Assignment 3: An Introduction to the World of SDN

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

October 16, 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 Part 3: Layer3-like Shortest Path Routing

The following experiments were conducted to validate the L3 routing controller.

3.1 Ping Test

A 5-packet ping test was conducted from host `h1` to `h2` to verify inter-subnet connectivity.

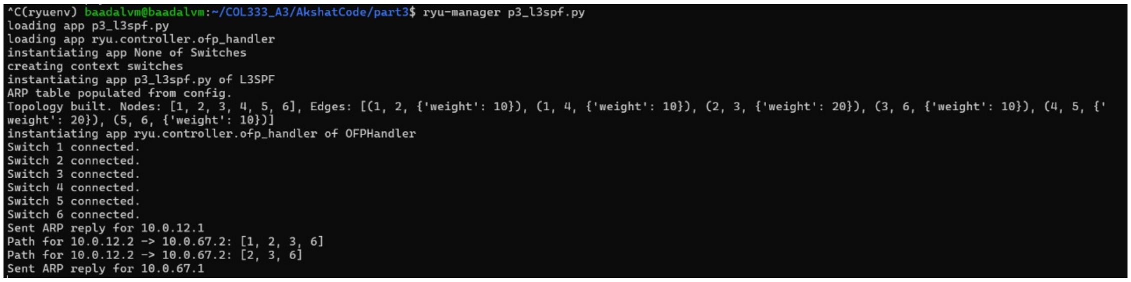


Figure 1: Controller logs showing ARP and path calculation for the first ping.

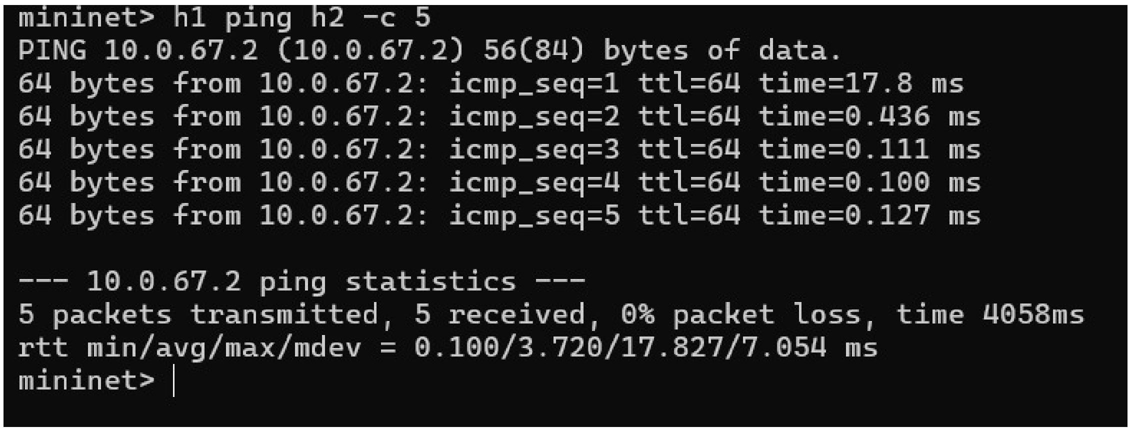


Figure 2: Mininet CLI showing results of `h1 ping h2 -c 5`.

Observations

The successful ping with 0% packet loss confirms that the controller correctly handles inter-subnet routing. The following latency characteristics were observed:

- **Initial Latency:** The first ping has a significantly higher response time (17.8 ms). This is expected behavior as the first packet triggers a table-miss on switch `s1`, which is sent to the controller. The controller then calculates the shortest path, installs flow rules on all switches along the computed path (`s1, s2, s4, s6`), and forwards the packet.
- **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 at line-rate, bypassing the controller and demonstrating the efficiency of the established forwarding path.

3.2 Installed Flow Rules

The flow tables of the switches were dumped after the ping test to inspect the installed rules.

Observations

The rules confirm the L3 routing behavior:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=8.192s, table=0, n_packets=4, n_bytes=392, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.67.2 actions=mod_dl_dst:00:00:0
0:00:02:01,output:"s1-eth2"
 cookie=0x0, duration=8.192s, table=0, n_packets=5, n_bytes=490, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.12.2 actions=mod_dl_dst:00:00:0
0:00:01:02,output:"s1-eth1"
 cookie=0x0, duration=10.354s, table=0, n_packets=83, n_bytes=10530, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=8.203s, table=0, n_packets=5, n_bytes=490, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.12.2 actions=mod_dl_dst:00:00:0
0:00:01:02,output:"s2-eth1"
 cookie=0x0, duration=8.197s, table=0, n_packets=4, n_bytes=392, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.67.2 actions=mod_dl_dst:00:00:0
0:00:03:01,output:"s2-eth2"
 cookie=0x0, duration=10.324s, table=0, n_packets=77, n_bytes=10038, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
*** s3 ------------------------------------------------------------------------
 cookie=0x0, duration=8.211s, table=0, n_packets=5, n_bytes=490, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.67.2 actions=mod_dl_dst:00:00:0
0:00:06:01,output:"s3-eth2"
 cookie=0x0, duration=8.211s, table=0, n_packets=5, n_bytes=490, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.12.2 actions=mod_dl_dst:00:00:0
0:00:02:02,output:"s3-eth1"
 cookie=0x0, duration=10.296s, table=0, n_packets=78, n_bytes=10118, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
*** s4 ------------------------------------------------------------------------
 cookie=0x0, duration=10.265s, table=0, n_packets=77, n_bytes=10028, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
*** s5 ------------------------------------------------------------------------
 cookie=0x0, duration=10.231s, table=0, n_packets=79, n_bytes=10190, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
*** s6 ------------------------------------------------------------------------
 cookie=0x0, duration=8.237s, table=0, n_packets=5, n_bytes=490, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.12.2 actions=mod_dl_dst:00:00:0
0:00:03:02,output:"s6-eth1"
 cookie=0x0, duration=8.235s, table=0, n_packets=4, n_bytes=392, idle_timeout=30, hard_timeout=60, priority=1,ip,nw_dst=10.0.67.2 actions=mod_dl_dst:00:00:0
0:00:06:02,output:"s6-eth3"
 cookie=0x0, duration=10.195s, table=0, n_packets=81, n_bytes=10352, idle_timeout=30, hard_timeout=60, priority=0 actions=CONTROLLER:65535
```

Figure 3: Flow rules installed on switches along the path.

- Each rule matches on the destination IP address (`nw_dst`) only, emulating a traditional routing table.
- The actions include `mod_dl_src`, `mod_dl_dst` (MAC address rewriting), and `DEC_NW_TTL` (TTL decrement), which are characteristic of a Layer 3 router.
- The final action is `output:<port>`, forwarding the modified frame to the next hop.

3.3 Assumptions

The implementation of the L3 routing controller is based on the following key assumptions:

1. **Static Configuration:** The controller relies entirely on the `p3_config.json` file for all network information (topology, IPs, MACs). The configuration is assumed to be accurate and complete. No new links can be added on the fly.
2. **Centralized ARP Proxy:** The controller intercepts all ARP requests and generates synthetic replies from its static configuration. Hosts do not perform dynamic ARP discovery among themselves.
3. **Reactive Path Calculation:** End-to-end paths and their corresponding flow rules are calculated and installed reactively, only upon receiving the first packet of a new flow.
4. **Destination-Based Forwarding:** All routing decisions and installed flow rules are based solely on the destination IP address of a packet, consistent with standard IP routing.

4 Part 4: Comparison with Traditional Routing (OSPF)

This section details the comparison between a traditional OSPF-based routing setup and our custom SDN controller, focusing on performance during a link failure event.

4.1 Warm-up Experiment

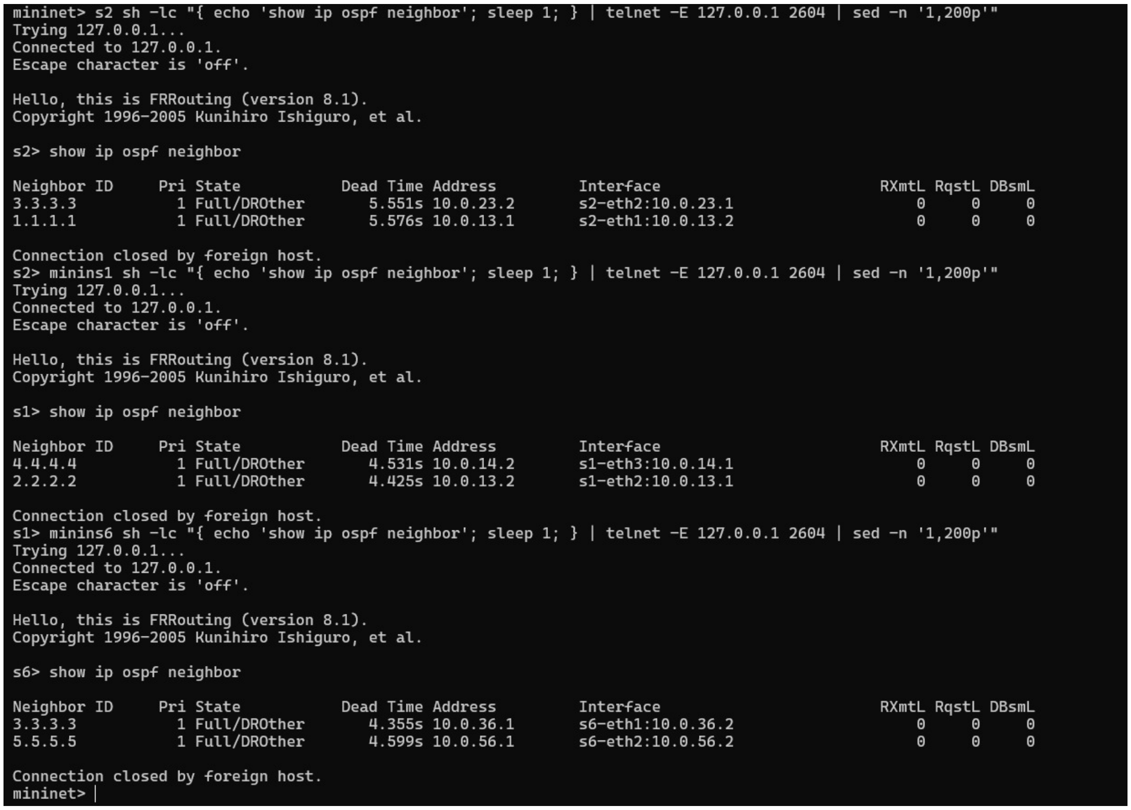
Before the failure simulation, a warm-up experiment established a baseline. An `iperf` test between h1 and h2 confirmed that OSPF had converged and established a stable path, achieving a throughput of approximately 95.5 Mbits/sec (Figure 4). The established OSPF neighbor relationships (Figure 5) and the resulting IP routes on the switches (Figure 6) confirmed a stable network state prior to the test. Finally, the forwarding rules on switches s1 and s6 were recorded (Figure 7).

```
*** Starting 0 switches

*** Assign gateway IPs/MACs on host-facing switch ports
*** Assign IPs/MACs on ALL inter-switch links (per config)
*** Configure hosts: IP/MAC + default routes
*** FRR (zebra+ospfd) started on all routers; waiting a bit…
*** Waiting for OSPF convergence (<= 60s)...
✅ OSPF converged (routes present)
*** 1. Running OSPF Warm-up Experiment ***
--- Warm-up iperf running for 10s. Logs: h1_iperf_warmup.log, h2_iperf_warmup.log

==== iperf CLIENT (h1) Warm-up ====
------------------------------------------------------------
Client connecting to 10.0.67.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.12.2 port 49790 connected with 10.0.67.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  12.0 MBytes   101 Mbits/sec
[  1] 1.0000-2.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 2.0000-3.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 3.0000-4.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 4.0000-5.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 5.0000-6.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 6.0000-7.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 7.0000-8.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 8.0000-9.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 9.0000-10.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 0.0000-10.0803 sec   115 MBytes  95.5 Mbits/sec


==== iperf SERVER (h2) Warm-up ====
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.67.2 port 5001 connected with 10.0.12.2 port 49790
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0680 sec   115 MBytes  95.6 Mbits/sec
```

Figure 4: Baseline `iperf` throughput between h1 and h2 after OSPF convergence.

4.2 Link Failure Analysis

The core experiment involved running a 30-second `iperf` test between h1 and h2, during which a primary link was brought down at T=2s and restored at T=7s. The raw `iperf` logs provide the data for this analysis.



```
mininet> s2 sh -lc "{ echo 'show ip ospf neighbor'; sleep 1; } | telnet -E 127.0.0.1 2604 | sed -n '1,200p'"
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is 'off'.

Hello, this is FRRouting (version 8.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

s2> show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface                        RXmtL RqstL DBsmL
3.3.3.3           1 Full/DROther       5.551s 10.0.23.2       s2-eth2:10.0.23.1                    0     0     0
1.1.1.1           1 Full/DROther       5.576s 10.0.13.1       s2-eth1:10.0.13.2                    0     0     0

Connection closed by foreign host.
s2> minins1 sh -lc "{ echo 'show ip ospf neighbor'; sleep 1; } | telnet -E 127.0.0.1 2604 | sed -n '1,200p'"
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is 'off'.

Hello, this is FRRouting (version 8.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

s1> show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface                        RXmtL RqstL DBsmL
4.4.4.4           1 Full/DROther       4.531s 10.0.14.2       s1-eth3:10.0.14.1                    0     0     0
2.2.2.2           1 Full/DROther       4.425s 10.0.13.2       s1-eth2:10.0.13.1                    0     0     0

Connection closed by foreign host.
s1> minins6 sh -lc "{ echo 'show ip ospf neighbor'; sleep 1; } | telnet -E 127.0.0.1 2604 | sed -n '1,200p'"
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is 'off'.

Hello, this is FRRouting (version 8.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

s6> show ip ospf neighbor

Neighbor ID     Pri State           Dead Time Address         Interface                        RXmtL RqstL DBsmL
3.3.3.3           1 Full/DROther       4.355s 10.0.36.1       s6-eth1:10.0.36.2                    0     0     0
5.5.5.5           1 Full/DROther       4.599s 10.0.56.1       s6-eth2:10.0.56.2                    0     0     0

Connection closed by foreign host.
mininet> |
```

Figure 5: OSPF neighbor relationships established on key switches.

4.2.1 OSPF Performance

Data Plane Convergence The plotted throughput graph (Figure 8) is derived from the client-side `iperf` logs (Figure 9). The data shows the OSPF network reacted to the link failure by rerouting traffic.

- **0-2s:** Throughput is stable at the maximum rate of the primary path (≈100 Mbits/sec).
- **2-3s (Failure):** The link fails. Throughput drops immediately to the alternate path's capacity (≈10 Mbits/sec).
- **7s (Recovery):** The primary link is restored.
- **7-15s:** Throughput remains low on the alternate path while the OSPF control plane re-converges.
- **15s onwards:** Throughput returns to the maximum rate as traffic is redirected back to the primary, faster path.

From a data plane perspective, it took OSPF approximately **8 seconds** (from T=7s to T=15s) to restore traffic to the optimal path after the link was physically restored.

```
mininet> s1 ip route
10.0.12.0/24 dev s1-eth1 proto kernel scope link src 10.0.12.1
10.0.13.0/24 dev s1-eth2 proto kernel scope link src 10.0.13.1
10.0.14.0/24 dev s1-eth3 proto kernel scope link src 10.0.14.1
10.0.23.0/24 nhid 30 via 10.0.13.2 dev s1-eth2 proto ospf metric 20
10.0.36.0/24 nhid 30 via 10.0.13.2 dev s1-eth2 proto ospf metric 20
10.0.45.0/24 nhid 16 via 10.0.14.2 dev s1-eth3 proto ospf metric 20
10.0.56.0/24 nhid 16 via 10.0.14.2 dev s1-eth3 proto ospf metric 20
10.0.67.0/24 nhid 30 via 10.0.13.2 dev s1-eth2 proto ospf metric 20
mininet> s6 ip route
10.0.12.0/24 nhid 14 via 10.0.36.1 dev s6-eth1 proto ospf metric 20
10.0.13.0/24 nhid 14 via 10.0.36.1 dev s6-eth1 proto ospf metric 20
10.0.14.0/24 nhid 16 via 10.0.56.1 dev s6-eth2 proto ospf metric 20
10.0.23.0/24 nhid 14 via 10.0.36.1 dev s6-eth1 proto ospf metric 20
10.0.36.0/24 dev s6-eth1 proto kernel scope link src 10.0.36.2
10.0.45.0/24 nhid 16 via 10.0.56.1 dev s6-eth2 proto ospf metric 20
10.0.56.0/24 dev s6-eth2 proto kernel scope link src 10.0.56.2
10.0.67.0/24 dev s6-eth3 proto kernel scope link src 10.0.67.1
mininet> h1 ping -c 3 h2
PING 10.0.67.2 (10.0.67.2) 56(84) bytes of data.
64 bytes from 10.0.67.2: icmp_seq=1 ttl=60 time=0.244 ms
64 bytes from 10.0.67.2: icmp_seq=2 ttl=60 time=0.158 ms
64 bytes from 10.0.67.2: icmp_seq=3 ttl=60 time=0.171 ms

--- 10.0.67.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.158/0.191/0.244/0.037 ms
```

Figure 6: IP routes and successful ping test after OSPF convergence.

4.2.2 SDN Controller Performance

Data Plane Convergence The SDN controller's recovery profile is plotted in Figure 11, based on the logs in Figure 12.

- **0-2s:** Throughput is stable at the maximum path rate (≈100 Mbits/sec).
- **2s (Failure):** The link fails.
- **2-8s:** Throughput drops to nearly zero, representing a complete data plane outage.
- **7s (Recovery):** The primary link is restored.
- **8-9s:** Throughput rapidly recovers as the controller immediately reinstalls the optimal path.

The SDN approach restored traffic to the optimal path within **2 seconds** of the link's physical recovery.

4.3 Comparative Analysis and Conclusion

The results clearly demonstrate the superior convergence speed of the SDN architecture over traditional, distributed routing protocols like OSPF.

```
--- Recording forwarding rules on s1 and s6:

==== s1 OSPF Routing Table ====
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* 10.0.12.0/24 [110/51] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O>* 10.0.13.0/24 [110/50] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O>* 10.0.14.0/24 [110/60] via 10.0.56.1, s6-eth2, weight 1, 00:00:09
O>* 10.0.23.0/24 [110/30] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O   10.0.36.0/24 [110/20] is directly connected, s6-eth1, weight 1, 00:00:26
O>* 10.0.45.0/24 [110/40] via 10.0.56.1, s6-eth2, weight 1, 00:00:09
O   10.0.56.0/24 [110/20] is directly connected, s6-eth2, weight 1, 00:00:26
O   10.0.67.0/24 [110/1] is directly connected, s6-eth3, weight 1, 00:00:26


==== s6 OSPF Routing Table ====
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O>* 10.0.12.0/24 [110/51] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O>* 10.0.13.0/24 [110/50] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O>* 10.0.14.0/24 [110/60] via 10.0.56.1, s6-eth2, weight 1, 00:00:09
O>* 10.0.23.0/24 [110/30] via 10.0.36.1, s6-eth1, weight 1, 00:00:14
O   10.0.36.0/24 [110/20] is directly connected, s6-eth1, weight 1, 00:00:26
O>* 10.0.45.0/24 [110/40] via 10.0.56.1, s6-eth2, weight 1, 00:00:09
O   10.0.56.0/24 [110/20] is directly connected, s6-eth2, weight 1, 00:00:26
O   10.0.67.0/24 [110/1] is directly connected, s6-eth3, weight 1, 00:00:26
```

Figure 7: OSPF forwarding rules installed on s1 and s6 after convergence.

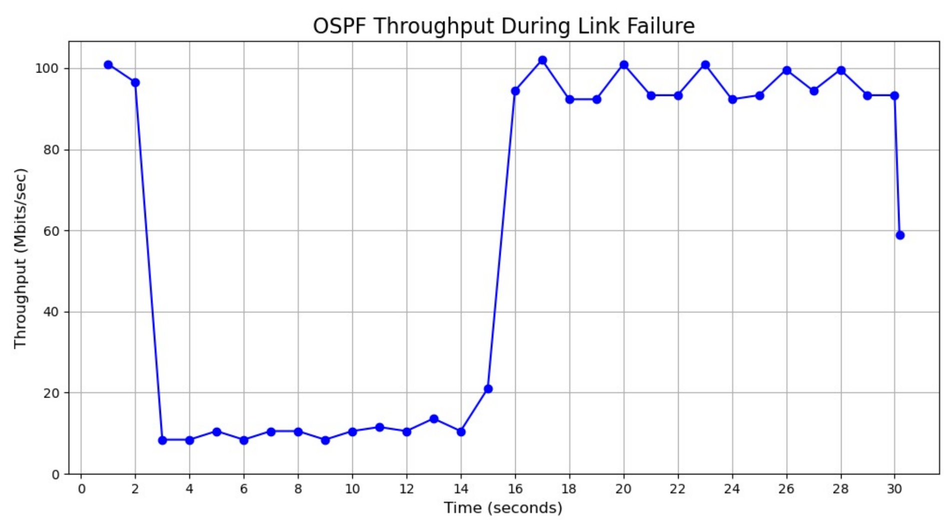


Figure 8: OSPF throughput graph during the link failure experiment.

```
==== iperf CLIENT (h1) Link Failure Test ====
------------------------------------------------------------
Client connecting to 10.0.67.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.12.2 port 47006 connected with 10.0.67.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  12.0 MBytes   101 Mbits/sec
[  1] 1.0000-2.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 2.0000-3.0000 sec  1.00 MBytes  8.39 Mbits/sec
[  1] 3.0000-4.0000 sec  1.00 MBytes  8.39 Mbits/sec
[  1] 4.0000-5.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 5.0000-6.0000 sec  1.00 MBytes  8.39 Mbits/sec
[  1] 6.0000-7.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 7.0000-8.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 8.0000-9.0000 sec  1.00 MBytes  8.39 Mbits/sec
[  1] 9.0000-10.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 10.0000-11.0000 sec  1.38 MBytes  11.5 Mbits/sec
[  1] 11.0000-12.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 12.0000-13.0000 sec  1.62 MBytes  13.6 Mbits/sec
[  1] 13.0000-14.0000 sec  1.25 MBytes  10.5 Mbits/sec
[  1] 14.0000-15.0000 sec  2.50 MBytes  21.0 Mbits/sec
[  1] 15.0000-16.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 16.0000-17.0000 sec  12.1 MBytes   102 Mbits/sec
[  1] 17.0000-18.0000 sec  11.0 MBytes  92.3 Mbits/sec
[  1] 18.0000-19.0000 sec  11.0 MBytes  92.3 Mbits/sec
[  1] 19.0000-20.0000 sec  12.0 MBytes   101 Mbits/sec
[  1] 20.0000-21.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 21.0000-22.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 22.0000-23.0000 sec  12.0 MBytes   101 Mbits/sec
[  1] 23.0000-24.0000 sec  11.0 MBytes  92.3 Mbits/sec
[  1] 24.0000-25.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 25.0000-26.0000 sec  11.9 MBytes  99.6 Mbits/sec
[  1] 26.0000-27.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 27.0000-28.0000 sec  11.9 MBytes  99.6 Mbits/sec
[  1] 28.0000-29.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 29.0000-30.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 0.0000-30.1815 sec   212 MBytes  58.8 Mbits/sec
```

Figure 9: Client-side `iperf` logs for the OSPF link failure test.

```
==== iperf SERVER (h2) Link Failure Test ====
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.67.2 port 5001 connected with 10.0.12.2 port 47006
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.1793 sec   212 MBytes  58.8 Mbits/sec
```

Figure 10: Server-side `iperf` logs for the OSPF link failure test.

In conclusion, OSPF's convergence is limited by its distributed, timer-based design. In contrast, the SDN controller's centralized global view and event-driven notifications allow

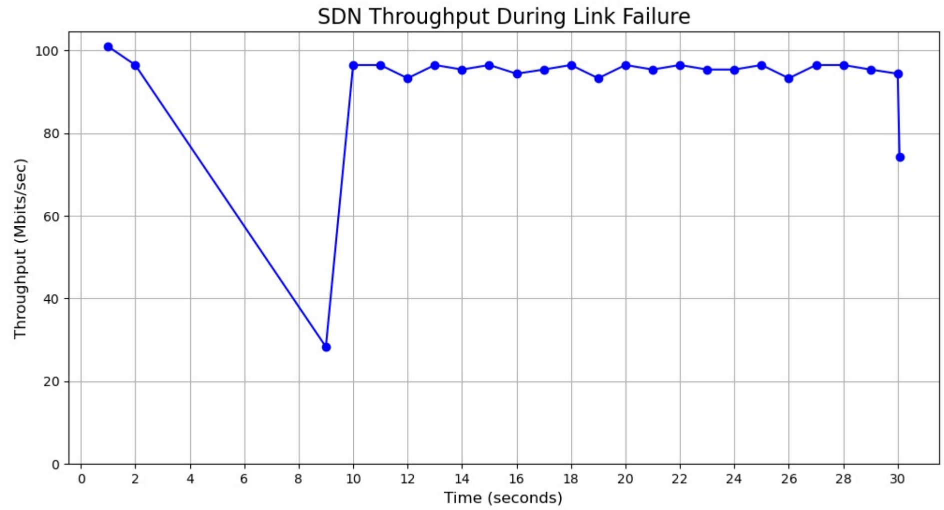


Figure 11: SDN throughput graph during the link failure experiment.

Metric	**OSPF**	**SDN Controller**
Control Plane Failure Detection	≈6 seconds (Timer-based)	Milliseconds (Event-drive
Control Plane Recovery Detection	≈7 seconds (Adjacency-based)	Milliseconds (Event-drive
Data Plane Recovery to Optimal Path	≈8 seconds	≈2 seconds

Table 2: Comparison of OSPF and SDN Convergence Times.

it to react almost instantaneously. While this led to a brief, total data plane outage, the network recovered to its optimal state far more quickly than OSPF.

```
==== iperf CLIENT (h1) SDN Link Failure Test ====
------------------------------------------------------------
Client connecting to 10.0.67.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.12.2 port 60052 connected with 10.0.67.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-1.0000 sec  12.0 MBytes   101 Mbits/sec
[  1] 1.0000-2.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 2.0000-3.0000 sec  11.8 KBytes  96.9 Kbits/sec
[  1] 3.0000-4.0000 sec  63.6 KBytes   521 Kbits/sec
[  1] 4.0000-5.0000 sec  0.000 Bytes  0.000 bits/sec
[  1] 5.0000-6.0000 sec  0.000 Bytes  0.000 bits/sec
[  1] 6.0000-7.0000 sec  0.000 Bytes  0.000 bits/sec
[  1] 7.0000-8.0000 sec  0.000 Bytes  0.000 bits/sec
[  1] 8.0000-9.0000 sec  3.38 MBytes  28.3 Mbits/sec
[  1] 9.0000-10.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 10.0000-11.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 11.0000-12.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 12.0000-13.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 13.0000-14.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 14.0000-15.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 15.0000-16.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 16.0000-17.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 17.0000-18.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 18.0000-19.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 19.0000-20.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 20.0000-21.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 21.0000-22.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 22.0000-23.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 23.0000-24.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 24.0000-25.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 25.0000-26.0000 sec  11.1 MBytes  93.3 Mbits/sec
[  1] 26.0000-27.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 27.0000-28.0000 sec  11.5 MBytes  96.5 Mbits/sec
[  1] 28.0000-29.0000 sec  11.4 MBytes  95.4 Mbits/sec
[  1] 29.0000-30.0000 sec  11.2 MBytes  94.4 Mbits/sec
[  1] 0.0000-30.0662 sec   266 MBytes  74.3 Mbits/sec


==== iperf SERVER (h2) SDN Link Failure Test ====
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 10.0.67.2 port 5001 connected with 10.0.12.2 port 60052
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.0505 sec   266 MBytes  74.3 Mbits/sec
```

Figure 12: Client and server `iperf` logs for the SDN link failure test.