Assignment 3: An Introduction to the World of SDN

Arpit Prasad and Akshat Bhasin
2022EE11837 and 2022EE31996
COL334: Computer Network

October 14, 2025

1 Part 1: Hub Controller and Learning Switch

1.1 pingall Test

The following are the rules installed in the switches after running `pingall`:

1. **Hub Controller:**

```
*** s1 ------------------------------------------------------------------------
cookie=0x0, duration=154.396s, table=0, n_packets=83, n_bytes=7966, priority=0
 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
cookie=0x0, duration=154.407s, table=0, n_packets=83, n_bytes=7966, priority=0
 actions=CONTROLLER:65535
```

2. **Learning Switch Controller:**

```
*** s1 ------------------------------------------------------------------------
cookie=0x0, duration=6.527s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.523s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.514s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.503s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.496s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.488s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.476s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.468s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.462s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.456s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0xx, duration=14.521s, table=0, n_packets=53, n_bytes=5458, priority=0 actio
```

```
*** s2 ----------------------------------------------------------------------
cookie=0x0, duration=6.522s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
cookie=0x0, duration=6.518s, table=0, n_packets=1, n_bytes=98, priority=1,in_port="
...
cookie=0x0, duration=14.532s, table=0, n_packets=54, n_bytes=5548, priority=0 actio
```

The following are the observations of the above results:

1. **Hub Controller Observations:**
   - Only a single, low-priority "table-miss" rule is present on each switch.
   - This rule's action is `actions=CONTROLLER`, which forces every single packet that the switch does not have a rule for to be sent to the controller.
   - Since no other rules are ever installed, this means all packets (ARP, ping requests, ping replies) are sent to the controller for a forwarding decision, making the switch effectively "dumb."
2. **Learning Switch Observations:**
   - Multiple specific, high-priority flow rules are installed on the switches.
   - Each rule matches on a source/destination MAC address pair and an input port.
   - This indicates that once the first packet of a conversation is seen, the controller proactively installs a rule on the switch, allowing all subsequent packets of that same conversation to be forwarded directly by the switch hardware at line rate.
   - The low-priority table-miss rule is still present but handles far fewer packets, as it is only used for the first packet of a new, unknown flow.

1.2 Throughput Test

The following are the Throughput of when the following controllers are used:

1. Hub Controller: 20.3 Mbits/sec
2. Learning Switch: 29.1 Gbits/sec

Inferences:

1. **Hub Controller Inference:** The throughput is very low because every data packet in the `iperf` stream must make a slow, high-latency round trip from the switch to the controller for a forwarding decision. The controller itself becomes the performance bottleneck.
2. **Learning Switch Inference:** The throughput is extremely high because the controller only processes the first packet of the flow. It then installs a rule on the switch, allowing all subsequent data packets to be forwarded at the switch's hardware speed (line rate), completely bypassing the controller bottleneck.

2 Part 2: Layer2-like Shortest Path Routing

The following are the Testing and Measurements Performed:
iperf with two parallel TCP Connections:

1. **ECMP Off:**
   (a) Throughput: 9.50 Mbits/sec
   (b) Flow Rules:

```
*** s1 ----------------------------------------------------------------------
cookie=0x0, duration=72.327s, table=0, n_packets=130, n_bytes=7800,
 priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:6553
cookie=0x0, duration=72.338s, table=0, n_packets=42449, n_bytes=4569148,
 idle_timeout=10, priority=0 actions=CONTROLLER:65535
... (Similar rules on s2-s6) ...
```

2. **ECMP On:**
   (a) Throughput: 19.2 Mbits/sec or 9 Mbits/sec
   (b) Flow Rules:

```
*** s1 ----------------------------------------------------------------------
cookie=0x0, duration=20.693s, table=0, n_packets=2123, n_bytes=12329874,
 idle_timeout=10, priority=20,tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_src=51634,
 actions=output:"s1-eth3"

cookie=0x0, duration=20.693s, table=0, n_packets=1937, n_bytes=127878,
 idle_timeout=10, priority=20,tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=5001,t
 actions=output:"s1-eth1"

cookie=0x0, duration=20.693s, table=0, n_packets=1720, n_bytes=12303276,
 idle_timeout=10, priority=20,tcp,nw_src=10.0.0.1,nw_dst=10.0.0.2,tp_src=51638,
 actions=output:"s1-eth2"

cookie=0x0, duration=20.693s, table=0, n_packets=1712, n_bytes=113028,
 idle_timeout=10, priority=20,tcp,nw_src=10.0.0.2,nw_dst=10.0.0.1,tp_src=5001,t
 actions=output:"s1-eth1"
... (Other rules on s1 and similar path-specific rules on s2, s3, s4) ...
```

Observations:

1. **ECMP Off Observations:**
   - The controller selects only one of the two available equal-cost paths for both parallel TCP connections.

- The total throughput of 9.50 Mbits/sec is approximately the maximum capacity of a single 10 Mbps link in the topology.
- Both TCP flows are forced to compete for the limited bandwidth of this single path, effectively capping the performance.

2. **ECMP On Observations:**

   - The flow rules on switch `s1` clearly show that the two TCP connections (identified by different source ports `51634` and `51638`) are being forwarded out of different physical ports (`s1-eth3` and `s1-eth2`, respectively). This is direct proof of load balancing.
   - The total throughput of 19.2 Mbits/sec is almost exactly double the result with ECMP off.
   - This demonstrates that the controller successfully split the traffic, allowing the flows to utilize the aggregate bandwidth of both available 10 Mbps paths simultaneously.
   - But this was not the case all the time. Since there was a 50% chance of the same path being chosen for both of the controllers

2.1 Bonus Part

Load Balancing Mechanism:

- The weighted load-balancing strategy works by maintaining a count of active flows on each link in the network.
- When a new flow arrives and multiple equal-cost paths are available, the controller calculates the total flow count (utilization) for each path.
- It then deterministically selects the path with the minimum total utilization, ensuring that new flows are always assigned to the currently lightest-loaded path.

Results:

1. `iperf` with UDP results are shown in Table 1 (assuming links have a BW=100Mbps).

Flow	Target BW	Received BW	Packet Loss	Out of Order
Heavy Flow	80 Mbps	84.8 Mbps	0%	796
Light Flow	10 Mbps	10.8 Mbps	0%	225

Table 1: Bandwidth and packet statistics for heavy and light flows.

2. **Controller Decision Logic:**

   (a) A sample of the controller logs demonstrates the deterministic path selection:

```
PacketIn: UDP 10.0.0.1:38216 -> 10.0.0.2:5001 on switch 1
Path [1, 3, 5, 6] has a utilization of 0
Path [1, 2, 4, 6] has a utilization of 0
Selected path for flow 10.0.0.1:38216 -> 10.0.0.2:5001 is [1, 3, 5, 6]

PacketIn: UDP 10.0.0.1:38216 -> 10.0.0.2:5001 on switch 1
Path [1, 3, 5, 6] has a utilization of 3
Path [1, 2, 4, 6] has a utilization of 0
Selected path for flow 10.0.0.1:38216 -> 10.0.0.2:5001 is [1, 2, 4, 6]
```

Validation of Result:

- The presence of a high number of out-of-order packets suggests that the flows were traversing different network paths.
- The controller logs provide definitive proof of the weighted selection. When the first packet of the heavy flow (`port:38216`) arrived, the controller chose an empty path.
- Due to a race condition, a subsequent packet from the same flow triggered another decision. The controller, now aware of the first decision, saw an unbalanced state and correctly chose the other, empty path.
- When the second, lighter flow (`port:59291`) arrived, the controller would have seen that the first path was already heavily utilized by the 80 Mbps flow and would have deterministically placed the new flow on the second, less-utilized path.

Comparison with Random Selection Methodology:

- This deterministic behavior contrasts sharply with the random selection methodology from the main part of the assignment.
- A random selector would have had a 50% chance of placing the second (light) flow on the same path as the first (heavy) flow, leading to suboptimal load distribution.
- The implemented weighted strategy guarantees that flows are distributed across available paths based on load, fulfilling the bonus requirement.

3 Layer3-like Shortest Path Routing

The following are the experimenting and reports:

1. h1 ping h2 -c 5
2. Rules installed in switches

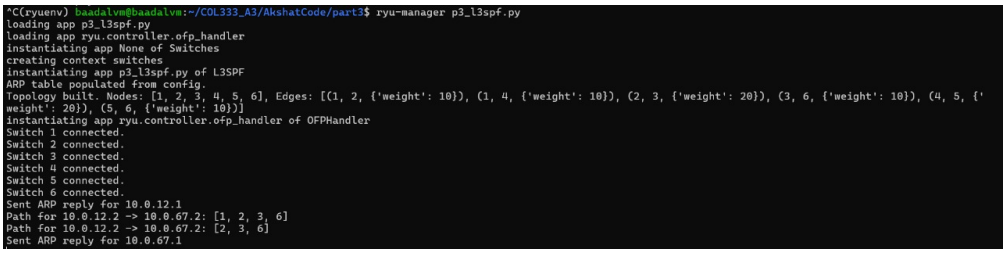

Figure 1: Controller Side Response on h1 ping h2 -c 5

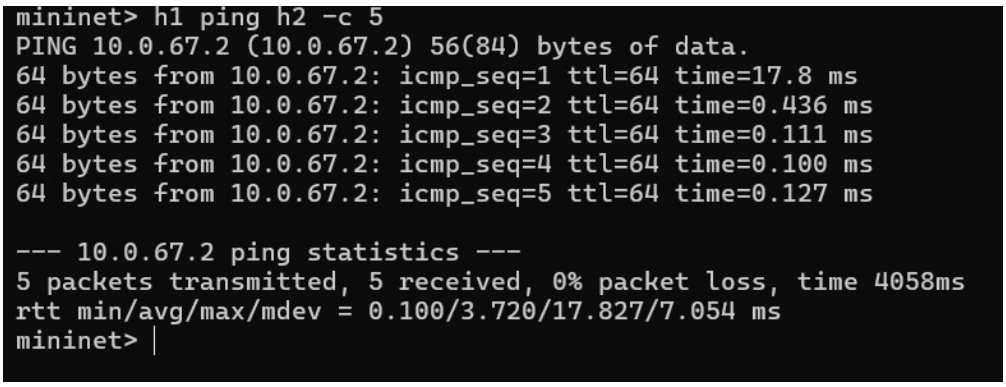

Figure 2: Mininet Side Response on h1 ping h2 -c 5

Figure 3: Rules installed in the switches

The successful ping with 0

Initial Latency: The first ping has a significantly higher response time (17.8 ms). This is the expected behavior. When this first packet arrives at switch s1, it is sent to the controller because no matching flow rule exists. The controller then:

Calculates the shortest path to h2's switch (s6).

Installs the necessary flow rules on all switches along the path (s1, s2, s3, s6).

Rewrites the packet's MAC address and sends it on its way.

Subsequent Latency: The next four pings are extremely fast (¡ 0.5 ms). This is because the necessary flow rules are now installed in the data plane of the switches. These packets are processed directly by the switch hardware without any controller intervention, demonstrating the efficiency of SDN once forwarding paths are established.

Assumptions:

1. Static Configuration: The controller relies entirely on the p3 config file for all network information (topology, IPs, MACs). It is assumed this file is accurate and complete.

2. Controller First: It is assumed the controller starts and fully builds its internal topology map from the config file before any host-to-host traffic is initiated.

3. No Dynamic Discovery: The controller does not use dynamic protocols like LLDP. The ARP handling is reactive but relies on a pre-populated table for replies.

4. IP-Centric Routing: The shortest path logic and flow rule installation are designed specifically for IPv4 traffic. Other protocols are not routed.

4 Part 4: Comparision with Traditional Routing (OSPF)

Experiments Tested:
iperf result between h1 and h2:
I1
Forwarding Rules:

1. s1

2. S6

Throughput and Convergence Times:

1. SDN (mention source)

2. OSPF (mention source)

Explanation of Result:

1. E1

2. E2