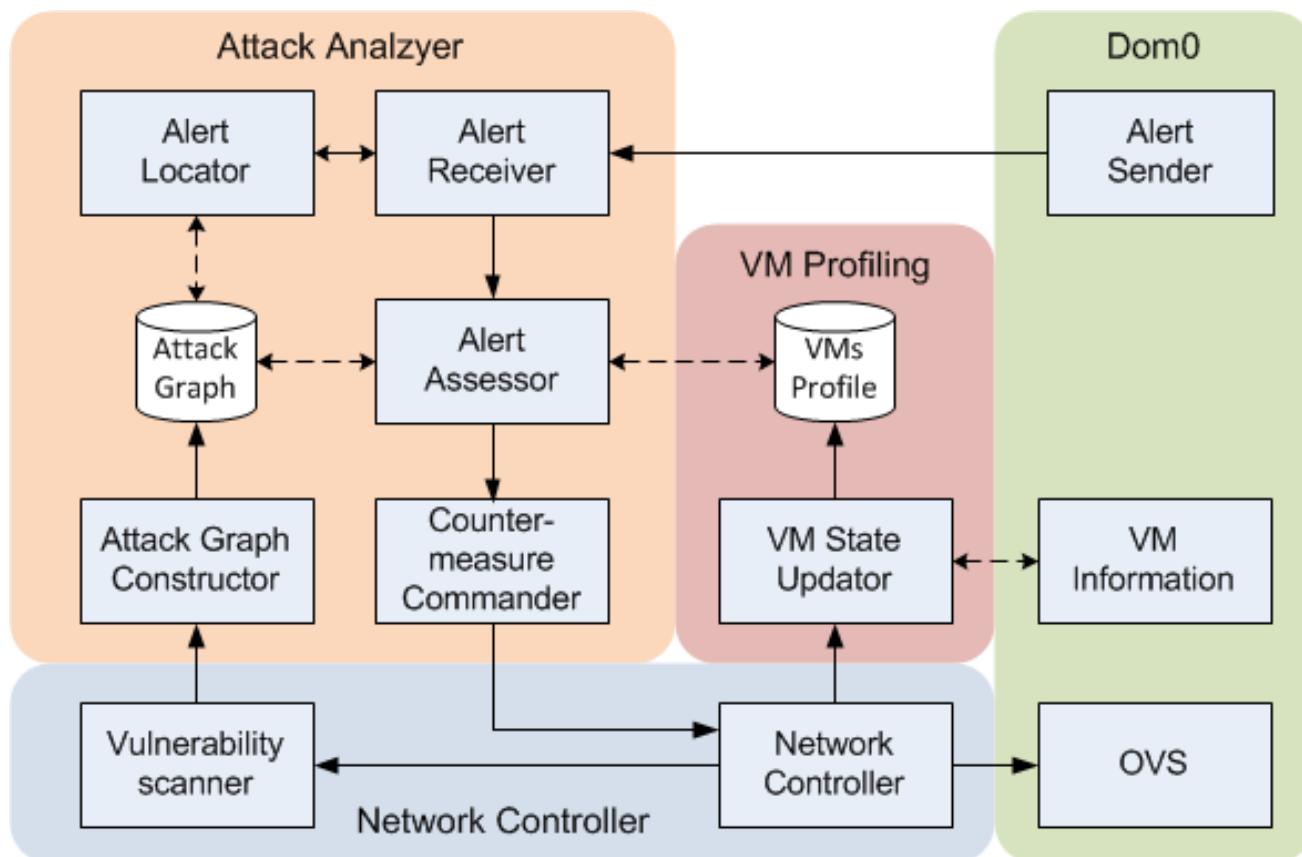
IPS Virtual Networking



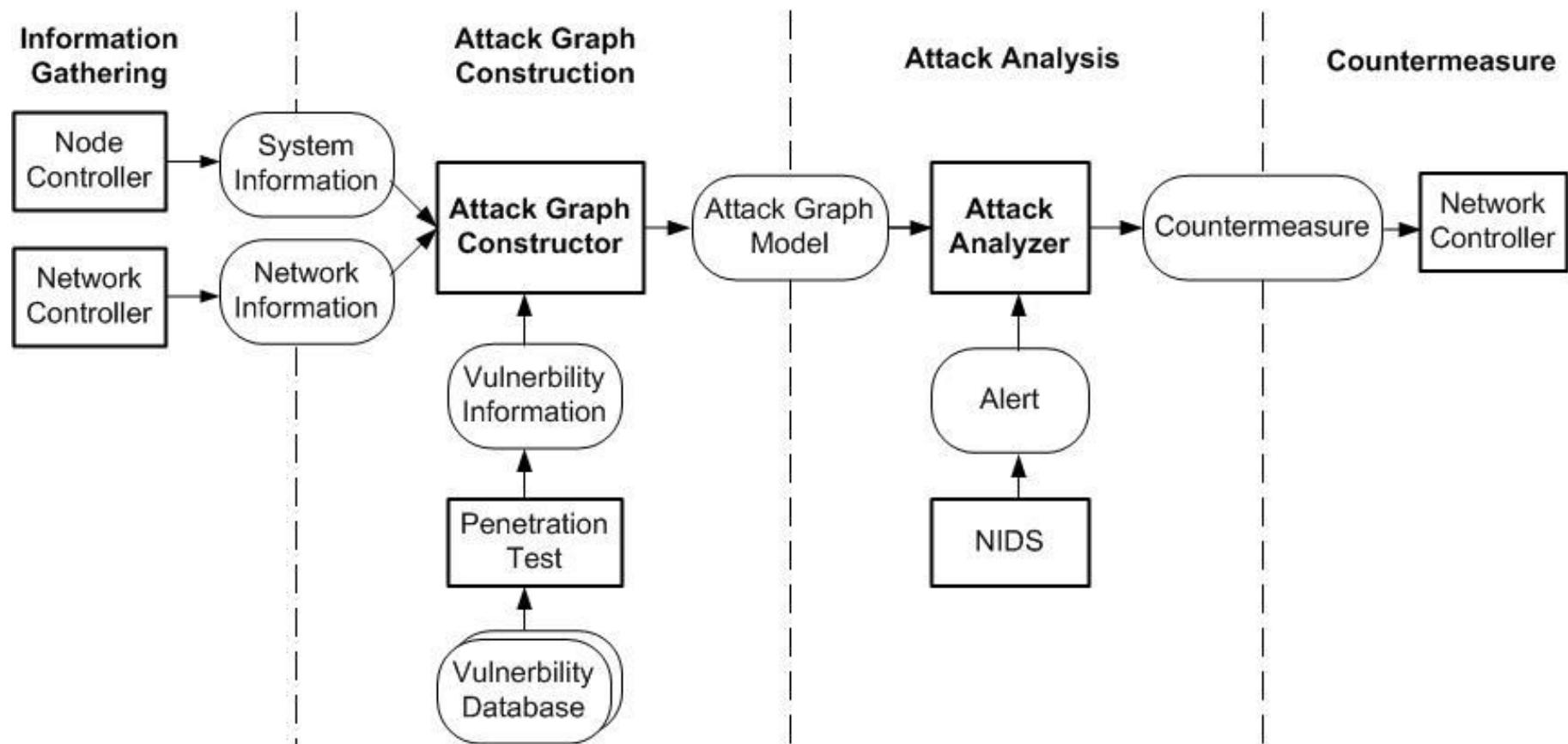
Mirroring-based IDS Workflow



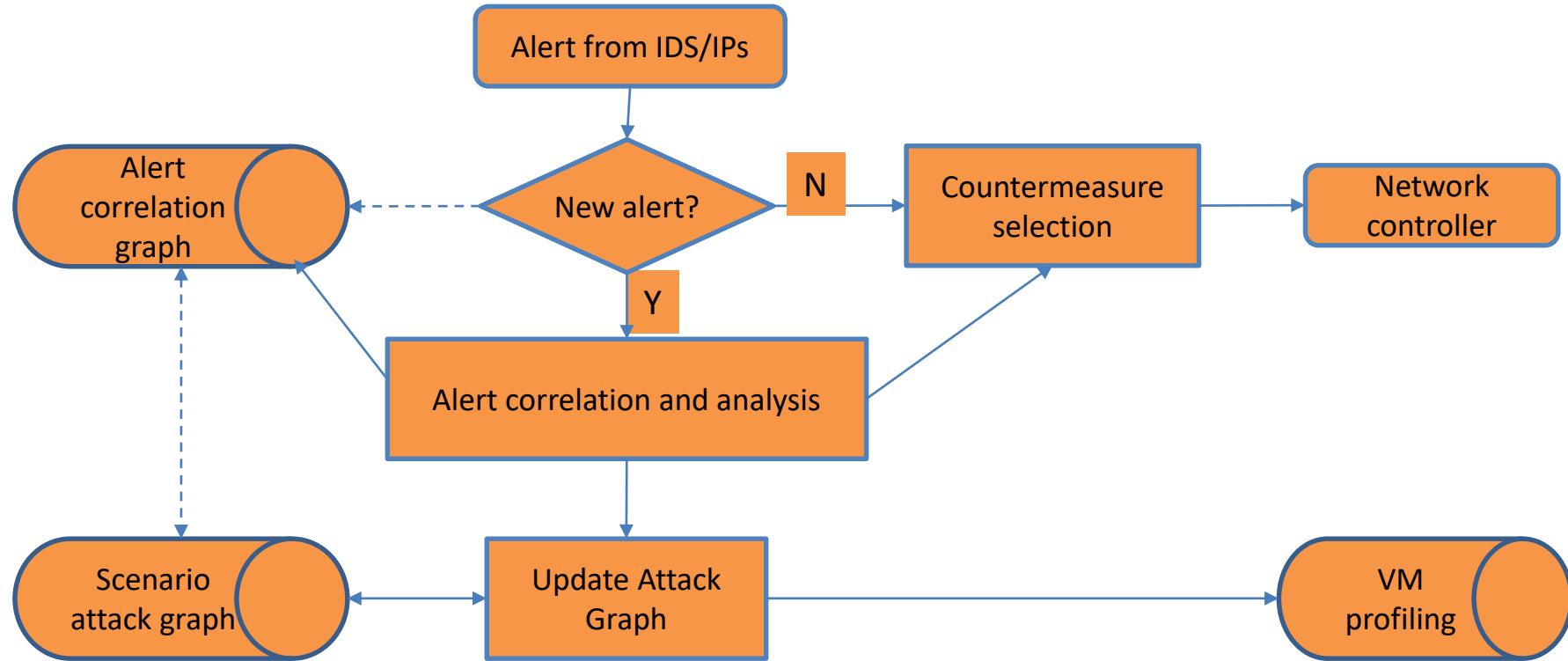
Security Analysis Modules



Attack Analyzer Workflow



Attack Events Handling

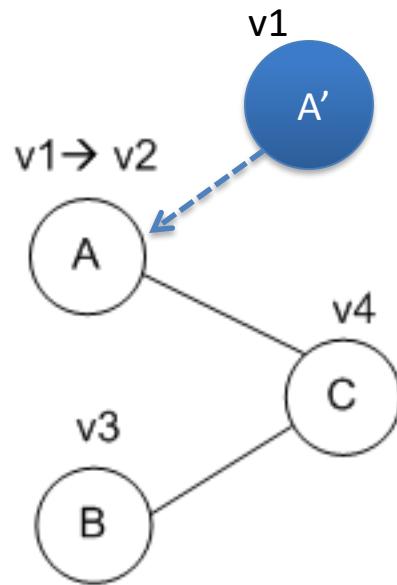




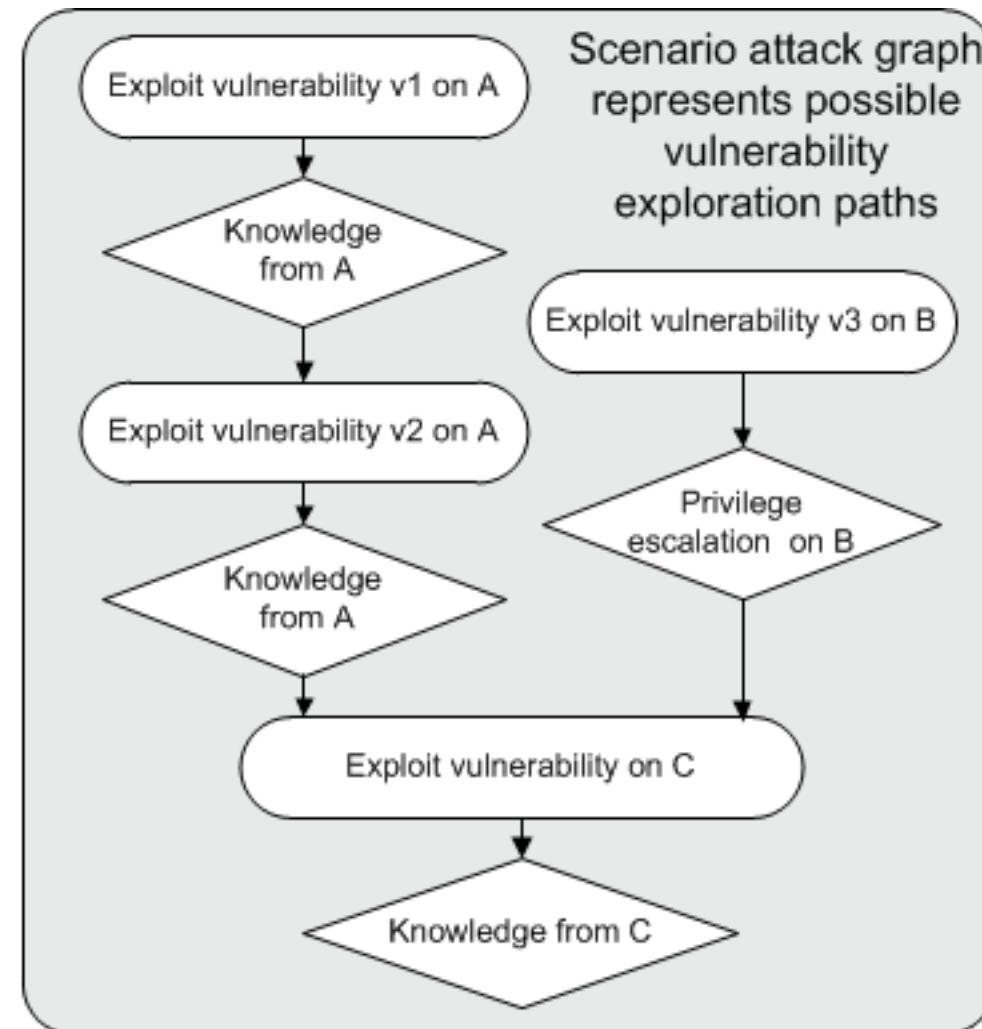
Attack Analyzer's Input

- System Information
 - Collected from node controller (Xenserver Dom0)
 - # of VMs, the status of each VM, running service on each VM, vif information
- Network Information
 - Collected from OpenFlow network controller (NOX)
 - Network topology, host connectivity, IP address, Mac address, Port information, Flow
- Vulnerability Information
 - Generated by penetration testing tools based on well-known vulnerability Database
 - Penetration testing tools: OVAL scanner, NESSUS, Metasploit and OpenVAS
- Vulnerability Database
 - Open Source Vulnerability Database (OSVDB), Common Vulnerabilities and Exposures List (CVE), National Vulnerability Database (NVD)

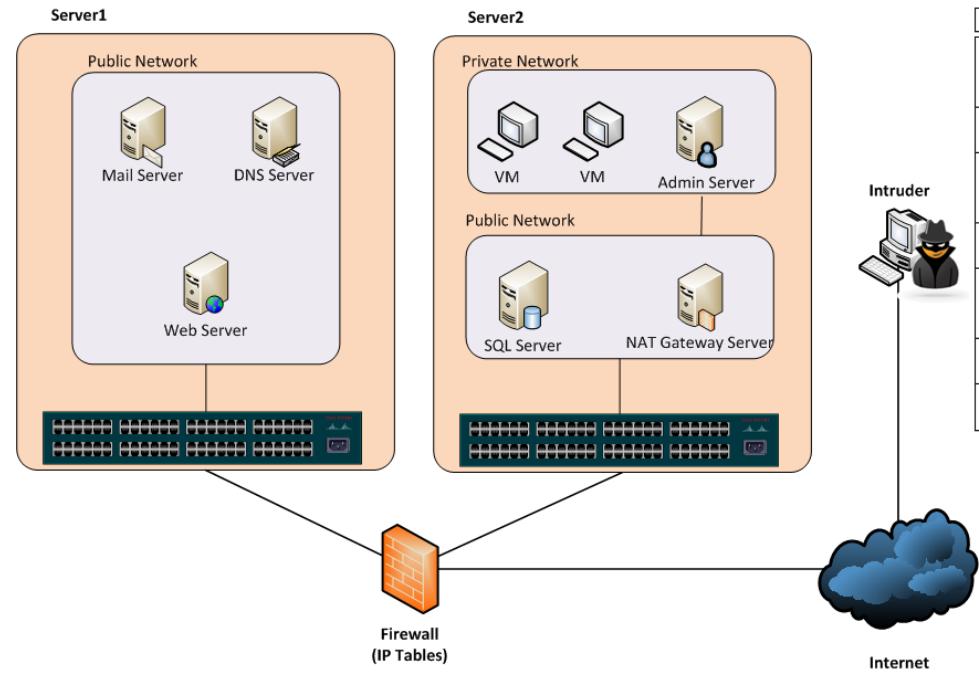
Attack Graph Model



Vulnerabilities on nodes A, B, and C for a simple network system.



An Example Network Topology



Host	Vulnerability	CVE#	Type of Attack
Private Network	Remote login LICQ Buffer Overflow (BOF) MS Video ActiveX Stack BOF	CA 1996-83 CVE 2001-0439 CVE 2009-0015	remote-2-user remote-2-user remote-2-root
Admin machine	MS SMV service Stack BOF	CVE 2008-4050	local-2-root
Gateway server	OpenSSL uses predictable random Heap corruption in OpenSSH Improper cookies handler in OpenSSH	CVE 2008-0166 CVE 2003-0693 CVE 2007-4752	information leakage local-2-root authentication bypass
SQL Server	SQL Injection	CVE 2008-5416	remote-2-root
Mail Server	Remote code execution in SMTP Error message information leakage Squid port scan vulnerability	CVE 2004-0840 CVE 2008-3060 CVE 2001-1030	remote-2-root account information theft information leakage
DNS Server	DNS Cache Poisoning	CVE 2008-1447	integrity
Web Server	IIS vulnerability in WebDAV service	CVE 2009-1535	remote-2-local authentication bypass

From Host	To Host	Protocol / Port#
0.0.0.0	Gateway Server	SSHD (22)
	Mail Server	IMAP (143)
	Web Server	HTTP (80)
Web Server	SQL Server	SQL (1433)
Gateway Server	Local Desktops	Basic Network protocols
Root machine	Root machine	Basic Network protocols
Local Desktops and Root-machine	Gateway Server	Basic Network protocols
	Mail Server	IMAP (143) SMTP (25)
	SQL Server	SQL (1433)
	Web Server	HTTP (80)
	DNS Server	DNS (53 & 1024)

Vulnerability Base Score and Attack Probability

Host	Vulnerability	Node	CVE	Base Score
VM group	LICQ buffer overflow	10	CVE 2001-0439	0.75
	MS Video ActiveX Stack buffer overflow	5	CVE 2008-0015	0.93
	GNU C Library loader flaw	22	CVE-2010-3847	0.69
Admin Server	MS SMV service Stack buffer overflow	2	CVE 2008-4050	0.93
Gateway server	OpenSSL uses predictable random variable	15	CVE 2008-0166	0.78
	Heap corruption in OpenSSH	4	CVE 2003-0693	1
	Improper cookies handler in OpenSSH	9	CVE 2007-4752	0.75
Mail server	Remote code execution in SMTP	21	CVE 2004-0840	1
	Squid port scan	19	CVE 2001-1030	0.75
Web server	WebDAV vulnerability in IIS	13	CVE 2009-1535	0.76

Base Score (BS) is defined from Common Vulnerability Scoring System (CVSS) <http://www.first.org/cvss/cvss-guide.html>

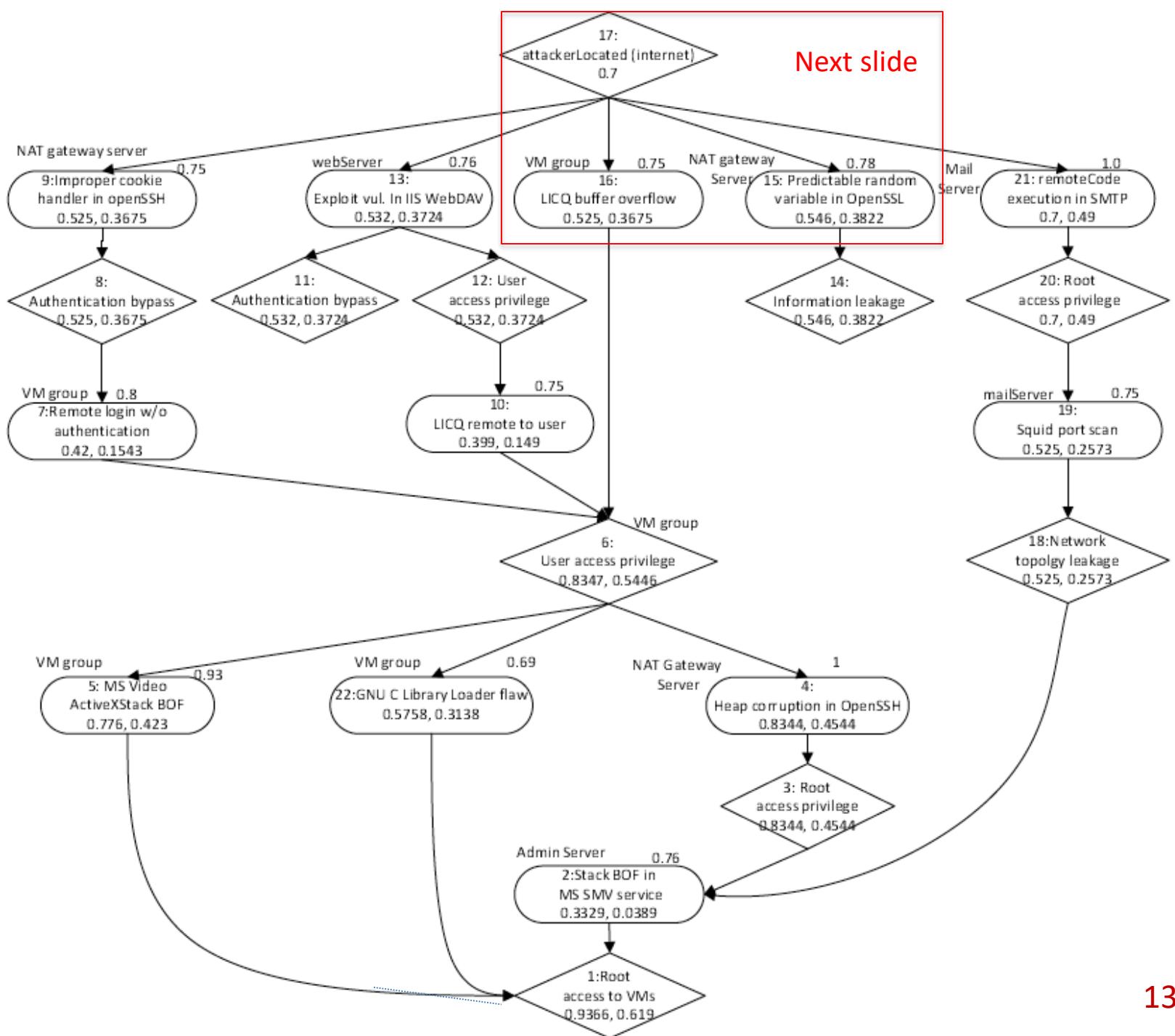
$$BS = (0.6 \times IV + 0.4 \times E - 1.5) \times f(IV)$$

where $IV = 10.41 \times (1 - (1 - C) \times (1 - I) \times (1 - A))$

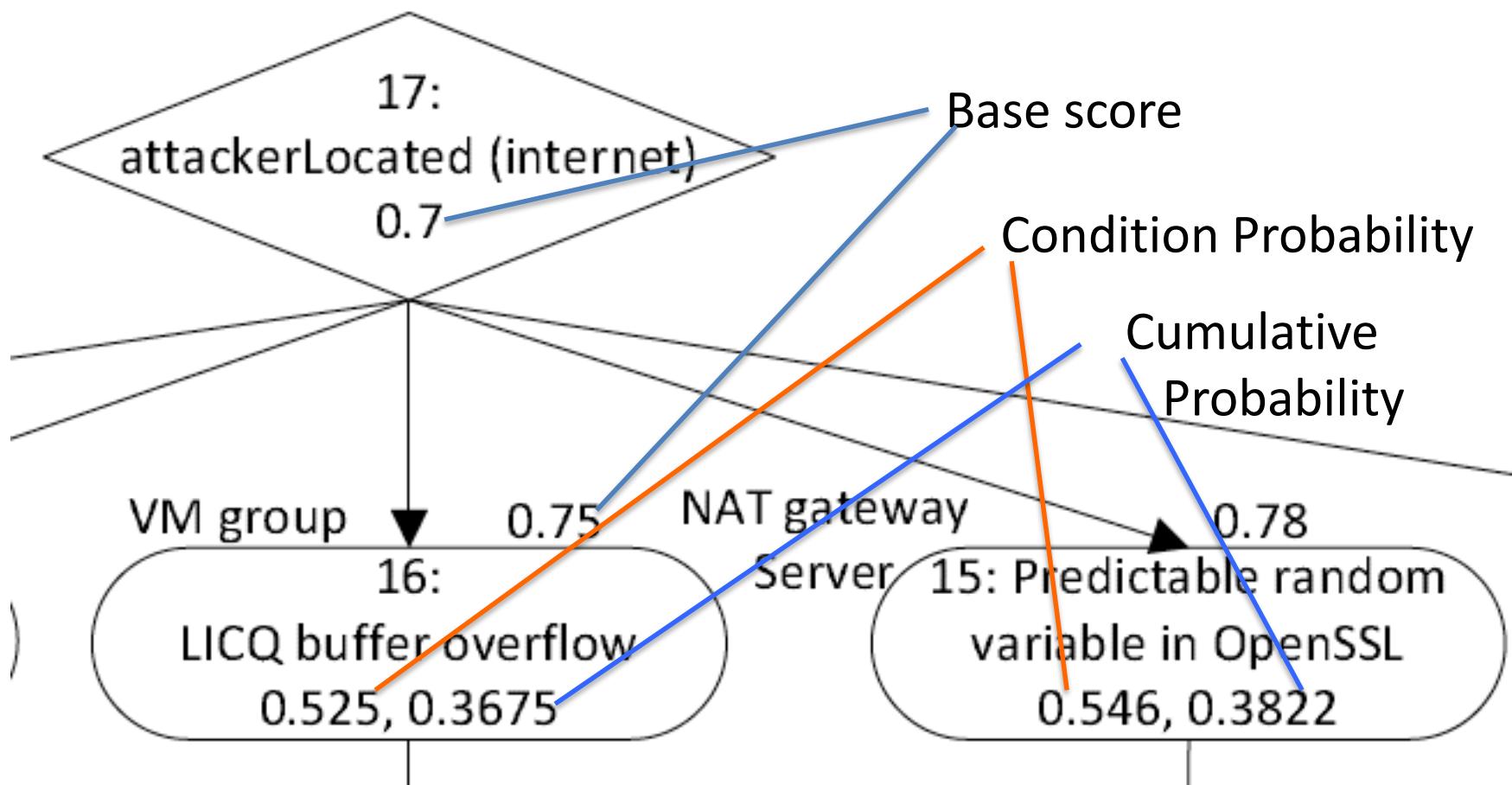
$$E = 20 \times AC \times AU \times AV,$$

$$f(IV) = \begin{cases} 0 & \text{if } IV = 0, \\ 1.176 & \text{otherwise.} \end{cases}$$

- The impact value (IV) is computed from three basic parameters of security namely
 - confidentiality (C),
 - Integrity (I), and
 - availability (A).
- The exploitability (E) score consists of
 - access vector (AV),
 - access complexity (AC), and
 - authentication instances (AU).



A Bayesian Analysis Approach



Countermeasure & Benefit Measurement

- Countermeasure *CM* is a four tuple <*cost, intrusiveness, conditions, effectiveness*>
 - *Cost* is the unit which describes the expenses required to apply the countermeasure and ranges from 1:least costly to 5:most costly.
 - *Intrusiveness* is the negative effect that the countermeasure will have on the services. It ranges from 1:least to 5:most intrusive.
 - *Condition* specifies when and where the countermeasure is applicable.
 - *Effectiveness* is the change in probability of the node to which the countermeasure is applied.
- *Benefit* is the percentage change in the probability of target node after countermeasure is applied.
 - $Benefit(CM) = \Delta Probability(targeting\ node)$

Examples of Network-based Countermeasures

#	Countermeasure	Intrusiveness	Cost
1	Traffic redirection	Attack Containment	3
2	Traffic isolation		2
3	Deep Packet Inspection		3
4	Filtering rules	Moving target	2
5	MAC address change		1
6	IP address change		1
7	Port Blocking		1
8	Software Patch		4
9	Quarantine	Dynamic change	2
10	Network Reconfiguration		5
11	Network topology change		5



Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

<http://snac.eas.asu.edu>

Evaluation Metric: Return of Investment (ROI)

- The countermeasure with the highest value of the Return of Investment (ROI) is the best countermeasure to counter the attack event. The ROI of a countermeasure cm applying on VM k is defined as:

$$ROI_{cm}^k = \frac{benefit(k, cm)}{cm.cost + cm.intrusiveness},$$

where $benefit(k, cm)$ is for VM k after applying the cm , $cm.cost$ is the cost to apply the cm , and $cm.intrusiveness$ means the intrusive level of the cm .

- The optimal countermeasure selection for VM:

$$\max_{\forall cm} ROI_{cm}^k,$$

Algorithm for Countermeasure Selection (no Host Inspection)

Algorithm 2 Countermeasure_Selection

Require: *Alert*, $G(E, V)$, CM

```

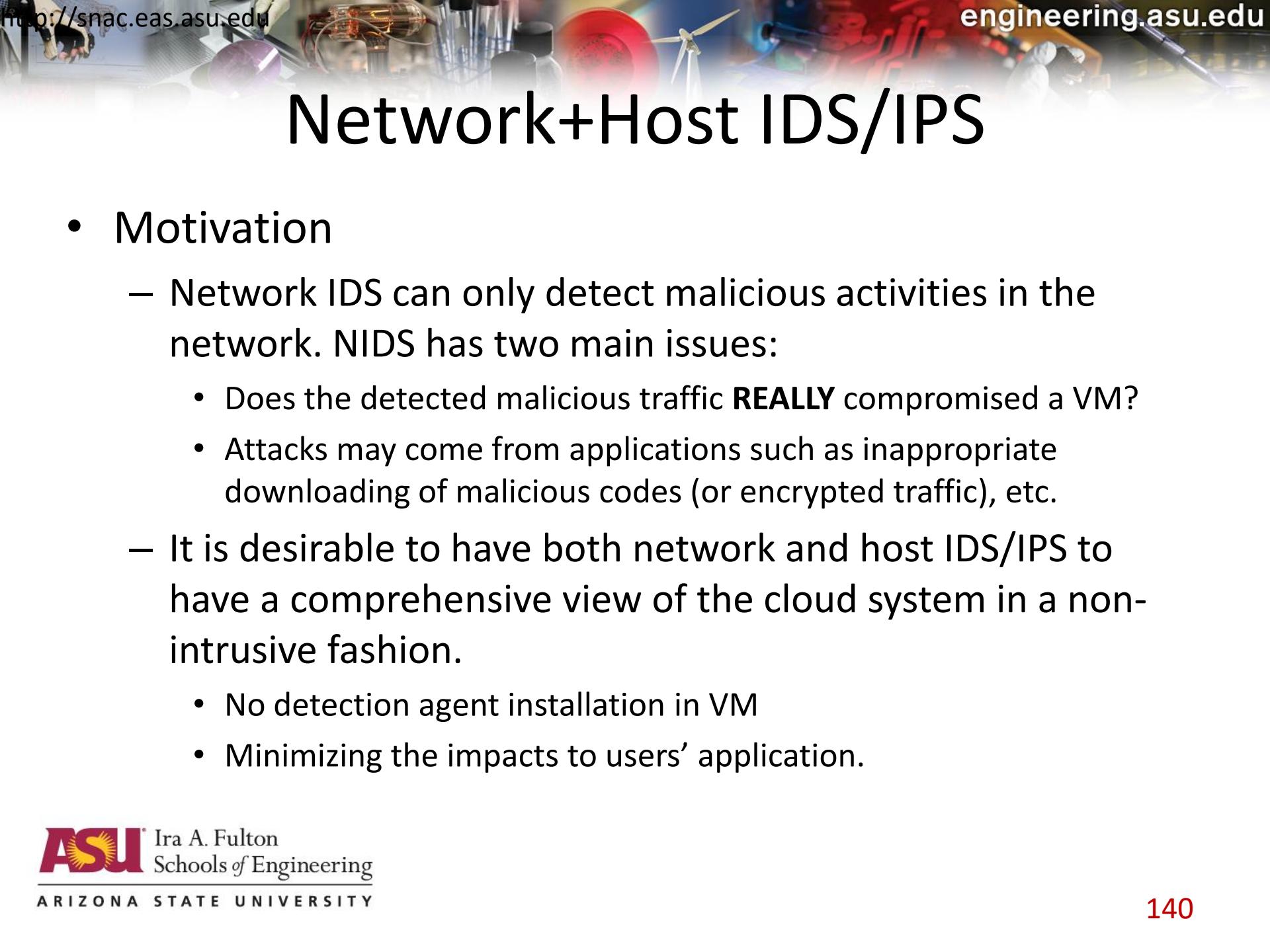
1: Let  $v_{Alert}$  = Source node of the Alert
2: if  $\text{Distance\_to\_target}(v_{Alert}) < \text{threshold}$  then
3:   Update_ACG
4:   return
5: end if
6: Let  $T = \text{Descendant}(v_{Alert}) \cup v_{Alert}$ 
7: Set  $Pr(v_{Alert}) = 1$ 
8: calculate_risk_prob( $T$ )
9: Let  $benefit[|T|, |CM|] = \emptyset$ 
10: for each  $t \in T$  do
11:   for each  $cm \in CM$  do
12:     if  $cm$  is applicable to  $t$  then
13:        $Pr(t) = Pr(t) * (1 - cm.\text{effectiveness})$ 
14:       calculate_risk_prob( $\text{Descendant}(t)$ )
15:
16:        $benefit[t, cm] = \Delta Pr(\text{target\_node}). \quad (7)$ 
17:     end if
18:   end for
19: end for

```

```

20: for each  $t \in T$  do
21:   for each  $cm \in CM$  do
22:
23:    $ROI[t, cm] = \frac{benefit[t, cm]}{cost(cm) + intrusive(cm)}.$  (8)
24: end for
25: end for
26: Update_SAG and Update_ACG
27: return Select_Optimal_CM( $ROI$ )

```

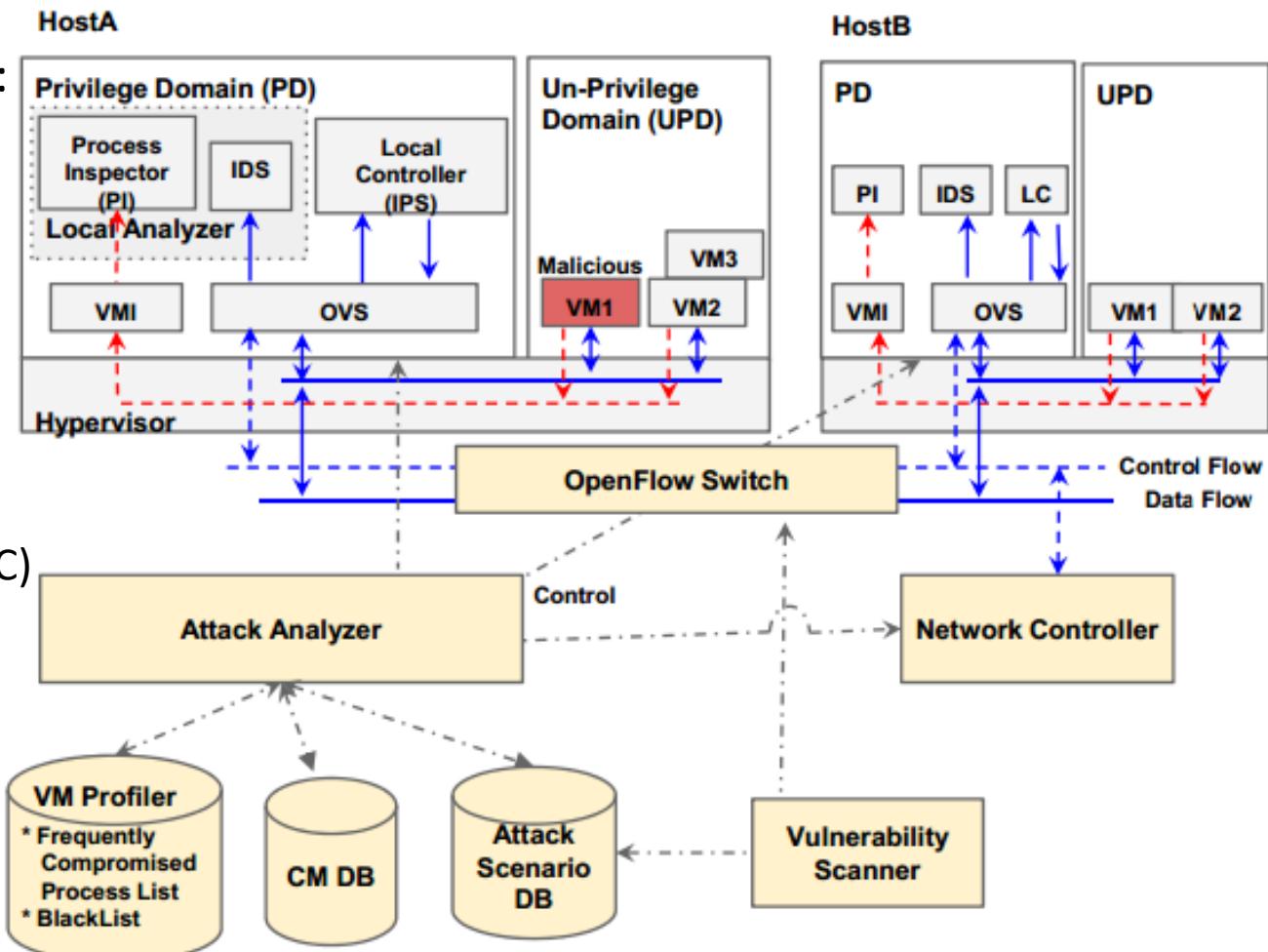


Network+Host IDS/IPS

- Motivation
 - Network IDS can only detect malicious activities in the network. NIDS has two main issues:
 - Does the detected malicious traffic **REALLY** compromised a VM?
 - Attacks may come from applications such as inappropriate downloading of malicious codes (or encrypted traffic), etc.
 - It is desirable to have both network and host IDS/IPS to have a comprehensive view of the cloud system in a non-intrusive fashion.
 - No detection agent installation in VM
 - Minimizing the impacts to users' application.

Non-intrusive Network+Host IDS/IPS

- Local components in PD:
 - Local Analyzer (LA)
 - Process Inspector (PI)
 - Local Controller (LC)
- System components
 - Attack Analyzer (AA)
 - Network Controller (NC)
 - Attack Scenario DB
 - Countermeasure DB
 - VM Profiler



VM Security Index (VSI)

- VSI represents security healthiness of a VM. It is measured by three metrics:
 - Vulnerability (V), Exploitability (E), and Health status of processes (H)
- The VSI for a VM k defined as:

$$VSI_k = \alpha \times V_k + \beta \times E_k + \gamma \times H_k,$$

where α , β and γ are weights for each parameter respectively with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0$

- Higher VSI measure means easier to be attacked.

VSI - Vulnerability

- V_k is the vulnerability score for VM k . It is measured by the exponential average of the base score from each vulnerability of the VM with the maximum value of 10:

$$V_k = \min\{10, \ln \sum e^{BaseScore(v)}\}.$$

- Vulnerability score considers the base scores of all the vulnerabilities on a VM.
 - A base score depicts how easy it is for an attacker to exploit the vulnerability and how much damage it may incur.
- The exponential addition of base scores allows the vulnerability score to incline towards higher base score values and increases in logarithm-scale based on the number of vulnerabilities.

VSI - Exploitability

- E_k is exploitability score for VM k . It is measured by the average of exploitability score with the maximum value of 10:

$$E_k = \min\{10, \ln \sum e^{ES(v) \times \frac{S_k}{S_k + NS_k} + ES(v') \times \frac{NS_k}{S_k + NS_k}}\},$$

- $ES(v)$: the exploitability score of vulnerability v in VM k ,
- $ES(v')$: the exploitability score of vulnerability v' from other systems that the VM k is able to access.
- S_k : the number of services provided by VM k .
- NS_k : the number of network services the VM k can access.
- The exploitability score shows the accessibility of a target VM, and depends on the ratio of the number of services to the number of network services.
- Higher exploitability score means that there are many possible paths for the attacker to reach the target

VSI – Health Status of Processes

- H_k is the health status of processes in VM k . Higher value of H_k means more malicious processes in action and more read operation in VM k .
 - E.g., in windows VMs, H_k can be decided by the difference between two process lists: the list from process view (pl_p) and the list from the thread view (pl_t).
 - If $pl_t(k) = pl_p(k)$, which means no hidden process exists, we set $H_k = 0$.
 - Currently, we only inspect malicious process into directions:
 - Identify hidden process
 - Monitor processes dependency to identify which process is triggered by the network detected malicious traffic
 - Identify next step attack process or network activities (e.g., open ports/sockets).
 - System call level monitor and inspection is needed for future research.

VSI – Health Status of Processes

- In the case of $pl_t(k) \neq pl_p(k)$, which means some hidden processes exist, we set H_k :

$$H_k = (1 - \prod_{p \in pl_t(k) - pl_p(k)} (1 - \frac{cpu(k, p, t)}{t}) \times e^{-read(k, p)}) \times 10,$$

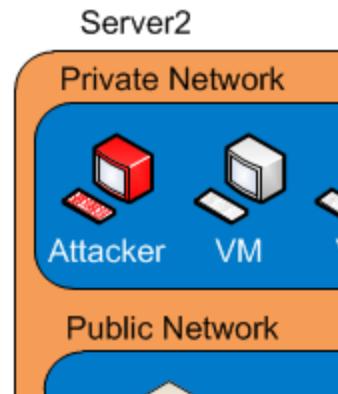
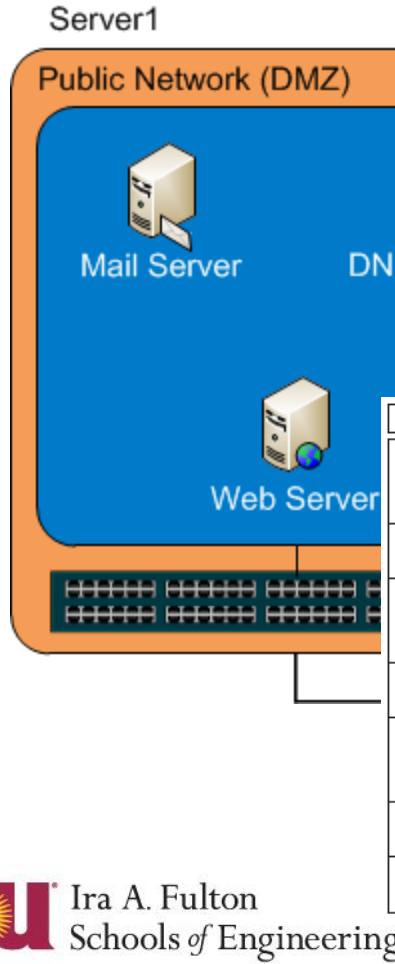
- plp(k): the process linked list retrieved from the EPROCESS chain (i.e., a process chain in Windows) in VM k.
- plt(k): the process linked list retrieved from the process information in each thread in VM k.
- read(k, p): the value represents the count of reading operations performed by the process p in VM k.
- cpu(k, p, t): the elapsed time of process p in action in the system of VM k during the measurement time period of t seconds.

Countermeasure Selection (with Host Inspection)

- *Benefit* is then evaluated as follows:
 - $Benefit(CM) = \Delta VSI(\text{targeting node})$
 - i.e., the VSI measurement deduced by applying the countermeasure CM .
- The ROI Evaluation remains the same

$$ROI_{cm}^k = \frac{\text{benefit}(k, cm)}{cm.\text{cost} + cm.\text{intrusiveness}},$$

A Testing Scenario Setup



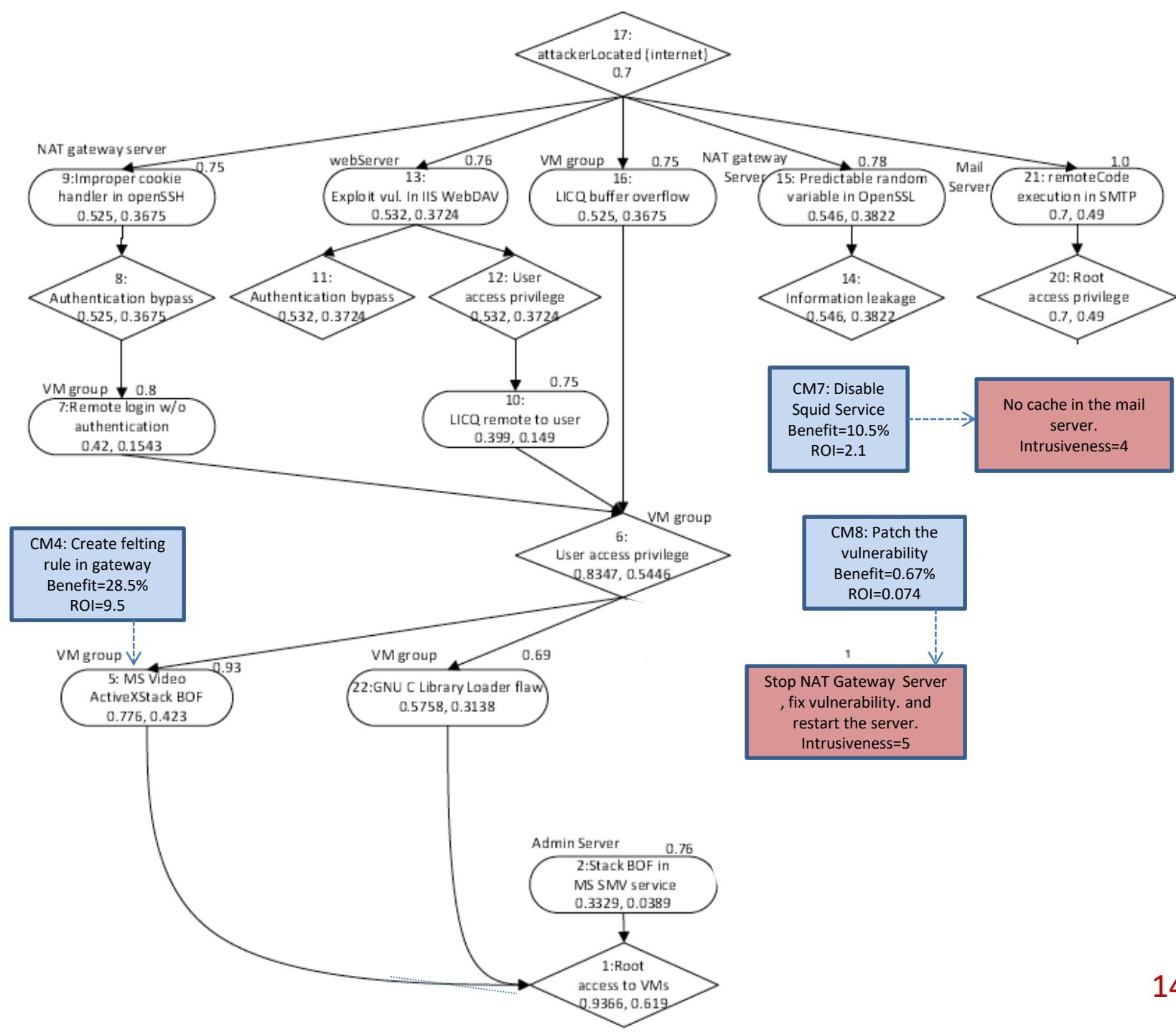
From Host	To Host	Protocol / Port#
0.0.0.0	Gateway Server	SSHD (22)
	Mail Server	IMAP (143) SMTP (25)
	Web Server	HTTP (80)
Web Server	SQL Server	SQL (1433)
Gateway Server	Local Desktops	Basic Network protocols
	Root machine	Basic Network protocols
Local Desktops and Root-machine	Gateway Server	Basic Network protocols
	Mail Server	IMAP (143) SMTP (25)
	SQL Server	SQL (1433)
	Web Server	HTTP (80)
	DNS Server	DNS (53 & 1024)

Host	Vulnerability	CVE#	Type of Attack
Private Network	Remote login LICQ Buffer Overflow (BOF) MS Video ActiveX Stack BOF	CA 1996-83 CVE 2001-0439 CVE 2009-0015	remote-2-user remote-2-user remote-2-root
Admin machine	MS SMV service Stack BOF	CVE 2008-4050	local-2-root
Gateway server	OpenSSL uses predictable random Heap corruption in OpenSSH Improper cookies handler in OpenSSH	CVE 2008-0166 CVE 2003-0693 CVE 2007-4752	information leakage local-2-root authentication bypass
SQL Server	SQL Injection	CVE 2008-5416	remote-2-root
Mail Server	Remote code execution in SMTP Error message information leakage Squid port scan vulnerability	CVE 2004-0840 CVE 2008-3060 CVE 2001-1030	remote-2-root account information theft information leakage
DNS Server	DNS Cache Poisoning	CVE 2008-1447	integrity
Web Server	IIS vulnerability in WebDAV service	CVE 2009-1535	remote-2-local authentication bypass

wall
ables)



Ira A. Fulton
Schools of Engineering



Network Countermeasures

- CM1: Traffic redirection (intrusiveness: 3, cost: 3)
- CM2: Traffic isolation (intrusiveness: 4, cost: 2)
- CM3: Deep Packet Inspection (DPI) (intrusiveness: 3, cost: 3)
- CM4: Create filtering rules (intrusiveness: 1, cost: 2)
- CM5: MAC address change (intrusiveness: 2, cost: 1)
- CM6: IP address change (intrusiveness: 2, cost: 1)
- CM7: Block port (intrusiveness: 4, cost: 1)
- CM8: Software Patch (intrusiveness: 5, cost: 4)
- CM9: Quarantine (intrusiveness: 5, cost: 2)
- CM10: Network Reconfiguration (intrusiveness: 1, cost: 5)
- CM11: Network Topology Change (intrusiveness: 1, cost: 5)
-

Result of Algorithm (no host inspection)

<i>Benefit (%)</i>	Node2	Node4	Node5	Node29
CM1	0.447	0.447	26.82	10.5
CM2	0	0	30.17	11.82
CM3	0.523	0.523	25.15	9.85
CM4	0.852	0.852	28.5	11.16
CM5	0.447	0	26.82	10.5
CM6	0.6	0	26.82	10.5
CM7	0	0	26.82	10.5
CM8	0.67	0.67	30.17	11.82
CM9	0	0	26.82	10.5
CM10	0.523	0.523	23.47	9.19
CM11	0.523	0.523	23.47	9.19

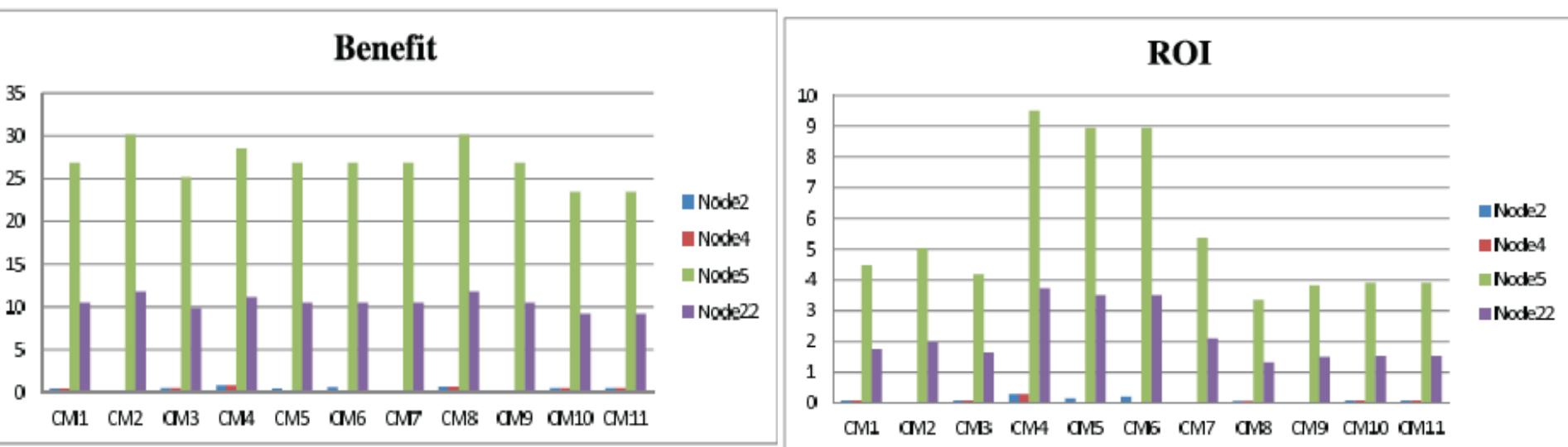
Table showing the *benefit* values of various countermeasures given $v_{Alert}=16$.

<i>ROI</i>	Node2	Node4	Node5	Node29
CM1	0.075	0.075	4.47	1.75
CM2	0	0	5.028	1.97
CM3	0.087	0.087	4.192	1.642
CM4	0.284	0.284	9.5	3.72
CM5	0.149	0	8.94	3.5
CM6	0.2	0	8.94	3.5
CM7	0	0	5.364	2.1
CM8	0.074	0.074	3.352	1.313
CM9	0	0	3.831	1.5
CM10	0.087	0.087	3.912	1.532
CM11	0.087	0.087	3.912	1.532

Table showing the *ROI (return of Investment)* values of various countermeasures

$$ROI(CM4_5) = \frac{benefit(CM4_5)}{cost(CM4) + intrusiveness(CM4)} = \frac{28.5}{1 + 2} = 9.5$$

Experiment Result (no host inspection)

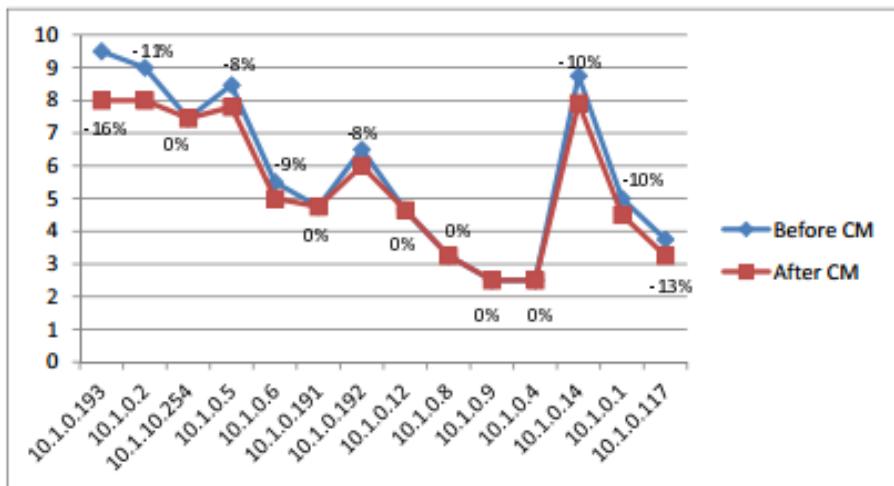
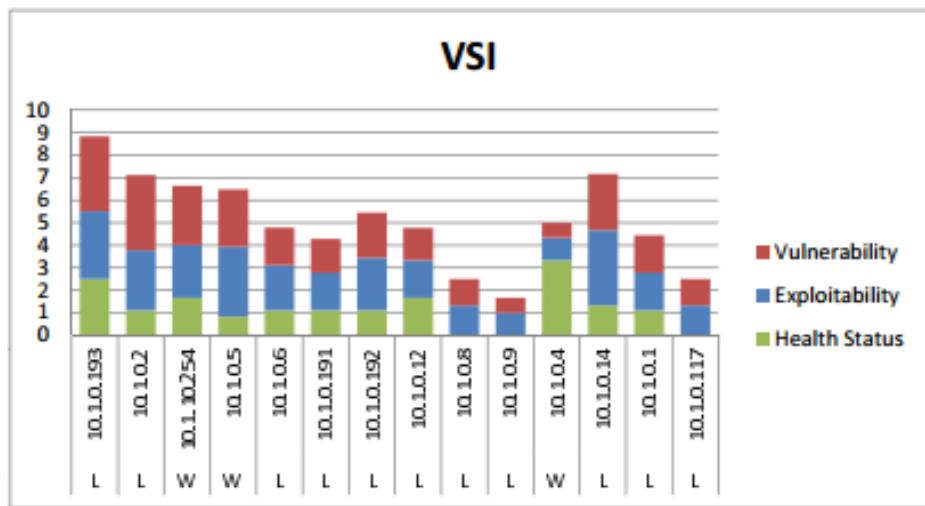


The result shows that CM2 and CM8 on node 5 have the maximum benefit, but their cost and impact scores indicate that they might not be good candidates for the optimal countermeasure and ROI graph confirms this.

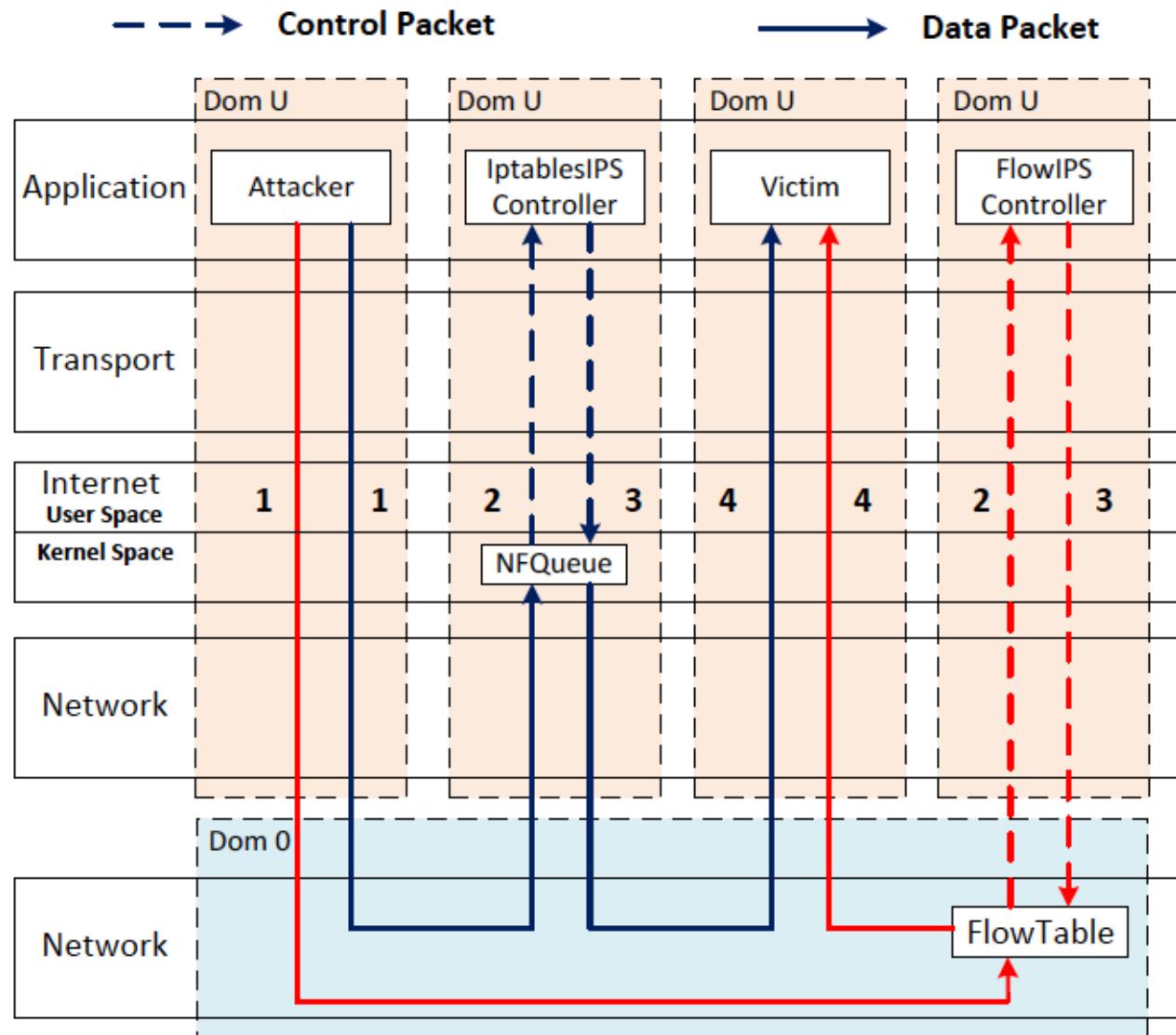
The ROI graph shows that CM4 on node 5 is the optimal solution.

Security Evaluation (with host inspection)

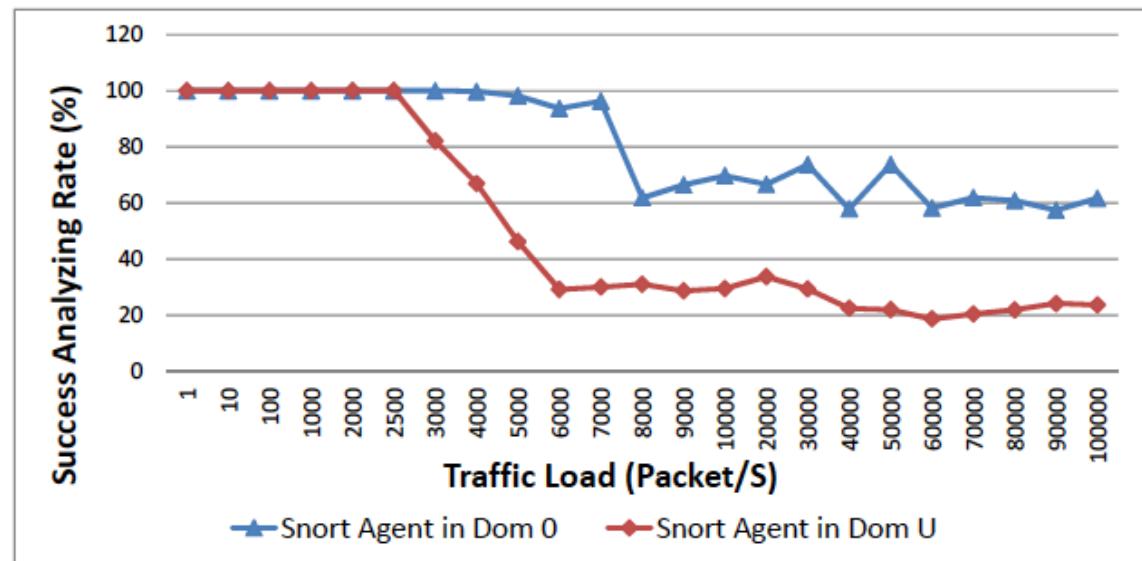
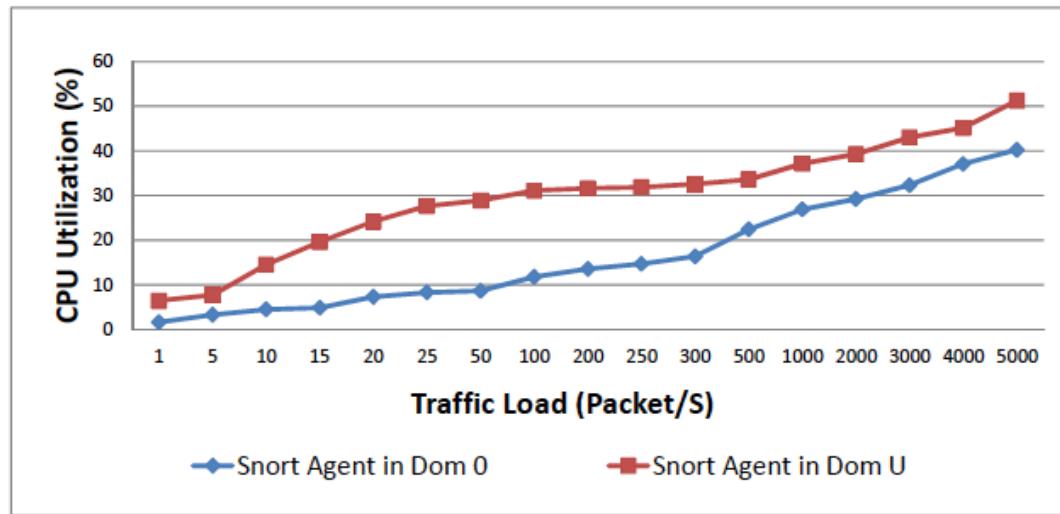
- The figure on the bottom left shows the plotting of VSI for these virtual machines before countermeasure selection and application.
- The figure on the bottom right compares VSI values before and after applying the countermeasure an optimal countermeasure.



System Performance Evolution: SDN+Snort IPS vs Iptables+Snort IPS



Dom0 detection agent is preferred

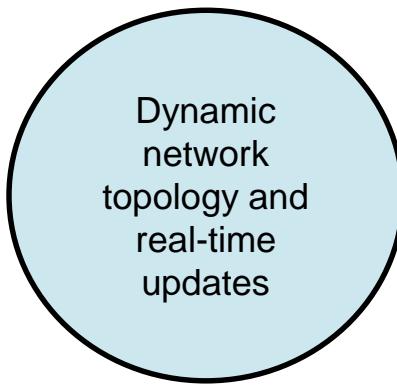
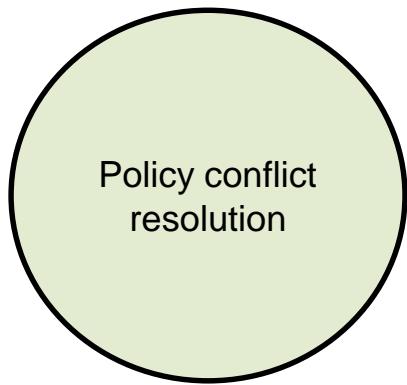
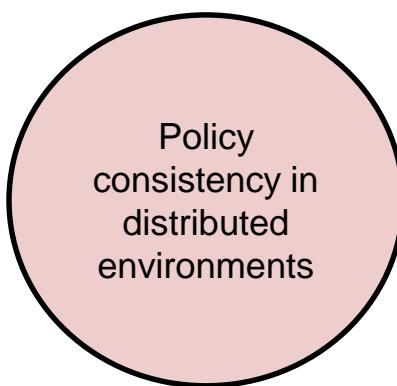
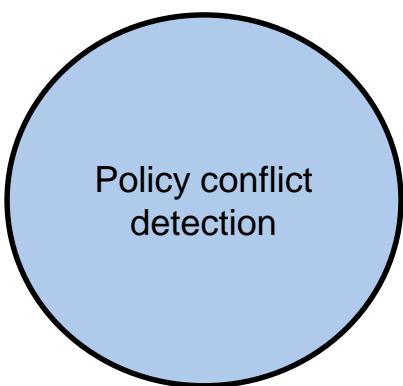


SDN-based security policy management

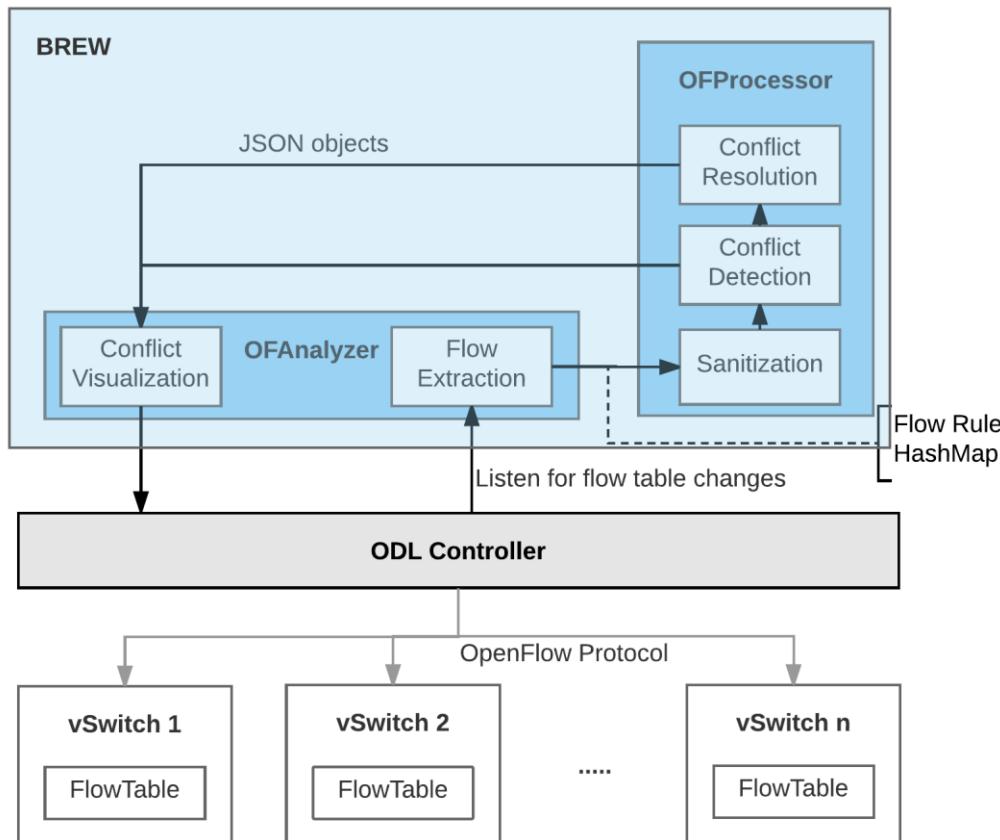
Flow Rule Conflicts

- More complex than traditional firewalls
 - More layers to consider
 - Set-field actions
 - Wildcard matching on rules
 - Networks and subnets have no meaning
 - 192.202.5.50/24 and 192.168.5.1/24 both match 192.*.5.1/24
 - Cross-layer interaction

Problem Space

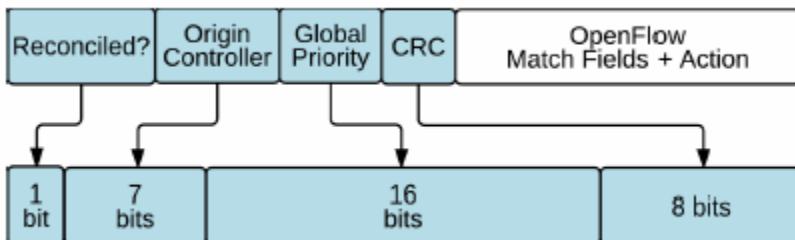


System Design

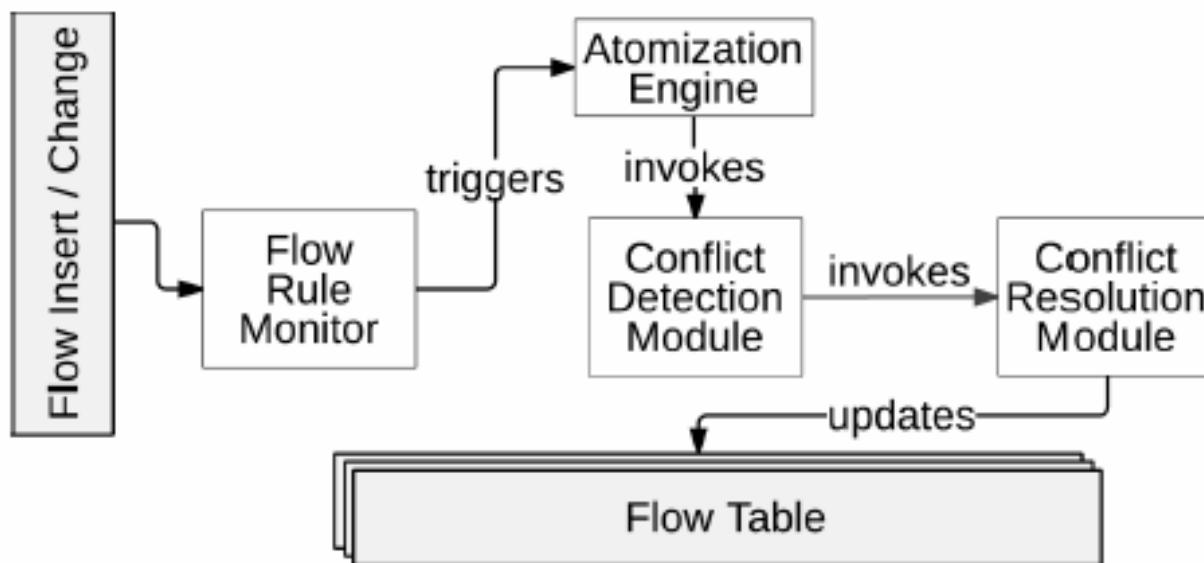


System Modules

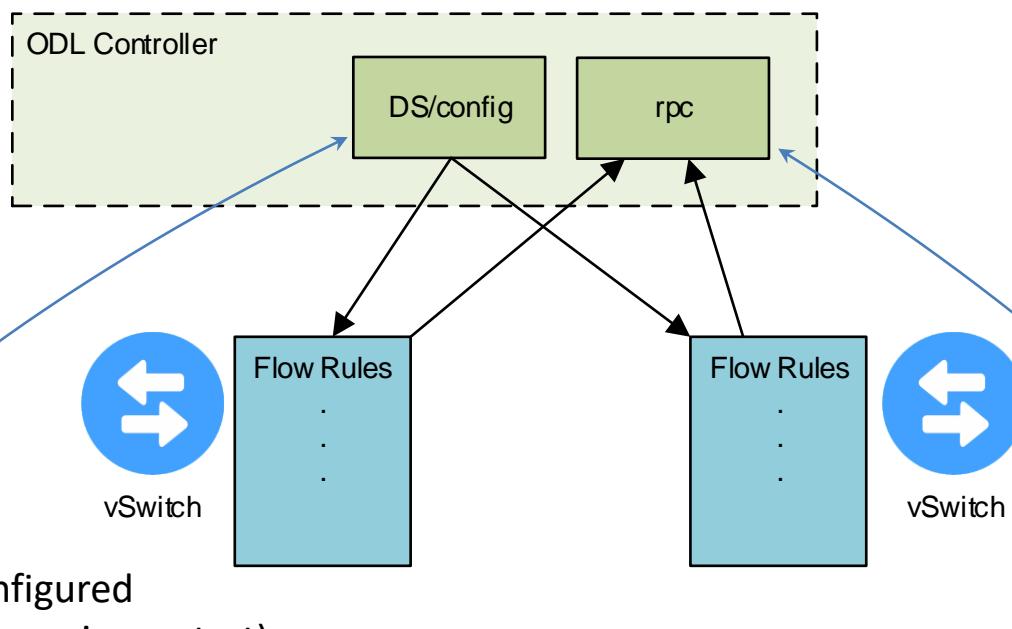
- Flow Rule Monitor
 - Intercepts candidate flow rule that is to be added to the flow table.
- Atomization Engine
 - Convert to atomic actions



Brew System



Flow Rule Monitor



- flows can be preconfigured
- can be stored (thus survive restart)
- no support for bulk operations based on masks
- unable to accurately determine if push succeeded
- can make use of bulk operations based on mask
- can *eventually* read flow on device which comes from DS/config
- returns response
- ability to delete whole table in one step!

Flow Rule Injection

- Policy based injection
 - Manual injection
 - Depending on system conditions
 - Energy/environmental factors
- To implement appliance functionality
- As countermeasure to attacks
- Adversarial injection

Sanitization

- Resolve all rules recursively to atomic actions
 - Terminal actions of Permit/Drop/QoS
- In case of action sets
 - Insert additional flow rule for each action
- Reconciliation of rules
 - Currently doing 1-1 mapping
 - Need to consider
 - Multiple tenant situation
 - Wildcard rules involving L2

Conflict Detection

- Host Partitioning
 - Host split
 - Local rules are preferred
- Hierarchical Controllers
 - Host + Application split
 - Local (or leaf) controllers have the lowest priority

Conflict Detection

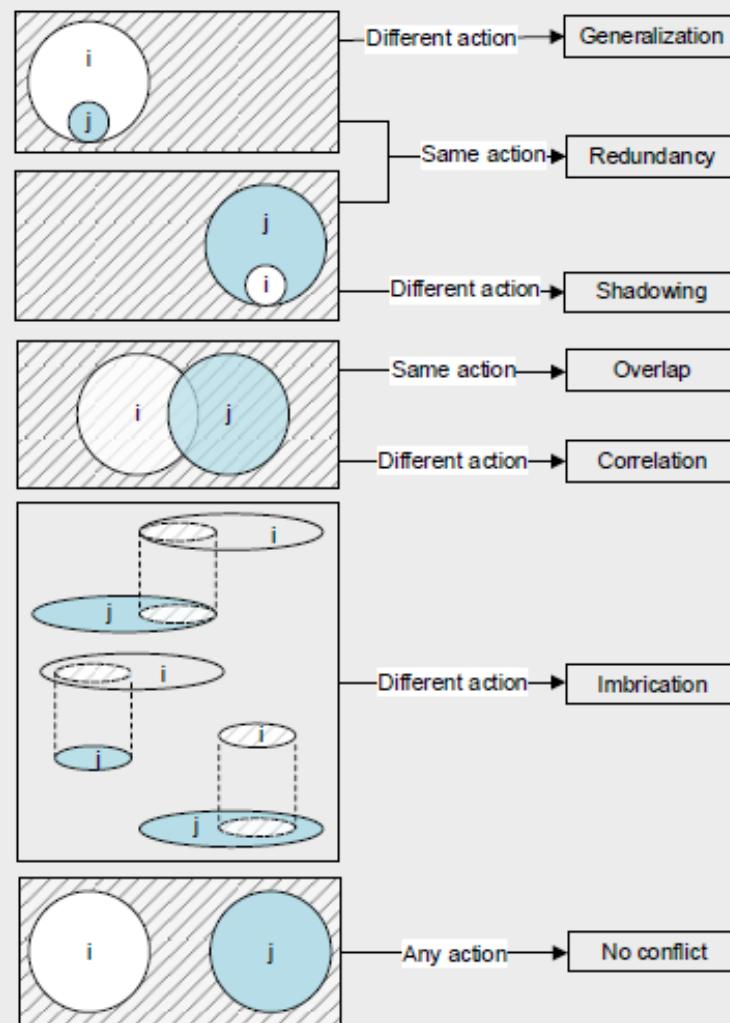
- Application Partitioning
 - Application split
 - Weighted priorities depending on generating application
 - Security
 - Reliability
 - Manual ordering

Conflict Detection

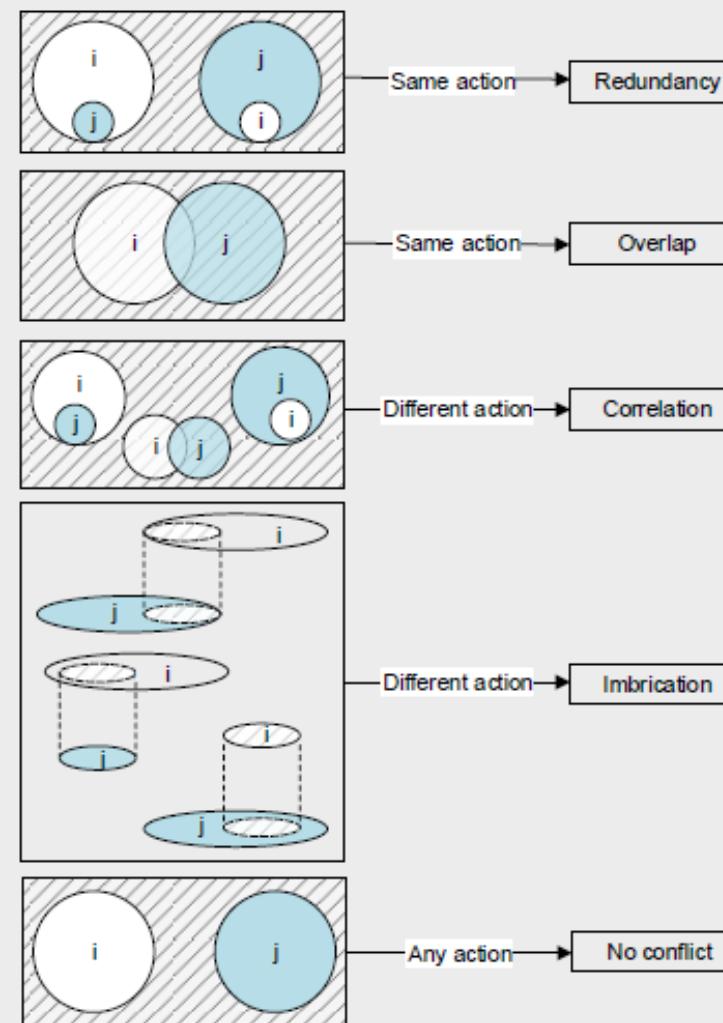
- Decentralization strategies
 - Hierarchical deployment
 - Split applications into local/global
 - Each controller runs a specific application
- Global priority depends on strategy

Conflict Classification

Priority of Rule i > Priority of Rule j



Priority of Rule i = Priority of Rule j



Key:



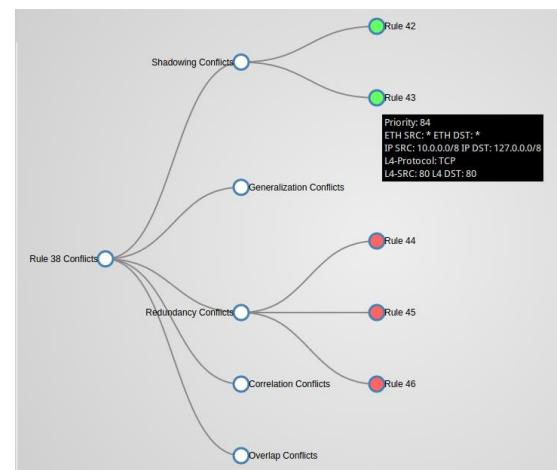
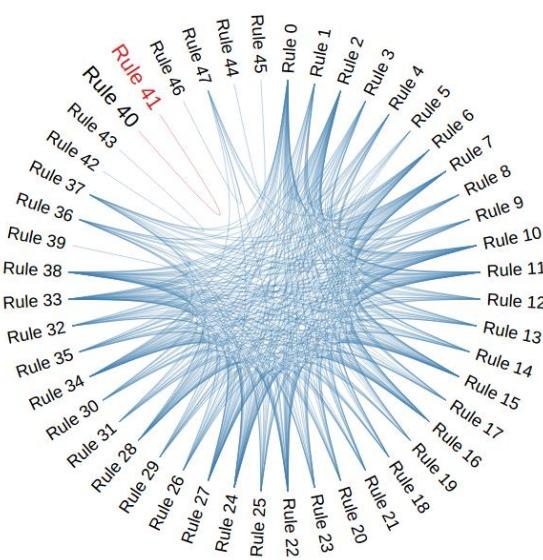
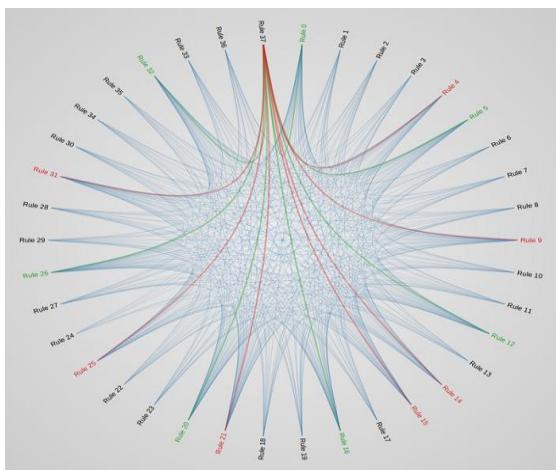
Conflict Resolution Strategies

- Conflict Resolution
 - Intelligible conflict
 - Resolved without loss of information
 - Interpretative conflict
 - Lossy resolution
 - Use global priorities

Conflict Resolution Strategies

- Assign global priorities
 - Least privilege
 - Module security precedence
 - Static policy enforcement
 - Tenant policy enforcement
 - Others
 - Software reliability
 - Environment calibrated
 - Administrator assistance

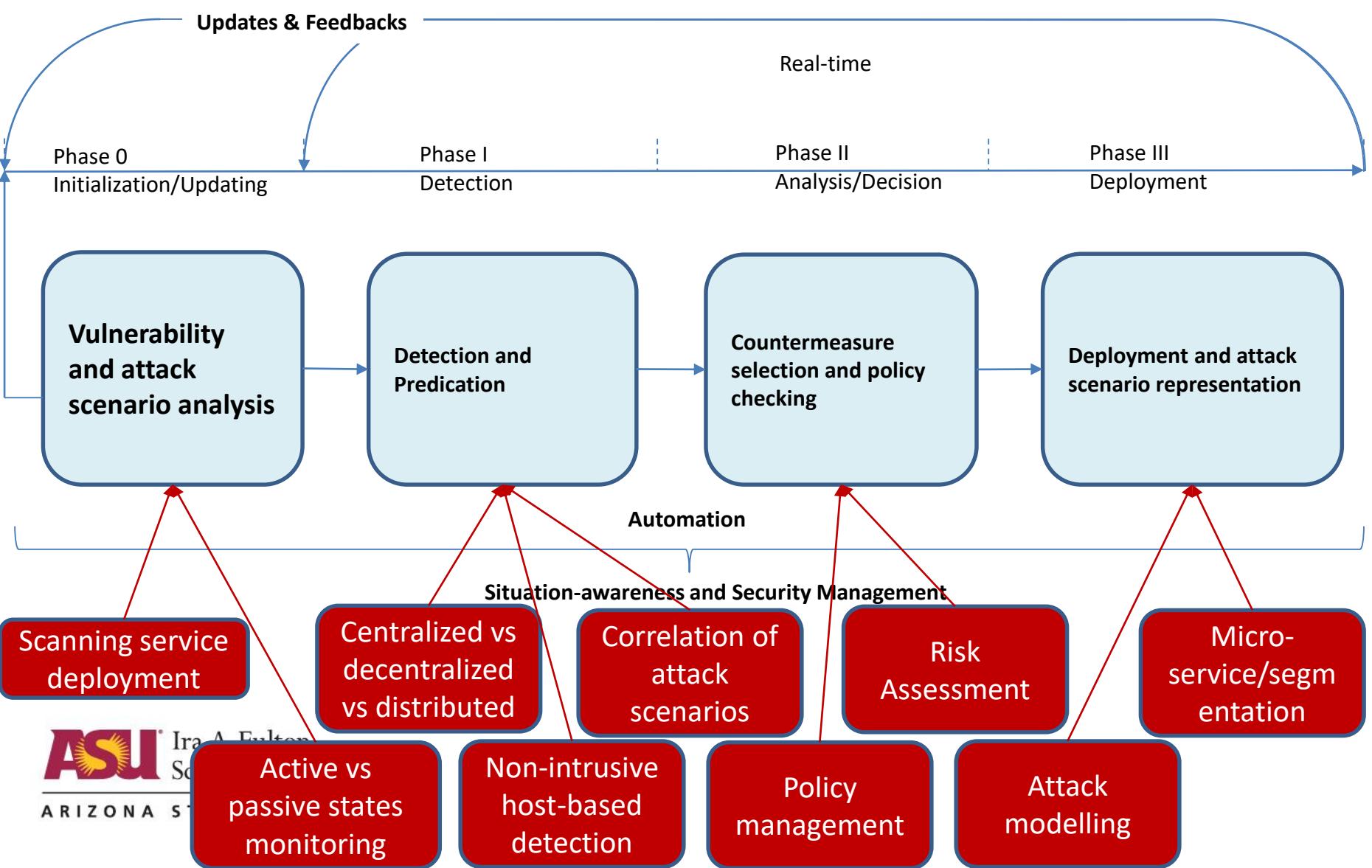
Visualization



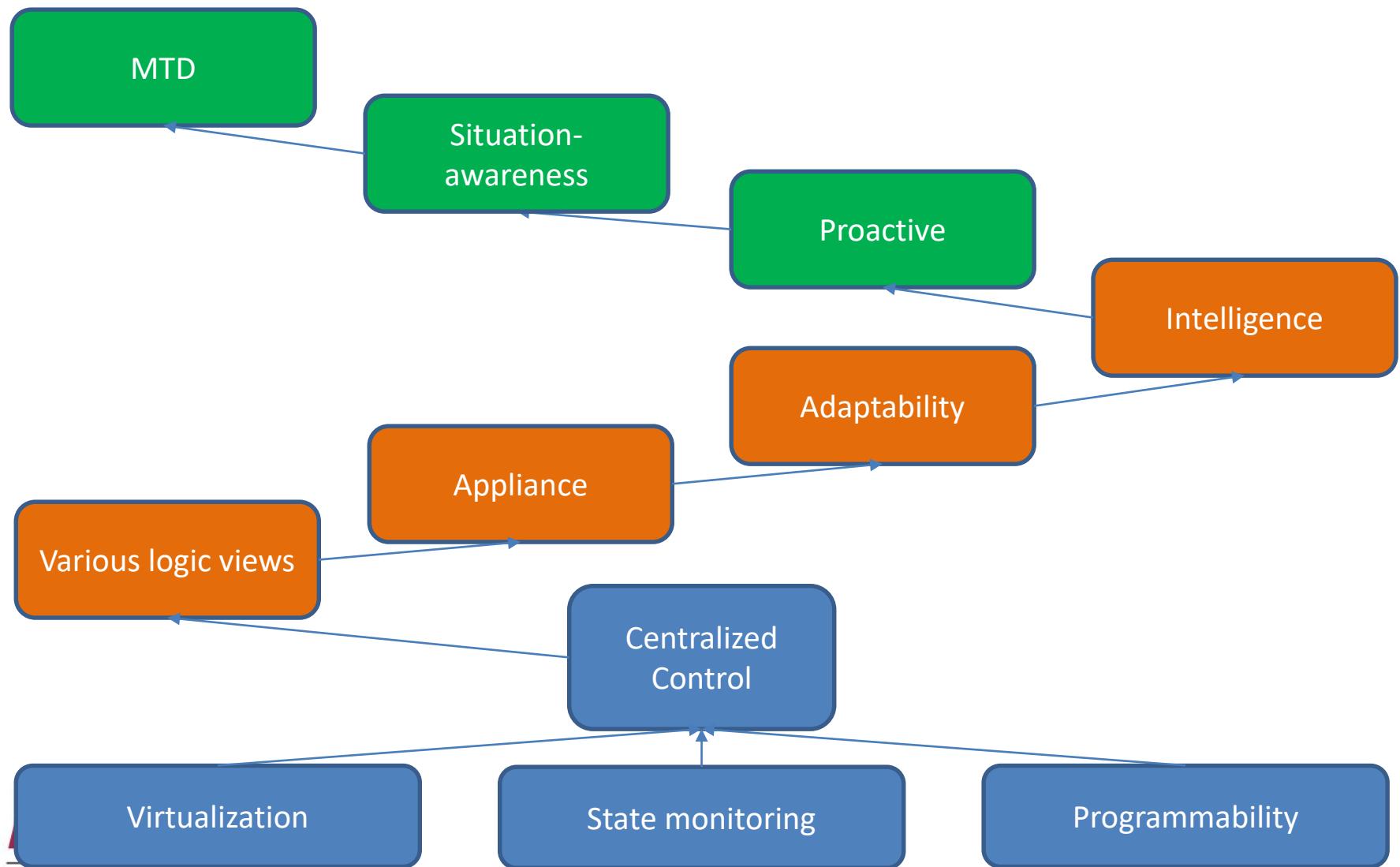
Agenda

1. Moving Target Defense (MTD)
2. **SDN-Based MTD Approaches**
 - SDN Basis
 - A Case Study: An Intelligent SDN-based MTD Approach
 - **Research challenges, opportunities, and future directions**
3. Q&A and Discussion

MTD is a systematic approach

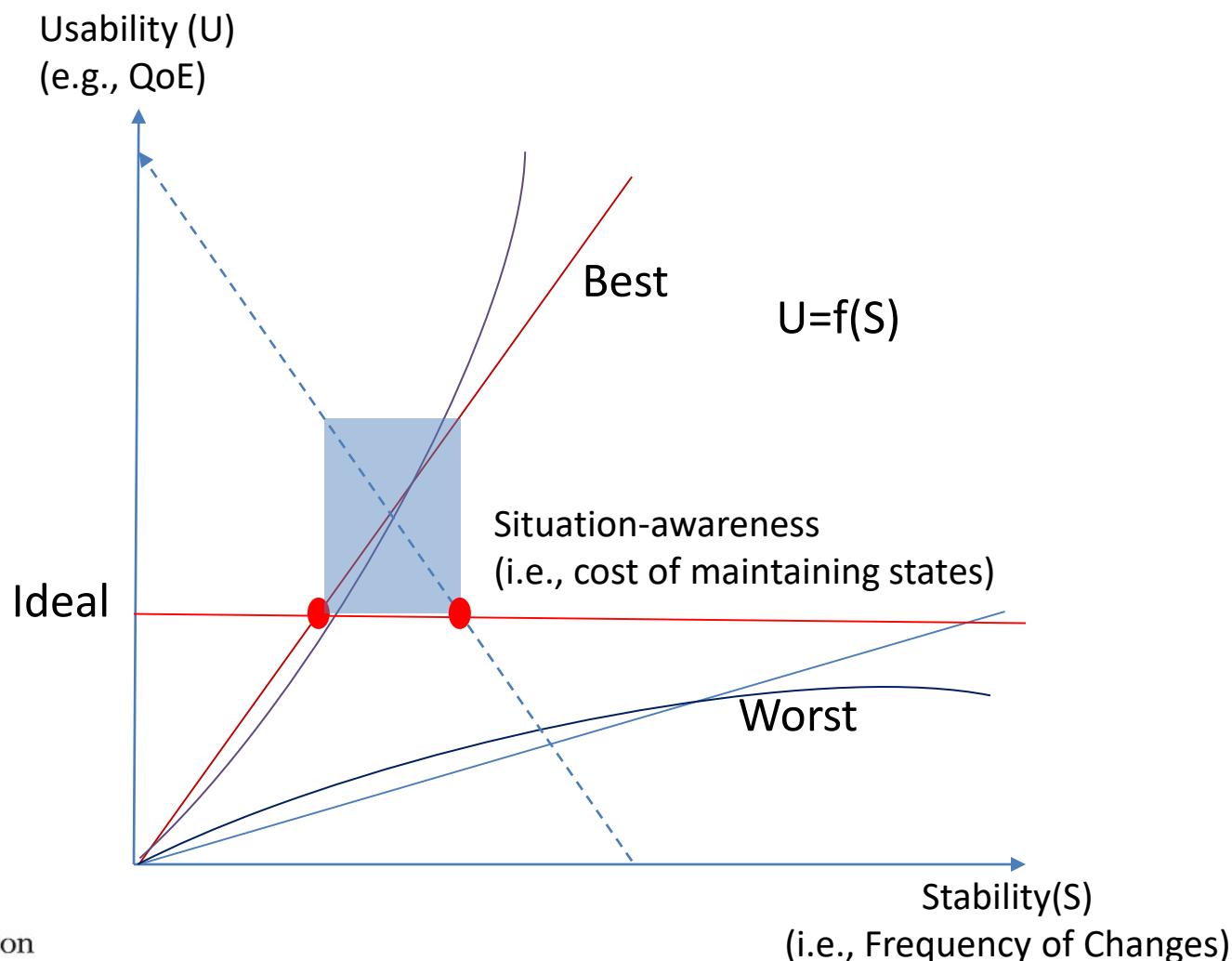


SDN/NFV and built-up Features to Support MTD

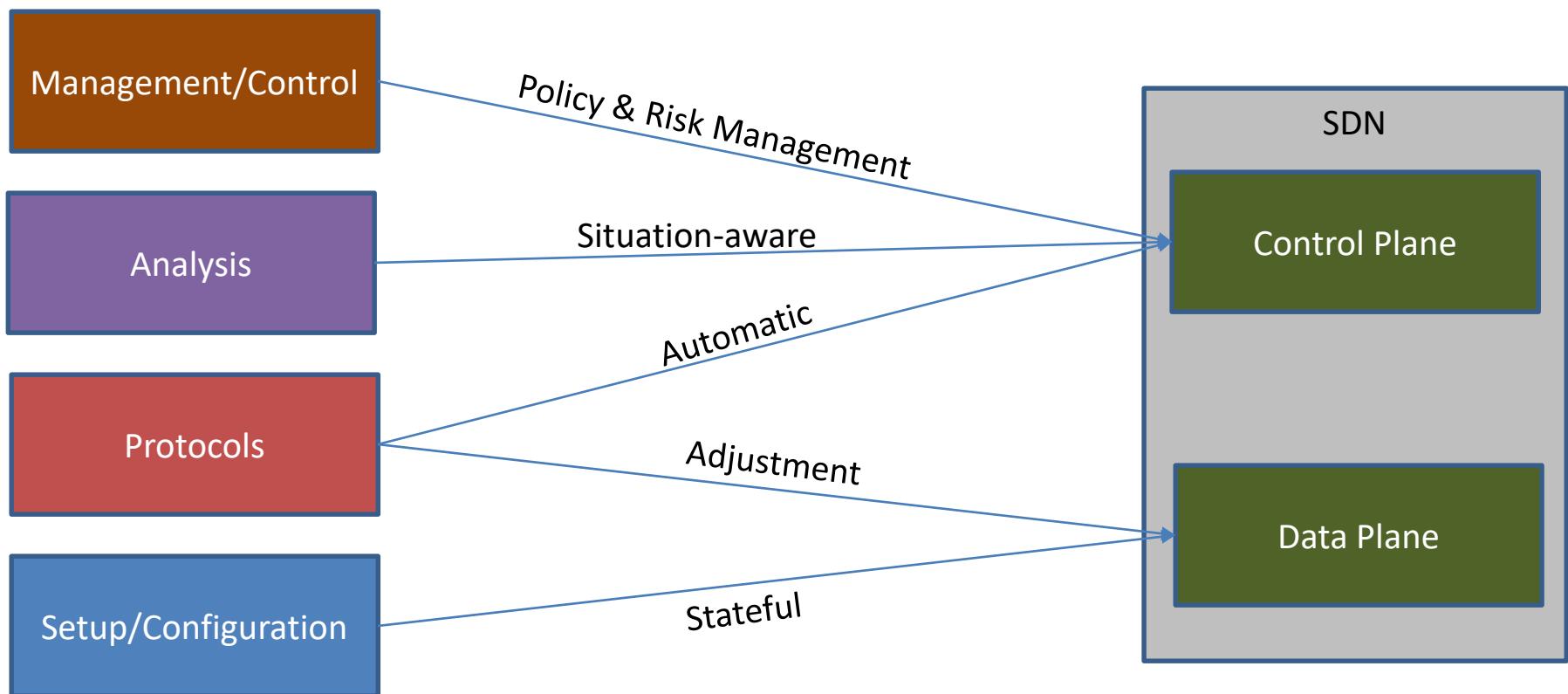


IS Frequently Changing Networking Setup a GOOD or BAD Idea?

MTD Aspect of Network Stability vs Usability

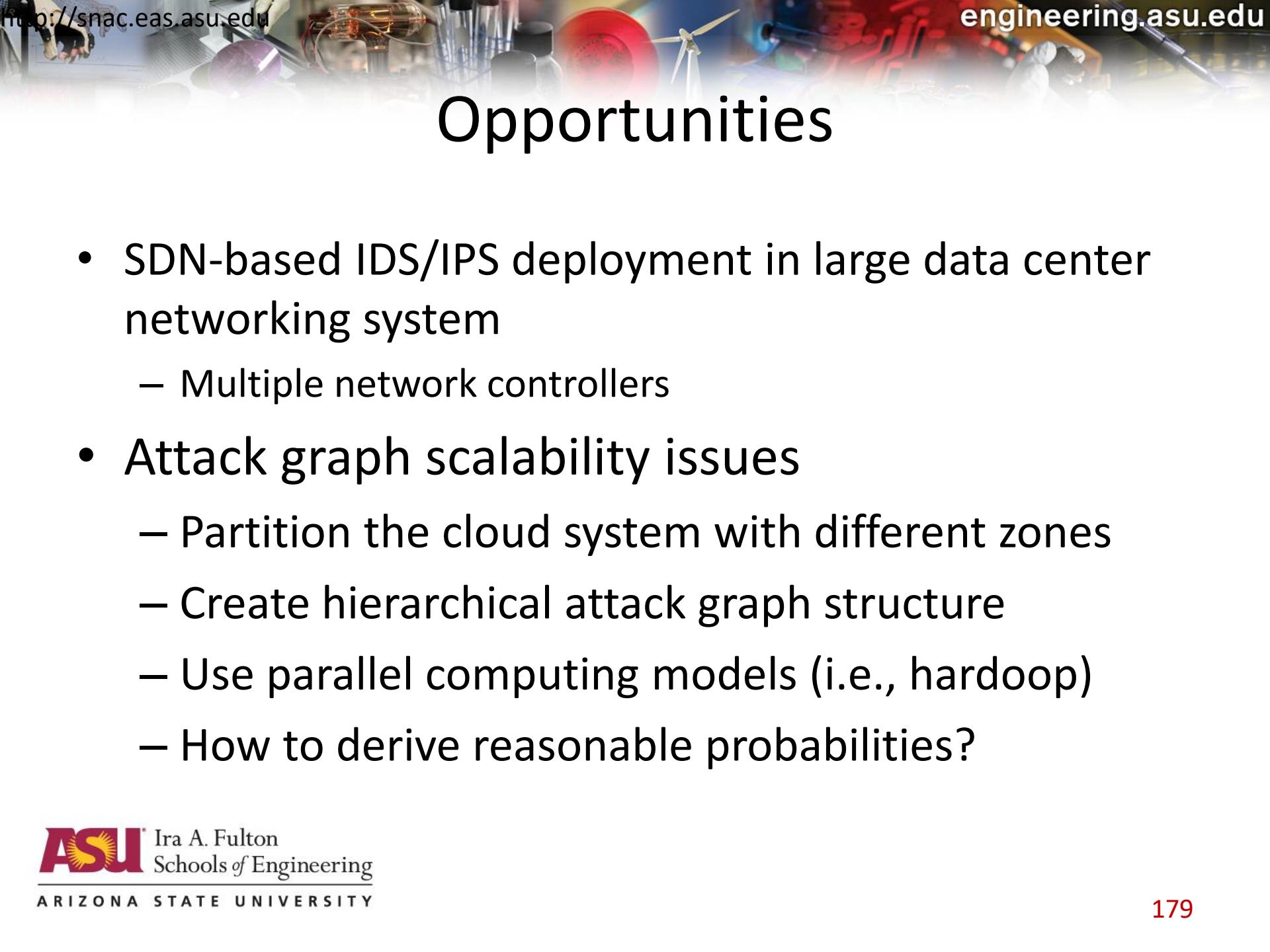


SDN/MTD Spectrum



Opportunities

- Implementing consistent and conflict-free security policies in SDN environment is progressively challenging.
- This work formalizes, detects and resolves conflicts in a multiple controller environment.
- Next steps
 - Parallelize processing
 - Adaptive prioritization
 - Diverse controllers

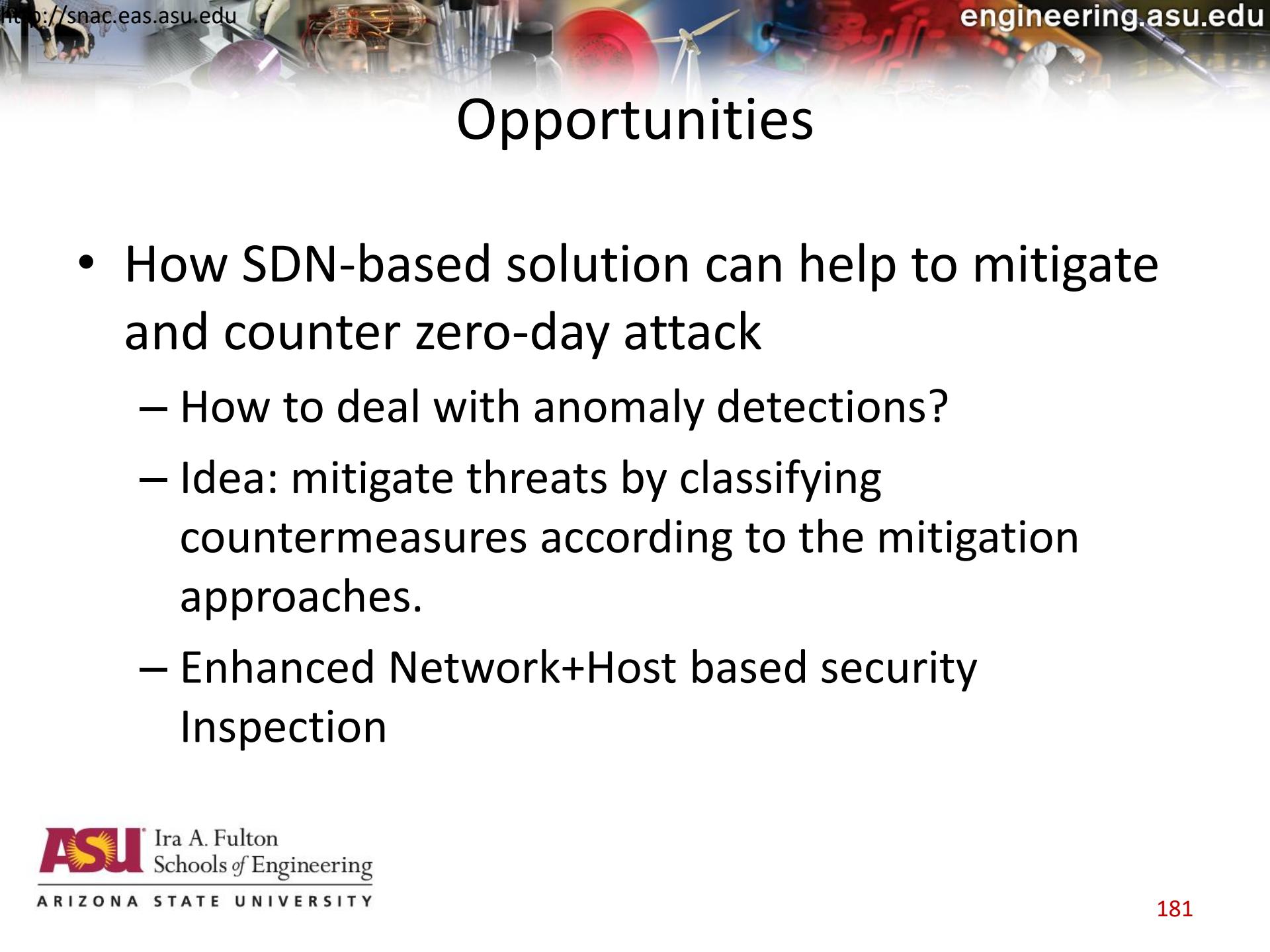


Opportunities

- SDN-based IDS/IPS deployment in large data center networking system
 - Multiple network controllers
- Attack graph scalability issues
 - Partition the cloud system with different zones
 - Create hierarchical attack graph structure
 - Use parallel computing models (i.e., hadoop)
 - How to derive reasonable probabilities?

Opportunities

- Game-based model
 - Can we emulate attackers and defenders to train the system to derive the best countermeasure selection?

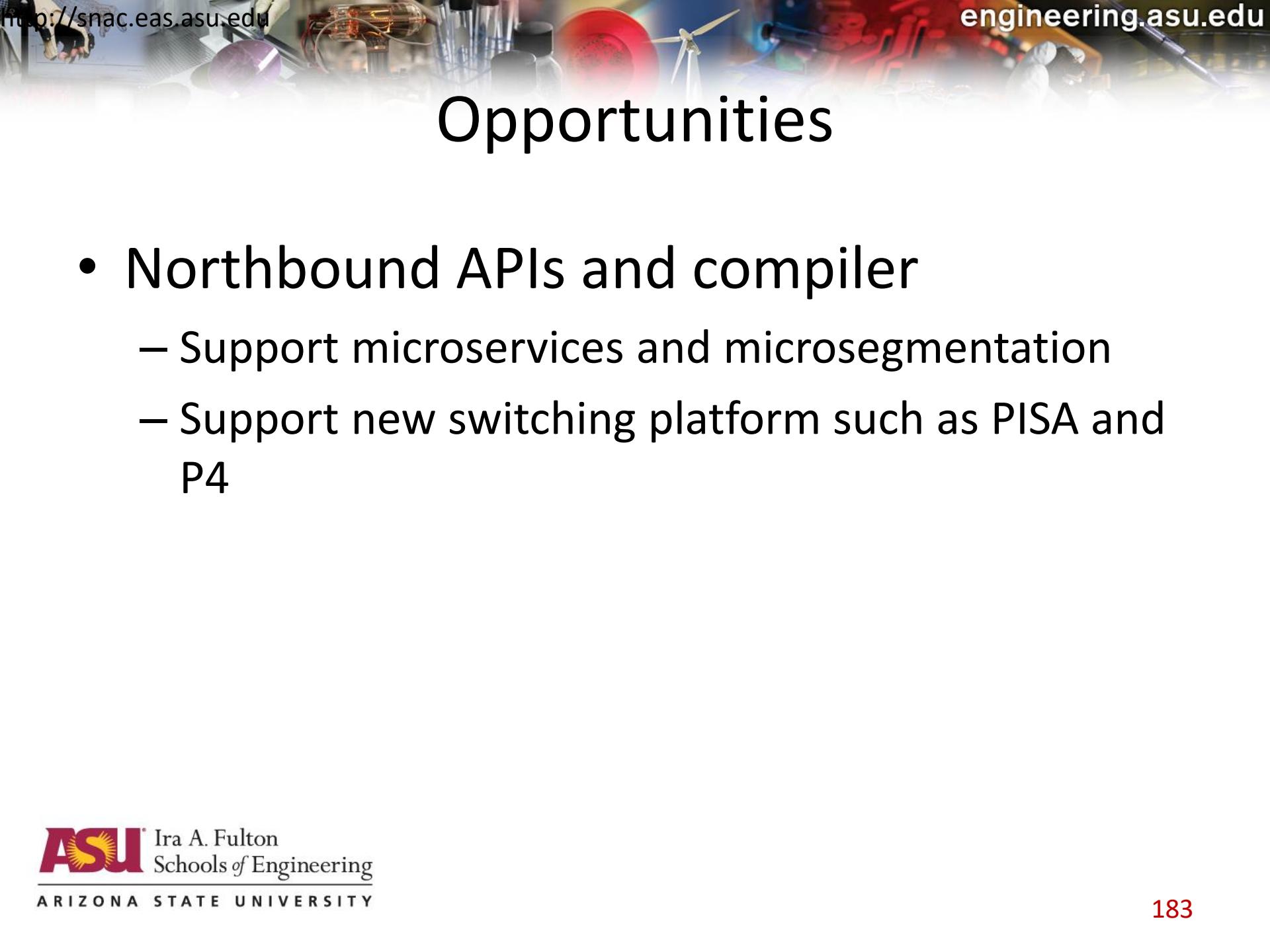


Opportunities

- How SDN-based solution can help to mitigate and counter zero-day attack
 - How to deal with anomaly detections?
 - Idea: mitigate threats by classifying countermeasures according to the mitigation approaches.
 - Enhanced Network+Host based security Inspection

Opportunities

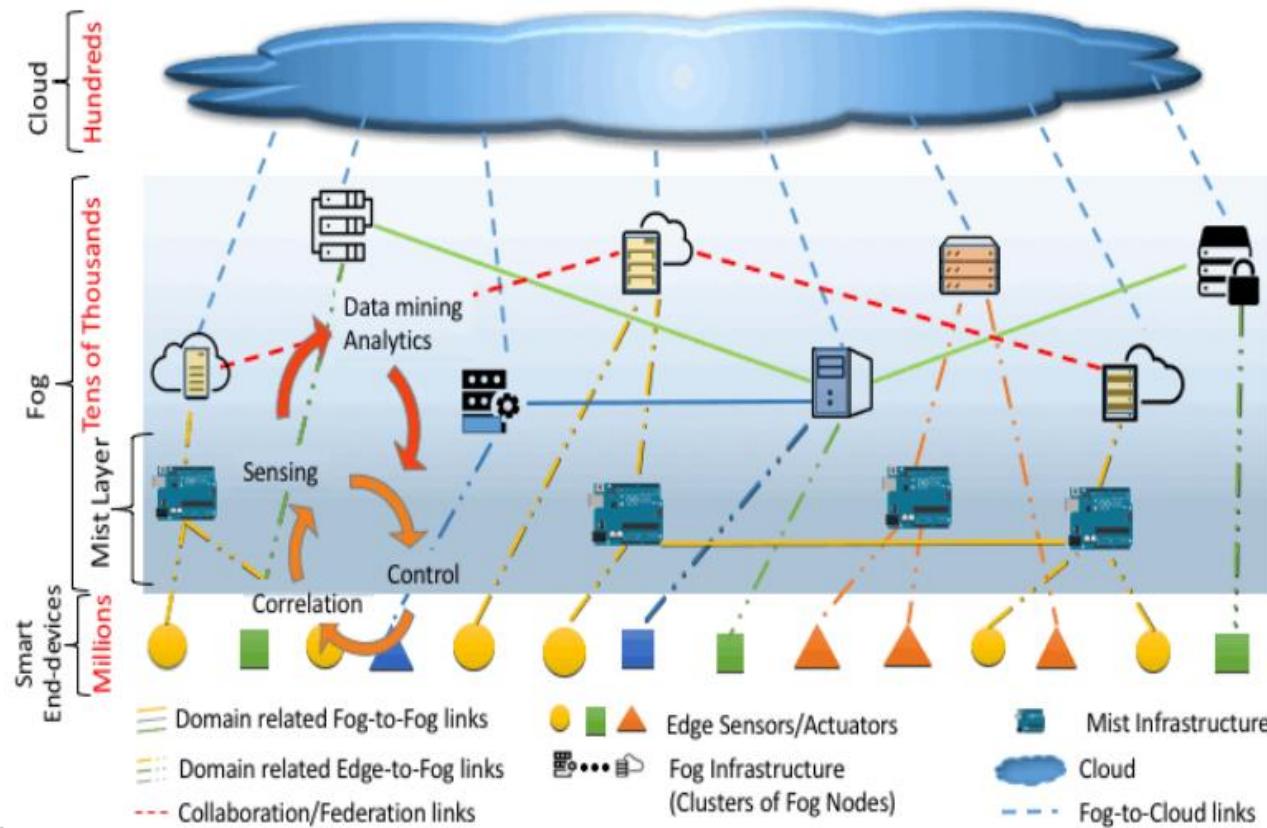
- Secure traffic engineering by considering
 - reliability,
 - survivability,
 - resource allocations,
 - User SLA,
 - QoS,
 - etc.



Opportunities

- Northbound APIs and compiler
 - Support microservices and microsegmentation
 - Support new switching platform such as PISA and P4

Era of Fog/Edge Computing



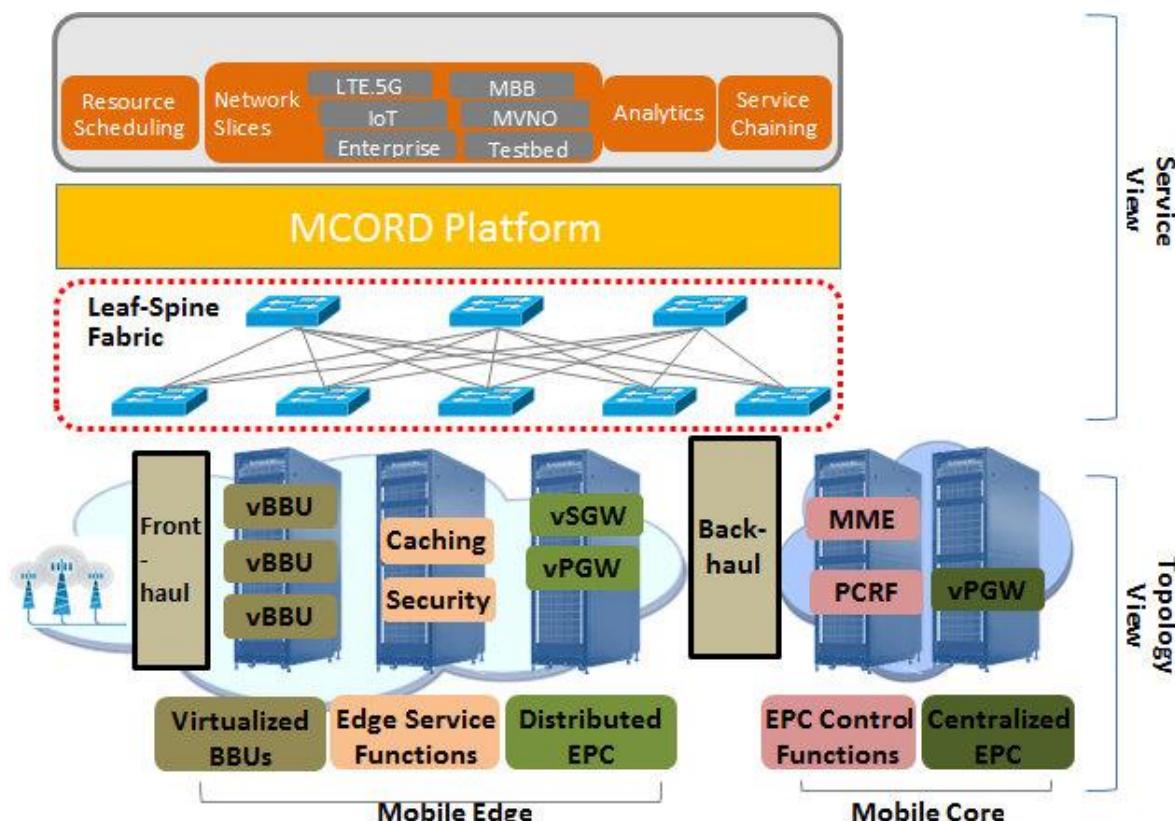
Source: NIST



Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

M-CORD Architecture



Source: OpenCord.org

ASU Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

Key Enabling Technologies for IoT Applications

- Convergence of NFV and SDN in Edge Cloud
- Automated Orchestration
- Dynamic Offloading

Agenda

1. Moving Target Defense (MTD)
2. SDN-Based MTD Approaches
- 3. Q&A and Discussion**



THANK YOU

Back up slides

MTD - How to defeat Buffer Overflow attacks?

Approaches: Address Space Layout Randomization

Threat Model

- Data leakage attacks, e.g., steal crypto keys from memory
- Denial of Service attacks, i.e., exhaust or manipulate resources in the systems
- Injection attacks
 - Code injection: buffer overflow, return-oriented programming (ROP)
 - Control injection: return-oriented programming (ROP)
- Spoofing attack, e.g., man-in-the-middle
- Authentication exploitation: cross-site scripting (XSS)
- Scanning, e.g., port scanning
- Physical attack: malicious processor

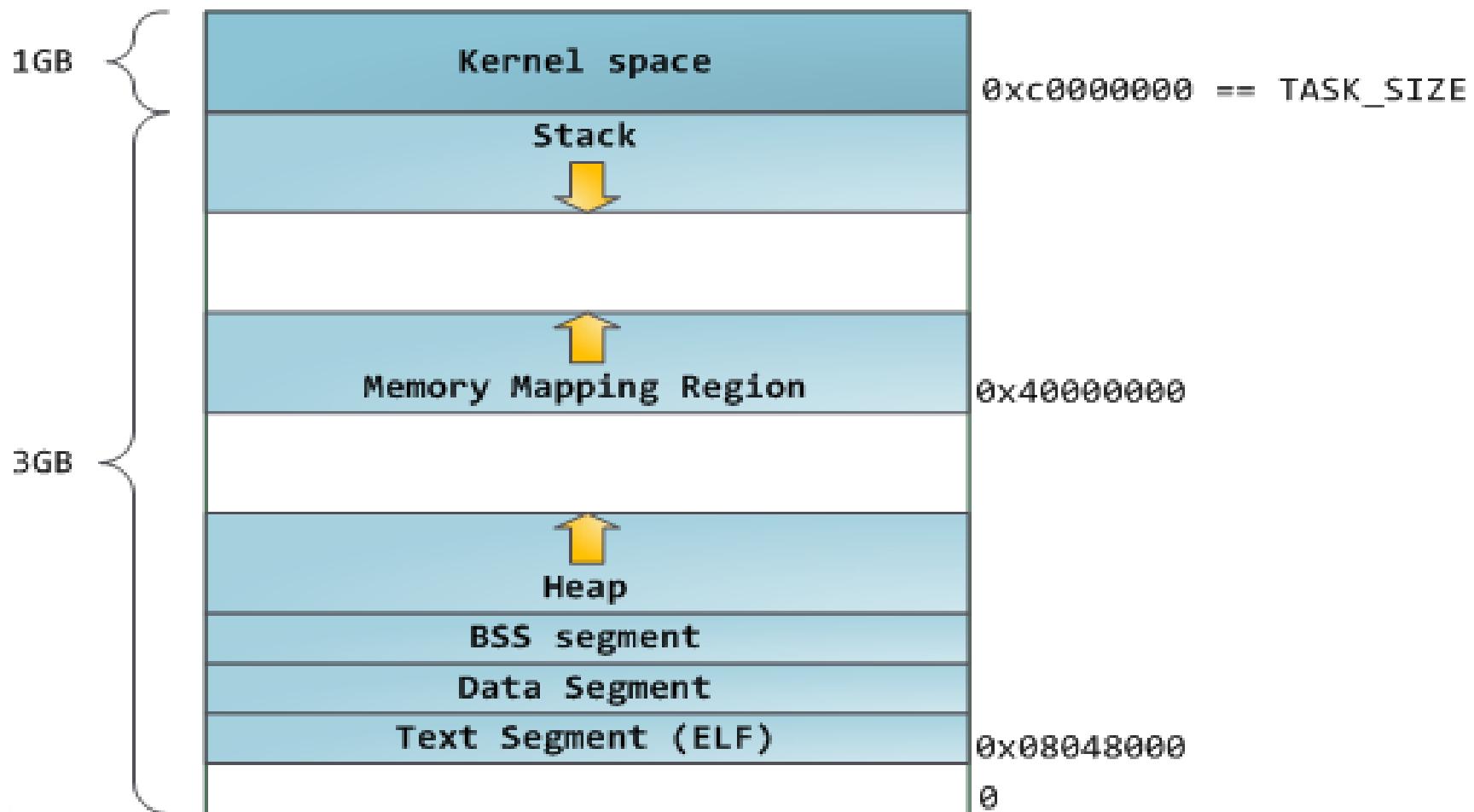
Buffer Overflow Attacks

- Also known as Buffer overrun, and BOF for short.
- First major exploit: **1988** Internet Worm, Robert Morris.
 - May exploit buffer overflow in fingerd service.
 - 26 years old techniques
- Heartbleed attack, **2014**
 - Due to implementation bug on OpenSSL library → Fail to check the length of Heartbeat request message
 - Leaking encryption key and user/password
 - Easy to fix, but might have been used as a zero-day attack for at least two years.

Computer Buffer

- Buffer: A contiguous block of computer memory, can be used for
 - Data: variables (static/global, dynamic/local), arrays
 - Code: user programs, shared libraries, kernel programs.
- To shield User/kernel programs from each other, virtual memory is used
- Within a virtual memory address space, different OS/CPUs have different ways to allocate buffers.
- On **Linux**, static/global variables allocated at load time on the data segment, **dynamic/local variables** are allocated at run time on the **stack**.

Segment Layout of Linux Process



Ira A. Fulton
Schools of Engineering

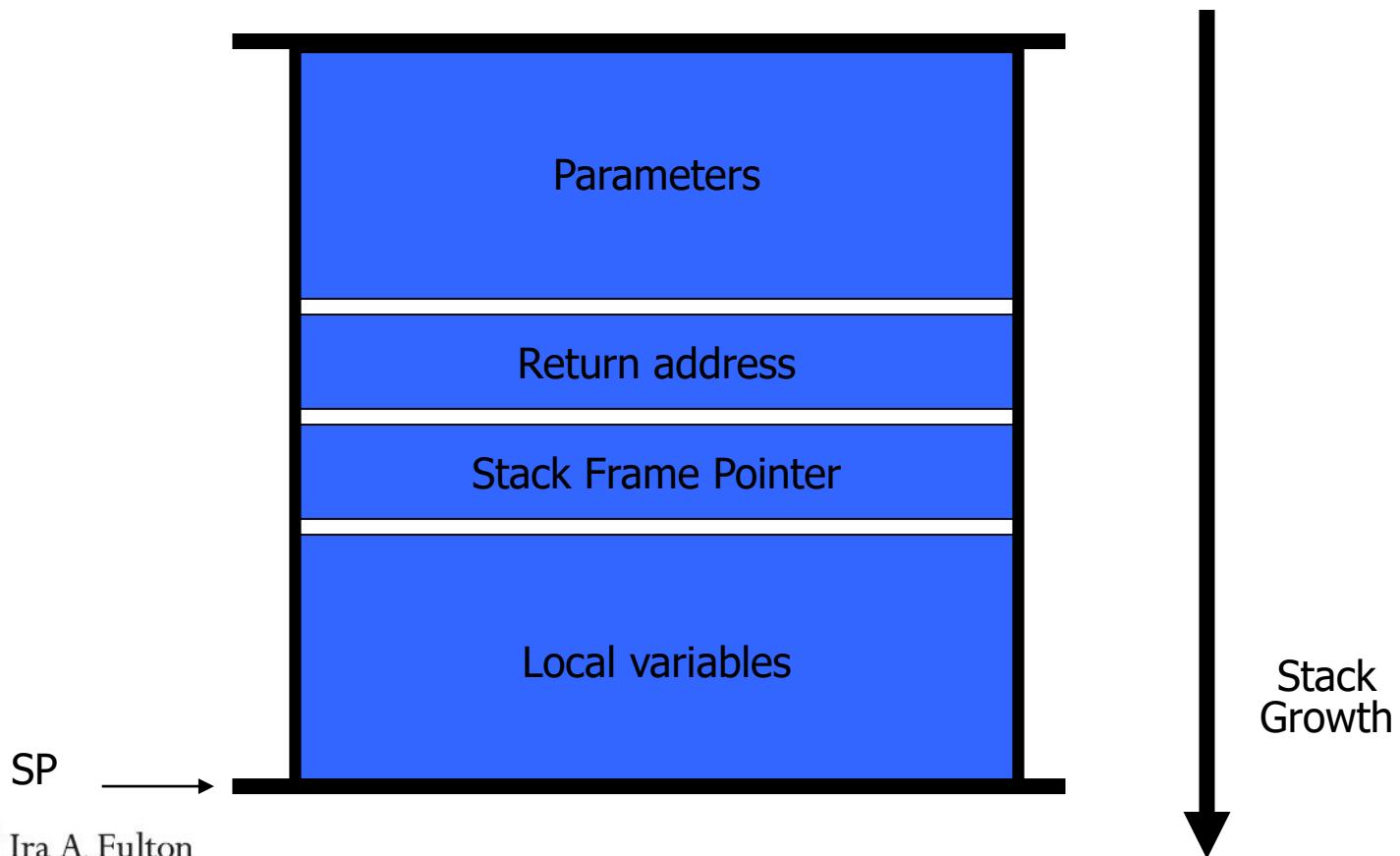
ARIZONA STATE UNIVERSITY

<http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory/>

What are BOF attacks?

- They attack corrupts data values in memory adjacent to a buffer by writing outside its bounds
- Stack-based exploitation
 - Discover vulnerable code
 - Overwrite the return address
 - New return address points to alternate code
 - Inject shellcode in to the stack or use existing code (return-oriented programming, ROP)
- Heap-based exploitation
 - Insert instructions in to the heap and then trick the program in to executing them.

Stack Frame

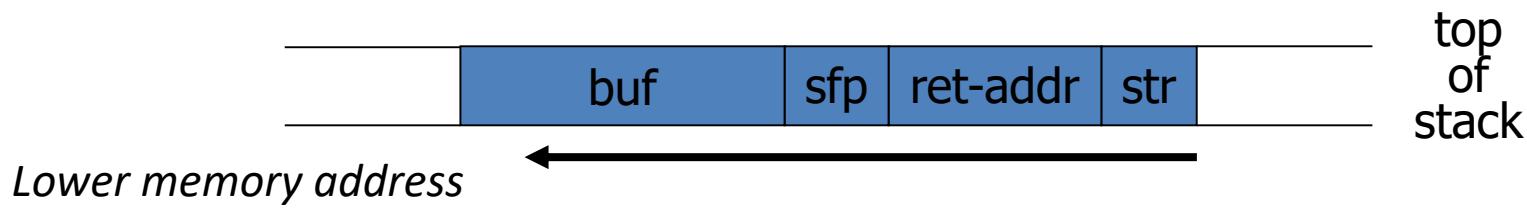


Stack Overflow Example

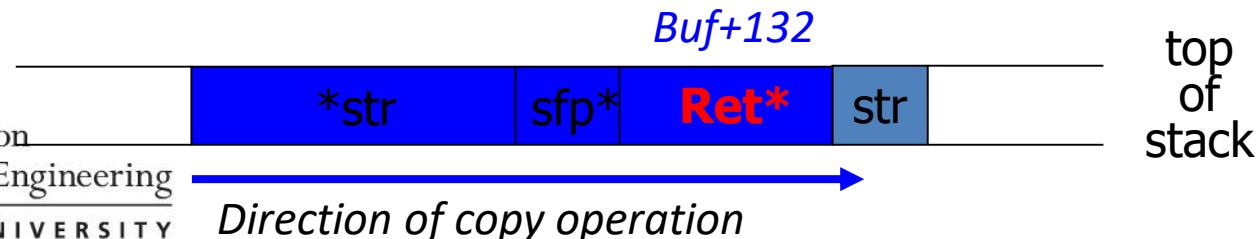
- Suppose a web server contains a function:

```
char a[30];  
void func(char *str) {  
    char buf[128];  
  
    strcpy(buf, str)  
  
    do-something(buf);  
}
```

- When the function is invoked the stack looks like:

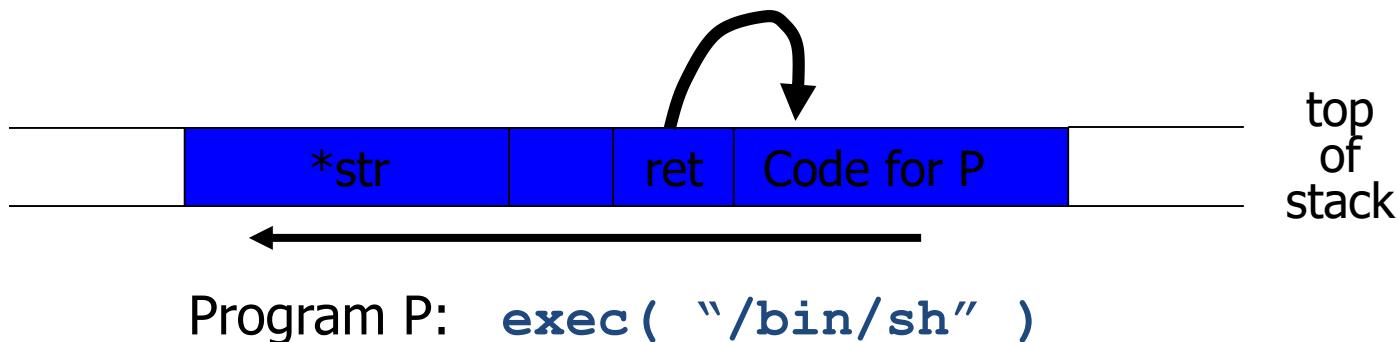


- What if `*str` is 136 bytes long? After `strcpy`:



Basic Stack Exploit

- Main problem: no range checking in `strcpy()`.
- Suppose `*str` is such that after `strcpy` stack looks like:



- When `func()` exits, the user will be given a shell.
- Note: attack code runs *in stack*.
- To determine `ret` **guess** position of stack when `func()` is called.

BOF Mitigations

- Proper programming language application
- Safe library usage
- Executable Space Protection
- Address Space Layout Randomization (ASLR)
- Deep Packet Inspection (DPI)
- Pointer Protection

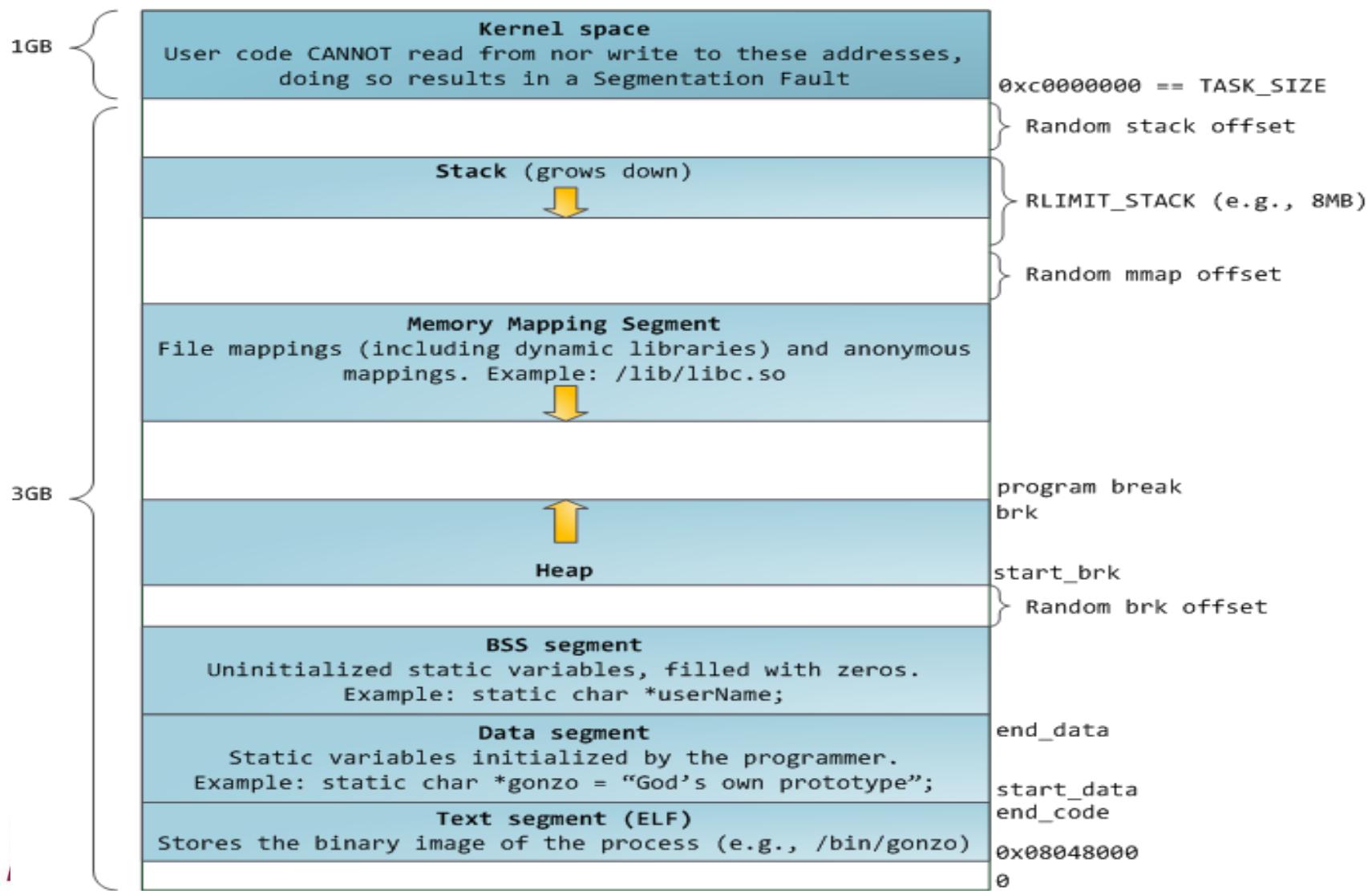
Problem: Lack of Diversity

- Buffer overflow and return-to-libc exploits need to know the (virtual) address to hijack control
 - Address of attack code in the buffer
 - Address of a standard kernel library routine
- Same address is used on many machines
 - Slammer infected 75,000 MS-SQL servers using same code on every machine
- Idea: introduce artificial diversity
 - Make stack addresses, addresses of library routines, etc. unpredictable and different from machine to machine
 - prevents attackers from using the same exploit code against all instantiations of the same program.

Address Space Layout Randomization (ASLR)

- Randomly choose base address of stack, heap, code segment
- Randomly pad stack frames and malloc() calls
- Randomize location of Global Offset Table
- Randomization can be done at compile- or link-time, or by rewriting existing binaries
 - Threat: attack repeatedly probes randomized binary

Segment Layout of Linux Process with PaX



Pax

- Linux kernel patch
- Goal: prevent execution of arbitrary code in an existing process's memory space
- Enable executable/non-executable memory pages
- Any section not marked as executable in ELF binary is non-executable by default
 - Stack, heap, anonymous memory regions
- Access control in mmap(), mprotect() prevents unsafe changes to protection state at runtime
- **Randomize address space layout**



Ira A. Fulton

Schools of Engineering

<http://pax.grsecurity.net>

ARIZONA STATE UNIVERSITY

PaX ASLR

- PaX applies ASLR to ELF binaries and dynamic libraries.
- User address space consists of three areas
 - Executable, mapped, stack
- Base of each area shifted by a random “delta” (on x86)
 - **Executable:** 16-bit random shift
 - Program code, uninitialized data, initialized data
 - **Mapped:** 16-bit random shift
 - Heap, dynamic libraries, thread stacks, shared memory
 - **Stack:** 24-bit random shift
 - Main user stack

PaX RANDSTACK

- Responsible for randomizing userspace stack
- Userspace stack is created by the kernel upon each `execve()` system call
 - Allocates appropriate number of pages
 - Maps pages to process's virtual address space
 - Userspace stack is usually mapped at `0xFFFFFFFF`, but PaX chooses a random base address
- In addition to base address, PaX randomizes the range of allocated memory

PaX RANDKSTACK

- Linux assigns two pages of kernel memory for each process to be used during the execution of system calls, interrupts, and exceptions
- PaX randomizes each process's kernel stack pointer before returning from kernel to userspace
 - 5 bits of randomness
- Each system call is randomized differently
 - By contrast, user stack is randomized once when the user process is invoked for the first time

PaX RANDMMAP

- Linux heap allocation: `do_mmap()` starts at the base of the process's unmapped memory and looks for the first unallocated chunk which is large enough
- PaX: add a random `delta_mmap` to the base address before looking for new memory
 - 16 bits of randomness

PaX RANDEXEC

- Randomizes location of ELF binaries in memory
- Problem if the binary was created by a linker which assumed that it will be loaded at a fixed address and omitted relocation information
 - PaX maps the binary to its normal location, but makes it non-executable + creates an executable mirror copy at a random location
 - Access to the normal location produces a page fault
 - Page handler redirects to the mirror “if safe”
 - Looks for “signatures” of return-to-libc attacks and may result in false positives

Base-Address Randomization

- Only the base address is randomized
 - **Layouts** of stack and library table remain the same
 - Relative distances between memory objects are not changed by base address randomization
- To attack, it's enough to guess the base shift
- A 16-bit value can be guessed by brute force
 - Try 2^{15} (on average) overflows with different values for addr of known library function – how long does it take?
 - Shacham et al. attacked Apache with return-to-libc
 - 216 seconds
 - If address is wrong, target will simply crash

Summary

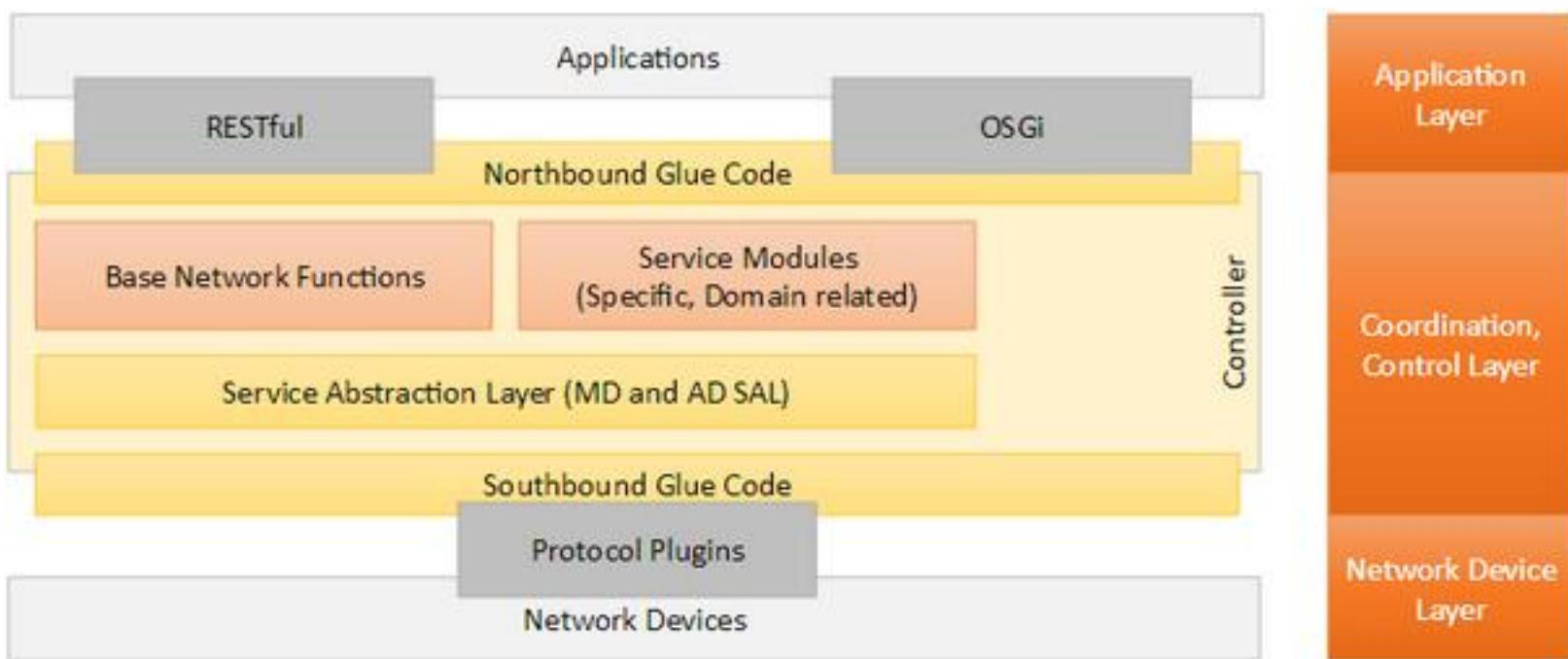
- Randomness is a potential defense mechanism
- Many issues for proper implementation
- Serious limitations on 32-bit architecture
 - On 32-bit systems, runtime randomization cannot provide more than 16-20 bits of entropy
- How about being combined with “a crash detection and reaction mechanism” called watcher?
 - May not respond quick enough
 - May suffer from DoS attack

Improvements

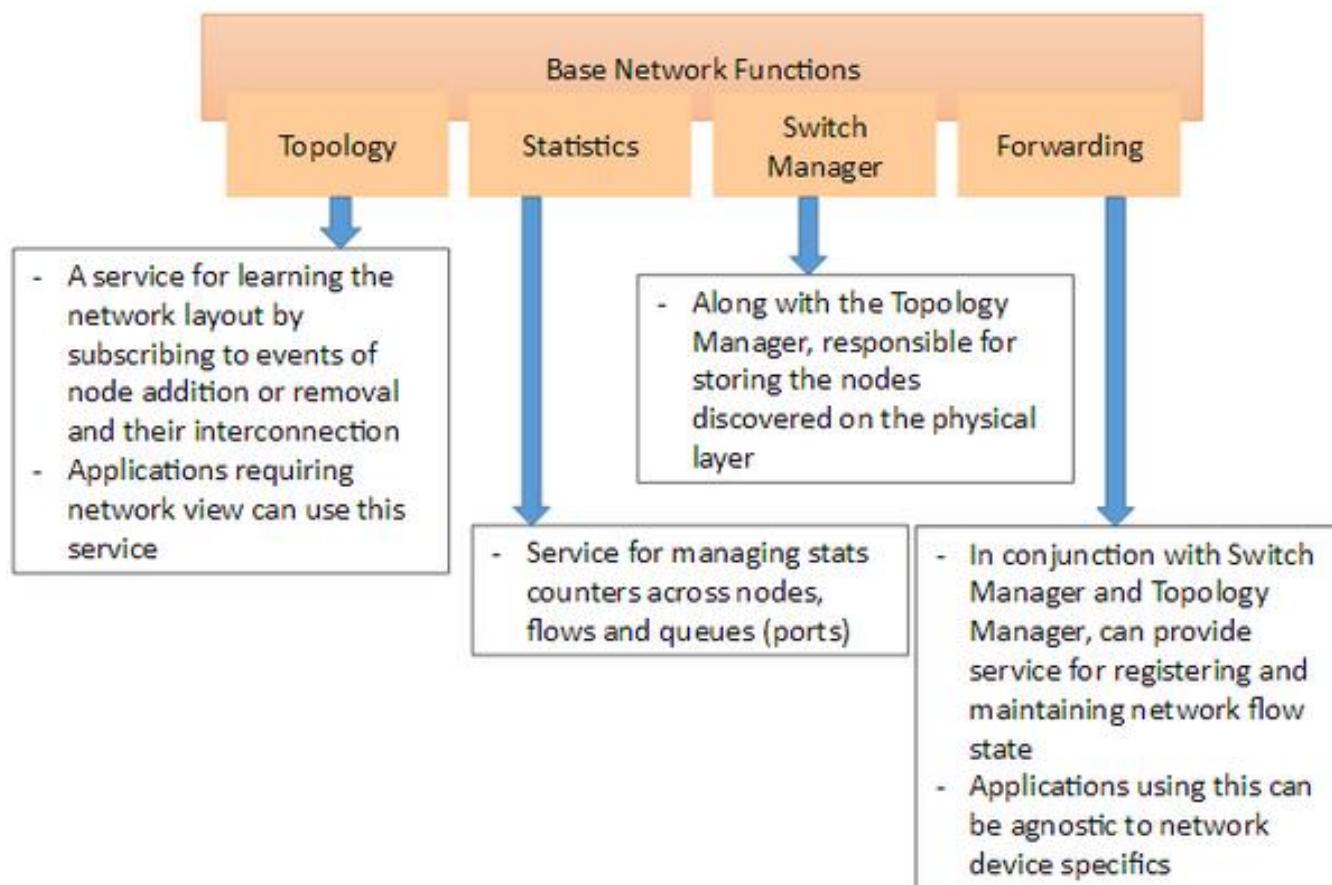
- Move to 64-bit architecture
 - At least 40 bits of randomization
 - Entropy is high enough, and easy to detect attacks of this magnitude.
- Frequent Re-randomization
 - Randomize the address space layout of a process more frequently after process creation.
 - Adds no more than 1 bit of security against brute force attacks regardless of the frequency, 2^{n-1} vs. 2^n
 - It can mitigate the damage when the layout of a fixed randomized address space is leaked through other channels

BACKUP

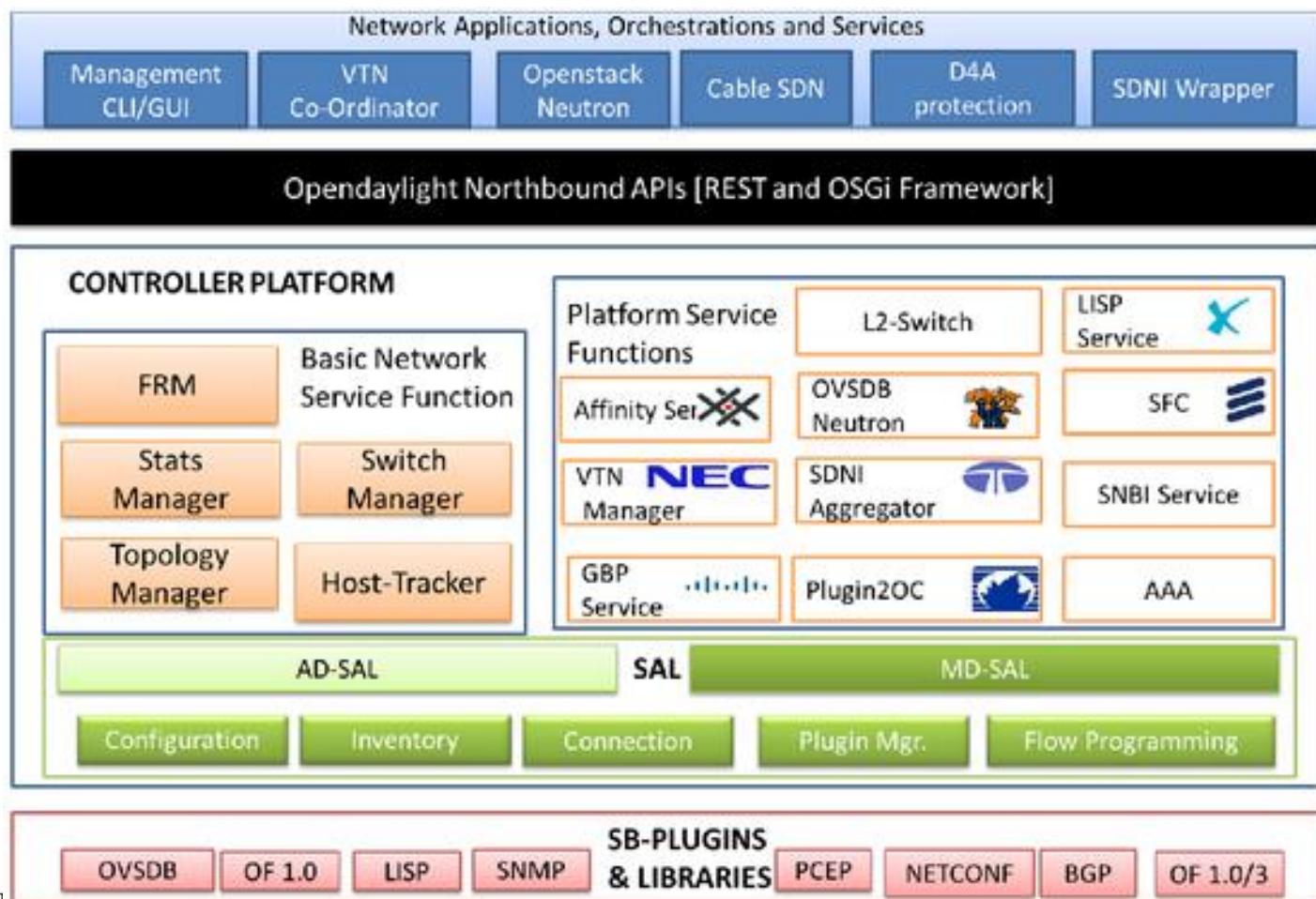
OpenDaylight



OpenDaylight

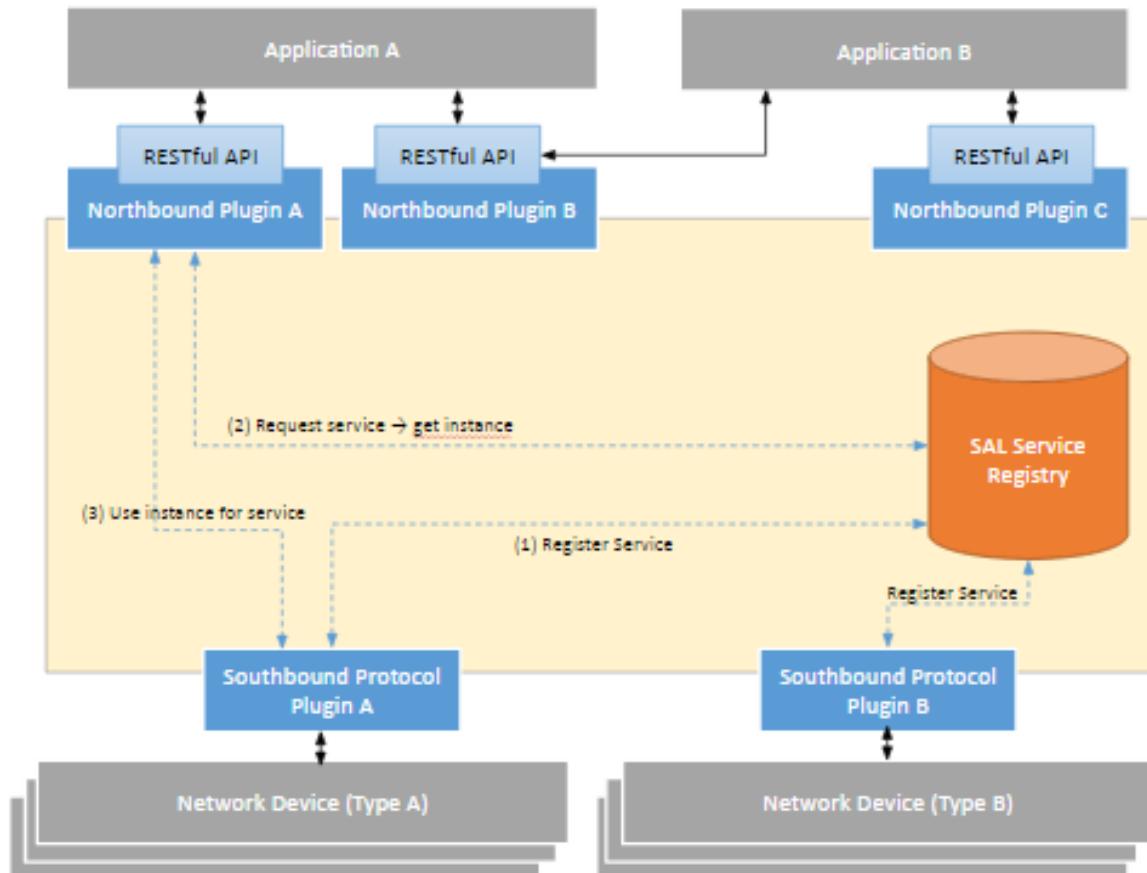


OpenDaylight



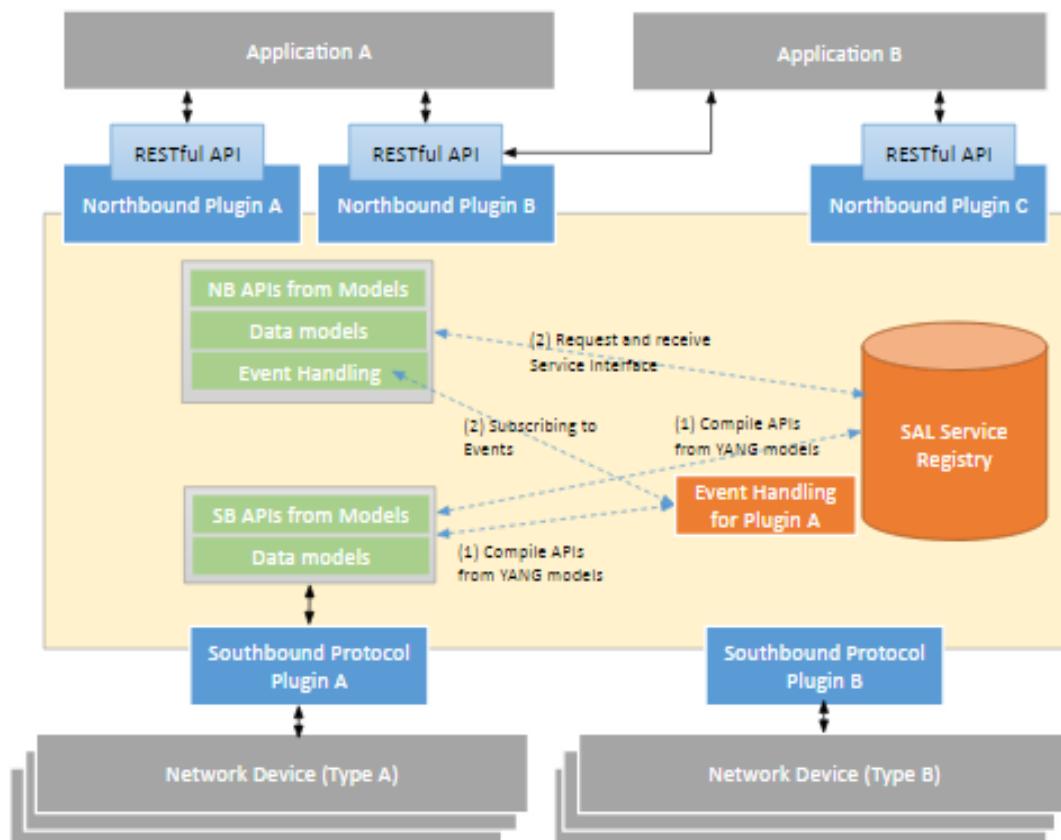
Application-Driven SAL

- AD-SAL: to provide abstraction through use of a generic set of APIs that provide all device functions.
- AD-SAL gives a developer independence from handling device-level complexities.
- AD-SAL was designed for north-south transparency.



Model-Driven SAL

- The providers (generally southbound plugins) create a model of the data or services they expose. Models are in form of YANG definitions.
- a YANG compiler is used to create uniform APIs for the consumers that then are made part of the plugin.
- These APIs are tool generated, allowing a very high level of uniformity between them in terms of definitions and usage.



HEADER SPACE ANALYSIS

Header Space Analysis

- Even with the most stringent policies, some questions are hard to answer:
 - Can host A talk to host B?
 - Is Group X provably isolated from Group Y?

Header Space Analysis

- Provable verification of security policies
- Apply flow table rules as transform functions
- Packet processing depends on
 - Header bits
 - Physical location (logical... really)

Recap - SDN Switch Functionalities

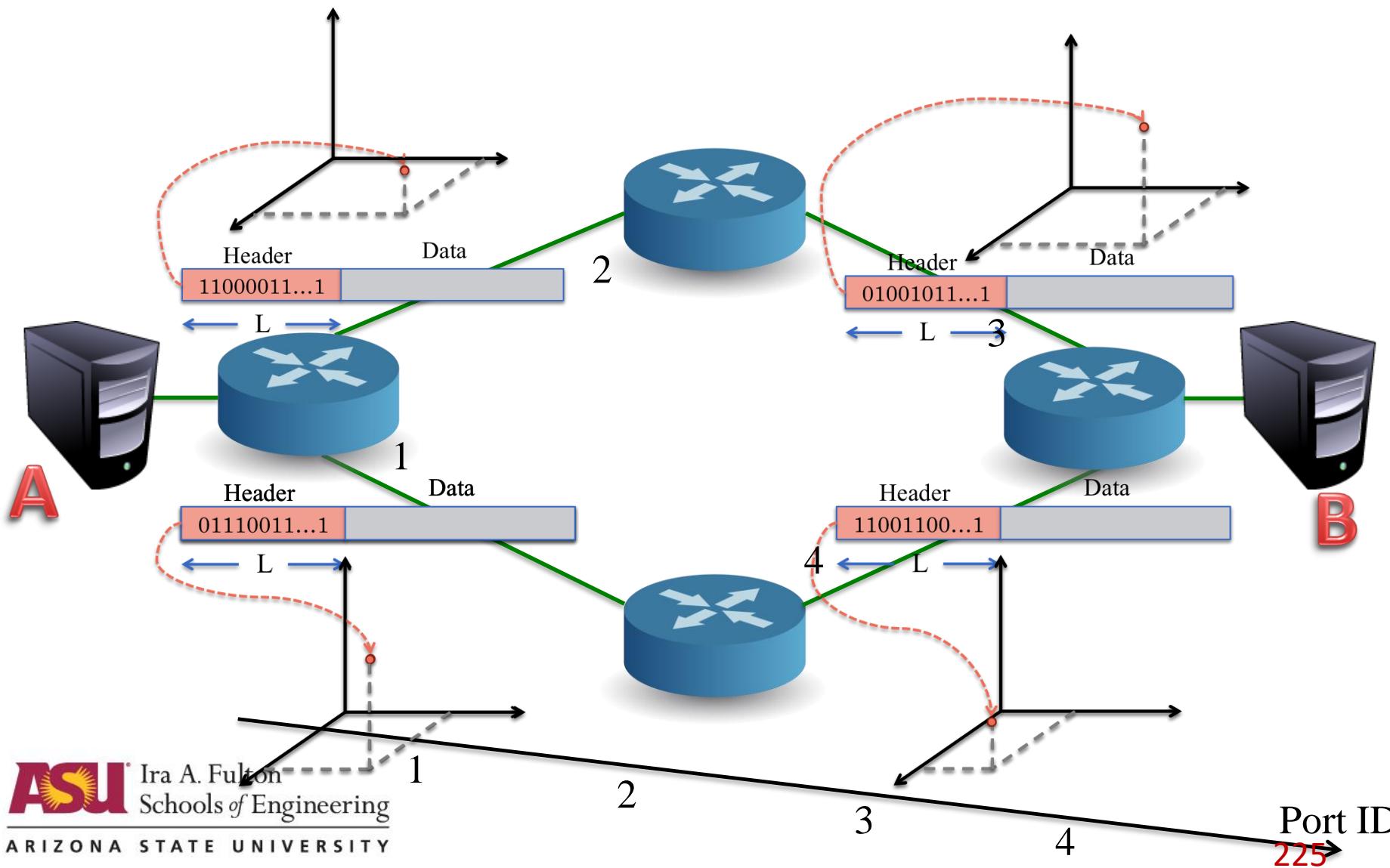
- Router
 - Longest IP prefix match
 - Forward
- Switch
 - Destination MAC address match
 - Forward or flood
- Firewall
 - Match network 5-tuple
 - Permit or deny
- NAT
 - Match source L3 and L4
 - Rewrite L3 and L4 addresses

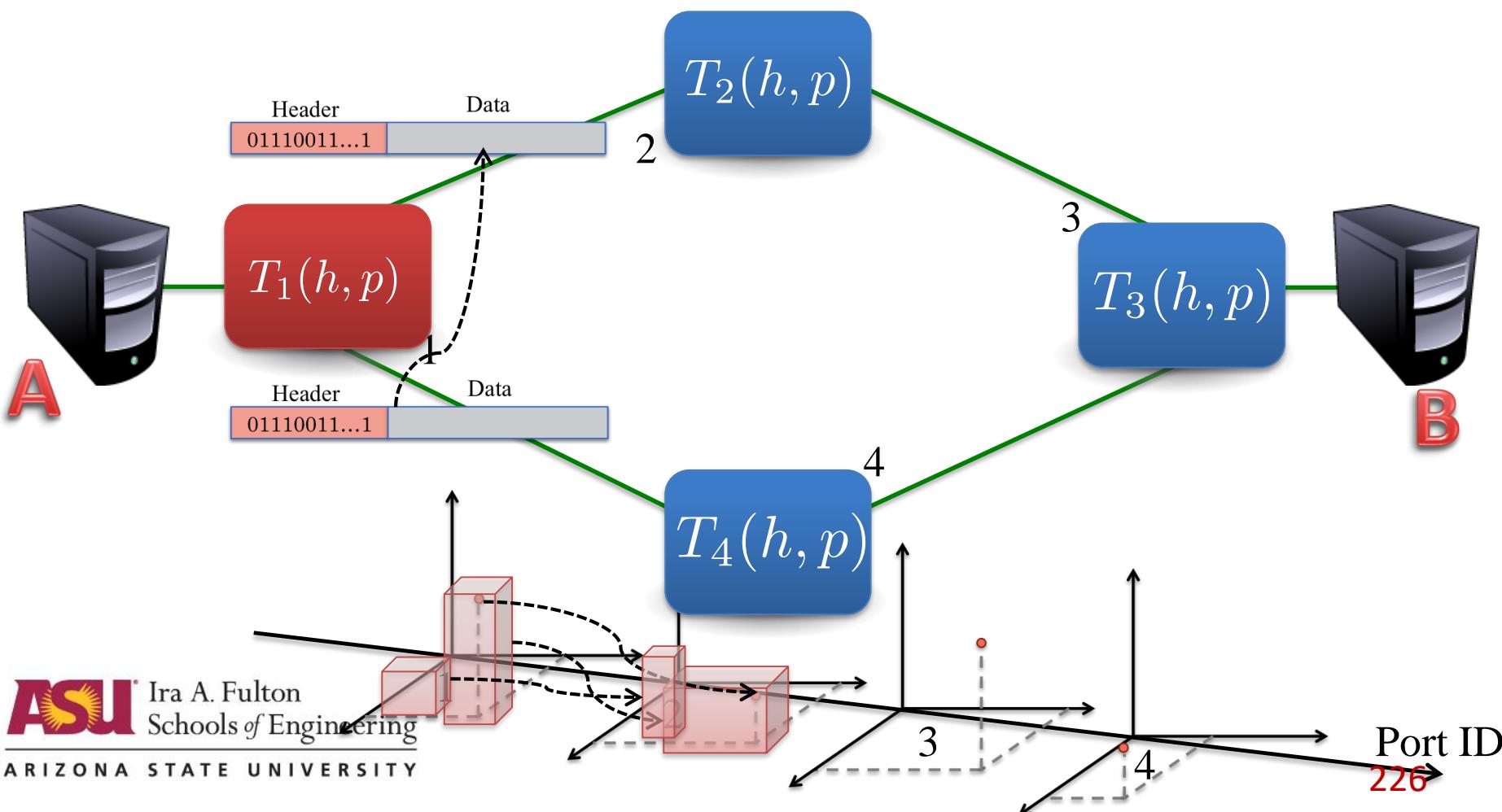
Header Space Analysis

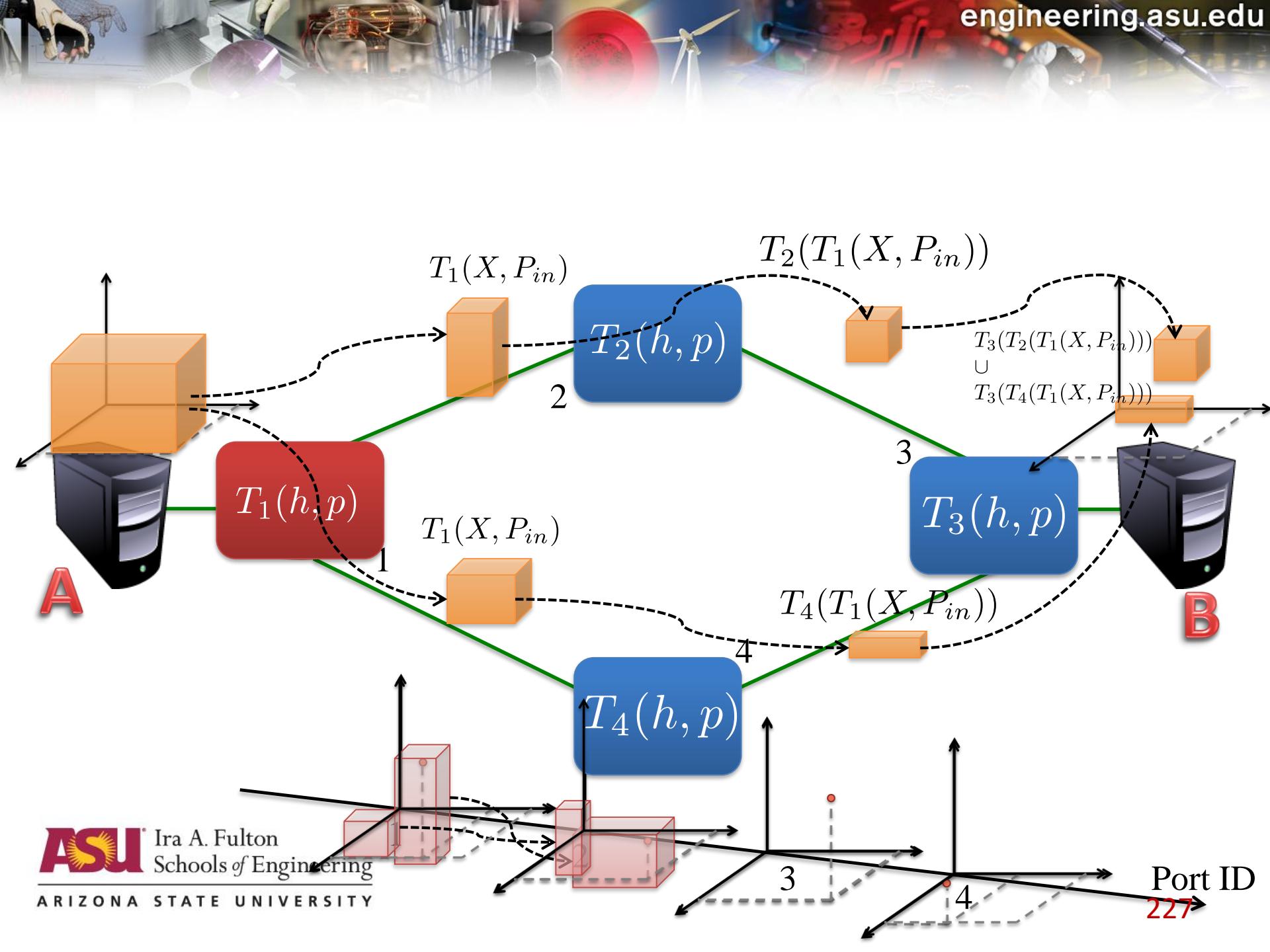
- Packets are modeled as a unique point in an L-dimensional space
- Forwarding behavior of devices modeled as a transform function
 - $T: (h,p) \rightarrow \{(h_1,p_1), \dots, (h_n,p_n)\}$

Header Space Analysis

- End-to-end behavior can be modeled by composing these transform functions
 - Example: $T_3(T_2(T_1(h,p)))$
- Use set algebra to establish reachability and isolation

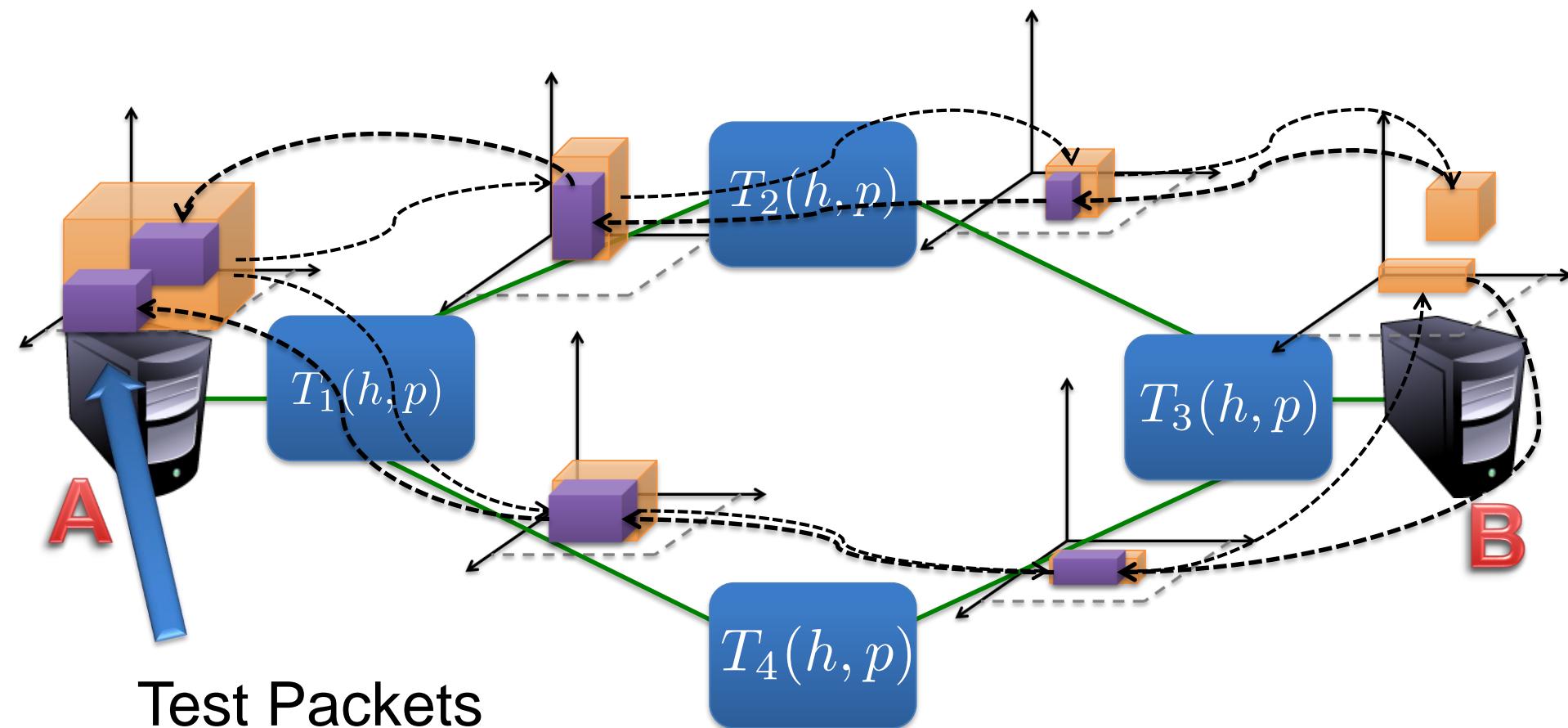






Header Space Analysis

- More applications
 - Detect network loops
 - No packet should be reachable from any port to itself
 - Ensure two flows don't have the same path
 - Automatic test packets
 - Use inverse transform functions



Test Packets

PAXOS

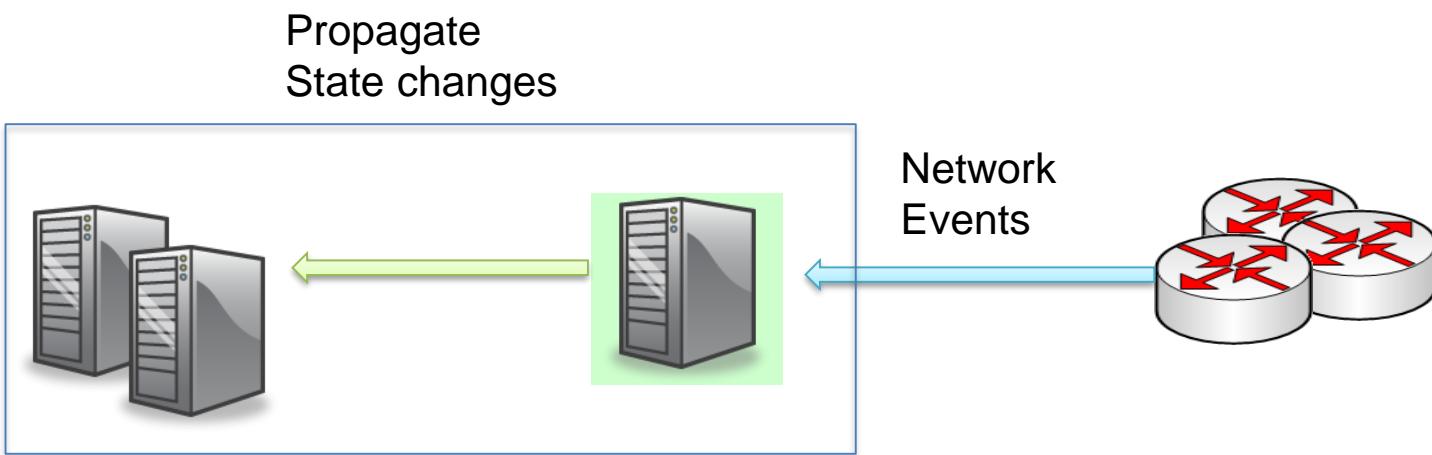
Google's SDN WAN

- Each site runs a set of controllers
 - Primary is determined by Paxos

Paxos

- N controllers
 - One primary
 - All N maintain the same state
- Switches interact with primary
- Change by consensus
- In case primary controller fails
 - N-1 elect new primary by consensus

Paxos



Paxos

- Pros
 - Well researched
 - Good FT
 - Many implementations in the wild
- Cons
 - Time to recover
 - Impacts throughput

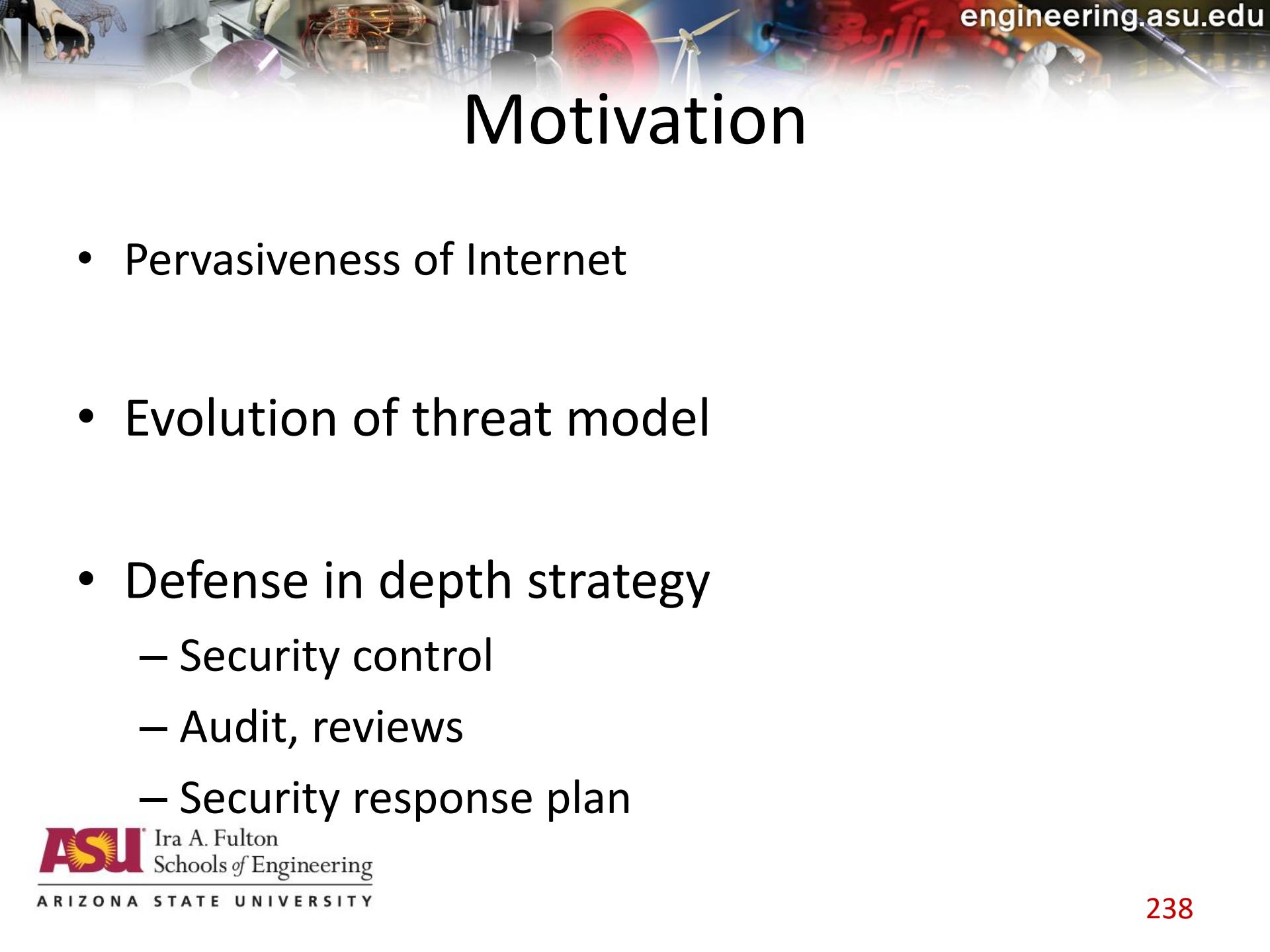
RELATED WORKS

Threat Model

- Internet threat model
 - Trust thy neighbor
 - System centric
- Asset centric model
 - Protect the Cyber Key Terrain (CKT)

Motivation

- Evolution of threat model
- Defense in depth strategy
 - Security control
 - Audit, reviews
 - Security response plan

A collage of various engineering and technology-related images, including a wind turbine, a laboratory setup with glassware, a red vinyl record, a computer screen displaying code, and a person working in a lab.

Motivation

- Pervasiveness of Internet
- Evolution of threat model
- Defense in depth strategy
 - Security control
 - Audit, reviews
 - Security response plan

Firewall Evolution

- Routers
- Security boxes (IPTables)
- Distributed firewalls
- Middleboxes

— Add SDN!

Management Problems

Problem Setup: A firewall F contains a rule set $R = \{r_1, r_2, \dots, r_n\}$ with each rule $r_i \in R$ containing the following information: a) priority p ; b) the network 5-tuple $n = (\text{source IP}, \text{source port}, \text{destination IP}, \text{destination port}, \text{protocol})$; and c) action a . Thus, we represent $r_i = (p_i, n_i, a_i)$.

Management Problems

Packet Classification Problem: For an incoming packet Π_{in} with the network 5-tuple n_{in} , the packet classification problem [6] in firewalls seeks to find out the set $R_m \subseteq R$ where $R_m = \{r_i = (p_i, n_i, a_i) \mid r_i \in R \wedge n_i = n_{in}\}$. The problem can be further extended to determine rule $r_x = (p_x, n_x, a_x) \in R_m$ such that $p_x > p_y \forall r_y \in R_m$.

Management Problems

Conflict Detection Problem: The conflict detection problem [6] seeks to find rules $r_i = (p_i, n_i, a_i)$, $r_j = (p_j, n_j, a_j)$ such that $r_i, r_j \in R$ and $n_i = n_j \wedge (a_i \neq a_j \vee p_i \neq p_j)$.

Firewall Anomalies

- Redundancy
- Shadowing
- Generalization
- Correlation

KeyNote

- Described in RFC 2704
- Language to describe security policies
- Incorporates delegation
- Security implementation using digital signature

Ioannidis *et. al.*

- Use KeyNote to express centralized security policy
- Use a central policy server that contains corporate security policy
 - Distributed firewall nodes request updates, or they can be pushed by the server
 - Use IPsec or digital signature to ensure integrity

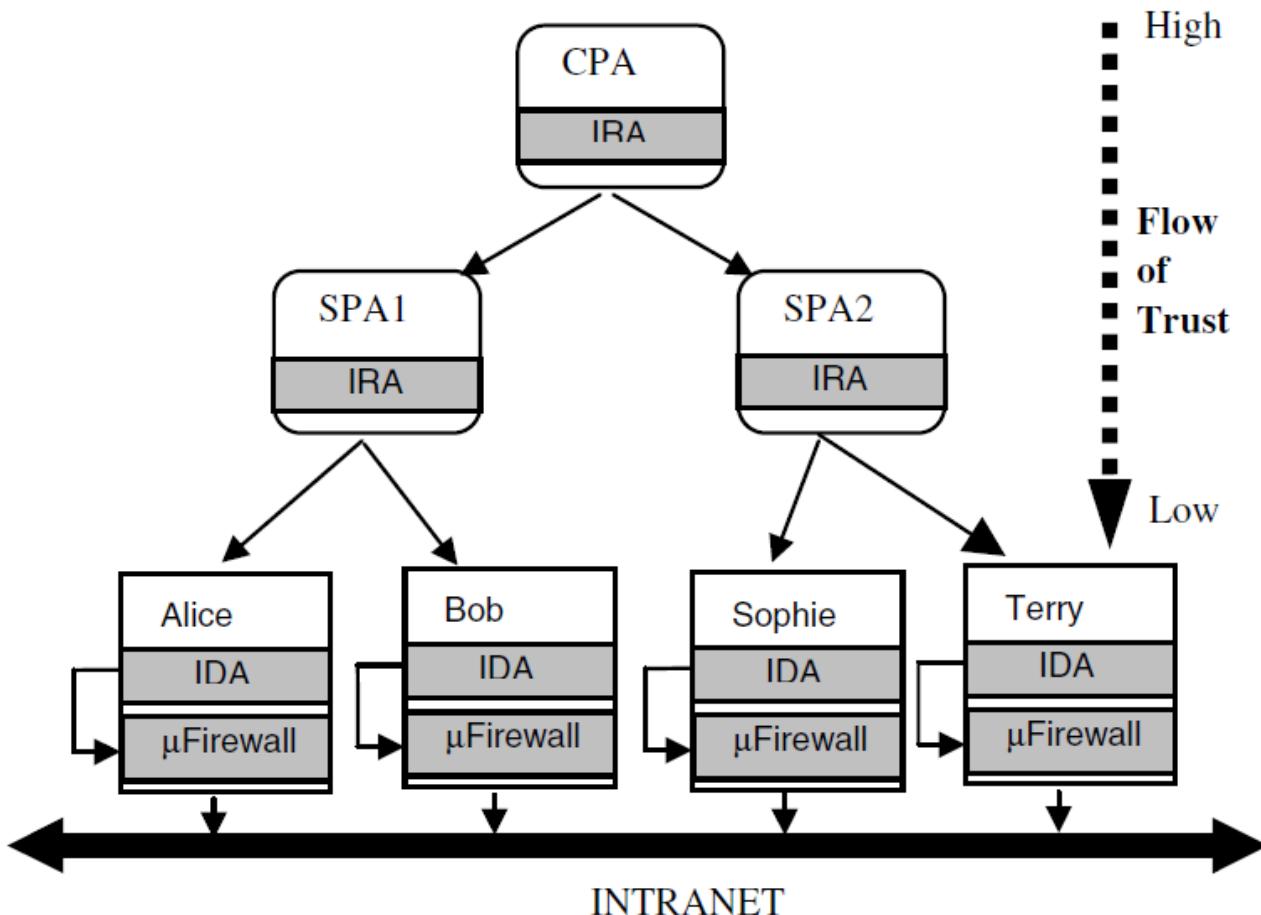
Ioannidis *et. al.*

- Central server needs to be always on
- Single point of failure
- Scalability issues
- How does KeyNote deal with anomalies

μ Firewall

- Hierarchical management model
- Central Policy Administrator (CPA)
 - Communicates with Sub-Policy Administrators (SPA)
 - Depth dependent on network characteristics

μ Firewall



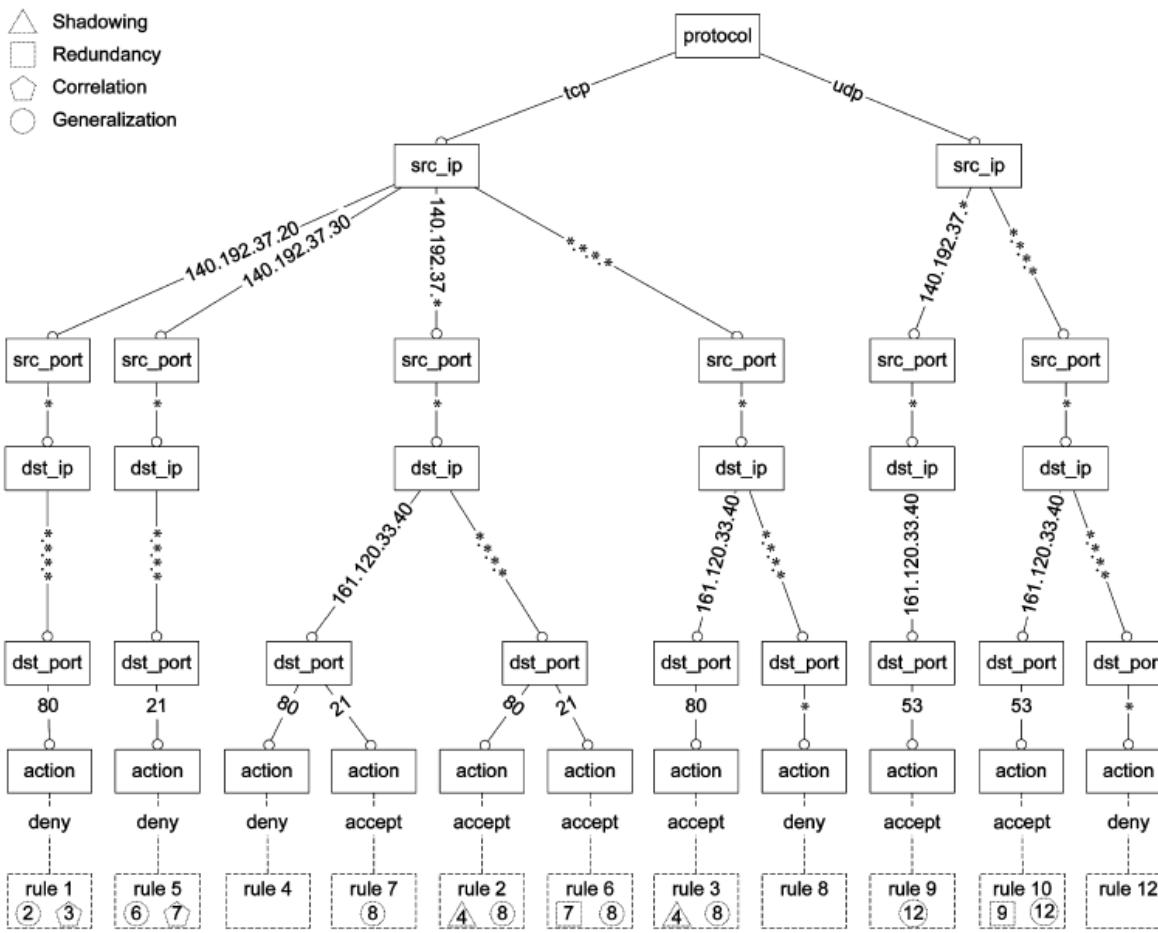
μ Firewall

- ~~Central server needs to be always on~~
- ~~Single point of failure~~
- ~~Scalability issues~~
- How does KeyNote deal with anomalies

Firewall Policy Advisor

- Single rooted policy tree
- Simple visual representation of firewall rules
- Both inter- and intra-firewall anomalies

Firewall Policy Advisor



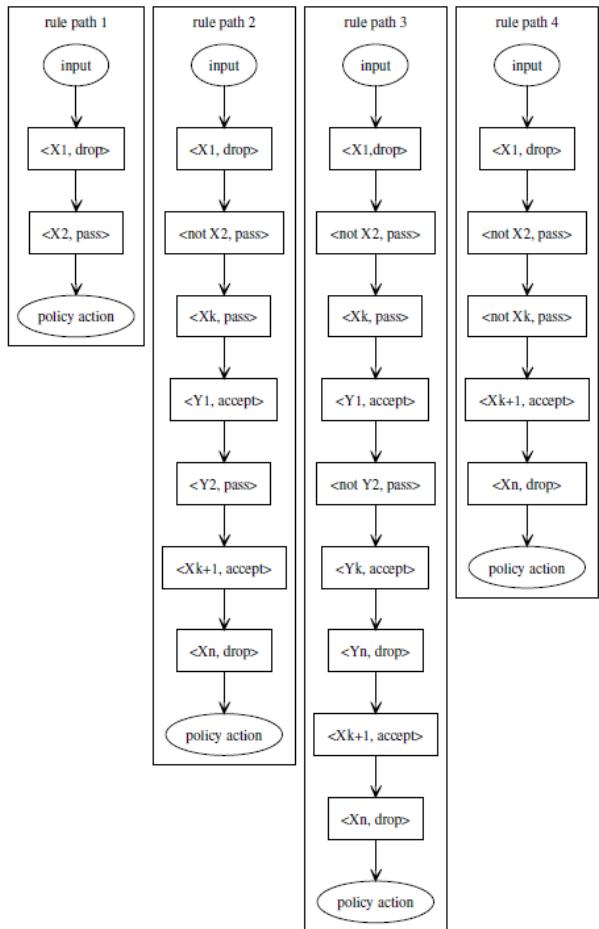
Firewall Policy Advisor

- Identifies best place to insert new rules
- Ensures removal without creating new anomalies
- Updating
 - Leaves it up to administrator

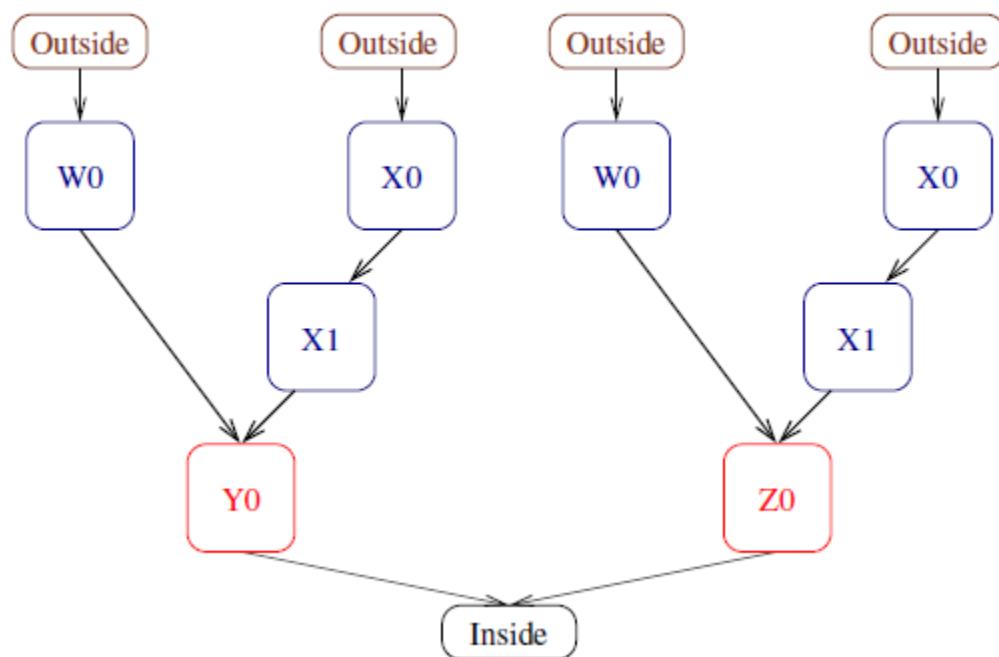
FIREMAN

- Detects inter- and intra-firewall inconsistencies
- Rule graph, local checks for intra-firewall
- ACL graph for inter-firewall
- Builds BDD for decision making

FIREMAN

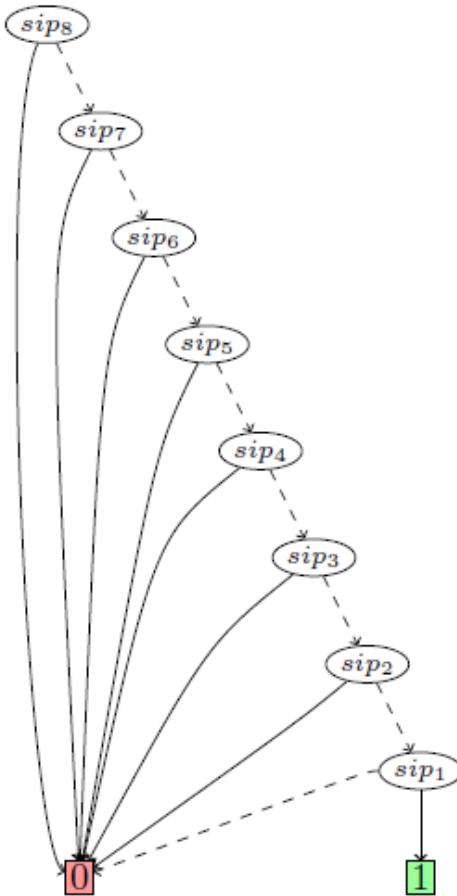


FIREMAN



FIREMAN

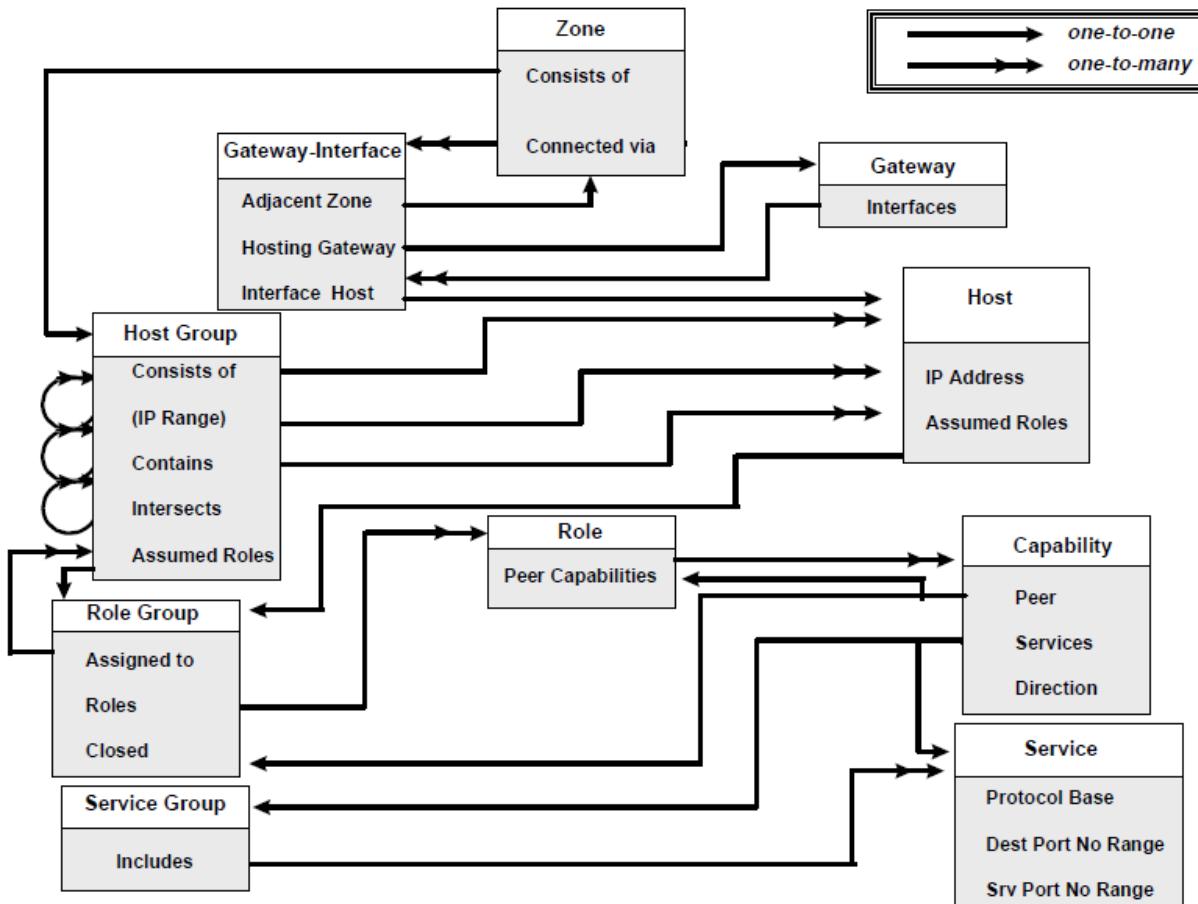
Serial	Parallel
$A = \bigcap_{acl \in n} A_{acl}$	$A = \bigcup_{acl \in n} A_{acl}$
$D = \bigcup_{acl \in n} D_{acl}$	$D = \bigcap_{acl \in n} D_{acl}$



Firmato

- Modular approach
- Define an entity-relationship model for representing
 - Security policy
 - Network topology
- Use a Model Definition Language (MDL) to translate into firewall configuration

Firmato



Firmato

- No anomalies, human error
- Single point of failure (controller)
- Doesn't discuss dynamic updates, and transmitting those updates to end points

Ponder

- Declarative and object-oriented language
- Related policies are grouped into roles, and interactions between roles are defined as relationships
 - This structure could better facilitate the reuse and flexibility of policies.

Ponder

```
inst auth+ switchPolicyOps {
    subject          /NetworkAdmin;
    target <PolicyT> /Nregion/switches;
    action           load(), remove(), enable(), disable() ;
}

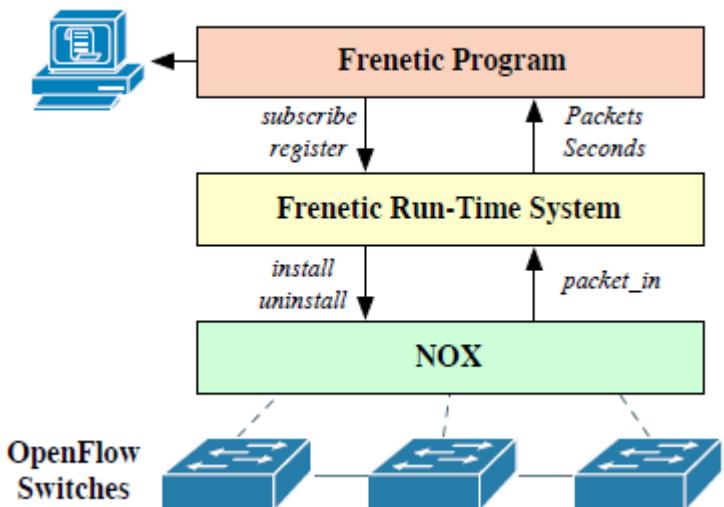
inst auth+ filter1 {
    subject  /Agroup + /Bgroup ;
    target   USAStaff - NYgroup ;
    action   VideoConf(BW, Priority)
        { in BW=2 ; in Priority=3 ; }      // default filter
        if (time.after("1900")) {in BW=3; in Priority = 1; }
}
```

REI

- Declarative language based on deontic logic
- Policies are defined as constraints over set of allowable actions on resources
- Ensures that actions are consistent and conflict-free

Frenetic

- Network programming language for defining high level policies
 - Provides a high-level abstraction of network function



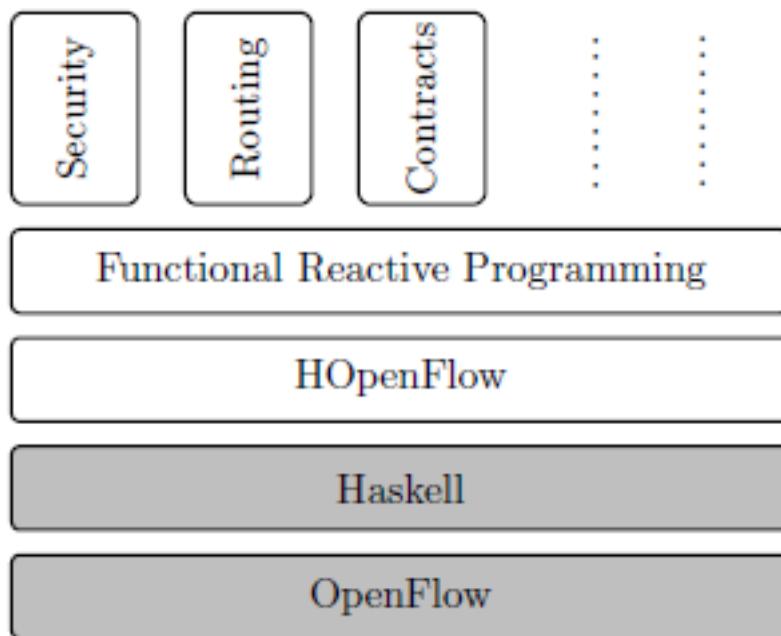
Frenetic

- Uses Python library
- Based on Functional Reactive Programming (FRP), so ECA
- Program parses every packet
 - Will **NOT** scale to networks of realistic size

Nettle

- Like Frenetic, Nettle is based on FRP
- Supports
 - network-wide control and domain-specific languages for different tasks
- Lacks Frenetic's support for overlapping module actions

Nettle

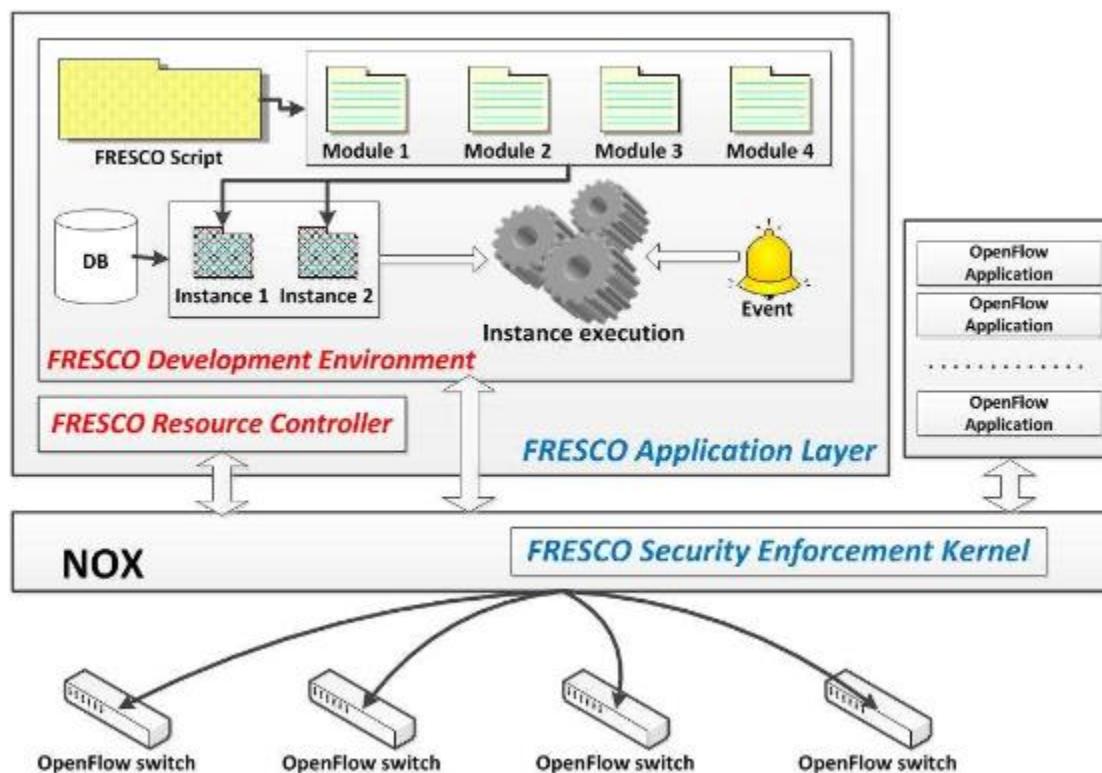


```
sf2 :: (HasSwitchEvents i, HasConsoleOutput o, HasSwitchCommands o) ⇒ SF i (Event o)
sf2 = proc i → do
  returnA ← packetInE i ⇒ λe → consoleOut (show e) ⊕ sendReceivedPacket e [flood]
```

FRESCO

- Allows the deployment of security services for OpenFlow
- Implemented as an application built on the NOX controller
- Comprises of Python scripts and API that allows the development of security services

FRESCO



Ira A. Fulton
Schools of Engineering

ARIZONA STATE UNIVERSITY

FRESCO

- Modules
 - Python objects that have input, output, action, event and parameters
- Development environment (DE)
 - Converts scripts into modules
 - Database management
 - Shares info across modules

FRESCO

- In addition to permit, drop and modify; set action enables rewriting of packet header fields.
 - Helps implement redirect, mirror and quarantine
 - Enables to isolate suspected malicious hosts / traffic
- Rule source identification
 - Allows applications to digitally sign flow rules enabling SEK to know if the origin of each flow

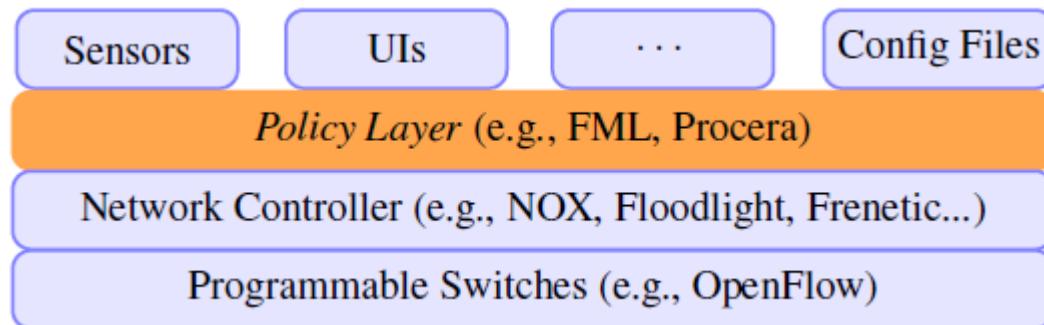
FRESCO

- Rule conflict detection
 - Inline rule conflict analysis algorithm (details needed)
 - Conflicts are resolved using a hierarchical authority model (based on origin as determined by digital signatures)

FRESCO

- Security Enforcement Kernel
 - Also the FortNOX project
- Applies a “lock” on rules placed by security applications
 - No rules which conflict can be inserted

Procera



```
proc world → do
    returnA ←
        λreq → if destIP req ‘inSubnet‘ ipAddr 128 36 5 0 // 24
            then allow else deny
```

Procera

- Based on FRP, like Nettle and Frenetic
 - Based on Haskell like Nettle
- Incorporates events that originate from sources other than ovSwitch
 - user authentications
 - time of day
 - bandwidth use

Procera

- Very similar to Frenetic
 - Can Frenetic take non-packet inputs?

Language Based Implementations

- Advantages
 - Predefined policy stored in a repository are retrieved and deployed immediately.
 - Formal foundations introduce automated analysis and verification, resulting in consistency.
 - Because of the abstraction from technical specifics, policies can be dynamically updated at run time, without it changing the underlying implementation

Comparison

	Paradigm	Anomaly Detection	Anomaly Correction	Distributed Environments	Dynamic topology	Automation
Ponder	ECA	✓	✗	✓	✗	✗
REI	CA	✓	✓	✗	✗	✓
Frenetic	ECA	✓	✗	✗	✓	✓
Nettle	ECA	✗	✗	✓	✓	✗
Fresco	ECA	✓	✗	✗	✓	✓
Proceria	ECA	✗	✗	✓	✓	✓

SCENARIO

Scenario

- Customer A: 10.0.0.0/16
- Customer B: 10.1.0.0/16
- Traffic between different customers is not permitted by default.

Src	Dst	Protocol	Src Port	Dst Port	Action
10.0.0.0/16	10.1.0.0/16	any	any	any	deny
10.1.0.0/16	10.0.0.0/16	any	any	any	deny

Scenario

- Customer A provides web hosting
 - Needs port 80 (in) to be allowed
 - Also needs public ports

Src	Dst	Protocol	Src Port	Dst Port	Action
any	10.0.0.0/16	TCP	any	80	permit
10.0.0.0/16	any	any	any		NAT

Scenario

- Customer B provides Dynamic DNS service, and requires port 53 (in) open for requests, and 8080 (in) open for updates

Src	Dst	Protocol	Src Port	Dst Port	Action
any	10.1.0.0/16	UDP	any	53	permit
any	10.1.0.0/16	UDP	any	8080	permit

- Customer A can now access / update
 - Security policy violation

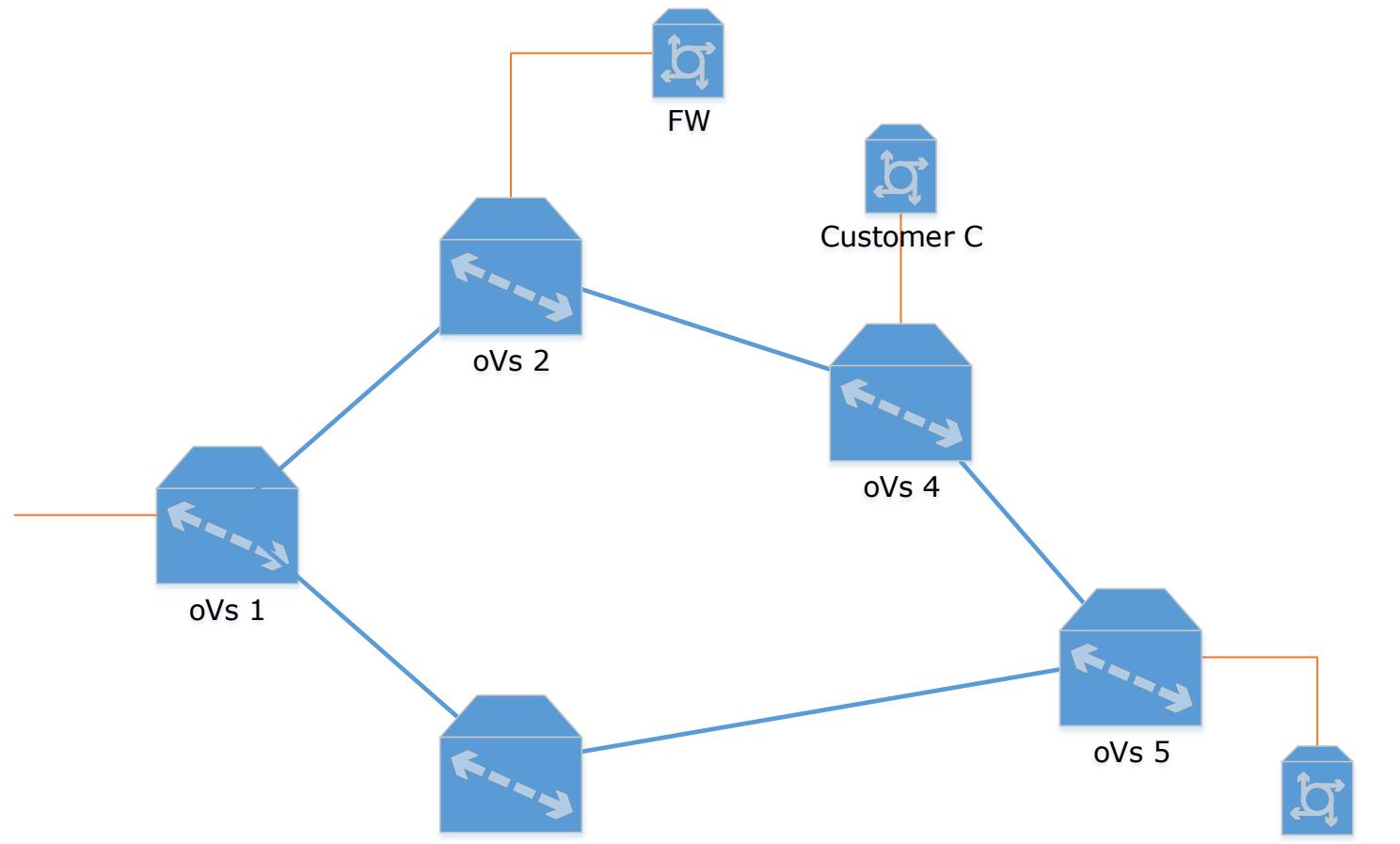
Scenario

- Customer C
 - Offers security as a service
 - Provide technical support to A and B
 - Inserts rules allowing SSH traffic only from C to A and B

Provides IDS/IPS services

Src	Dst	Protocol	Src Port	Dst Port	Action
10.10.0.0/16	10.0.0.0/16	TCP	any	22	permit
10.10.0.0/16	10.1.0.0/16	TCP	any	22	permit
any	10.10.0.0/16	TCP	any	any	permit

Scenario



Scenario

- Customer C
 - Offers security as a service
 - Provide technical support to A and B
 - Inserts rules allowing SSH traffic only from C to A and B

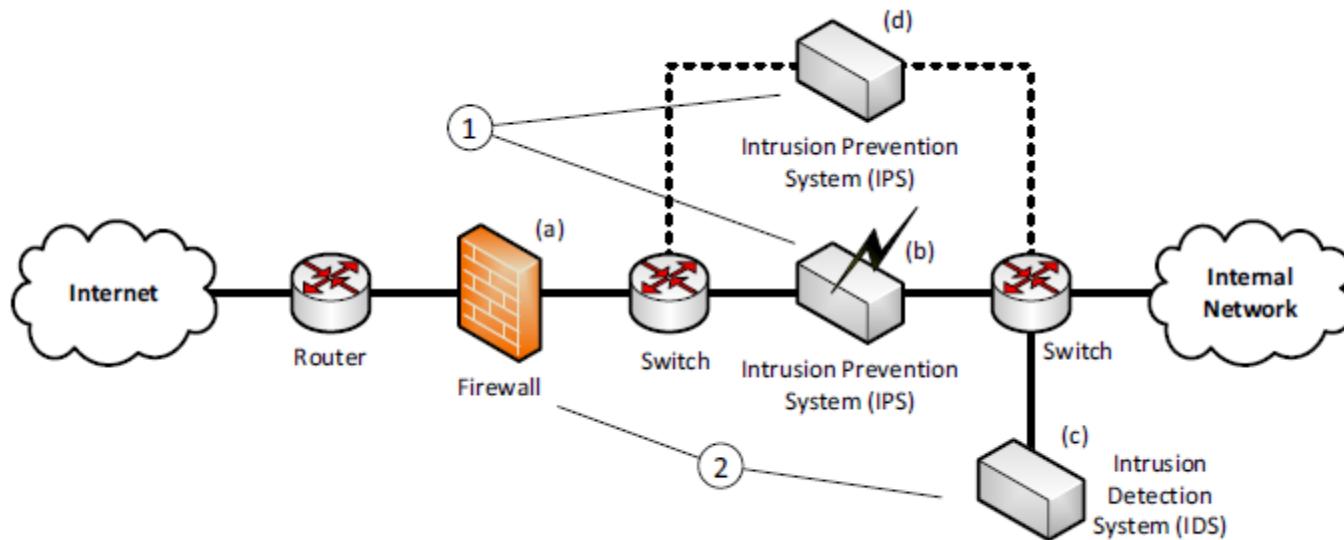
Provides IDS/IPS services

Src	Dst	Protocol	Src Port	Dst Port	Action
10.10.0.0/16	10.0.0.0/16	TCP	any	22	permit
10.10.0.0/16	10.1.0.0/16	TCP	any	22	permit
any	10.10.0.0/16	TCP	any	any	permit
cc:cc:cc:cc:cc:cc	aa:aa:aa:aa:aa:aa				permit
cc:cc:cc:cc:cc:cc	bb:bb:bb:bb:bb:bb				permit

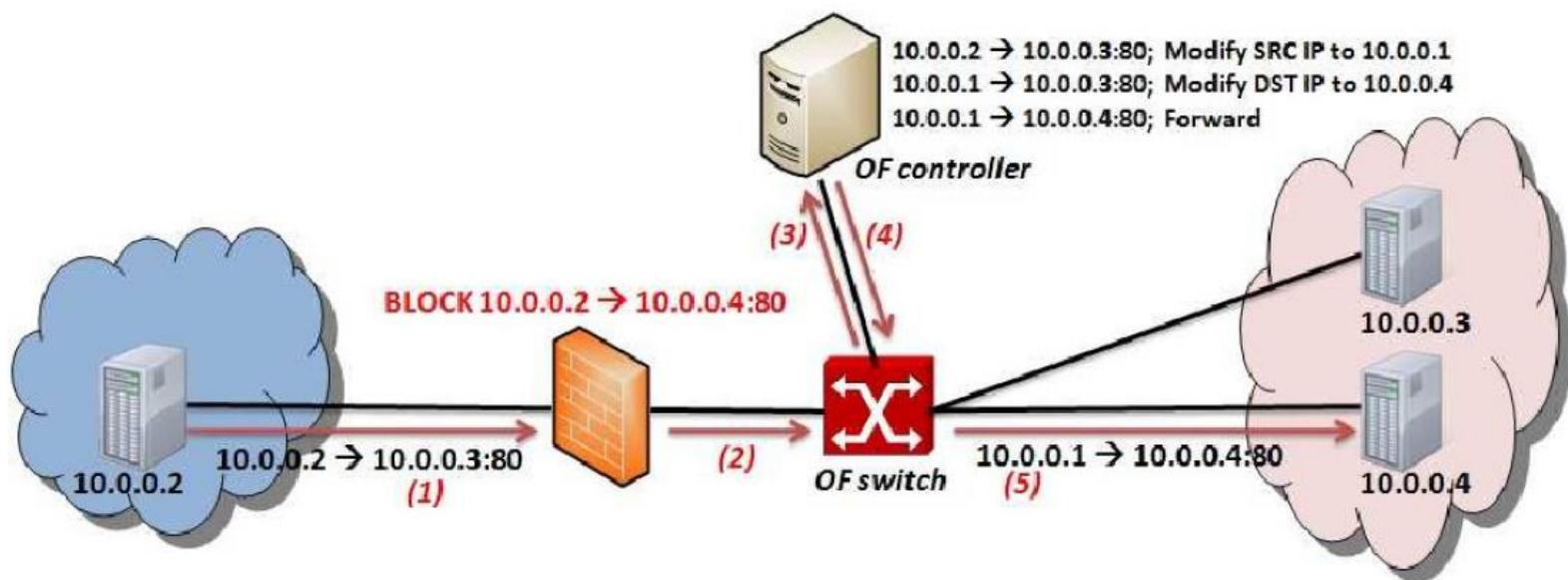
Scenario

- All unwanted traffic to A and B *could* still reach them, since the inserted rules counter the ones existing prior.

Motivating Scenario



Motivating Scenario



PATRICIA TRIE

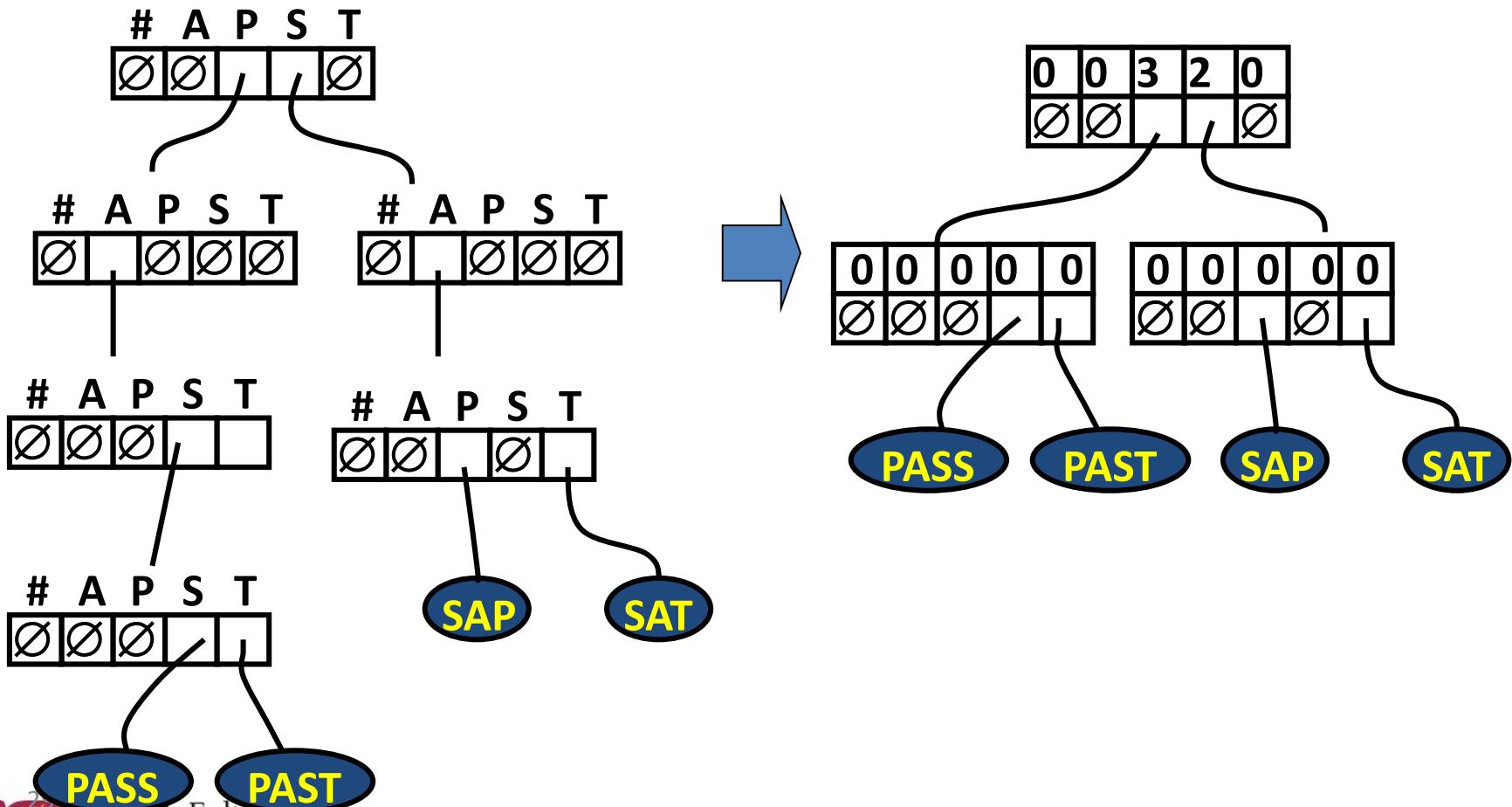
Patricia Trie

- Data structure for storing a set of strings.
- Each edge of the tree is labeled with a character.
- Each leaf node corresponds to the stored string, which is a concatenation of characters on a path from the root to this node.

Patricia Trie

- Collapse chains of nodes that have only one child
- For each branch indicate how many characters should be skipped (i.e. what the length of the collapsed chain is)

Patricia Trie

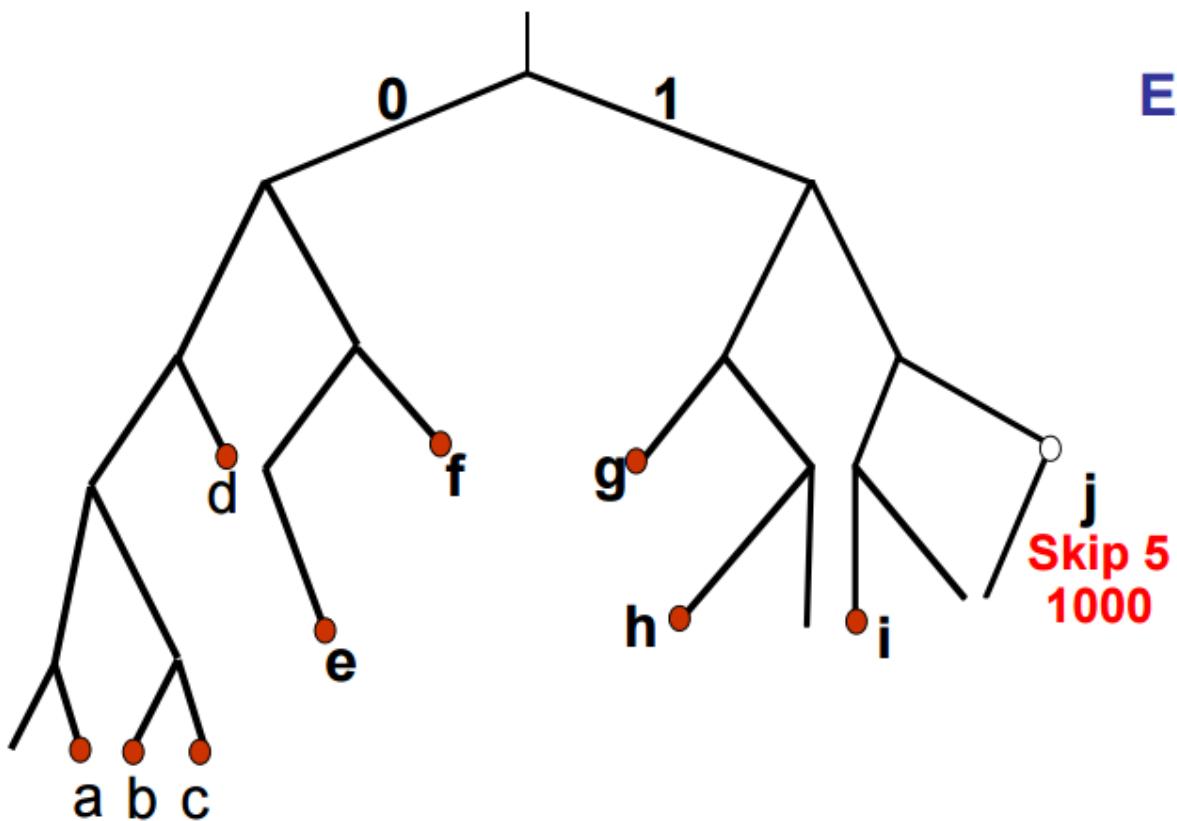


Patricia Trie

- Why Patricia Trie?
 - Faster than linear scan
 - Proportional to number of bits in the address
 - Trie is not high
 - Space overhead problem is not valid when we know the maximum storage required

Left-ptr	Bit #	Right-ptr
----------	-------	-----------

Patricia Trie



Example Prefixes

- a) 00001
- b) 00010
- c) 00011
- d) 001
- e) 0101
- f) 011
- g) 100
- h) 1010
- i) 1100
- j) 11110000

Complexity

- D-bit prefixes: $O(D)$ lookup, $O(ND)$ storage and $O(D)$ update complexity
 - Since tree is binary, average tree height for N keys is $O(\log D)$
- Worst Case:
 - $O(ND)$, where N is the number of flow entries and D is the number of prefix bits

SECURITY THROUGH SDN

Secure & Resilient Networking

- Programmable Network MTD
 - Network based countermeasures for Moving Target Defense
 - Requires framework that supports dynamic changes
 - Address changes
 - Topology changes

Secure & Resilient Networking

- Detect vulnerabilities
 - Minimize attack graph
- Detect compromise
- Traffic engineering
 - To minimize exposed attack surface
 - Optimize user experience

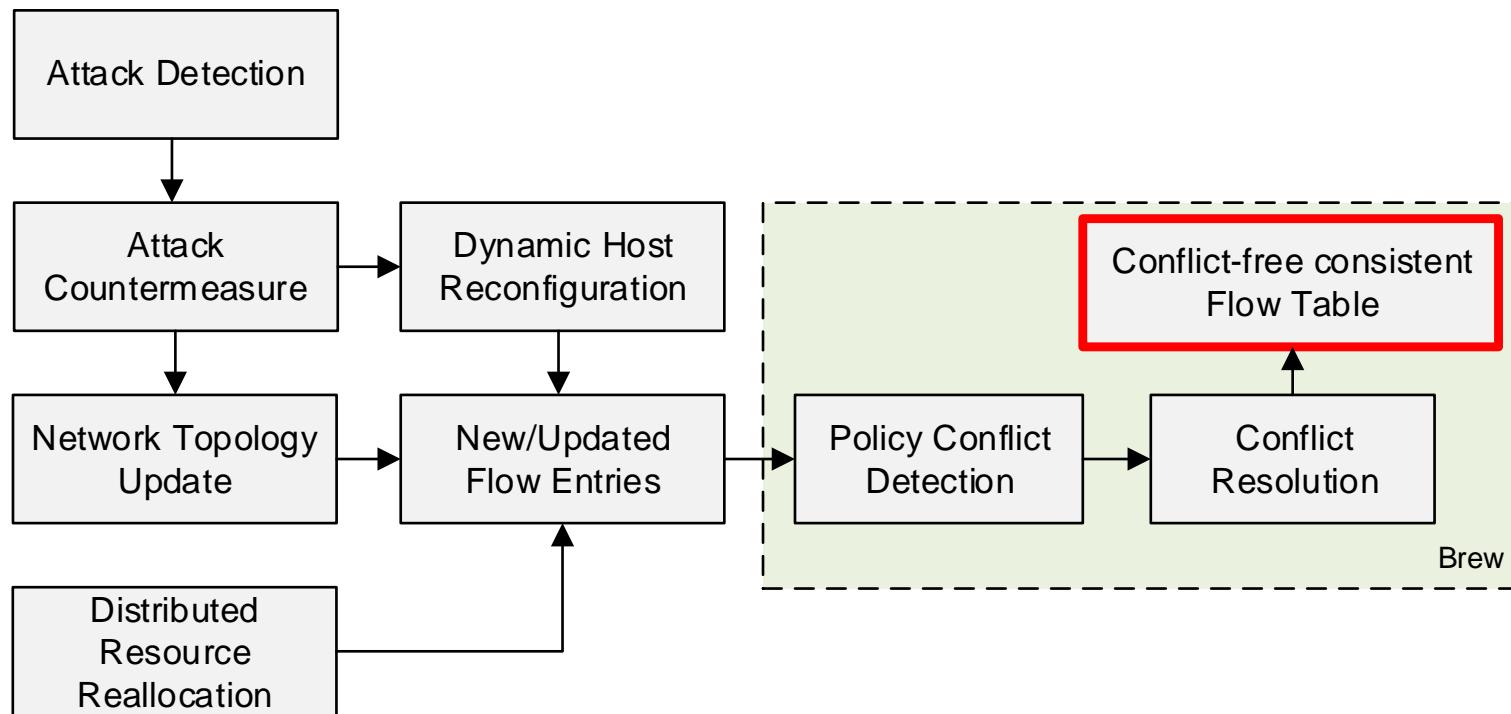
SDN for Security

- Moving Target Defense
 - Dynamically change network upon detection of an attack
 - Set-Field actions mean you can use L2, L3 and L4 to help move the target
 - Insider attack
 - Detect compromised host
 - Host can be quarantined instantaneously

SDN for Security

- Moving Target Defense
 - Honeypots
 - IP address mutation scheme to mask real IP addresses*
- Countermeasure actions can include
 - Reconfigure network
 - Spawn/kill servers
 - Traffic engineering

Brew



Enhancing Security in SDN

- Add diversity to systems
 - Avoiding shared vulnerabilities
- In-built trust between controllers, devices and admin stations
- Self-identify network anomalies and dynamically update configuration

Research Plan

- Policy conflict detection
 - Cross-layer conflict detection
 - And resolution
 - Conflict detection in distributed environments
- Make traffic shaping flow rule analysis more robust
- Validation
- Parallel processing
- Optimization of rules
 - Positioning
 - Adaptive prioritization
- Stateful flow rule considerations
- Incorporate with Ankur's work

Research Plan

- Verification if the firewall is doing what is required of it
 - Quantifying information loss using header space analysis
- Integrating functionalities of IPS/IDS
- Establishing that device configurations meet management goals
 - Adapting work in the area of regulatory compliance
 - Describe business SLAs using specification language
 - Use logical formalism or deontic logic
- Quantizing effect of resolution strategies
- Different flavors
 - Host based SDN firewalls
 - Mobile (lightweight) SDN firewall
- Address scalability and dynamicity

Collaboration Plan

- Involve traffic shaping as a MTD countermeasure
 - Identify conflicts
- Software reliability metric
 - Use as tiebreaker for conflict resolution
 - Identify potential malicious core

Secure & Resilient Networking

- Ensure policy compliance
 - Detect conflicts
- Ensure reliability of software modules
 - Reliability metric?

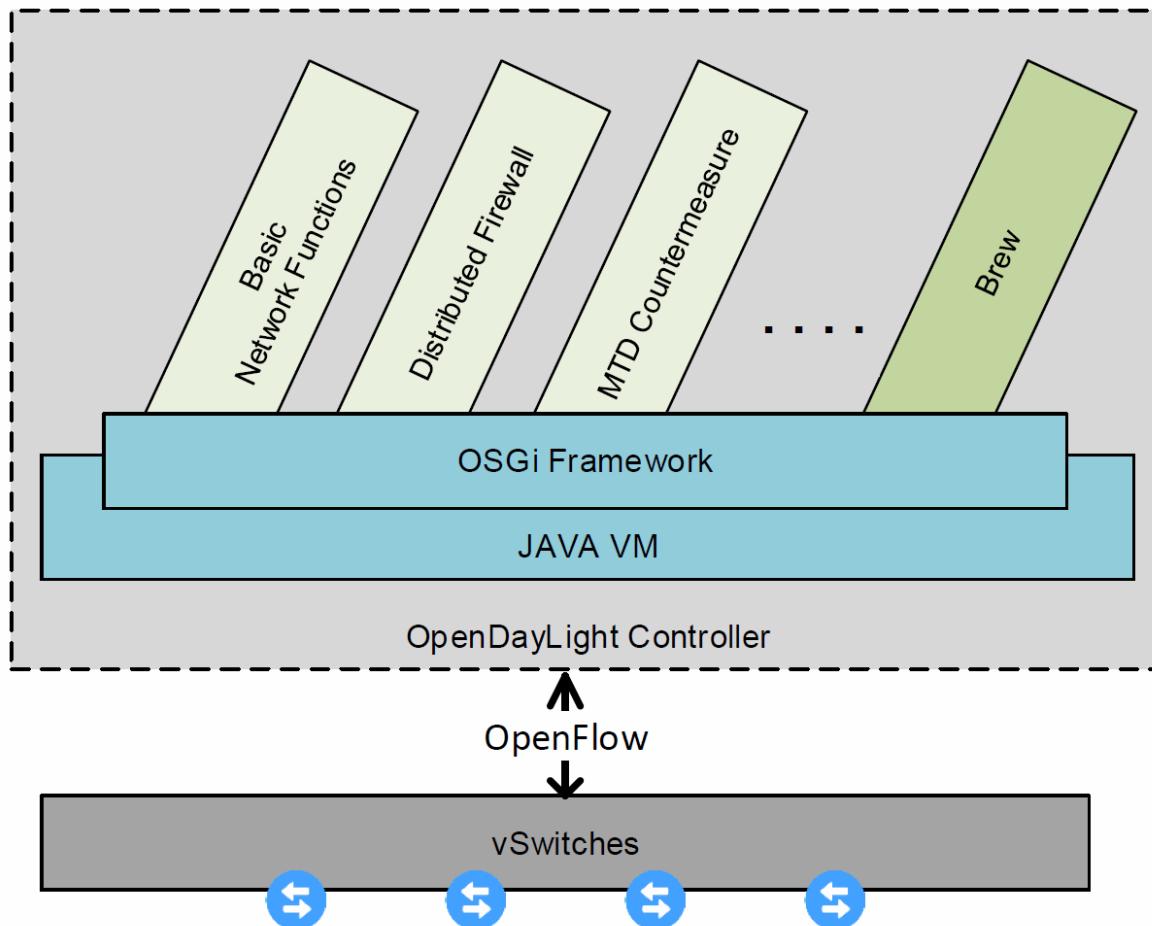
Datacenter Implementation

- Approach #1: Periodic snapshots
 - When compromised
 - Force whitelisted applications to freeze sessions by using flow control
 - Revert to last “good” snapshot
 - If snapshotting begins at time n , and takes time d , restored state will lose information between n and $n+d$

Datacenter Implementation

- Approach #2: High Availability
 - Maintain 3 virtual machines as concurrent clusters in active/standby/backup mode
 - Synchronize whitelisted apps
 - When compromised
 - Migrate connections to standby
 - Change standby to active, and backup to standby
 - Rejuvenate active, and bring back as backup
 - Resource intensive

Brew



Artisan

- What makes this different
 - Toe the line between being completely unaware, and being the MS Office paper clip
 - Extendible framework to be used as plugin for multiple SDN controllers

Motivation

- SDN Wide Area Network
 - High latency between controller and some switches
- Single centralized controller
 - Limits network growth
- HA, FT...

References

- E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 10, pp. 2069–2084, 2005.
- H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building robust firewalls for software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 97–102.
- C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker *et al.*, "Composing software defined networks." in *NSDI*, 2013, pp. 1–13.
- D. R. Morrison, "Patricia - Practical algorithm to retrieve information coded in alphanumeric," *Journal of the ACM (JACM)*, vol. 15, no. 4, pp. 514–534, 1968.
- K. Poornaselvan, S. Suresh, C. D. Preya, and C. Gayathri, "Efficient IP lookup algorithm," *Strengthening the Role of ICT in Development*, p. 111, 2007.
- P. Gupta and N. McKeown, "Algorithms for packet classification," *Network, iEEE*, vol. 15, no. 2, pp. 24–32, 2001.
- F. B. Schneider, "Least privilege and more," in *Computer Systems*. Springer, 2004, pp. 253–258.