FRESCO: Modular Composable Security Services for Software-Defined Networks

Seungwon Shin, Phil Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, and Mabry Tyson

SUCCESS LAB, Texas A&M and SRI International

Contents

- Background
  - SDN and OpenFlow
- FRESCO
  - Design
  - Use cases
  - Evaluation
- Summary

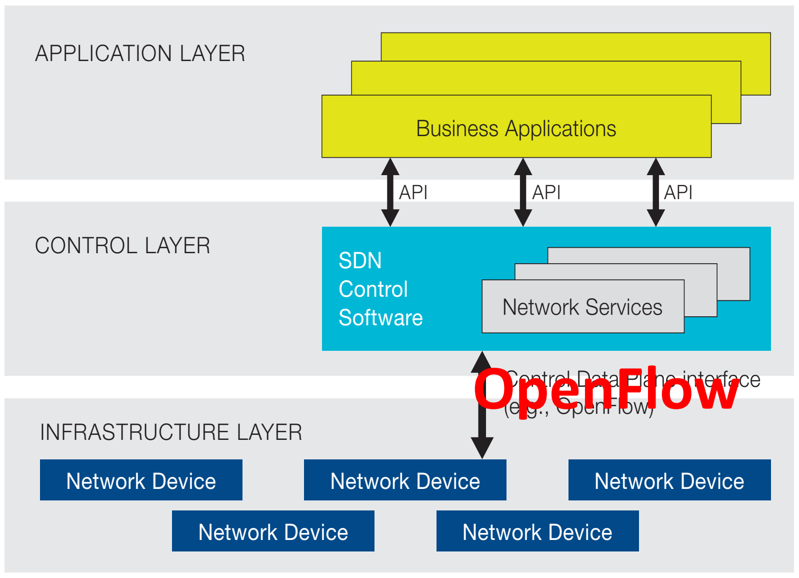
Problem of Legacy Network Devices

- Complicated and Closed platform
  - Complicated S/W and ASIC
  - Vendor specific
- Inflexible: Hard to modify (nearly impossible)
- Non-extensible: Difficult to support emerging technologies
  - E.g., VM mobility

- New proposal: Software Defined Networking
  - Separate the control plane from the data plane

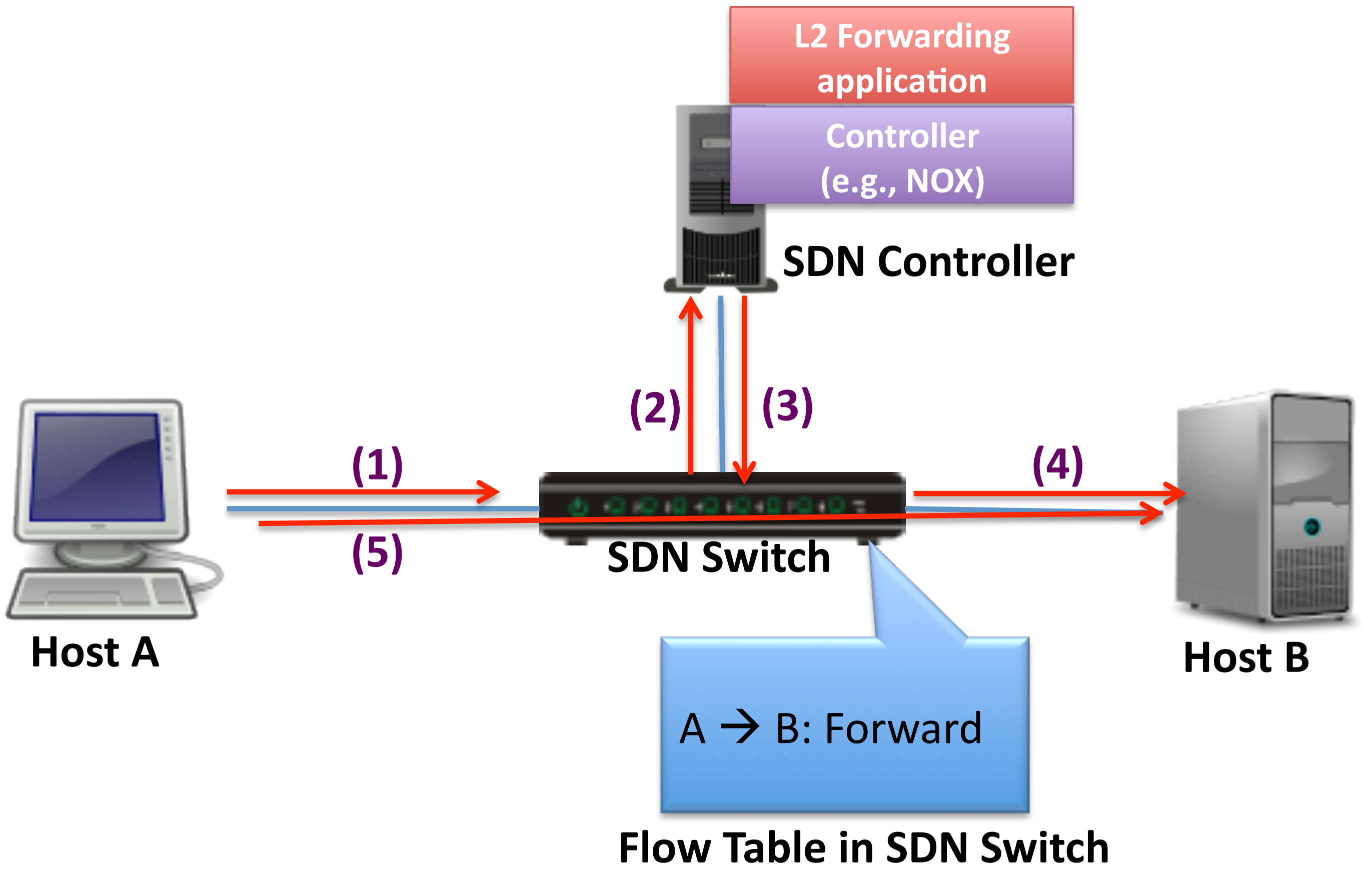
Software Defined Networking (SDN)

- Three layers
  - Application layer
    - Application part
    - Implements logic
  - Control layer
    - Kernel part
    - Runs applications
  - Infrastructure layer
    - Data plane
    - Network switch or router



From Open Networking Foundation

SDN Operation



Killer Application of SDN?

- Reducing energy in data center networks
- Dynamic virtual machine migration in cloud networks
- .... diverse network applications

- What about security?
  - **Can SDN enable new capabilities to improve network security?**

Exemplar SDN Security Apps

- Security functions can be applications of SDN
  - Firewall
  - DDoS detection
  - Scan detection
  - Reflector net
  - Tarpit
  - Dynamic quarantine
  - and more…

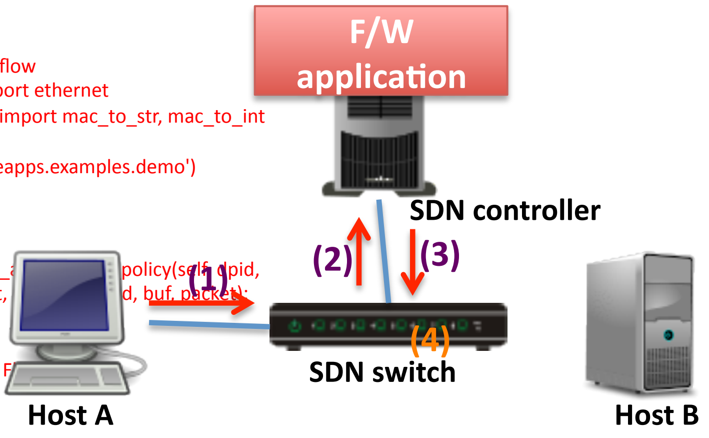
```
import logging

from nox.lib.core import *
import nox.lib.openflow as openflow
from nox.lib.packet.ethernet import ethernet
from nox.lib.packet.packet_utils import mac_to_str, mac_to_int

log = logging.getLogger('nox.coreapps.examples.demo')


class demo(Component):
def __init__(self, ctxt):def create_a                policy(self, dpid,
policy_type, outport_find, inport,                d, buf, packet):

if outport_find == 0:
print 'DBG: No Specific Out Port: F
if policy_type == 'ARP':
print 'DBG: ARP packet’
self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)
elif policy_type == 'REQ':
attrs = extract_flow(packet)
attrs = {core.IN_PORT:inport,
core.DL_TYPE:ethernet.IP_TYPE,
core.NW_PROTO:ipv4.ipv4.TCP_PROTOCOL
core.NW_SRC:'10.0.0.2'}
```



(1) Host A sends packet to Host B
(2) Switch asks a controller form a flow rule
(3) F/W application decides to block the packet
(4) Switch drops this packet

SDN Security App Development Challenges

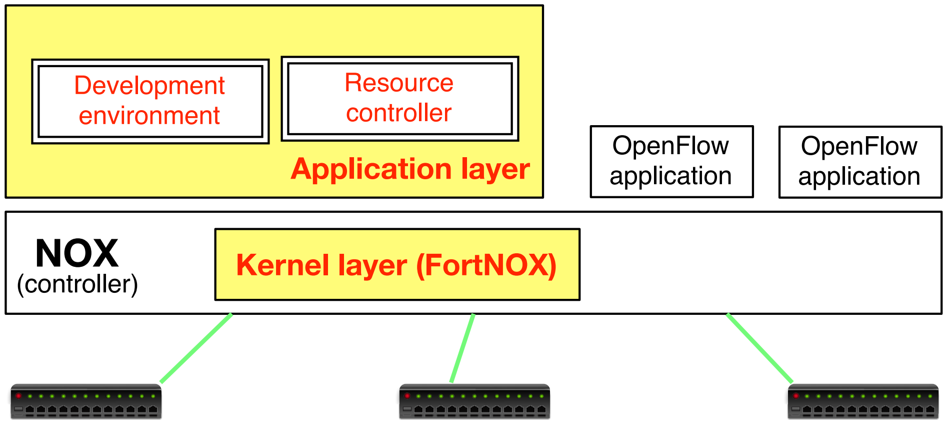
- However, it is not easy to create security apps in SDN
  - Security service creation and composition challenge
    - How do we simplify development of security applications?
  - Information deficiency challenge
    - E.g., TCP session, network status
  - Threat response translation challenge
    - How do we enforce security policies to the network devices?

FRESCO

- FRESCO is a new frame work that
  - Provides a new development environment for security applications
  - Effectively manages shared resources among security applications
  - Simplifies deployment of security policies
    - provides a set of 7 new intelligent security action primitives
      - E.g., block, deny, allow, redirect, and quarantine

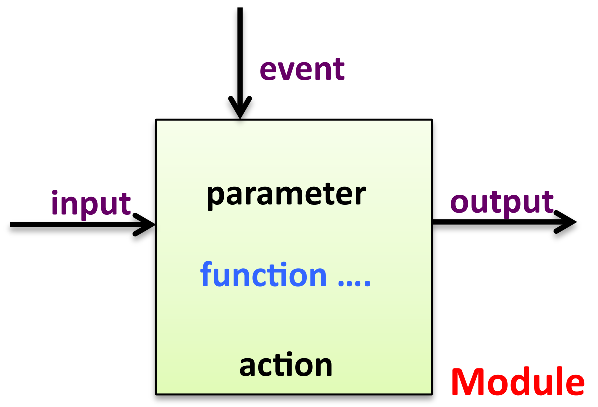
Architecture

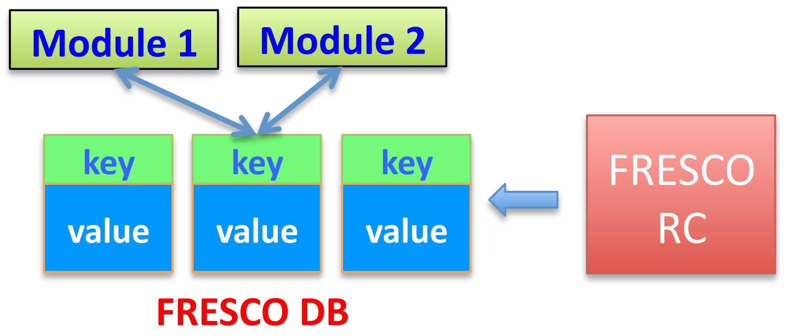
- Component
  - **Application layer**
    - **Development env. (DE)**
    - **Resource controller (RC)**
  - Kernel layer
    - Security enforcement kernel
    - FortNOX
      - paper in HotSDN 2012



Development Environment

- FRESCO Module
  - Basic operation unit
- FRESCO DB
  - Simple database
    - (key,value) pairs
- FRESCO script
  - Define interfaces
  - Connect multiple modules





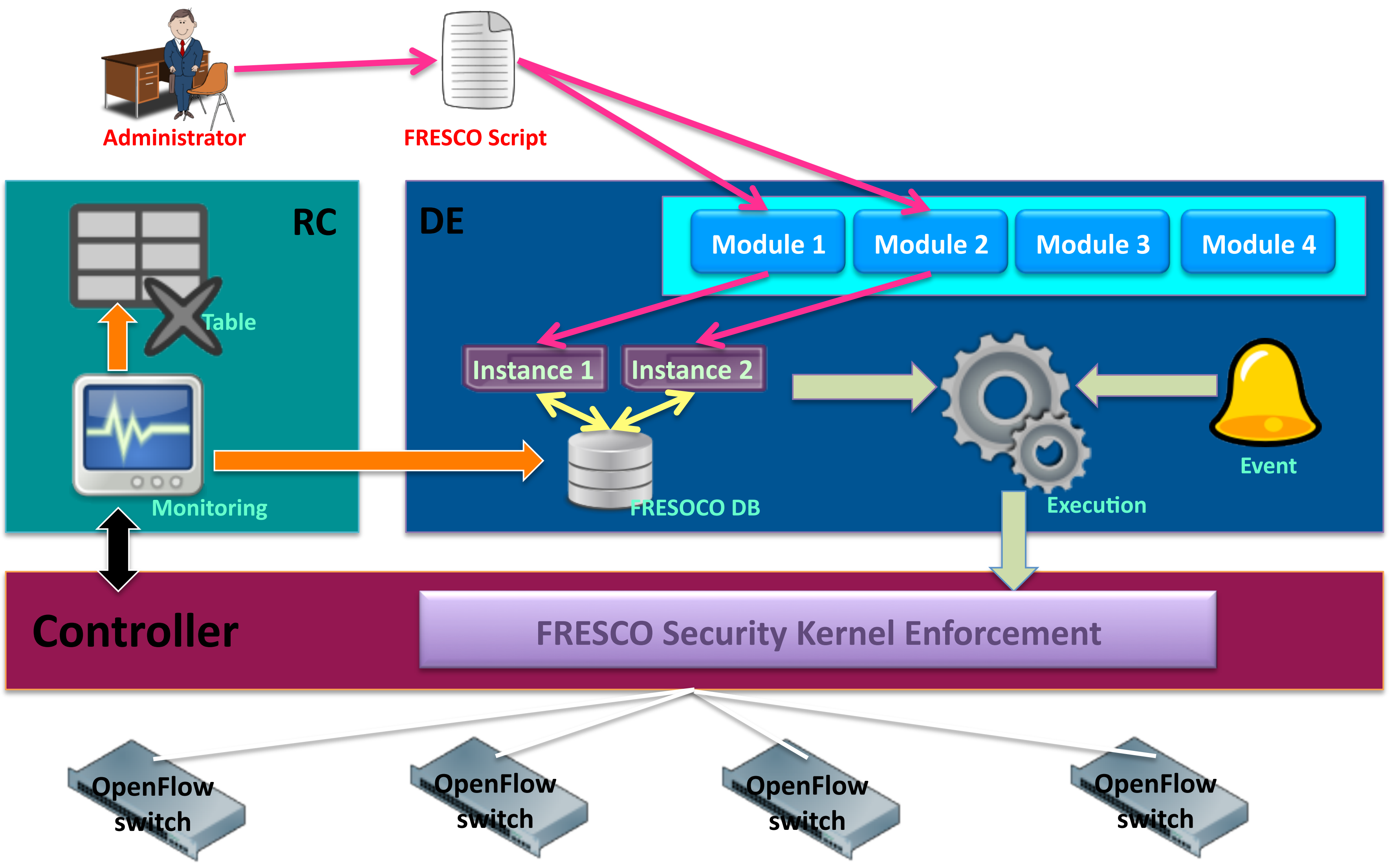
Development Environment

- FRESCO script
  - Format
    - **Instance name (# of input) (# of output)**
    - **type**: class of this module
    - **input**: input for this module
    - **output**: output of this module
    - **parameter**: define some variables
    - **event**: trigger a module
    - **action**: conduct this action

```
port_comparator (1)(1) {
   type: Comparator
   event: PUSH
   input: destination_port
   output: comparison_result
   parameter: 80
   action: -
}
```

Inspired by Click Modular Router
Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. Proceedings of SOSP '99

Operational Scenario

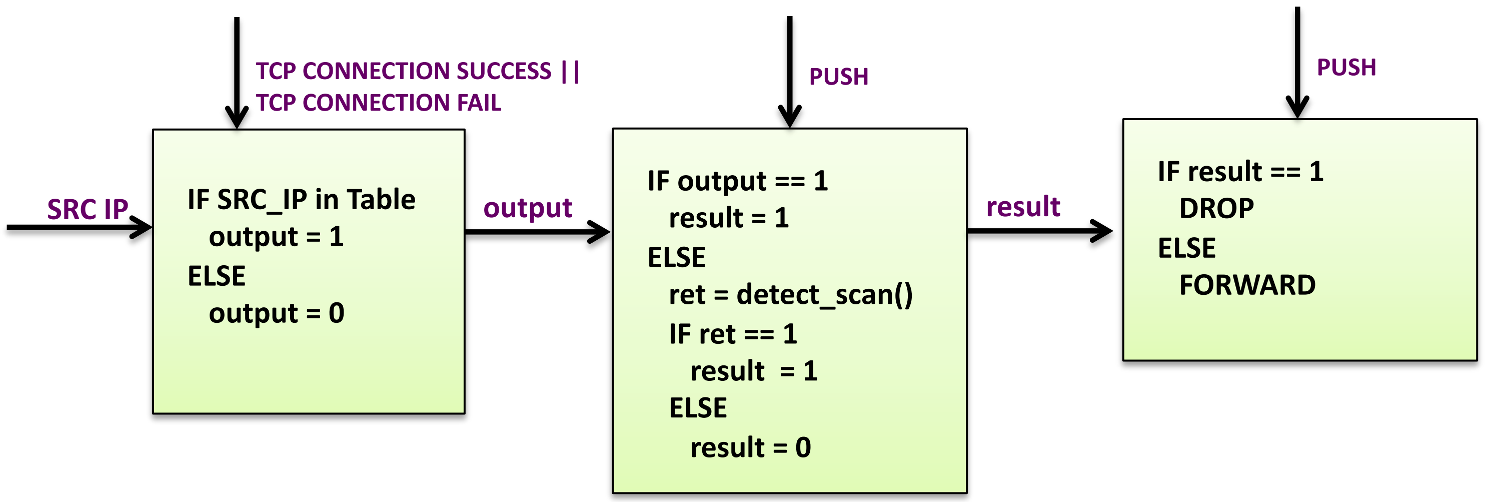


Implementation

- NOX (open source OpenFlow controller) based
  - Development environment
    - NOX based Python application
  - Resource controller
    - NOX based Python application
  - Security enforcement kernel
    - Modify NOX (C++)

Example: Scan Detection

- Steps
  - Check blacklist → Threshold based scan detection → Drop or Forward



```
IF SRC_IP in Table
  output = 1
ELSE
  output = 0
```

```
IF output == 1
  result = 1
ELSE
  ret = detect_scan()
  IF ret == 1
    result  = 1
  ELSE
    result = 0
```

```
IF result == 1
  DROP
ELSE
  FORWARD
```

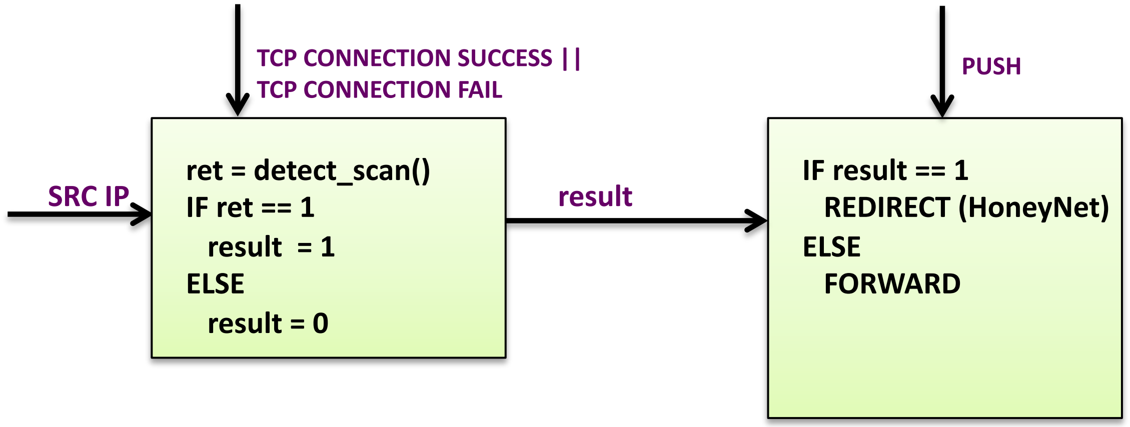
```
blacklist_check (1)(1) {
   type: TableLookup
   event: TCP_CONNECTION_FAIL,
         TCP_CONNECTION_SUCCESS
   input: SRC_IP
   output: blacklist_out
   parameter: NONE
   action: NONE
}
```

```
scan_detect (1)(1) {
   type: ScanDetect
   event: PUSH
   input: blacklist_out
   output: detect_result
   parameter: NONE
   action: NONE
}
```

```
final_action (1)(0) {
   type: ActionHandler
   event: PUSH
   input: detect_result
   output: NONE
   parameter: NONE
   action: detect_result == 1
            ?DROP:FORWARD
}
```

Example: Reflector Net

- Confuse network scan attackers
- Steps
  - Threshold based scan detection → Reflect or Forward



```
ret = detect_scan()
IF ret == 1
  result  = 1
ELSE
  result = 0
```

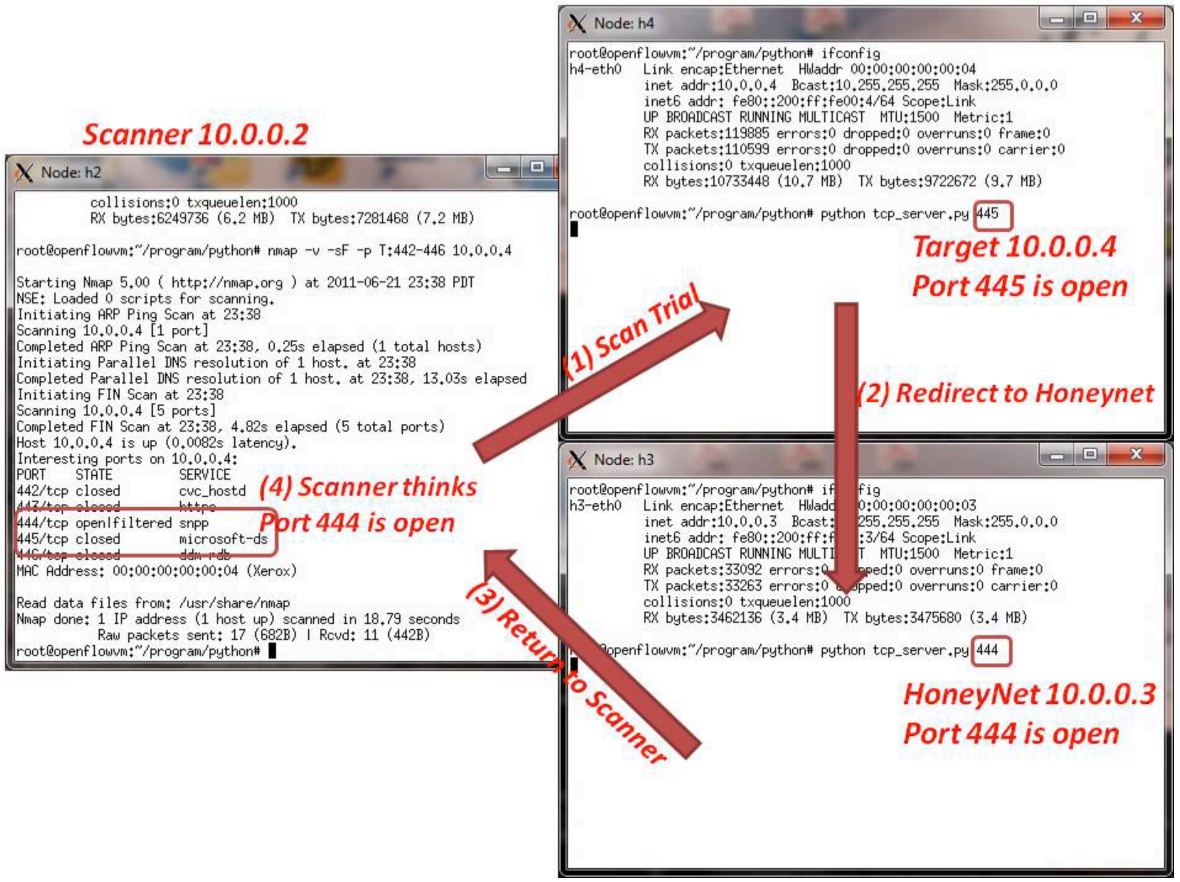
```
IF result == 1
  REDIRECT (HoneyNet)
ELSE
  FORWARD
```

```
scan_detect (1)(1) {
   type: ScanDetect
  event: TCP_CONNECTION_FAIL,
             TCP_CONNECTION_SUCCESS
input: blacklist_out
   output: detect_result
   parameter: NONE
   action: NONE
}
```

```
final_action (1)(0) {
   type: ActionHandler
   event: PUSH
   input: detect_result
   output: NONE
   parameter: NONE
   action: detect_result ==  ?REDIRECT(10.0.0.3):FORWARD
}
```

Example: Reflector Net

- Test result



```
        collisions:0 txqueuelen:1000
        RX bytes:6249736 (6.2 MB)  TX bytes:7281468 (7.2 MB)

root@openflowvm:~/program/python# nmap -v -sF -p T:442-446 10.0.0.4

Starting Nmap 5.00 ( http://nmap.org ) at 2011-06-21 23:38 PDT
NSE: Loaded 0 scripts for scanning.
Initiating ARP Ping Scan at 23:38
Scanning 10.0.0.4 [1 port]
Completed ARP Ping Scan at 23:38, 0.25s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:38
Completed Parallel DNS resolution of 1 host. at 23:38, 13.03s elapsed
Initiating FIN Scan at 23:38
Scanning 10.0.0.4 [5 ports]
Completed FIN Scan at 23:38, 4.82s elapsed (5 total ports)
Host 10.0.0.4 is up (0.0082s latency).
Interesting ports on 10.0.0.4:
PORT    STATE         SERVICE
442/tcp closed        cvc_hostd
443/tcp closed        https
444/tcp open|filtered snpp
445/tcp closed        microsoft-ds
446/tcp closed        ddm-rdb
MAC Address: 00:00:00:00:00:04 (Xerox)

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 18.79 seconds
           Raw packets sent: 17 (682B) | Rcvd: 11 (442B)
root@openflowvm:~/program/python#
```

```
root@openflowvm:~/program/python# ifconfig
h4-eth0   Link encap:Ethernet  HWaddr 00:00:00:00:00:04
          inet addr:10.0.0.4  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::200:ff:fe00:4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:119885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:110599 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10733448 (10.7 MB)  TX bytes:9722672 (9.7 MB)

root@openflowvm:~/program/python# python tcp_server.py 445
```

```
root@openflowvm:~/program/python# if   fig
h3-eth0   Link encap:Ethernet  HWadd  0:00:00:00:00:03
          inet addr:10.0.0.3  Bcast:   255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::200:ff:f   :3/64 Scope:Link
          UP BROADCAST RUNNING MULTI   T  MTU:1500  Metric:1
          RX packets:33092 errors:0   pped:0 overruns:0 frame:0
          TX packets:33263 errors:0   pped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3462136 (3.4 MB)  TX bytes:3475680 (3.4 MB)

    openflowvm:~/program/python# python tcp_server.py 444
```

Examples

More Examples in the paper

- BotMiner

- P2P Plotter

And etc..

Evaluation

Source code length comparison

Algorithm	Implementation		
	Standard	OpenFlow	FRESCO
TRW-CB	1,060	741	66 (58 + 8)
Rate Limit	991	814	69 (61 + 8)

Results for Standard and OpenFlow are obtained in the following paper,
S. A. Mehdi, J. Khalid, and S. A. Khayam.
Revisiting Traffic Anomaly Detection Using Software Defined Networking, In Proceedings of Recent Advances in Intrusion Detection, 2011.

Flow rule setup time

	NOX	Simple Flow Tracker	Simple Scan Detector	Threshold Scan Detector	BotMiner	P2P Plotter Detector
Time (ms)	0.823	1.374	2.461	7.196	15.461	11.775

Please refer to our paper for the explanation of each test case

Summary and Future Work

- FRESCO
  - Create security applications easily
  - Deploy security applications easily
  - Focus on creating security applications

- Future work
  - Port FRESCO to other controllers for open source release
    - E.g., POX or Floodlight
  - Create more modules (now 16 basic modules)

More Information

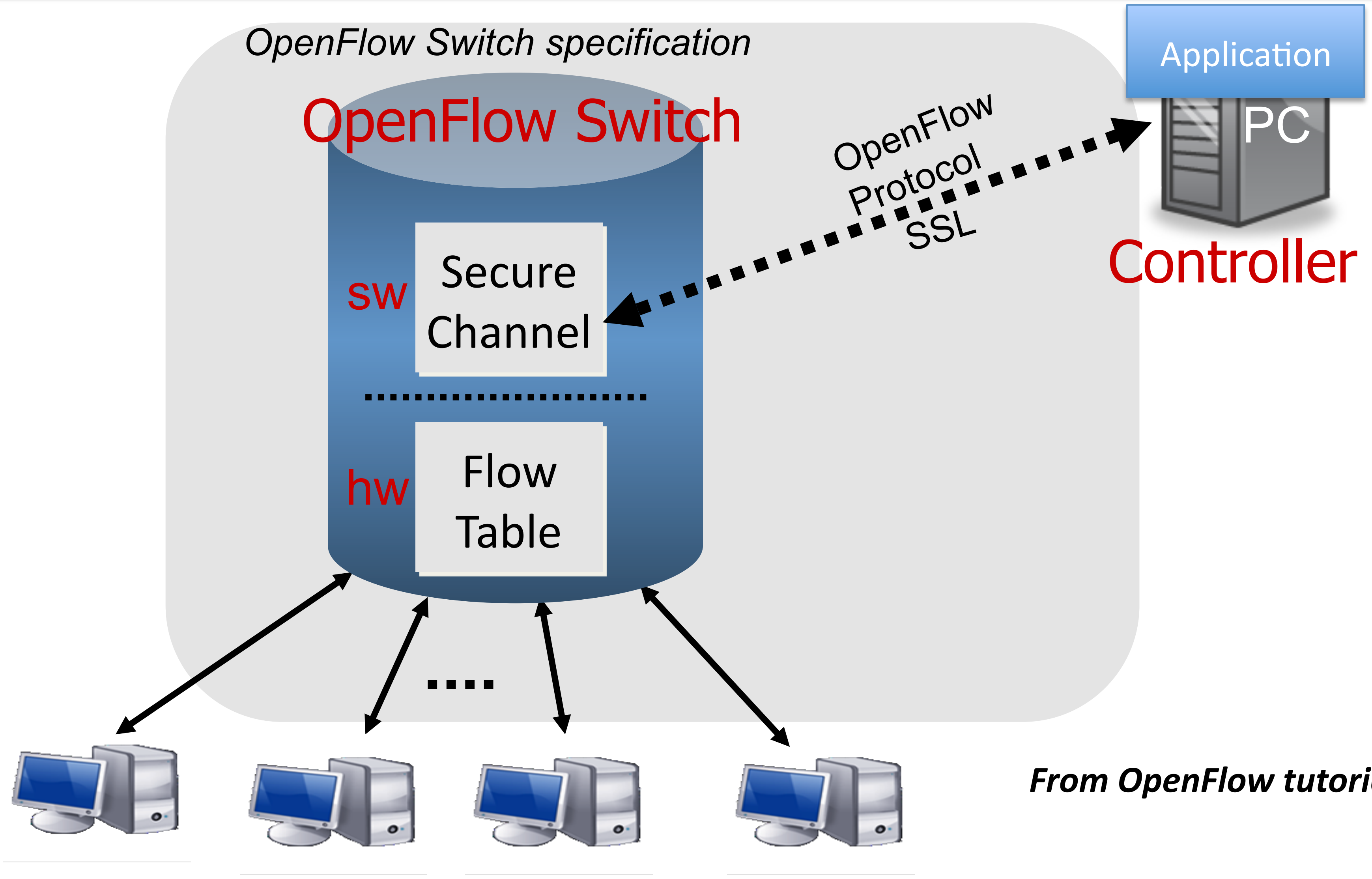
www.openflowsec.org

- Demo movies
  - Security Constraints Enforcement
  - Reflector Nets
  - Automated Quarantine

Thank you,
Question ?

Optional

OpenFlow Architecture



From OpenFlow tutorial

Resource Controller

- Monitor OpenFlow switches
  - Check current status of security policies
    - How many security policies are active
    - How many packets are matched to policies
    - And etc..
- Remove some old policies
  - Save some space for FRESCO security constraints
  - Garbage collection