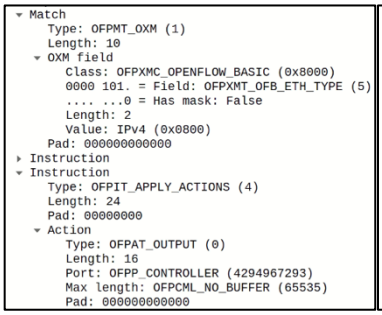
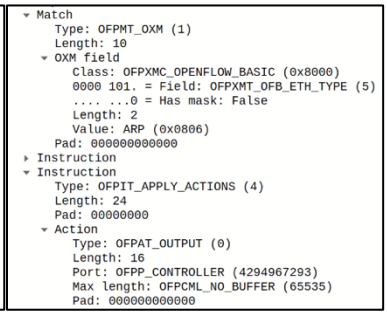
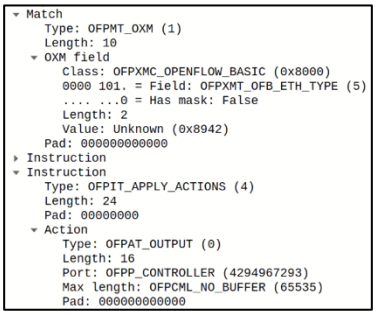
Lab 2

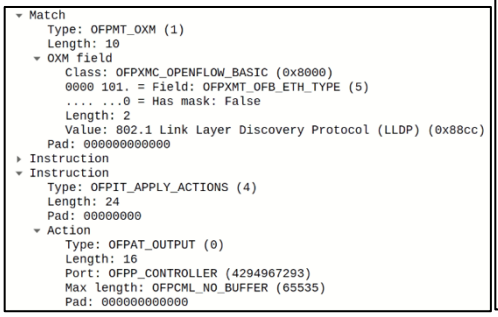
Part 1: Answer Questions

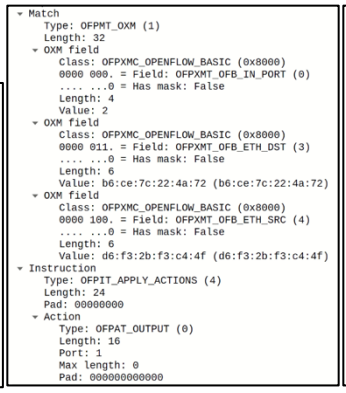
1. How many OpenFlow headers with type “OFPT_FLOW_MOD” and command “OFPFC_ADD” are there among all the packets?
   Ans: There are 6 distinct “OFPT_FLOW_MOD” headers during the experiment.
2. What are the match fields and the corresponding actions in each “OFPT_FLOW_MOD” message?

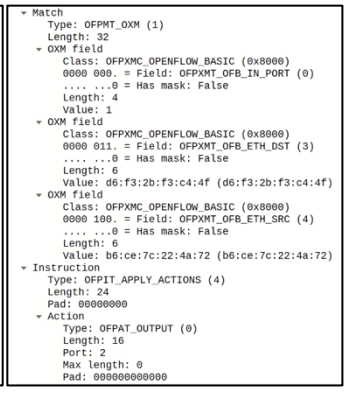




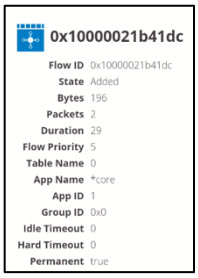


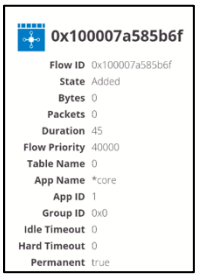


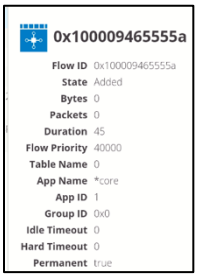


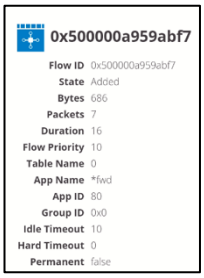


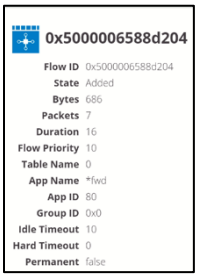
3. What are the Idle Timeout values for all flow rules on s1 in GUI?











Match fields	Actions	Timeout
OFPXMT_OFB_ETH_TYPE (5) Value = IPv4 (0x0800)	OFPAT_OUTPUT (0) Port = OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_ETH_TYPE (5) Value = ARP (0x0806)	OFPAT_OUTPUT (0) Port = OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_ETH_TYPE (5) Value = Unknown (0x8942) → Broadcast Domain Discovery Protocol	OFPAT_OUTPUT (0) Port = OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_ETH_TYPE (5) Value = 802.1 LLDP (0x88cc) → Link Layer Discovery Protocol	OFPAT_OUTPUT (0) Port = OFPP_CONTROLLER (4294967293)	0
OFPXMT_OFB_IN_PORT (0) Value = 2 OFPXMT_OFB_ETH_DST (3) Value = b6:ce:7c:22:4a:72 OFPXMT_OFB_ETH_SRC (4) Value = d6:f3:2b:f3:c4:4f	OFPAT_APPLY_ACTIONS (4) Port = 1	10
OFPXMT_OFB_IN_PORT (0) Value = 1 OFPXMT_OFB_ETH_DST (3) Value = d6:f3:2b:f3:c4:4f OFPXMT_OFB_ETH_SRC (4) Value = b6:ce:7c:22:4a:72	OFPAT_APPLY_ACTIONS (4) Port = 2	10

Part 2: Install Flow Rules

Arping

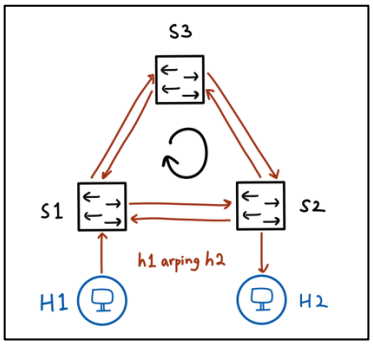
```
mininet> h1 arping h2
ARPING 10.0.0.2
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=0 time=933.365 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=1 time=11.068 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=2 time=9.504 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=3 time=3.029 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=4 time=3.168 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=5 time=9.679 usec
```

Ping

```
mininet> h1 arping h2
ARPING 10.0.0.2
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=0 time=933.365 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=1 time=11.068 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=2 time=9.504 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=3 time=3.029 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=4 time=3.168 usec
42 bytes from fa:2b:c3:35:65:aa (10.0.0.2): index=5 time=9.679 usec
```

After applying the flow rules, h1 can arping/ping h2 normally without activating the fwd app.

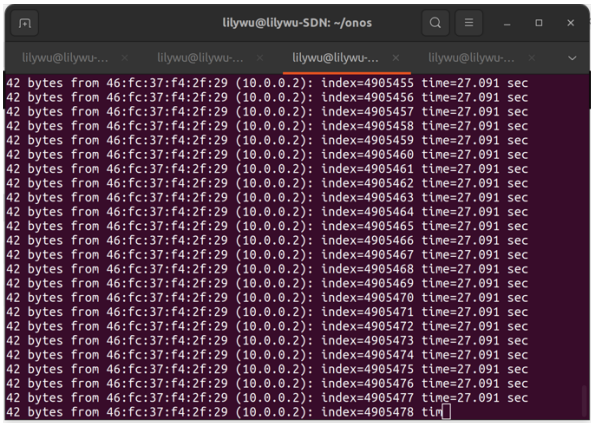
Part3: Create Topology with Broadcast Storm




```
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s1-1_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s1-2_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s1-3_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s2-1_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s2-2_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s2-3_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000002'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s3-1_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s3-2_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000003'
lilywu@lilywu-SDN:~/onos$ curl -u onos:rocks -X POST -H 'Content-Type: application/json'
 -d @flows_s1-1_110550091.json 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
```

With my topology shown in the left picture above, there are three switches and two hosts. The three switches forms a loop and h1 will try to arping h2. After implementing the flow rules to all the corresponding switches as shown on the right picture, h1 successfully arpings h2 and all the packets form a loop inside the topology, creating a Broadcasting Storm.

The left picture below shows the two hosts arpinging each other successfully and the right picture shows that the CPU uses significant resources on this Broadcasting Storm.



Part4: Trace ReactiveForwarding

1. h1 Sends a Ping

- **Data Plane**:
    - h1 sends an ICMP request (ping) to h2.
    - The first switch doesn't know where to send it because there's no rule yet, so it asks the controller (ONOS) what to do.

2. ONOS Gets Involved

- **Control Plane**:
    - The switch sends the ping packet to ONOS (the controller).
    - ONOS looks at the packet and figures out the best path from h1 to h2.
    - ONOS installs new rules in the switches along the path so they know how to handle future packets between h1 and h2.

3. Packet is Forwarded to h2

- **Data Plane**:
    - The switch now has a rule, so it forwards the ping from h1 to h2.
    - The packet follows the path through the switches and reaches h2.

4. h2 Responds

- **Data Plane**:
    - h2 replies with an ICMP Echo Reply (the ping response) back to h1.
    - Because ONOS already installed rules for both directions, the response travels back through the switches to h1 without any further help from ONOS.

5. h1 Gets the Reply

- **Data Plane**:
    - The reply reaches h1, completing the ping operation.

In Short:

- h1 sends a ping, the switch asks ONOS for help.
- ONOS installs rules in the switches so they know how to forward the packet.
- The ping reaches h2, h2 replies, and the reply comes back to h1.

What I've learned or solved.

1. **Observe Communication Between OpenFlow Controller and Switch**: I learned how to use Wireshark to capture and analyze the messages exchanged between the OpenFlow controller and switches. This allowed me to understand the different types of messages and their corresponding functions in the network.
2. **Creating and Installing Simple Flow Rules on Switches**:
   I practiced creating basic flow rules and installing them on switches. These rules

direct the switch on how to handle packets without needing constant guidance from the controller.

3. **Understanding Broadcast Storms**:
   I learned about broadcast storms, where excessive broadcast traffic can overwhelm the network and cause failure. This often happens due to misconfigurations or loops in the network.

4. **Switch and Controller Roles in Packet Transmission**:
   I observed how switches send packets to the controller when no flow rule is available. The controller then installs new flow rules to guide future packet forwarding.