Part 1

There are 6 distinct `OFPT_FLOW_MOD` headers during the experiment.

| Match fields	Actions	Timeout Values
ETH_TYPE=ARP (0x0806)	OUTPUT_PORT=OFPP_CONTROLLER	0
ETH_TYPE=Unknown (0x8942)	OUTPUT_PORT=OFPP_CONTROLLER	0
ETH_TYPE=LLDP (0x88cc)	OUTPUT_PORT=OFPP_CONTROLLER	0
ETH_TYPE=IPv4 (0x0800)	OUTPUT_PORT=OFPP_CONTROLLER	0
IN_PORT=2 ETH_DST=2e:97:50:01:34:cd ETH_SRC=0a:3e:36:a0:71:3f	OUTPUT_PORT=1	0
IN_PORT=1 ETH_DST=0a:3e:36:a0:71:3f ETH_SRC=2e:97:50:01:34:cd	OUTPUT_PORT=2	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.
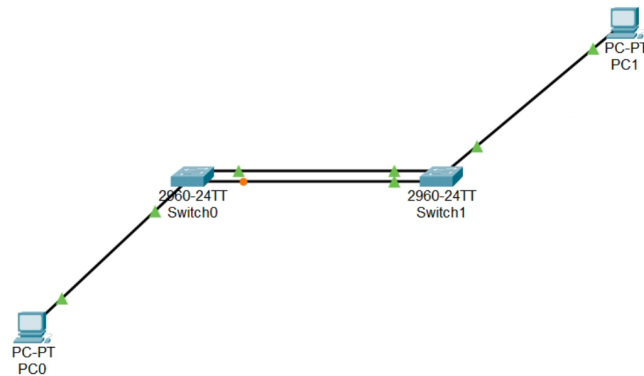
In this topology, there is a loop between switch 1 and switch 2. When a broadcast packet is sent from host 1, it will be forwarded to switch 1, then switch 2, and finally back to switch 1. This will cause the broadcast packet to be forwarded infinitely between switch 1 and switch 2, creating a broadcast storm, which makes the CPU usage of the virtual machine to be 100%.

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.