Lab 1 - Part 2

- Mininet and Flow tables

# Objective

Now that you understand the basic functionalities of Mininet, we can start looking into the core of the SDN and OpenFlow operations: installing rules into the switching table.

This exercise puts together the elements of the protocol to transmit data from one host to another through an openflow switch using TCP/IP

# What you will learn

Manipulate Openflow switch (OVS) tables using dpctl, ovs-ofctl

Construct table rules for arp, icmp, ip, ethernet

# Prerequisites

Network commands : tcpdump

Protocol : ethernet addressing, ethernet protocol id

# Approach

* Create simple topology of 2 hosts that communicate with each other through an openflow switch.
* Delete all rules from the switch
* Start a ping from host 1 to host 2
* Inspect how traffic is trying to make its way from host 1 to host 2
* Install first set of (partial) rules to allow h1 and h2 to discover where each other
* Install second set of rules to allow h1 to ping h2.
* Discover alternate/simple approaches to achieving the same result (using a self-learning switch)

# Topology

The topology is composed of 2 hosts H1 and H2,, an openflow switch S1 and a controller C1.

Host H1

Switch S1

Host H2

Controller C1

H1-eth0

H2-eth0

S1-eth0

S1-eth1

Control Plane

Data Plane

Openflow

channel

# Steps

1. Start mininet with single OVS switch and dummy controller. Simplify mac addresses

**sudo -E mn --switch ovsk --controller remote --mac**

Notes:

* The - E option needed pass mininet bash (DISPLAY) variables to root
* The “–switch ovsk” instruction indicates the use of a specific Ethernet virtual switch component called OVS, instead of the basic Linux one.
* The “--controller remote” instruction indicates that there is no default controller in the topology, which create switching rules. Mininet expects rules to be installed by an external controller (or input from a command line in or case)

1. Find out network details using dump and net from mininet prompt

**mininet> net**

*h1 h1-eth0:s1-eth1* → indicates h1 interface eth0 is connected to switch s1 interface eth1

*h2 h2-eth0:s1-eth2* → indicates h2 interface eth0 is connected to switch s1 interface eth2

*s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0* → indicates the connections above but seen from the

switch s1 perspective

*c0* → indicates the controller

1. Start 2 xterms for each of the hosts

**mininet> xterm h1 h1 h2 h2**

On node h2 run ifconfig to find the name of your interfaces

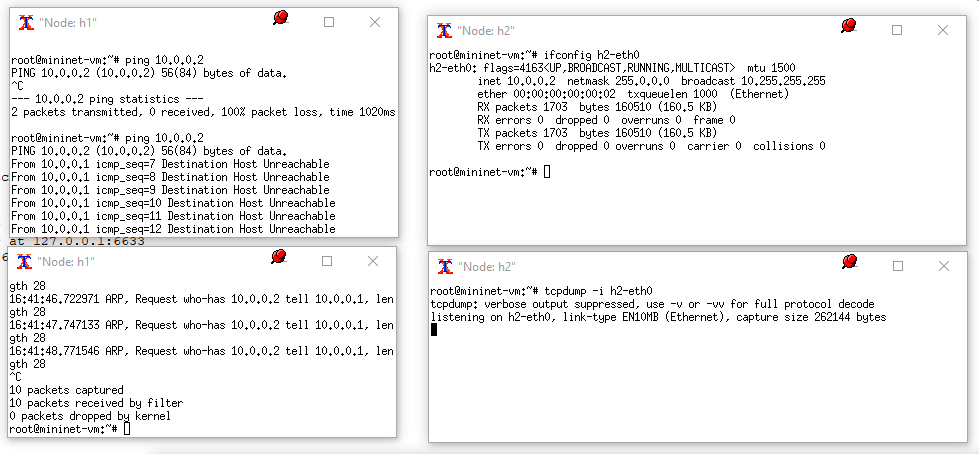
Then run the command

**root@mininet-vm:~# tcpdump -i h2-eth0**

This will show all packets crossing the node 2 interface

Repeat the same for the h1 node

Try to ping h2 from h1. You should see a result similar to the figure below on the left. The right side shows that the packets do not reach h2.



1. On a linux prompt, clear the open flow tables

**mininet@mininet-vm:~$ sudo ovs-ofctl del-flows s1**

Then you can verify that S1 has no active flow rules, running:

**mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s1**

1. Start a ping from h1 (10.0.0.1) to h2 (10.0.0.2) and verify that ping \*fails\* using tcpdump on separate screens on each host.

*What protocol fragment do you see on h1?*

*What is the function of Arp ?*

*What don’t you see?*

1. Now you will be able to add flows manually into the switch. This is a key feature for SDN. Now we will do it manually but in future labs you will be designing software scripts to automate the process.

Now you can add flow rules to connect h1 to h2 in the switch.

**sudo ovs-ofctl add-flow s1 dl\_src=00:00:00:00:00:02,dl\_type=0x806,actions=output:\"s1-eth1\"**

**sudo ovs-ofctl add-flow s1 dl\_src=00:00:00:00:00:01,dl\_type=0x806,actions=output:\"s1-eth2\"**

Take a moment to analyse the command.

*What is the purpose of this rule?*

Notice that in the type dl\_type we are including 0x806, which is the ARP protocol, so these flows will only work for the ARP protocol.

Now you can run dump-flows again and you will see information on active flow rules:

*cookie=0x0, duration=24.529s, table=0, n\_packets=0, n\_bytes=0, arp,dl\_src=00:00:00:00:00:02 actions=output:"s1-eth1"*

*cookie=0x0, duration=2.493s, table=0, n\_packets=0, n\_bytes=0, arp,dl\_src=00:00:00:00:00:01 actions=output:"s1-eth2"*

Again, take a moment to analyse the response and understand the information provided from the flow rules

1. Verify that arp table is populated on each host

On node h1 and on node h2: **root@mininet-vm:~# arp -a**

Now you can try the ping again and see what happens

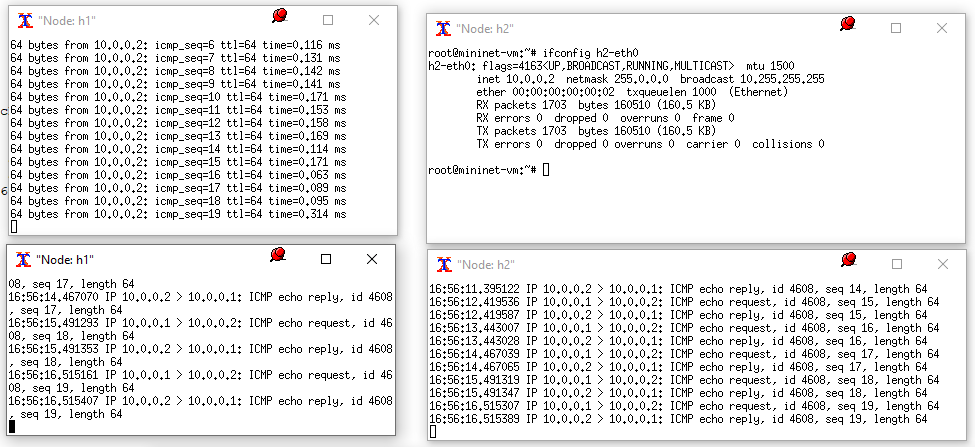
Now you can see that s1 is forwarding ARP messages and h2 receive an ARP reply, so it can start sending the ICMP (i.e., the ping) message.

*However, the ping is still unsuccessful, why?*

1. We need to add additional flow rules on how to reach certain destinations. Instead of specifying the dl\_type option, you can leave that blank, which will give it a wildcard. i.e., the rule will be valid for any dl\_type option.

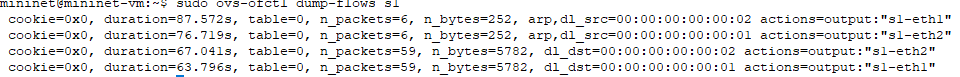
***sudo ovs-ofctl add-flow s1 dl\_dst=00:00:00:00:00:02,actions=output:\"s1-eth2\"***

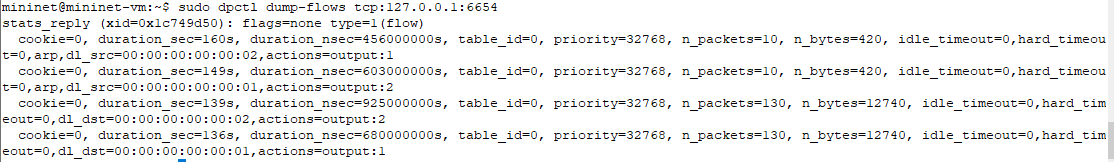
***sudo ovs-ofctl add-flow s1 dl\_dst=00:00:00:00:00:01,actions=output:\"s1-eth1\"***



Now you should see the ping command working properly.

1. Show the openflow table using ovs-ofctl





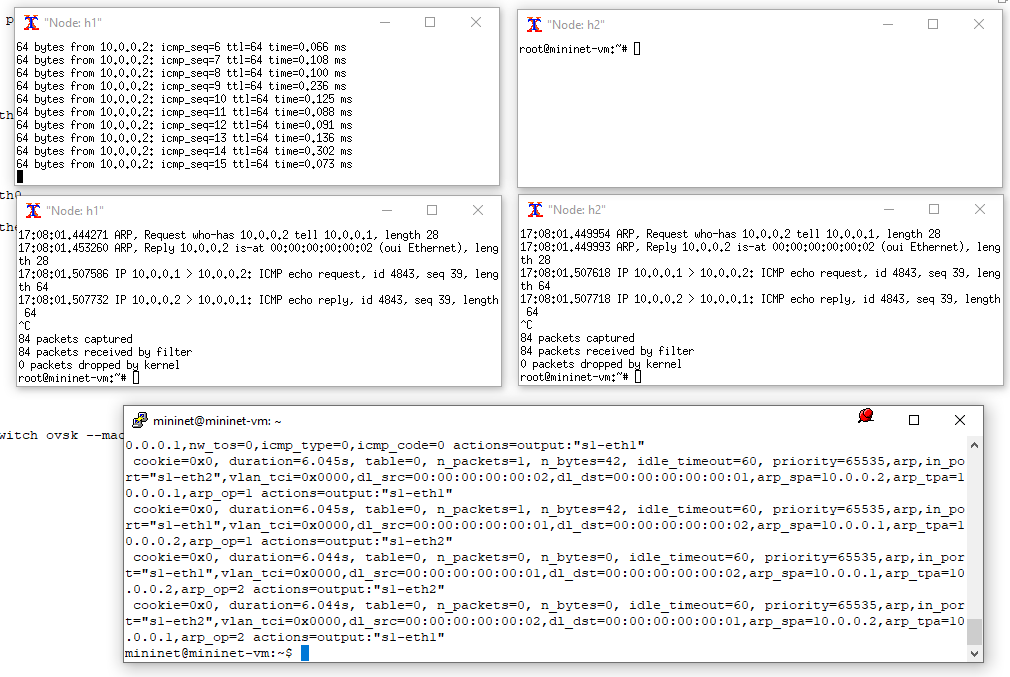
1. Of course, if we instead use a self-learning controller instead of dummy controller to connect hosts through the switch, we wouldn’t need to add flow rules, as these are automatically added by the controller.

**mininet@mininet-vm:~$ sudo -E mn --switch ovsk --mac**

Now first do a flow dump to see which flow rules are on.

Then run a ping again from h1 and h2. This now should work straight away.

Then run a flow dump again and see the rules that have been installed by the self-learning controller.



# References

Mininet Walkthrough: additional information on how to use mininet

<http://mininet.org/walkthrough/>

You can find more information and instructions related to ofctl at:

<https://www.openvswitch.org/support/dist-docs-2.5/ovs-ofctl.8.txt>