EL6363 Lab 2

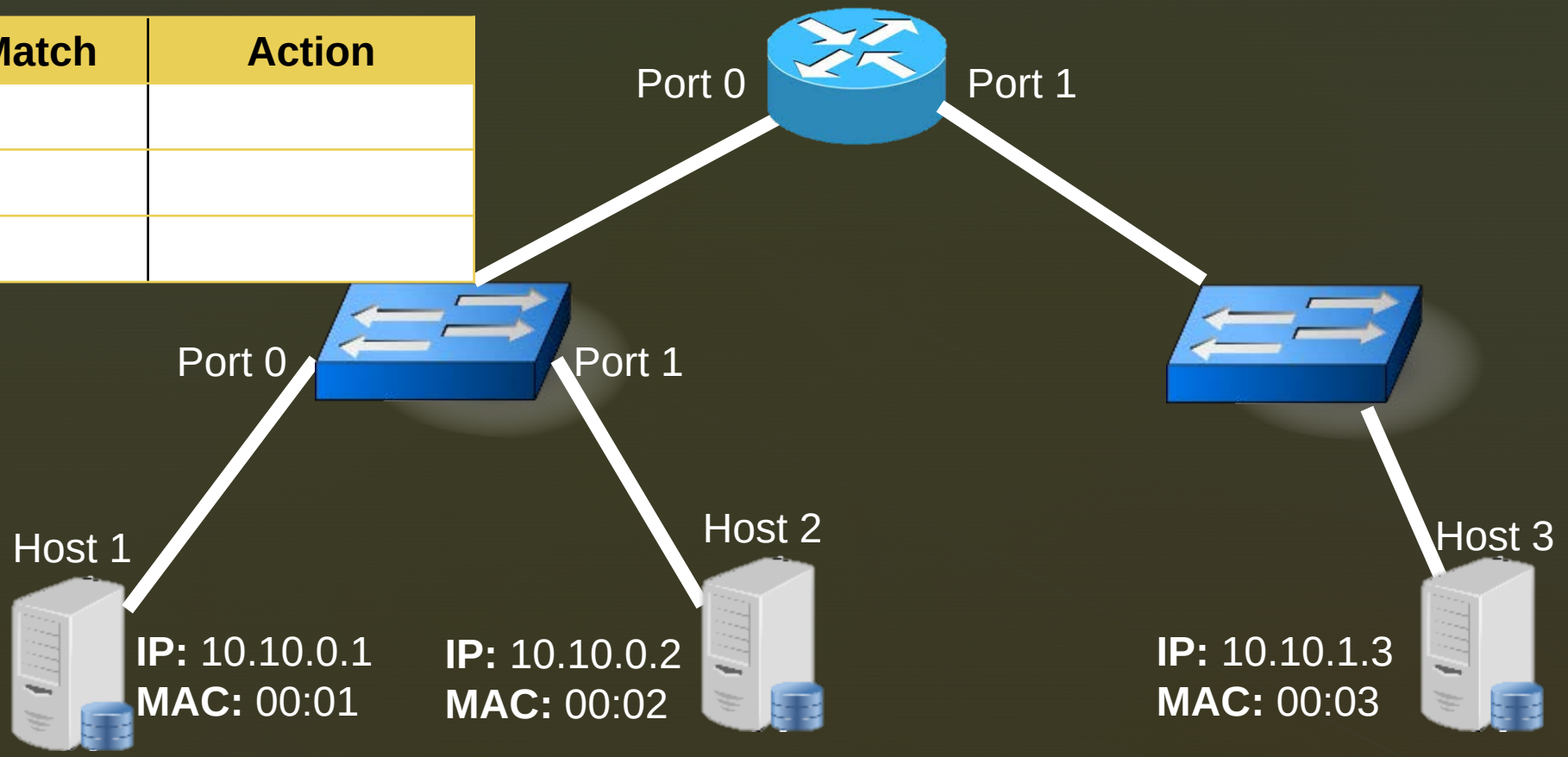
Mininet Tutorial

Lab2 Objectives

- Introduction to **Software Defined Network**
- Master the simulation tool: **Mininet**
- Understand the operation of **Openflow** and observe its messages
- Interfacing with **Open Virtual Switch (OVS)**
- Implement a simple flow-based routing algorithm

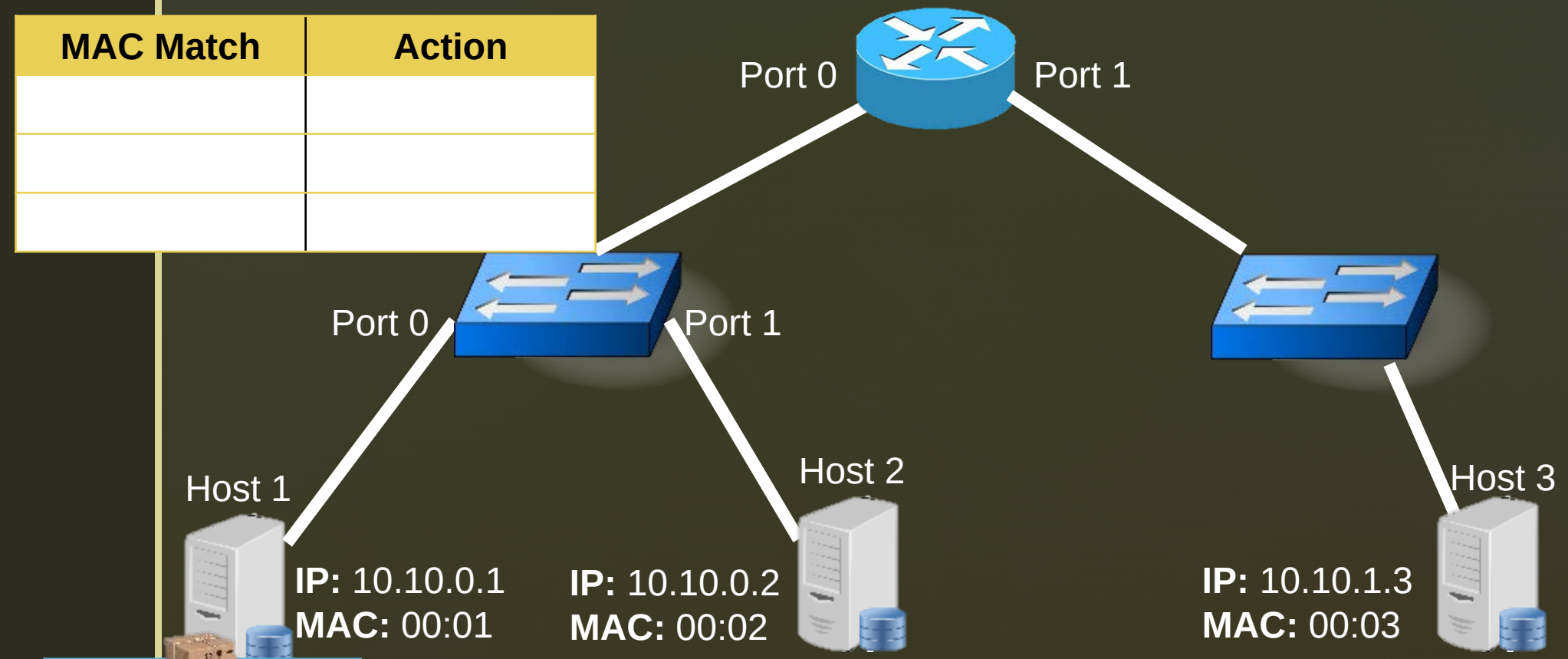
Switch & Networking

MAC Match	Action



Switch & Networking

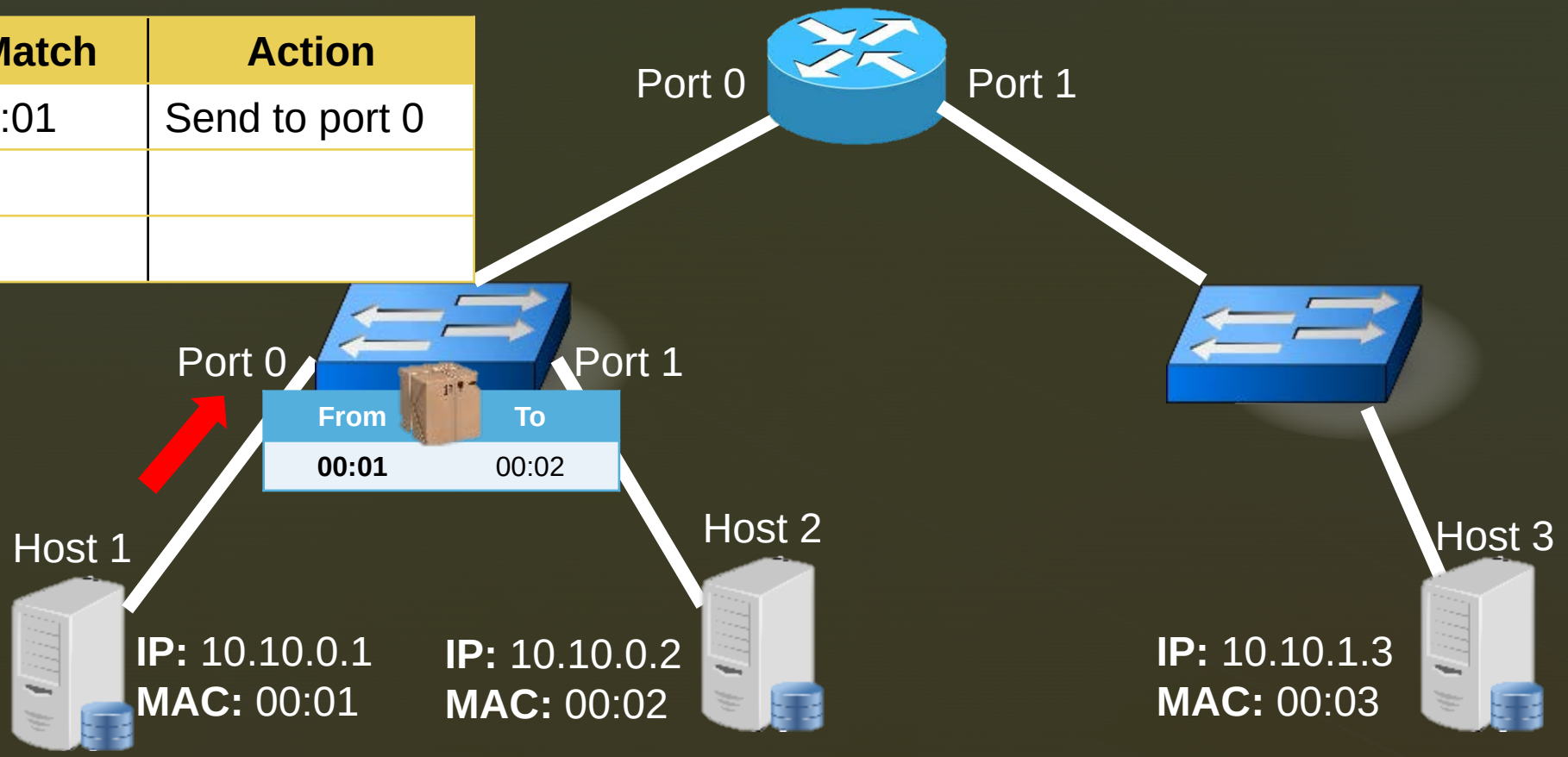
MAC Match	Action



From	To
00:01	00:02

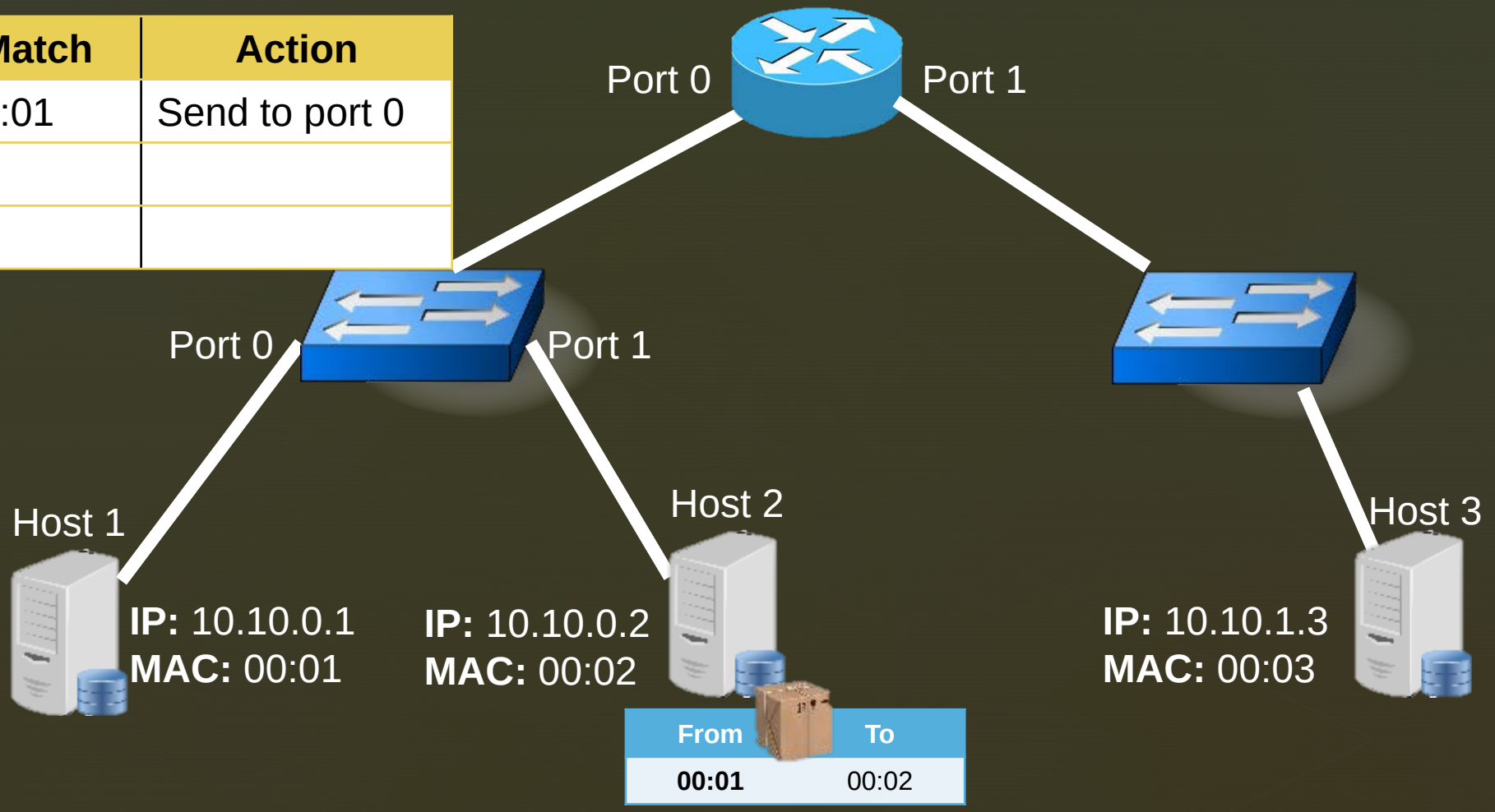
Switch & Networking

MAC Match	Action
MAC: 00:01	Send to port 0



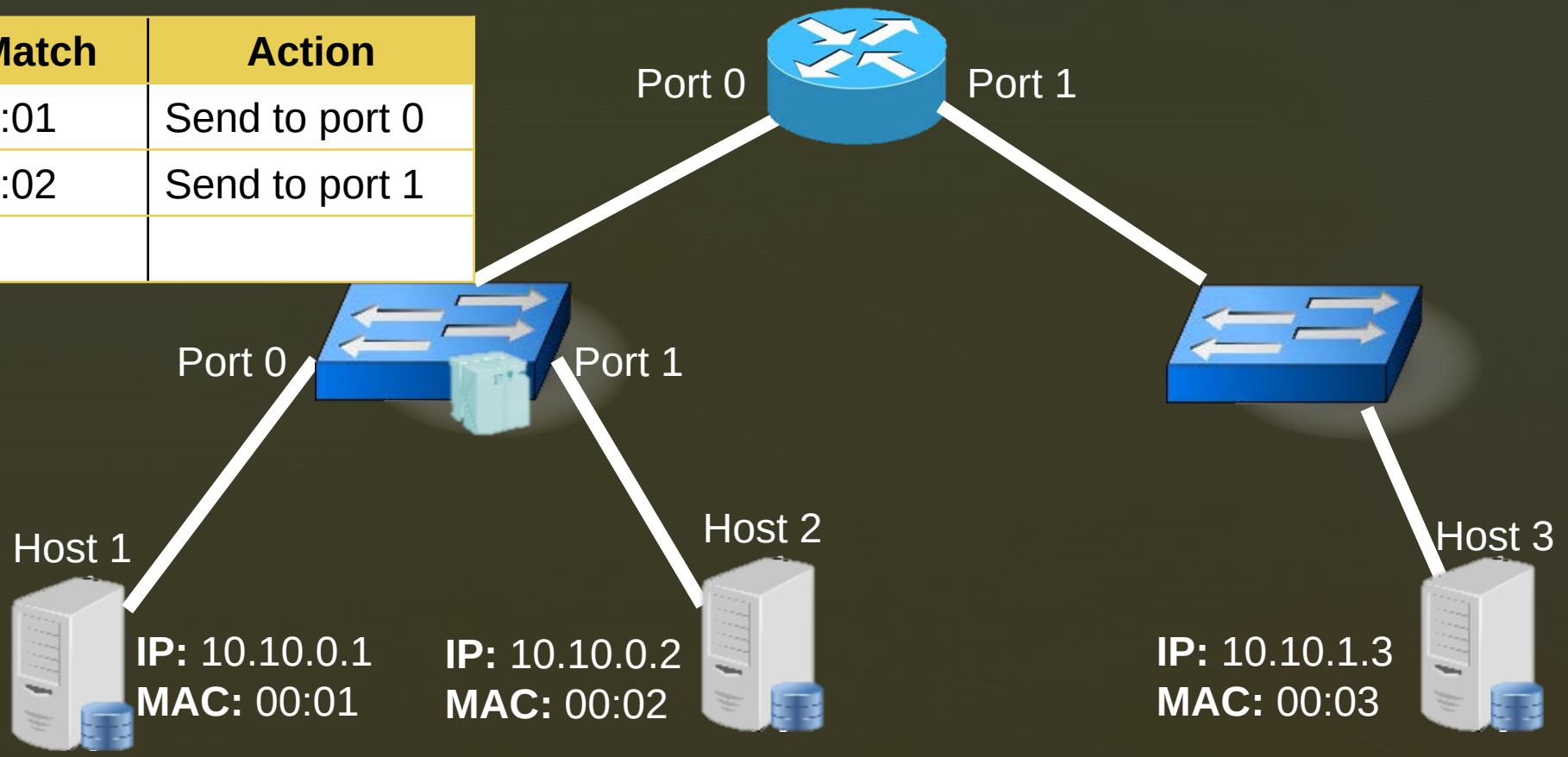
Switch & Networking

MAC Match	Action
MAC: 00:01	Send to port 0



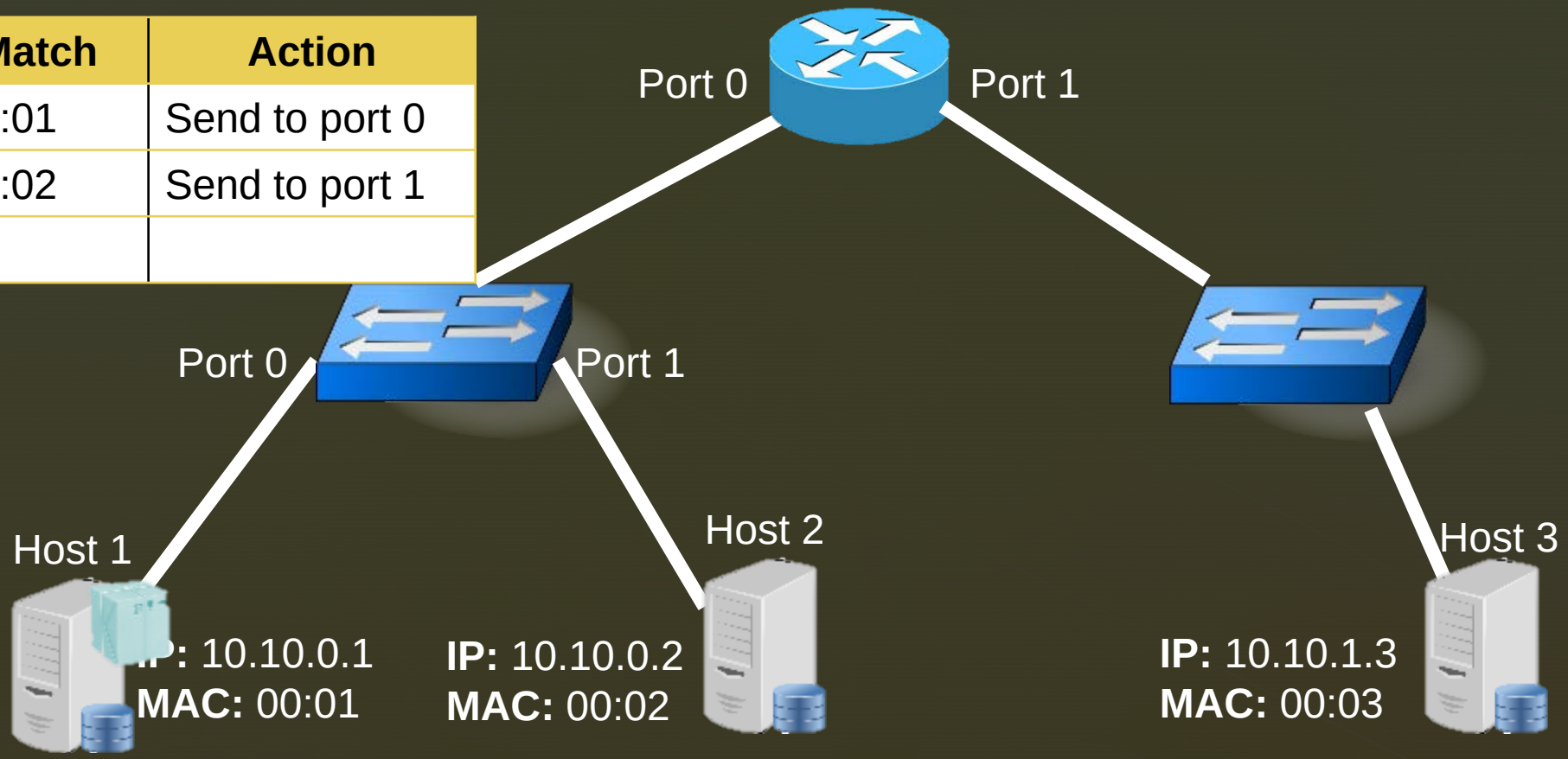
Switch & Networking

MAC Match	Action
MAC: 00:01	Send to port 0
MAC: 00:02	Send to port 1



Switch & Networking

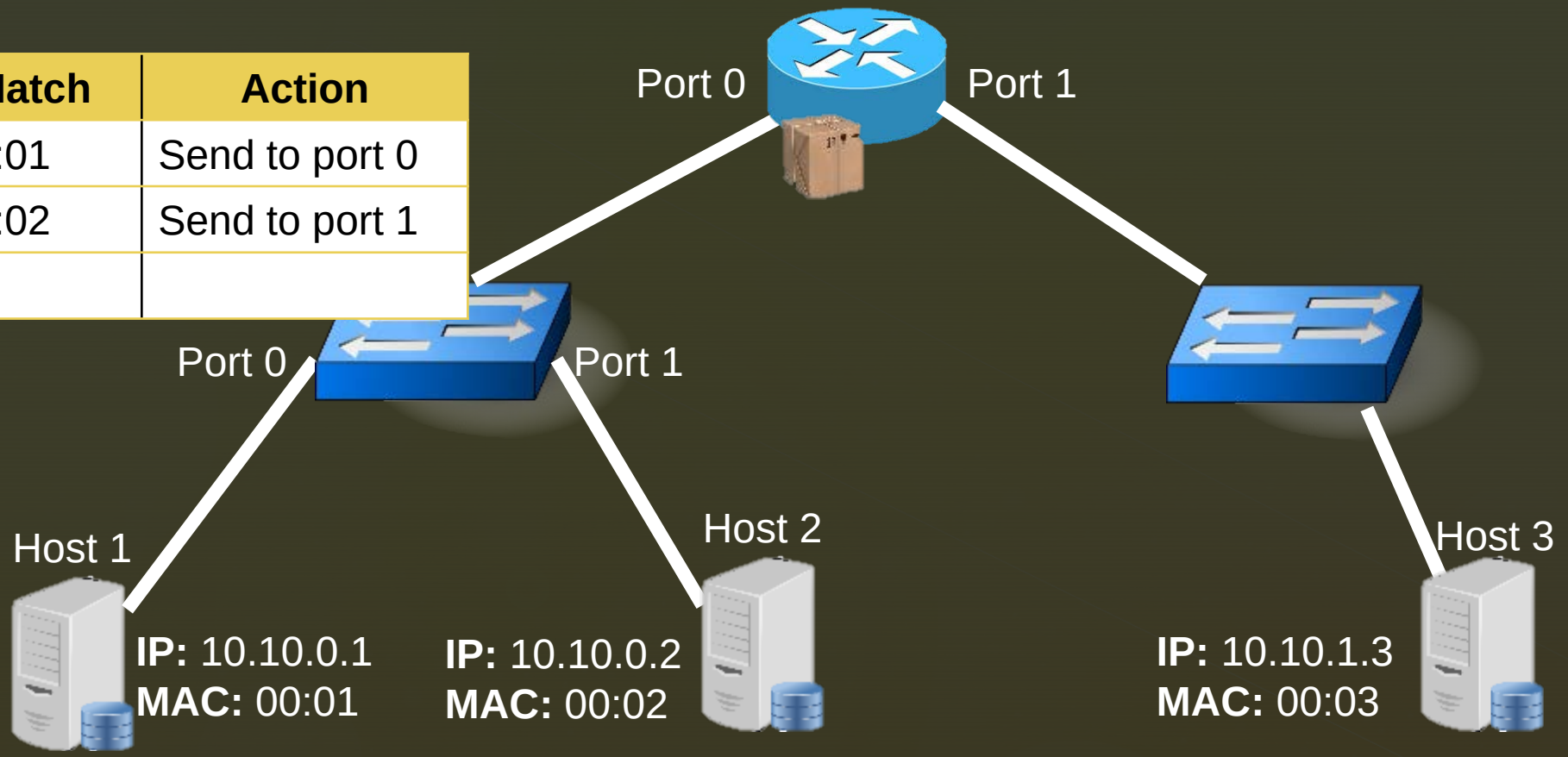
MAC Match	Action
MAC: 00:01	Send to port 0
MAC: 00:02	Send to port 1



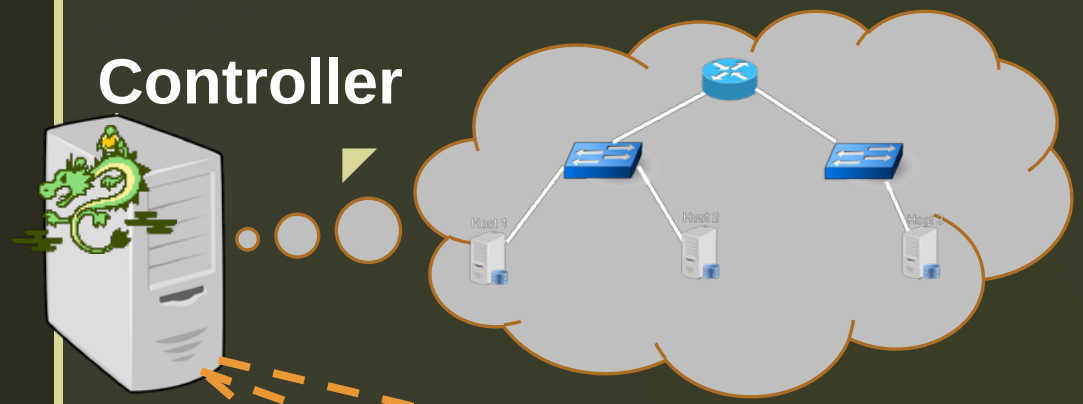
Switch & Networking

IP Match	Action
IP: 10.10.0.0/24	Send to port 0
IP: 10.10.1.0/24	Send to port 1

MAC Match	Action
MAC: 00:01	Send to port 0
MAC: 00:02	Send to port 1



Software Defined Network



Let the controller tells SDN switches what to do

Match	Action

Port 0

Port 1

Match	Action

Port 1

Host 1

IP: 10.10.0.1
MAC: 00:01

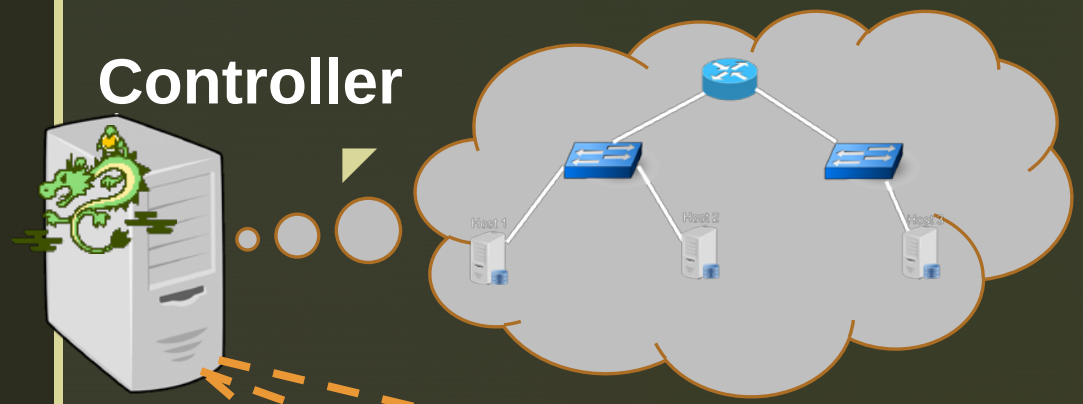
Host 2

IP: 10.10.0.2
MAC: 00:02

Host 3

IP: 10.10.1.3
MAC: 00:03

Software Defined Network



Switches becomes SDN switches, accepting **all kinds of matching fields**

Match	Action

Port 0

Port 1

Match	Action
IP=	
MAC=	
TCP	

Port 1

Host 1

IP: 10.10.0.1
MAC: 00:01

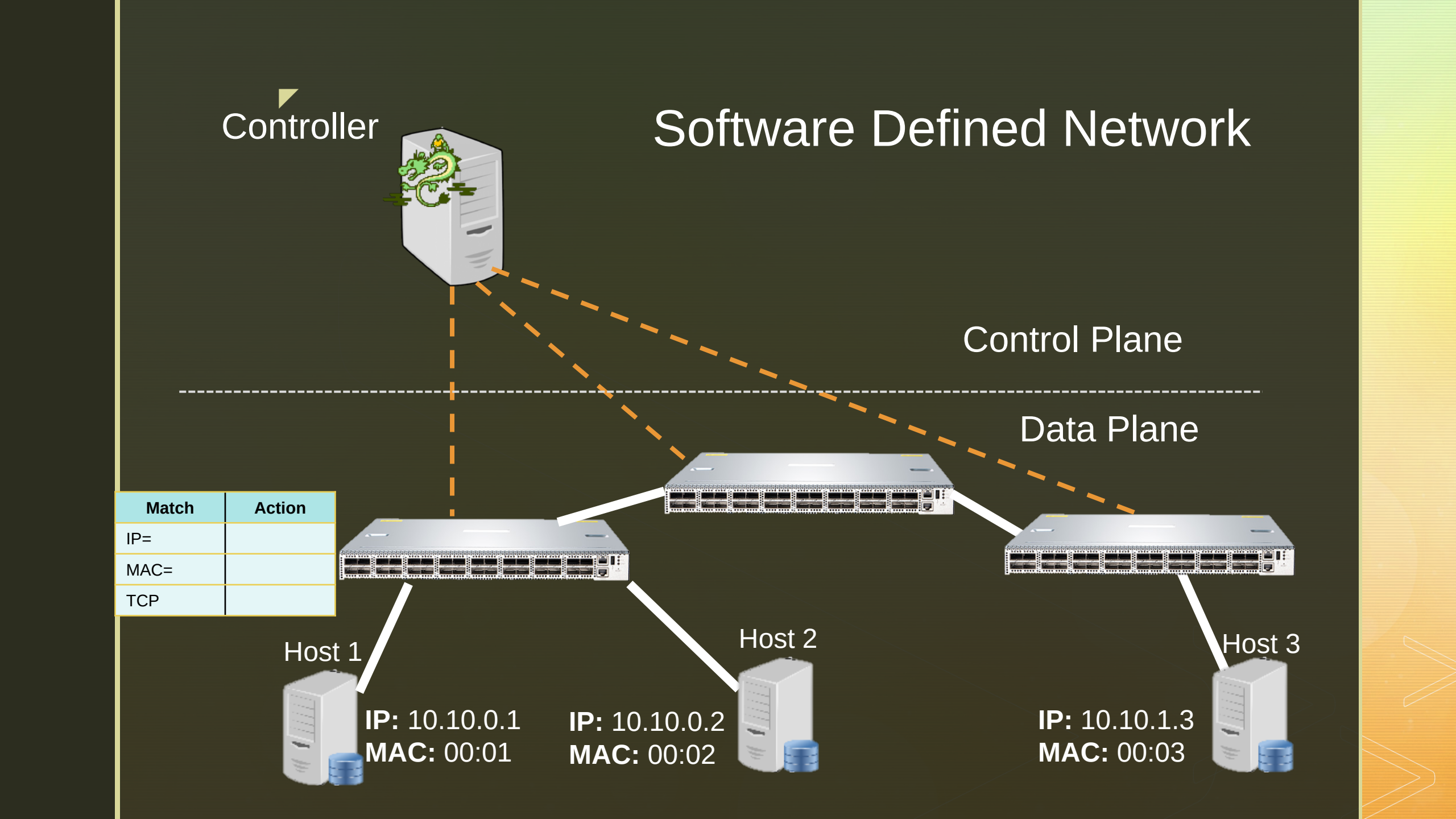
Host 2

IP: 10.10.0.2
MAC: 00:02

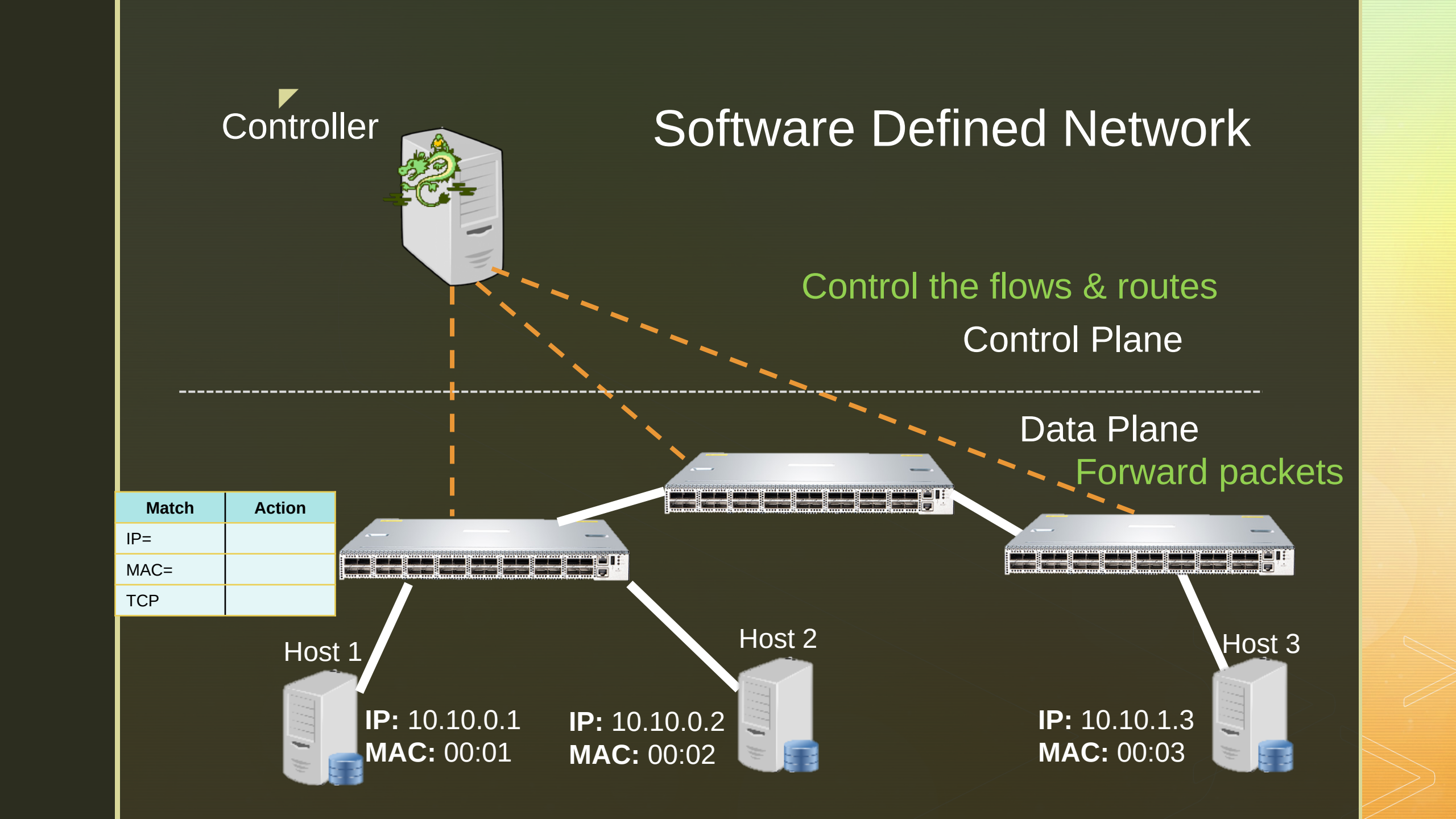
Host 3

IP: 10.10.1.3
MAC: 00:03

Software Defined Network



Software Defined Network



Controller

Software Defined Network

Flow & Packets

	From	To
MAC	00:01	00:02
IP	10.0.0.1	10.0.0.2
TCP	8080	80

Flow

Host 1

Host 2

packets

Controller

Software Defined Network

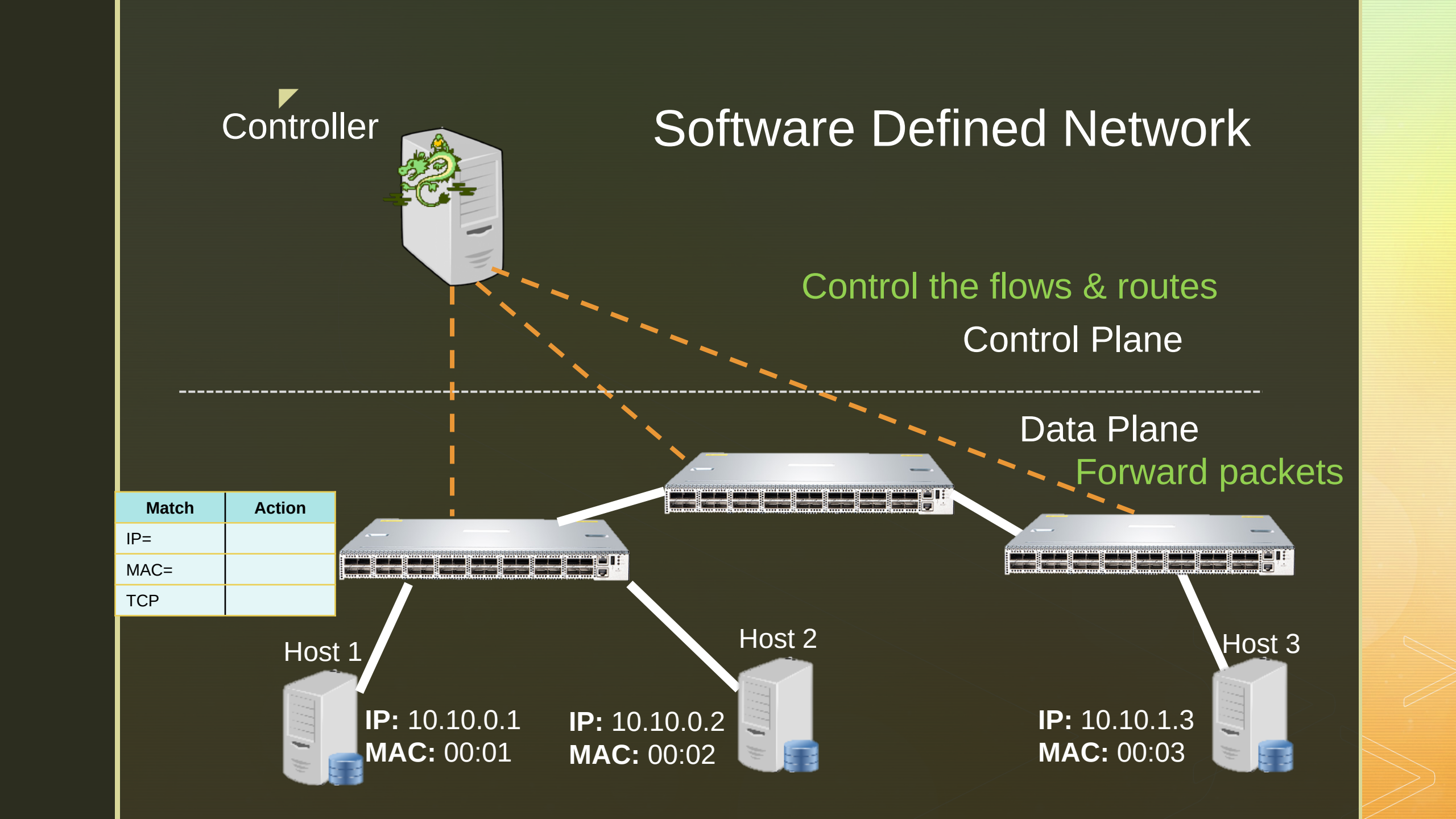
Flow table & Flow entry

Flow entry

Flow table

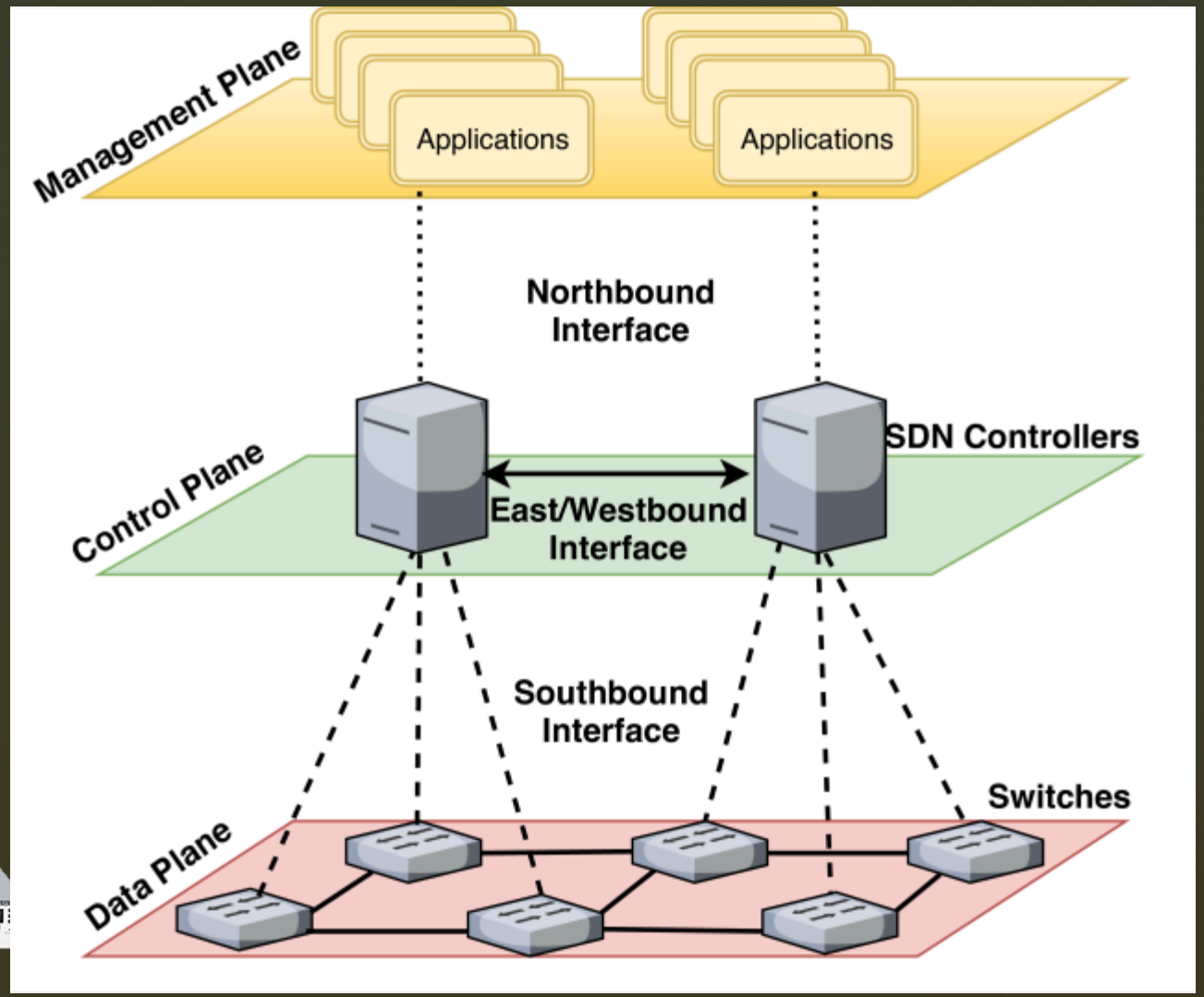
Match	Action
IP=	
MAC=	
TCP	

Software Defined Network

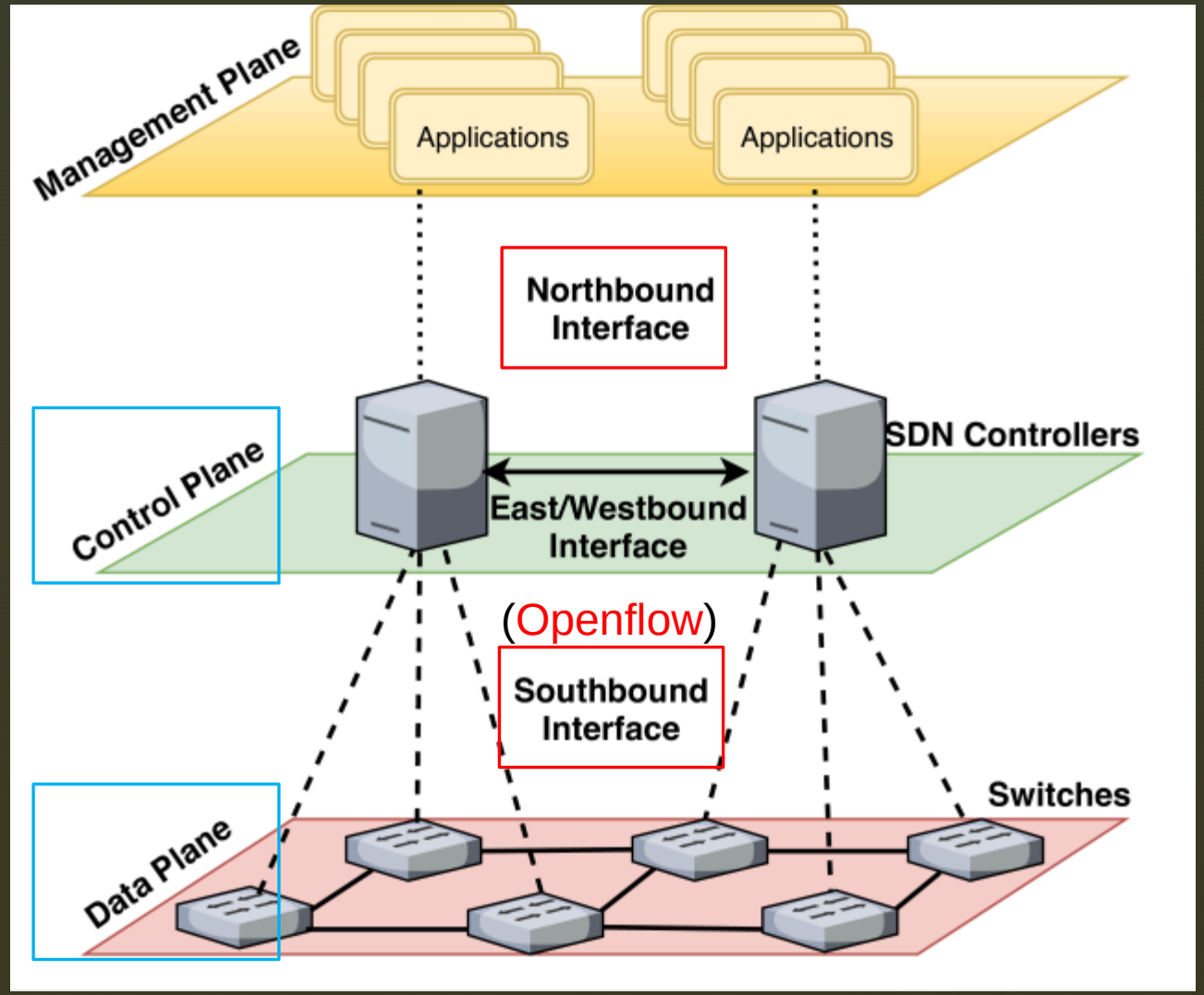


Overview

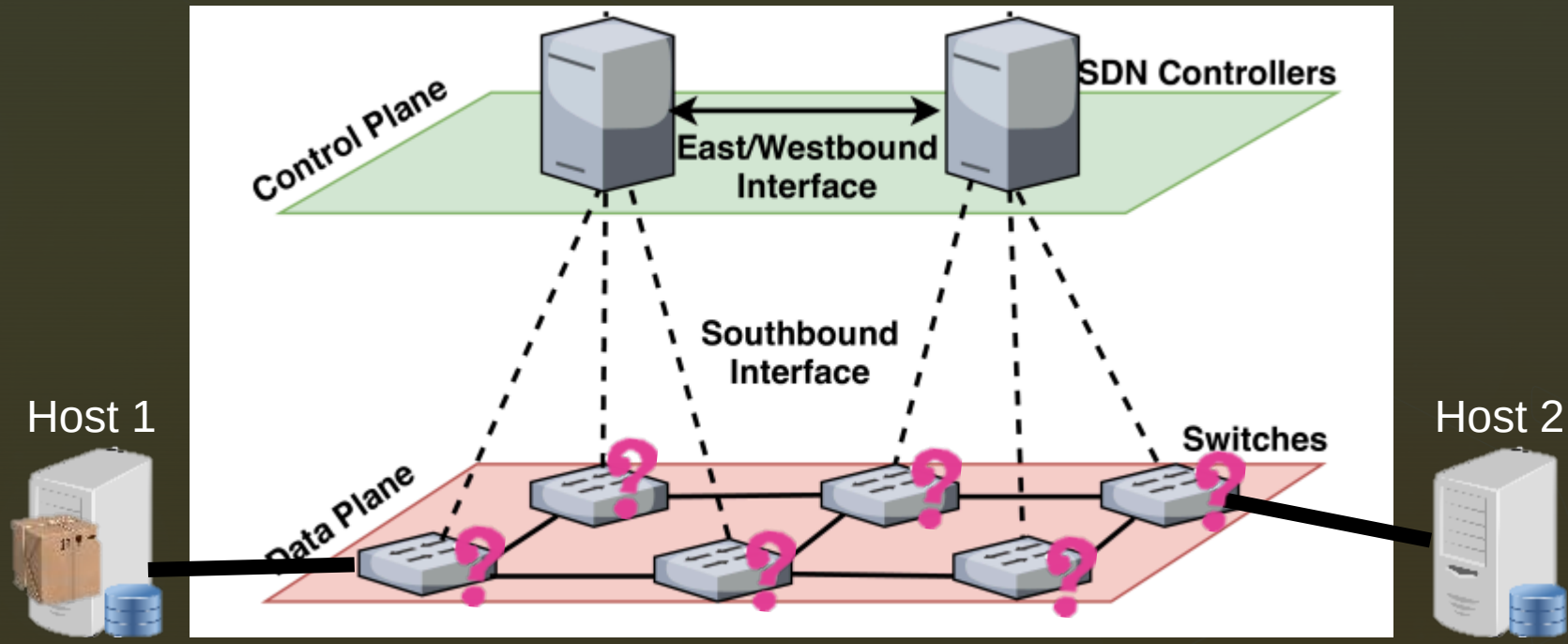
Controller



Overview

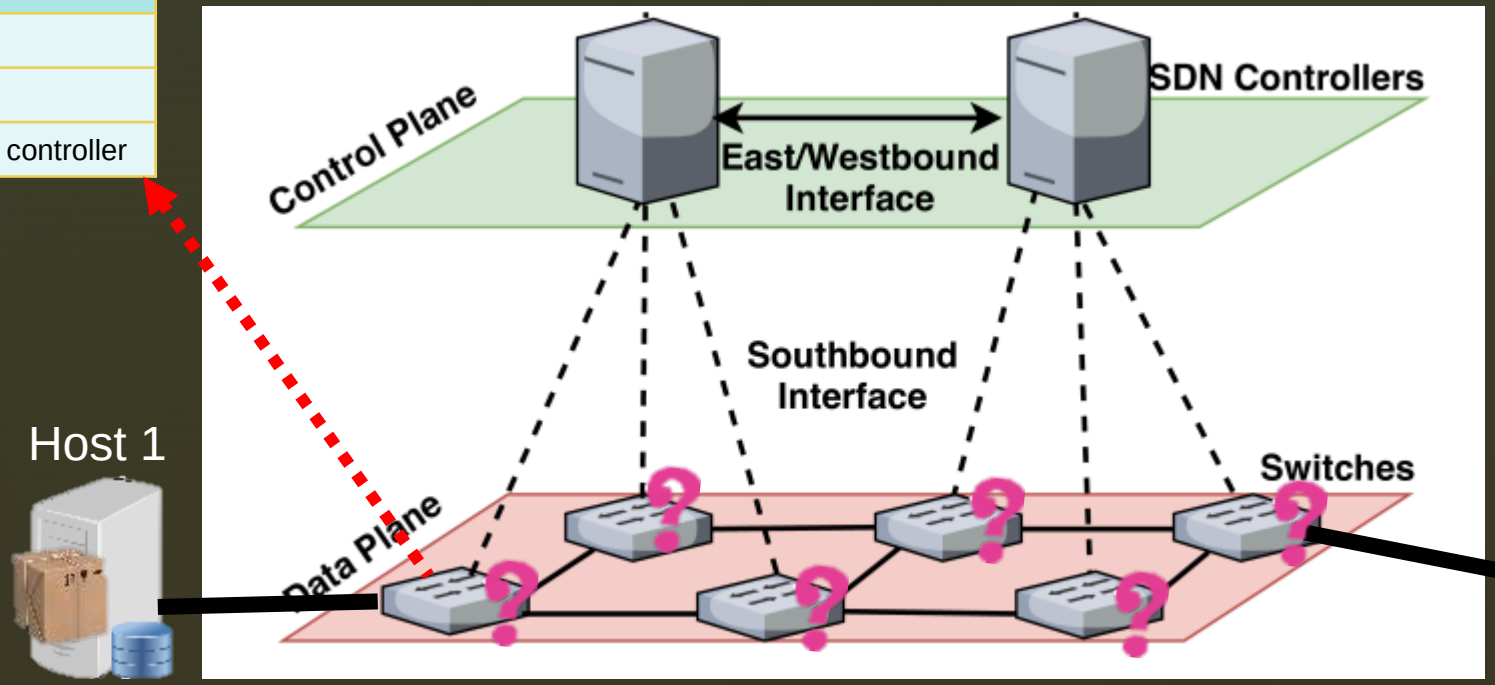

Management Plane
Applications
Applications
Northbound Interface
Control Plane
SDN Controllers
East/Westbound Interface
(Openflow)
Southbound Interface
Switches
Data Plane

Overview


Control Plane
SDN Controllers
East/Westbound
Interface
Southbound
Interface
Switches
Data Plane
Host 1
Host 2

Overview

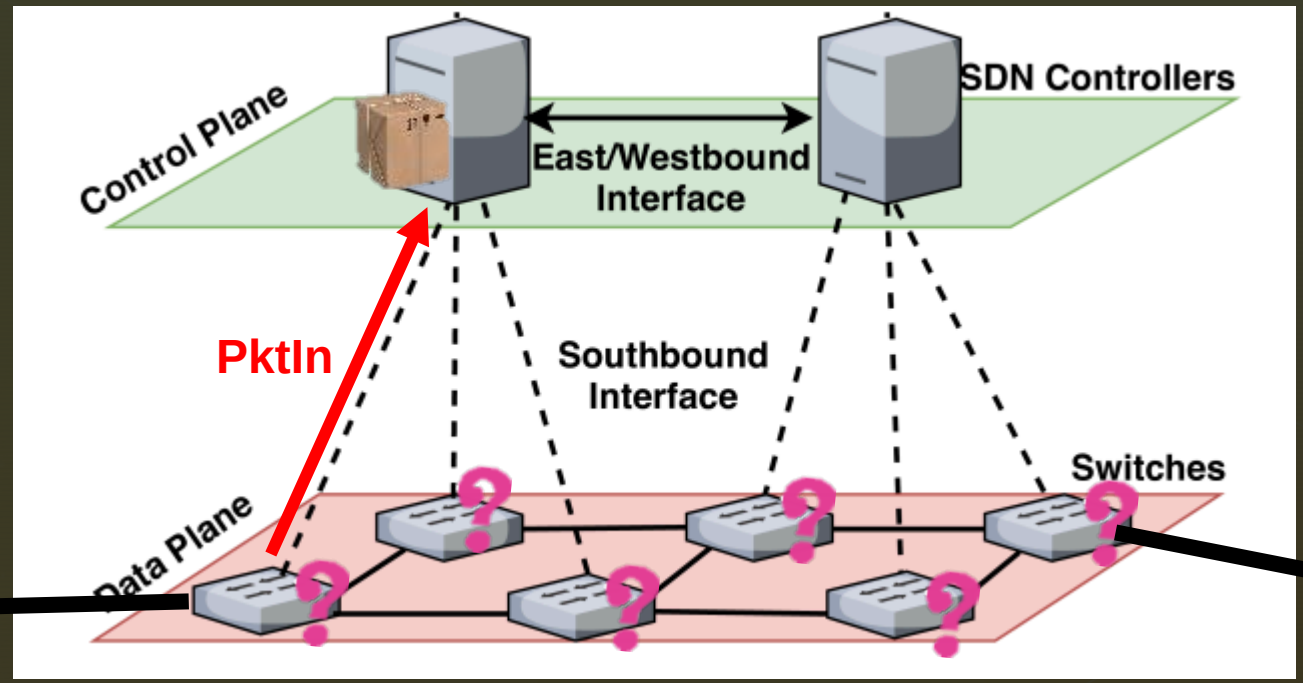
Match	Action
any	To controller



Overview

PktIn: Packet-in

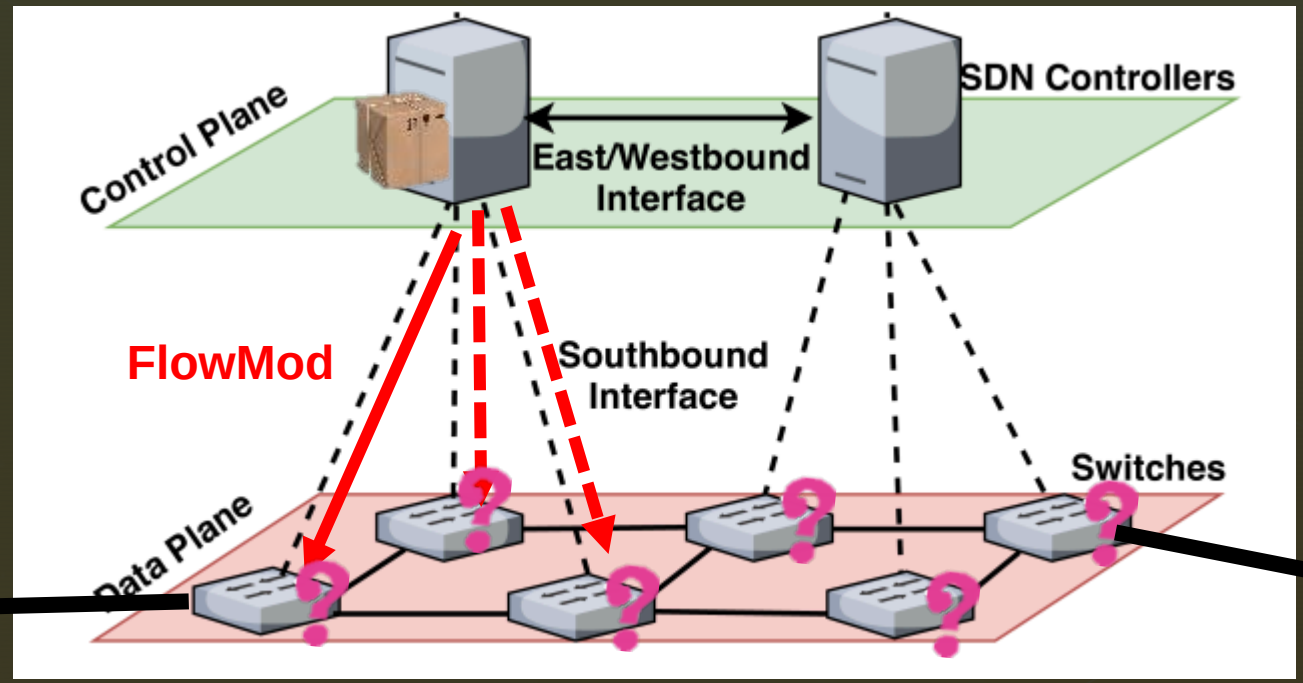
Match	Action
any	To controller



Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification

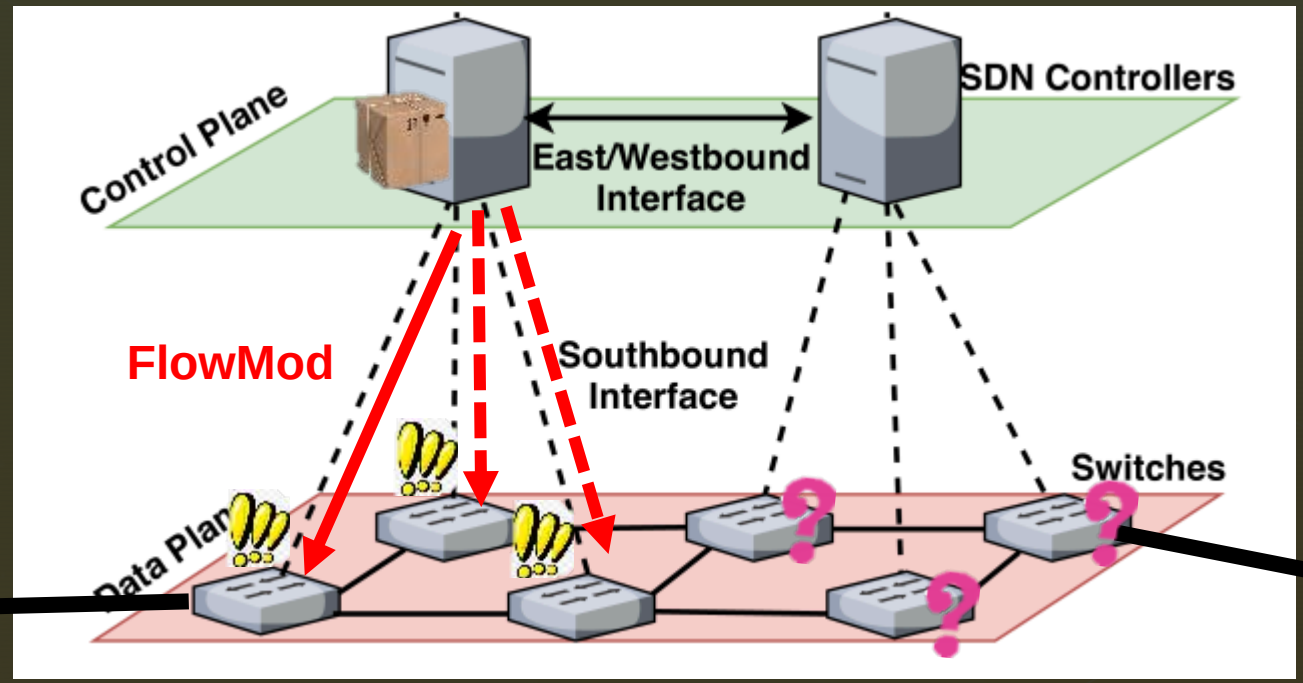
Match	Action
any	To controller



Overview

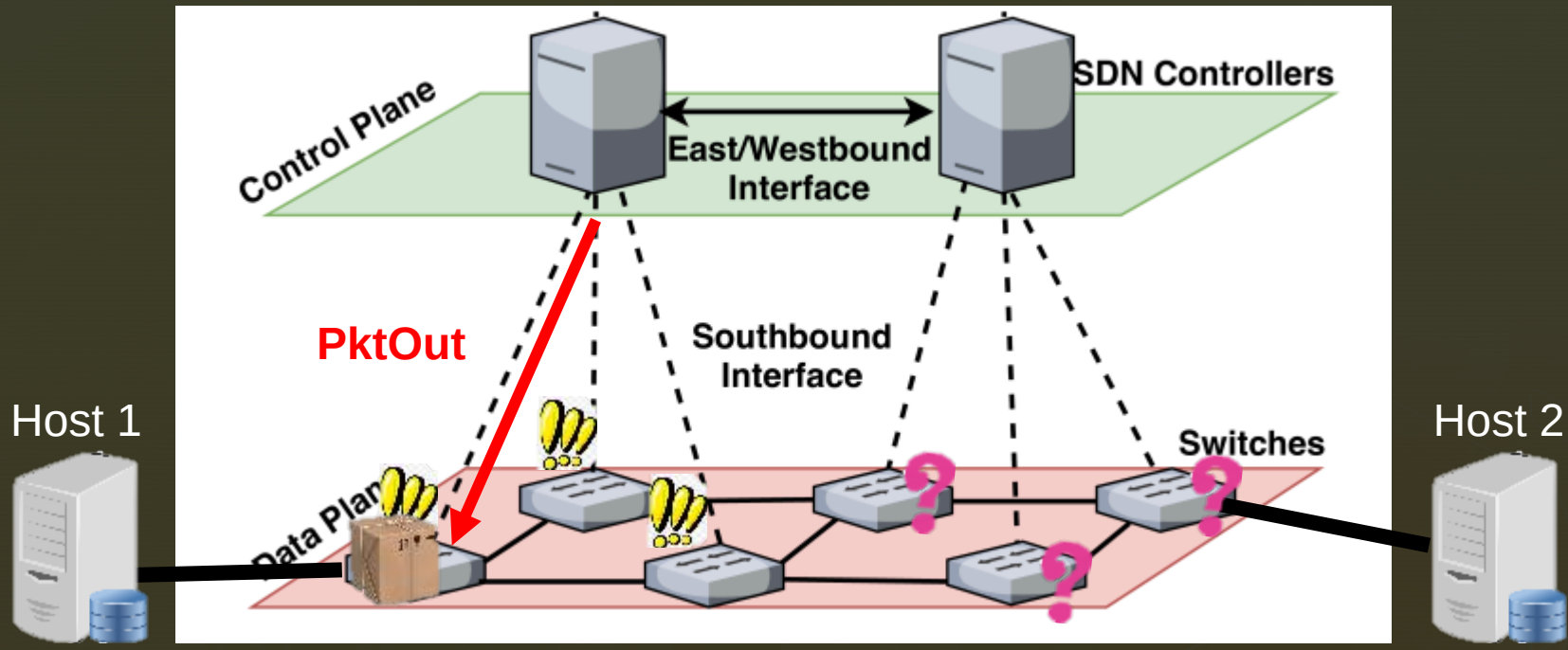
PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification

Match	Action
IP= ;TCP=	To port 1
any	To controller



Overview

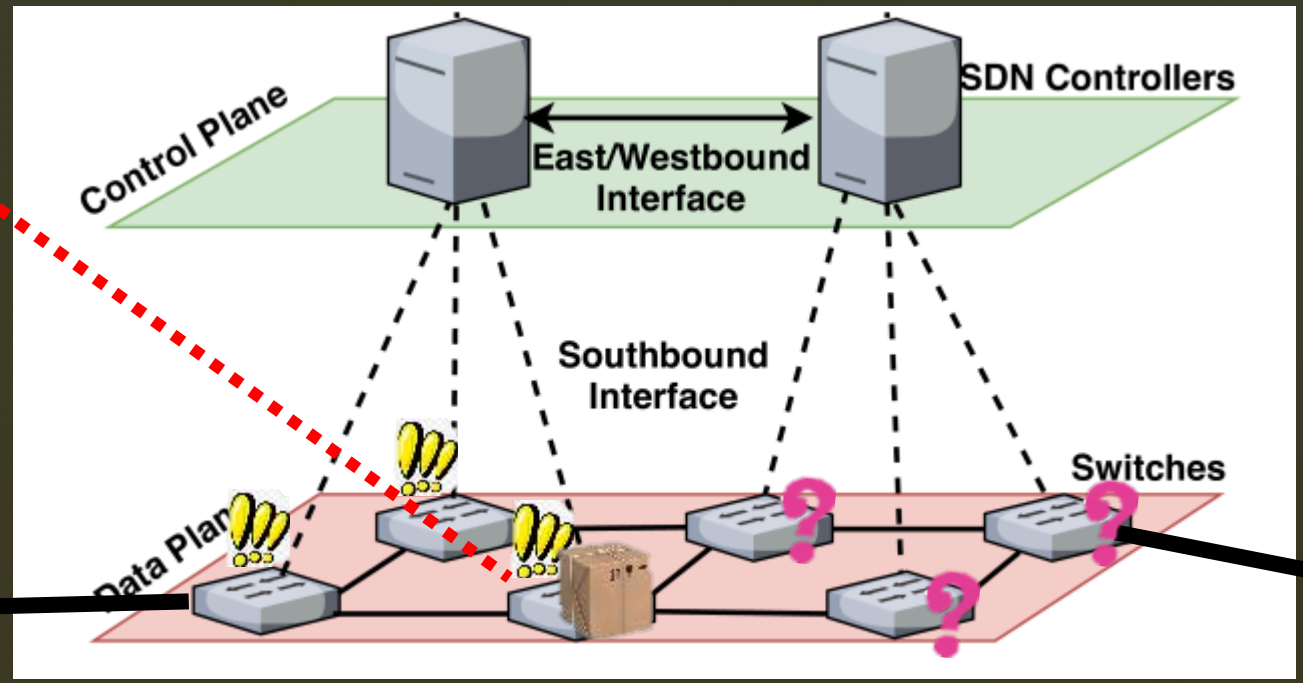
PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification

Match	Action
IP= ;TCP=	To port 2
any	To controller

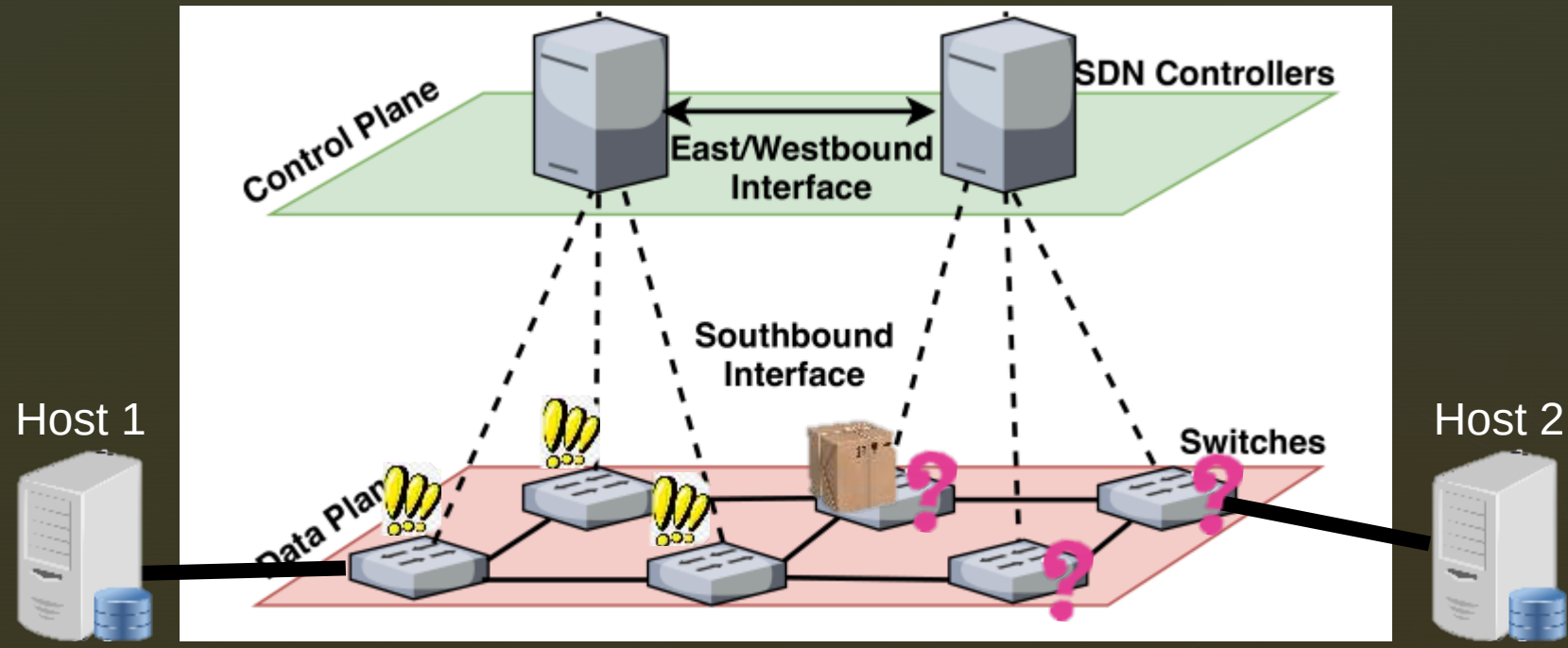


Host 1

Host 2

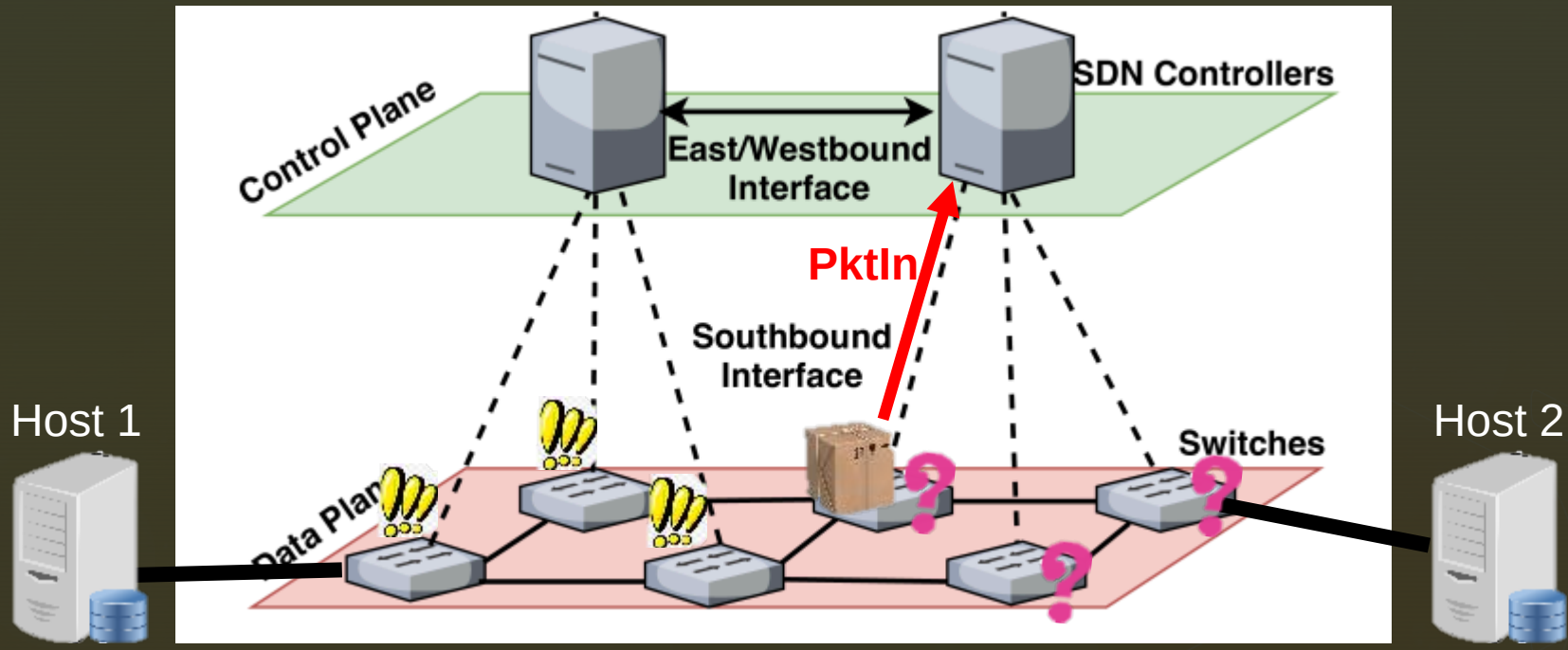
Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



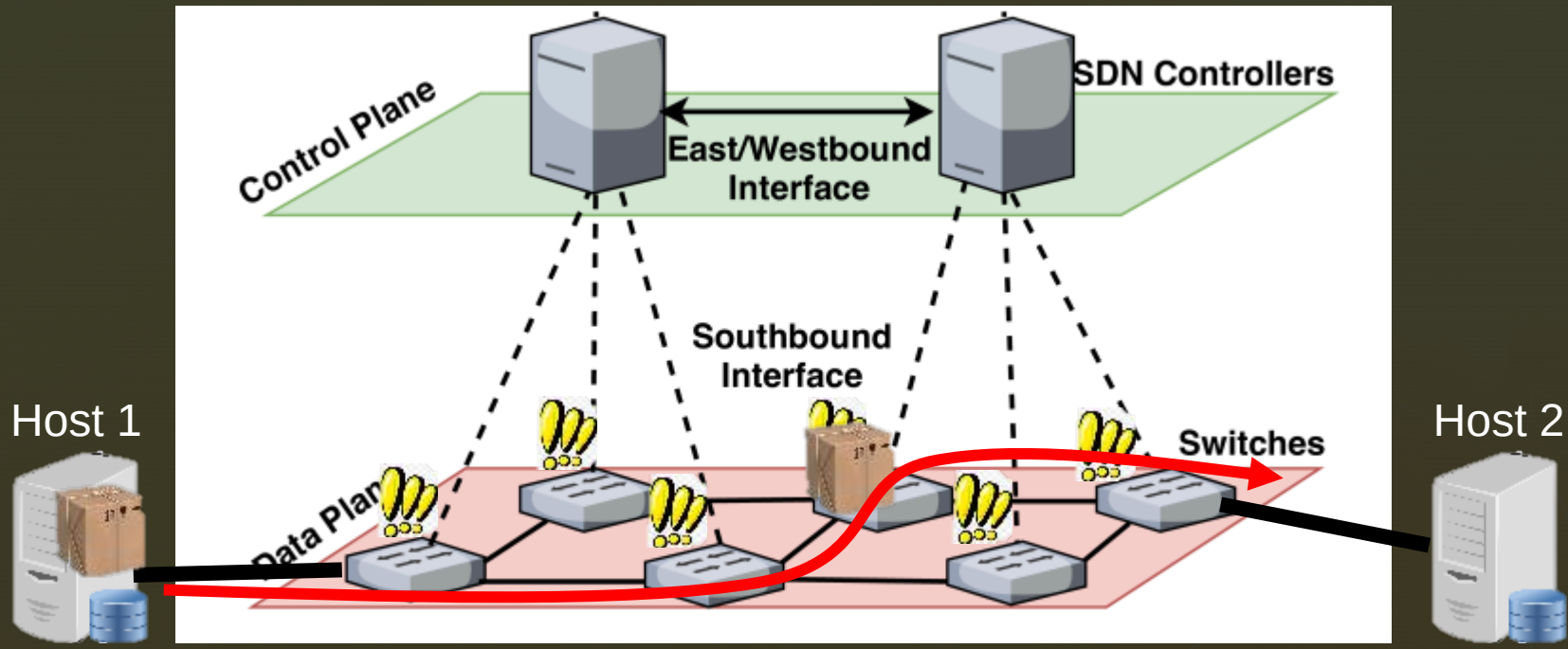
Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



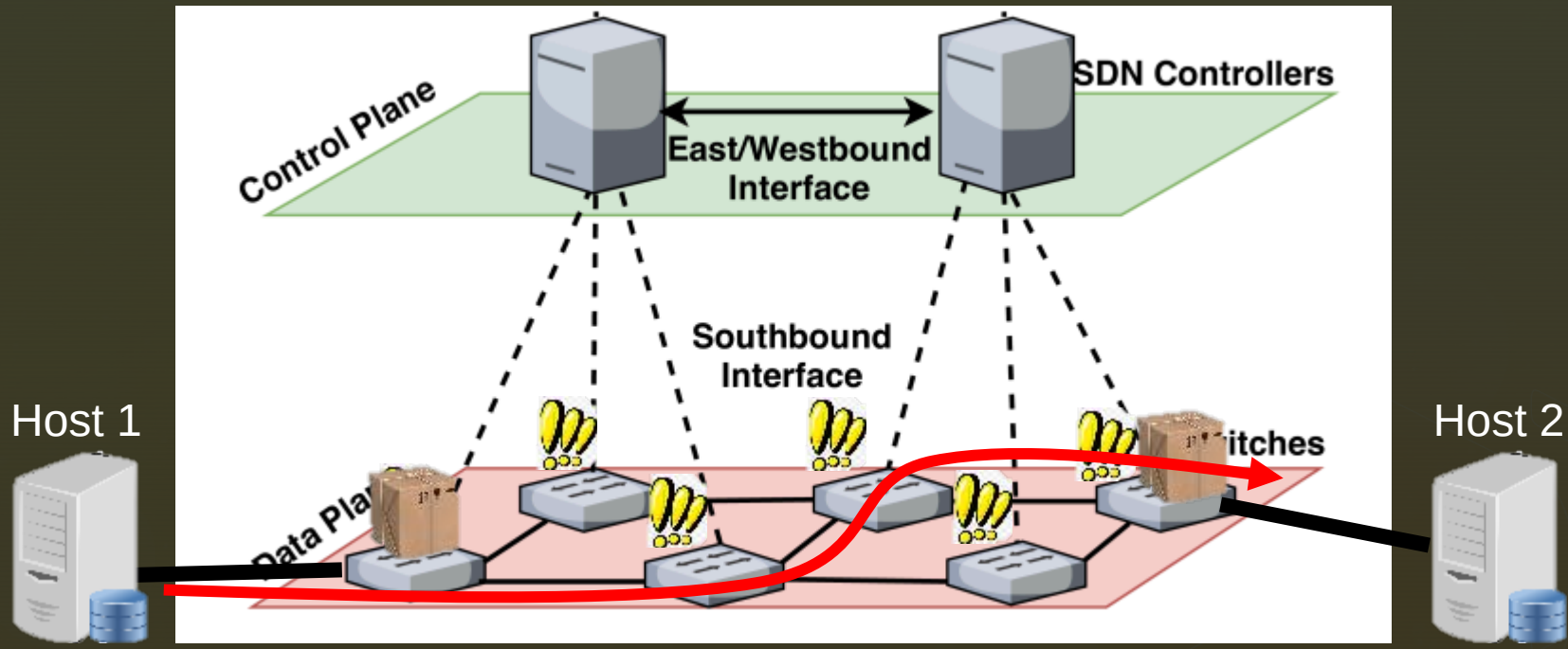
Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



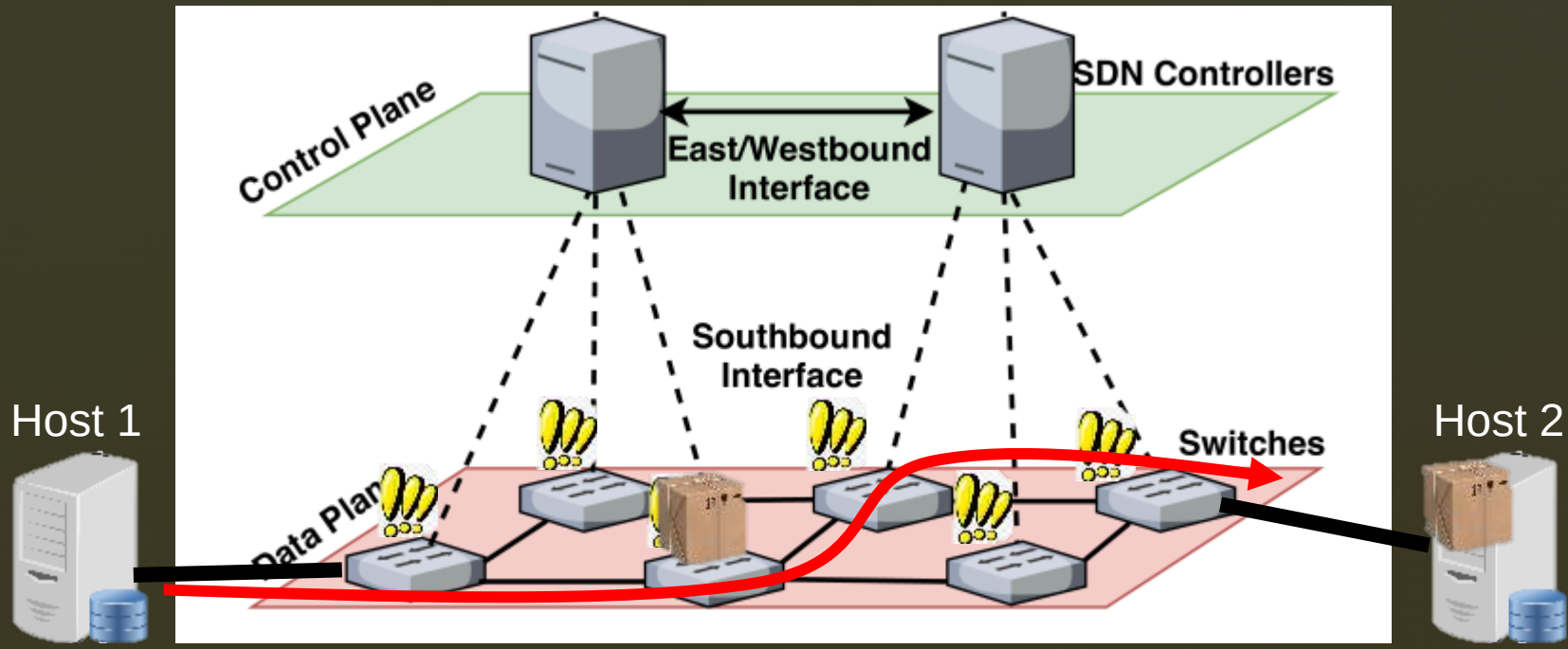
Overview

PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



Overview

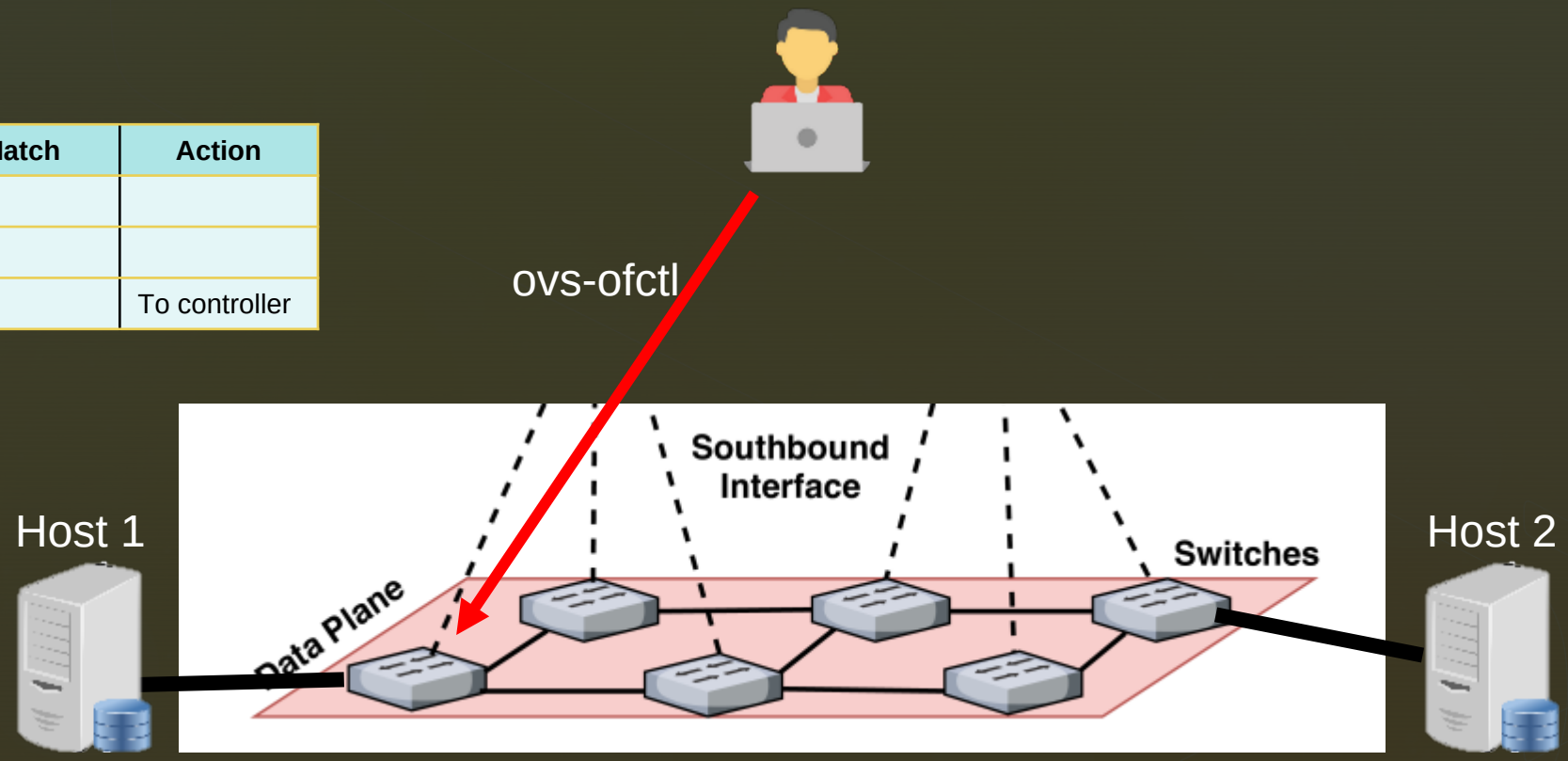
PktIn: Packet-in
PktOut: Packet-out
FlowMod: flow entry modification



In Lab 2

We use OpenFlow command to control SDN switches.
(set up flow entries)

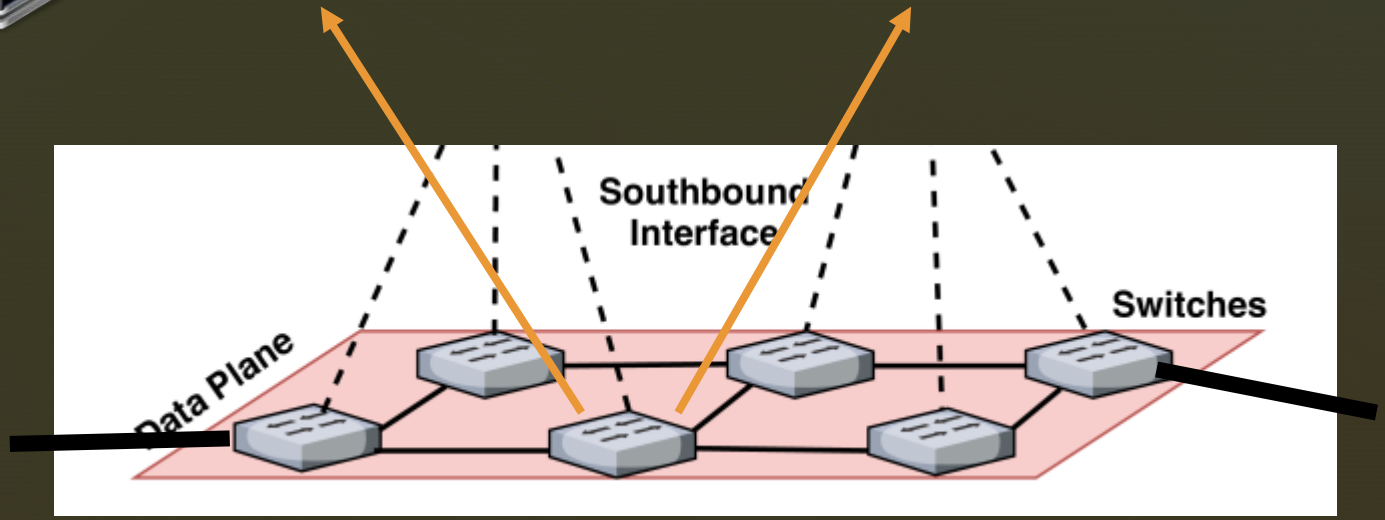
Match	Action
any	To controller



SDN Switch

Software switch

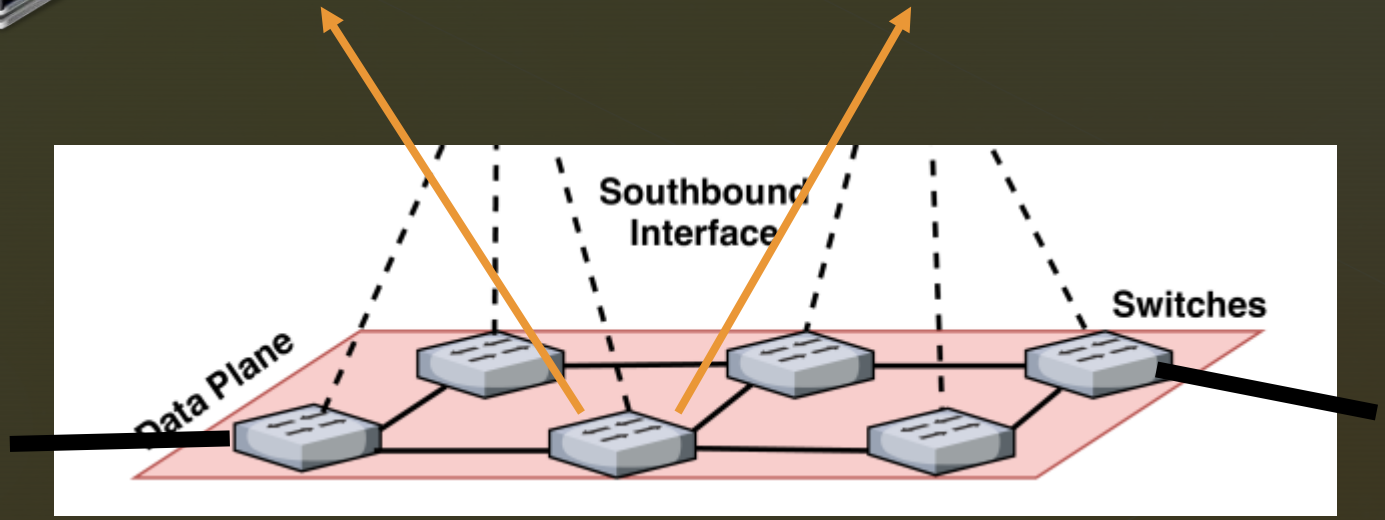
Hardware switch



SDN Switch

Software switch

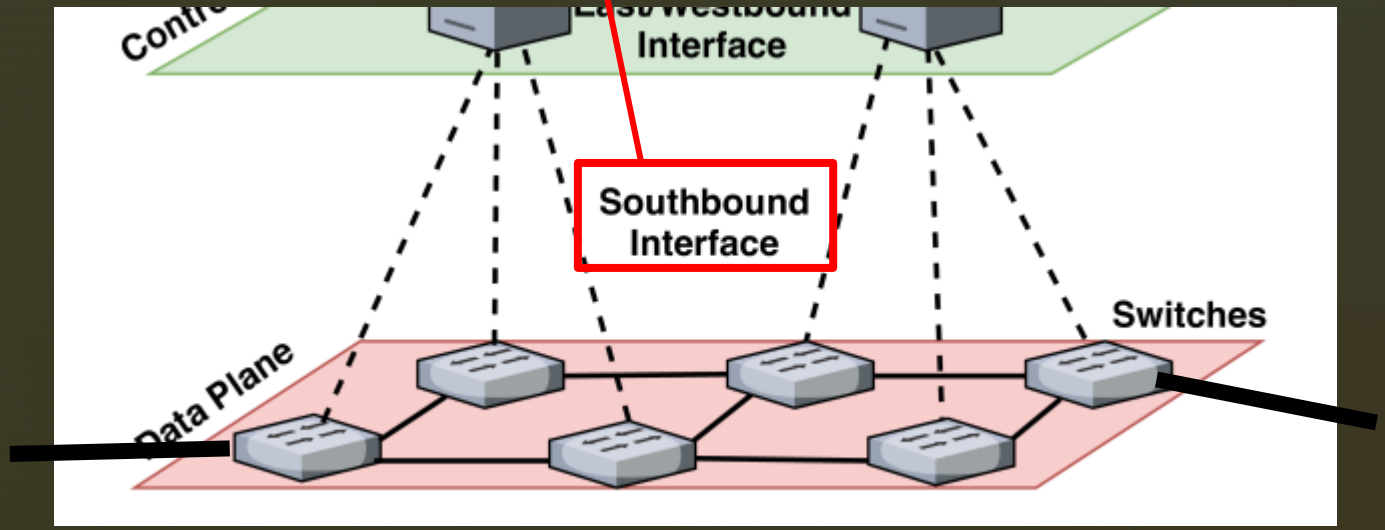
Hardware switch



Terms

Open vSwitch: Software switch

OpenFlows: Protocol between control plane & data plane

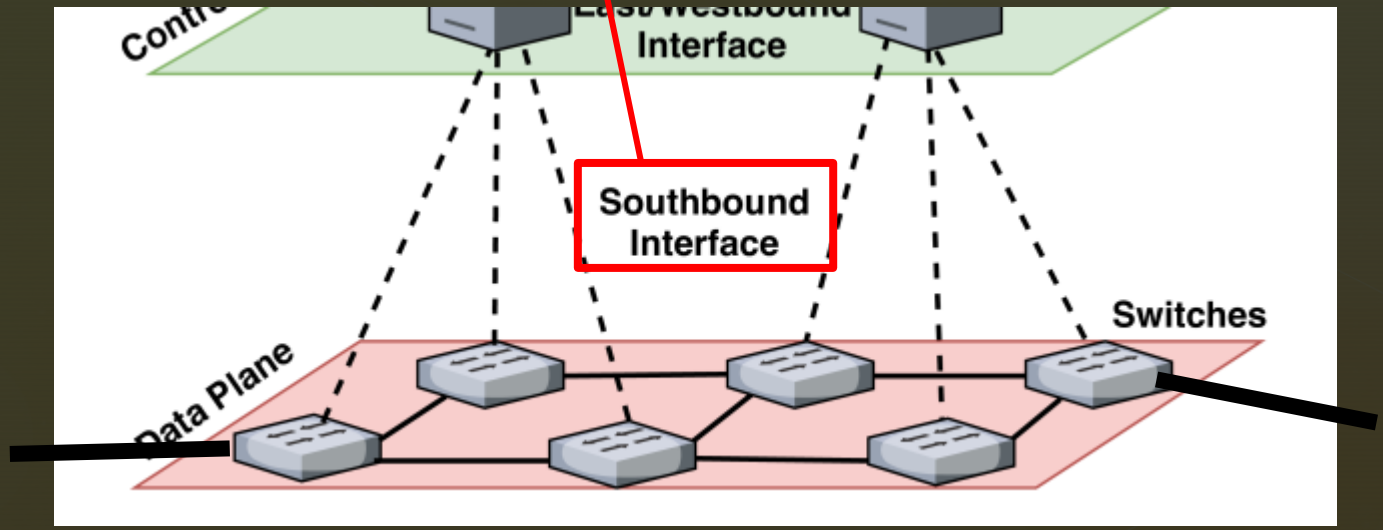


Terms

Open vSwitch: Software switch

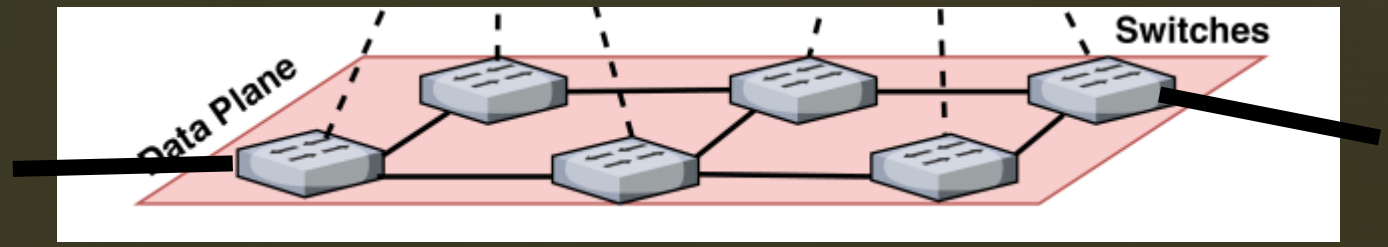
Packet-in
Packet-out
FlowMod

OpenFlows: Protocol between control plane & data plane



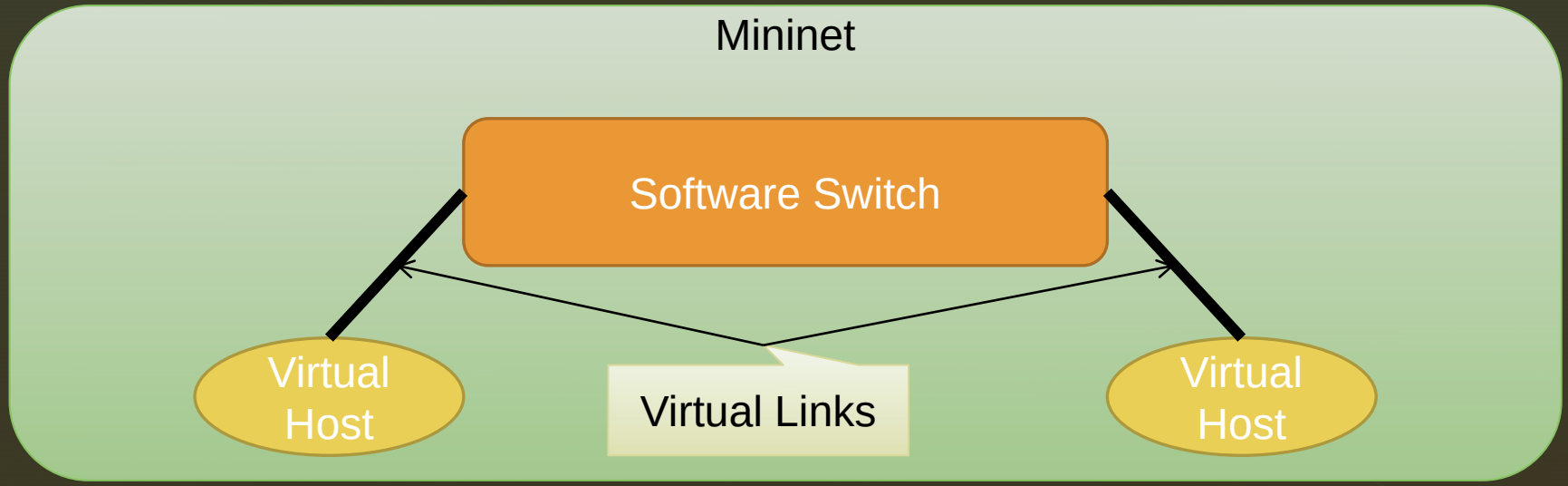
Mininet

- Creates scalable Networks (up to hundreds of nodes)
- Emulator (send real packets within your PC)
- Able to work with any kind of OpenFlow controller
- Easy to program
- Open source project (Free!!)



What does Mininet provide?

- Mininet emulates software defined networks.
- Create virtual hosts and virtual switches
- Test the connectivity and performance in a virtual environment



Lab2

Tasks & Steps

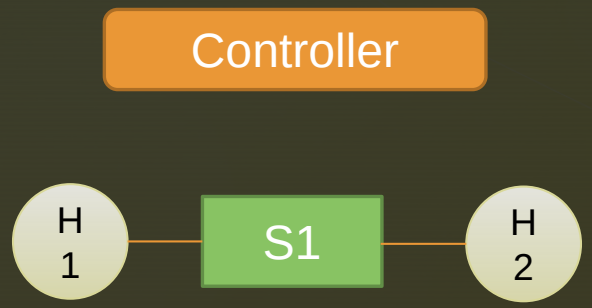
Lab2 Steps

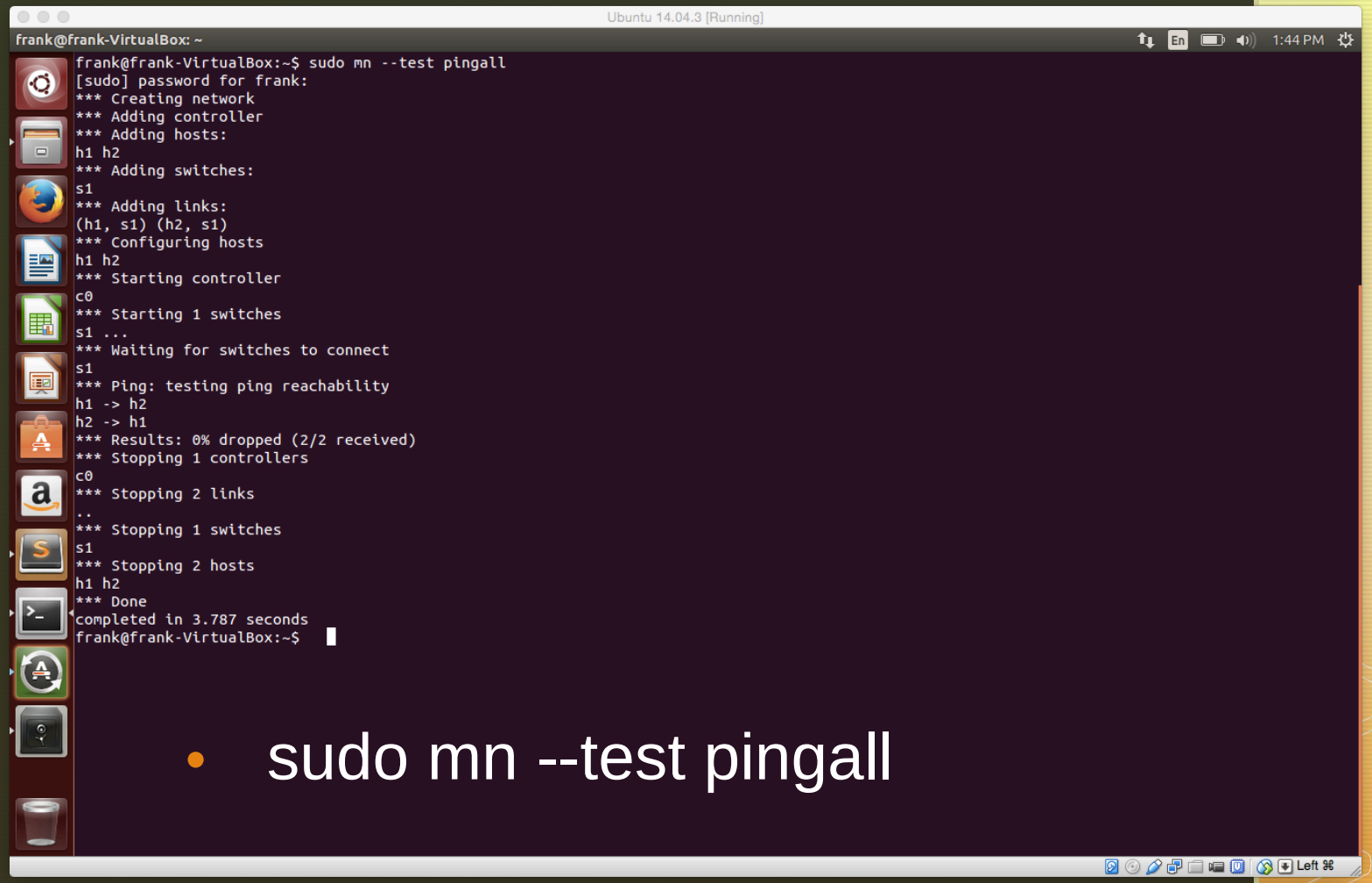
- Install and run mininet
- Follow mininet walkthrough and learn its commands
- Capture OpenFlow messages passed through mininet hosts
- Create a customized topology
- Install custom forwarding rules on switches

Installing mininet (Option 1) - recommended

- Install a Ubuntu VM using either VMWare Player or VirtualBox
- Install mininet through apt-get (remember to update first)
  - sudo apt-get update
  - sudo apt-get install mininet
- Test installed mininet
  - sudo mn --test pingall
- To install default ovs controller
  - sudo apt-get install openvswitch-testcontroller
  - sudo cp /usr/bin/ovs-testcontroller /usr/bin/ovs-controller

Running mininet





- sudo mn --test pingall

Mininet Walkthrough

- Follow the official mininet walkthrough
  - http://mininet.org/walkthrough/
  - Example:

Display Startup Options

Let's get started with Mininet's startup options.

Type the following command to display a help message describing Mininet's startup options:

```
$ sudo mn -h
```

- Learn
  - How to Run mininet
  - mininet's basic commands

1. Observe the OpenFlow Control message

1. Running Mininet with Custom Topology

1. Run the Mininet topology

Open Wireshark in Ubuntu and start!!

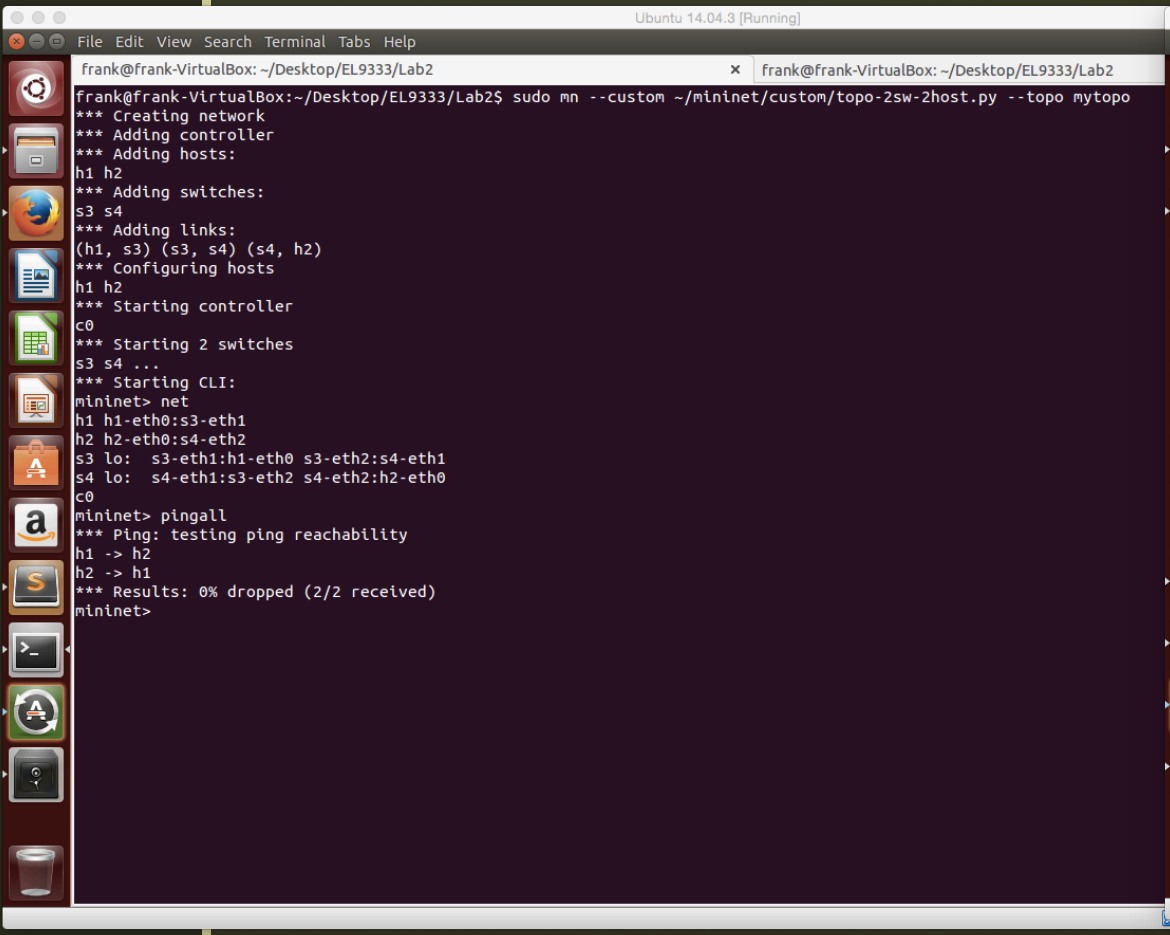
- To launch Mininet with **default controller**:
  - sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo

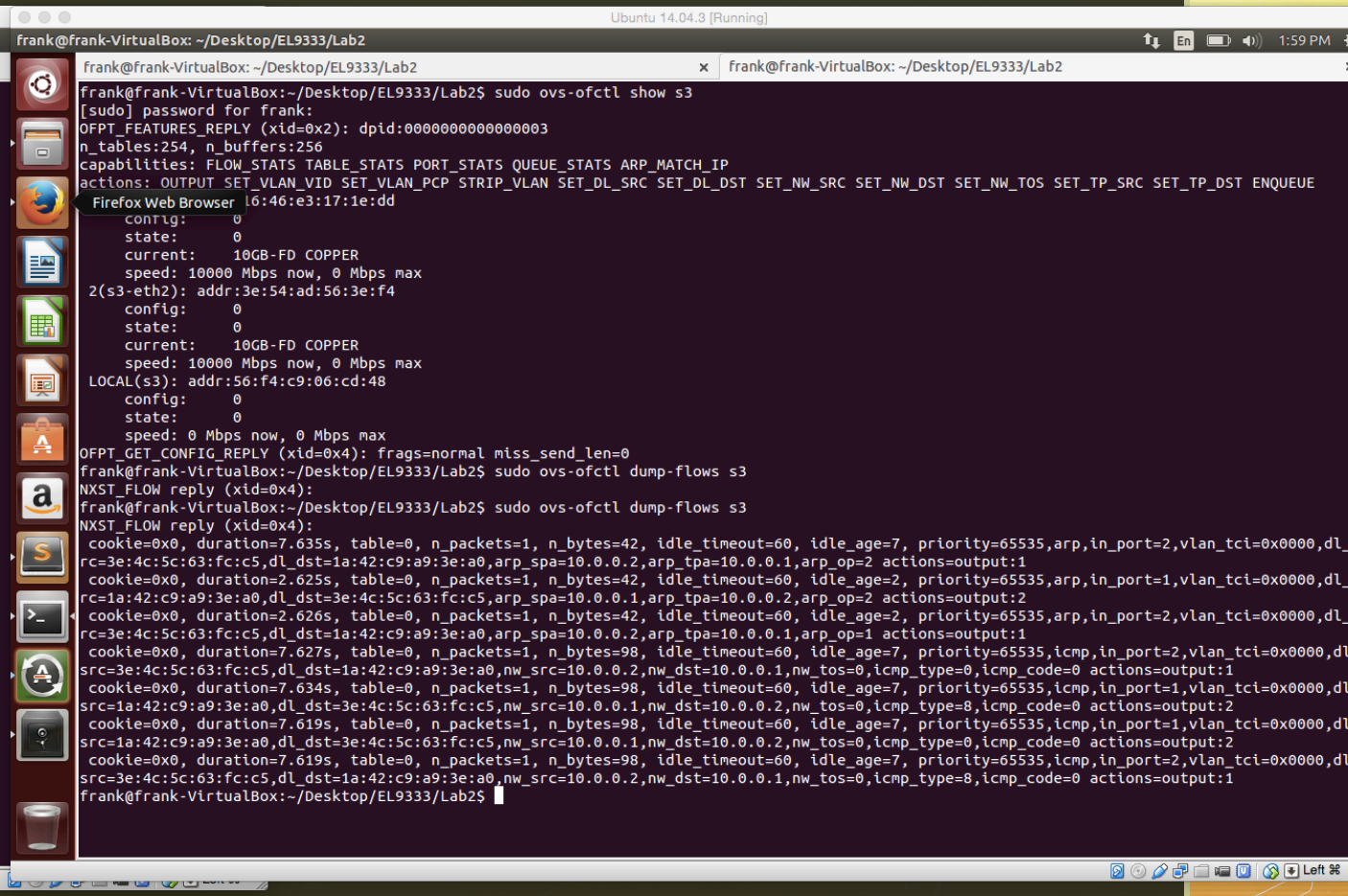
No “--controller remote” !

1. Running Mininet with Custom Topology

1. Run the Mininet topology

2. Open another terminal for observation





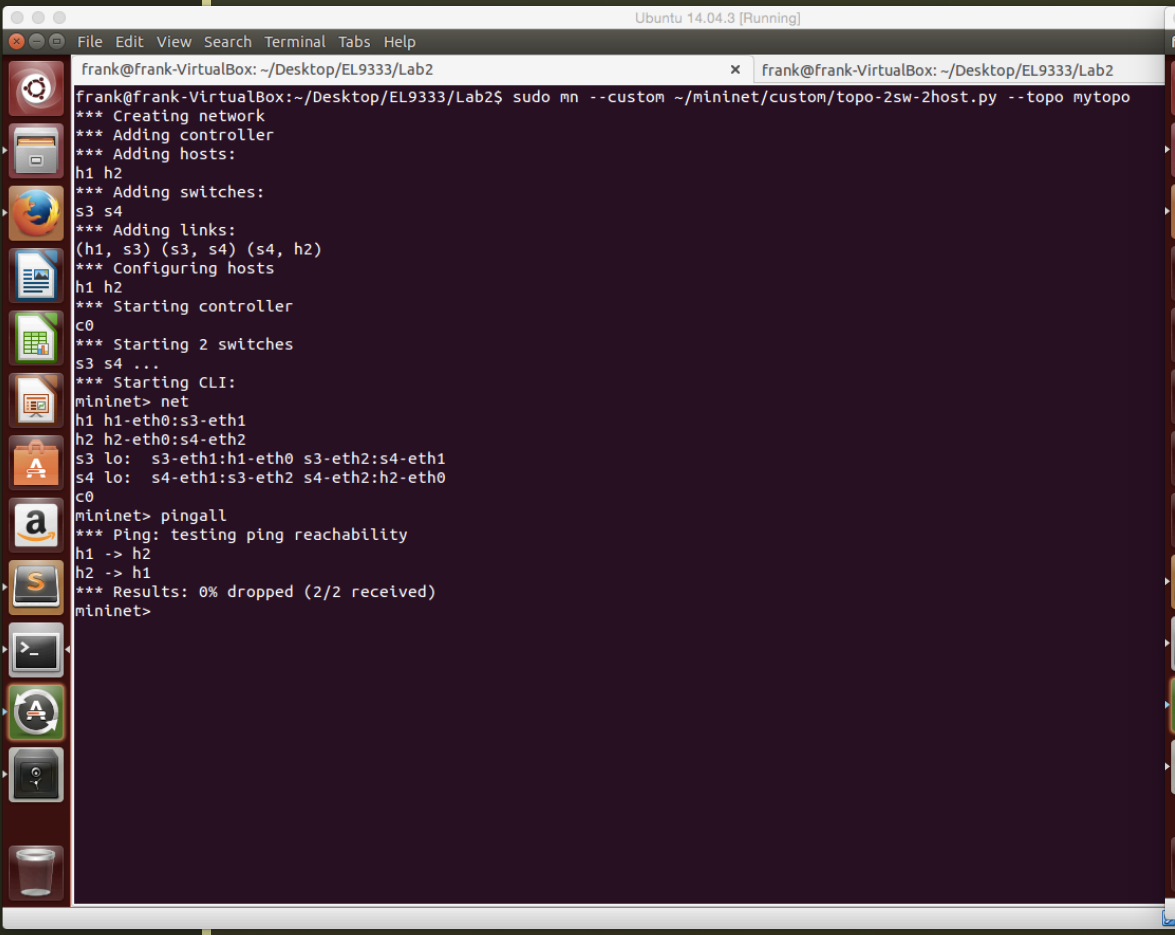
1. Running Mininet with Custom Topology

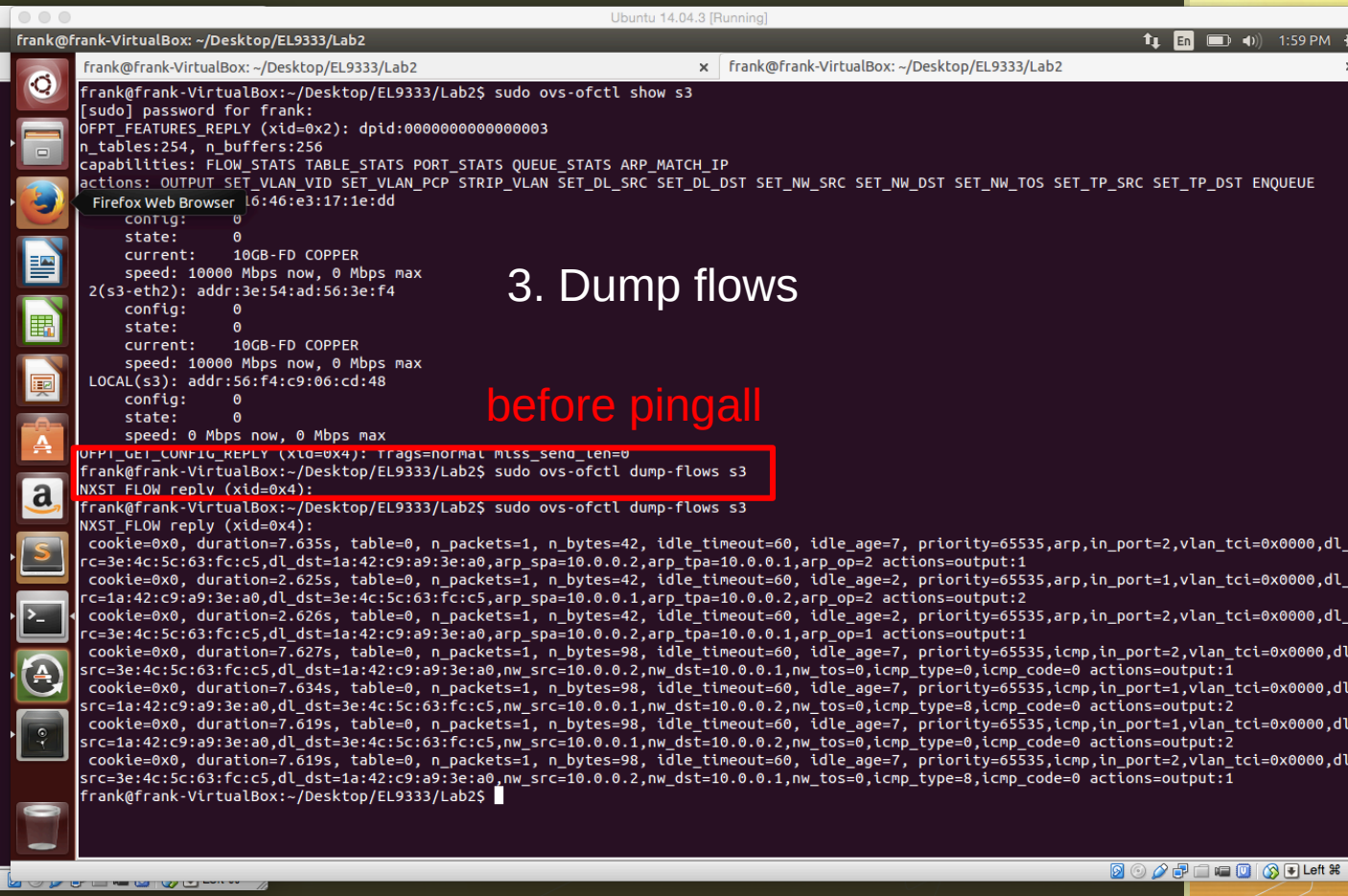
1. Run the Mininet topology

2. Open another terminal for observation

3. Dump flows

before pingall

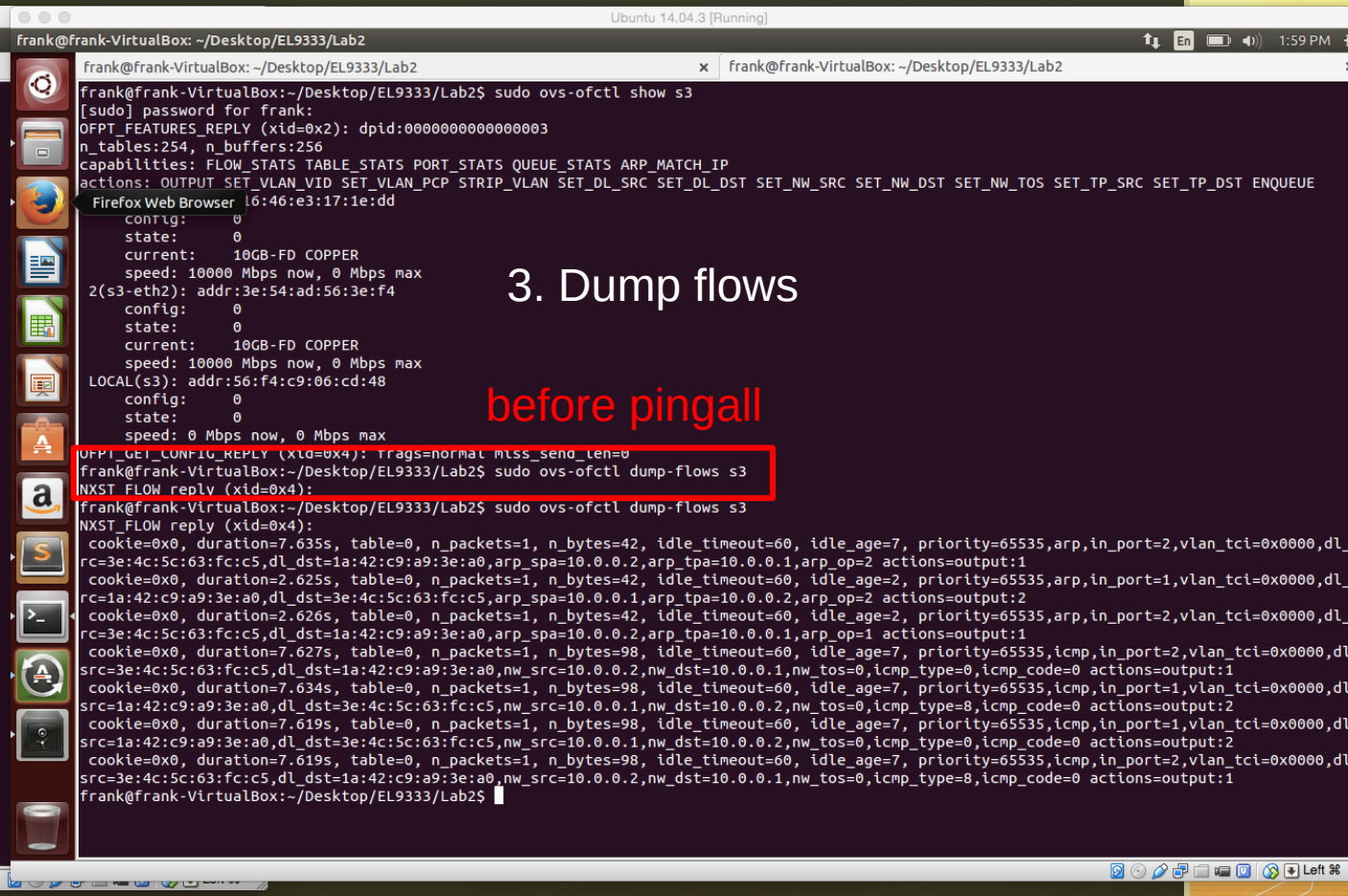




1. Running Mininet with Custom Topology

1. Run the Mininet topology

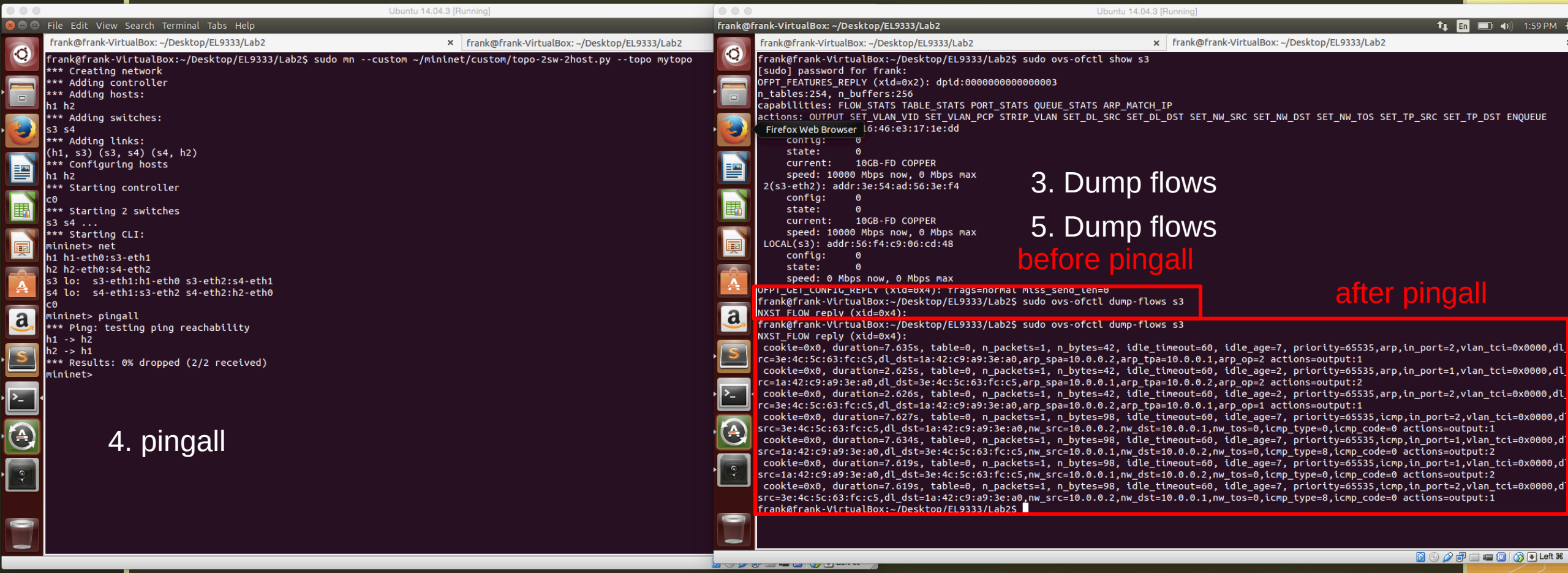
2. Open another terminal for observation



1. Running Mininet with Custom Topology

1. Run the Mininet topology

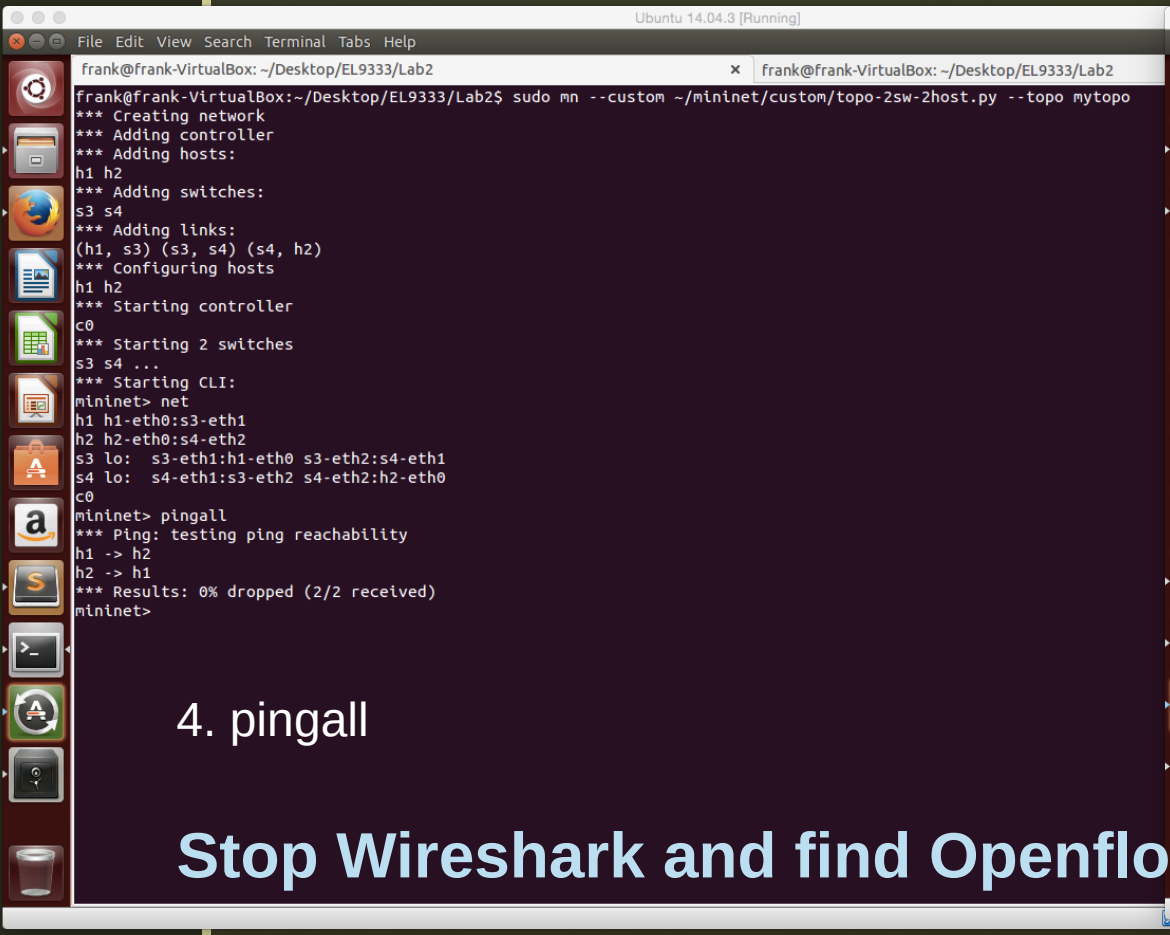
2. Open another terminal for observation



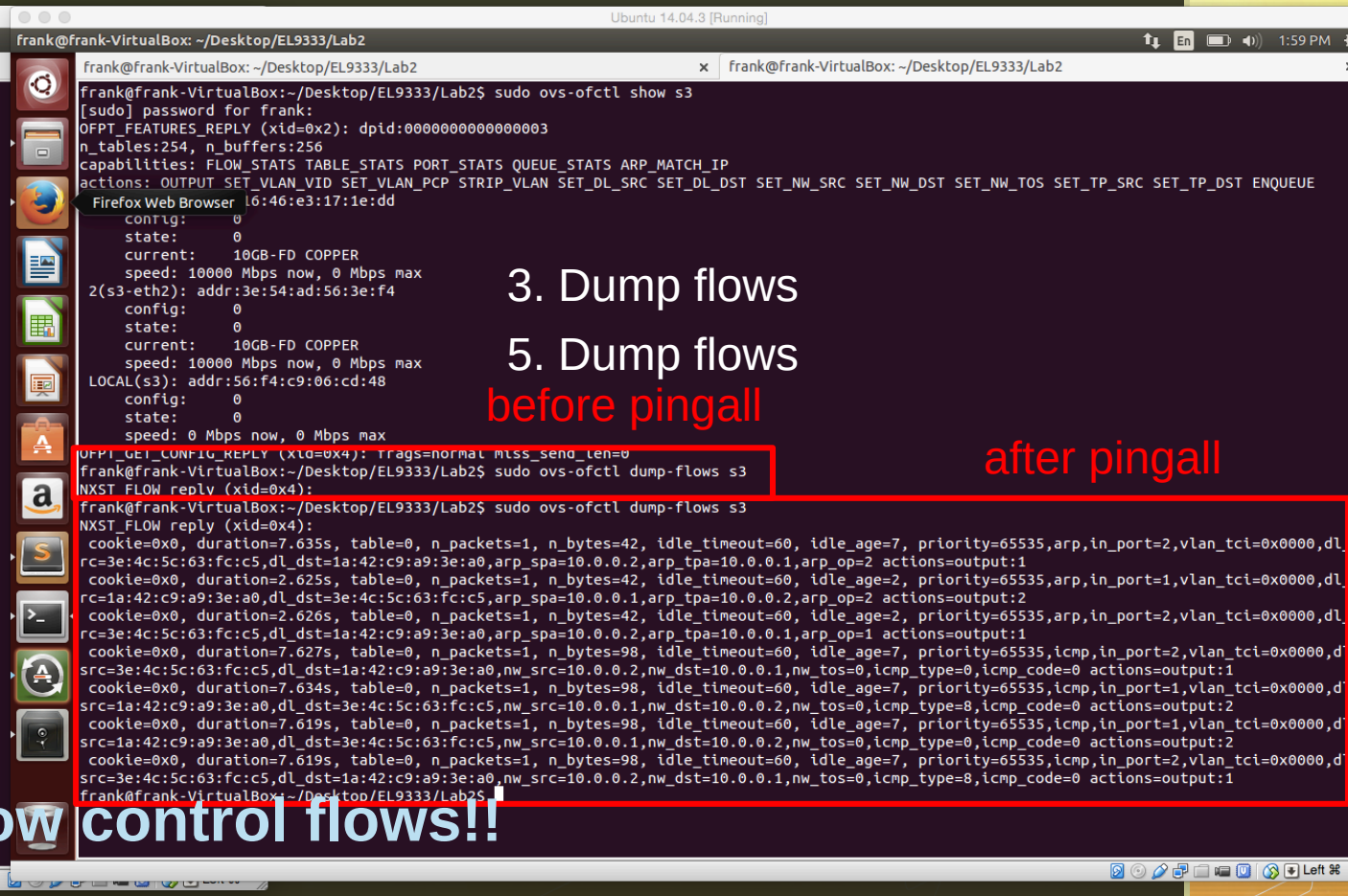
Running Mininet with Custom Topology

1. Run the Mininet topology

2. Open another terminal for observation



4. pingall



3. Dump flows

5. Dump flows

before pingall

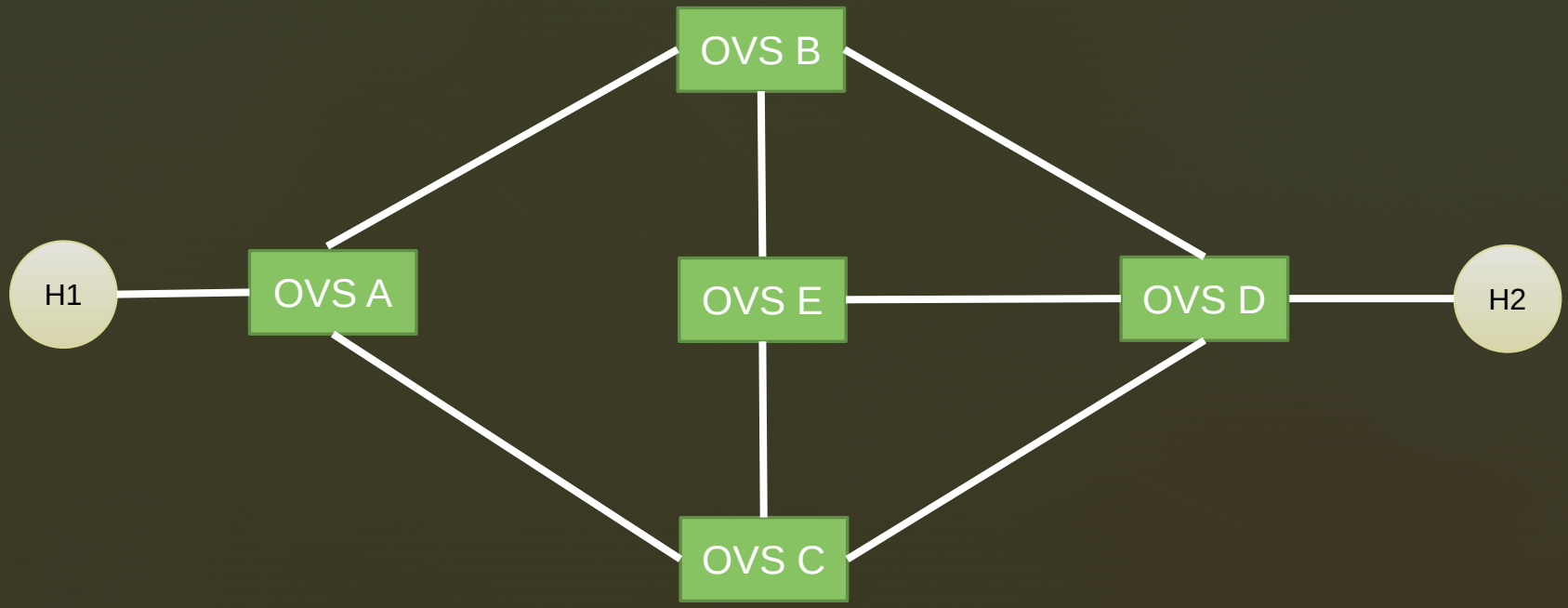
after pingall

Stop Wireshark and find Openflow control flows!!

2. Create custom topology and install rules

2. Create custom topology and install rules

- Simulate a software defined network with 2 hosts and 5 switches



Creating a custom topology

```
# Add hosts and switches
H1 = self.addHost( ‘H1' )
H2 = self.addHost( ‘H2' )
S1 = self.addSwitch( ‘S1' )
S2 = self.addSwitch( ‘S2' )

# Add links
self.addLink( H1 , S1 )
self.addLink( S1,  S2 )
self.addLink( S2,  H2 )
```

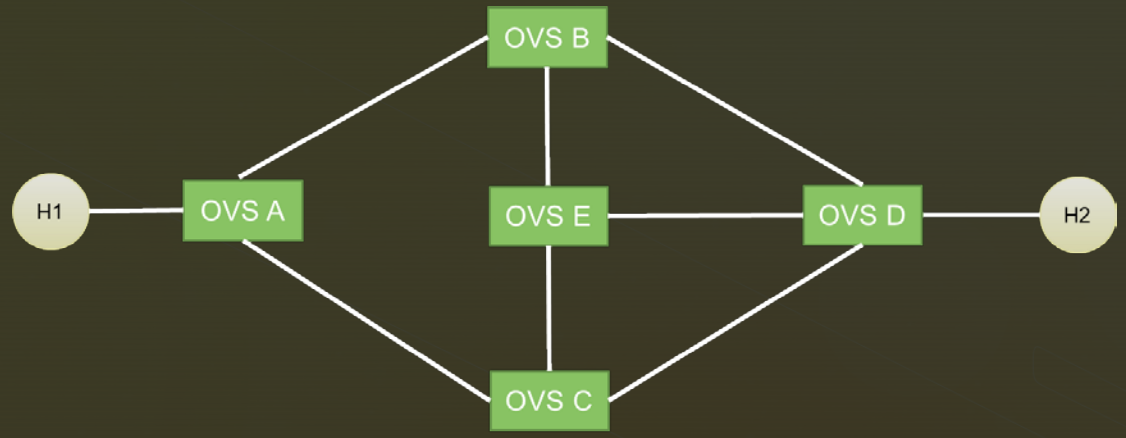
Refer to: https://github.com/mininet/mininet/blob/master/custom/topo-2sw-2host.py

Creating a custom topology

```
# Add hosts and switches
H1 = self.addHost( ‘H1' )
H2 = self.addHost( ‘H2' )
S1 = self.addSwitch( ‘S1' )
S2 = self.addSwitch( ‘S2' )

# Add links
self.addLink( H1 , S1 )
self.addLink( S1,  S2 )
self.addLink( S2,  H2 )
```

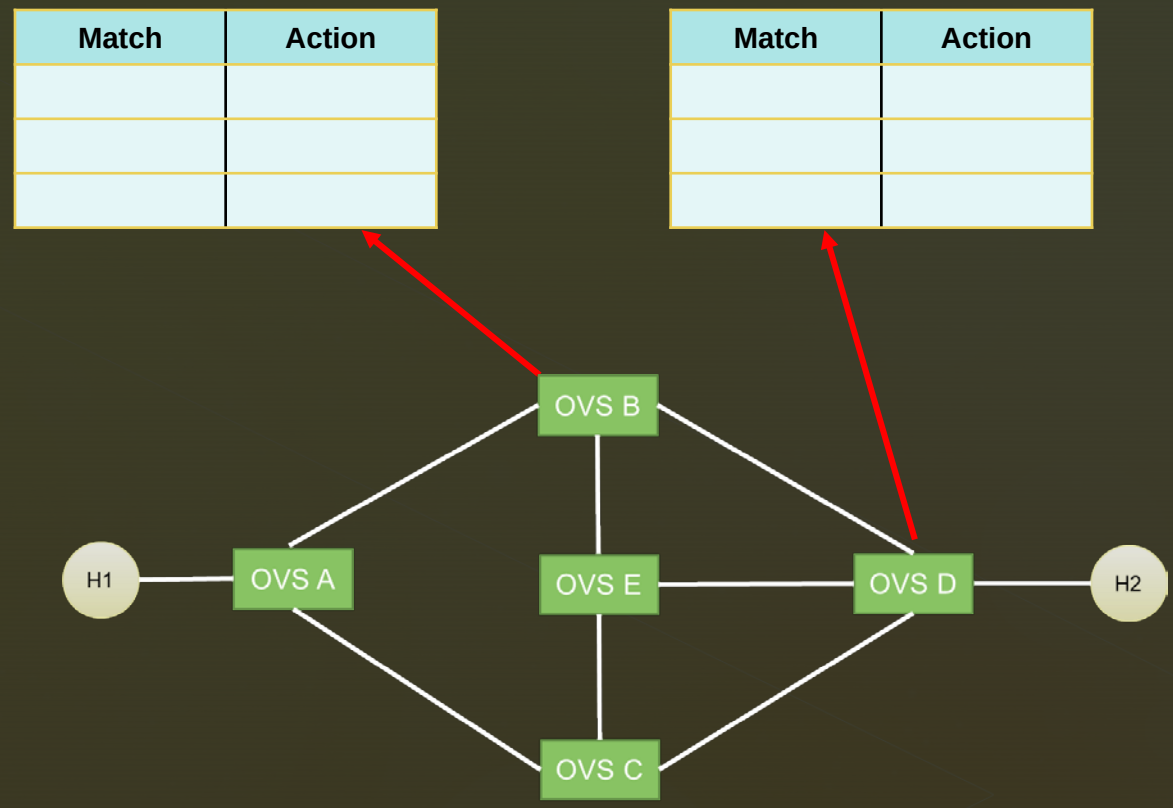
- sudo mn --custom *.py --topo mytopo --controller remote



Refer to: https://github.com/mininet/mininet/blob/master/custom/topo-2sw-2host.py

Manually installing rules on OVS nodes

Write the rule yourself
“Teach” the switches



(a switch is also called a “bridge”)

Manually installing rules on OVS nodes

- In the lab we need these commands (you may need to sudo):
  - ovs-ofctl show s1
  - ovs-ofctl dump-flows s1
  - ovs-ofctl add-flow s1 in_port=1,actions=output:2
  - ovs-ofctl add-flow s1 priority=500,in_port=1,dl_type=0x0800,nw_proto=6,actions=output:2

Match	Action

- Example:

```
mininet> sh ovs-ofctl dump-flows s2 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=582.062s, table=0, n_packets=1728516, n_bytes=2613493008, in_port=1 actions=output:2
 cookie=0x0, duration=582.036s, table=0, n_packets=52, n_bytes=15918, in_port=2 actions=output:1
mininet> sh ovs-ofctl dump-flows s3 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=584.106s, table=0, n_packets=345719, n_bytes=522722774, in_port=1 actions=output:2
 cookie=0x0, duration=584.082s, table=0, n_packets=0, n_bytes=0, in_port=2 actions=output:1
mininet> sh ovs-ofctl dump-flows s4 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=586.256s, table=0, n_packets=1037149, n_bytes=1568109186, in_port=1 actions=output:2
 cookie=0x0, duration=586.233s, table=0, n_packets=0, n_bytes=0, in_port=2 actions=output:1
mininet>
```

- **(Important!!!)** For more details refer to: https://man7.org/linux/man-pages/man7/ovs-fields.7.html
  http://manpages.ubuntu.com/manpages/xenial/man8/ovs-ofctl.8.html

Manually installing rules on OVS nodes

- In the lab we need these commands (you may need to sudo):
  - ovs-ofctl show s1
  - ovs-ofctl dump-flows s1
  - ovs-ofctl add-flow s1 in_port=1,actions=output:2
  - ovs-ofctl add-flow s1 priority=500,in_port=1,dl_type=0x0800,nw_proto=6,actions=output:2

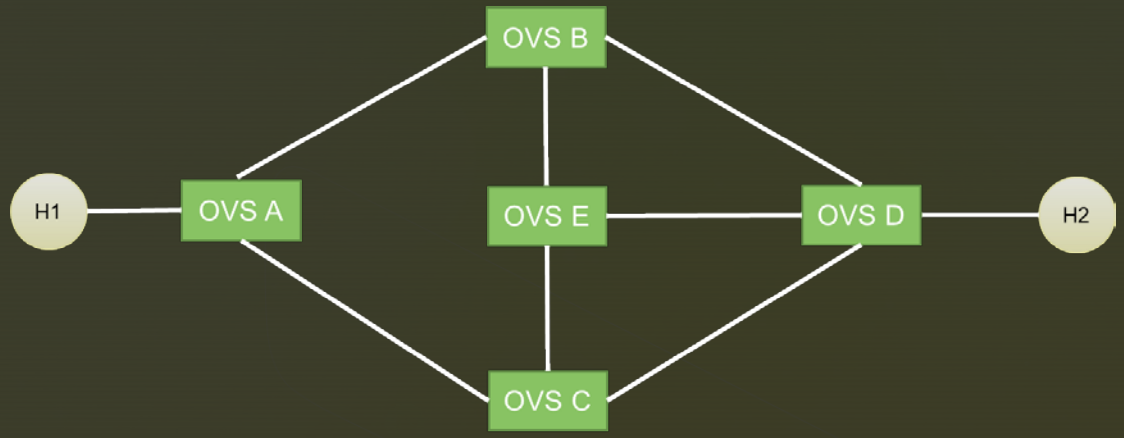
Match	Action
in_port=1	Output: 2

- Example:

```
mininet> sh ovs-ofctl dump-flows s2 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=582.062s, table=0, n_packets=1728516, n_bytes=2613493008, in_port=1 actions=output:2
 cookie=0x0, duration=582.036s, table=0, n_packets=52, n_bytes=15918, in_port=2 actions=output:1
mininet> sh ovs-ofctl dump-flows s3 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=584.106s, table=0, n_packets=345719, n_bytes=522722774, in_port=1 actions=output:2
 cookie=0x0, duration=584.082s, table=0, n_packets=0, n_bytes=0, in_port=2 actions=output:1
mininet> sh ovs-ofctl dump-flows s4 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=586.256s, table=0, n_packets=1037149, n_bytes=1568109186, in_port=1 actions=output:2
 cookie=0x0, duration=586.233s, table=0, n_packets=0, n_bytes=0, in_port=2 actions=output:1
mininet>
```

- **(Important!!!)** For more details refer to: https://man7.org/linux/man-pages/man7/ovs-fields.7.html
  http://manpages.ubuntu.com/manpages/xenial/man8/ovs-ofctl.8.html

2. Create custom topology and install rules

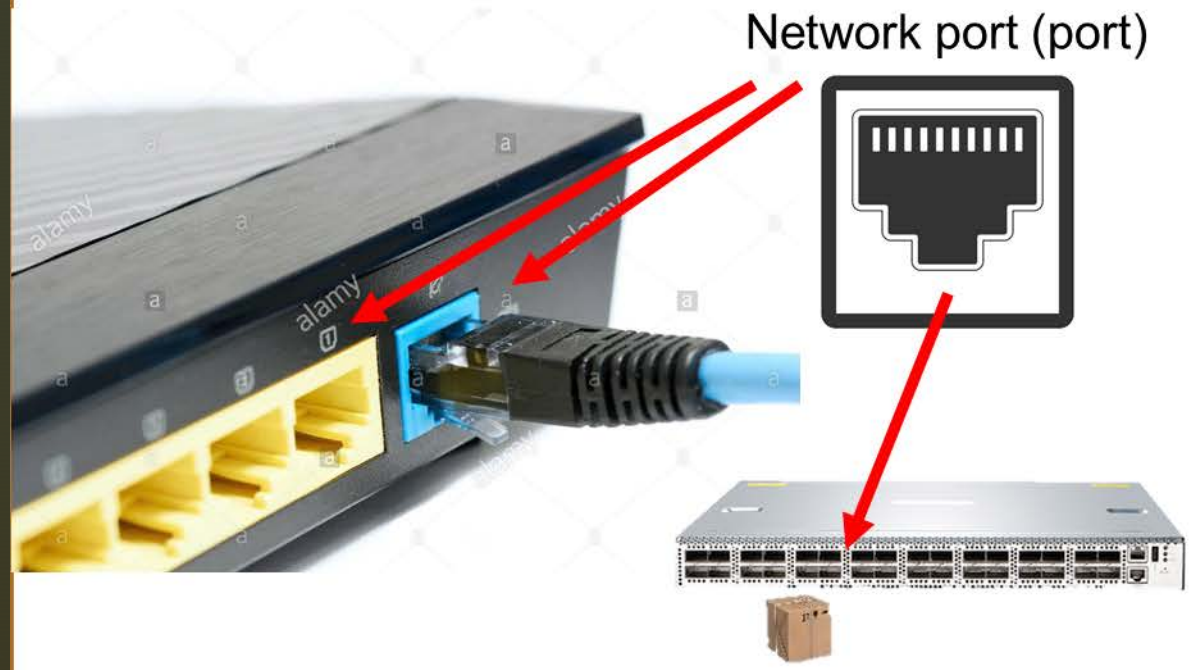


- Implement flow-based routing instead of destination-based routing
  - Traffic from H1 ➔ H2
    - **HTTP traffic with d_port=80**, follow path: A-B-D
    - other traffic, follow path: A-C-E-D
  - Traffic from H2 ➔ H1
    - HTTP traffic with s_port=80, follow path: D-C-A
    - other traffic, follow path: D-B-E-C-A
- Generate traffic and verify your result with Wireshark

Lab2 Overview (part 1)

OVS B

Port vs TCP port



TCP port

Labels for application

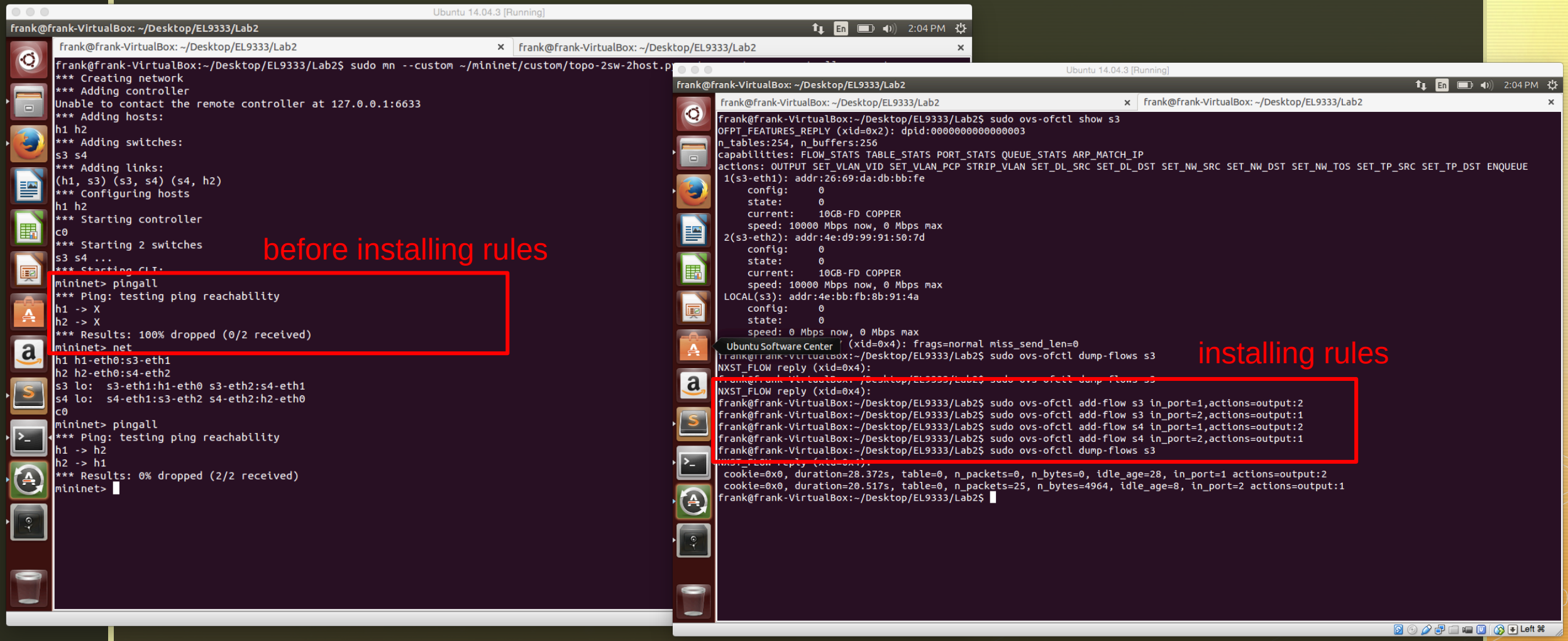
Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
25	TCP	SMTP
53	UDP, TCP	DNS
80	TCP	HTTP (WWW)
110	TCP	POP3
443	TCP	SSL

- Generate traffic and verify your result with Wireshark

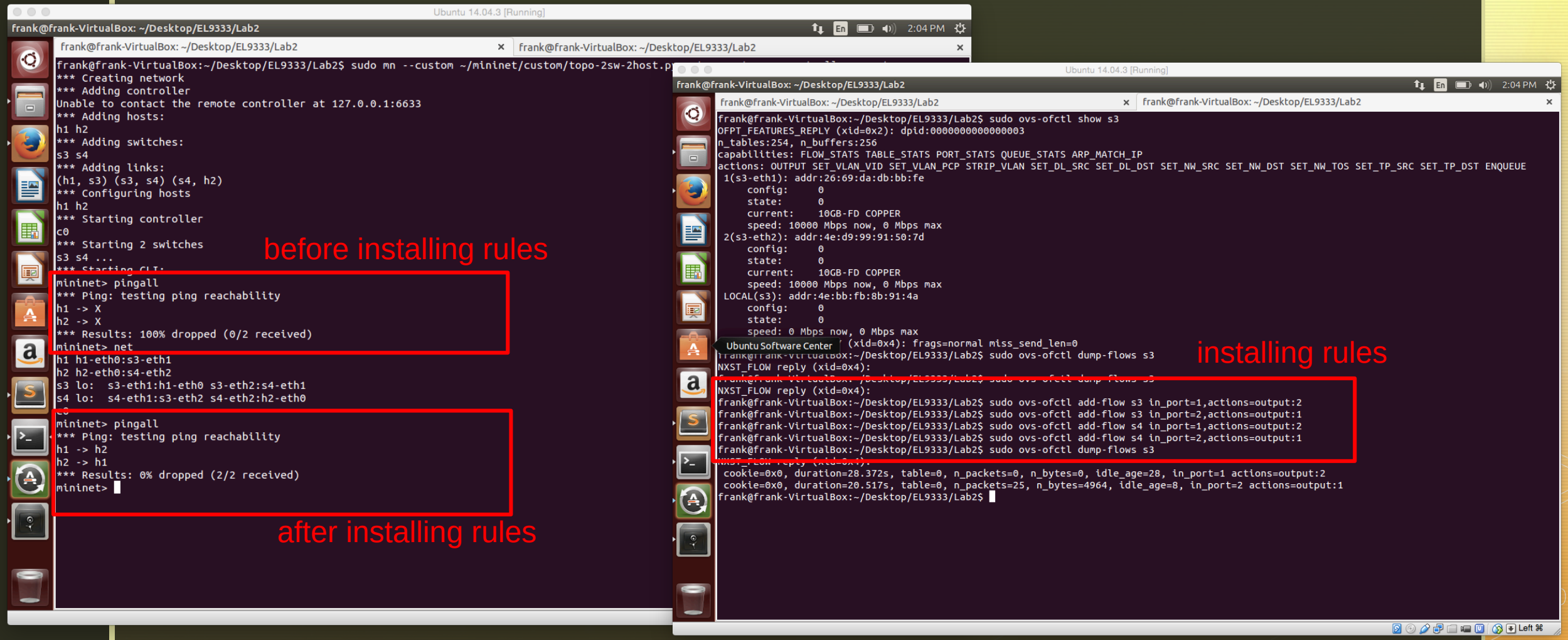
2. Running Mininet with Custom Topology

- To launch Mininet **without** default controller:
- sudo mn --custom *.py --topo mytopo **--controller remote**

2. Running Mininet with Custom Topology



2. Running Mininet with Custom Topology

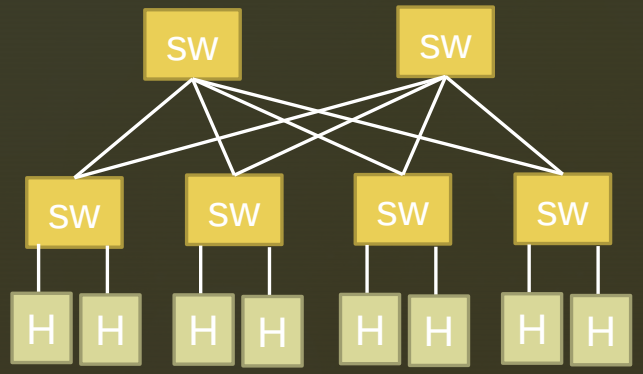


Useful tips

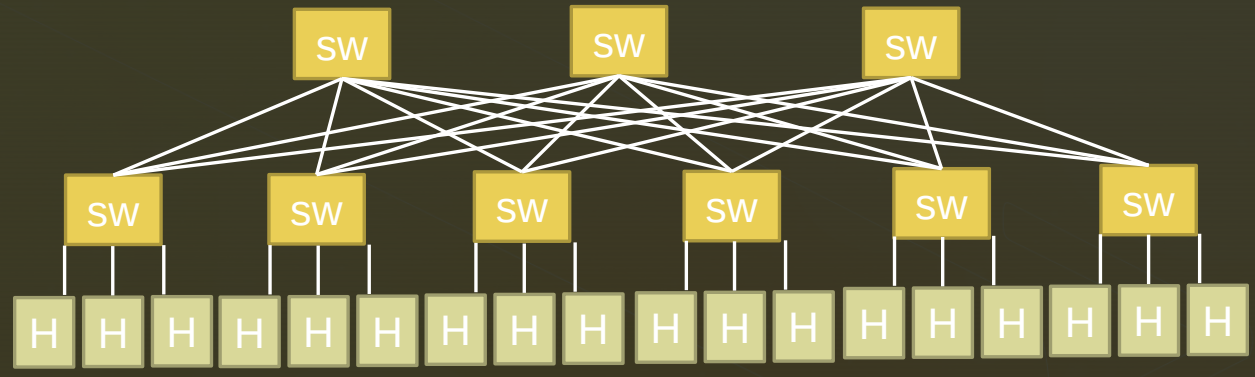
- To clean up the Mininet
  - $ mn –c
- Open terminal on one host in Mininet
  - mininet> xterm h1
- Run command on host
  - mininet> h1 <commands>
- **To generate traffic**
  - **iperf/hping (Refer to lab 1) for HTTP packets**
  - **ping to generate ICMP packets**

Lab2 Overview (part 2)

- Given switches with N ports, create a 2-stage **fat-tree topology** with hosts in Mininet
  - Make N a variable in your topology file/program
- Ex:



N=4



N=6