**CELL iteration 3 10/01/16: Tasks 3.0 through 3.2 Elective 1, Dottie Kessler 10/2/16**

1. Start VM, login (mininet/mininet) and obtain ip (**sudo dhclient eth1**)
2. Run Xming: then xterm from within mininet
3. Putty to VM; run wireshark then filter of ; capture->interfaces->loopback (**sudo wireshare &**)
4. Putty to VM run ODL (**cd ~/opendaylight; sudo ./run.sh -virt ovsdb**)
5. Browser to host open IP of VM (admin/admin) [**http://192.168.56.101:8080**](http://192.168.56.101:8080)

**Task 3.0 OpenFlow Recap**

OpenFlow is a protocol, standard, for interacting with the forwarding behaviors of switch. We control the behavior of the switches dynamically and programmatically using an openflow reference controller.

**Task 3.1 Learn about OpenStack**

**Task 3.2 Hands on Training with Open vSwitch elective 1 SOUTHBOUND**

3.2.1 Peruse [Open vSwitch site](http://openvswitch.org/).

3.2.2 Look at [Open vSwitch in Wikipedia](http://en.wikipedia.org/wiki/Open_vSwitch).

3.2.3 View [Open vSwitch YouTube Video 1](http://www.youtube.com/watch?v=rYW7kQRyUvA) (*Introduction to Open vSwitch; 14 mins*).

3.2.4 View [Open vSwitch YouTube Video](http://video.search.yahoo.com/video/play?p=open+vswitch&vid=3ac05486e824d905206a15d2bd8a1aad&l=35%3A50&turl=http%3A%2F%2Fts2.mm.bing.net%2Fth%3Fid%3DVN.608038391376314881%26pid%3D15.1&rurl=http%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D_PCRNUB7oNw&tit=An+Introduction+to+Open+vSwitch+-+Simon+Horman&c=5&sigr=11ajae23e&sigt=11eqog6b4&ct=p&age=0&fr=b2ie7&tt=b) 2 (*An Introduction to Open vSwitch; 36 mins*).

**Open vSwitch OVS – Main points**

1. sh from mininet cli to escape to the normal shell
2. OVS = opensource, virtual, provides a switching stack; comes installed with mininet
3. to connect a host and 2 tenant vms on that host to each other,
4. and to memorable networks, represent network connectivity
5. interconnect virtual machines within a host
6. interconnect virtual machines between different hosts across networks
7. supports traditional switch features, eg vlan
8. **ovs-vsctl** is the command to use to configure the virtual switch
9. **ovs-appctl fdb/show <argument>**  = to show mac addresses
10. an ovs needs to connect to an openflow controller
11. however, it starts with a default flow entry make it act like a layer 2 switch
12. Default action is normal L2 forwarding, like a simple switch

**(Task 3.2) Run Open vSwitch Commands SOUTHBOUND (Brent Salisbury exercises)**

**Core Components (Plus Kernel Module)**

mininet@mininet-vm:~$ **ps -ea | grep ov**

1002 ? 00:00:00 ovsdb-server **(conf changes are persistent, will survive a reboot)**

1080 ? 00:00:00 ovs-vswitchd **(core component runs in user space)**

**ovs-vsctl list bridge ( information from the ovsdb-server)**

**ovs-vsclt list port**

**ovs-vsclt list interface**

mininet@mininet-vm:~$ **sudo ovs-vsctl show**

0b8ed0aa-67ac-4405-af13-70249a7e8a96

ovs\_version: "2.0.2"

**add the bridge, a container to hold ids**

mininet@mininet-vm:~$ **sudo ovs-vsctl add-br br0**

mininet@mininet-vm:~$ **sudo ovs-vsctl show**

0b8ed0aa-67ac-4405-af13-70249a7e8a96

Bridge "br0"

Port "br0" **(Has an internal port with the same name as the bridge)**

Interface "br0" **(Currently not connected to anything except the local ip stack)**

type: internal

ovs\_version: "2.0.2"

**Setup the open vSwitch features, ports veth0 and vif1**

mininet@mininet-vm:~$ **sudo ovs-vsctl add-port br0 veth0**

mininet@mininet-vm:~$ **sudo ovs-vsctl add-port br0 vif1.0**

mininet@mininet-vm:~$ **sudo ovs-vsctl show**

0b8ed0aa-67ac-4405-af13-70249a7e8a96

Bridge "br0"

Port "vif1.0" **(ports can have > 1 interface, e.g. bonds, here we have a 1 to 1 mapping)**

Interface "vif1.0"

Port "veth0"

Interface "veth0"

Port "br0"

Interface "br0"

type: internal

ovs\_version: "2.0.2"

**You can easily remove a port**

mininet@mininet-vm:~$ **sudo ovs-vsctl -- --if-exists del-port br0 veth0**

mininet@mininet-vm:~$ **sudo ovs-vsctl show**

0b8ed0aa-67ac-4405-af13-70249a7e8a96

Bridge "br0"

Port "vif1.0" (only vif1 now since we deleted veth0)

Interface "vif1.0"

Port "br0"

Interface "br0"

type: internal

ovs\_version: "2.0.2"

**You can easily remove a bridge**

mininet@mininet-vm:~$ **sudo ovs-vsctl -- --if-exists del-br br0**

mininet@mininet-vm:~$ **sudo ovs-vsctl show**

0b8ed0aa-67ac-4405-af13-70249a7e8a96

ovs\_version: "2.0.2"

**(Task 3.2) OVS with Mininet; flow entries in an OpenFlow enabled switch (Dave Mahler exercises)**

(Source: [Dave Mahler’s Open vSwitch YouTube videos](http://www.youtube.com/watch?v=rYW7kQRyUvA))

<http://www.cse.iitb.ac.in/synerg/lib/exe/fetch.php?media=public:students:rinku:ovs_tut.pdf>

**Create a mininet network without a controller; we are going to manually enter flow entries**

mininet@mininet-vm**:~$ sudo mn --topo=single,3 --controller=none --mac**

\*\*\* Creating network

\*\*\* Adding controller

\*\*\* Adding hosts:

h1 h2 h3

\*\*\* Adding switches:

s1

\*\*\* Adding links:

(h1, s1) (h2, s1) (h3, s1)

\*\*\* Configuring hosts

h1 h2 h3

\*\*\* Starting controller

\*\*\* Starting 1 switches

s1 ...

\*\*\* Starting CLI:

**Show Nodes:**

mininet> **dump**

<Host h1: h1-eth0:10.0.0.1 pid=1399>

<Host h2: h2-eth0:10.0.0.2 pid=1402>

<Host h3: h3-eth0:10.0.0.3 pid=1404>

<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=1409>

**Show Links**

mininet> **net**

h1 h1-eth0:s1-eth1

h2 h2-eth0:s1-eth2

h3 h3-eth0:s1-eth3

s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0

**Show mapping**

mininet> **sh ovs-ofctl show s1**

OFPT\_FEATURES\_REPLY (xid=0x2): dpid:0000000000000001

n\_tables:254, n\_buffers:256

capabilities: FLOW\_STATS TABLE\_STATS PORT\_STATS QUEUE\_STATS ARP\_MATCH\_IP

actions: OUTPUT SET\_VLAN\_VID SET\_VLAN\_PCP STRIP\_VLAN SET\_DL\_SRC SET\_DL\_DST SET\_NW\_SRC SET\_NW\_DST SET\_NW\_TOS SET\_TP\_SRC SET\_TP\_DST ENQUEUE

1(s1-eth1): addr:1a:8b:6e:26:3f:e3

config: 0

state: 0

current: 10GB-FD COPPER

speed: 10000 Mbps now, 0 Mbps max

2(s1-eth2): addr:de:1a:74:79:ab:95

config: 0

state: 0

current: 10GB-FD COPPER

speed: 10000 Mbps now, 0 Mbps max

3(s1-eth3): addr:be:14:79:9e:39:53

config: 0

state: 0

current: 10GB-FD COPPER

speed: 10000 Mbps now, 0 Mbps max

LOCAL(s1): addr:92:62:6d:73:65:22

config: PORT\_DOWN

state: LINK\_DOWN

speed: 0 Mbps now, 0 Mbps max

OFPT\_GET\_CONFIG\_REPLY (xid=0x4): frags=normal miss\_send\_len=0

**Add first flow entry for the flow table**

mininet> **sh ovs-ofctl add-flow s1 action=normal**

mininet>

**pingall – all hosts are reachable**

mininet> **pingall**

\*\*\* Ping: testing ping reachability

h1 -> h2 h3

h2 -> h1 h3

h3 -> h1 h2

\*\*\* Results: 0% dropped (6/6 received)

**Show the flow entries**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=111.428s, table=0, n\_packets=24, n\_bytes=1680, idle\_age=46, actions=NORMAL

**Now let’s delete the flows and start over (note this deletes ALL flows)**

mininet> **sh ovs-ofctl del-flows s1**

**Check that the flows are gone**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

**pingall – Now no hosts are reachable**

mininet> **pingall**

\*\*\* Ping: testing ping reachability

h1 -> X X

h2 -> X X

h3 -> X X

\*\*\* Results: 100% dropped (0/6 received)

**flows with anything to s1 Port 1 will be sent to Port 2**

**flows with anything to s1 Port 2 will be sent to Port 1**

mininet> **sh ovs-ofctl add-flow s1 priority=500,in\_port=1,actions=output:2**

mininet> **sh ovs-ofctl add-flow s1 priority=500,in\_port=2,actions=output:1**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=79.031s, table=0, n\_packets=0, n\_bytes=0, idle\_age=79, priority=500,in\_port=1 actions=output:2

cookie=0x0, duration=38.341s, table=0, n\_packets=0, n\_bytes=0, idle\_age=38, priority=500,in\_port=2 actions=output:1

**h1 ping to h2 works**

mininet> **h1 ping -c2 h2**

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

64 bytes from 10.0.0.2: icmp\_seq=1 ttl=64 time=2.01 ms

64 bytes from 10.0.0.2: icmp\_seq=2 ttl=64 time=0.087 ms

--- 10.0.0.2 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms

rtt min/avg/max/mdev = 0.087/1.048/2.010/0.962 ms

**You cannot ping from h3, no flow has been added, that should be dropped, timeout**

mininet> **h3 ping -c2 h2**

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

From 10.0.0.3 icmp\_seq=1 Destination Host Unreachable

From 10.0.0.3 icmp\_seq=2 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---

2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1001ms

pipe 2

**Show that flows have packets**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=307.906s, table=0, n\_packets=4, n\_bytes=280, idle\_age=180, priority=500,in\_port=1 actions=output:2

cookie=0x0, duration=267.216s, table=0, n\_packets=4, n\_bytes=280, idle\_age=180, priority=500,in\_port=2 actions=output:1

mininet>

**Default priority is 32768, we are going to give action drop, a wild card flow**

mininet> sh ovs-ofctl add-flow s1 priority=32768,actions=drop

**Now no pings work because the flow to drop has highest priority**

mininet> **h1 ping -c2 h2**

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---

2 packets transmitted, 0 received, 100% packet loss, time 1008ms

**Show the flows**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=560.253s, table=0, n\_packets=4, n\_bytes=280, idle\_age=432, priority=500,in\_port=1 actions=output:2

cookie=0x0, duration=519.563s, table=0, n\_packets=4, n\_bytes=280, idle\_age=432, priority=500,in\_port=2 actions=output:1

cookie=0x0, duration=146.921s, table=0, n\_packets=5, n\_bytes=322, idle\_age=61, actions=drop

**Remove the wildcard flow that causes the drops (use strict to only delete particular flow)**

mininet> **sh ovs-ofctl del-flows s1 --strict**

**Drop flow is now gone**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=724.932s, table=0, n\_packets=4, n\_bytes=280, idle\_age=597, priority=500,in\_port=1 actions=output:2

cookie=0x0, duration=684.242s, table=0, n\_packets=4, n\_bytes=280, idle\_age=597, priority=500,in\_port=2 actions=output:1

**Now pings work between h1/h2**

mininet> **h1 ping -c2 h2**

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

64 bytes from 10.0.0.2: icmp\_seq=1 ttl=64 time=1.94 ms

64 bytes from 10.0.0.2: icmp\_seq=2 ttl=64 time=0.083 ms

--- 10.0.0.2 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms

rtt min/avg/max/mdev = 0.083/1.014/1.946/0.932 ms

mininet> **h2 ping -c2 h1**

PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

64 bytes from 10.0.0.1: icmp\_seq=1 ttl=64 time=0.855 ms

64 bytes from 10.0.0.1: icmp\_seq=2 ttl=64 time=0.086 ms

--- 10.0.0.1 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1001ms

rtt min/avg/max/mdev = 0.086/0.470/0.855/0.385 ms

**ping still fails to h3, no flow for that**

mininet> **pingall**

\*\*\* Ping: testing ping reachability

h1 -> h2 X

h2 -> h1 X

h3 -> X X

\*\*\* Results: 66% dropped (2/6 received)

**Remove all the flows**

mininet> **sh ovs-ofctl del-flows s1**

**Layer 2 Matching; MAC addresses in the flows**

**create the network**

mininet@mininet-vm:~$ **sudo mn --topo=single,3 --controller=none --mac**

\*\*\* Creating network

\*\*\* Adding controller

\*\*\* Adding hosts:

h1 h2 h3

\*\*\* Adding switches:

s1

\*\*\* Adding links:

(h1, s1) (h2, s1) (h3, s1)

\*\*\* Configuring hosts

h1 h2 h3

\*\*\* Starting controller

\*\*\* Starting 1 switches

s1 ...

\*\*\* Starting CLI:

**Add flows using mac addresses (dl\_src)**

**mininet> sh ovs-ofctl add-flow s1 dl\_src=00:00:00:00:00:01, dl\_dst=00:00:00:00:00:02,actions=output:2**

**h2 MAC address will flow to s1 Port 2 and out Port 1 to MAC address of h1:**

**mininet> sh ovs-ofctl add-flow s1 dl\_src=00:00:00:00:00:02, dl\_dst=00:00:00:00:00:01,actions=output:1**

**We must account for ARP** so that hosts can learn each others’ ARP addresses. Flood sends packet to all ports except the one it arrived on; nm\_proto=1 is the ARP request:

**mininet> sh ovs-ofctl add-flow s1 dl\_type=0x806,nw\_proto=1,actions=flood**

**pingall**, (h1/h2 works, h3 still not in the flow)

h1 -> h2 X

h2 -> h1 X

h3 -> X X

\*\*\* Results: 66% dropped (2/6 received)

**Show the flows**

mininet> **sh ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=477.072s, table=0, n\_packets=3, n\_bytes=238, idle\_age=128, dl\_src=00:00:00:00:00:01,dl\_dst=00:00:00:00:00:02 actions=output:2

cookie=0x0, duration=443.609s, table=0, n\_packets=4, n\_bytes=280, idle\_age=128, dl\_src=00:00:00:00:00:02,dl\_dst=00:00:00:00:00:01 actions=output:1

cookie=0x0, duration=330.607s, table=0, n\_packets=13, n\_bytes=546, idle\_age=113, arp,arp\_op=1 actions=FLOOD

**Remove all the flows**

mininet> **sh ovs-ofctl del-flows s1**

**Layer 3 Matching (match flows at the IP address level)** (dl\_type=0x800 (IPv4):)

**Add flows** (I added from separate putty window instead of the sh from within the mininet cli

mininet@mininet-vm:~$ **sh ovs-ofctl add-flow s1 priority=500,dl\_type=0x800,nw\_src=10.0.0.0/24,nw\_dst=10.0.0.0/24,actions=normal**

mininet@mininet-vm:~$ **sudo ovs-ofctl add-flow s1 priority=800,ip,nw\_src=10.0.0.3,actions=mod\_nw\_tos:184,normal**

**Add ARP flows**

mininet> **sh ovs-ofctl add-flow s1 arp,nw\_dst=10.0.0.1,action=output:1**

mininet> **sh ovs-ofctl add-flow s1 arp,nw\_dst=10.0.0.2,action=output:2**

mininet> **sh ovs-ofctl add-flow s1 arp,nw\_dst=10.0.0.3,action=output:3**

mininet> **sh ovs-ofctl dump-flows s1 (no activity yet)**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=46.415s, table=0, n\_packets=0, n\_bytes=0, idle\_age=46, arp,arp\_tpa=10.0.0.3 actions=output:3

cookie=0x0, duration=69.844s, table=0, n\_packets=0, n\_bytes=0, idle\_age=69, arp,arp\_tpa=10.0.0.2 actions=output:2

cookie=0x0, duration=73.898s, table=0, n\_packets=0, n\_bytes=0, idle\_age=73, arp,arp\_tpa=10.0.0.1 actions=output:1

cookie=0x0, duration=144.024s, table=0, n\_packets=0, n\_bytes=0, idle\_age=144, priority=800,ip,nw\_src=10.0.0.3 actions=mod\_nw\_tos:184,NORMAL

cookie=0x0, duration=168.074s, table=0, n\_packets=0, n\_bytes=0, idle\_age=168, priority=500,ip,nw\_src=10.0.0.0/24,nw\_dst=10.0.0.0/24 actions =NORMAL

mininet> **pingall**

\*\*\* Ping: testing ping reachability

h1 -> h2 h3

h2 -> h1 h3

h3 -> h1 h2

\*\*\* Results: 0% dropped (6/6 received)

mininet**> sh ovs-ofctl dump-flows s1** (now all have activity)

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=256.415s, table=0, n\_packets=4, n\_bytes=168, idle\_age=79, arp,arp\_tpa=10.0.0.3 actions=output:3

cookie=0x0, duration=279.844s, table=0, n\_packets=4, n\_bytes=168, idle\_age=79, arp,arp\_tpa=10.0.0.2 actions=output:2

cookie=0x0, duration=283.898s, table=0, n\_packets=4, n\_bytes=168, idle\_age=79, arp,arp\_tpa=10.0.0.1 actions=output:1

cookie=0x0, duration=354.024s, table=0, n\_packets=4, n\_bytes=392, idle\_age=84, priority=800,ip,nw\_src=10.0.0.3 actions=mod\_nw\_tos:184,NORMAL

cookie=0x0, duration=162.797s, table=0, n\_packets=8, n\_bytes=784, idle\_age=84, priority=500,ip,nw\_src=10.0.0.0/24,nw\_dst=10.0.0.0/24 actions=NORMAL

**Remove all the flows**

mininet> **sh ovs-ofctl del-flows s1**

**Layer 4 Matching**

mininet@mininet-vm:~$ **sudo mn --topo=single,3 --controller=none --mac**

\*\*\* Creating network

\*\*\* Adding controller

\*\*\* Adding hosts:

h1 h2 h3

\*\*\* Adding switches:

s1

\*\*\* Adding links:

(h1, s1) (h2, s1) (h3, s1)

\*\*\* Configuring hosts

h1 h2 h3

\*\*\* Starting controller

\*\*\* Starting 1 switches

s1 ...

\*\*\* Starting CLI:

mininet> **h3 python –m SimpleHTTPServer 80 &**

mininet> **sh ovs-ofctl add-flow s1 arp,actions=normal**

**Add L4 matching, all traffic to port 3**

mininet> **sh ovs-ofctl add-flow s1 priority=500,dl\_type=0x800,nw\_proto=6,tp\_dst=80,actions=output:3**

mininet> **h1 curl h3**

mininet> **h2 curl h3**

**Open vSwitch Multiple Flow Tables**

(Source: [Dave Mahler Multiple Flow Tables video](http://www.youtube.com/watch?v=TD5wmoD7XOE))

Flows move from the lowest in order to the highest, matches highest priority

mininet@mininet-vm: **sudo mn --topo=single,3 --controller=none --mac**

\*\*\* Creating network

\*\*\* Adding controller

\*\*\* Adding hosts:

h1 h2 h3

\*\*\* Adding switches:

s1

\*\*\* Adding links:

(h1, s1) (h2, s1) (h3, s1)

\*\*\* Configuring hosts

h1 h2 h3

\*\*\* Starting controller

\*\*\* Starting 1 switches

s1 ...

\*\*\* Starting CLI:

**Give h1 a default gateway (doesn’t even exist)**

**Give h1 and h2 static ARP entries**

**(so it thinks it knows how to connect and does not do arp request)**

mininet> **h1 route add default gw 10.0.0.254 h1-eth0**

mininet> **h1 arp -s 10.0.0.254 00:00:00:00:11:11**

**h2**

mininet> **h2 route add default gw 10.0.0.254 h2-eth0**

mininet> **h2 arp -s 10.0.0.254 00:00:00:00:11:11**

**Give h3 an ip address, and the static ARP address**

mininet> **h3 ifconfig h3-eth0 30.0.0.3 netmask 255.255.255.0**

mininet> **h3 route add default gw 30.0.0.254 h3-eth0**

mininet> **h3 arp -s 30.0.0.254 00:00:00:00:33:33**

**Launch Python Server**

mininet> **h3 sudo python -m SimpleHTTPServer 80 &**

**(vi tables.txt) Create tables.txt to add flows**

**Resubmit is what instructs the flow to the next table**

**h1 and h2 are in the same submit, h3 is in a different subnet**

**Flow Table 0 Access Control**

table=0,ip,nw\_src=10.0.0.0/24,nw\_dst=10.0.0.0/24,actions=resubmit(,1)

table=0,arp,nw\_src=10.0.0.0/24,nw\_dst=10.0.0.0/24,actions=resubmit(,1)

table=0,icmp,nw\_src=10.0.0.1,nw\_dst=30.0.0.3,actions=resubmit(,1)

table=0,tcp,nw\_src=10.0.0.1,nw\_dst=30.0.0.3,tp\_dst=80,actions=resubmit(,1)

table=0,ip,nw\_src=30.0.0.3,actions=resubmit(,1) // table and ingress port for next table

table=0,priority=0,actions=drop //default drop

**Flow Table 1 NAT (h1 to h2, src is changed to 5.5.5.5)**

table=1,ip,nw\_src=10.0.0.1,nw\_dst=30.0.0.3,actions=mod\_nw\_src=5.5.5.5,resubmit(,2)

table=1,ip,nw\_src=30.0.0.3,nw\_dst=5.5.5.5,actions=mod\_nw\_dst=10.0.0.1,resubmit(,2)

table=1,priority=0,actions=resubmit(,2) // default rule, just forward to table 2

**Flow Table 2 Routing**

table=2,ip,nw\_dst=10.0.0.1,actions=mod\_dl\_dst=00:00:00:00:00:01,output:1

table=2,ip,nw\_dst=10.0.0.2,actions=mod\_dl\_dst=00:00:00:00:00:02,output:2

table=2,ip,nw\_dst=30.0.0.3,actions=mod\_dl\_dst=00:00:00:00:00:03,output:3

priority=0,table=2,arp,nw\_dst=10.0.0.1,actions=output:1

priority=0,table=2,arp,nw\_dst=10.0.0.2,actions=output:2

**Add flow entries for our new flow table**

mininet> **sh ovs-ofctl add-flows s1 tables.txt**

**h1 and h2 should be able to ping each other (ping h2 from h1)**

mininet> **h1 ping -c2 h2**

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

64 bytes from 10.0.0.2: icmp\_seq=1 ttl=64 time=1.16 ms

64 bytes from 10.0.0.2: icmp\_seq=2 ttl=64 time=0.083 ms

--- 10.0.0.2 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1002ms

rtt min/avg/max/mdev = 0.083/0.621/1.160/0.539 ms

**h1 should be able to ping h3 on the different subnet**

mininet> h1 ping -c2 30.0.0.3

PING 30.0.0.3 (30.0.0.3) 56(84) bytes of data.

64 bytes from 30.0.0.3: icmp\_seq=1 ttl=64 time=1.29 ms

64 bytes from 30.0.0.3: icmp\_seq=2 ttl=64 time=0.173 ms

--- 30.0.0.3 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1002ms

rtt min/avg/max/mdev = 0.173/0.732/1.291/0.559 ms

**h2 should not be able to ping h3, this failure is correct**

**Access control table 0 prevented this action**

mininet> **h2 ping -c2 30.0.0.3**

PING 30.0.0.3 (30.0.0.3) 56(84) bytes of data.

--- 30.0.0.3 ping statistics ---

2 packets transmitted, 0 received, 100% packet loss, time 999ms

**h1 can still connect to h3 web server**

**mininet>** **h1 curl 30.0.0.3**

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>

<title>Directory listing for /</title>

<body>

<h2>Directory listing for /</h2>

<hr>

<ul>

<li><a href=".bash\_history">.bash\_history</a>

<li><a href=".bash\_logout">.bash\_logout</a>

<li><a href=".bashrc">.bashrc</a>

<li><a href=".cache/">.cache/</a>

<li><a href=".dbus/">.dbus/</a>

<li><a href=".gitconfig">.gitconfig</a>

<li><a href=".mininet\_history">.mininet\_history</a>

<li><a href=".profile">.profile</a>

<li><a href=".rnd">.rnd</a>

<li><a href=".ssh/">.ssh/</a>

<li><a href=".viminfo">.viminfo</a>

<li><a href=".wireshark/">.wireshark/</a>

<li><a href=".Xauthority">.Xauthority</a>

<li><a href="dak/">dak/</a>

<li><a href="distributions-virtualization-0.1.1-osgipackage.zip">distributions-virtualization-0.1.1-osgipackage.zip</a>

<li><a href="freshpox/">freshpox/</a>

<li><a href="install-mininet-vm.sh">install-mininet-vm.sh</a>

<li><a href="loxigen/">loxigen/</a>

<output truncated>

**Wireshark output for h1 ping –c 30.0.0.3**

**This is when I had the entry in the 3rd table wrong, you can see in Wireshark is shows the 5.5.5.5 entries. Once I fixed the line that was wrong, this was corrected and Wireshark view was correct**

**Correct Table:**

**Flow Table 2 Routing**

table=2,ip,nw\_dst=10.0.0.1,actions=mod\_dl\_dst=00:00:00:00:00:01,output:1

table=2,ip,nw\_dst=10.0.0.2,actions=mod\_dl\_dst=00:00:00:00:00:02,output:2

table=2,ip,nw\_dst=30.0.0.3,actions=mod\_dl\_dst=00:00:00:00:00:03,output:3

priority=0,table=2,arp,nw\_dst=10.0.0.1,actions=output:1

priority=0,table=2,arp,nw\_dst=10.0.0.2,actions=output:2

**The way I had it, I had a wrong entry**

**Flow Table 2 Routing**

table=2,ip,nw\_dst=10.0.0.1,actions=mod\_dl\_dst=00:00:00:00:00:01,output:1

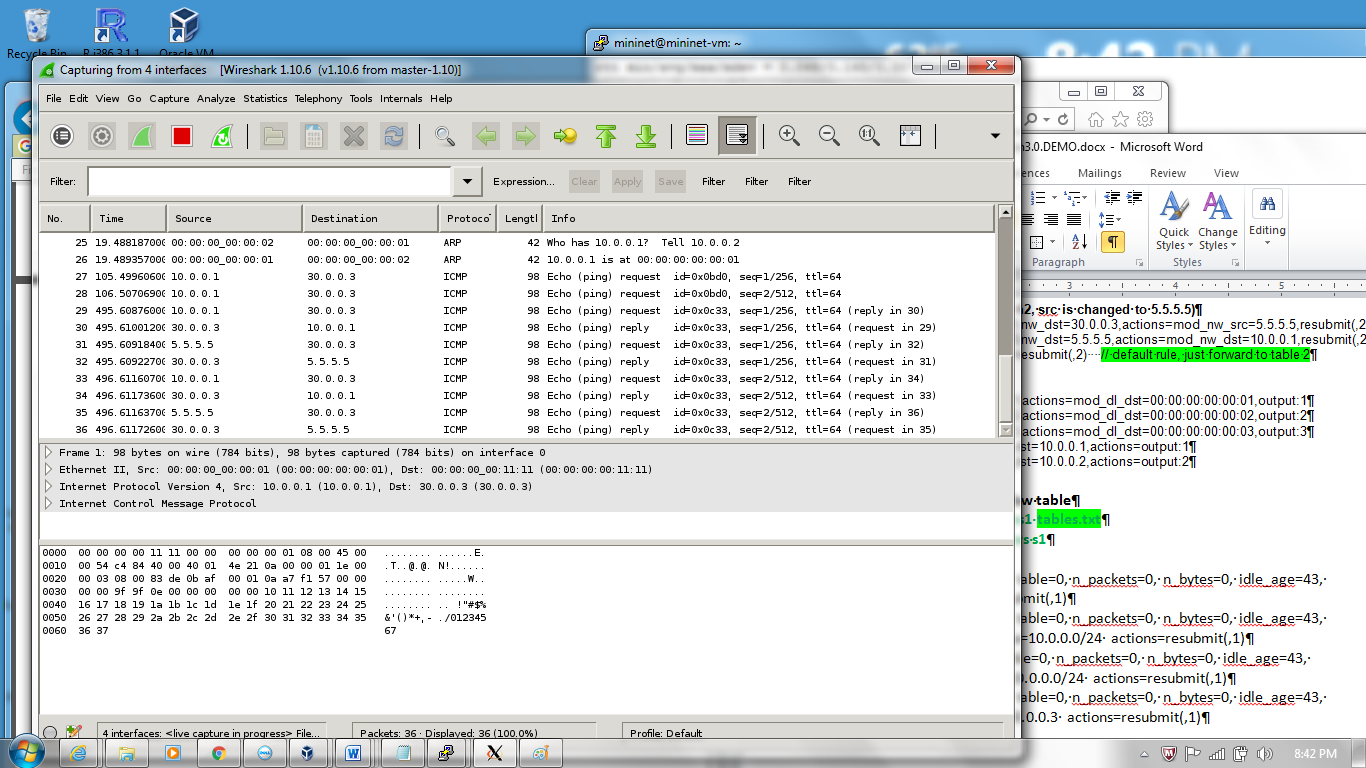
table=2,ip,nw\_dst=10.0.0.2,actions=mod\_dl\_dst=00:00:00:00:00:02,output:2

table=2,ip,nw\_dst=10.0.0.3,actions=mod\_dl\_dst=00:00:00:00:00:03,output:3

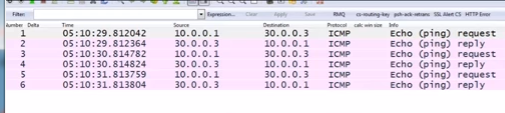
priority=0,table=2,arp,nw\_dst=10.0.0.1,actions=output:1

priority=0,table=2,arp,nw\_dst=10.0.0.2,actions=output:2

**Wrong:**



**Right: (thisis from a YouTube video I used to help me problem solve)**



**Command history within mininet cli**

**sudo mn --topo=single,3 --controller=none --mac**

**h1 route add default gw 10.0.0.254 h1-eth0**

**h1 arp -s 10.0.0.254 00:00:00:00:11:11**

**h2 route add default gw 10.0.0.254 h2-eth0**

**h2 arp -s 10.0.0.254 00:00:00:00:11:11**

**h3 ifconfig h3-eth0 30.0.0.3 netmask 255.255.255.0**

**h3 route add default gw 30.0.0.254 h3-eth0**

**h3 arp -s 30.0.0.254 00:00:00:00:33:33**

**h3 sudo python -m SimpleHTTPServer 80 &**

**sh ovs-ofctl add-flows s1 tables.txt**

**h1 ping -c2 h2**

**h1 ping -c2 30.0.0.3**

**h1 curl 30.0.0.3**