Software-defined Networking
Session V

Introduction to Software-defined Networking

Block Course – Winter 2015/16

David Koll

SDN Security

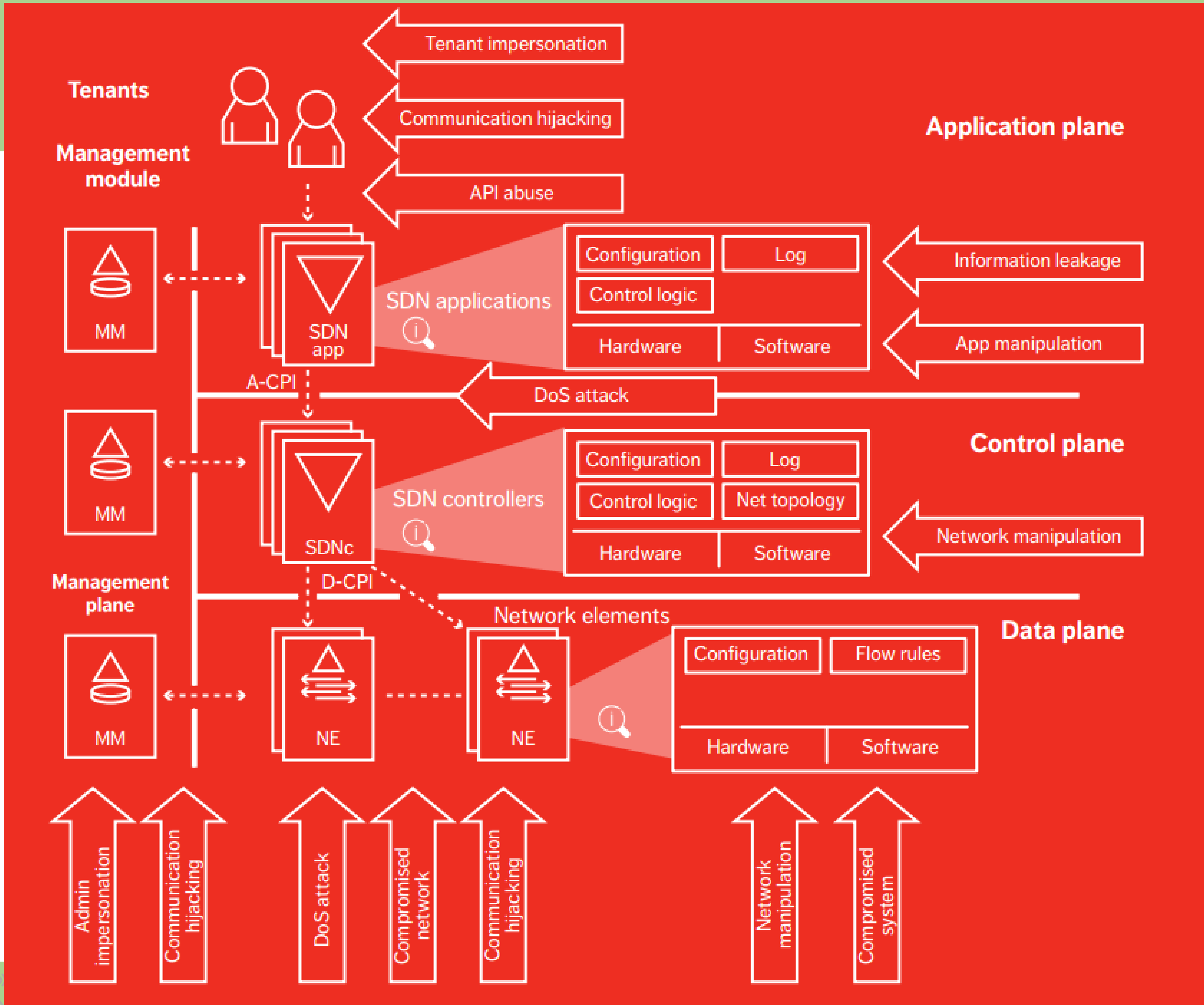
SDN and Security

“The first thing we hear from customers is, 'We see security as the No. 1 inhibitor to SDN’ “

- Matthew Palmer, Co-Founder SDN Central

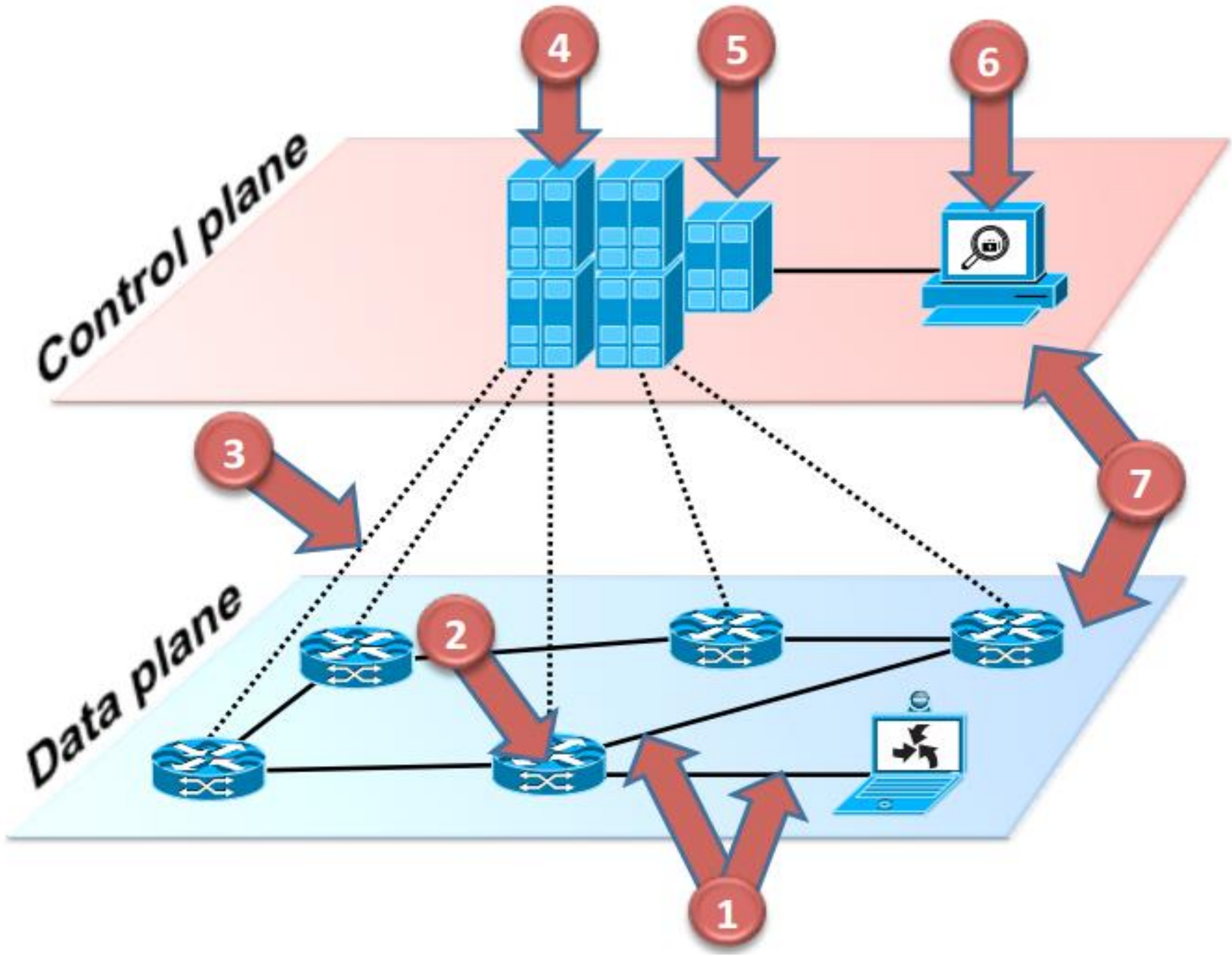
SDN and Security

- SDN research: lots of new concepts
- A lot of functionality implemented in software
  - E.g., controller, virtual switches, SDN applications
- Many proposals to use SDN to *increase security*
- But what about protection against attacks *on SDN*?
  - Software components = easy targets!?



Ericsson Technology Review, August 2015

SDN Security Threats



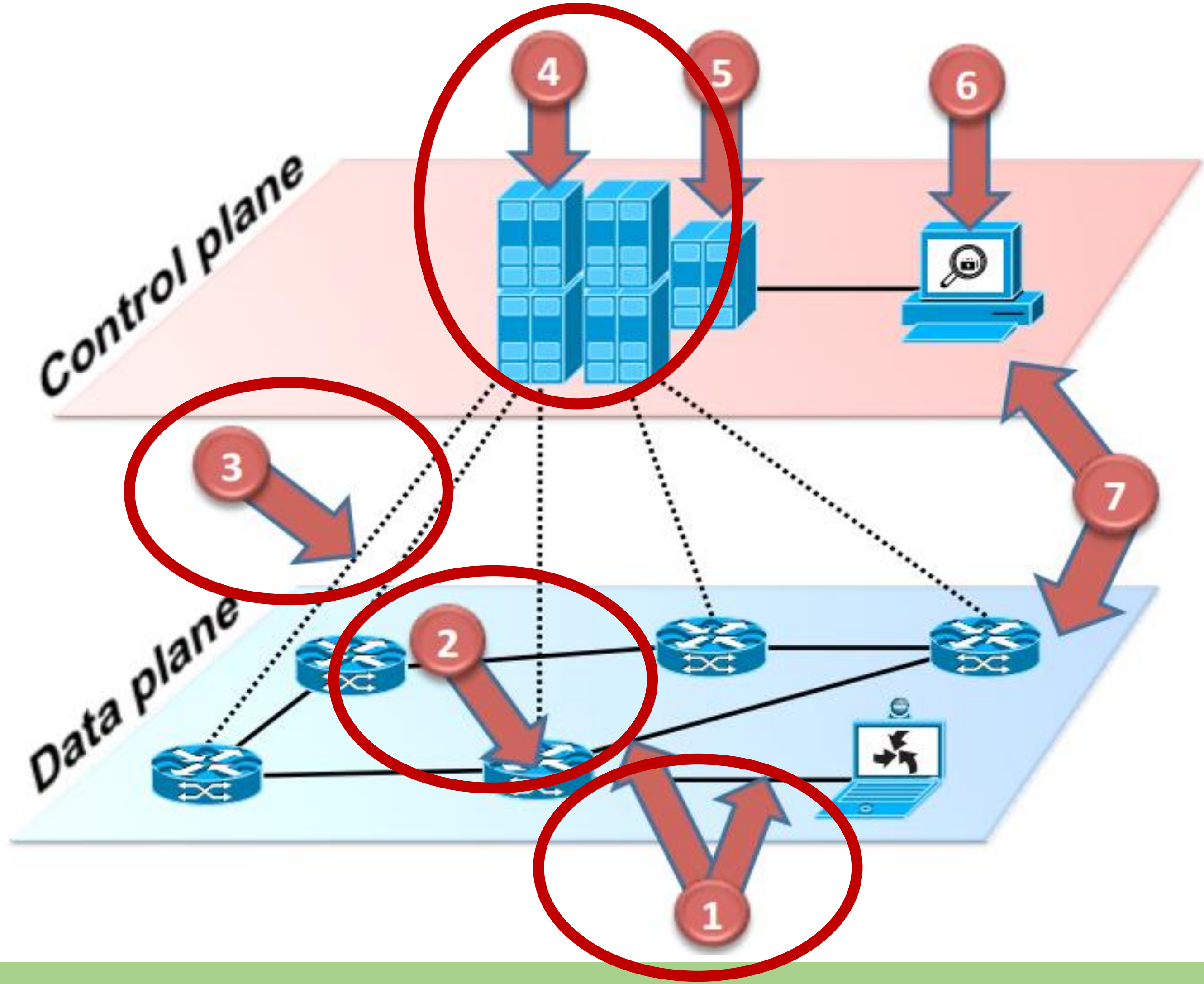
Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE* 103.1 (2015): 14-76.

SDN Security Threats

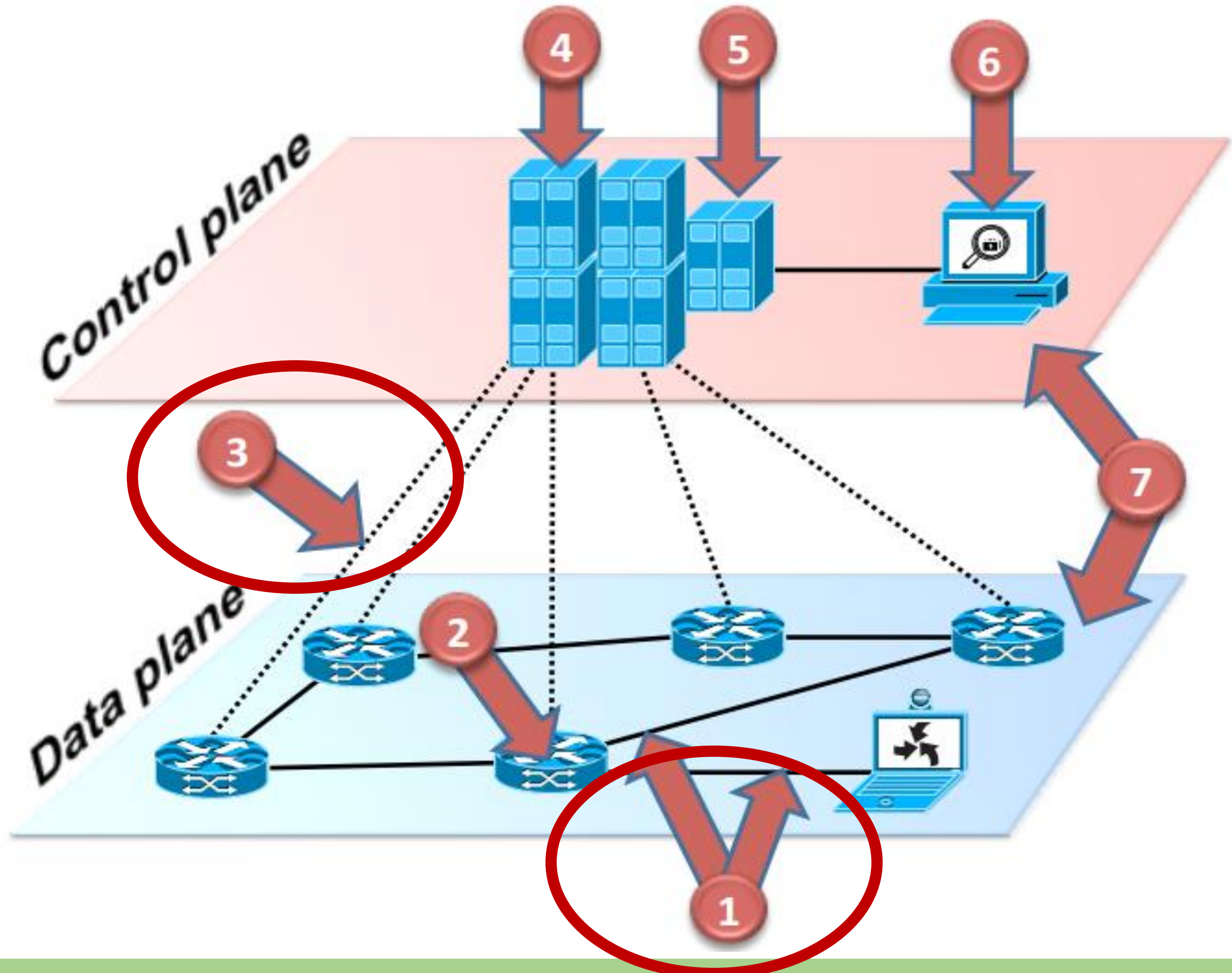
Threat vectors	Specific to SDN?	Consequences in software-defined networks
Vector 1	no	Open door for DDoS attacks.
Vector 2	no	Potential attack inflation.
Vector 3	yes	Exploiting logically centralized controllers.
Vector 4	yes	Compromised controller may compromise the entire network.
Vector 5	yes	Development and deployment of malicious applications on controllers.
Vector 6	no	Potential attack inflation.
Vector 7	no	Negative impact on fast recovery and fault diagnosis.

1

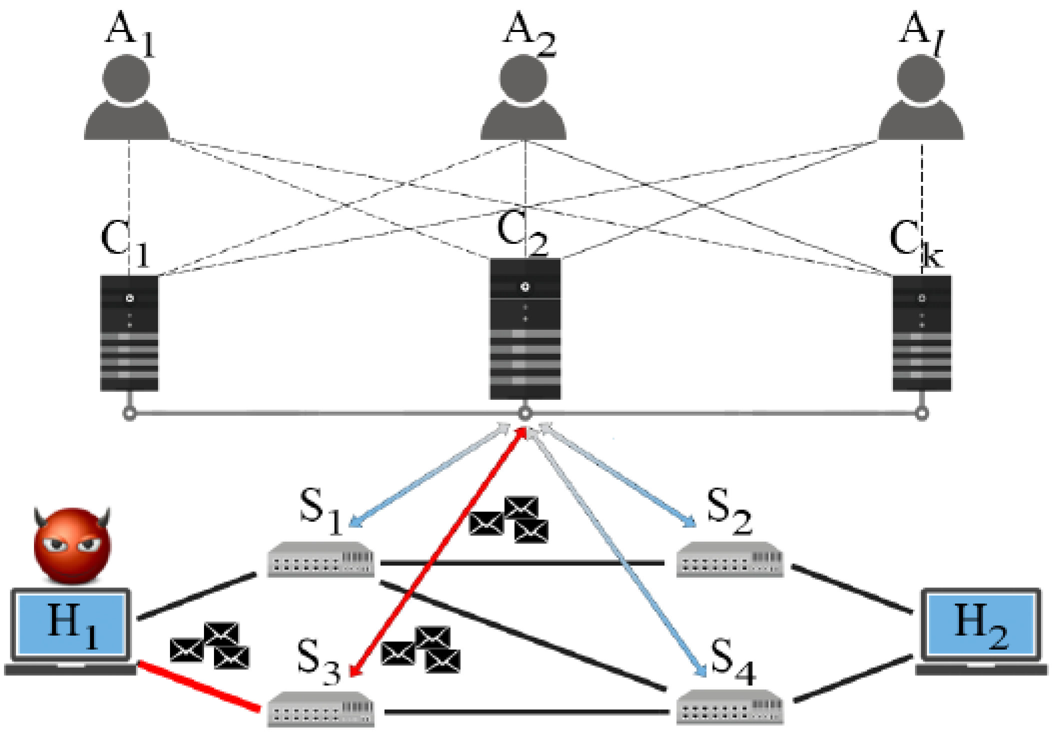
SDN Security Threats



Compromised End Hosts



Compromised End Hosts



(a) Compromised host

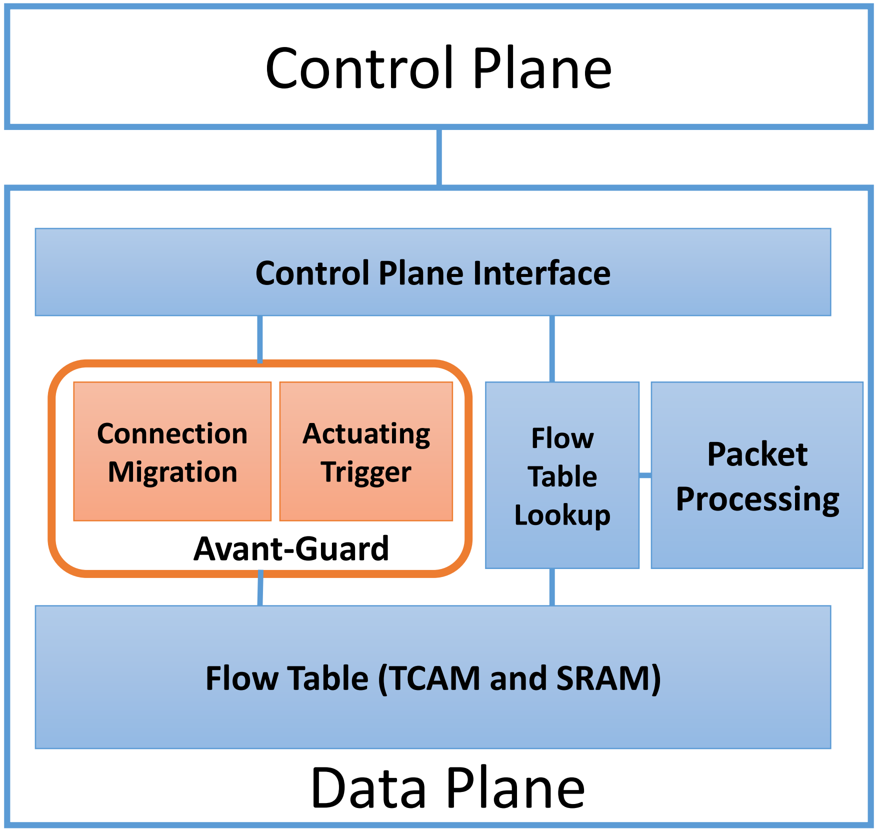
Prasad, A. S, Koll, D., and Fu, X. "On the Security of Software-Defined Networks." *EWSDN 2015*.

Attack Scenario

- OpenFlow property: all unknown packets are forwarded to controller
- Various attacks on network resources possible:
  - Exhaust control plane bandwidth
  - Exhaust processing capability of controller
  - Exhaust memory at switches

Avant-Guard [1]

- Security extension to the OpenFlow data plane
  - Connection migration
    - Differentiate attacker con-nections from benign ones
  - Actuating trigger



[1] Shin, Seungwon, et al. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks." *Proceedings of the 2013 ACM SIGSAC CCS.*

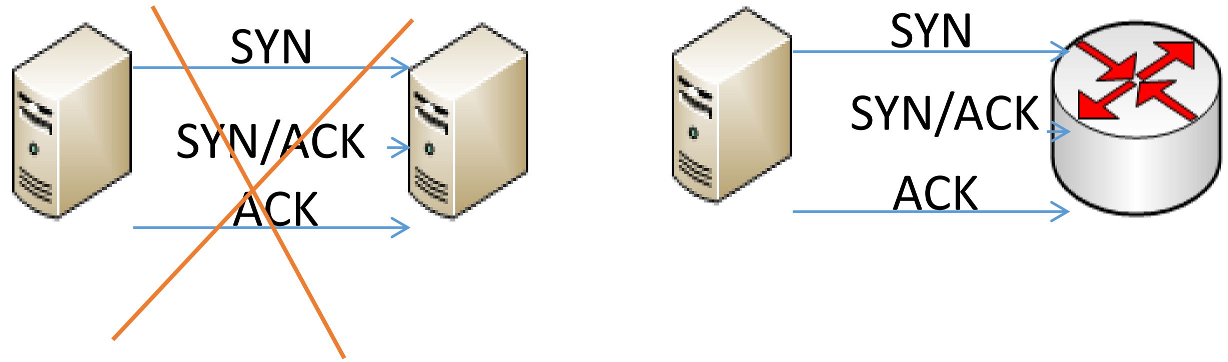
Connection Migration - Idea

- Inspired by TCP SYN Cookie

- Concept
  - TCP connection will start from a SYN packet, and an initiator will wait for TCP SYN/ACK packet

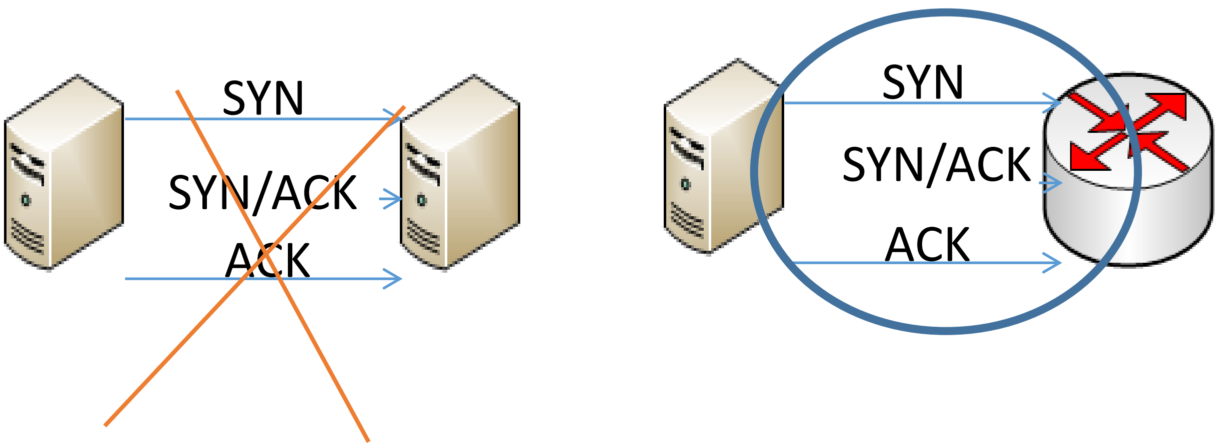
  - Often exploited by attackers to launch DoS attack

  - How about treating this TCP-handshake at **network devices instead of target hosts**



Connection Migration - Idea

- **Basic principle:**
  - Data plane proxies connection establishment
  - Only forwards successful flow requests to control plane



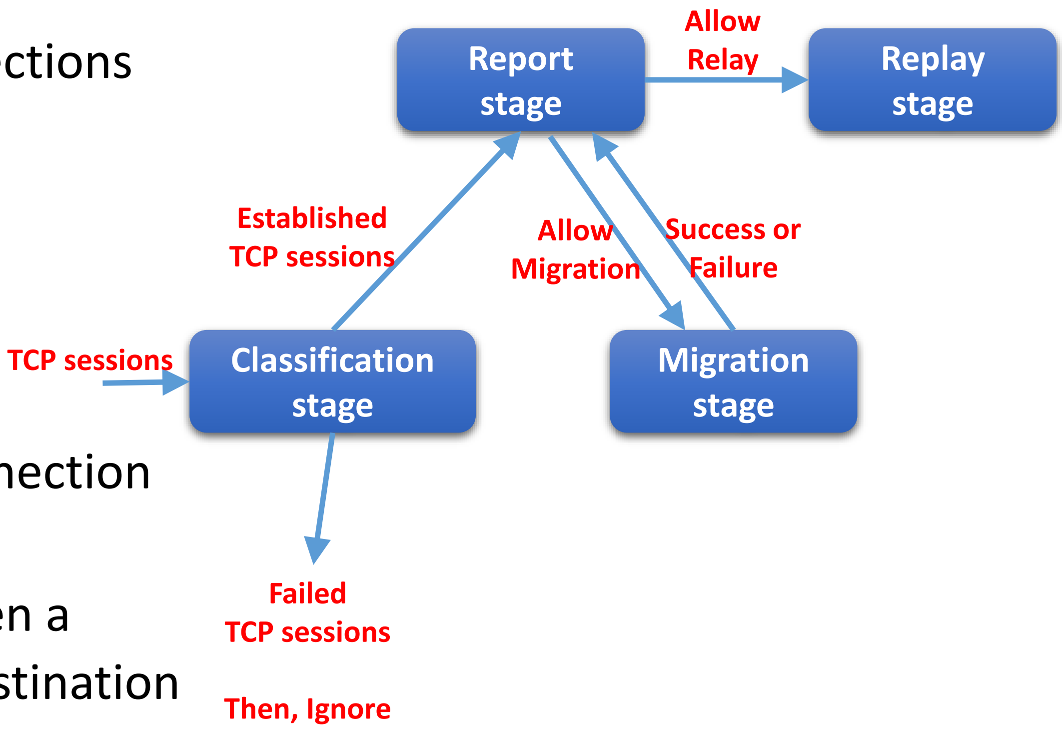
Connection Migration – Access Table

- List of visiting clients
  - Format
    - Client IP address: # of TCP connection trials
      - # of TCP connection trials include wrong trials
    - Simple data structure : 6 bytes (4 bytes for IP and 2 bytes for counter)
- Overhead
  - 1,000,000 client IP addresses → less than 6 MB of memory
- A controller application can read this table

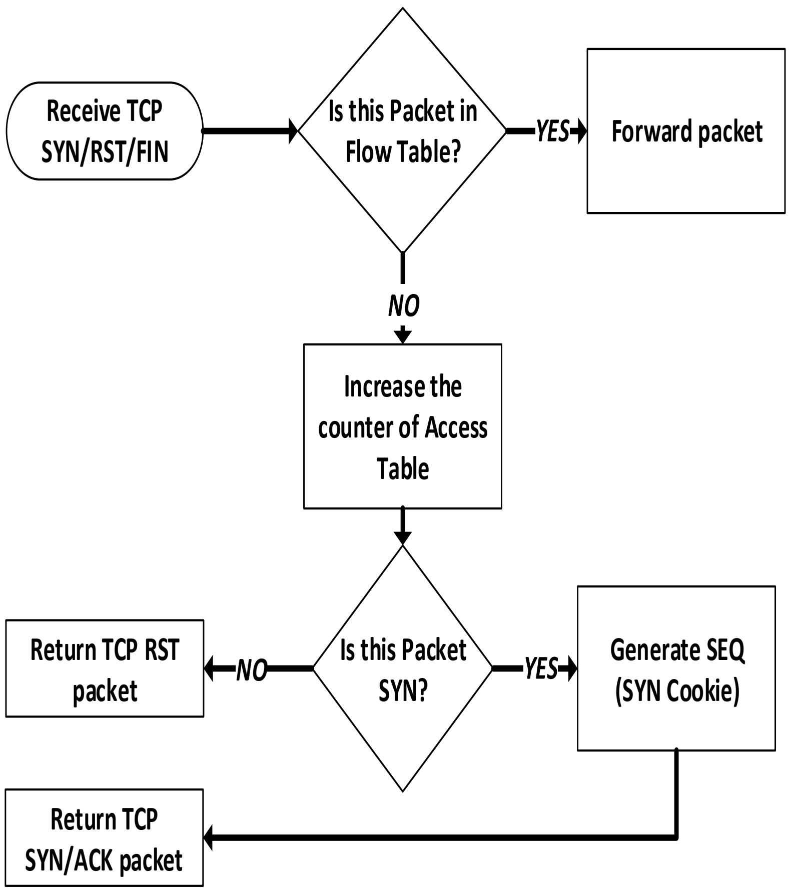
IP Address	Counter
10.0.0.1	15
12.2.0.1	1
40.0.0.4	100

Connection Migration – State Diagram

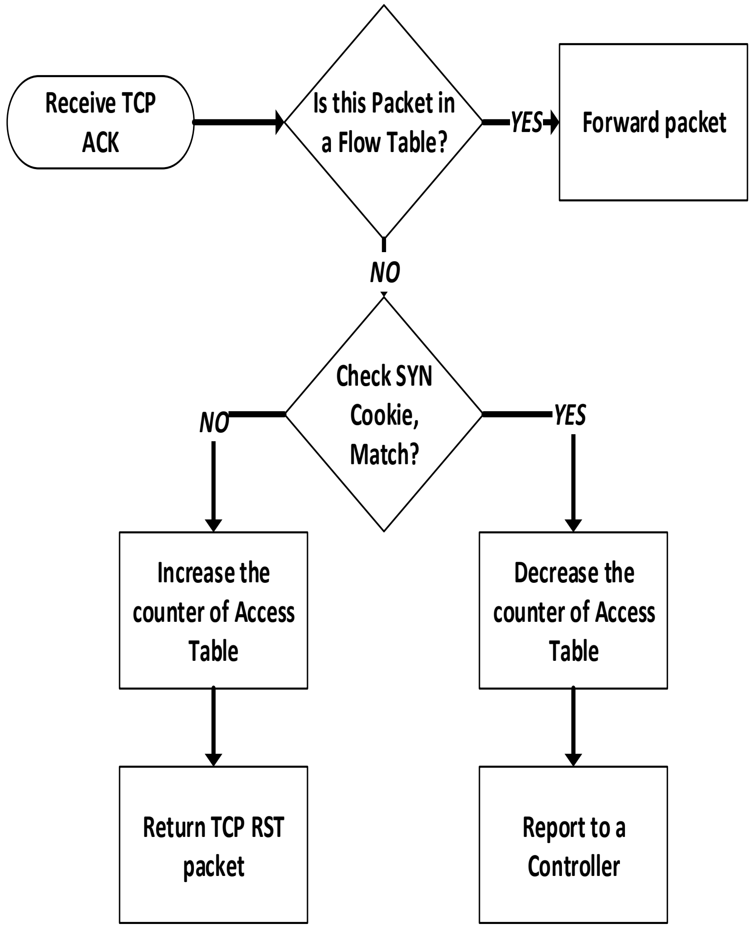
- 4 states
  - Classification
    - Distinguish useful TCP connections via SYN Cookies
  - Report
    - Report to a controller
  - Migration
    - Migrate a TCP connection if it is a useful (or valid) connection
  - Relay
    - Relay all TCP packets between a connection source and a destination



Connection Migration – Flow Chart

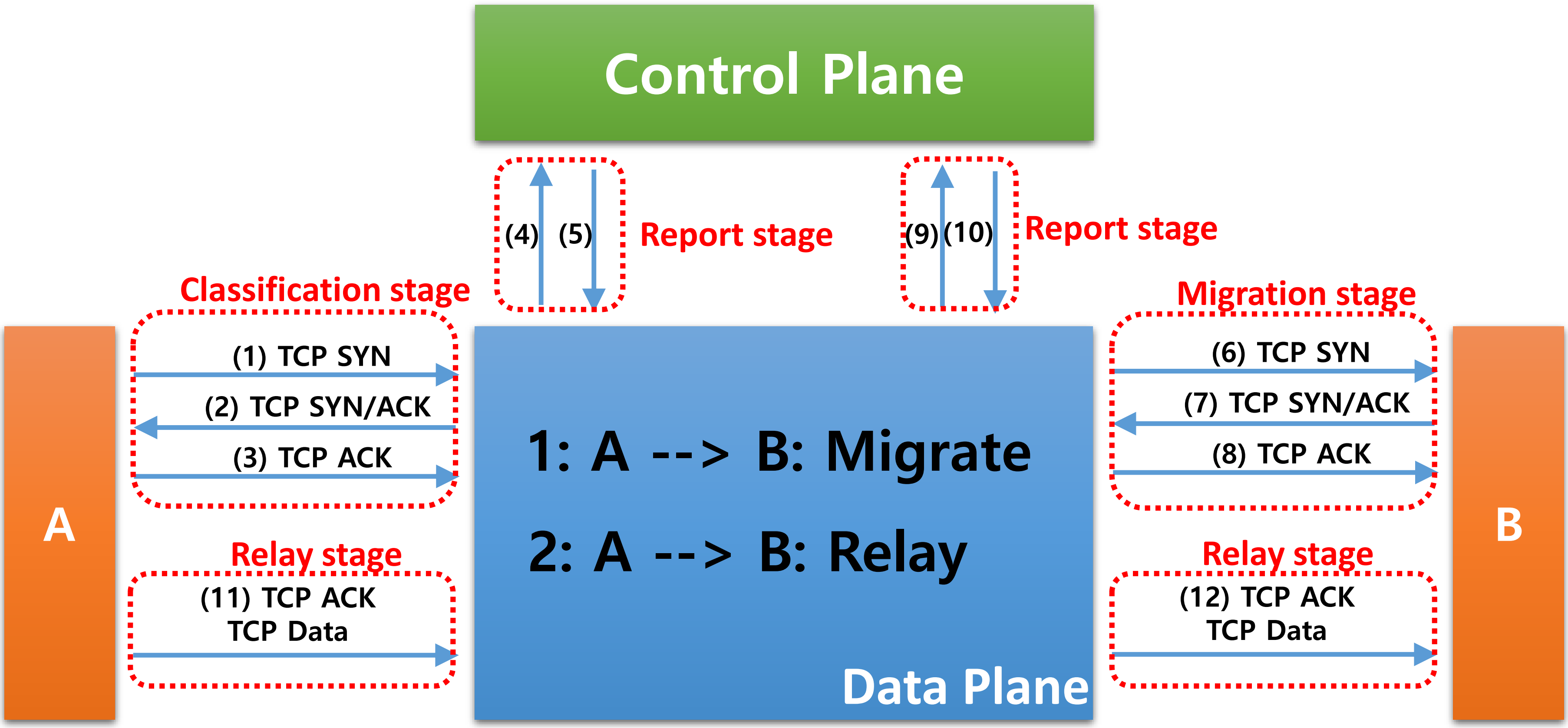


Flow chart
- The case of receiving TCP SYN/RST/FIN packet



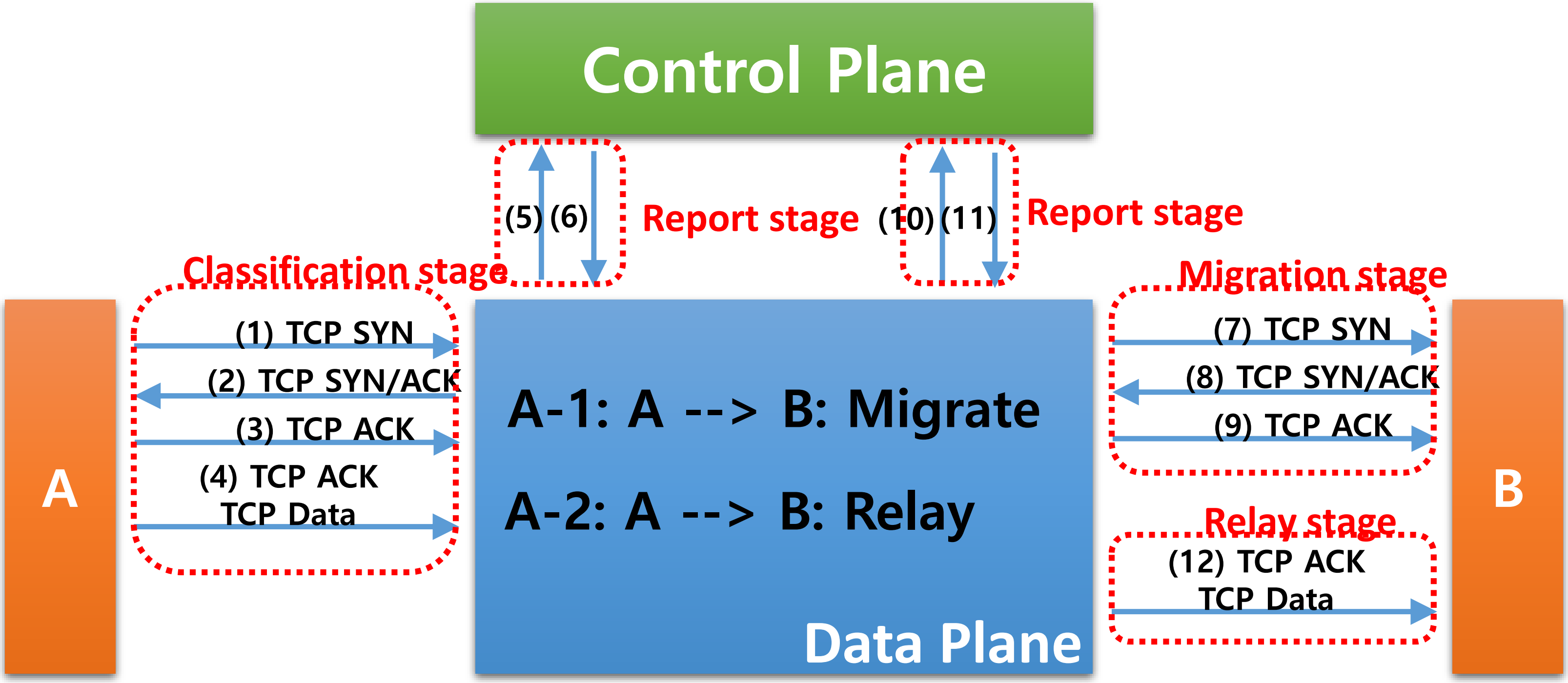
Flow chart
- The case of receiving TCP ACK packet

Connection Migration – Packet Diagram



Delayed Connection Migration

- Concept
  - Delay Connection Migration until the data plane receives (a) data packet(s)
- Why?
  - Good for reducing the effects of some advanced attacks
    - E.g., fake TCP connection setup (e.g., HTTP)

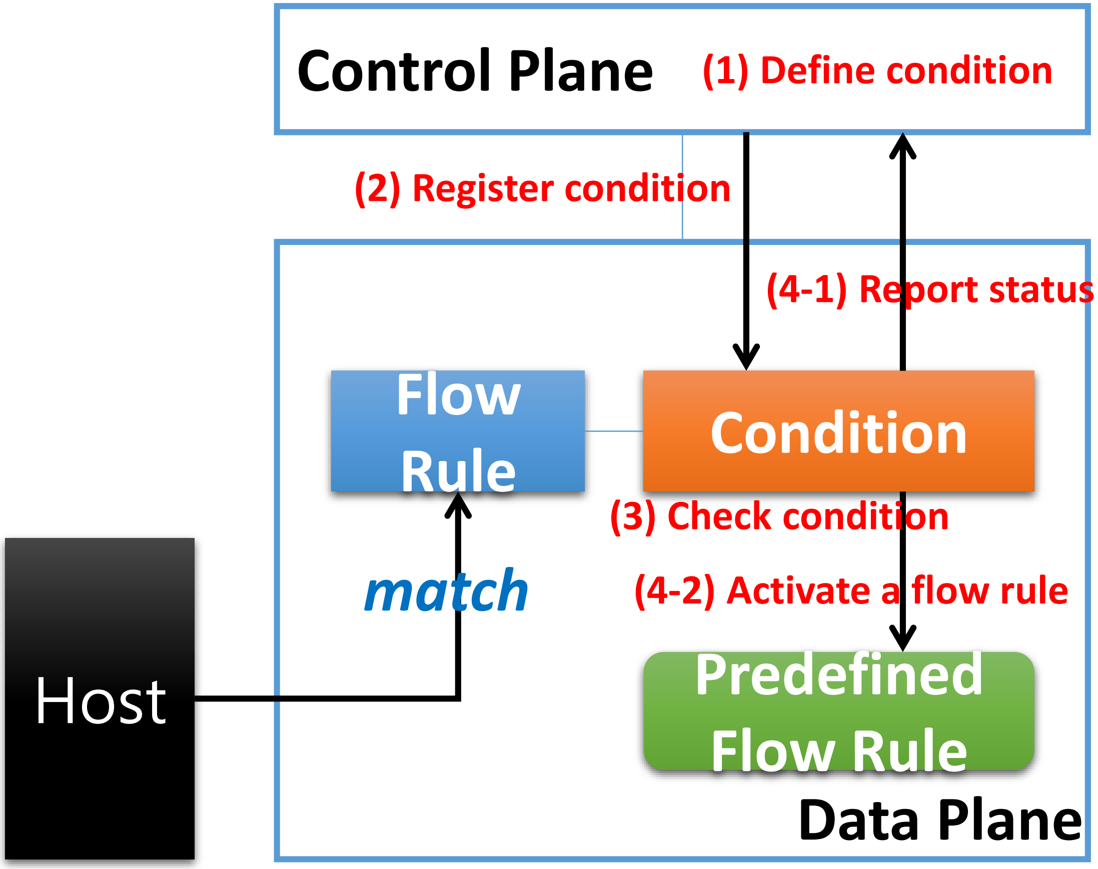


Actuating Trigger - Idea

- Two functions
  - Report the following items to the control plane asynchronously
    - Network status
    - Payload information
  - Activate flow rules based on some predefined conditions
    - Security application can use this feature to turn on security policies without delay

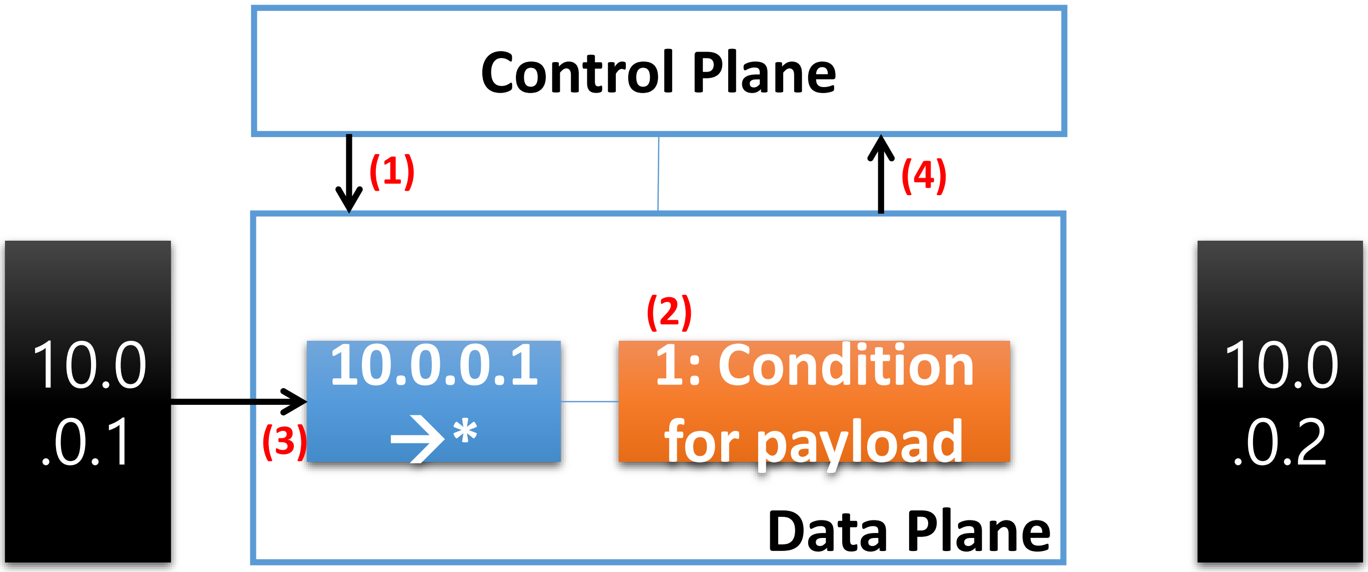
Actuating Trigger – Operations

- 4 main operations
  - In the control plane
    - Define a condition
    - Register the condition
  - In the data plane
    - Check the condition
    - When the condition is satisfied,
      - Report a network status or payload
      - Activate a flow rule



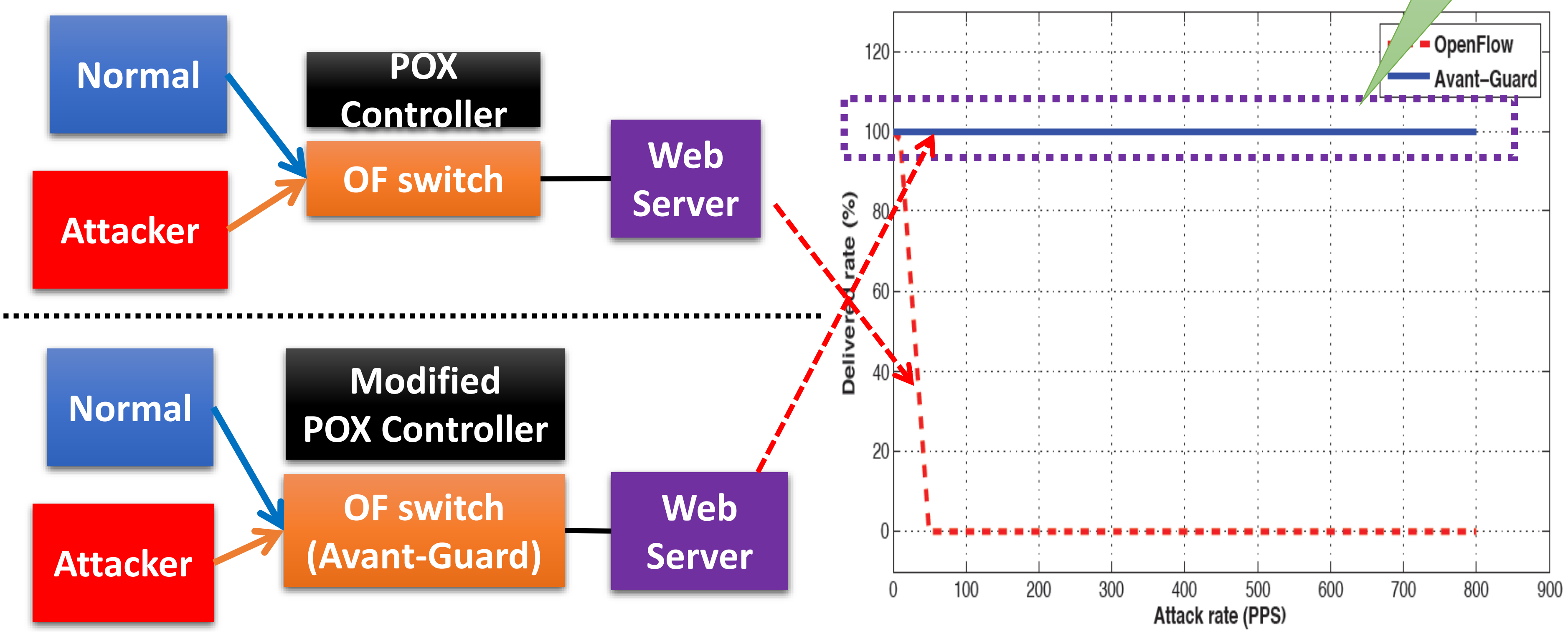
Actuating Trigger - Example

- Example of reporting payload
  - 1) defined a condition : want to see payloads of packet from 10.0.0.1
  - 2) register this condition to the data plane
  - 3) packet is delivered from 10.0.0.1
  - 4) payload is delivered to the control plane



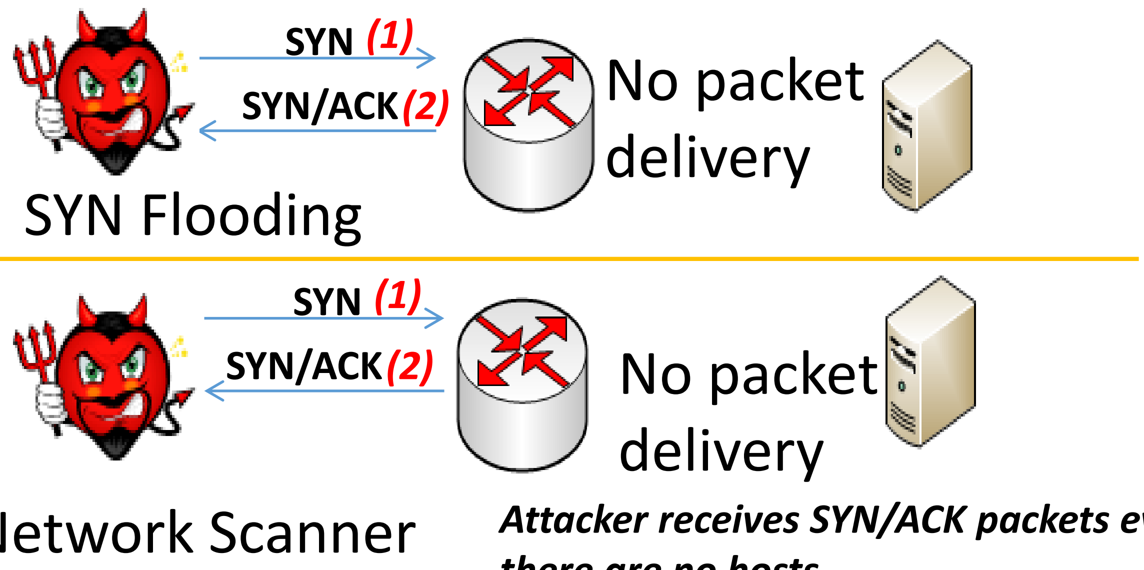
Evaluation – Use Case

- Network saturation attack case
  - A normal client sends HTTP requests to a web server
  - An attacker tries a SYN flooding attack to a web server



Evaluation – Use Case

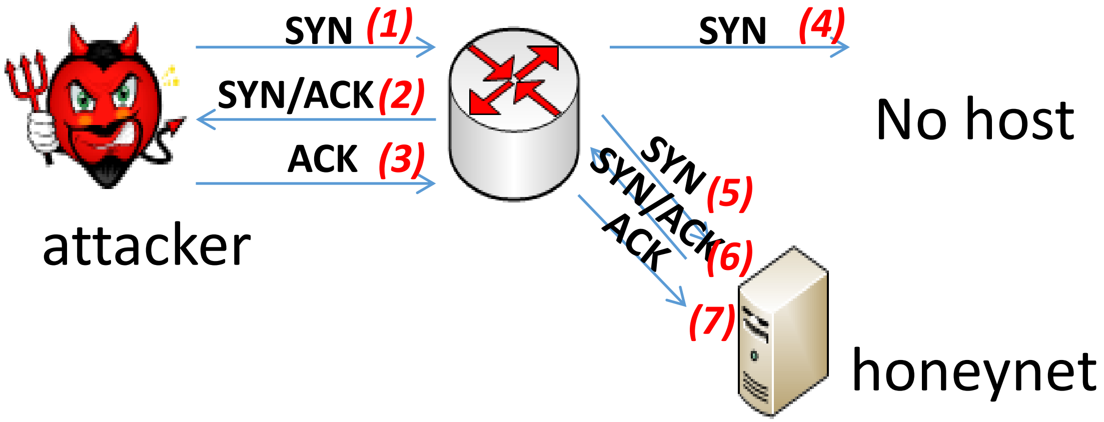
- Detecting SYN flooding/scanning
  - Approach
    - SYN flooding packets are automatically rejected
    - Network scanning attackers will be confused by response packets
      - They may think that all network hosts are alive and all network ports are open (a kind of **White hole**)



Attacker receives SYN/ACK packets even though there are no hosts
→ White hole

Evaluation – Use Case

- Intelligent Honeynet
  - Approach
    - When we try to do connection migration,
      - If we can not find a real target host, we may consider this connection as suspicious
    - Then, a security application can redirect this connection to our honeynet automatically
    - Finally, this attacker will perform malicious operations inside a honenet



Evaluation - Overhead

- Connection migration

normal	connection migration	overhead
1608.6 us	1618.74 us	0,626 %

- Actuating trigger

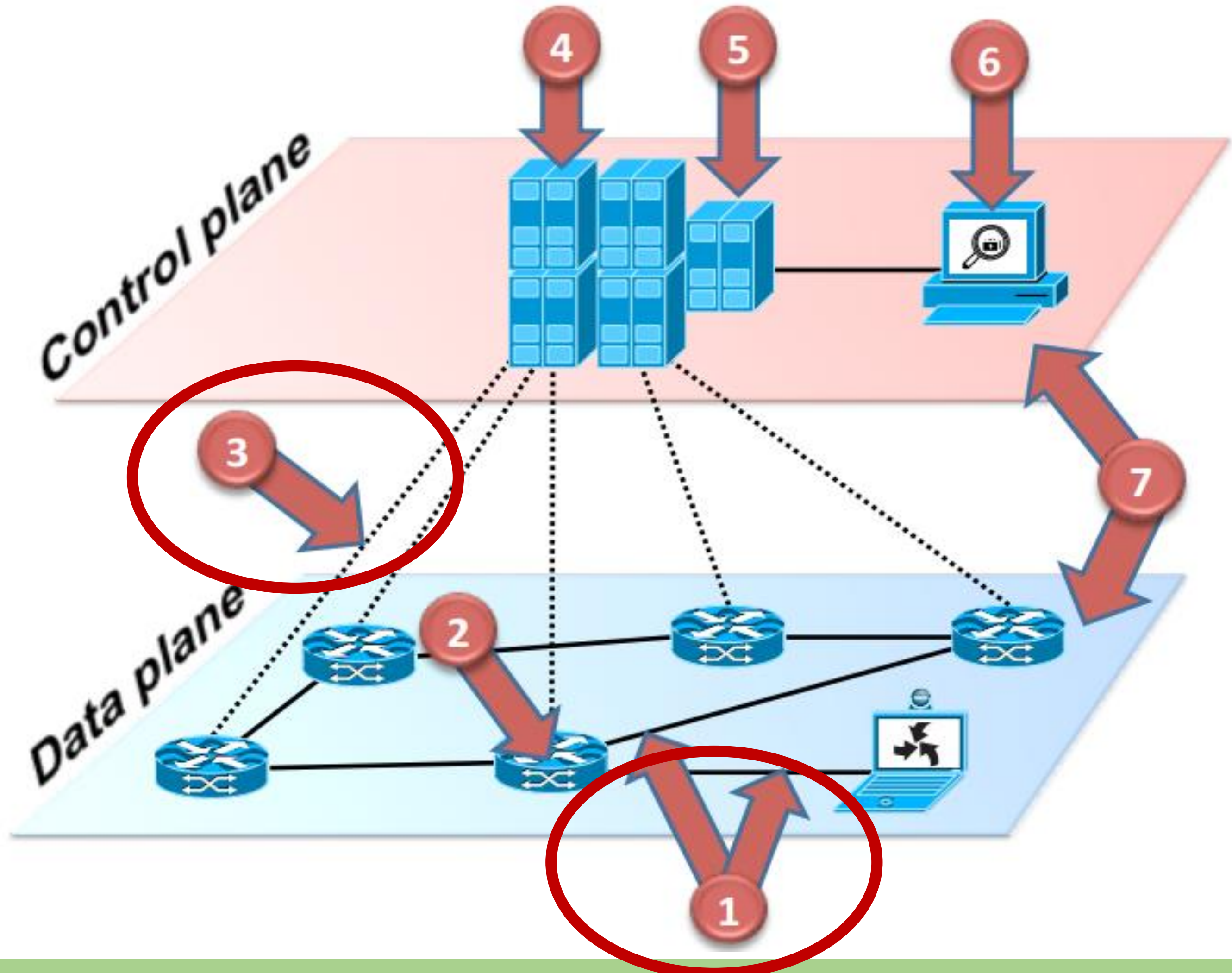
item	time
Traffic-rate based condition check	0.322 us
Payload based condition check	= 0
Rule activation	1.697 us

Critique

- Needs to extend OpenFlow protocol!
  - Connection migration
    - E.g., OFPFC_MIGRATE, …
  - Actuating trigger
    - E.g., OFPFC_REG_PAYLOAD, …

- Also brings intelligence to data plane – may or may not be a good idea

Compromised End Hosts



Another Vulnerability: OF Topology Management

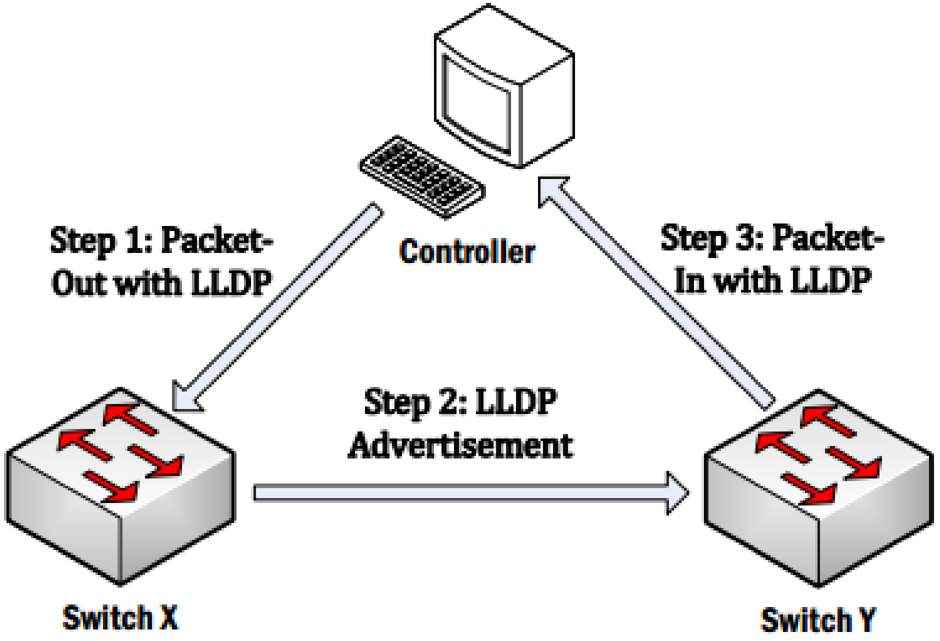
- Topology management includes three parts: switch discovery, host discovery and internal links (switch-to-switch link) discovery.

- Within the OpenFlow controller:
  - Host Tracking Service (HTS) maintains a host profile that includes MAC address, IP address, location information and VLAN ID. Host profile is maintained to track the location of a host and is updated dynamically.

  - Link Discovery Service (LDS) uses Open Flow Discovery Protocol (OFDP) to detect internal links between switches.trolled by Topology Management Services.

Recap - Link Discovery Service

- Open Flow Discovery Protocol (OFDP), which refers to LLDP (Link Layer Discovery Protocol) packets, to detect internal links between switches.

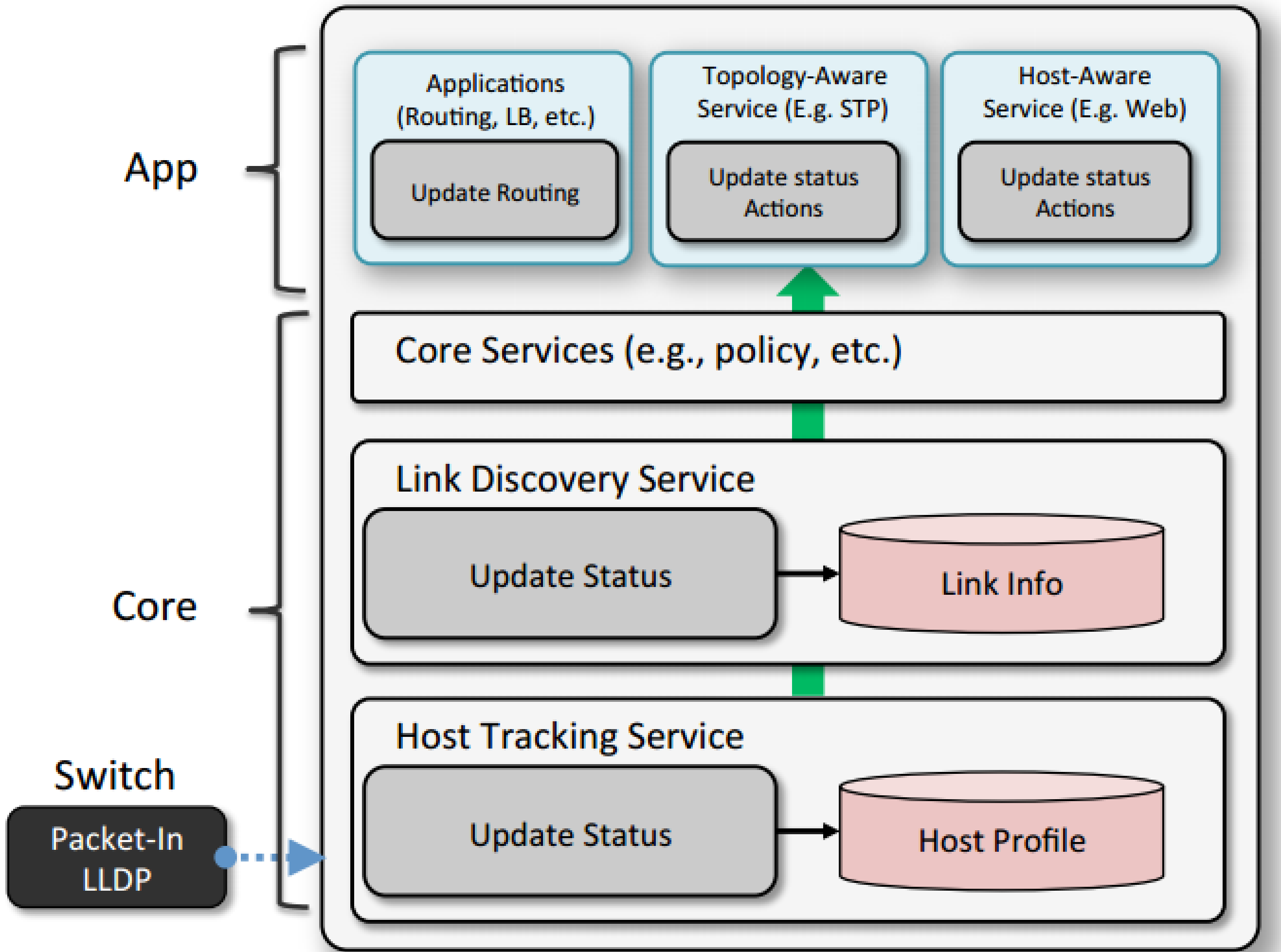


Threat [1]

- If fundamental network topology information is poisoned
  - all the dependent network services are affected
- Host location hijacking attack and link fabrication attacks are possible

[1] Hong, Sungmin, et al. "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures." *NDSS*. 2015.

Controller Host Tracking Systems



Hong, Sungmin, et al. "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Counter-measures." *NDSS* 2015.

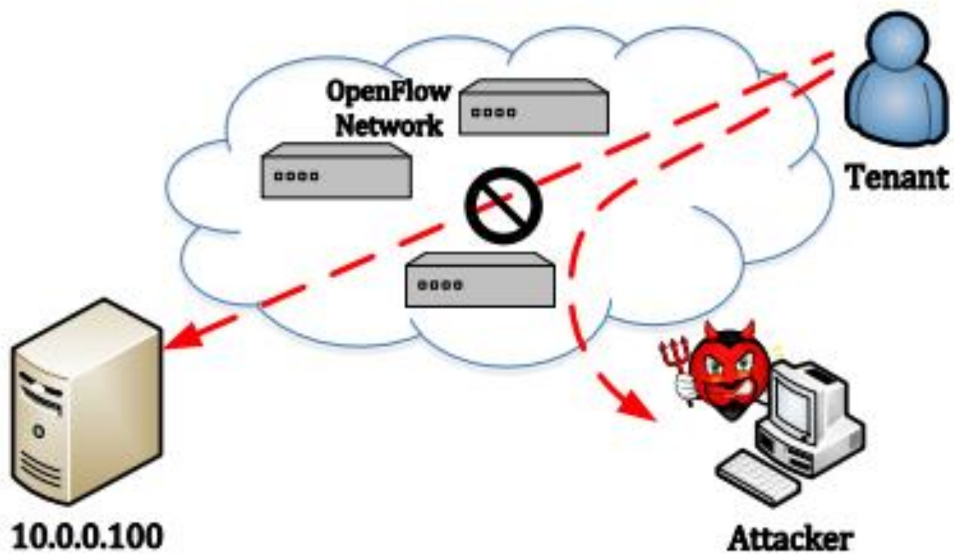
Controller Host Tracking Systems

Controller Platform	Link Discovery Service	TLVs	Host Tracking Service	Host Profile
Ryu	switches.py	DPID, Port ID, TTL	host_tracker.py	MAC, ***IP***, Location
Maestro	DiscoveryApp.java	DPID, Port ID, TTL	LocationManagementApp.java	***MAC***, Location
NOX	discovery.py	DPID, Port ID, TTL	hosttracker.cc	***MAC***, Location
POX	discovery.py	DPID, Port ID, TTL, System Description	host_tracker.py	***MAC***, IP, Location
Floodlight	LinkDiscoveryManager.java	DPID, Port ID, TTL, System Description	DeviceManagerImpl.java	***MAC***, ***VLAN ID***, IP, Location
OpenDayLight	DiscoveryService.java	DPID, Port ID, TTL, System Description	DeviceManagerImpl.java	***MAC***, ***VLAN ID***, IP, Location
OpenIRIS	OFMLinkDiscovery.java	DPID, Port ID, TTL, System Description	OFMDeviceManager.java	***MAC***, ***VLAN ID***, IP, Location
Beacon	TopologyImpl.java	DPID, Port ID, TTL, Full Version of DPID	DeviceManagerImpl.java	***MAC***, VLAN ID, ***IP***, Location

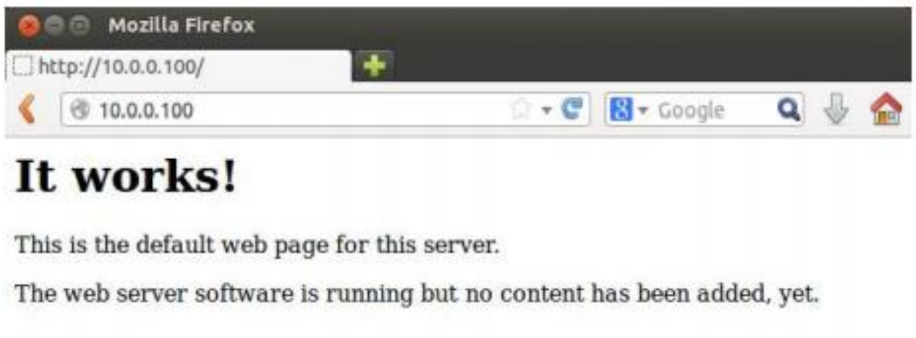
- (1) MAC address
- (2) IP address
- (3) Location information (i.e., the DPID and the port number of the attached switch as well as the last seen timestamp).

Host Location Hijacking Attack

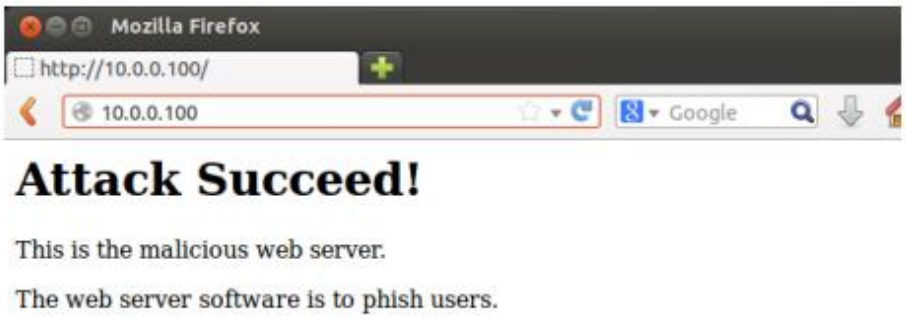
- Host Tracking Service maintains a host profile for each end host to track network mobility.
- Adversary can tamper host location information which in turns affects routing decisions and hijack the traffic towards the host.
- Caused by lack of security considerations in current controllers



Web Impersonation Attack



(a) Connected to Genuine Server



(b) Hijacked by Malicious Server

LLDP – why is it dangerous?

- OpenFlow controllers use a discovery service built on LLDP for topology discovery

- In theory, discovery service should:
  - Ensure integrity and origin of a LLDP packet (*integrity invariant*)
  - Ensure that only switches are on the path of LLDP packets (and no hosts; *path invariant*)

- Unfortunately, current controllers don't care much…
  - E.g., no feature in any controller to check integrity of LLDP packets

- Also: most controllers are open source – danger of circumvention of possible precautions

Attacker Options

- Create falsified LLDP packets (violation of integrity/origin invariant) or relay LLDP packets between switches (violation of path invariant)

- Both are possible since OpenFlow allows LLDP packets to originate from switch ports that are assigned to a host

Link Fabrication Attack

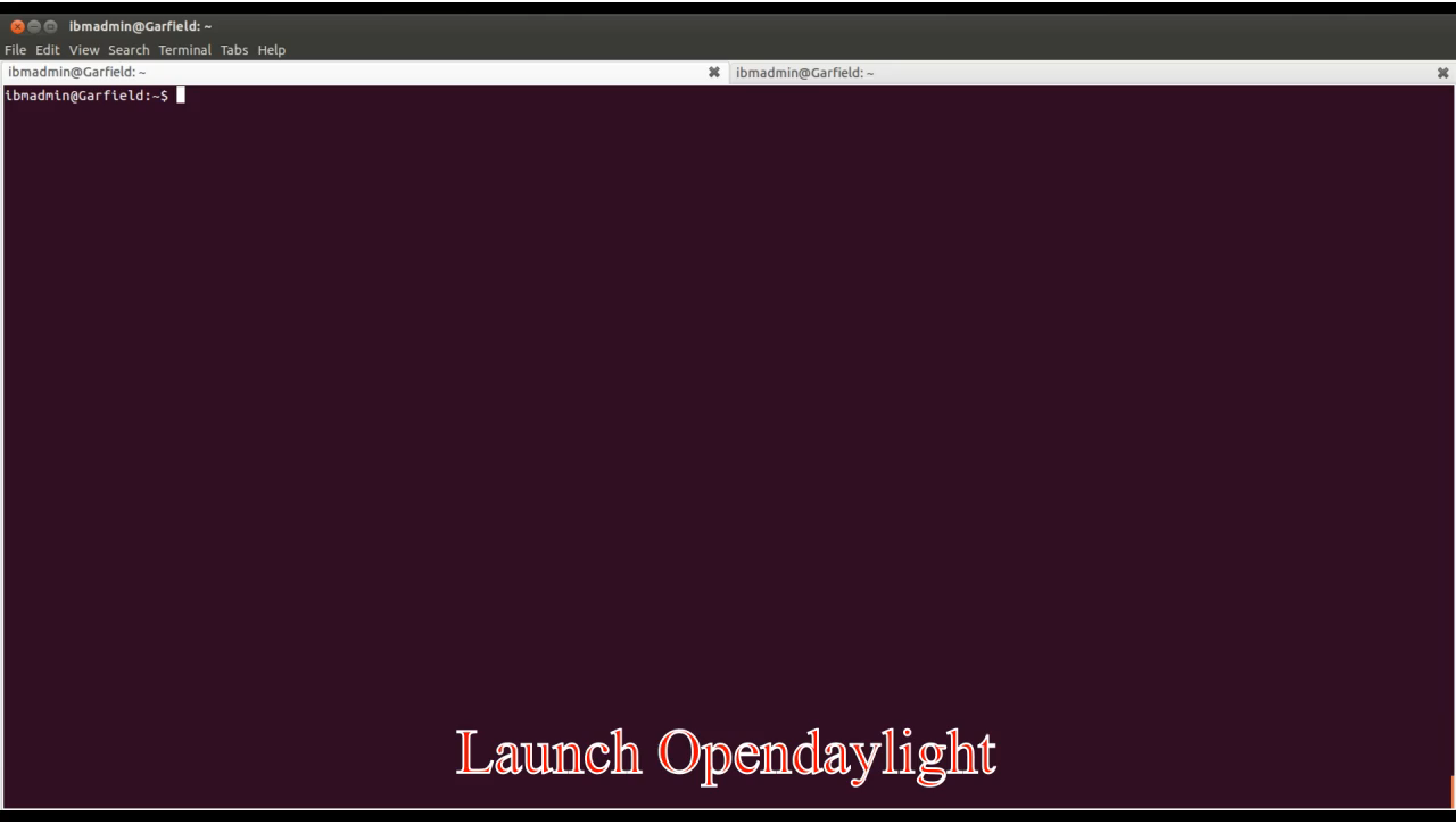
- Fake LLDP Injection plus monitoring the traffic from OpenFlow switches: the attacker can
  - Learn LLDP packet structure by observing benign LLDP packets,
  - modify the specific contents of a captured LLDP packet,
  - generate fake LLDP packets (e.g., with falsified ports or DPIDs) to announce bogus internal links between two switches.

Dl_dst	Dl_src	Eth_type	Chassis ID TLV	Port ID TLV	TTL TLV
01:80:C2:00:00:0E	Outgoing Port MAC	0X88CC	DPID of Switch	Port Number of Switch	Time to Live

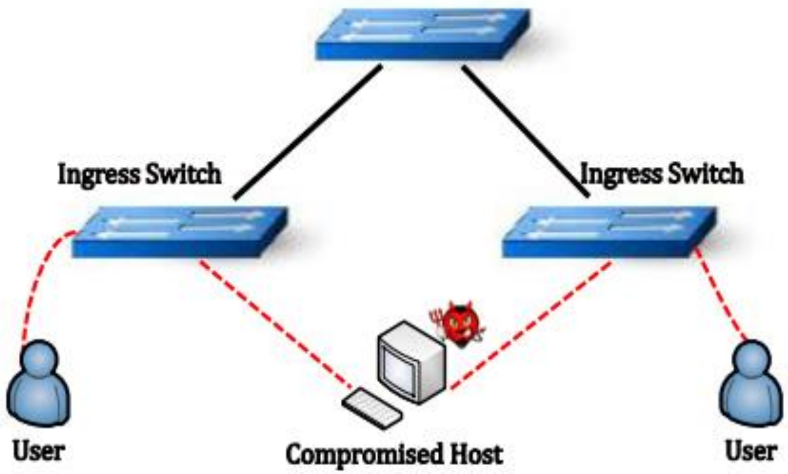
TABLE II: The Format of LLDP Packets

- LLDP relay:
  - when receiving an LLDP packet from one target switch,
  - the attacker repeats it to another target switch without any modification constructing a fake topology view

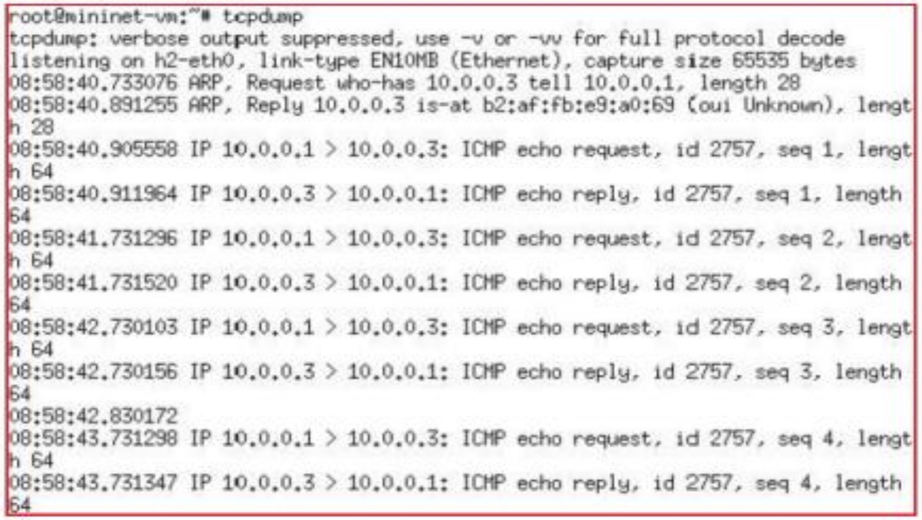
Fake Topology Attack



Man-In-The-Middle Attack



(a) Attack Topology



(b) Attack Result

Dynamic Defense Strategies against Host Location Hijack

- Authenticate Host Entity: public-key infrastructure
  - Overhead for keeping public keys in the OpenFlow controller side and computation overhead for handling each Packet-In message.

- Verify the Legitimacy of Host Migration
  - verify the legitimacy of the host migration by checking the ***precondition (Port-Down) and post condition (Host unreachable in old location)***
  - Performance overhead but lighter and more feasible

Dynamic Defense Strategies against Link Fabrication

- Authentication for LLDP packets
  - Adds extra controller-signed authenticator ((HMAC) code) TLVs in the LLDP packet and check the signature when receiving the LLDP packets.
  - Fails to defend against the relay/tunneling link fabrication attack

- Verification for Switch Port Property
  - Check if any host resides inside the LLDP propagation
  - If OpenFlow controllers detect host-generated traffic (e.g., DNS) from a specific switch port, Device Type of that port is set as HOST, otherwise switch ports are set as SWITCH.

TopoGuard - Automatic and real-time detection

- Port Manager tracks dynamics of switch ports (ANY, SWITCH and HOST)
- Port Property maintains host list to verify the trustworthiness of a host migration.
- The Host Prober tests the liveness of the host in a specific location by issuing a host probing packet.
- Topology Update Checker verifies the legitimacy of a host migration, the integrity/origin of an LLDP packet and switch port property

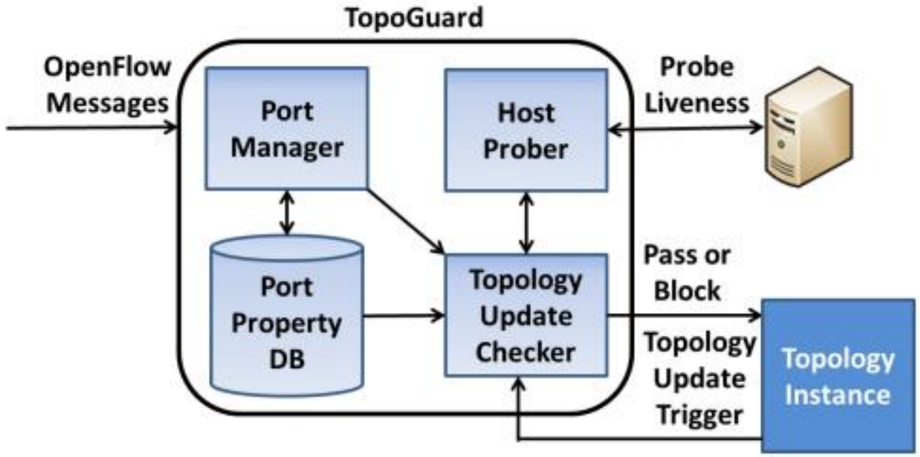


Fig. 8: The Architecture of *TopoGuard*

Port Property Management

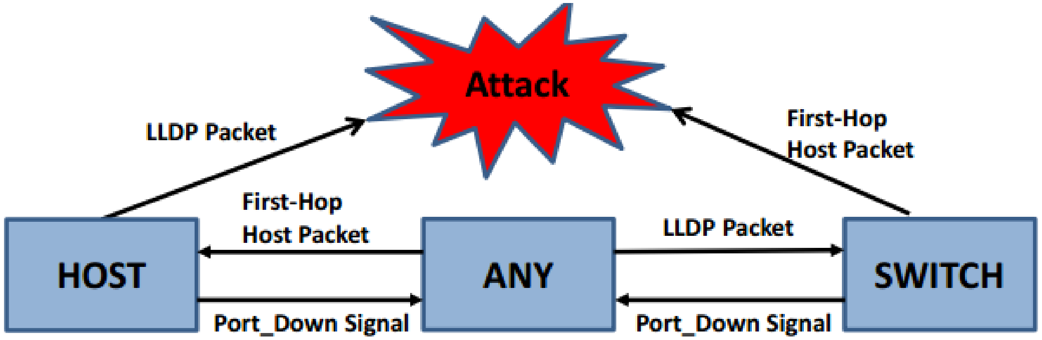


Fig. 9: The Transition Graph of *Device Type*

- Properties for each switch port in an OpenFlow controller.
  - Shows to what kind of device (host, switch, any) a switch port connects
- Upon receiving a mismatching packet (e.g., LLDP from host), raise alert
- Host movement only autorised if matching Port_Down signals are received

TopoGuard Implementation - Effectiveness

- TopoGuard with Floodlight implementation

```
lew I/O server worker #2-1] Link added: Link [src=00:00:00:00:00:00:00:01 outPort=3, dst=00:00:00:00:00:00:
:rver-main] Starting DebugServer on :6655
`$PortManager:New I/O server worker #2-1] Device:7a:f1:0d:d6:31:fd is added on:
`$PortManager:New I/O server worker #2-1] sw:1,port:1
`$PortManager:New I/O server worker #2-1] Device:96:2a:7e:28:2f:54 is added on:
`$PortManager:New I/O server worker #2-1] sw:1,port:2
`$PortManager:New I/O server worker #2-2] Device:ca:81:f9:df:0f:b1 is added on:
`$PortManager:New I/O server worker #2-2] sw:2,port:2
`$PortManager:New I/O server worker #2-1] Device:2a:45:f6:50:b9:cf is added on:
`$PortManager:New I/O server worker #2-1] sw:3,port:3
`$PortManager:New I/O server worker #2-2] Violation: Host Move from switch 1 port 1 without Port ShutDown
`$PortManager:New I/O server worker #2-1] Violation: Host Move from switch 1 port 1 is still reachable
```

Fig. 10: The Detection of Host Location Hijacking Attack

TopoGuard Implementation - Effectiveness

```
lew I/O server worker #2-2] Link added: Link [src=00:00:00:00:00:00:00:03 outPort=1, dst=00:00:00:00:00:00:
ler:New I/O server worker #2-1] Inter-switch link detected: Link [src=00:00:00:00:00:00:00:02 outPort=3, ds
lew I/O server worker #2-1] Link added: Link [src=00:00:00:00:00:00:00:02 outPort=3, dst=00:00:00:00:00:00:
ler:New I/O server worker #2-1] Inter-switch link detected: Link [src=00:00:00:00:00:00:00:01 outPort=3, ds
lew I/O server worker #2-1] Link added: Link [src=00:00:00:00:00:00:00:01 outPort=3, dst=00:00:00:00:00:00:
lew I/O server worker #2-1] Link added: Link [src=00:00:00:00:00:00:00:03 outPort=2, dst=00:00:00:00:00:00:
ler:New I/O server worker #2-1] Inter-switch link updated: Link [src=00:00:00:00:00:00:00:03 outPort=2, dst
lew I/O server worker #2-1] Link updated: Link [src=00:00:00:00:00:00:00:03 outPort=2, dst=00:00:00:00:00:0
rver-main] Starting DebugServer on :6655
$PortManager:New I/O server worker #2-2] Violation: Receive LLDP packets from HOST port: SW 1 port 2
$PortManager:New I/O server worker #2-2] Violation: Receive LLDP packets from HOST port: SW 1 port 2
$PortManager:New I/O server worker #2-2] Violation: Receive LLDP packets from HOST port: SW 1 port 2
```

Fig. 11: The Detection of Link Fabrication Attack

When the compromised hosts start relaying LLDP packets, TopoGuard detects the violation of Device Type of particular ports

TopoGuard Implementation - Performance

Link Discovery Snippet	Impact of *TopoGuard* (Percentage)	Controller Overall Cost
LLDP Construction(First time with computing HMAC)	0.431ms(80.4%)	0.536ms
LLDP Construction	0.005ms(2.92%)	0.171ms
LLDP Verification	0.005ms(1.64%)	0.304ms

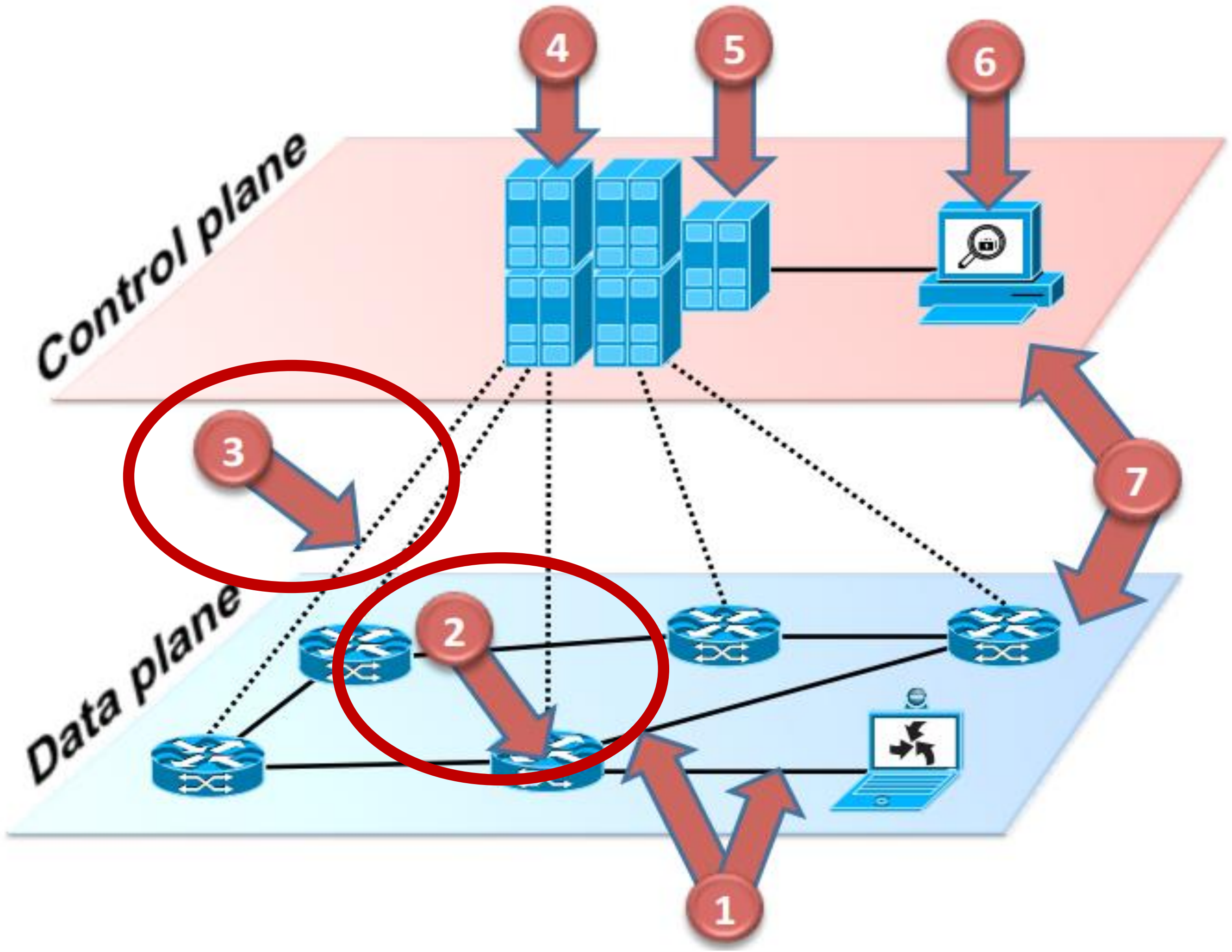
TABLE V: HMAC Overhead on the Floodlight controller

- The performance penalty imposed by TopoGuard mainly comes from the Link Discovery Module and the Packet-In message processing.
- Port Manager incurs a slight delay over the normal LLDP and host-generated packets processing.

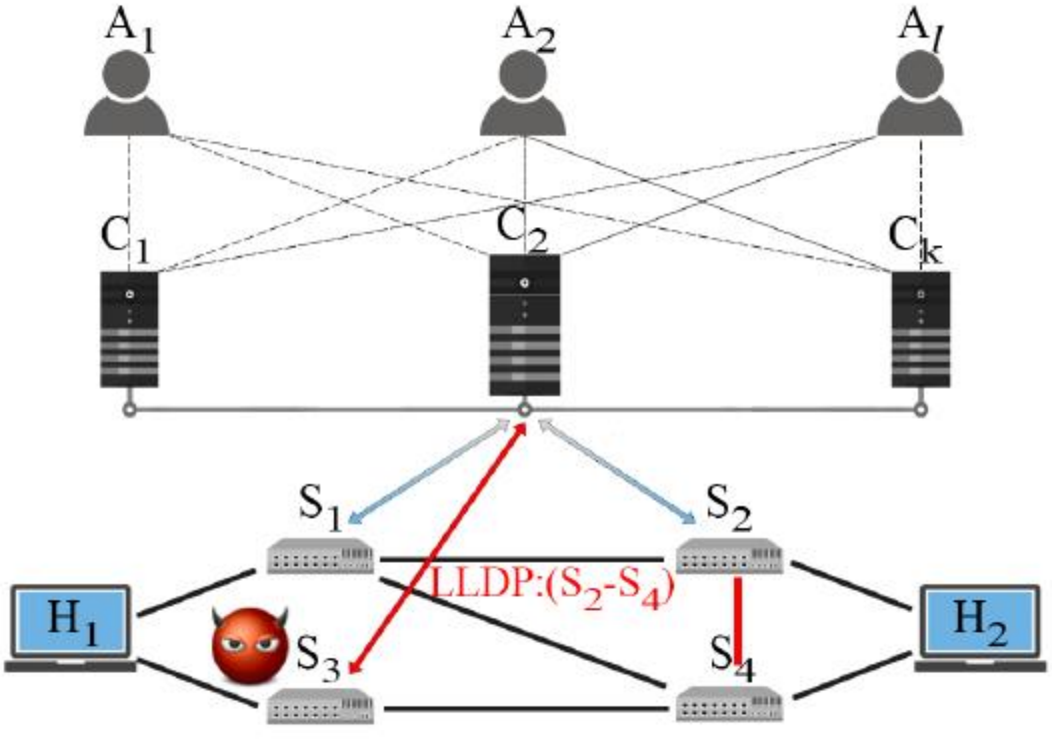
Limitations of TopoGuard

- Security of the network and underlying networking components are essential.
  - What if these are compromised?

Compromised Switches

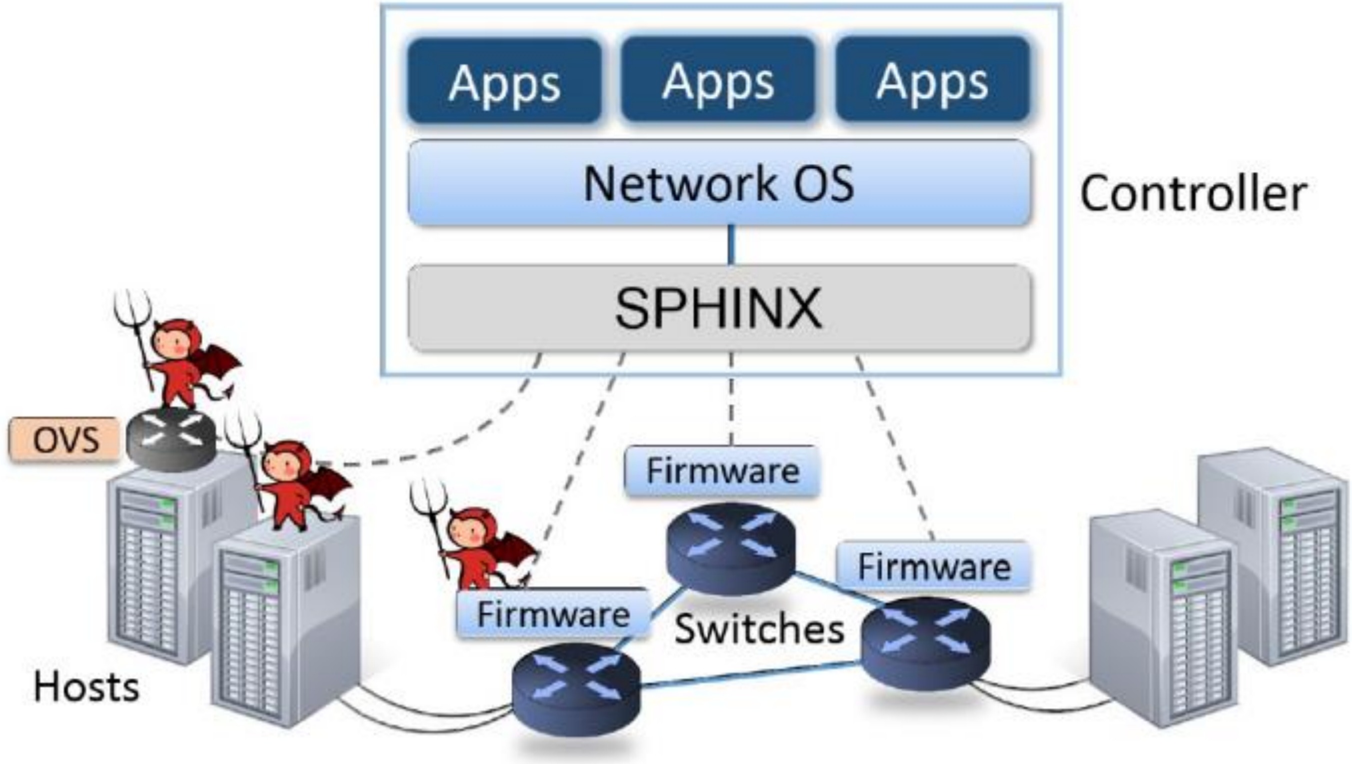


Compromised Switches



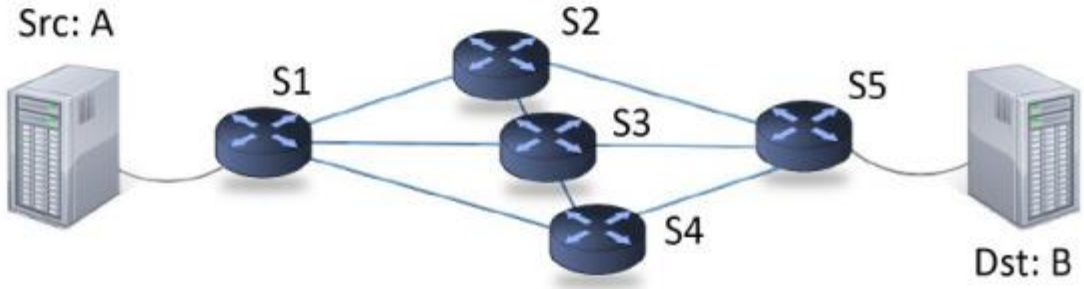
(b) Compromised switch

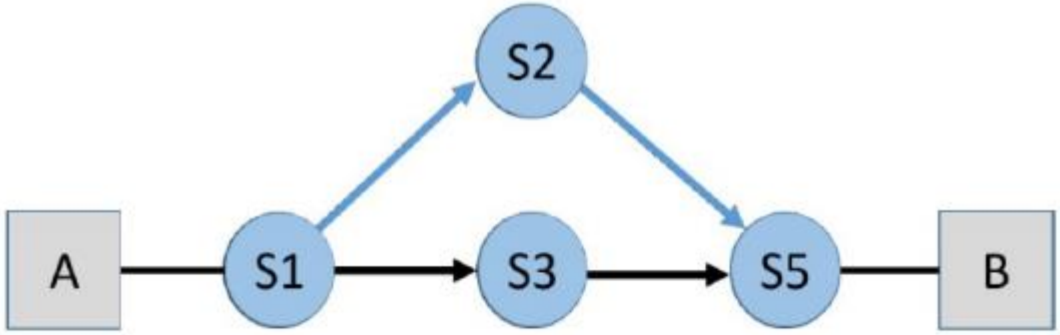
SPHINX [1]



[1] Dhawan, Mohan, et al. "SPHINX: Detecting Security Attacks in Software-Defined Networks." *NDSS* 2015.

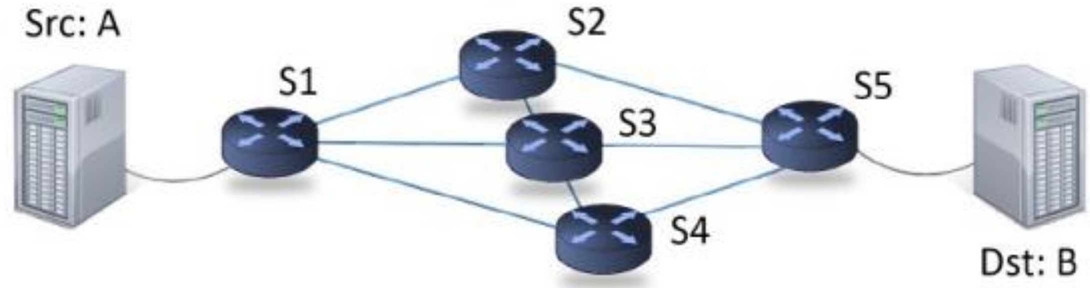
SPHINX

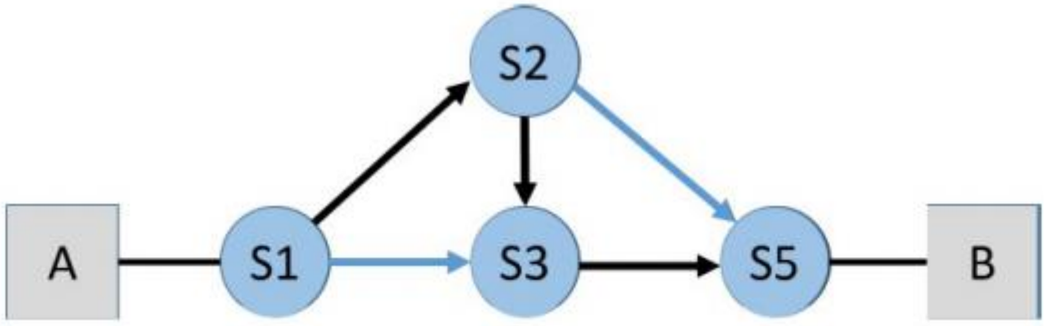




Time T1

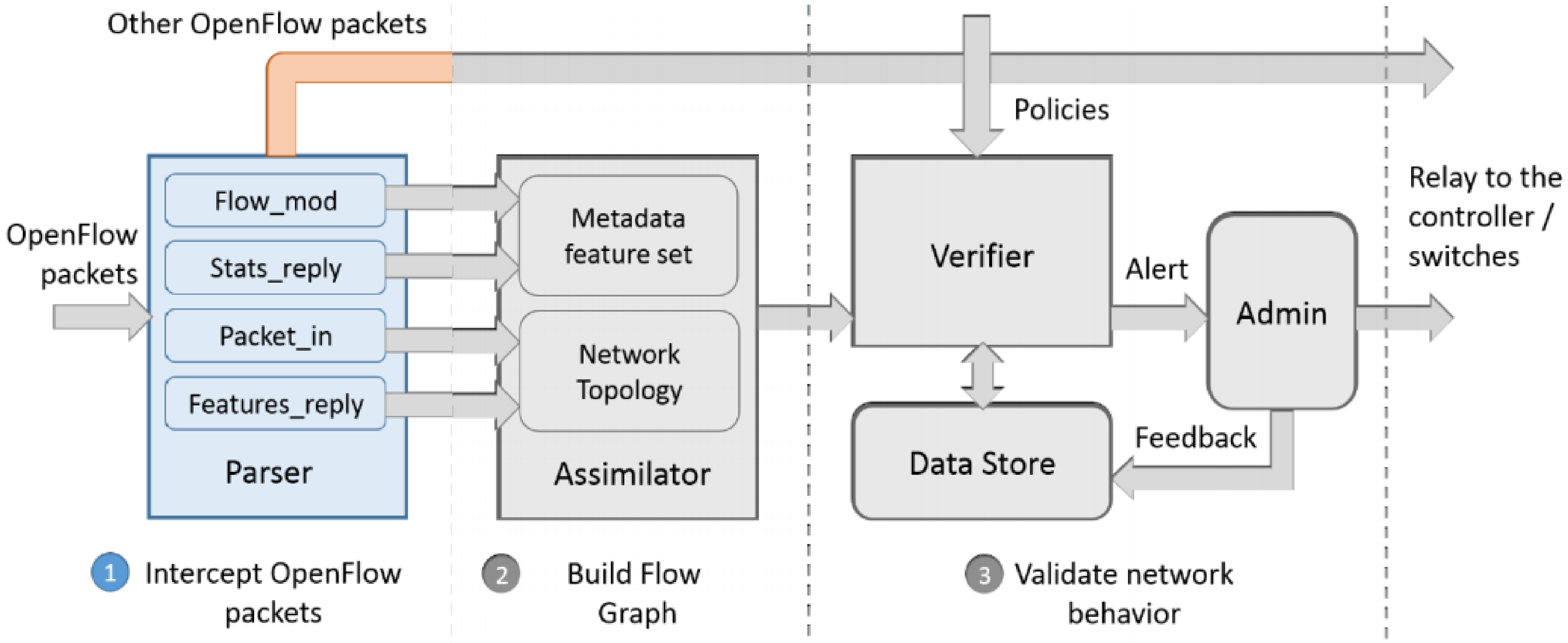
SPHINX





Time T2

SPHINX



Other OpenFlow packets
Policies
OpenFlow packets
Flow_mod
Stats_reply
Packet_in
Features_reply
Parser
Metadata feature set
Network Topology
Assimilator
Verifier
Alert
Admin
Relay to the controller / switches
Data Store
Feedback
1 Intercept OpenFlow packets
2 Build Flow Graph
3 Validate network behavior

SPHINX

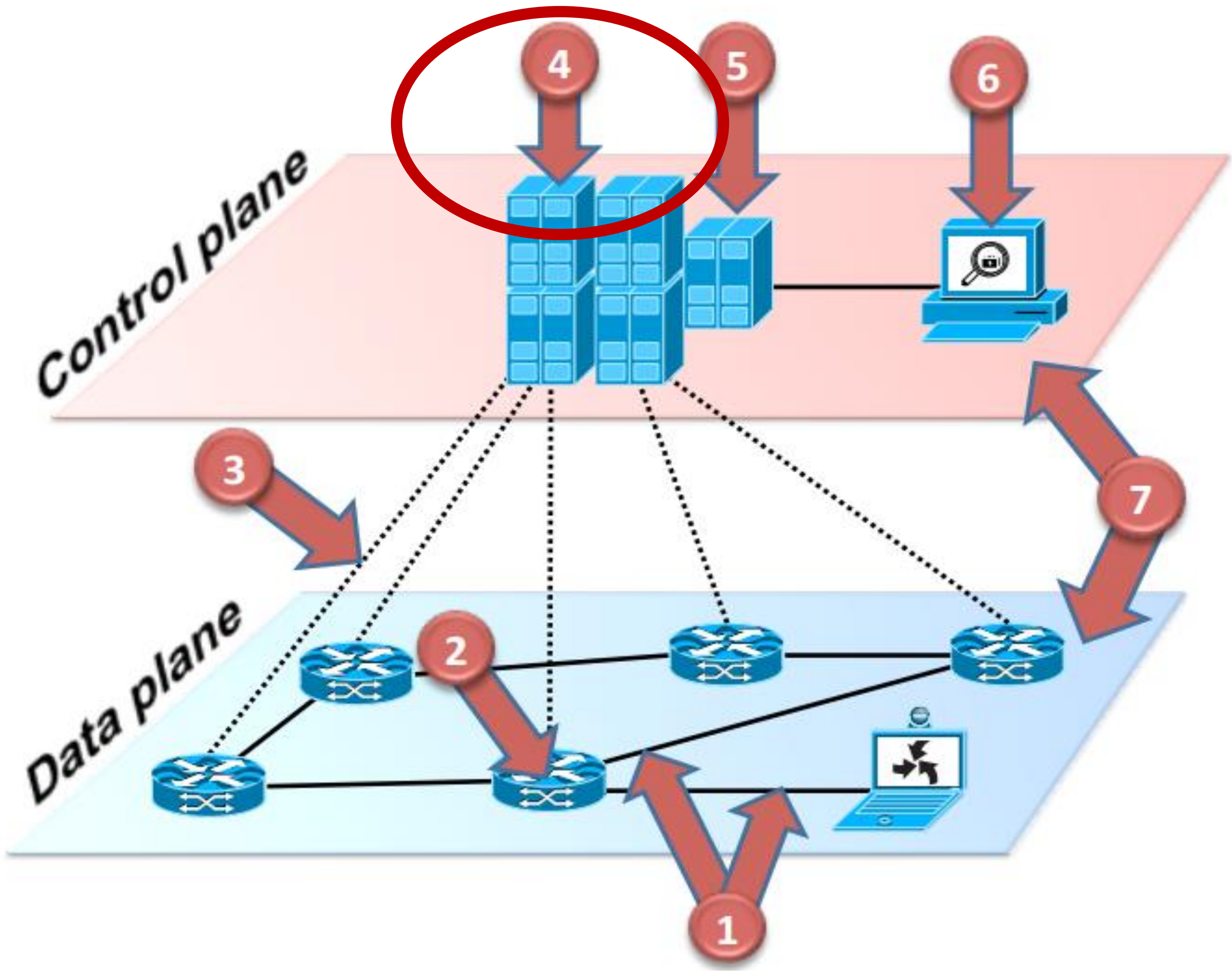
- Specified in constraint language

Feature	Description
Subject	(SRCID, DSTID), where ∀ SRCID and DSTID ∈ {CONTROLLER | WAYPOINTID | HOSTID | ∗}
Object	{COUNTERS | THROUGHPUT | OUT-PORTS | PACKETS | BYTES | RATE | MATCH | WAYPOINT(S) | HOST(S) | LINK(S) | PORT(S) | etc.}
Operation	IN | UNIQUE | BOOL (TRUE, FALSE) | COMPARE ($\leq, \geq, =, \neq$) | etc.
Trigger	PACKET_IN | FLOW_MOD | PERIODIC

- Example policy to check if all flows from host H3 pass through specified waypoints S2 and S3

```
<Policy PolicyId="Waypoints">
    <Subjects><Subject value="H3, *" /></Subjects>
    <Objects>
        <Object><Waypoint value="S2" /></Object>
        <Object><Waypoint value="S3" /></Object>
    </Objects>
    <Operation value="IN" />
    <Trigger value="Periodic" />
</Policy>
```

Compromised Controllers



Compromised Controllers?

Controllers are pieces of software!

"By compromising an SDN controller—a critical component that tells switches how data packets should be forwarded—an attacker would have control over the entire network"

- David Jorm, OpenDayLight Security Team Lead

Compromised Controllers?

- Route flows around security devices
- Controller subverts new flows
- Send traffic to compromised nodes
- “Man in the Middle” attacks
- Modify content
- Insert malware
- Monitor traffic
- Subvert DNS responses
- …

Compromised Controllers?

“In late 2014 an XXE flaw was found in OpenDaylight’s netconf interface. […] OpenDaylight’s netconf implementation did not disable external entities when processing user-supplied XML documents, thereby exposing an XXE flaw. […] A remote attacker, if able to interact with one of OpenDaylight’s netconf interfaces, could use this flaw to exfiltrate files on the OpenDaylight controller. This could include configuration details and plaintext credentials.”

- http://onosproject.org/2015/04/03/sdn-and-security-david-jorm/

Steps towards more secure controllers

- Security-mode ONOS, S(ecurity)E(nhanced)-Floodlight
  - both more or less deal with northbound interface security (app policies)

- Current best practice: make controller machine secure, monitor it, etc.