Part 1

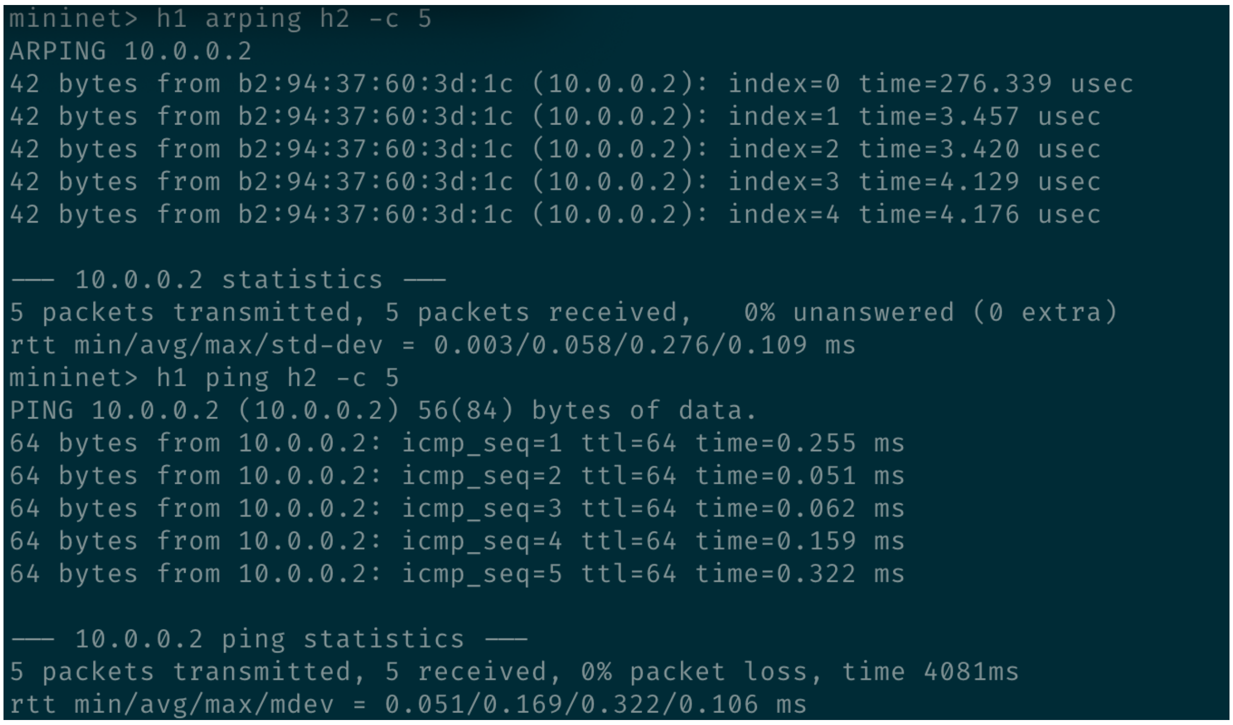
There are 2 distinct `OFPT_FLOW_MOD` headers during the experiment.

Match fields	Actions	Timeout Values

IN_PORT=1
ETH_DST=22:90:3d:2f:19:39
ETH_SRC=ca:93:59:74:3c:4c
	OUTPUT_PORT=2	0

ETH_TYPE=0x0806	OUTPUT_PORT=OFPP_CONTROLLER	0

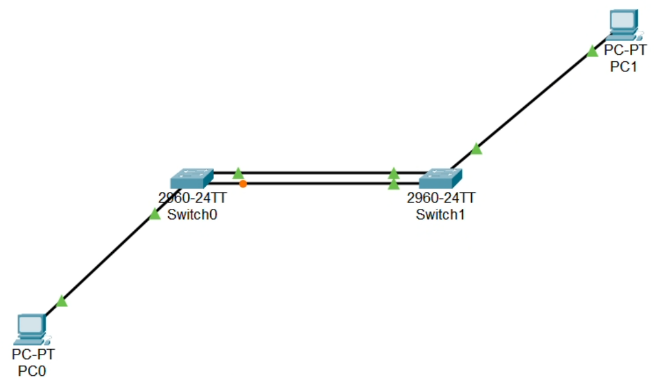
Part 2



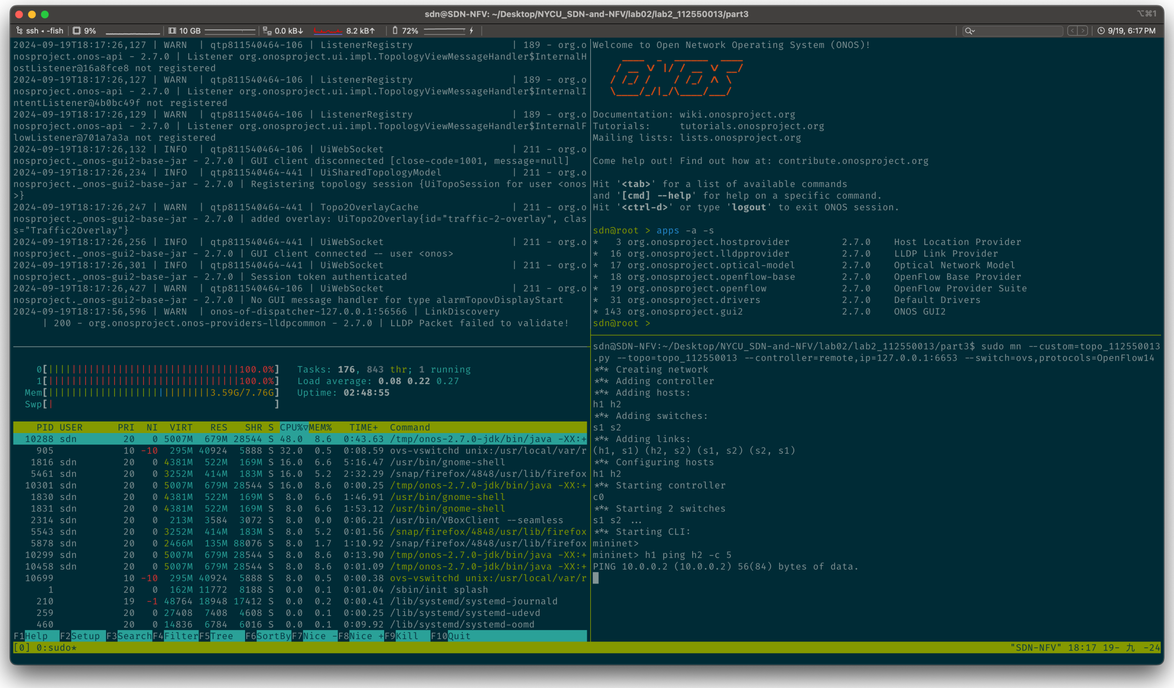
```
mininet> h1 arping h2 -c 5
ARPING 10.0.0.2
42 bytes from b2:94:37:60:3d:1c (10.0.0.2): index=0 time=276.339 usec
42 bytes from b2:94:37:60:3d:1c (10.0.0.2): index=1 time=3.457 usec
42 bytes from b2:94:37:60:3d:1c (10.0.0.2): index=2 time=3.420 usec
42 bytes from b2:94:37:60:3d:1c (10.0.0.2): index=3 time=4.129 usec
42 bytes from b2:94:37:60:3d:1c (10.0.0.2): index=4 time=4.176 usec

--- 10.0.0.2 statistics ---
5 packets transmitted, 5 packets received,   0% unanswered (0 extra)
rtt min/avg/max/std-dev = 0.003/0.058/0.276/0.109 ms
mininet> h1 ping h2 -c 5
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=0.255 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.159 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.322 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4081ms
rtt min/avg/max/mdev = 0.051/0.169/0.322/0.106 ms
```

Part 3

Below is the topology of the network that would cause a broadcast storm.



In this topology, there is a loop between switch 1 and switch 2. When a broadcast packet is sent from host 1, it will be forwarded to switch 1, then switch 2, and finally back to switch 1. This will cause the broadcast packet to be forwarded infinitely between switch 1 and switch 2, creating a broadcast storm, which makes the CPU usage of the virtual machine to be 100%.



```
sdn@SDN-NFV: ~/Desktop/NYCU_SDN-and-NFV/lab02/lab2_112550013/part3
ssh ‹ -fish   9%   10 GB   0.0 kB↓   8.2 kB↑   72%                9/19, 6:17 PM
```

```
2024-09-19T18:17:26,127 | WARN  | qtp811540464-106 | ListenerRegistry                 | 189 - org.o
nosproject.onos-api - 2.7.0 | Listener org.onosproject.ui.impl.TopologyViewMessageHandler$InternalH
ostListener@16a8fce8 not registered
2024-09-19T18:17:26,127 | WARN  | qtp811540464-106 | ListenerRegistry                 | 189 - org.o
nosproject.onos-api - 2.7.0 | Listener org.onosproject.ui.impl.TopologyViewMessageHandler$InternalI
ntentListener@4b0bc49f not registered
2024-09-19T18:17:26,129 | WARN  | qtp811540464-106 | ListenerRegistry                 | 189 - org.o
nosproject.onos-api - 2.7.0 | Listener org.onosproject.ui.impl.TopologyViewMessageHandler$InternalF
lowListener@701a7a3a not registered
2024-09-19T18:17:26,132 | INFO  | qtp811540464-106 | UiWebSocket                      | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | GUI client disconnected [close-code=1001, message=null]
2024-09-19T18:17:26,234 | INFO  | qtp811540464-441 | UiSharedTopologyModel            | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | Registering topology session {UiTopoSession for user <onos
>}
2024-09-19T18:17:26,247 | WARN  | qtp811540464-441 | Topo2OverlayCache                | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | added overlay: UiTopo2Overlay{id="traffic-2-overlay", clas
s="Traffic2Overlay"}
2024-09-19T18:17:26,256 | INFO  | qtp811540464-441 | UiWebSocket                      | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | GUI client connected -- user <onos>
2024-09-19T18:17:26,301 | INFO  | qtp811540464-441 | UiWebSocket                      | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | Session token authenticated
2024-09-19T18:17:26,427 | WARN  | qtp811540464-106 | UiWebSocket                      | 211 - org.o
nosproject._onos-gui2-base-jar - 2.7.0 | No GUI message handler for type alarmTopovDisplayStart
2024-09-19T18:17:56,596 | WARN  | onos-of-dispatcher-127.0.0.1:56566 | LinkDiscovery
     | 200 - org.onosproject.onos-providers-lldpcommon - 2.7.0 | LLDP Packet failed to validate!
```

```
    0[||||||||||||||||||||||||||||||||100.0%]   Tasks: 176, 843 thr; 1 running
    1[||||||||||||||||||||||||||||||||100.0%]   Load average: 0.08 0.22 0.27
  Mem[|||||||||||||||||||||||||||3.59G/7.76G]   Uptime: 02:48:55
  Swp[|                                     ]

    PID USER       PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
  10288 sdn         20   0 5007M  679M 28544 S 48.0  8.6  0:43.63 /tmp/onos-2.7.0-jdk/bin/java -XX:+
    905             10 -10  295M 40924  5888 S 32.0  0.5  0:08.59 ovs-vswitchd unix:/usr/local/var/r
   1816 sdn         20   0 4381M  522M  169M S 16.0  6.6  5:16.47 /usr/bin/gnome-shell
   5461 sdn         20   0 3252M  414M  183M S 16.0  5.2  2:32.29 /snap/firefox/4848/usr/lib/firefox
  10301 sdn         20   0 5007M  679M 28544 S 16.0  8.6  0:00.25 /tmp/onos-2.7.0-jdk/bin/java -XX:+
   1830 sdn         20   0 4381M  522M  169M S  8.0  6.6  1:46.91 /usr/bin/gnome-shell
   1831 sdn         20   0 4381M  522M  169M S  8.0  6.6  1:53.12 /usr/bin/gnome-shell
   2314 sdn         20   0  213M  3584  3072 S  8.0  0.0  0:06.21 /usr/bin/VBoxClient --seamless
   5543 sdn         20   0 3252M  414M  183M S  8.0  5.2  0:01.56 /snap/firefox/4848/usr/lib/firefox
   5878 sdn         20   0 2466M  135M 88076 S  8.0  1.7  1:10.92 /snap/firefox/4848/usr/lib/firefox
  10299 sdn         20   0 5007M  679M 28544 S  8.0  8.6  0:13.90 /tmp/onos-2.7.0-jdk/bin/java -XX:+
  10458 sdn         20   0 5007M  679M 28544 S  8.0  8.6  0:01.09 /tmp/onos-2.7.0-jdk/bin/java -XX:+
  10699             10 -10  295M 40924  5888 S  8.0  0.5  0:00.38 ovs-vswitchd unix:/usr/local/var/r
      1             20   0  162M 11772  8188 S  0.0  0.1  0:01.04 /sbin/init splash
    210             19  -1 48764 18948 17412 S  0.0  0.2  0:00.41 /lib/systemd/systemd-journald
    259             20   0 27408  7408  4608 S  0.0  0.1  0:00.25 /lib/systemd/systemd-udevd
    460             20   0 14836  6784  6016 S  0.0  0.1  0:09.92 /lib/systemd/systemd-oomd
F1Help  F2Setup  F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

```
Welcome to Open Network Operating System (ONOS)!
     ____  _  ______  ____
    / __ \/ |/ / __ \/ __/
   / /_/ /    / /_/ /\ \
   \____/_/|_/\____/___/

Documentation: wiki.onosproject.org
Tutorials:     tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

sdn@root > apps -a -s
*   3 org.onosproject.hostprovider         2.7.0    Host Location Provider
*  16 org.onosproject.lldpprovider         2.7.0    LLDP Link Provider
*  17 org.onosproject.optical-model        2.7.0    Optical Network Model
*  18 org.onosproject.openflow-base        2.7.0    OpenFlow Base Provider
*  19 org.onosproject.openflow             2.7.0    OpenFlow Provider Suite
*  31 org.onosproject.drivers              2.7.0    Default Drivers
* 143 org.onosproject.gui2                 2.7.0    ONOS GUI2
sdn@root >
```

```
sdn@SDN-NFV:~/Desktop/NYCU_SDN-and-NFV/lab02/lab2_112550013/part3$ sudo mn --custom=topo_112550013
.py --topo=topo_112550013 --controller=remote,ip=127.0.0.1:6653 --switch=ovs,protocols=OpenFlow14
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s2) (s1, s2) (s2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
mininet> h1 ping h2 -c 5
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
```

```
[0] 0:sudo*                                                        "SDN-NFV" 18:17 19- 九 -24
```

Part 4

Control Plane Operations

1. **Switch sends Packet-In for ARP Request to ONOS**: The switch receives h1's ARP Request without a matching flow entry and sends it to the controller (`ReactiveForwarding.forward()` method).
2. **ReactiveForwarding floods ARP Request**: The controller decides to flood the ARP Request using the `flood()` method in the `ReactiveForwarding` application.
3. **Switch sends Packet-In for ARP Reply to ONOS**: The switch receives h2's ARP Reply without a matching flow and sends it to the controller.
4. **ReactiveForwarding forwards ARP Reply to h1**: The controller sends a Packet-Out to the switch to forward the ARP Reply to h1 (`ReactiveForwarding.forward()` method).
5. **Switch sends Packet-In for ICMP Echo Request to ONOS**: The switch receives h1's ICMP Echo Request to h2 and, lacking a flow entry, sends it to the controller.
6. **ReactiveForwarding installs flow rules for ICMP traffic**: The controller installs flow rules for traffic between h1 and h2 using the `installRule()` method in `ReactiveForwarding`.

Data Plane Operations

1. **h1 broadcasts ARP Request**: h1 sends an ARP Request to discover h2's MAC address.
2. **Switch floods ARP Request per controller's instruction**: The switch floods the ARP Request based on the controller's Packet-Out message.
3. **h2 replies with ARP Reply to h1**: h2 responds with an ARP Reply containing its MAC address.
4. **Switch forwards ARP Reply to h1**: The switch forwards the ARP Reply to h1 as instructed by the controller.
5. **h1 sends ICMP Echo Request to h2**: With h2's MAC address known, h1 sends an ICMP Echo Request directly to h2.
6. **Switch forwards ICMP packet based on flow rule**: The switch forwards the ICMP packet to h2 using the newly installed flow rule.
7. **h2 receives ICMP Echo Request**: h2 successfully receives the ICMP Echo Request from h1.

What you've learned or solved

- How to capture and analyze OpenFlow messages between ONOS and switches using Wireshark and observe different types of OpenFlow messages.
- How to install and delete flow rules in ONOS using REST APIs through JSON files and command-line tools like curl.
- How to create a network topology that leads to a broadcast storm, analyze the network behavior, and monitor CPU utilization during the event.