SDN with OpenFlow

OpenFlow

https://opennetworking.org/software-defined-standards/specifications/

Software Defined Networks A Comprehensive Approach – Chapter 5

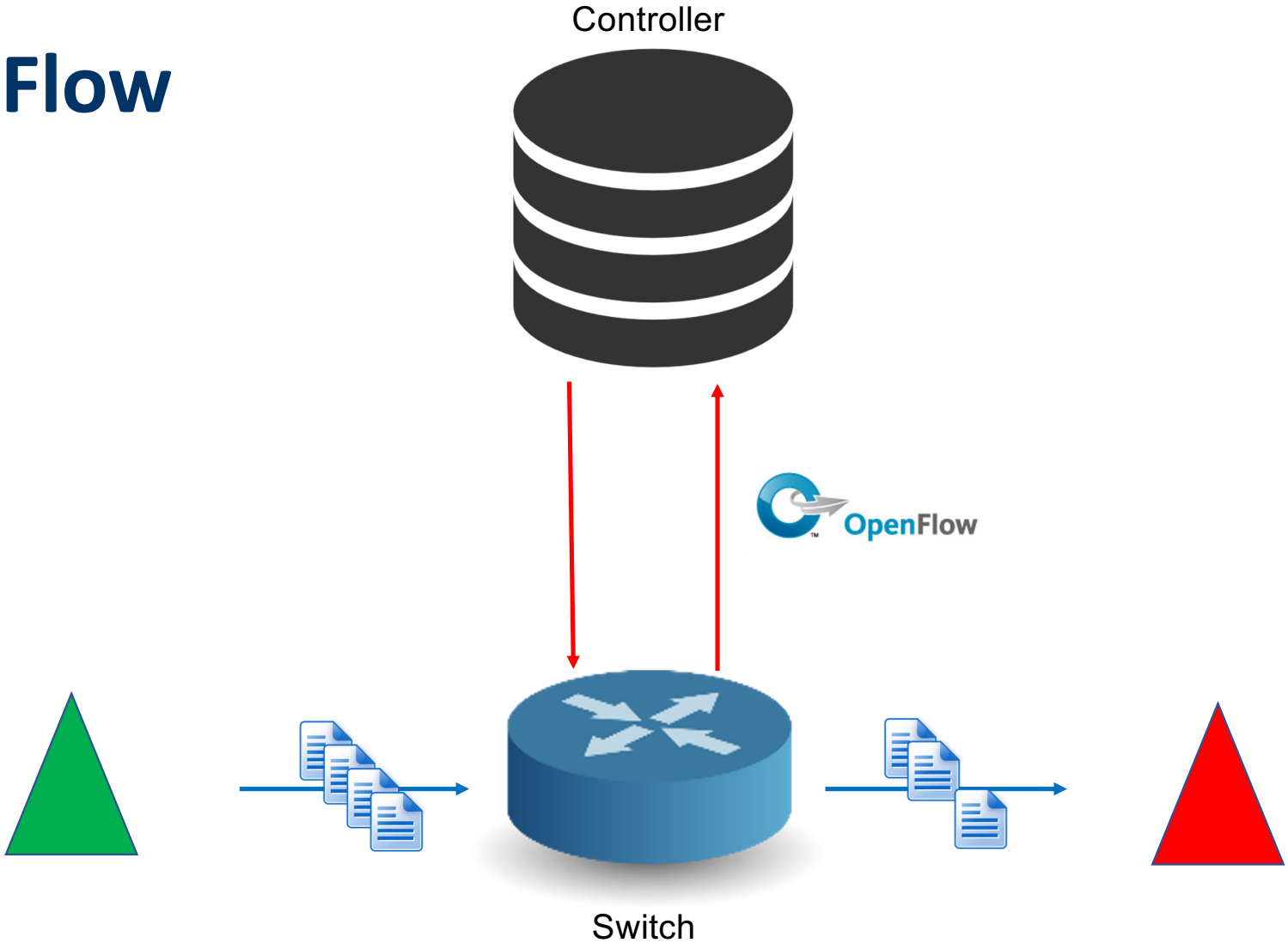
Floodlight

Docs: floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview

VM: floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM

Antonio Virdis
Assistant Professor@ University of Pisa
antonio.virdis@unipi.it

Open Flow



Rules Examples

- ***Cross-layer rules*** for packet classification
- Different ***functionalities*** can be implemented:
  - *Switching*
  - *Routing*
  - *Firewall*

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

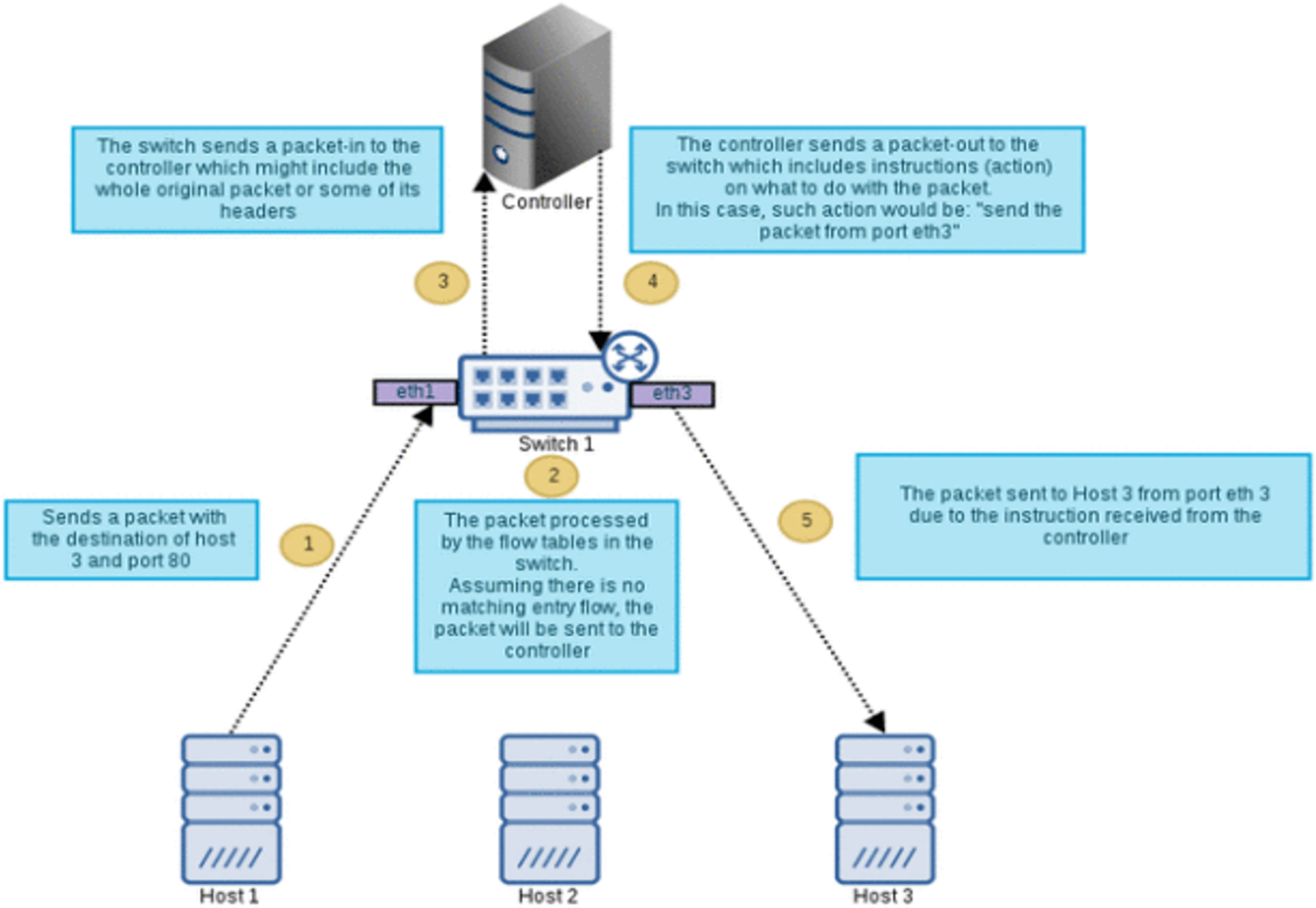
Firewall

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

Routing

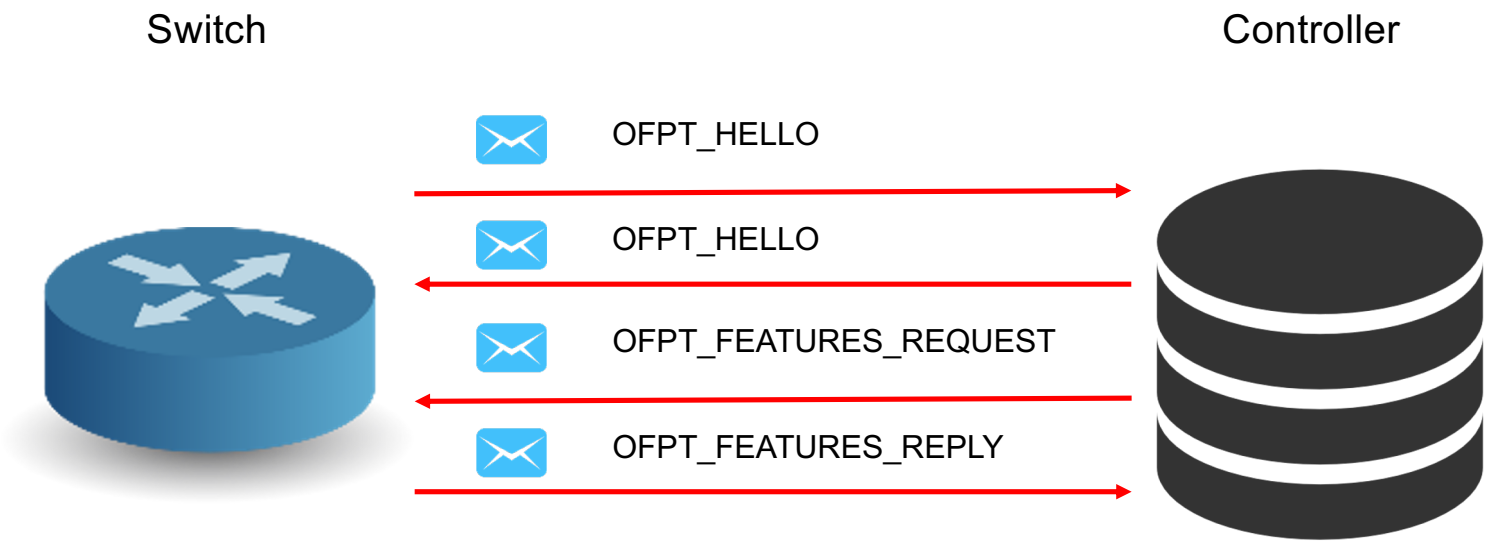
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

Communication Example



OF Messages - Startup

- At startup a set of startup messages is sent to allow the controller to discover the capabilities of the switch

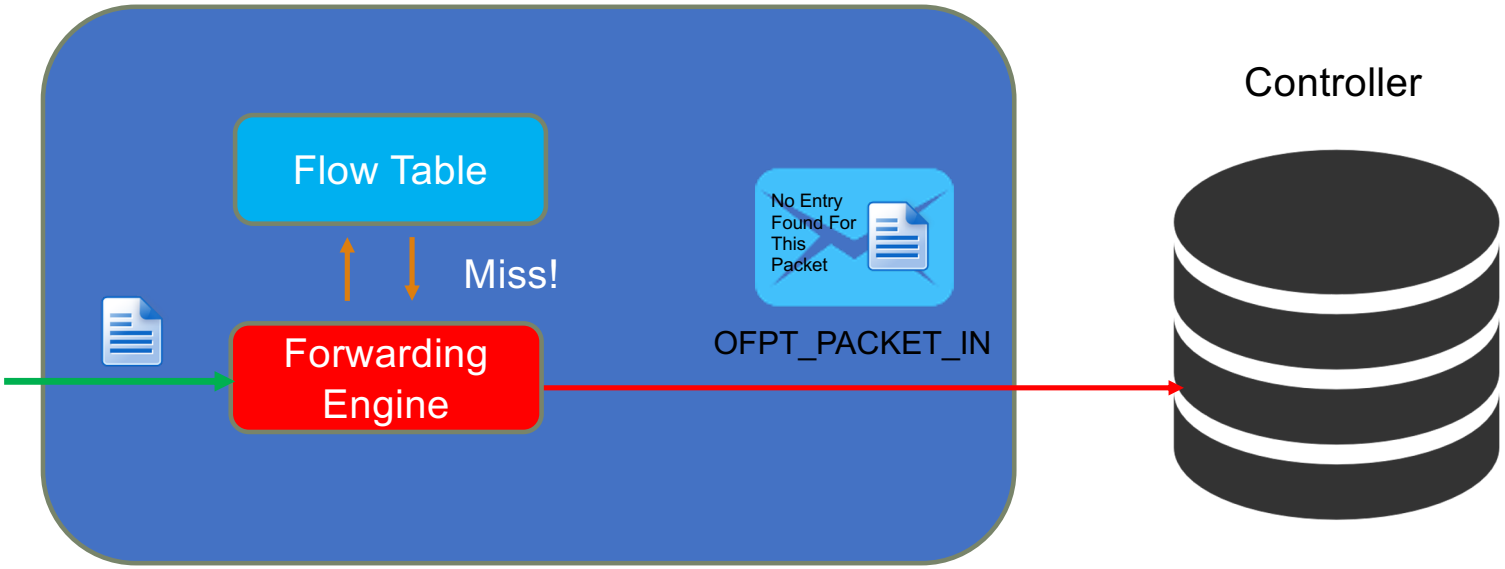


List of capabilities

```
/* Capabilities supported by the datapath. */
enum ofp_capabilities {
    OFPC_FLOW_STATS     = 1 << 0,  /* Flow statistics. */
    OFPC_TABLE_STATS    = 1 << 1,  /* Table statistics. */
    OFPC_PORT_STATS     = 1 << 2,  /* Port statistics. */
    OFPC_GROUP_STATS    = 1 << 3,  /* Group statistics. */
    OFPC_IP_REASM       = 1 << 5,  /* Can reassemble IP fragments. */
    OFPC_QUEUE_STATS    = 1 << 6,  /* Queue statistics. */
    OFPC_PORT_BLOCKED   = 1 << 8,  /* Switch will block looping ports. */
    OFPC_BUNDLES        = 1 << 9,  /* Switch supports bundles. */
    OFPC_FLOW_MONITORING = 1 << 10,  /* Switch supports flow monitoring. */
};
```

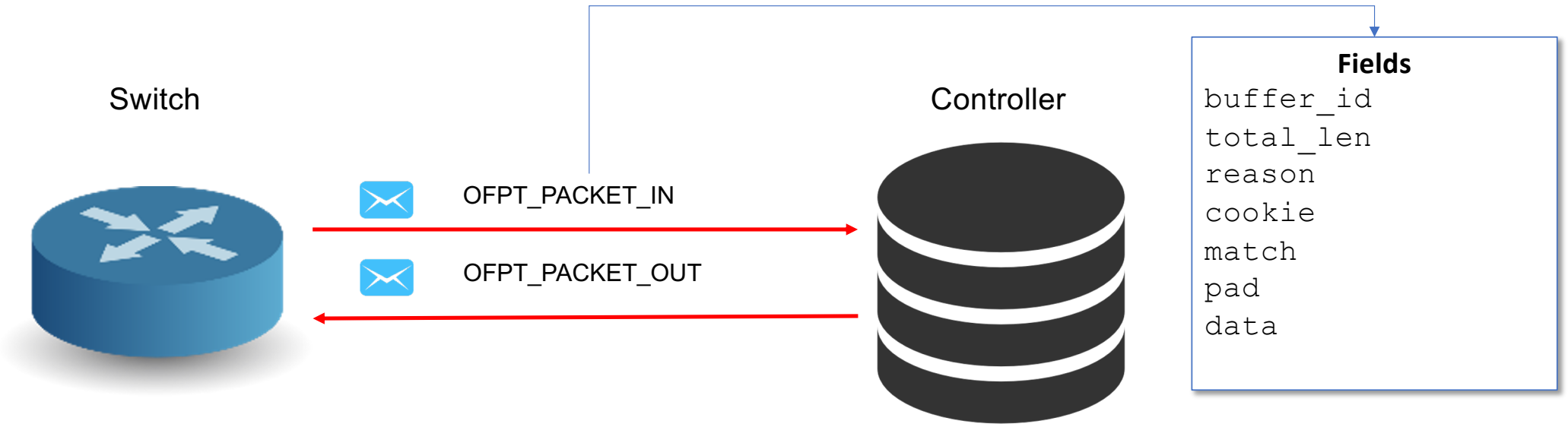
Switch Operations

- For each received packet the Flow Table is looked up
- If a match is ***found*** the action is executed, otherwise the ***packet (or a reference) is forwarded to the controller encapsulated into a Packet-In***



OF Messages – Normal Operations

- During normal operations switch and controller interacts with IN and OUT packets

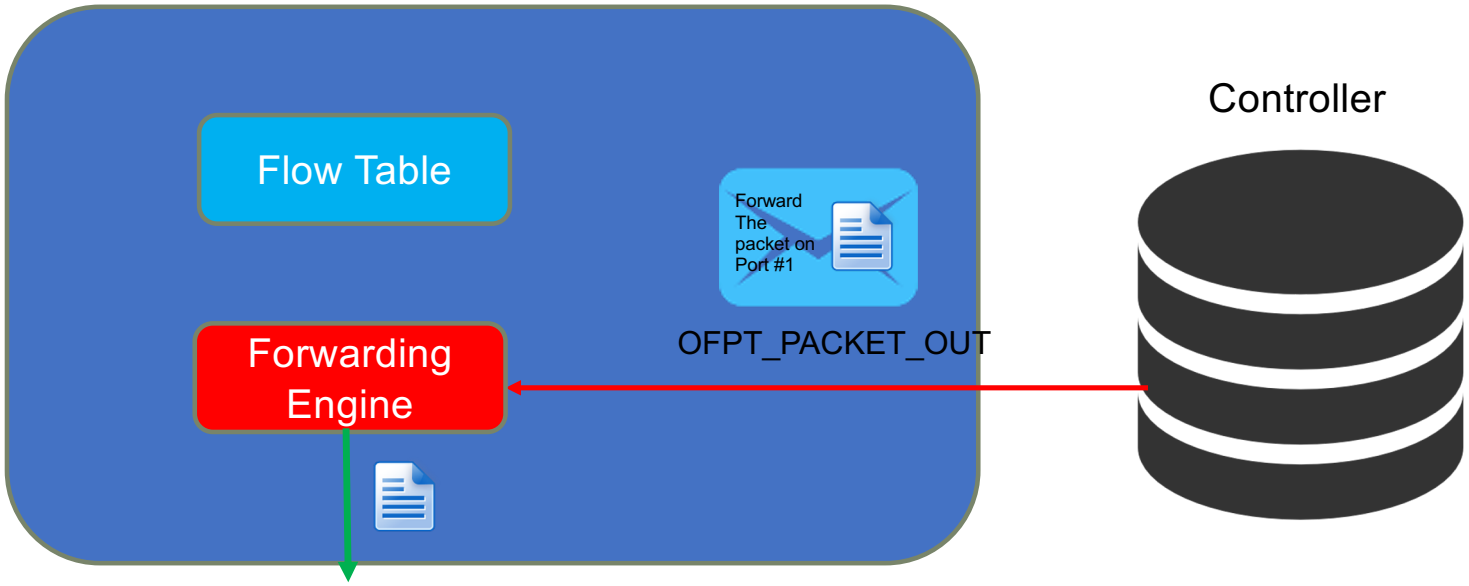


reasons for OFPT_IN

```
/* Why is this packet being sent to the controller? */
enum ofp_packet_in_reason {
    OFPR_NO_MATCH    = 0,   /* No matching flow (table-miss flow entry). */
    OFPR_ACTION      = 1,   /* Action explicitly output to controller. */
    OFPR_INVALID_TTL = 2,   /* Packet has invalid TTL */
};
```

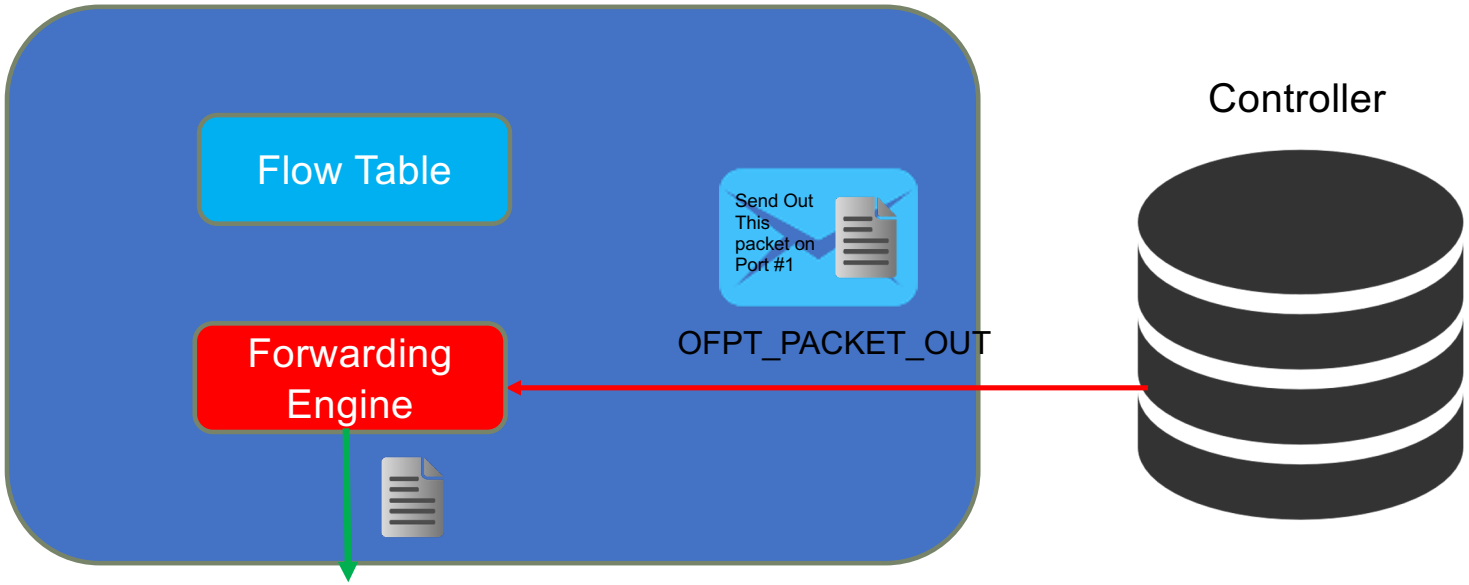
Switch Operations

- The controller can reply with a Packet-Out specifying the action to be performed (e.g. forward the packet on port #1) and the packet or the reference
- It will be executed only once (no modifications to the Flow Table)



Switch Operations

- The controller can reply with a Packet-Out specifying a new packet to be sent out
- It will be executed only once (no modifications to the Flow Table)



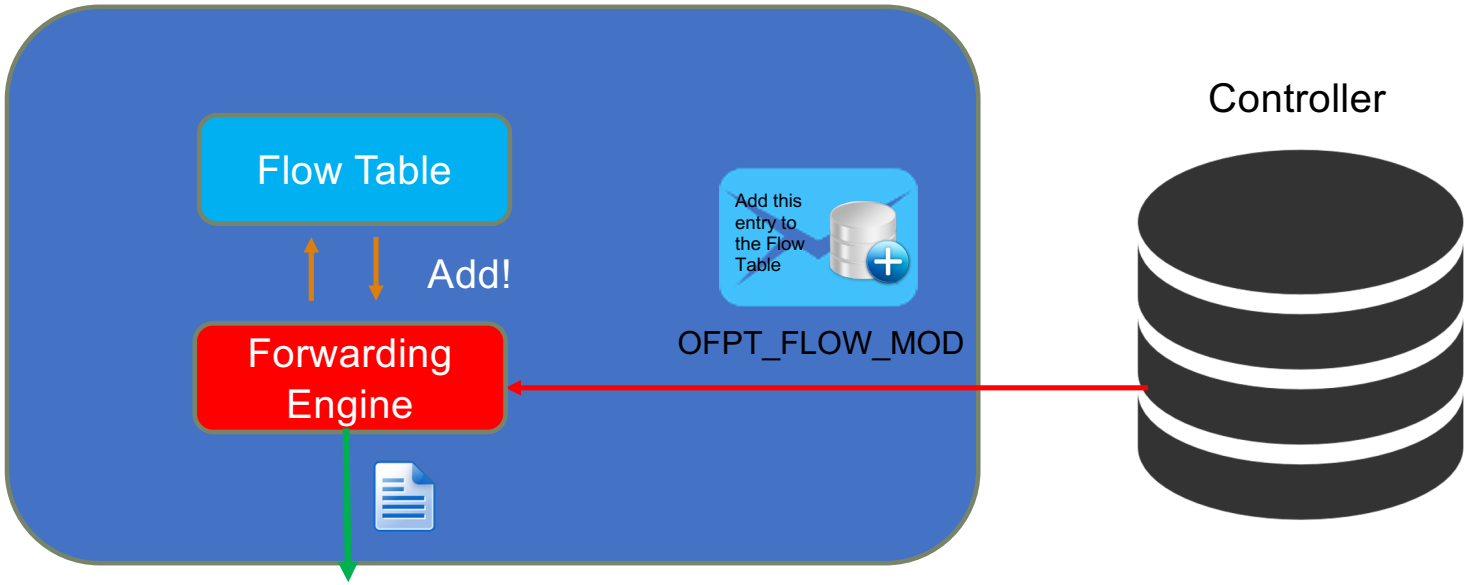
Types of actions

```
enum ofp_action_type {
    OFPAT_OUTPUT       = 0,  /* Output to switch port. */
    OFPAT_COPY_TTL_OUT = 11, /* Copy TTL "outwards" -- from next-to-outermost
                                to outermost */
    OFPAT_COPY_TTL_IN  = 12, /* Copy TTL "inwards" -- from outermost to
                                next-to-outermost */
    OFPAT_SET_MPLS_TTL = 15, /* MPLS TTL */
    OFPAT_DEC_MPLS_TTL = 16, /* Decrement MPLS TTL */

    OFPAT_PUSH_VLAN    = 17, /* Push a new VLAN tag */
    OFPAT_POP_VLAN     = 18, /* Pop the outer VLAN tag */
    OFPAT_PUSH_MPLS    = 19, /* Push a new MPLS tag */
    OFPAT_POP_MPLS     = 20, /* Pop the outer MPLS tag */
    OFPAT_SET_QUEUE    = 21, /* Set queue id when outputting to a port */
    OFPAT_GROUP        = 22, /* Apply group. */
    OFPAT_SET_NW_TTL   = 23, /* IP TTL. */
    OFPAT_DEC_NW_TTL   = 24, /* Decrement IP TTL. */
    OFPAT_SET_FIELD    = 25, /* Set a header field using OXM TLV format. */
    OFPAT_PUSH_PBB     = 26, /* Push a new PBB service tag (I-TAG) */
    OFPAT_POP_PBB      = 27, /* Pop the outer PBB service tag (I-TAG) */
    OFPAT_EXPERIMENTER = 0xffff
};
```

Switch Operations

- The controller can reply with a ***Flow-Mod*** message that instructs the switch to add a new entry to its table
- The new entry will instruct the switch to perform a certain operation without contacting the controller
- The operation associated with the new entry is then executed

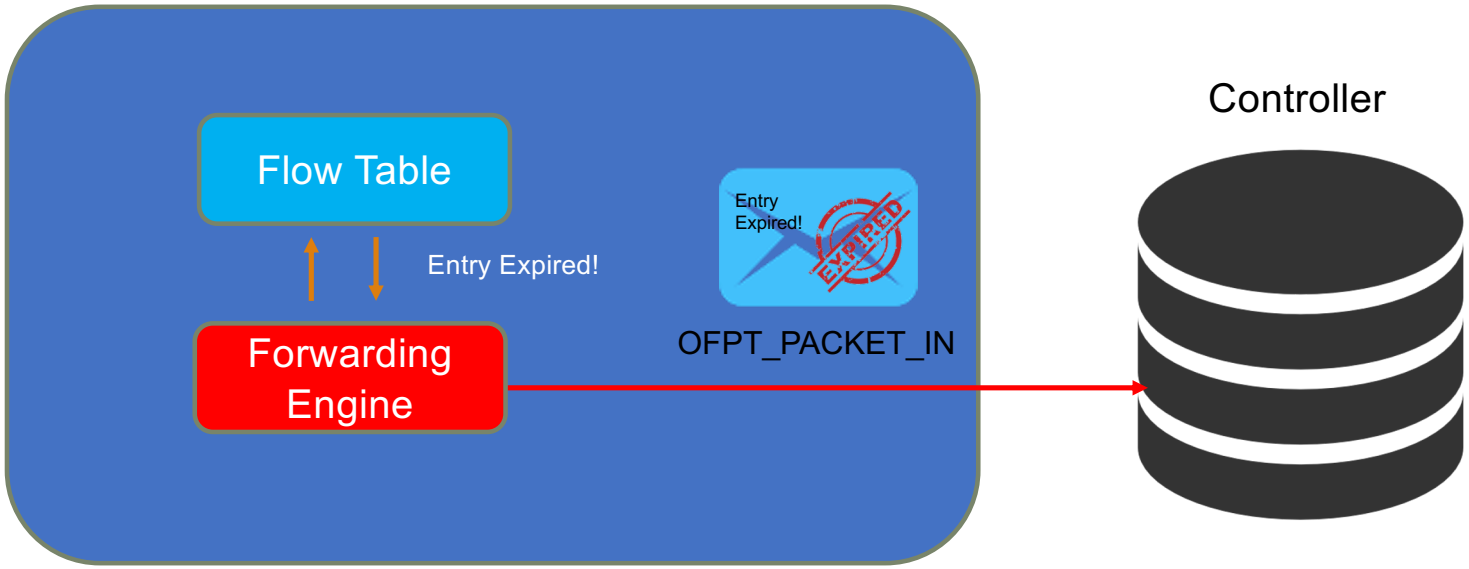


Flow Mod Commands

```
enum ofp_flow_mod_command {
    OFPFC_ADD           = 0, /* New flow. */
    OFPFC_MODIFY        = 1, /* Modify all matching flows. */
    OFPFC_MODIFY_STRICT = 2, /* Modify entry strictly matching wildcards and
                                priority. */
    OFPFC_DELETE        = 3, /* Delete all matching flows. */
    OFPFC_DELETE_STRICT = 4, /* Delete entry strictly matching wildcards and
                                priority. */
};
```

Entry Management

- Entries in the flow table expire
- As the entry is expired a Packet-In is sent to the Controller containing a ***Flow-Expired*** message
- Entries expire after a hard timeout (always) or after an idle timeout (if packets matching with the entry are not received)



Labs

mininet + Floodlight

https://drive.google.com/file/d/1rjM-YTrzIzE7034-ejy-spn8f_7SzG6A/view?usp=sharing

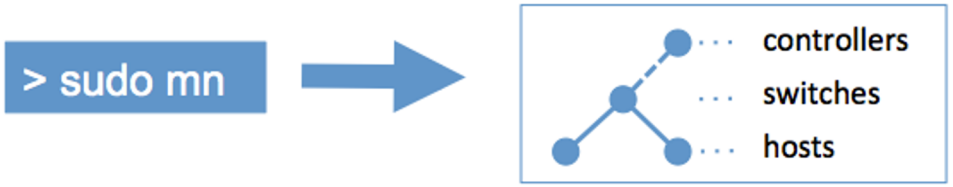
Mininet

Antonio Virdis
Assistant Professor@ University of Pisa
antonio.virdis@unipi.it

Mininet

*"Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine"*

- Mininet is a virtual network emulator for testing of SDN deployments
- It allows in one program to emulate a network composed of OpenFlow switches and hosts which can generate traffic
- The network of OpenFlow switches can be connected to a real controller



Mininet

- Launch the simulator:

```
sudo mn --topo single,3
--ipbase=10.0.0.0
```

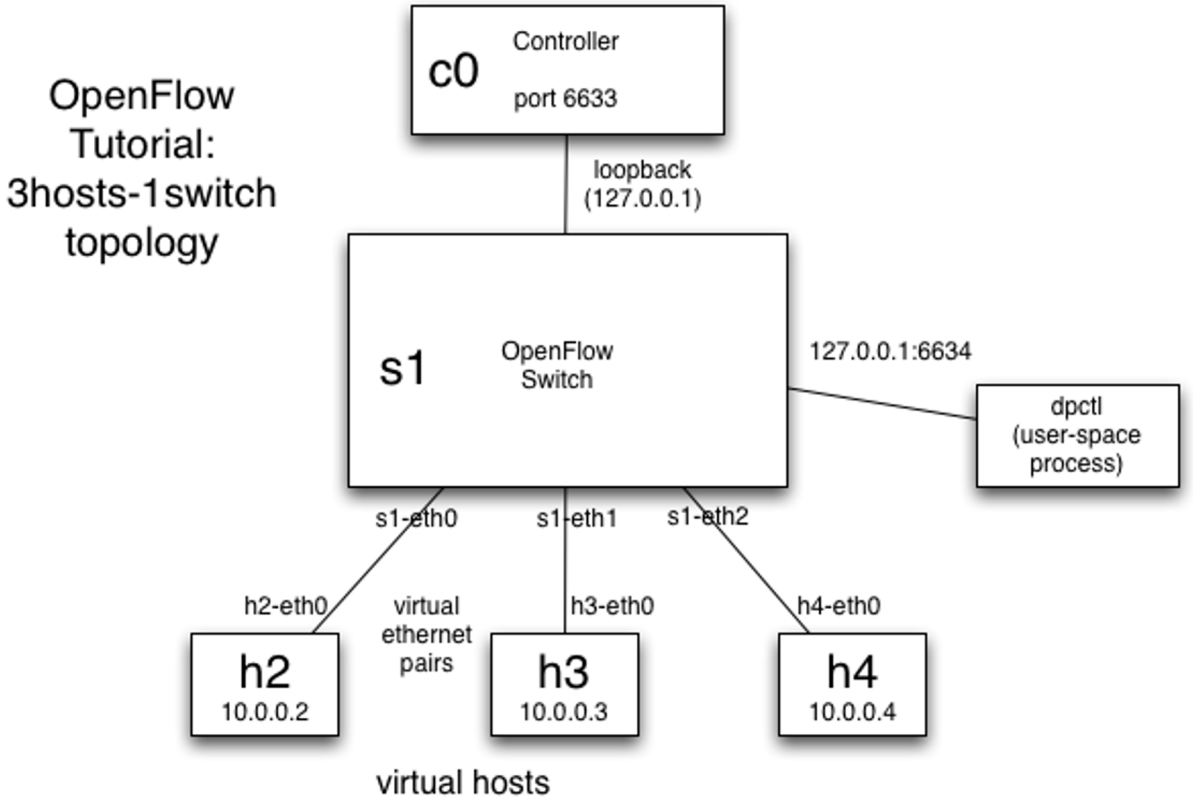
Type of the topology, with one single switch and three hosts

IP network subnet for simulated hosts

Without a controller the simulated switches behave as normal swtiches bridging the different networks

Simulated Architecture

- The basic topology has the following architecture



Mininet Basics

- Run a program on a host
  - `host_name command`
    - `h1 ping 10.0.0.2`
    - `h1 ifconfig -a`
    - `h1 ifconfig h1-eth0 10.0.0.1`
- Open a separate terminal on a host
  - `xterm host_name`
    - `xterm h1`
    - `From the terminal for example you can run wireshark`

Floodlight

Antonio Virdis
Assistant Professor@ University of Pisa
antonio.virdis@unipi.it

Floodlight

- Floodlight is a java framework that allows the implementation of OpenFlow controllers
- It not only provides an implementation of the OpenFlow protocol but also an implementation of some basic operations implemented by controllers

External Controller

How to connect mininet to an external controller

- Launch the simulator:

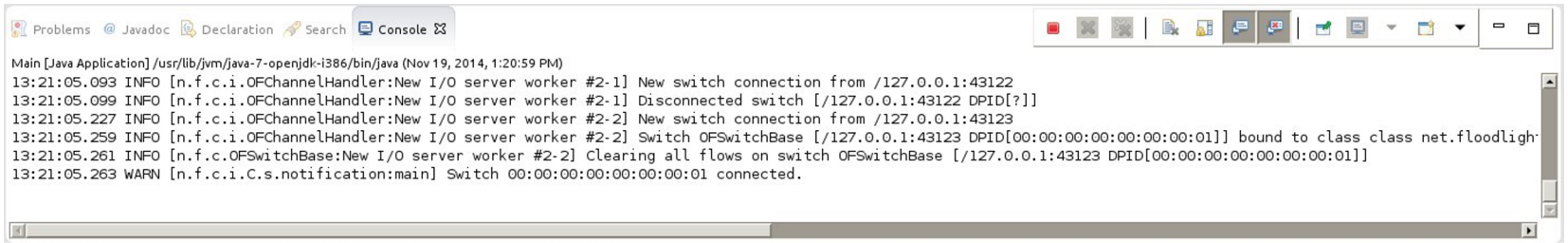
```
sudo mn --topo single,3
--mac --switch ovsk
--controller remote,
ip=127.0.0.1,port=6653,protocols=OpenFlow13
--ipbase=10.0.0.0
```

The version of the OpenFlow protocol the switch will use to communicate with the controller, 1.3 in this case

IP and port of the real OpenFlow controller (in this case it runs on localhost listening on port 6653

Controller

- If you launch the controller now you obtain the following message from the emulator:
  - `Unable to contact the remote controller at 127.0.0.1:6633`
- Compile and Execute the controller using the "*ant run*" command
- Or open eclipse to run floodlight directly into eclipse
- Wireshark can be executed on the loopback interface to capture OpenFlow messages between controller and switch

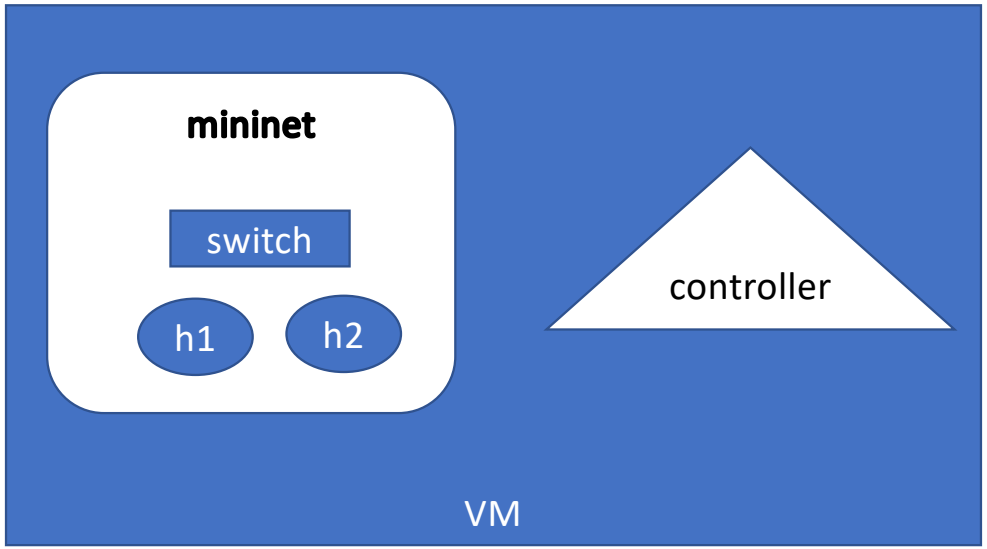


Floodlight

- Floodlight has a modular structure, each module implements one functionality
- Inbound packets are processed in cascade by each module, each module can interrupt the pipeline
- The modules included in the pipeline and their order are specified inside the file
  - `src/main/resources/floodlightdefault.properties`

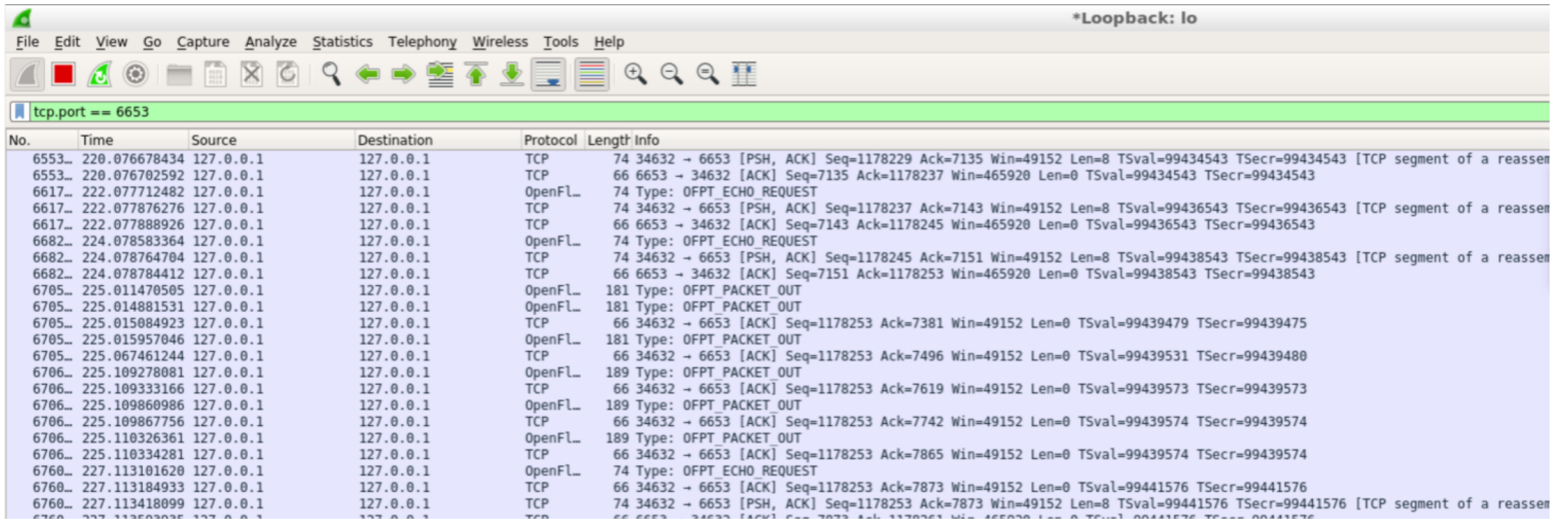
```
floodlight.modules=\
net.floodlightcontroller.jython.JythonDebugInterface,\
net.floodlightcontroller.counter.CounterStore,\
net.floodlightcontroller.storage.memory.MemoryStorageSource,\
net.floodlightcontroller.core.internal.FloodlightProvider,\
net.floodlightcontroller.threadpool.ThreadPool,\
net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl,\
net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier,\
net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher,\
net.floodlightcontroller.firewall.Firewall,\
net.floodlightcontroller.hub.Hub,\
net.floodlightcontroller.forwarding.Forwarding,\
net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager,\
net.floodlightcontroller.topology.TopologyManager,\
net.floodlightcontroller.flowcache.FlowReconcileManager,\
net.floodlightcontroller.debugcounter.DebugCounter,\
```

Lab Configuration



Intercept OpenFlow packets

- You can use wireshark to intercept OpenFlow packets
- Open Wireshark and capture the packets on the Loopback interface
- Set the following filter to get only OpenFlow packets: *tcp.port == 6653*



Check OVS

```
ip -d link
```

```
root@floodlight:~# ip -d link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 promiscuity 0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:44:8b:b0 brd ff:ff:ff:ff:ff:ff promiscuity 0
5: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
    link/ether 1a:14:8f:8a:4b:61 brd ff:ff:ff:ff:ff:ff promiscuity 1
25: s1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 1a:f8:f3:37:b7:5b brd ff:ff:ff:ff:ff:ff promiscuity 1
    veth
26: s1-eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP mode DEFAULT group default qlen 1000
    link/ether 4a:60:5c:68:e5:58 brd ff:ff:ff:ff:ff:ff promiscuity 1
    veth
27: s1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default
    link/ether ce:42:37:29:72:4c brd ff:ff:ff:ff:ff:ff promiscuity 1
root@floodlight:~#
```

```
ovs-vsctl show
```

```
root@floodlight:~# ovs-vsctl show
9e8c9fbd-81b6-4b4d-8743-c2b60a84c826
    Bridge "s1"
        Controller "tcp:192.168.57.3:6653"
            is_connected: true
        Controller "ptcp:6634"
        fail_mode: secure
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.3.90"
```

Test: Ping h1->h2

```
sudo ovs-ofctl dump-flows s1 -O OpenFlow13
```

```
root@floodlight:~# sudo ovs-ofctl dump-flows s1 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=425.401s, table=0, n_packets=13, n_bytes=1038, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=425.401s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=425.401s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=425.401s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=425.401s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
root@floodlight:~#
```

```
h1 ping h2
```

```
root@floodlight:~# sudo ovs-ofctl dump-flows s1 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x20000000000000, duration=9.232s, table=0, n_packets=9, n_bytes=882, idle_timeout=5, priority=1,ip,in_port=1,dl_src=02:44:6e:17:d8:ef
10.0.0.2 actions=output:2
 cookie=0x20000000000000, duration=9.230s, table=0, n_packets=9, n_bytes=882, idle_timeout=5, priority=1,ip,in_port=2,dl_src=fe:c4:8a:1a:ad:b3
10.0.0.1 actions=output:1
 cookie=0x20000000000000, duration=0.224s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port=2,dl_src=fe:c4:8a:1a:ad:b3,
 cookie=0x20000000000000, duration=0.219s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port=1,dl_src=02:44:6e:17:d8:ef,
 cookie=0x0, duration=537.653s, table=0, n_packets=26, n_bytes=1808, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=537.653s, table=1, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=537.653s, table=2, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=537.653s, table=3, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x0, duration=537.653s, table=4, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
```