EN2150 - Communication Network Engineering

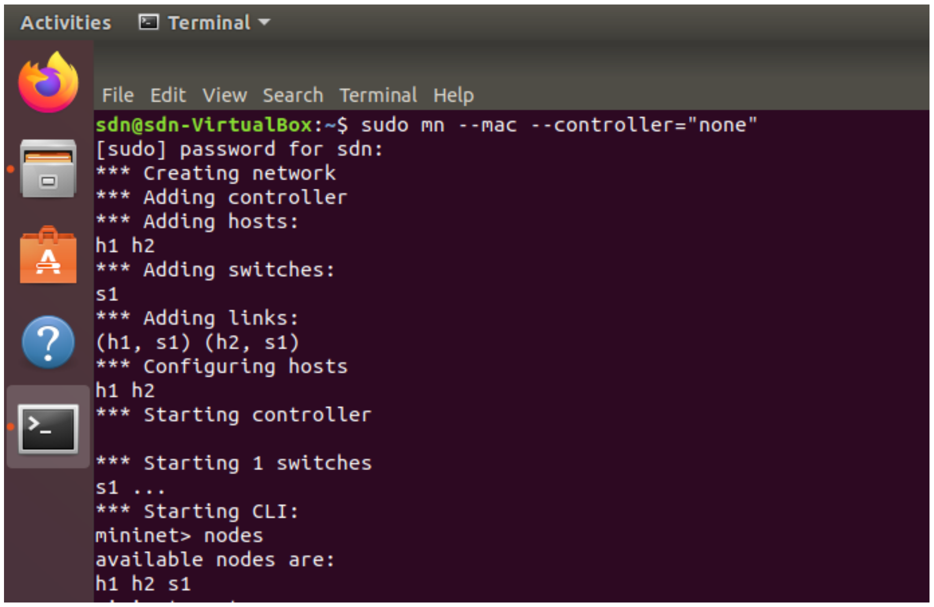
SDN Lab (Individual)

210129P - Dilshan N.L

OpenFlow Configuration Lab

Part – 01

1. **Create a simple network on the Mininet:**
   sudo mn --mac --controller="none"

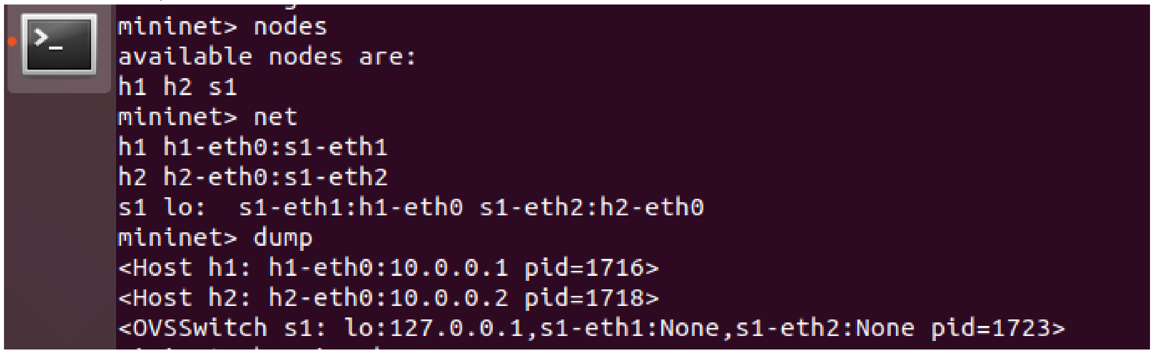


```
sdn@sdn-VirtualBox:~$ sudo mn --mac --controller="none"
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
h1 h2 s1
```

   This command starts a Mininet network with two hosts (h1, h2) and one switch (s1) without a controller. It creates and configures the network, and then starts the Mininet CLI.

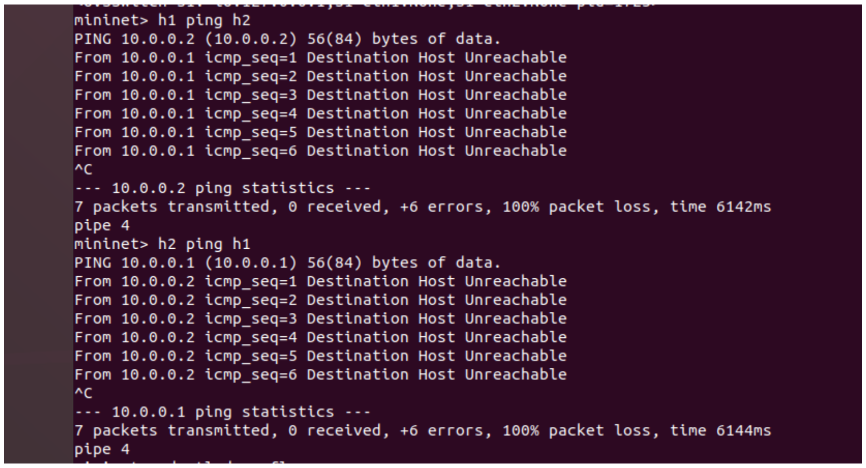
2. **Try the following commands in the Mininet CLI to get an understanding of the network:**
   mininet> nodes
   mininet> net
   mininet> dump

   The ‘nodes’ command lists the nodes: ‘h1’, ‘h2’, and ‘s1’. The ‘net’ command shows that ‘h1’ and ‘h2’ are connected to ‘s1’. The ‘dump’ command provides detailed info, including IPs for ‘h1’ and ‘h2’, and interface details for ‘s1’.



```
mininet> nodes
available nodes are:
h1 h2 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1716>
<Host h2: h2-eth0:10.0.0.2 pid=1718>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1723>
```

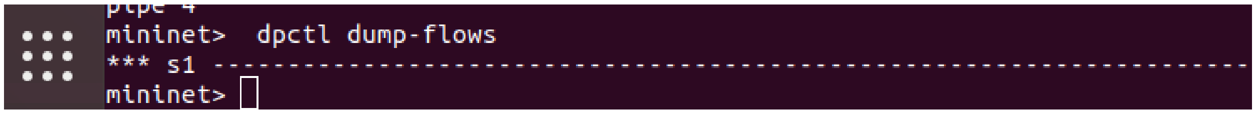
3. **Attempt pings between the hosts:**
   mininet> h1 ping h2
   mininet> h2 ping h1



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
From 10.0.0.1 icmp_seq=5 Destination Host Unreachable
From 10.0.0.1 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6142ms
pipe 4
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
From 10.0.0.2 icmp_seq=4 Destination Host Unreachable
From 10.0.0.2 icmp_seq=5 Destination Host Unreachable
From 10.0.0.2 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.0.1 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6144ms
pipe 4
```

The ping attempts failed because the network setup does not include an SDN controller. In an SDN environment, the controller is essential for managing packet flow between hosts. Since the network was initialized with the --controller="none" option, the switch lacks the necessary flow rules to route packets between the hosts. As a result, the absence of a controller prevented the successful transmission of pings between the hosts.

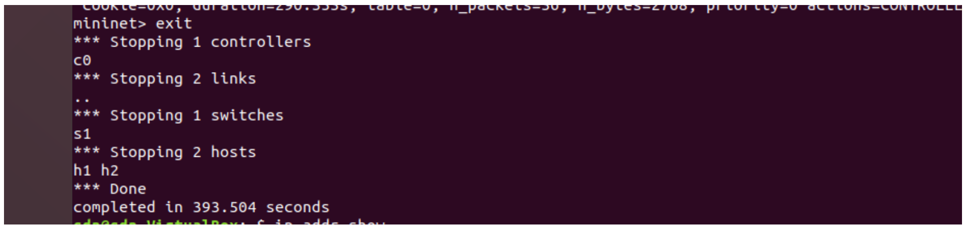
4. **Check the flow table of the switch:**
   mininet> dpctl dump-flows



```
mininet>  dpctl dump-flows
*** s1 ------------------------------------------------------------------------
mininet> 
```

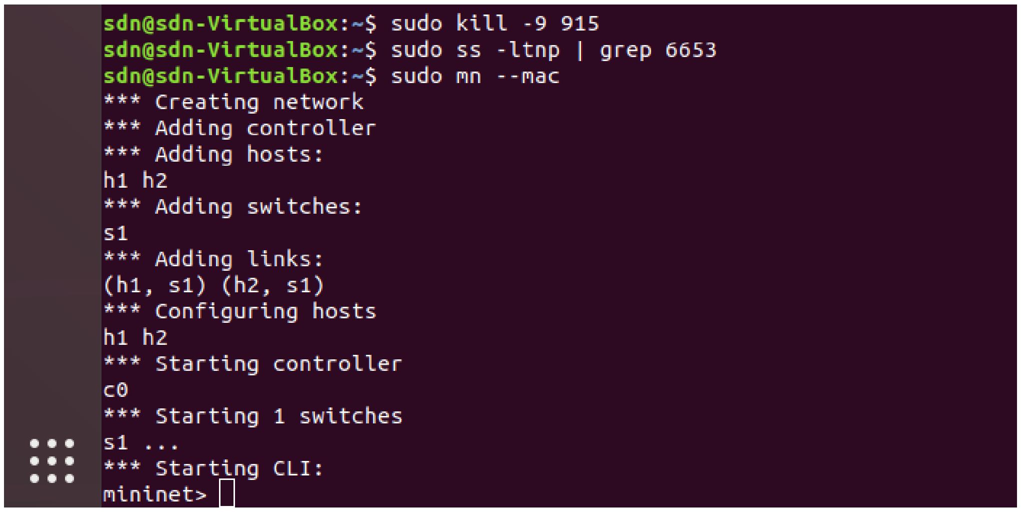
The flow table is empty because there's no SDN controller connected to the switch to add rules. The SDN controller normally populates the flow table with rules to direct how packets should be handled. Without these rules, the switch doesn't know how to process incoming packets, which is why you're seeing "Destination Host Unreachable" messages during ping attempts. Essentially, the lack of flow rules means the switch isn't set up to handle traffic, causing the ping failures.

5. **Exit from the Mininet CLI:**
   mininet> exit



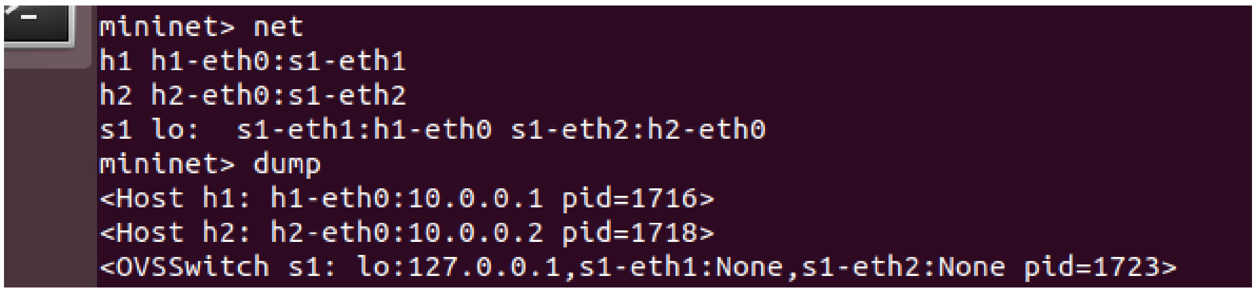
```
 cookie=0x0, duration=290.333s, table=0, n_packets=30, n_bytes=2708, priority=0 actions=CONTROLLE
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 393.504 seconds
```

6. **Create another simple network on the Mininet VM but this time use the default controller:**
   sudo mn –mac



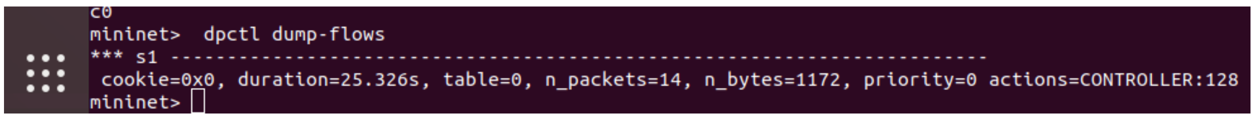
```
sdn@sdn-VirtualBox:~$ sudo kill -9 915
sdn@sdn-VirtualBox:~$ sudo ss -ltnp | grep 6653
sdn@sdn-VirtualBox:~$ sudo mn --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> 
```

7. **Try the following commands in the Mininet CLI to get an understanding of the network:**
   mininet> nodes
   mininet> net
   mininet> dump



```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1716>
<Host h2: h2-eth0:10.0.0.2 pid=1718>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1723>
```

8. **Check the flow table of the switch:**
   mininet> dpctl dump-flows



```
c0
mininet>  dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=25.326s, table=0, n_packets=14, n_bytes=1172, priority=0 actions=CONTROLLER:128
mininet> 
```

The output shows a single flow entry in the switch's flow table. Here's a breakdown of the details:

- Cookie: The unique identifier for this flow entry is 0x0.
- Duration: The flow entry has been active for 25.326 seconds.
- Table: This entry is in table number 0.
- Packets and Bytes: The entry has matched 14 packets and 1172 bytes of data.
- Priority: The priority level of the flow entry is 0.
- Actions: The action for this flow is CONTROLLER:128, meaning packets matching this flow are sent to the SDN controller for further processing.
- This setup indicates that the switch is configured to forward certain packets to the SDN controller instead of handling them locally. This often happens when there are specific rules or default actions that direct packets to the controller for more detailed handling or decision-making.

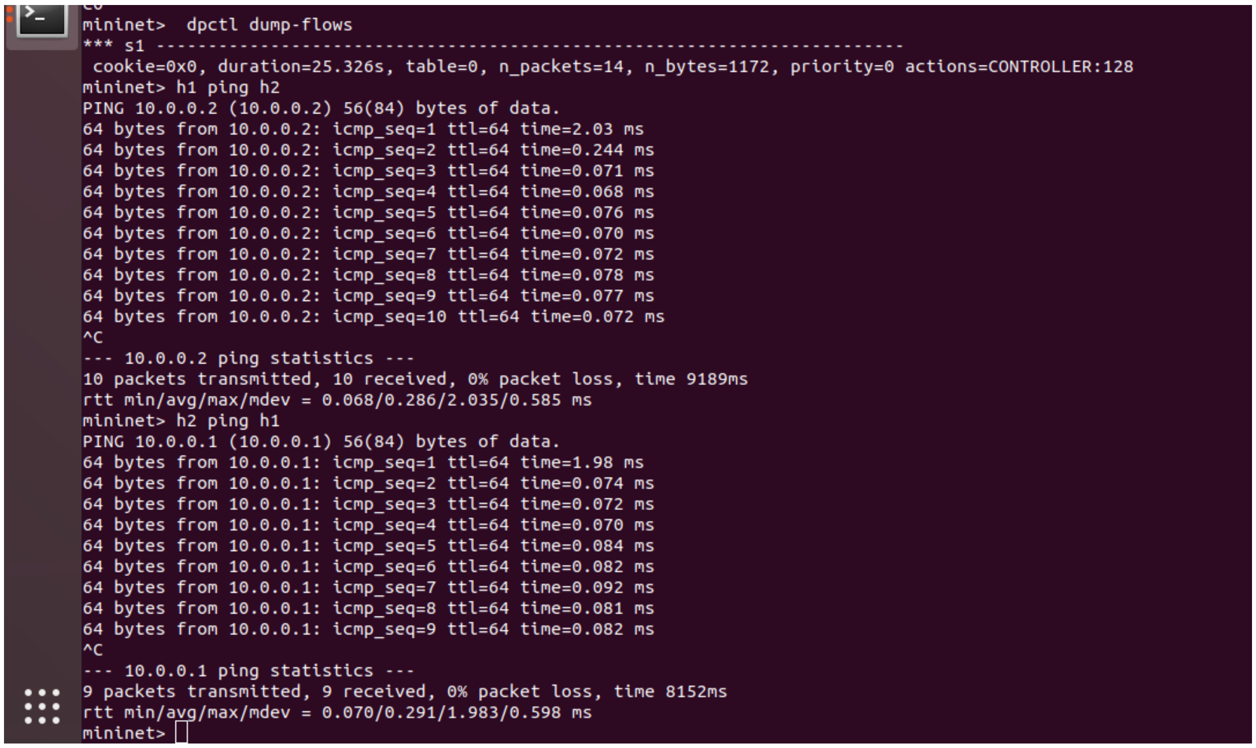
9. **Attempt pings between the hosts**

```
mininet> h1 ping h2
mininet> h2 ping h1
```

The pings between hosts h1 and h2 were successful, as shown by the results. This success indicates that network communication between the two hosts is working properly. The successful pings are due to the presence of an SDN controller in the network setup. The controller manages the flow rules, ensuring that packets are correctly routed between hosts.
In Software-Defined Networking (SDN), the controller is crucial because it determines how switches handle and forward packets. Without the controller, the switch would not have the necessary rules to route packets between hosts, leading to failed pings. With the controller in place, the switch can dynamically set up its flow table to enable communication between h1 and h2.
The ping results show that the first ping from h2 to h1 took slightly longer than subsequent pings. This delay is mainly due to the Address Resolution Protocol (ARP) process:

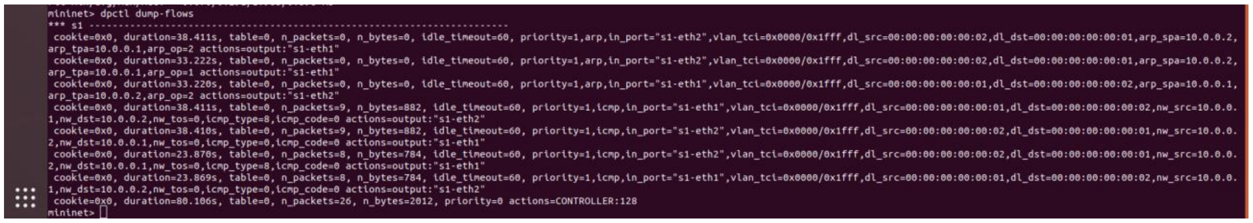
- **ARP Request**: When the first ping is sent, h2 needs to find out the MAC address of h1. It does this by broadcasting an ARP request to the network asking for the MAC address corresponding to 10.0.0.1.
- **ARP Reply**: h1 responds to the ARP request with its MAC address. Once h2 receives this information, it can send the ping packet to h1.
- **Subsequent Pings**: After the initial ARP process, h2 caches the MAC address of h1. Therefore, later pings do not require ARP resolution, leading to quicker response times and lower latency for subsequent packets.

Overall, the network communication is functioning correctly, and the presence of the SDN controller allows for effective packet forwarding and minimal delay after the initial ARP resolution.



```
mininet>  dpctl dump-flows
*** s1 -----------------------------------------------------------------------
 cookie=0x0, duration=25.326s, table=0, n_packets=14, n_bytes=1172, priority=0 actions=CONTROLLER:128
mininet> h1 ping 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=2.03 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.244 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.076 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.077 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.072 ms
^C
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9189ms
rtt min/avg/max/mdev = 0.068/0.286/2.035/0.585 ms
mininet> h2 ping 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=1.98 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.072 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.084 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.082 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.092 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.081 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.082 ms
^C
--- 10.0.0.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8152ms
rtt min/avg/max/mdev = 0.070/0.291/1.983/0.598 ms
mininet> 
```

10. **Check the flow table of the switch again:**
    mininet> dpctl dump-flows

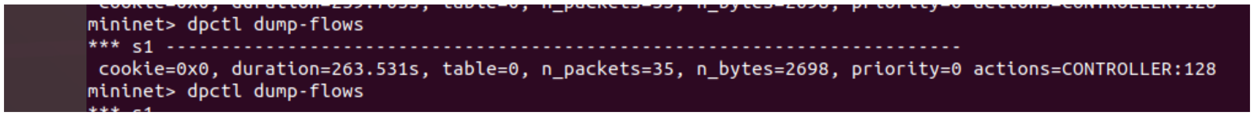


```
mininet> dpctl dump-flows
*** s1 ----------------------------------------------------------------------
 cookie=0x0, duration=38.411s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
arp_tpa=10.0.0.1,arp_op=2 actions=output:"s1-eth1"
 cookie=0x0, duration=33.222s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,arp_spa=10.0.0.2,
arp_tpa=10.0.0.1,arp_op=1 actions=output:"s1-eth1"
 cookie=0x0, duration=33.220s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.0.1,
arp_tpa=10.0.0.2,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=38.411s, table=0, n_packets=9, n_bytes=882, idle_timeout=60, priority=1,icmp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.
1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth2"
 cookie=0x0, duration=38.410s, table=0, n_packets=9, n_bytes=882, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.0.
2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth1"
 cookie=0x0, duration=23.870s, table=0, n_packets=8, n_bytes=784, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.0.
2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth1"
 cookie=0x0, duration=23.869s, table=0, n_packets=8, n_bytes=784, idle_timeout=60, priority=1,icmp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.0.
1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth2"
 cookie=0x0, duration=80.106s, table=0, n_packets=26, n_bytes=2012, priority=0 actions=CONTROLLER:128
mininet> 
```

The flow table entries handle ARP and ICMP traffic between hosts. For ARP packets, fields like in port, dl src (source MAC), dl dst (destination MAC), arp spa (source IP), arp tpa (target IP), and arp op (operation) define how ARP requests and replies are processed. For ICMP traffic, fields like nw src (source IP), nw dst (destination IP), icmp type, and icmp code specify the ICMP messages and actions for packet handling.

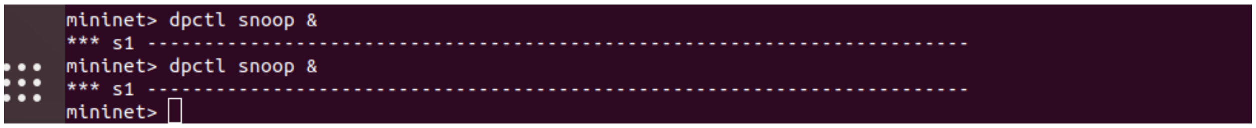
- Duration: How long the entry has been active.
- n packets and n bytes: Number of packets and bytes matching the flow.
- Idle Timeout: Time before the entry is removed if not used.
- Priority: Determines the order of flow entry matching.
- Actions: Specifies how packets are processed, such as forwarding to specific ports.
- These rules were added as the switch processed ARP and ICMP traffic, allowing it to forward packets correctly and achieve successful pings between the hosts.

11. **Wait for the flows to timeout. That is, wait until the output of the following command is again empty:**
    mininet> dpctl dump-flows



```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------
 cookie=0x0, duration=263.531s, table=0, n_packets=35, n_bytes=2698, priority=0 actions=CONTROLLER:128
mininet> dpctl dump-flows
```

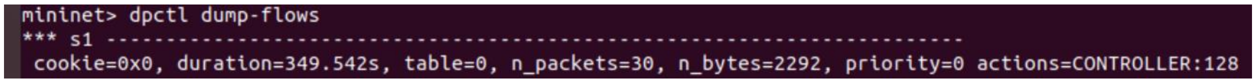
12. **We will now enable snooping to see the message flow between the switch and controller:**
    mininet> dpctl snoop &



```
mininet> dpctl snoop &
*** s1 ------------------------------------------------------------------
mininet> dpctl snoop &
*** s1 ------------------------------------------------------------------
mininet> 
```

13. **Wait for the flows to timeout. That is, wait until the output of the following command is again empty:**
    mininet> dpctl dump-flows

14. **Check the flow table again:**
    mininet> dpctl dump-flows



```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------
 cookie=0x0, duration=349.542s, table=0, n_packets=30, n_bytes=2292, priority=0 actions=CONTROLLER:128
```

The flow table is empty for a few possible reasons:

- Idle Timeout: Flow entries might be removed automatically if no packets match them within a set period. If the entries had an idle timeout and no packets matched, they would be deleted.

- Table Clearance: The flow table may have been cleared by the switch or controller, either manually or automatically, to make space for new entries or after a reset.

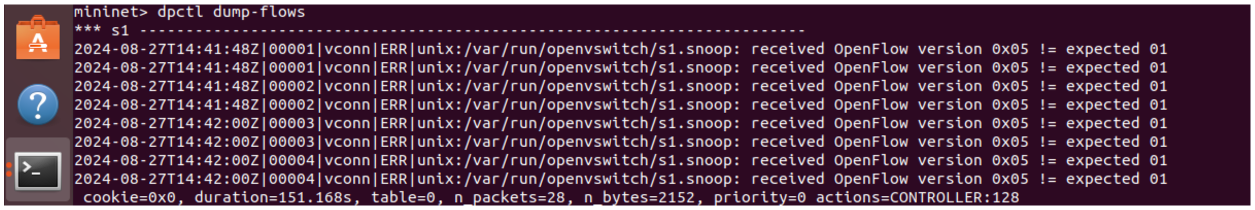
- Switch or Controller Actions: The switch or controller might have cleared the entries as part of its normal operation or management.

Without entries, the switch will handle new packets based on default settings or wait for new instructions from the controller. New flow entries will appear when new traffic is processed, depending on the switch's and controller's setup.

15. **Pings between the hosts again:**
    mininet> h1 ping h2

mininet> h2 ping h1

16. **Check the flow table npw:**
    mininet> dpctl dump-flows



```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
2024-08-27T14:41:48Z|00001|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:41:48Z|00001|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:41:48Z|00002|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:41:48Z|00002|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:42:00Z|00003|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:42:00Z|00003|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:42:00Z|00004|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
2024-08-27T14:42:00Z|00004|vconn|ERR|unix:/var/run/openvswitch/s1.snoop: received OpenFlow version 0x05 != expected 01
 cookie=0x0, duration=151.168s, table=0, n_packets=28, n_bytes=2152, priority=0 actions=CONTROLLER:128
```

The flow entries in the table were related to specific network packets matched and managed according to rules set by the OpenFlow controller. These entries included attributes such as priority, idle timeout, hard timeout, and actions for handling packets. The duration field showed how long each entry had been active.

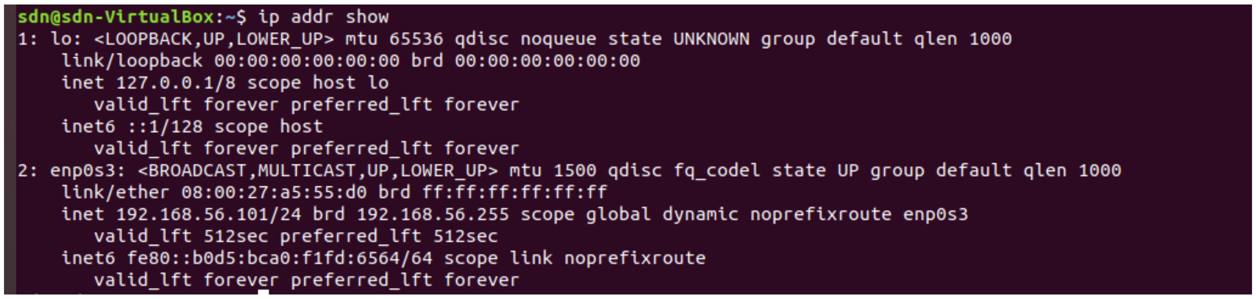
As the experiment continued, snooping was enabled and the system waited for the flow entries to time out. Flow entries in OpenFlow switches have an idle timeout, meaning they are removed if no packets match them within a certain period. After the timeout period, running dpctl dump-flows showed an empty flow table, indicating that all previous entries had expired or been removed.

When pings were initiated between the hosts again (h1 ping h2 and h2 ping h1), new flow entries were created in the table by the OpenFlow switch to handle these packets. This is expected, as the switch dynamically adds new entries for incoming packets that don't match existing flows.

This behavior reflects how OpenFlow and SDN manage flow tables: they start with entries based on network activity, clear them after timeouts, and re-populate the table with new entries as needed.

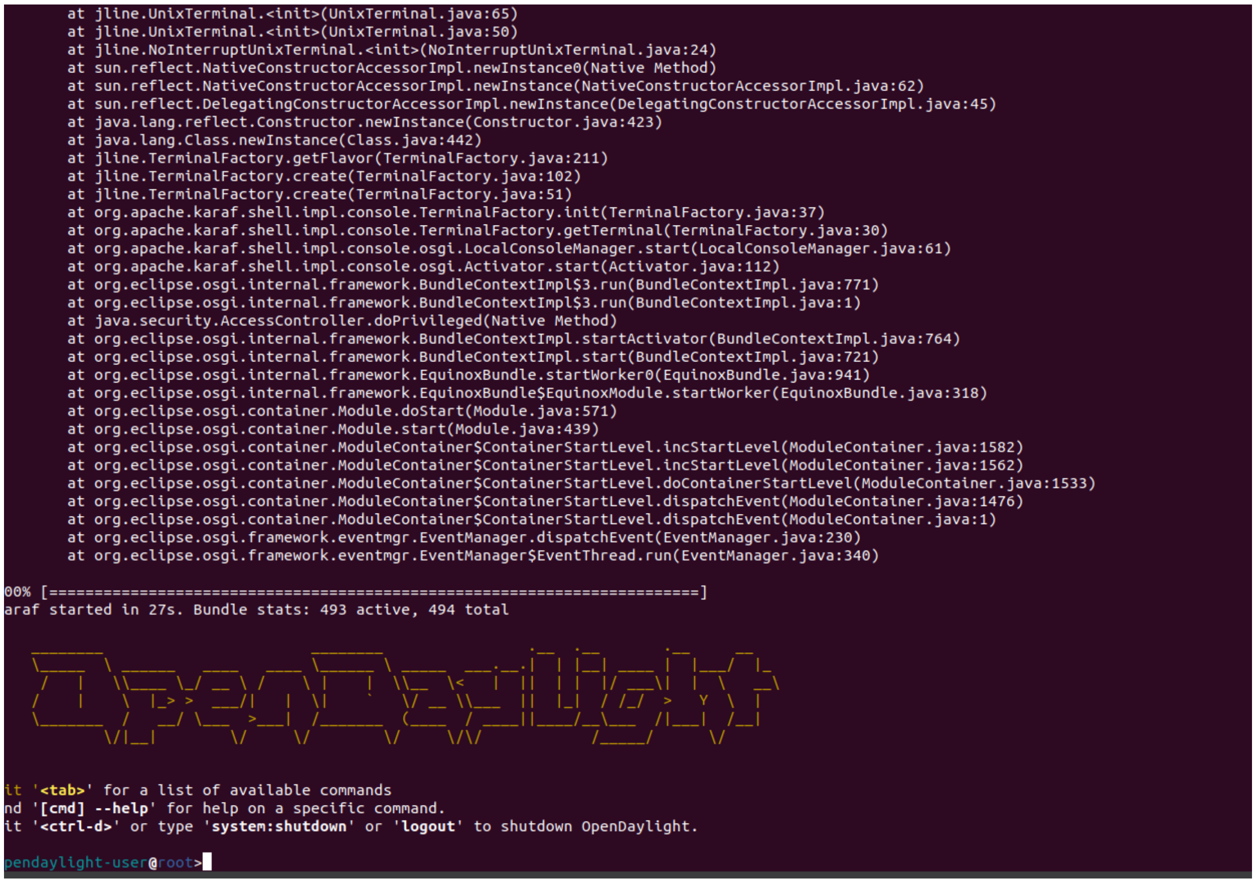
Part – 02

1. **Check the IP address configuration of the Ubuntu VM from :**
   ip addr show

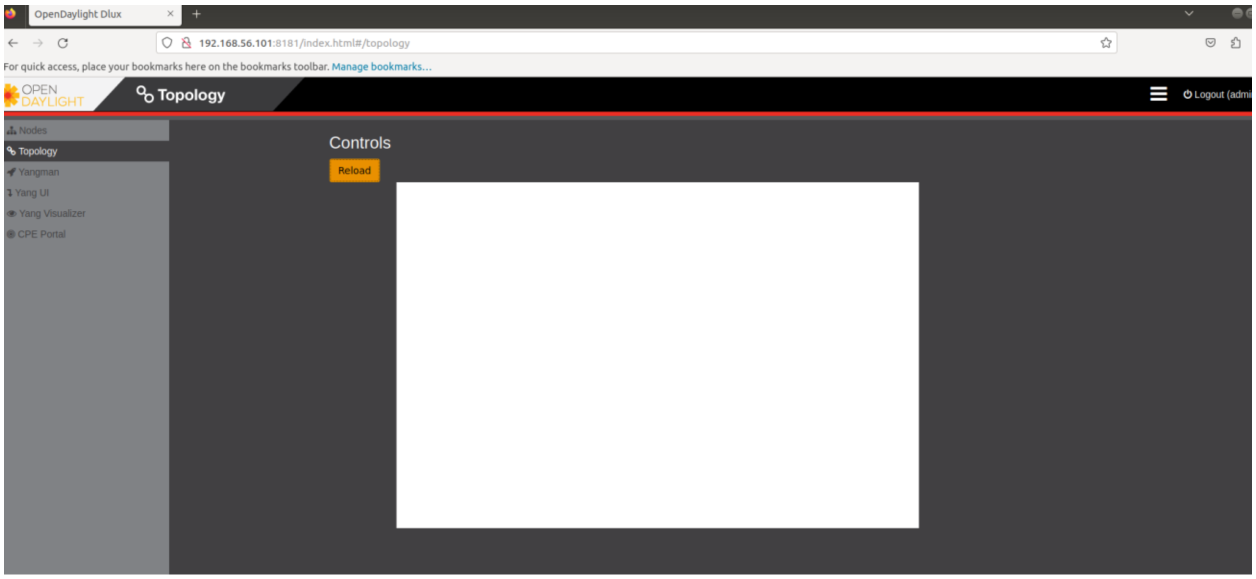


```
sdn@sdn-VirtualBox:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:a5:55:d0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s3
       valid_lft 512sec preferred_lft 512sec
    inet6 fe80::b0d5:bca0:f1fd:6564/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

2. **Open a New terminal window in Ubuntu VM and Run OpenDaylight:**
   sudo ./karaf-0.7.2/bin/karaf



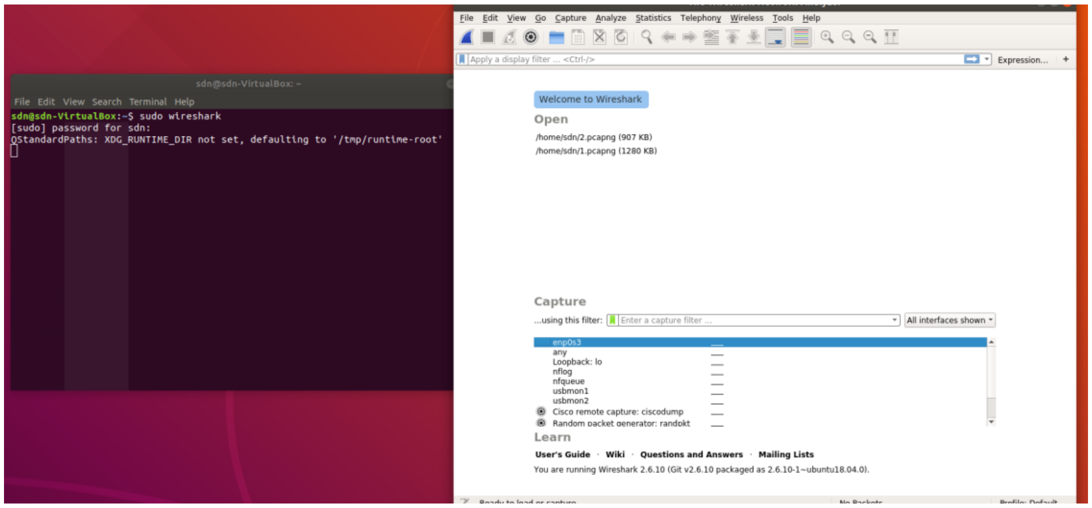
```
        at jline.UnixTerminal.<init>(UnixTerminal.java:65)
        at jline.UnixTerminal.<init>(UnixTerminal.java:50)
        at jline.NoInterruptUnixTerminal.<init>(NoInterruptUnixTerminal.java:24)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at java.lang.Class.newInstance(Class.java:442)
        at jline.TerminalFactory.getFlavor(TerminalFactory.java:211)
        at jline.TerminalFactory.create(TerminalFactory.java:102)
        at jline.TerminalFactory.create(TerminalFactory.java:51)
        at org.apache.karaf.shell.impl.console.TerminalFactory.init(TerminalFactory.java:37)
        at org.apache.karaf.shell.impl.console.TerminalFactory.getTerminal(TerminalFactory.java:30)
        at org.apache.karaf.shell.impl.console.osgi.LocalConsoleManager.start(LocalConsoleManager.java:61)
        at org.apache.karaf.shell.impl.console.osgi.Activator.start(Activator.java:112)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$3.run(BundleContextImpl.java:771)
        at org.eclipse.osgi.internal.framework.BundleContextImpl$3.run(BundleContextImpl.java:1)
        at java.security.AccessController.doPrivileged(Native Method)
        at org.eclipse.osgi.internal.framework.BundleContextImpl.startActivator(BundleContextImpl.java:764)
        at org.eclipse.osgi.internal.framework.BundleContextImpl.start(BundleContextImpl.java:721)
        at org.eclipse.osgi.internal.framework.EquinoxBundle.startWorker0(EquinoxBundle.java:941)
        at org.eclipse.osgi.internal.framework.EquinoxBundle$EquinoxModule.startWorker(EquinoxBundle.java:318)
        at org.eclipse.osgi.container.Module.doStart(Module.java:571)
        at org.eclipse.osgi.container.Module.start(Module.java:439)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.incStartLevel(ModuleContainer.java:1582)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.incStartLevel(ModuleContainer.java:1562)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.doContainerStartLevel(ModuleContainer.java:1533)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.dispatchEvent(ModuleContainer.java:1476)
        at org.eclipse.osgi.container.ModuleContainer$ContainerStartLevel.dispatchEvent(ModuleContainer.java:1)
        at org.eclipse.osgi.framework.eventmgr.EventManager.dispatchEvent(EventManager.java:230)
        at org.eclipse.osgi.framework.eventmgr.EventManager$EventThread.run(EventManager.java:340)

00% [========================================================================]
araf started in 27s. Bundle stats: 493 active, 494 total

   ________                       ________                .__  .__       .__     __
   \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |___/  |_
    /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
   /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
   \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
           \/|__|        \/     \/        \/     \/\/            /_____/      \/


it '<tab>' for a list of available commands
nd '[cmd] --help' for help on a specific command.
it '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

pendaylight-user@root>
```



OpenDaylight Dlux
192.168.56.101:8181/index.html#/topology
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...
OPEN DAYLIGHT
Topology
Logout (admi
Nodes
Topology
Yangman
Yang UI
Yang Visualizer
CPE Portal
Controls
Reload

Part – 03

1. **Start Wireshark in the Ubuntu VM and select interface corresponding to your LAN:**



```
sdn@sdn-VirtualBox:~$ sudo wireshark
[sudo] password for sdn:
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Welcome to Wireshark

Open

/home/sdn/2.pcapng (907 KB)
/home/sdn/1.pcapng (1280 KB)

Capture

...using this filter: Enter a capture filter ... All interfaces shown

enp0s3
any
Loopback: lo
nflog
nfqueue
usbmon1
usbmon2
Cisco remote capture: ciscodump
Random packet generator: randpkt

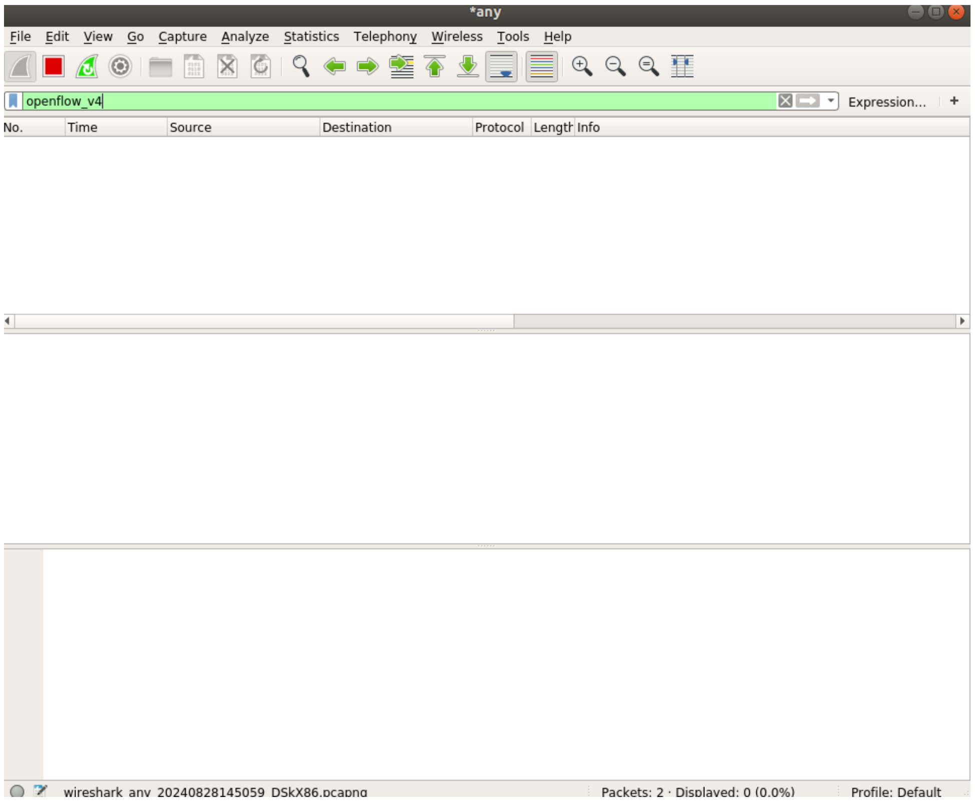
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.6.10 (Git v2.6.10 packaged as 2.6.10-1~ubuntu18.04.0).

File Edit View Search Terminal Help
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

2. **Create a display filter within Wireshark for openflow_v4:**



*any

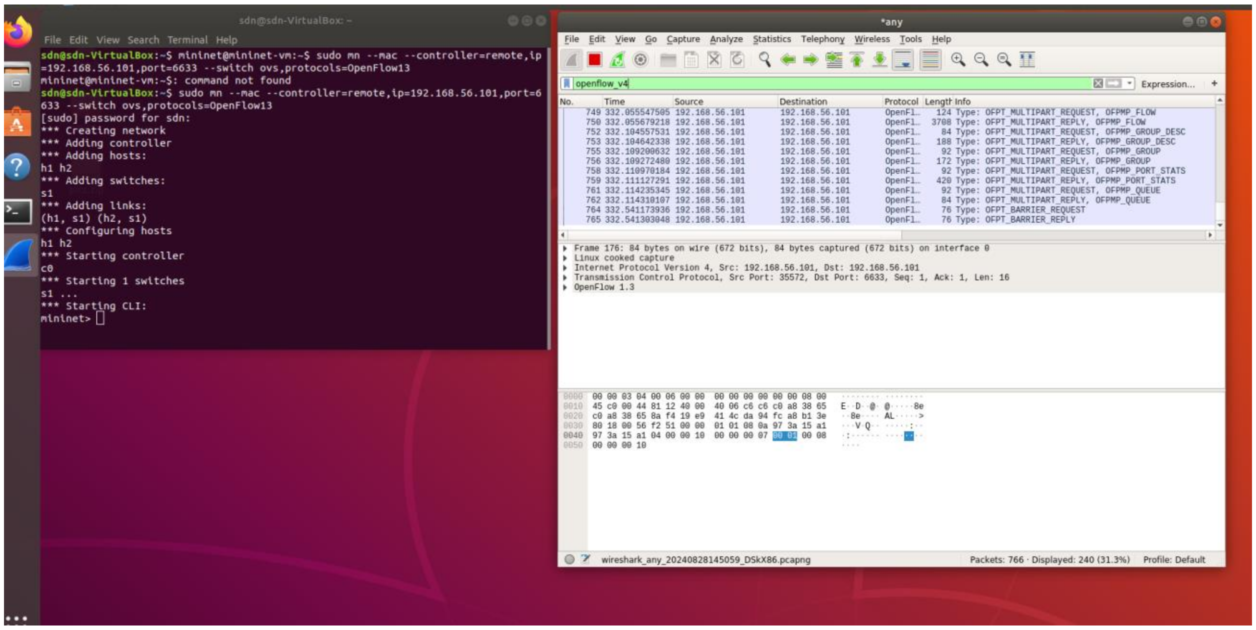
openflow_v4

No.	Time	Source	Destination	Protocol	Length	Info

wireshark_any_20240828145059_DSkX86.pcapng   Packets: 2 · Displayed: 0 (0.0%)   Profile: Default

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

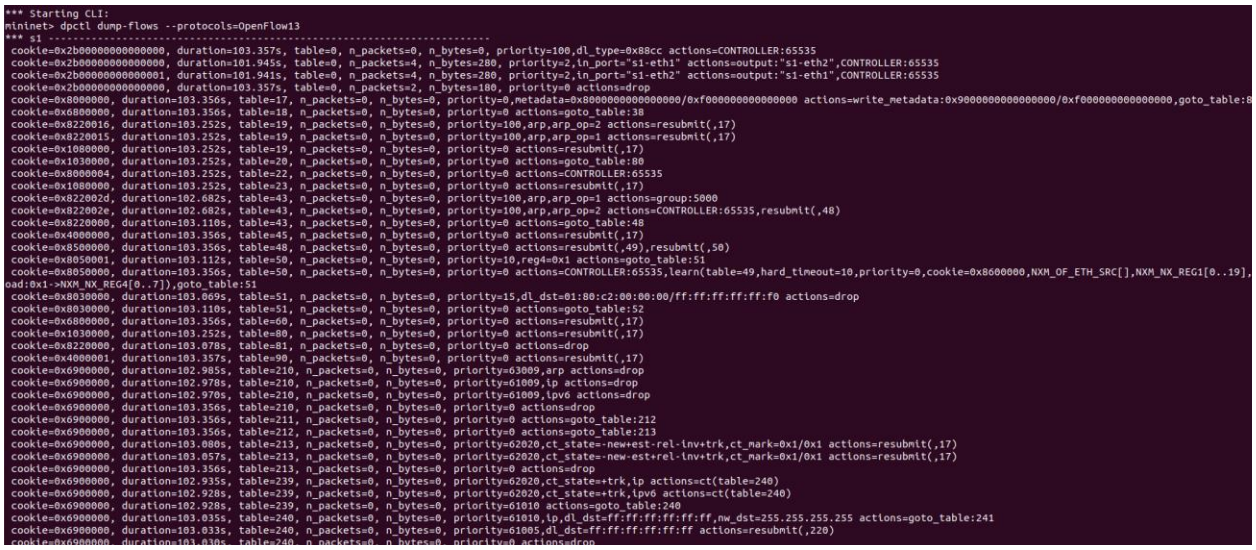
3. **Create a simple network on the Mininet VM:**



```
sdn@sdn-VirtualBox:~$ mininet@mininet-vm:~$ sudo mn --mac --controller=remote,ip
=192.168.56.101,port=6633 --switch ovs,protocols=OpenFlow13
mininet@mininet-vm:~$: command not found
sdn@sdn-VirtualBox:~$ sudo mn --mac --controller=remote,ip=192.168.56.101,port=6
633 --switch ovs,protocols=OpenFlow13
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> 
```

```
openflow_v4
No.  Time           Source          Destination     Protocol Length Info
 749 332.055547505  192.168.56.101  192.168.56.101  OpenFl…   124 Type: OFPT_MULTIPART_REQUEST, OFPMP_FLOW
 750 332.055679218  192.168.56.101  192.168.56.101  OpenFl…  3708 Type: OFPT_MULTIPART_REPLY, OFPMP_FLOW
 752 332.104557531  192.168.56.101  192.168.56.101  OpenFl…    84 Type: OFPT_MULTIPART_REQUEST, OFPMP_GROUP_DESC
 753 332.104642338  192.168.56.101  192.168.56.101  OpenFl…   188 Type: OFPT_MULTIPART_REPLY, OFPMP_GROUP_DESC
 755 332.109200632  192.168.56.101  192.168.56.101  OpenFl…    92 Type: OFPT_MULTIPART_REQUEST, OFPMP_GROUP
 756 332.109272480  192.168.56.101  192.168.56.101  OpenFl…   172 Type: OFPT_MULTIPART_REPLY, OFPMP_GROUP
 758 332.110970184  192.168.56.101  192.168.56.101  OpenFl…    92 Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_STATS
 759 332.111127291  192.168.56.101  192.168.56.101  OpenFl…   420 Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_STATS
 761 332.114235345  192.168.56.101  192.168.56.101  OpenFl…    92 Type: OFPT_MULTIPART_REQUEST, OFPMP_QUEUE
 762 332.114310107  192.168.56.101  192.168.56.101  OpenFl…    84 Type: OFPT_MULTIPART_REPLY, OFPMP_QUEUE
 764 332.541173936  192.168.56.101  192.168.56.101  OpenFl…    76 Type: OFPT_BARRIER_REQUEST
 765 332.541303048  192.168.56.101  192.168.56.101  OpenFl…    76 Type: OFPT_BARRIER_REPLY

▸ Frame 176: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.101
▸ Transmission Control Protocol, Src Port: 35572, Dst Port: 6633, Seq: 1, Ack: 1, Len: 16
▸ OpenFlow 1.3

0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 c0 00 44 81 12 40 00  40 06 c6 c6 c0 a8 38 65   E··D··@· @·····8e
0020  c0 a8 38 65 8a f4 19 e9  41 4c da 94 fc a8 b1 3e   ··8e···· AL·····>
0030  80 18 00 56 f2 51 00 00  01 01 08 0a 97 3a 15 a1   ···V·Q·· ·····:··
0040  97 3a 15 a1 04 00 00 10  00 00 00 07 00 01 00 08   ·:······ ········
0050  00 00 00 10                                        ····

wireshark_any_20240828145059_DSkX86.pcapng     Packets: 766 · Displayed: 240 (31.3%)     Profile: Default
```

4. **Check the flow table of the switch:**
mininet> dpctl dump-flows --protocols=OpenFlow13



```
*** Starting CLI:
mininet> dpctl dump-flows --protocols=OpenFlow13
*** s1 ------------------------------------------------------------------------
 cookie=0x2b00000000000000, duration=103.357s, table=0, n_packets=0, n_bytes=0, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000000, duration=101.945s, table=0, n_packets=4, n_bytes=280, priority=2,in_port="s1-eth1" actions=output:"s1-eth2",CONTROLLER:65535
 cookie=0x2b00000000000001, duration=101.941s, table=0, n_packets=4, n_bytes=280, priority=2,in_port="s1-eth2" actions=output:"s1-eth1",CONTROLLER:65535
 cookie=0x2b00000000000000, duration=103.357s, table=0, n_packets=2, n_bytes=180, priority=0 actions=drop
 cookie=0x8000000, duration=103.356s, table=17, n_packets=0, n_bytes=0, priority=0,metadata=0x8000000000000000/0xf000000000000000 actions=write_metadata:0x9000000000000000/0xf000000000000000,goto_table:8
 cookie=0x6800000, duration=103.356s, table=18, n_packets=0, n_bytes=0, priority=0 actions=goto_table:38
 cookie=0x8220016, duration=103.252s, table=19, n_packets=0, n_bytes=0, priority=100,arp,arp_op=2 actions=resubmit(,17)
 cookie=0x8220015, duration=103.252s, table=19, n_packets=0, n_bytes=0, priority=100,arp,arp_op=1 actions=resubmit(,17)
 cookie=0x1080000, duration=103.252s, table=19, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x1030000, duration=103.252s, table=20, n_packets=0, n_bytes=0, priority=0 actions=goto_table:80
 cookie=0x8000004, duration=103.252s, table=22, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535
 cookie=0x1080000, duration=103.252s, table=23, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x822002d, duration=102.682s, table=43, n_packets=0, n_bytes=0, priority=100,arp,arp_op=1 actions=group:5000
 cookie=0x822002e, duration=102.682s, table=43, n_packets=0, n_bytes=0, priority=100,arp,arp_op=2 actions=CONTROLLER:65535,resubmit(,48)
 cookie=0x8220000, duration=103.110s, table=43, n_packets=0, n_bytes=0, priority=0 actions=goto_table:48
 cookie=0x4000000, duration=103.356s, table=45, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x8500000, duration=103.356s, table=48, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,49),resubmit(,50)
 cookie=0x8050001, duration=103.112s, table=50, n_packets=0, n_bytes=0, priority=10,reg4=0x1 actions=goto_table:51
 cookie=0x8050000, duration=103.356s, table=50, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:65535,learn(table=49,hard_timeout=10,priority=0,cookie=0x8600000,NXM_OF_ETH_SRC[],NXM_NX_REG1[0..19],
oad:0x1->NXM_NX_REG4[0..7]),goto_table:51
 cookie=0x8030000, duration=103.069s, table=51, n_packets=0, n_bytes=0, priority=15,dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0 actions=drop
 cookie=0x8030000, duration=103.110s, table=51, n_packets=0, n_bytes=0, priority=0 actions=goto_table:52
 cookie=0x6800000, duration=103.356s, table=60, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x1030000, duration=103.252s, table=80, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x8220000, duration=103.078s, table=81, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x4000001, duration=103.357s, table=90, n_packets=0, n_bytes=0, priority=0 actions=resubmit(,17)
 cookie=0x6900000, duration=102.985s, table=210, n_packets=0, n_bytes=0, priority=63009,arp actions=drop
 cookie=0x6900000, duration=102.978s, table=210, n_packets=0, n_bytes=0, priority=61009,ip actions=drop
 cookie=0x6900000, duration=102.970s, table=210, n_packets=0, n_bytes=0, priority=61009,ipv6 actions=drop
 cookie=0x6900000, duration=103.356s, table=210, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x6900000, duration=103.356s, table=211, n_packets=0, n_bytes=0, priority=0 actions=goto_table:212
 cookie=0x6900000, duration=103.356s, table=212, n_packets=0, n_bytes=0, priority=0 actions=goto_table:213
 cookie=0x6900000, duration=103.080s, table=213, n_packets=0, n_bytes=0, priority=62020,ct_state=-new+est-rel-inv+trk,ct_mark=0x1/0x1 actions=resubmit(,17)
 cookie=0x6900000, duration=103.057s, table=213, n_packets=0, n_bytes=0, priority=62020,ct_state=-new-est+rel-inv+trk,ct_mark=0x1/0x1 actions=resubmit(,17)
 cookie=0x6900000, duration=103.356s, table=213, n_packets=0, n_bytes=0, priority=0 actions=drop
 cookie=0x6900000, duration=102.935s, table=239, n_packets=0, n_bytes=0, priority=62020,ct_state=+trk,ip actions=ct(table=240)
 cookie=0x6900000, duration=102.928s, table=239, n_packets=0, n_bytes=0, priority=62020,ct_state=+trk,ipv6 actions=ct(table=240)
 cookie=0x6900000, duration=102.928s, table=239, n_packets=0, n_bytes=0, priority=61010 actions=goto_table:240
 cookie=0x6900000, duration=103.035s, table=240, n_packets=0, n_bytes=0, priority=61010,ip,dl_dst=ff:ff:ff:ff:ff:ff,nw_dst=255.255.255.255 actions=goto_table:241
 cookie=0x6900000, duration=103.033s, table=240, n_packets=0, n_bytes=0, priority=61005,dl_dst=ff:ff:ff:ff:ff:ff actions=resubmit(,220)
 cookie=0x6900000, duration=103.030s, table=240, n_packets=0, n_bytes=0, priority=0 actions=drop
```

1. LLDP Packets:
   - Entry: Handles LLDP packets (dl_type=0x88cc) used for network topology discovery.
   - Priority: 100 (high).
   - Action: Forward to the controller.

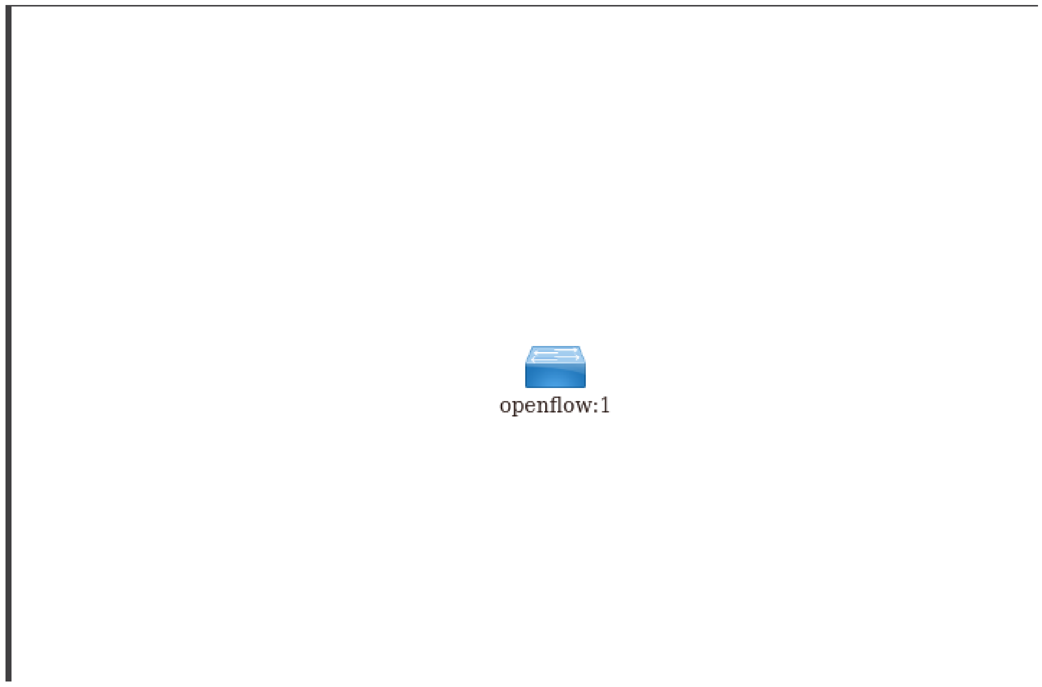
2. Host-to-Host Traffic:

- Entry 1: Matches packets from port s1-eth1 (host h1) and forwards them to s1-eth2 (host h2) and the controller.
- Entry 2: Matches packets from port s1-eth2 (host h2) and forwards them to s1-eth1 and the controller.

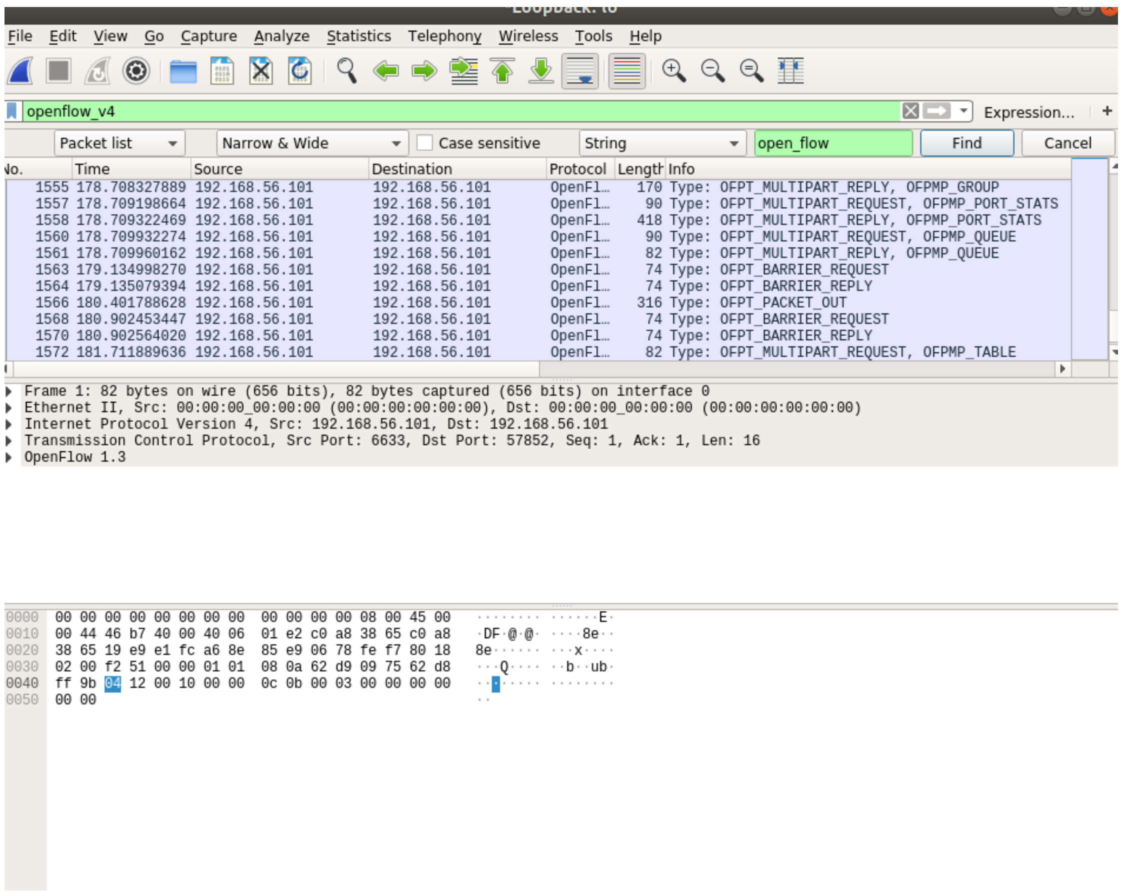
3. Default Drop Rule:
   - Entry: Lowest priority (0).
   - Action: Drops any packets that do not match higher-priority rules.

This setup handles LLDP for network discovery, manages traffic between two hosts, and discards unmatched packets.

5. **Open up OpenDaylight GUI:**
   http://192.168.56.101:8181/index.html



6. **Wireshark FLOW_MOD:**


Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…
Find
Cancel

openflow_v4

Packet list | Narrow & Wide | Case sensitive | String | open_flow

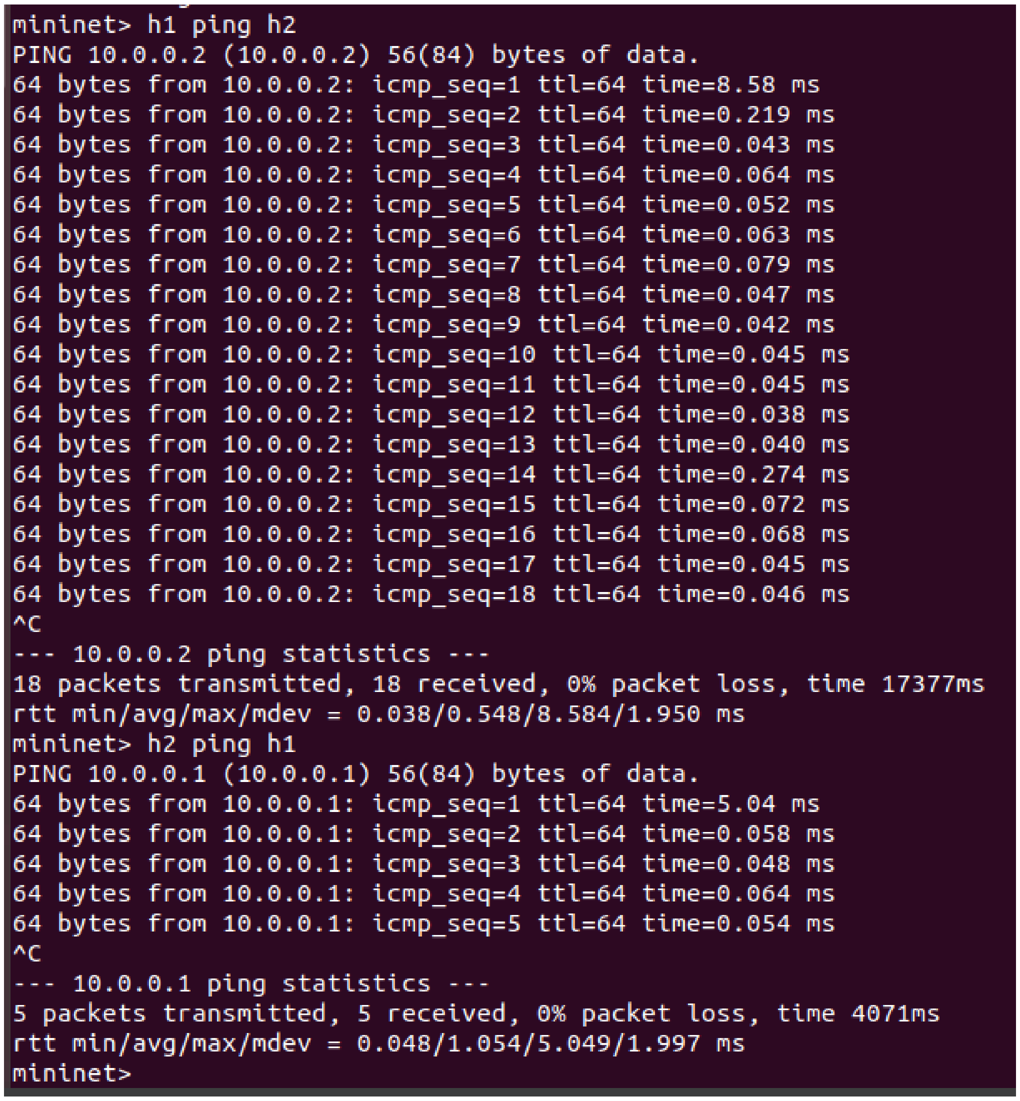
```
No.  Time          Source          Destination     Protocol Length Info
 1555 178.708327889 192.168.56.101 192.168.56.101 OpenFl…  170 Type: OFPT_MULTIPART_REPLY, OFPMP_GROUP
 1557 178.709198664 192.168.56.101 192.168.56.101 OpenFl…   90 Type: OFPT_MULTIPART_REQUEST, OFPMP_PORT_STATS
 1558 178.709322469 192.168.56.101 192.168.56.101 OpenFl…  418 Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_STATS
 1560 178.709932274 192.168.56.101 192.168.56.101 OpenFl…   90 Type: OFPT_MULTIPART_REQUEST, OFPMP_QUEUE
 1561 178.709960162 192.168.56.101 192.168.56.101 OpenFl…   82 Type: OFPT_MULTIPART_REPLY, OFPMP_QUEUE
 1563 179.134998270 192.168.56.101 192.168.56.101 OpenFl…   74 Type: OFPT_BARRIER_REQUEST
 1564 179.135079394 192.168.56.101 192.168.56.101 OpenFl…   74 Type: OFPT_BARRIER_REPLY
 1566 180.401788628 192.168.56.101 192.168.56.101 OpenFl…  316 Type: OFPT_PACKET_OUT
 1568 180.902453447 192.168.56.101 192.168.56.101 OpenFl…   74 Type: OFPT_BARRIER_REQUEST
 1570 180.902564020 192.168.56.101 192.168.56.101 OpenFl…   74 Type: OFPT_BARRIER_REPLY
 1572 181.711889636 192.168.56.101 192.168.56.101 OpenFl…   82 Type: OFPT_MULTIPART_REQUEST, OFPMP_TABLE
```

```
▸ Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 192.168.56.101, Dst: 192.168.56.101
▸ Transmission Control Protocol, Src Port: 6633, Dst Port: 57852, Seq: 1, Ack: 1, Len: 16
▸ OpenFlow 1.3
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 44 46 b7 40 00 40 06  01 e2 c0 a8 38 65 c0 a8   ·DF·@·@· ····8e··
0020  38 65 19 e9 e1 fc a6 8e  85 e9 06 78 fe f7 80 18   8e······ ···x····
0030  02 00 f2 51 00 00 01 01  08 0a 62 d9 09 75 62 d8   ···Q···· ··b··ub·
0040  ff 9b 04 12 00 10 00 00  0c 0b 00 03 00 00 00 00   ········ ········
0050  00 00                                              ··
```

7. **Re-start the Wireshark capture:**

8. **Attempt pinging between h1 and h2:**



```
mininet> h1 ping 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=8.58 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.219 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.079 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.038 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.040 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.274 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.046 ms
^C
--- 10.0.0.2 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17377ms
rtt min/avg/max/mdev = 0.038/0.548/8.584/1.950 ms
mininet> h2 ping 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=5.04 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.064 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.054 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4071ms
rtt min/avg/max/mdev = 0.048/1.054/5.049/1.997 ms
mininet>
```

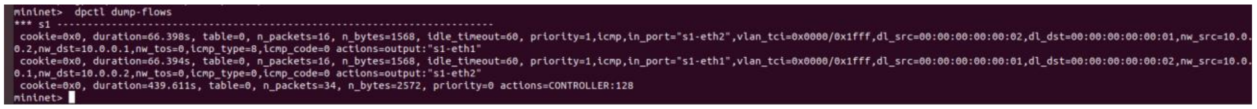
The successful ping test between h1 and h2 indicates that the OpenFlow switch has been correctly configured. The flow table entries in the switch, as detailed in the previous flow table dump, played a crucial role in this:

Traffic from h1 to h2: A specific flow entry managed traffic from s1-eth1 (connected to h1) to s1-eth2 (connected to h2). This entry ensured that packets were properly forwarded from h1 to h2.

Traffic from h2 to h1: Another flow entry handled traffic from s1-eth2 back to s1-eth1, allowing the return packets to travel from h2 to h1.

These entries effectively routed the ICMP packets between the two hosts, confirming that the network is set up correctly. The successful pings demonstrate that the OpenFlow switch is efficiently managing the traffic flow between h1 and h2 as expected.

9. **Check the flow table of the switch again:**
   mininet> dpctl dump-flows



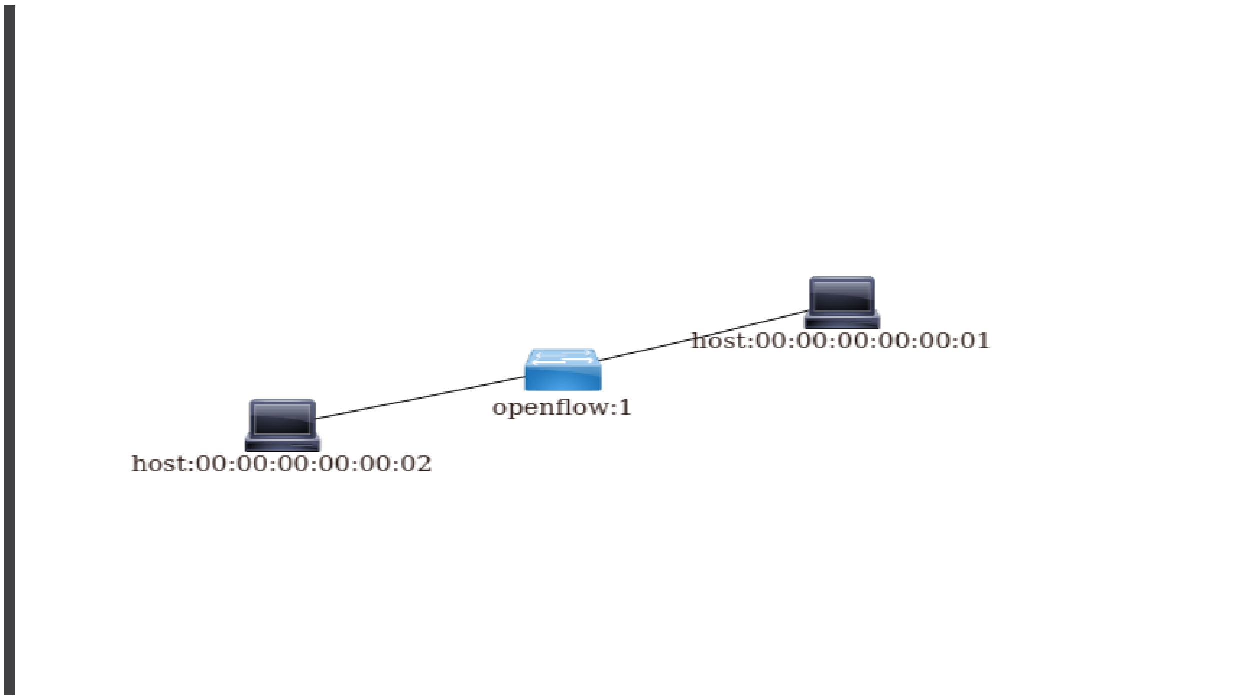
```
mininet>  dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=66.398s, table=0, n_packets=16, n_bytes=1568, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,nw_src=10.0.
0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth1"
 cookie=0x0, duration=66.394s, table=0, n_packets=16, n_bytes=1568, idle_timeout=60, priority=1,icmp,in_port="s1-eth1",vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,nw_src=10.0.
0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth2"
 cookie=0x0, duration=439.611s, table=0, n_packets=34, n_bytes=2572, priority=0 actions=CONTROLLER:128
mininet> 
```

The successful ping tests between h1 and h2 indicate that the OpenFlow switch is properly configured. This can be directly linked to the flow table entries observed in the previous dump:

- Forwarding Traffic: There was a flow entry in the switch that managed traffic from s1-eth1 (which connects to h1) to s1-eth2 (which connects to h2), allowing packets to travel from h1 to h2.
- Handling Responses: Another flow entry was responsible for managing traffic from s1-eth2 back to s1-eth1, ensuring that the ping responses from h2 were correctly routed back to h1.

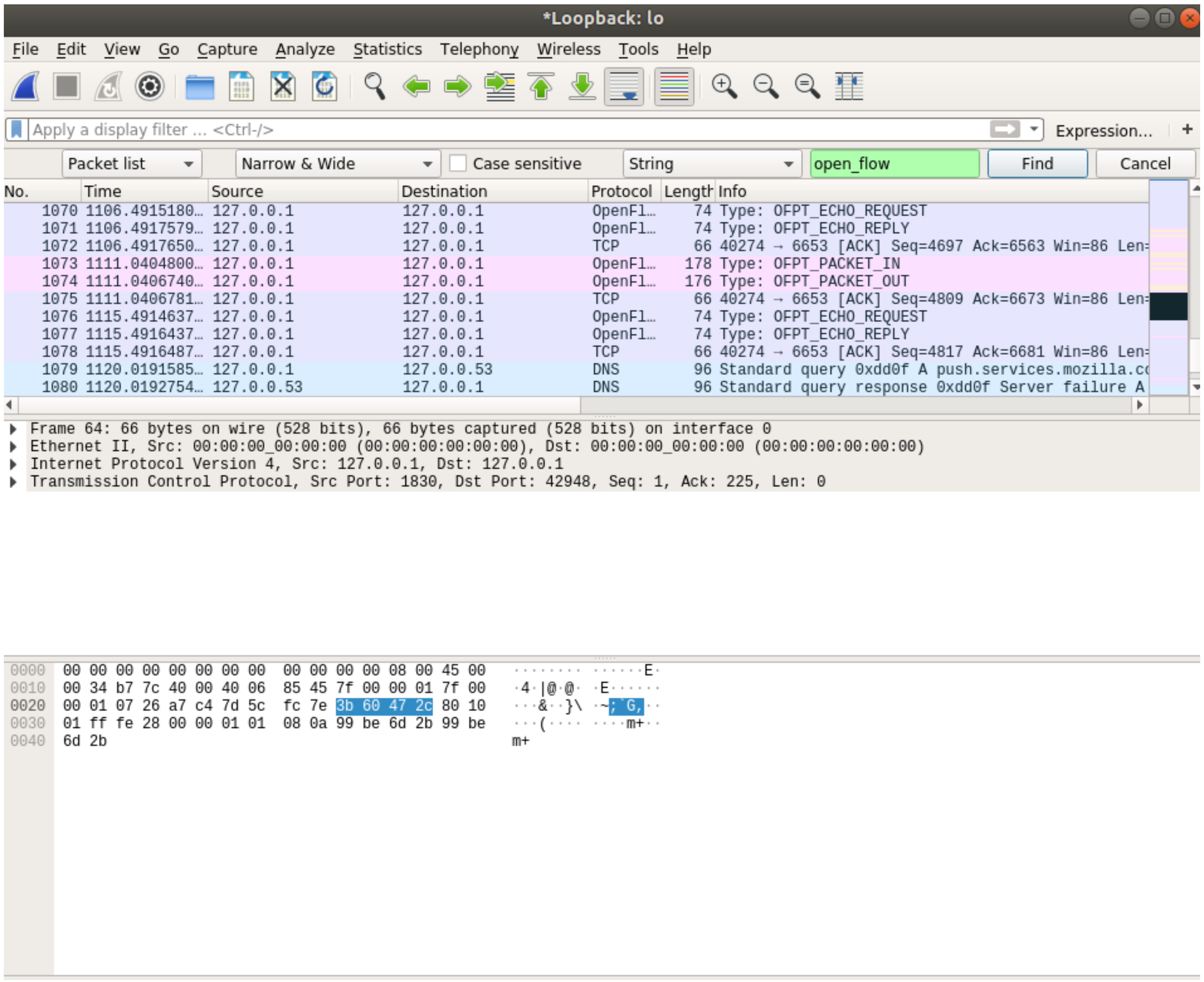
These specific flow entries were crucial in routing ICMP packets correctly between h1 and h2 and ensuring that replies were accurately delivered. The successful pings confirm that the OpenFlow switch is operating as expected and effectively handling the traffic between the hosts.

10. **Open up OpenDaylight GUI again:**



Right now this shows hosts in the topology.

11. Wireshark packet capture:


*Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Packet list
Narrow & Wide
Case sensitive
String
open_flow
Find
Cancel

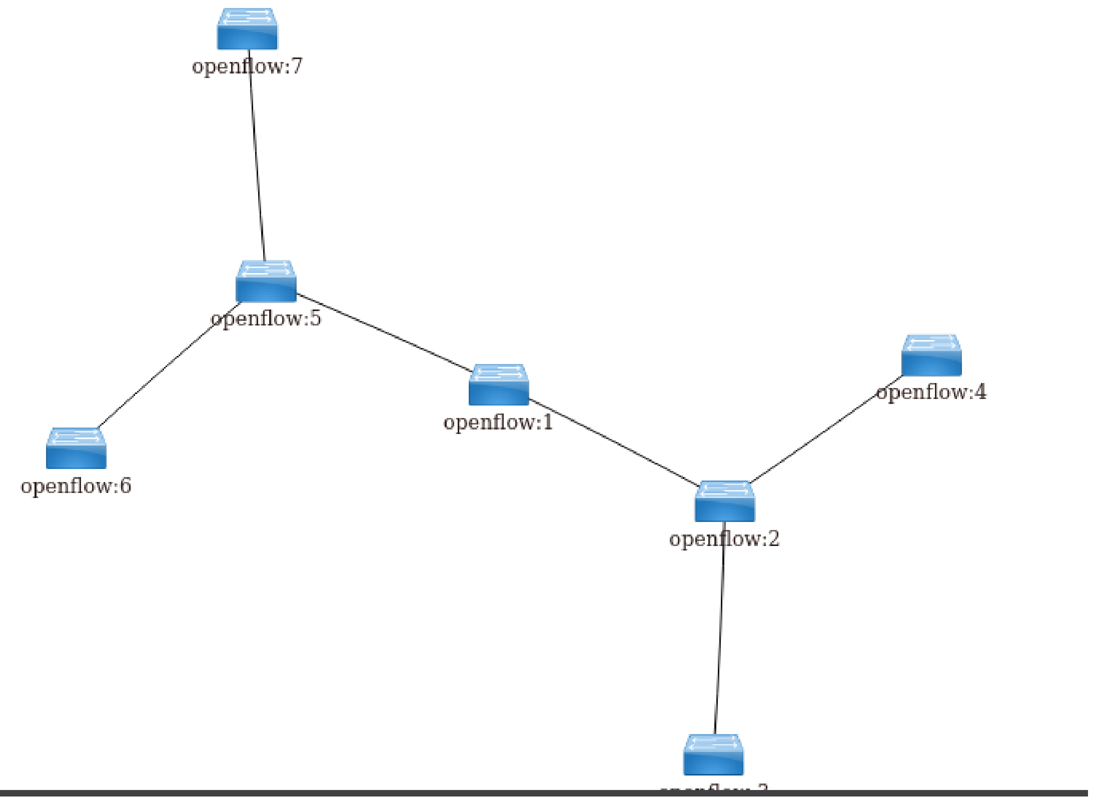
```
No.  Time            Source      Destination  Protocol Length Info
1070 1106.4915180…  127.0.0.1   127.0.0.1    OpenFl…   74 Type: OFPT_ECHO_REQUEST
1071 1106.4917579…  127.0.0.1   127.0.0.1    OpenFl…   74 Type: OFPT_ECHO_REPLY
1072 1106.4917650…  127.0.0.1   127.0.0.1    TCP       66 40274 → 6653 [ACK] Seq=4697 Ack=6563 Win=86 Len=
1073 1111.0404800…  127.0.0.1   127.0.0.1    OpenFl…  178 Type: OFPT_PACKET_IN
1074 1111.0406740…  127.0.0.1   127.0.0.1    OpenFl…  176 Type: OFPT_PACKET_OUT
1075 1111.0406781…  127.0.0.1   127.0.0.1    TCP       66 40274 → 6653 [ACK] Seq=4809 Ack=6673 Win=86 Len=
1076 1115.4914637…  127.0.0.1   127.0.0.1    OpenFl…   74 Type: OFPT_ECHO_REQUEST
1077 1115.4916437…  127.0.0.1   127.0.0.1    OpenFl…   74 Type: OFPT_ECHO_REPLY
1078 1115.4916487…  127.0.0.1   127.0.0.1    TCP       66 40274 → 6653 [ACK] Seq=4817 Ack=6681 Win=86 Len=
1079 1120.0191585…  127.0.0.1   127.0.0.53   DNS       96 Standard query 0xdd0f A push.services.mozilla.co
1080 1120.0192754…  127.0.0.53  127.0.0.1    DNS       96 Standard query response 0xdd0f Server failure A
```

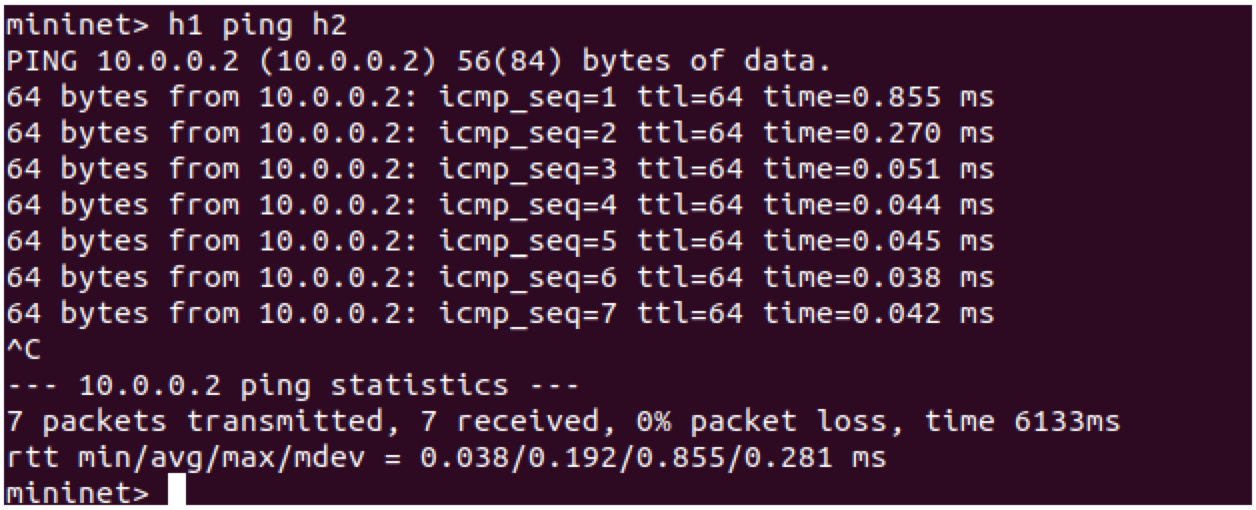
```
▸ Frame 64: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 1830, Dst Port: 42948, Seq: 1, Ack: 225, Len: 0
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 34 b7 7c 40 00 40 06  85 45 7f 00 00 01 7f 00   ·4·|@·@· ·E······
0020  00 01 07 26 a7 c4 7d 5c  fc 7e 3b 60 47 2c 80 10   ···&··}\ ·~;`G,··
0030  01 ff fe 28 00 00 01 01  08 0a 99 be 6d 2b 99 be   ···(···· ····m+··
0040  6d 2b                                              m+
```

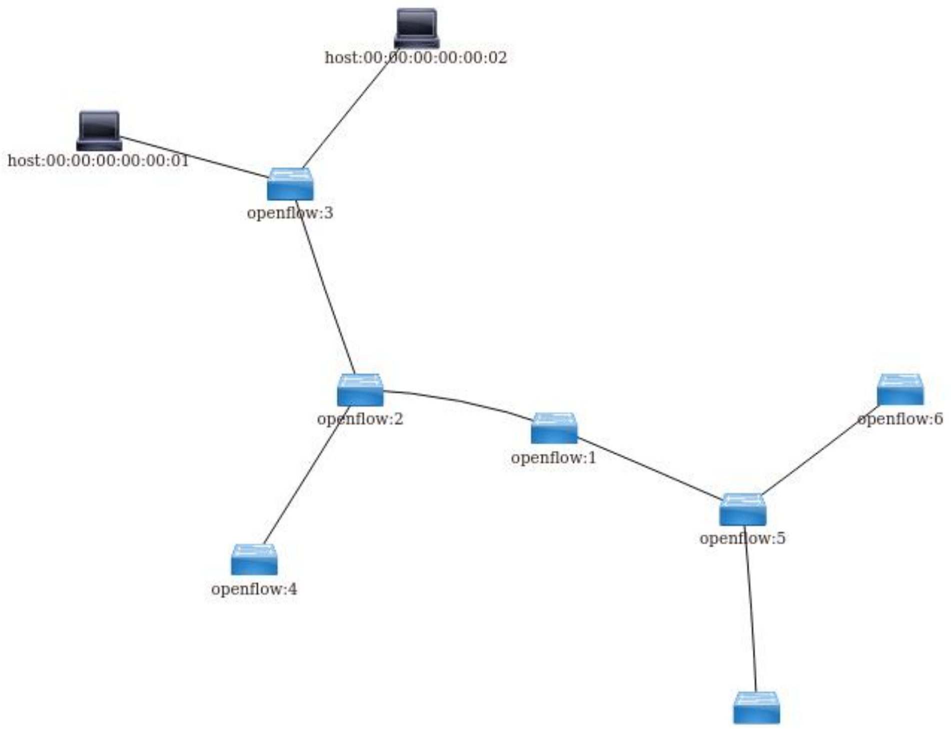
sudo mn --mac --topo=tree,3 --controller=remote,ip=192.168.56.101,port=6633 --switch ovs,protocols=OpenFlow13

Topology 02:



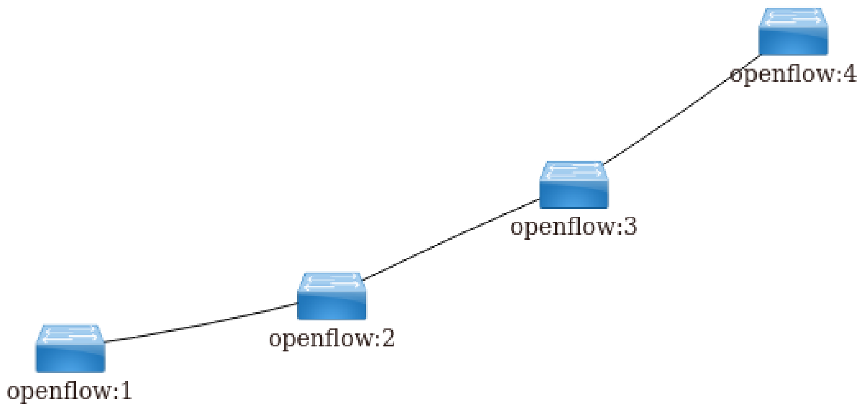


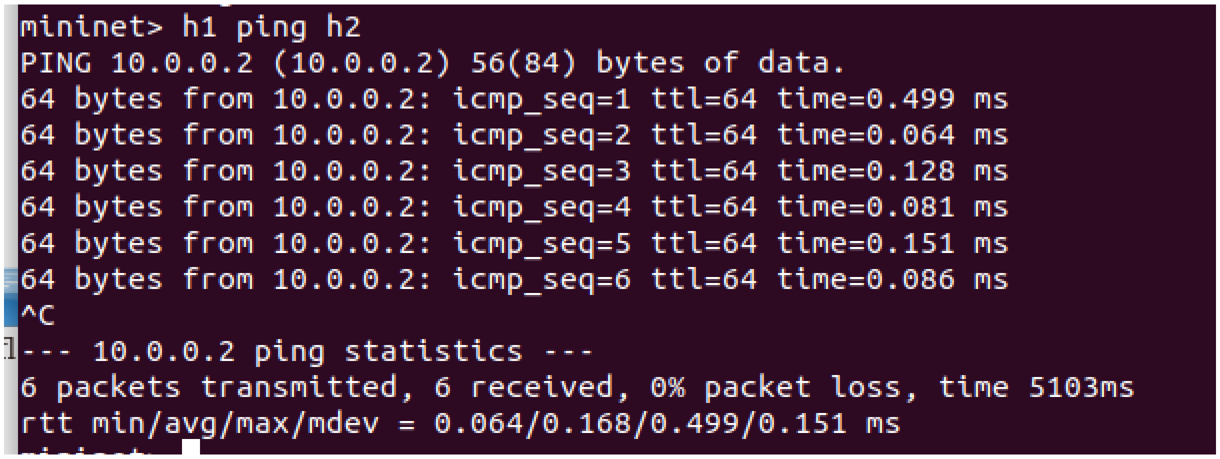
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.855 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.270 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.045 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.038 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.042 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6133ms
rtt min/avg/max/mdev = 0.038/0.192/0.855/0.281 ms
mininet> 
```



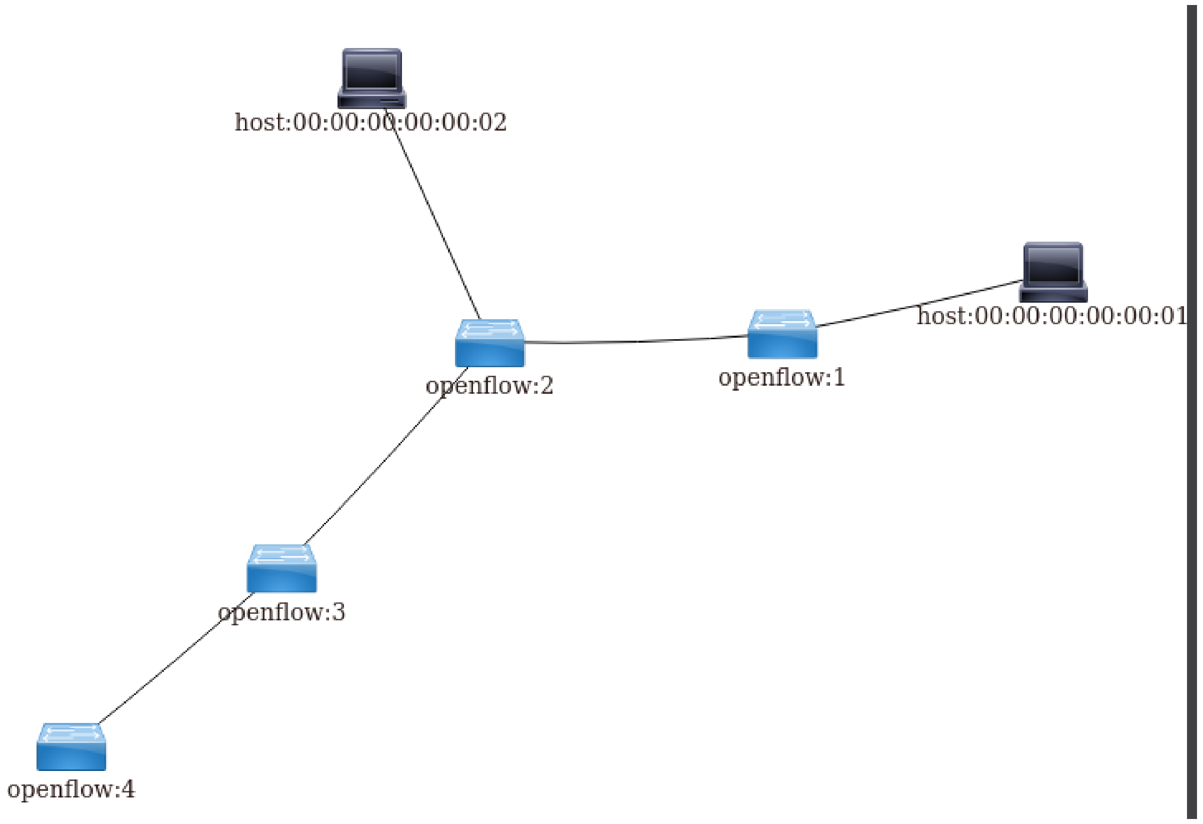
Topology 03:

```
sudo mn --mac --topo=linear,4 --controller=remote,ip=192.168.56.101,port=6633 --switch
ovs,protocols=OpenFlow13
```





```
mininet> h1 ping 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=0.499 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.128 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.151 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.086 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5103ms
rtt min/avg/max/mdev = 0.064/0.168/0.499/0.151 ms
```


host:00:00:00:00:00:02
host:00:00:00:00:00:01
openflow:2
openflow:1
openflow:3
openflow:4