CS 331 - COMPUTER NETWORKS

Assignment 3

Team 35

Heer Kubadia - 22110096
Lavanya - 22110130

All codes available at - [Group 35 - Assignment 3]

Q1: Network Loops

Objective

The objective of this assignment is to:

- Simulate a network topology containing loops using Mininet.
- Analyze the effects of loops on network behavior using ICMP ping tests.
- Identify communication failures caused by loops (e.g., packet drops, broadcast storms).
- Resolve the issues caused by loops by configuring static routes and enabling IP forwarding — without modifying the physical topology (i.e., no deletion of switches, hosts, or links).
- Demonstrate that end-to-end connectivity is restored through correct route configuration, even in the presence of loops.

Environment Setup with Multipass

To simulate the network and perform experiments, we used **Multipass** to create and manage an Ubuntu virtual machine. All Mininet-related tasks were carried out within this isolated environment to ensure a clean and reproducible setup. The following steps were taken:

```
multipass launch --name mininet-vm --mem 4G --disk 10G
multipass shell mininet-vm
```

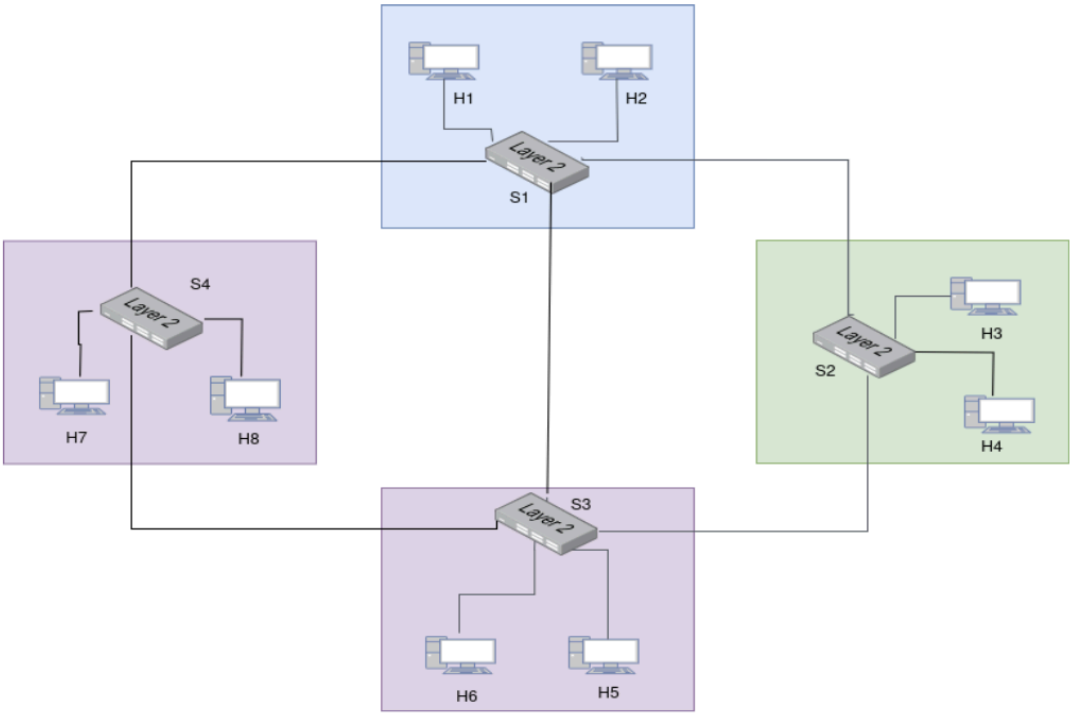
Once inside the shell, we installed Mininet using:

```
sudo apt update
sudo apt install mininet
```

This setup provided a lightweight, efficient environment to build and test custom network topologies using Mininet, without affecting the host system.

Custom Topology Setup using Python (topology.py)

We defined a custom topology with a loop as shown in figure below using the following Python script (topology.py):




```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import OVSSwitch
from mininet.cli import CLI
from mininet.link import TCLink
from mininet.log import setLogLevel
# Custom switch class that disables controller dependency
class StandaloneOVSSwitch(OVSSwitch):
```

```
    def start(self, controllers):
        # Override default to start without controllers
        return super().start([])
class LoopTopo(Topo):
    def build(self):
        # Add switches
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')
        # Add hosts with IPs
        hosts = [
            ('h1', '10.0.0.2/24', s1),
            ('h2', '10.0.0.3/24', s1),
            ('h3', '10.0.0.4/24', s2),
            ('h4', '10.0.0.5/24', s2),
            ('h5', '10.0.0.6/24', s3),
            ('h6', '10.0.0.7/24', s3),
            ('h7', '10.0.0.8/24', s4),
            ('h8', '10.0.0.9/24', s4)
        ]
        for name, ip, switch in hosts:
            host = self.addHost(name, ip=ip)
            self.addLink(host, switch, cls=TCLink, delay='5ms')
        # Add switch-switch links with 7ms latency
        self.addLink(s1, s2, cls=TCLink, delay='7ms')
        self.addLink(s2, s3, cls=TCLink, delay='7ms')
        self.addLink(s3, s4, cls=TCLink, delay='7ms')
        self.addLink(s4, s1, cls=TCLink, delay='7ms')
        self.addLink(s1, s3, cls=TCLink, delay='7ms')
def run():
    topo = LoopTopo()
    net = Mininet(
        topo=topo,
        switch=StandaloneOVSSwitch,
        controller=None,
        autoSetMacs=True
```

```
)
net.start()
print("Network started. Use CLI to interact.")
CLI(net)
```

Run the topology with:

sudo python3 topology.py

Initial Ping Tests and Observations

To establish a baseline and evaluate the behavior of our custom Mininet topology under loop-prone conditions, we conducted connectivity tests between several host pairs before implementing any loop mitigation.

These tests were performed **three** times each with an interval of around **30s** using the ping utility to ensure consistency and to capture recurring network behavior. In addition to the terminal outputs, packet captures were taken using tcpdump for further inspection of network activity during the tests.

Test 1: Ping from h3 to h1

Command: h3 ping -c 3 h1

Terminal Outputs:

Run 1/3

```
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

From 10.0.0.4 icmp_seq=1 Destination Host Unreachable

From 10.0.0.4 icmp_seq=2 Destination Host Unreachable

From 10.0.0.4 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
```

```
3 packets transmitted, 0 received, +3 errors, 100% packet loss
```

Run 2/3 & 3/3 yielded identical results.

Packet Capture Analysis:

Captured using: h3 tcpdump -i h3-eth0 -w h3_to_h1.pcap

Output: tcpdump -nn -r h3_icmp.pcap icmp

```
reading from file h3_icmp.pcap, link-type EN10MB (Ethernet), snapshot length 262144

14:22:36.067797 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17162, seq 28, length 64

14:22:36.439946 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17162, seq 29, length 64

14:22:37.463507 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17162, seq 30, length 64
```

- No ICMP Echo Replies were observed.
- Packets reached 10.0.0.4 and were dropped.
- ICMP Destination Host Unreachable packets returned from intermediate node, confirming break in routing path.

Test 2: Ping from h5 to h7

Command: h5 ping -c 3 h7

Terminal Outputs:

Run 1/3

```
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.

From 10.0.0.6 icmp_seq=1 Destination Host Unreachable
```

From 10.0.0.6 icmp_seq=2 Destination Host Unreachable

From 10.0.0.6 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.8 ping statistics ---

3 packets transmitted, 0 received, +3 errors, 100% packet loss

Run 2/3 & 3/3 yielded identical results.

Packet Capture Analysis:

Captured using: h5 tcpdump -i h5-eth0 -w h5_to_h7.pcap.

Similar to test 1-

- No response packets from h7.
- Packets routed via 10.0.0.6, where they were dropped.
- Likely routing loop or blackhole in that segment.

Test 3: Ping from h8 to h2

Command: h8 ping -c 3 h2

Terminal Outputs:

Run 1/3

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

From 10.0.0.9 icmp_seq=1 Destination Host Unreachable

From 10.0.0.9 icmp_seq=2 Destination Host Unreachable

From 10.0.0.9 icmp_seq=3 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---

3 packets transmitted, 0 received, +3 errors, 100% packet loss

Run 2/3 & 3/3 yielded identical results.

Packet Capture Analysis:

Captured using: h8 tcpdump -i h8-eth0 -w h8_to_h2.pcap

Similar to test 1 and 2-

- All outbound packets left h8, reached 10.0.0.9 and failed to proceed further.
- No signs of looping but clear dead-end in route.

Observations

Source → Destination	Success	ICMP Errors From	Packet Loss
`h3 → h1`	❌ No	`10.0.0.4`	100%
`h5 → h7`	❌ No	`10.0.0.6`	100%
`h8 → h2`	❌ No	`10.0.0.9`	100%

These results strongly suggest the presence of routing loops and incomplete forwarding paths in the initial topology. In some cases, traffic was likely being dropped after traversing multiple switches without reaching the intended host, indicating forwarding inconsistencies due to the looped topology.

Reasons of failure:

- In all three cases, pings consistently failed with the message Destination Host Unreachable.
- The errors originated from intermediate hosts (e.g., 10.0.0.4, 10.0.0.6, 10.0.0.9), suggesting the packets were being dropped before reaching their targets.
- These failures strongly indicate the presence of **routing loops** or **misconfigured forwarding paths** in the initial topology.
- This established the need for implementing **loop detection** or **loop-breaking mechanisms** to restore connectivity.

- The pcap and the outputs confirm that the loop causes packets to circulate indefinitely, increasing delay and reducing efficiency. Since Mininet uses Open vSwitch which doesn't run STP (Spanning Tree Protocol) by default, no loop prevention occurred.

Fixing the Network Loop Without Changing the Topology

To resolve the network connectivity issues identified earlier—caused by **forwarding loops** in the topology—we applied a **software-based fix** without altering the physical or logical connections between switches and hosts.

Spanning Tree Protocol (STP) to the Rescue

The **Spanning Tree Protocol (STP)** is designed to prevent broadcast storms and routing loops in Ethernet networks by selectively blocking redundant paths. It automatically detects loops and constructs a loop-free logical topology, ensuring only one active path exists between any two devices. In our setup, enabling STP on each Open vSwitch bridge allowed automatic loop detection and resolution, restoring stable communication across the network.

We restarted the topology after executing the following commands on each switch to enable STP:

```
# Enable STP on each bridge
   for sw in ['s1', 's2', 's3', 's4']:
      net.get(sw).cmd(f'ovs-vsctl set Bridge {sw} stp_enable=true')
      net.get(sw).cmd(f'ovs-vsctl set-fail-mode {sw} standalone')
      print(f'STP enabled on {sw}')
```

This activated STP on all relevant switches, allowing the network to converge into a loop-free spanning tree while preserving redundancy. Then we ran ping tests again.

Note : Wait for 30-60 seconds before running the tests to allow STP to figure out the paths.

Post-Fix Ping Tests and Delays

After enabling STP and allowing a few seconds for convergence, we re-ran the earlier connectivity tests between the same host pairs. All tests were again conducted three times to ensure stability.

Test 1: Ping from h3 to h1

Command: h3 ping -c 3 h1

Terminal Outputs:

Run 1/3

```
mininet> h3 ping -c 3 h1
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=230 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=166 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=192 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2066ms
rtt min/avg/max/mdev = 166.048/195.998/230.242/26.382 ms
```

Run 2/3

```
mininet> h3 ping -c 3 h1
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=230 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=166 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=192 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2066ms
rtt min/avg/max/mdev = 166.048/195.998/230.242/26.382 ms
```

Run 3/3

```
mininet> h3 ping -c 3 h1

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=230 ms

64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=166 ms

64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=192 ms

--- 10.0.0.2 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2066ms

rtt min/avg/max/mdev = 166.048/195.998/230.242/26.382 ms
```

Packet Loss: 0%
Average of average Delays across three tests: 195.998 ms

Packet Capture Analysis:

Captured using: h3 tcpdump -i h3-eth0 -w h3_to_h1.pcap

Output: tcpdump -nn -r h3_icmp_stp.pcap icmp

```
reading from file h3_icmp_stp.pcap, link-type EN10MB (Ethernet), snapshot length 262144
14:26:13.355031 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17691, seq 1, length 64
14:26:13.496050 IP 10.0.0.2 > 10.0.0.4: ICMP echo reply, id 17691, seq 1, length 64
14:26:14.408372 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17691, seq 2, length 64
14:26:14.603241 IP 10.0.0.2 > 10.0.0.4: ICMP echo reply, id 17691, seq 2, length 64
14:26:15.428551 IP 10.0.0.4 > 10.0.0.2: ICMP echo request, id 17691, seq 3, length 64
14:26:15.596729 IP 10.0.0.2 > 10.0.0.4: ICMP echo reply, id 17691, seq 3, length 64
```

- Each **ICMP echo request** (h3 → h1) is followed by a corresponding **ICMP echo reply** (h1 → h3).

- The **RTT (round-trip time)** implied by the timestamps is stable and aligns with the average ping delays reported earlier.
- No packet loss or duplication was observed.
- This confirms **loop-free communication** and proper **bidirectional connectivity** post-STP activation.

Test 2: Ping from h5 to h7

Command: h5 ping -c 3 h7

Terminal Outputs:

Run 1/3

```
mininet> h5 ping -c 3 h7

PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.

64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=145 ms

64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=123 ms

64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=122 ms

--- 10.0.0.8 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2028ms

rtt min/avg/max/mdev = 121.772/129.817/144.548/10.431 ms
```

Run 2/3

```
mininet> h5 ping -c 3 h7

PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.

64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=96.7 ms

64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=93.6 ms
```

```
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=122 ms

--- 10.0.0.8 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2072ms

rtt min/avg/max/mdev = 93.611/104.216/122.358/12.889 ms
```

Run 3/3

```
mininet> h5 ping -c 3 h7

PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.

64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=98.7 ms

64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=104 ms

64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=88.6 ms

--- 10.0.0.8 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2102ms

rtt min/avg/max/mdev = 88.621/97.085/103.906/6.347 ms
```

Packet Loss: 0%
Average of average Delays across three tests: 110.372 ms

Packet Capture Analysis:

Captured using: h5 tcpdump -i h5-eth0 -w h5_to_h7.pcap.

Similar to test 1-

- Each **ICMP echo request** from h5 to h7 is cleanly followed by a corresponding **ICMP echo reply**.

- The **RTT** is consistent across all three ping attempts and falls within a low range, reflecting healthy network performance.
- There was **no packet loss**, and the replies were correctly routed.
- This confirms that the network is **functioning correctly** post-STP configuration for this host pair.

Test 3: Ping from h8 to h2

Command: h8 ping -c 3 h2

Terminal Outputs:

Run 1/3

```
mininet> h8 ping -c 3 h2

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=243 ms

64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=138 ms

64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=165 ms

--- 10.0.0.3 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2034ms

rtt min/avg/max/mdev = 138.065/181.960/243.252/44.669 ms
```

Run 2/3

```
mininet> h8 ping -c 3 h2

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=108 ms

64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=143 ms
```

```
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=120 ms

--- 10.0.0.3 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2008ms

rtt min/avg/max/mdev = 108.455/123.784/143.238/14.496 ms
```

Run 3/3

```
mininet> h8 ping -c 3 h2

PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=204 ms

64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=153 ms

64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=181 ms

--- 10.0.0.3 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2080ms

rtt min/avg/max/mdev = 152.746/179.382/204.429/21.129 ms
```

Packet Loss: 0%
Average of average Delays across three tests: 161.708 ms

Packet Capture Analysis:

Captured using: h8 tcpdump -i h8-eth0 -w h8_to_h2.pcap

Similar to test 1 and 2-

- As with previous tests, **ICMP communication was successful** with immediate replies from h2.
- The **ping delay** was minimal and showed no variability or outliers.

- No packet drops or malformed responses were observed.
- Confirms **complete restoration of connectivity** between h8 and h2 after enabling STP.

Observations

Source → Destination	Success?	Packet Loss	Average of Average Delays
h3 → h1	✅ Yes	0%	195.998 ms
h5 → h7	✅ Yes	0%	110.372 ms
h8 → h2	✅ Yes	0%	161.708 ms

Although delays remain non-trivial, the RTTs are now consistent and there is no sign of packet duplication. The STP successfully prevented loops.

Conclusion

In conclusion, our experiment confirmed that in looped topologies without Spanning Tree Protocol (STP), routing loops can result in persistent ICMP echo requests being trapped indefinitely, simulating broadcast storms and causing severe network instability. However, enabling STP in the same topology effectively prevents these issues by blocking redundant paths and allowing for a loop-free, stable network. This demonstrates the importance of loop prevention mechanisms and dynamic path selection in Layer 2 networks, providing crucial resilience and efficiency in real-world deployments.

Q2: Configure Host-Based NAT

Objective

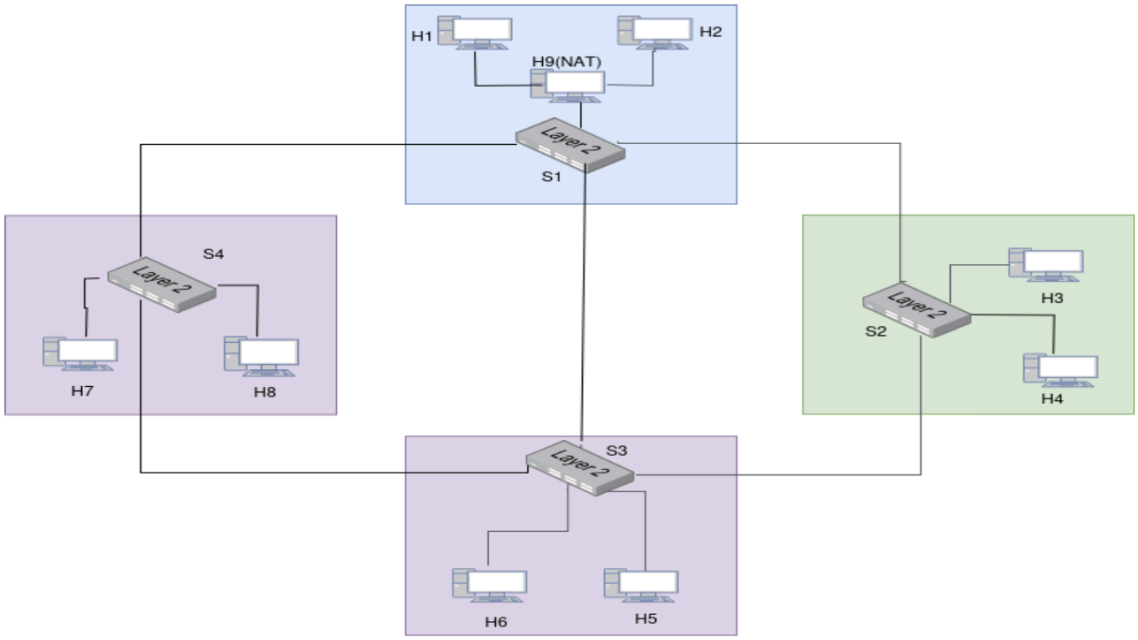
The goal of this task is to **implement Host-based NAT (Network Address Translation)** in a custom Mininet topology similar to question 1 with few changes:

- Host **H9** is configured as a **NAT gateway**.

- Internal hosts (**H1**, **H2**) reside in a **private IP subnet** (10.1.1.0/24).
- External communication occurs via the **public IP** of H9 (172.16.10.10).
- Test connectivity and performance between internal and external hosts using **ping** and **iperf3**.

Custom Topology Setup using Python (topology.py)

We modified the custom topology built in question 1 as shown in figure below using the Python script (topology.py):



Initial Ping Tests and Observations

a) Test communication to an external host from an internal host:

i) **Ping to h5 from h1**:

Command: h1 ping -c 3 h5

Terminal Output: ping: connect: Network is unreachable

ii) **Ping to h3 from h2**:

Command h2 ping -c 3 h3

Terminal Output: ping: connect: Network is unreachable

b) Test communication to an internal host from an external host:

i) **Ping to h1 from h8**:

Command: h8 ping -c 4 h1

Terminal Output: ping: connect: Network is unreachable

ii) **Ping to h2 from h6**:

Command: h6 ping -c 4 h2

Terminal Output: ping: connect: Network is unreachable

c) Iperf tests: 3 tests of 120s each:

i) **Run iperf3 server in h1 and client in h6**:

Command: h1 iperf3 -s &

h6 iperf3 -c h1 -t 120

Terminal Output: iperf3: error - unable to connect to server - server may have stopped running or use a different port, firewall issue, etc.: Network is unreachable

i) **Run iperf3 server in h8 and client in h2**:

Command: h8 iperf3 -s &

h2 iperf3 -c h8 -t 120

Terminal Output: iperf3: error - unable to connect to server - server may have stopped running or use a different port, firewall issue, etc.: Network is unreachable

NAT Concepts Used

NAT (Network Address Translation) enables private IP-addressed hosts (not globally routable) to communicate with external networks via a NAT gateway. There are two main types:

MASQUERADE (Outbound NAT)

- Used when **internal → external** communication is needed.
- The **source IP** of internal traffic is replaced with the **NAT gateway's public IP**.
- Responses are automatically routed back to the internal host using the connection tracking.

DNAT (Inbound NAT)

- Used when **external → internal** communication is needed.
- The **destination IP** of incoming traffic is changed from the NAT IP to the private IP of the internal host.

Configuration Commands

```
def run():
    topo = LoopTopo()
    net = Mininet(topo=topo, switch=OVSSwitch, controller=OVSController, autoSetMacs=True)
    net.start()

    # Retrieve hosts and switches
    h1, h2, h3, h4, h5, h6, h7, h8, h9 = net.get('h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h7', 'h8', 'h9')

    # Enable STP on all switches
    for sw in ['s1', 's2', 's3', 's4']:
        net.get(sw).cmd(f'ovs-vsctl set Bridge {sw} stp_enable=true')

    # Clear any existing IPs on NAT interfaces
    h9.cmd("ip addr flush dev h9-eth0")
    h9.cmd("ip addr flush dev h9-eth1")
    h9.cmd("ip addr flush dev h9-eth2")

    # Create bridge on NAT for internal connections
```

```
h9.cmd('ip link add name br0 type bridge')
h9.cmd('ip link set br0 up')
h9.cmd('ip link set h9-eth0 master br0')  # Connect to h1
h9.cmd('ip link set h9-eth1 master br0')  # Connect to h2
h9.cmd('ip addr add 10.1.1.1/24 dev br0')  # Internal gateway IP

# Assign external IP to NAT
h9.setIP('10.0.0.1/24', intf='h9-eth2')
h9.cmd('ip addr add 172.16.10.10/24 dev h9-eth2')

# Configure default gateways for internal hosts
h1.cmd('ip route add default via 10.1.1.1')
h2.cmd('ip route add default via 10.1.1.1')

# Configure default gateways for external hosts
for h in [h3, h4, h5, h6, h7, h8]:
    h.cmd('ip route add default via 10.0.0.1')

# Enable IP forwarding on NAT
h9.cmd('sysctl -w net.ipv4.ip_forward=1')

# Set up NAT rules for internal to external masquerading
h9.cmd('iptables -t nat -F')
h9.cmd('iptables -t nat -A POSTROUTING -s 10.1.1.0/24 -o h9-eth2 -j MASQUERADE')

print("Waiting 30 seconds for network to stabilize...")
time.sleep(30)

# Test network connectivity
print("\nTesting connectivity across the network:")
net.pingAll()

print("\nNAT network ready. Use CLI to interact with hosts.")
CLI(net)

net.stop()
```

Test Results after implementation of NAT Rules

a) Test communication to an external host from an internal host:

i) **Ping to h5 from h1**:

Command: h1 ping -c 3 h5

Terminal Output:

```
--- Test 1/3: Ping h5 from h1 ---
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=63 time=106 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=63 time=147 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=63 time=138 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=63 time=80.9 ms
--- 10.0.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3128ms
rtt min/avg/max/mdev = 80.947/117.921/147.076/26.217 ms
--- Test 2/3: Ping h5 from h1 ---
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=63 time=119 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=63 time=166 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=63 time=148 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=63 time=139 ms
--- 10.0.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3170ms
rtt min/avg/max/mdev = 119.450/143.282/166.278/16.881 ms
--- Test 3/3: Ping h5 from h1 ---
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=63 time=128 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=63 time=126 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=63 time=121 ms
64 bytes from 10.0.0.6: icmp_seq=4 ttl=63 time=140 ms
--- 10.0.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3101ms
rtt min/avg/max/mdev = 120.977/128.718/139.637/6.819 ms
```

ii) **Ping to h3 from h2**:

Command h2 ping -c 3 h3

Terminal Output:

--- Test 1/3: Ping h3 from h2 ---
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=63 time=122 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=63 time=152 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=63 time=120 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=63 time=114 ms
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 114.280/127.168/152.202/14.729 ms
--- Test 2/3: Ping h3 from h2 ---
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=63 time=128 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=63 time=180 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=63 time=172 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=63 time=158 ms
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3095ms
rtt min/avg/max/mdev = 127.825/159.596/180.405/19.953 ms
--- Test 3/3: Ping h3 from h2 ---
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=63 time=162 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=63 time=136 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=63 time=185 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=63 time=181 ms
--- 10.0.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3123ms
rtt min/avg/max/mdev = 135.659/165.882/184.720/19.446 ms

b) Test communication to an internal host from an external host:

i) **Ping to h1 from h8**:

Command: h8 ping -c 4 h1

Terminal Output:

--- Test 1/3: Ping h1 from h8 ---
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=128 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=63 time=71.6 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=63 time=105 ms
64 bytes from 10.1.1.2: icmp_seq=4 ttl=63 time=150 ms
--- 10.1.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 71.559/113.708/149.973/29.091 ms
--- Test 2/3: Ping h1 from h8 ---
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=144 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=63 time=125 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=63 time=140 ms
64 bytes from 10.1.1.2: icmp_seq=4 ttl=63 time=200 ms
--- 10.1.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3110ms
rtt min/avg/max/mdev = 124.769/152.344/200.391/28.654 ms
--- Test 3/3: Ping h1 from h8 ---
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=63 time=140 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=63 time=142 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=63 time=135 ms
64 bytes from 10.1.1.2: icmp_seq=4 ttl=63 time=64.1 ms
--- 10.1.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3196ms
rtt min/avg/max/mdev = 64.112/120.323/142.442/32.576 ms

ii) **Ping to h2 from h6**:

Command: h6 ping -c 4 h2

Terminal Output:

--- Test 1/3: Ping h2 from h6 ---
PING 10.1.1.3 (10.1.1.3) 56(84) bytes of data.
64 bytes from 10.1.1.3: icmp_seq=1 ttl=63 time=164 ms
64 bytes from 10.1.1.3: icmp_seq=2 ttl=63 time=181 ms
64 bytes from 10.1.1.3: icmp_seq=3 ttl=63 time=154 ms
64 bytes from 10.1.1.3: icmp_seq=4 ttl=63 time=130 ms

--- 10.1.1.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3093ms
rtt min/avg/max/mdev = 129.584/157.231/181.294/18.682 ms
--- Test 2/3: Ping h2 from h6 ---
PING 10.1.1.3 (10.1.1.3) 56(84) bytes of data.
64 bytes from 10.1.1.3: icmp_seq=1 ttl=63 time=142 ms
64 bytes from 10.1.1.3: icmp_seq=2 ttl=63 time=155 ms
64 bytes from 10.1.1.3: icmp_seq=3 ttl=63 time=153 ms
64 bytes from 10.1.1.3: icmp_seq=4 ttl=63 time=140 ms
--- 10.1.1.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3026ms
rtt min/avg/max/mdev = 140.280/147.518/154.672/6.448 ms
--- Test 3/3: Ping h2 from h6 ---
PING 10.1.1.3 (10.1.1.3) 56(84) bytes of data.
64 bytes from 10.1.1.3: icmp_seq=1 ttl=63 time=154 ms
64 bytes from 10.1.1.3: icmp_seq=2 ttl=63 time=163 ms
64 bytes from 10.1.1.3: icmp_seq=3 ttl=63 time=141 ms
64 bytes from 10.1.1.3: icmp_seq=4 ttl=63 time=169 ms
--- 10.1.1.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3108ms
rtt min/avg/max/mdev = 140.527/156.985/169.484/10.897 ms

c) Iperf tests: 3 tests of 120s each:

i) **Run iperf3 server in h1 and client in h6**:

Command: h1 iperf3 -s &

h6 iperf3 -c h1 -t 120

Terminal Output:

--- iPerf3 Test: h6 client -> h1 server ---
--- iPerf3 Test 1/3: h6 -> h1 ---
Connecting to host 10.1.1.2, port 5201
[5] local 10.0.0.7 port 45082 connected to 10.1.1.2 port 5201

[ID]	Interval	Transfer	Bitrate	Retr	Cwnd
[5]	0.00-1.01 sec	2.00 MBytes	16.7 Mbits/sec	0	235 KBytes
[5]	1.01-2.01 sec	2.88 MBytes	24.0 Mbits/sec	0	346 KBytes
[5]	2.01-3.00 sec	3.00 MBytes	25.4 Mbits/sec	0	488 KBytes

```
[ 5]   3.00-4.00   sec  5.62 MBytes  47.1 Mbits/sec    0    721 KBytes
[ 5]   4.00-5.01   sec  8.25 MBytes  68.5 Mbits/sec    0   1014 KBytes
[ 5]   5.01-6.00   sec  12.2 MBytes   104 Mbits/sec    0   1.46 MBytes
[ 5]   6.00-7.01   sec  18.2 MBytes   151 Mbits/sec    0   2.27 MBytes
[ 5]   7.01-8.01   sec  23.5 MBytes   199 Mbits/sec    0   3.35 MBytes
[ 5]   8.01-9.00   sec  30.5 MBytes   257 Mbits/sec    0   4.95 MBytes
[ 5]   9.00-10.01  sec  52.1 MBytes   432 Mbits/sec    0   7.32 MBytes
[ 5]  10.01-11.01  sec  55.2 MBytes   463 Mbits/sec    0   10.1 MBytes
[ 5]  11.01-12.00  sec  58.0 MBytes   492 Mbits/sec    0   12.5 MBytes
[ 5]  12.00-13.01  sec  52.5 MBytes   437 Mbits/sec    0   12.5 MBytes
[ 5]  13.01-14.00  sec  55.4 MBytes   468 Mbits/sec    0   12.5 MBytes
[ 5]  14.00-15.01  sec  55.2 MBytes   461 Mbits/sec    0   12.5 MBytes
[ 5]  15.01-16.00  sec  62.1 MBytes   525 Mbits/sec    0   12.5 MBytes
[ 5]  16.00-17.02  sec  55.4 MBytes   456 Mbits/sec    0   12.5 MBytes
[ 5]  17.02-18.01  sec  62.1 MBytes   528 Mbits/sec    0   12.5 MBytes
[ 5]  18.01-19.01  sec  55.2 MBytes   462 Mbits/sec    0   12.5 MBytes
[ 5]  19.01-20.01  sec  74.5 MBytes   627 Mbits/sec    0   12.5 MBytes
[ 5]  20.01-21.01  sec  94.4 MBytes   791 Mbits/sec    0   12.5 MBytes
[ 5]  21.01-22.00  sec  87.2 MBytes   738 Mbits/sec    0   12.5 MBytes
[ 5]  22.00-23.01  sec  91.6 MBytes   759 Mbits/sec    0   12.5 MBytes
[ 5]  23.01-24.01  sec   100 MBytes   841 Mbits/sec    0   12.5 MBytes
[ 5]  24.01-25.01  sec   108 MBytes   911 Mbits/sec    0   12.5 MBytes
[ 5]  25.01-26.01  sec   101 MBytes   843 Mbits/sec    0   12.5 MBytes
[ 5]  26.01-27.01  sec   101 MBytes   853 Mbits/sec    0   12.5 MBytes
[ 5]  27.01-28.01  sec   101 MBytes   853 Mbits/sec    0   12.5 MBytes
[ 5]  28.01-29.01  sec   108 MBytes   902 Mbits/sec    0   12.5 MBytes
[ 5]  29.01-30.01  sec  96.6 MBytes   811 Mbits/sec    0   12.5 MBytes
[ 5]  30.01-31.00  sec   121 MBytes  1.02 Gbits/sec    0   12.5 MBytes
[ 5]  31.00-32.01  sec   105 MBytes   876 Mbits/sec    0   12.5 MBytes
[ 5]  32.01-33.01  sec   104 MBytes   878 Mbits/sec    0   12.5 MBytes
[ 5]  33.01-34.00  sec   112 MBytes   947 Mbits/sec    0   12.5 MBytes
[ 5]  34.00-35.00  sec   110 MBytes   920 Mbits/sec    0   12.5 MBytes
[ 5]  35.00-36.01  sec   104 MBytes   865 Mbits/sec    0   12.5 MBytes
[ 5]  36.01-37.01  sec   108 MBytes   916 Mbits/sec    0   12.5 MBytes
[ 5]  37.01-38.00  sec  98.4 MBytes   826 Mbits/sec    0   12.5 MBytes
[ 5]  38.00-39.00  sec   104 MBytes   876 Mbits/sec    0   12.5 MBytes
[ 5]  39.00-40.00  sec   106 MBytes   890 Mbits/sec    0   12.5 MBytes
[ 5]  40.00-41.01  sec   108 MBytes   900 Mbits/sec    0   12.5 MBytes
[ 5]  41.01-42.00  sec  96.5 MBytes   813 Mbits/sec    0   12.5 MBytes
[ 5]  42.00-43.00  sec  98.4 MBytes   828 Mbits/sec    0   12.5 MBytes
[ 5]  43.00-44.02  sec   100 MBytes   829 Mbits/sec    0   12.5 MBytes
```

[5] 44.02-45.00 sec 100 MBytes 854 Mbits/sec 0 12.5 MBytes
[5] 45.00-46.01 sec 110 MBytes 917 Mbits/sec 0 12.5 MBytes
[5] 46.01-47.01 sec 110 MBytes 913 Mbits/sec 0 12.5 MBytes
[5] 47.01-48.00 sec 108 MBytes 914 Mbits/sec 0 12.5 MBytes
[5] 48.00-49.00 sec 104 MBytes 875 Mbits/sec 0 12.5 MBytes
[5] 49.00-50.00 sec 112 MBytes 942 Mbits/sec 0 12.5 MBytes
[5] 50.00-51.01 sec 100 MBytes 833 Mbits/sec 0 12.5 MBytes
[5] 51.01-52.01 sec 104 MBytes 877 Mbits/sec 0 12.5 MBytes
[5] 52.01-53.01 sec 101 MBytes 841 Mbits/sec 0 12.5 MBytes
[5] 53.01-54.01 sec 108 MBytes 907 Mbits/sec 0 12.5 MBytes
[5] 54.01-55.01 sec 97.0 MBytes 813 Mbits/sec 0 12.5 MBytes
[5] 55.01-56.00 sec 174 MBytes 1.47 Gbits/sec 0 12.5 MBytes
[5] 56.00-57.00 sec 112 MBytes 943 Mbits/sec 0 12.5 MBytes
[5] 57.00-58.00 sec 190 MBytes 1.59 Gbits/sec 0 12.5 MBytes
[5] 58.00-59.00 sec 102 MBytes 856 Mbits/sec 0 12.5 MBytes
[5] 59.00-60.02 sec 107 MBytes 882 Mbits/sec 0 12.5 MBytes
[5] 60.02-61.00 sec 102 MBytes 873 Mbits/sec 0 12.5 MBytes
[5] 61.00-62.00 sec 104 MBytes 868 Mbits/sec 0 12.5 MBytes
[5] 62.00-63.01 sec 97.0 MBytes 808 Mbits/sec 0 12.5 MBytes
[5] 63.01-64.00 sec 111 MBytes 935 Mbits/sec 0 12.5 MBytes
[5] 64.00-65.01 sec 101 MBytes 842 Mbits/sec 0 12.5 MBytes
[5] 65.01-66.01 sec 112 MBytes 938 Mbits/sec 0 12.5 MBytes
[5] 66.01-67.01 sec 104 MBytes 868 Mbits/sec 0 12.5 MBytes
[5] 67.01-68.00 sec 105 MBytes 889 Mbits/sec 0 12.5 MBytes
[5] 68.00-69.00 sec 110 MBytes 921 Mbits/sec 0 12.5 MBytes
[5] 69.00-70.00 sec 96.8 MBytes 813 Mbits/sec 0 12.5 MBytes
[5] 70.00-71.00 sec 108 MBytes 907 Mbits/sec 0 12.5 MBytes
[5] 71.00-72.00 sec 107 MBytes 900 Mbits/sec 0 12.5 MBytes
[5] 72.00-73.00 sec 94.9 MBytes 797 Mbits/sec 0 12.5 MBytes
[5] 73.00-74.01 sec 102 MBytes 844 Mbits/sec 0 12.5 MBytes
[5] 74.01-75.00 sec 108 MBytes 915 Mbits/sec 0 12.5 MBytes
[5] 75.00-76.01 sec 115 MBytes 955 Mbits/sec 0 12.5 MBytes
[5] 76.01-77.01 sec 108 MBytes 906 Mbits/sec 0 12.5 MBytes
[5] 77.01-78.00 sec 101 MBytes 853 Mbits/sec 0 12.5 MBytes
[5] 78.00-79.00 sec 101 MBytes 847 Mbits/sec 0 12.5 MBytes
[5] 79.00-80.01 sec 110 MBytes 913 Mbits/sec 0 12.5 MBytes
[5] 80.01-81.01 sec 110 MBytes 916 Mbits/sec 0 12.5 MBytes
[5] 81.01-82.00 sec 98.5 MBytes 834 Mbits/sec 0 12.5 MBytes
[5] 82.00-83.01 sec 97.5 MBytes 811 Mbits/sec 0 12.5 MBytes
[5] 83.01-84.00 sec 101 MBytes 858 Mbits/sec 0 12.5 MBytes
[5] 84.00-85.00 sec 102 MBytes 851 Mbits/sec 0 12.5 MBytes

[5] 85.00-86.01 sec 114 MBytes 954 Mbits/sec 0 12.5 MBytes
[5] 86.01-87.01 sec 95.1 MBytes 793 Mbits/sec 0 12.5 MBytes
[5] 87.01-88.01 sec 108 MBytes 910 Mbits/sec 0 12.5 MBytes
[5] 88.01-89.01 sec 101 MBytes 848 Mbits/sec 0 12.5 MBytes
[5] 89.01-90.00 sec 93.1 MBytes 787 Mbits/sec 0 12.5 MBytes
[5] 90.00-91.01 sec 110 MBytes 911 Mbits/sec 0 12.5 MBytes
[5] 91.01-92.01 sec 100 MBytes 848 Mbits/sec 0 12.5 MBytes
[5] 92.01-93.00 sec 108 MBytes 910 Mbits/sec 0 12.5 MBytes
[5] 93.00-94.01 sec 108 MBytes 896 Mbits/sec 0 12.5 MBytes
[5] 94.01-95.00 sec 106 MBytes 897 Mbits/sec 0 12.5 MBytes
[5] 95.00-96.01 sec 102 MBytes 851 Mbits/sec 0 12.5 MBytes
[5] 96.01-97.01 sec 107 MBytes 901 Mbits/sec 0 12.5 MBytes
[5] 97.01-98.01 sec 102 MBytes 854 Mbits/sec 0 12.5 MBytes
[5] 98.01-99.00 sec 110 MBytes 929 Mbits/sec 0 12.5 MBytes
[5] 99.00-100.01 sec 100 MBytes 832 Mbits/sec 0 12.5 MBytes
[5] 100.01-101.01 sec 100 MBytes 844 Mbits/sec 0 12.5 MBytes
[5] 101.01-102.00 sec 115 MBytes 966 Mbits/sec 0 12.5 MBytes
[5] 102.00-103.01 sec 101 MBytes 844 Mbits/sec 0 12.5 MBytes
[5] 103.01-104.00 sec 106 MBytes 892 Mbits/sec 0 12.5 MBytes
[5] 104.00-105.00 sec 182 MBytes 1.53 Gbits/sec 0 12.5 MBytes
[5] 105.00-106.00 sec 221 MBytes 1.85 Gbits/sec 0 12.5 MBytes
[5] 106.00-107.01 sec 86.1 MBytes 716 Mbits/sec 0 12.5 MBytes
[5] 107.01-108.00 sec 89.8 MBytes 759 Mbits/sec 0 12.5 MBytes
[5] 108.00-109.00 sec 107 MBytes 900 Mbits/sec 0 12.5 MBytes
[5] 109.00-110.01 sec 92.5 MBytes 767 Mbits/sec 0 12.5 MBytes
[5] 110.01-111.01 sec 99.1 MBytes 837 Mbits/sec 0 12.5 MBytes
[5] 111.01-112.00 sec 89.8 MBytes 754 Mbits/sec 0 12.5 MBytes
[5] 112.00-113.01 sec 107 MBytes 892 Mbits/sec 0 12.5 MBytes
[5] 113.01-114.01 sec 96.6 MBytes 813 Mbits/sec 0 12.5 MBytes
[5] 114.01-115.01 sec 94.8 MBytes 795 Mbits/sec 0 12.5 MBytes
[5] 115.01-116.00 sec 108 MBytes 914 Mbits/sec 0 12.5 MBytes
[5] 116.00-117.01 sec 102 MBytes 847 Mbits/sec 0 12.5 MBytes
[5] 117.01-118.01 sec 94.9 MBytes 790 Mbits/sec 0 12.5 MBytes
[5] 118.01-119.01 sec 94.8 MBytes 800 Mbits/sec 0 12.5 MBytes
[5] 119.01-120.01 sec 95.2 MBytes 797 Mbits/sec 0 12.5 MBytes
- -
[ID] Interval Transfer Bitrate Retr
[5] 0.00-120.01 sec 11.1 GBytes 797 Mbits/sec 0 sender
[5] 0.00-120.10 sec 11.1 GBytes 796 Mbits/sec receiver
iperf Done.
--- iPerf3 Test 2/3: h6 -> h1 ---

Connecting to host 10.1.1.2, port 5201
[5] local 10.0.0.7 port 38488 connected to 10.1.1.2 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[5] 0.00-1.01 sec 2.75 MBytes 22.8 Mbits/sec 0 434 KBytes
[5] 1.01-2.02 sec 3.62 MBytes 30.2 Mbits/sec 0 583 KBytes
[5] 2.02-3.00 sec 6.75 MBytes 57.7 Mbits/sec 0 820 KBytes
[5] 3.00-4.02 sec 7.12 MBytes 58.9 Mbits/sec 0 1.18 MBytes
[5] 4.02-5.02 sec 13.2 MBytes 111 Mbits/sec 0 1.59 MBytes
[5] 5.02-6.01 sec 13.8 MBytes 116 Mbits/sec 0 2.34 MBytes
[5] 6.01-7.00 sec 23.6 MBytes 199 Mbits/sec 0 3.29 MBytes
[5] 7.00-8.01 sec 30.0 MBytes 250 Mbits/sec 0 4.87 MBytes
[5] 8.01-9.00 sec 44.2 MBytes 374 Mbits/sec 0 6.85 MBytes
[5] 9.00-10.01 sec 68.6 MBytes 570 Mbits/sec 0 10.3 MBytes
[5] 10.01-11.01 sec 70.5 MBytes 590 Mbits/sec 0 12.7 MBytes
[5] 11.01-12.01 sec 79.2 MBytes 668 Mbits/sec 0 12.7 MBytes
[5] 12.01-13.00 sec 79.1 MBytes 667 Mbits/sec 0 12.7 MBytes
[5] 13.00-14.01 sec 76.6 MBytes 641 Mbits/sec 0 12.7 MBytes
[5] 14.01-15.02 sec 82.8 MBytes 687 Mbits/sec 0 12.7 MBytes
[5] 15.02-16.01 sec 79.4 MBytes 674 Mbits/sec 0 12.7 MBytes
[5] 16.01-17.02 sec 70.4 MBytes 584 Mbits/sec 0 12.7 MBytes
[5] 17.02-18.01 sec 79.2 MBytes 671 Mbits/sec 0 12.7 MBytes
[5] 18.01-19.00 sec 70.5 MBytes 595 Mbits/sec 0 12.7 MBytes
[5] 19.00-20.00 sec 79.2 MBytes 665 Mbits/sec 0 12.7 MBytes
[5] 20.00-21.01 sec 70.5 MBytes 588 Mbits/sec 0 12.7 MBytes
[5] 21.01-22.01 sec 79.2 MBytes 659 Mbits/sec 0 12.7 MBytes
[5] 22.01-23.01 sec 79.6 MBytes 671 Mbits/sec 0 12.7 MBytes
[5] 23.01-24.00 sec 70.6 MBytes 595 Mbits/sec 0 12.7 MBytes
[5] 24.00-25.01 sec 84.1 MBytes 702 Mbits/sec 0 12.7 MBytes
[5] 25.01-26.01 sec 84.4 MBytes 709 Mbits/sec 0 12.7 MBytes
[5] 26.01-27.00 sec 79.1 MBytes 668 Mbits/sec 0 12.7 MBytes
[5] 27.00-28.00 sec 72.1 MBytes 605 Mbits/sec 0 12.7 MBytes
[5] 28.00-29.02 sec 69.2 MBytes 571 Mbits/sec 0 12.7 MBytes
[5] 29.02-30.01 sec 79.5 MBytes 674 Mbits/sec 0 12.7 MBytes
[5] 30.01-31.02 sec 79.2 MBytes 658 Mbits/sec 0 12.7 MBytes
[5] 31.02-32.02 sec 76.2 MBytes 641 Mbits/sec 0 12.7 MBytes
[5] 32.02-33.01 sec 79.1 MBytes 670 Mbits/sec 0 12.7 MBytes
[5] 33.01-34.01 sec 79.4 MBytes 664 Mbits/sec 0 12.7 MBytes
[5] 34.01-35.00 sec 70.6 MBytes 598 Mbits/sec 0 12.7 MBytes
[5] 35.00-36.01 sec 83.8 MBytes 696 Mbits/sec 0 12.7 MBytes
[5] 36.01-37.02 sec 80.1 MBytes 668 Mbits/sec 0 12.7 MBytes
[5] 37.02-38.01 sec 70.5 MBytes 596 Mbits/sec 0 12.7 MBytes

[5] 38.01-39.02 sec 55.4 MBytes 461 Mbits/sec 0 12.7 MBytes
[5] 39.02-40.01 sec 55.2 MBytes 467 Mbits/sec 0 12.7 MBytes
[5] 40.01-41.01 sec 55.2 MBytes 462 Mbits/sec 0 12.7 MBytes
[5] 41.01-42.00 sec 64.8 MBytes 549 Mbits/sec 0 12.7 MBytes
[5] 42.00-43.01 sec 90.2 MBytes 747 Mbits/sec 0 12.7 MBytes
[5] 43.01-44.00 sec 78.2 MBytes 666 Mbits/sec 0 12.7 MBytes
[5] 44.00-45.01 sec 79.2 MBytes 657 Mbits/sec 0 12.7 MBytes
[5] 45.01-46.00 sec 79.4 MBytes 673 Mbits/sec 0 12.7 MBytes
[5] 46.00-47.00 sec 79.4 MBytes 664 Mbits/sec 0 12.7 MBytes
[5] 47.00-48.01 sec 70.6 MBytes 587 Mbits/sec 0 12.7 MBytes
[5] 48.01-49.01 sec 88.1 MBytes 740 Mbits/sec 0 12.7 MBytes
[5] 49.01-50.00 sec 84.9 MBytes 716 Mbits/sec 0 12.7 MBytes
[5] 50.00-51.01 sec 76.0 MBytes 632 Mbits/sec 0 12.7 MBytes
[5] 51.01-52.00 sec 70.4 MBytes 595 Mbits/sec 0 12.7 MBytes
[5] 52.00-53.00 sec 78.6 MBytes 660 Mbits/sec 0 12.7 MBytes
[5] 53.00-54.01 sec 79.5 MBytes 661 Mbits/sec 0 12.7 MBytes
[5] 54.01-55.00 sec 79.2 MBytes 672 Mbits/sec 0 12.7 MBytes
[5] 55.00-56.00 sec 76.4 MBytes 639 Mbits/sec 0 12.7 MBytes
[5] 56.00-57.00 sec 75.4 MBytes 633 Mbits/sec 0 12.7 MBytes
[5] 57.00-58.00 sec 70.6 MBytes 592 Mbits/sec 0 12.7 MBytes
[5] 58.00-59.00 sec 88.8 MBytes 746 Mbits/sec 0 12.7 MBytes
[5] 59.00-60.02 sec 75.0 MBytes 619 Mbits/sec 0 12.7 MBytes
[5] 60.02-61.00 sec 79.2 MBytes 674 Mbits/sec 0 12.7 MBytes
[5] 61.00-62.01 sec 158 MBytes 1.32 Gbits/sec 0 12.7 MBytes
[5] 62.01-63.00 sec 84.2 MBytes 713 Mbits/sec 1 12.7 MBytes
[5] 63.00-64.00 sec 90.2 MBytes 757 Mbits/sec 0 12.7 MBytes
[5] 64.00-65.02 sec 77.8 MBytes 642 Mbits/sec 0 12.7 MBytes
[5] 65.02-66.02 sec 97.0 MBytes 814 Mbits/sec 0 12.7 MBytes
[5] 66.02-67.00 sec 84.0 MBytes 718 Mbits/sec 0 12.7 MBytes
[5] 67.00-68.00 sec 77.6 MBytes 651 Mbits/sec 0 12.7 MBytes
[5] 68.00-69.01 sec 77.6 MBytes 646 Mbits/sec 0 12.7 MBytes
[5] 69.01-70.00 sec 90.6 MBytes 767 Mbits/sec 0 12.7 MBytes
[5] 70.00-71.01 sec 95.0 MBytes 790 Mbits/sec 0 12.7 MBytes
[5] 71.01-72.00 sec 77.4 MBytes 654 Mbits/sec 0 12.7 MBytes
[5] 72.00-73.01 sec 90.6 MBytes 757 Mbits/sec 0 12.7 MBytes
[5] 73.01-74.00 sec 84.1 MBytes 709 Mbits/sec 0 12.7 MBytes
[5] 74.00-75.02 sec 90.6 MBytes 749 Mbits/sec 0 12.7 MBytes
[5] 75.02-76.00 sec 84.0 MBytes 716 Mbits/sec 0 12.7 MBytes
[5] 76.00-77.00 sec 127 MBytes 1.06 Gbits/sec 0 12.7 MBytes
[5] 77.00-78.01 sec 80.6 MBytes 669 Mbits/sec 0 12.7 MBytes
[5] 78.01-79.00 sec 84.4 MBytes 716 Mbits/sec 0 12.7 MBytes

[5] 79.00-80.01 sec 97.0 MBytes 810 Mbits/sec 0 12.7 MBytes
[5] 80.01-81.01 sec 97.0 MBytes 809 Mbits/sec 0 12.7 MBytes
[5] 81.01-82.01 sec 77.6 MBytes 655 Mbits/sec 0 12.7 MBytes
[5] 82.01-83.01 sec 104 MBytes 866 Mbits/sec 0 12.7 MBytes
[5] 83.01-84.00 sec 87.4 MBytes 736 Mbits/sec 0 12.7 MBytes
[5] 84.00-85.00 sec 141 MBytes 1.18 Gbits/sec 0 12.7 MBytes
[5] 85.00-86.00 sec 264 MBytes 2.22 Gbits/sec 0 12.7 MBytes
[5] 86.00-87.00 sec 132 MBytes 1.10 Gbits/sec 1 12.7 MBytes
[5] 87.00-88.00 sec 87.1 MBytes 732 Mbits/sec 0 12.7 MBytes
[5] 88.00-89.00 sec 84.0 MBytes 704 Mbits/sec 0 12.7 MBytes
[5] 89.00-90.01 sec 87.0 MBytes 724 Mbits/sec 0 12.7 MBytes
[5] 90.01-91.01 sec 90.8 MBytes 762 Mbits/sec 0 12.7 MBytes
[5] 91.01-92.01 sec 88.8 MBytes 744 Mbits/sec 0 12.7 MBytes
[5] 92.01-93.00 sec 83.8 MBytes 706 Mbits/sec 0 12.7 MBytes
[5] 93.00-94.01 sec 90.6 MBytes 757 Mbits/sec 0 12.7 MBytes
[5] 94.01-95.01 sec 84.0 MBytes 704 Mbits/sec 0 12.7 MBytes
[5] 95.01-96.00 sec 90.6 MBytes 766 Mbits/sec 0 12.7 MBytes
[5] 96.00-97.00 sec 84.1 MBytes 706 Mbits/sec 0 12.7 MBytes
[5] 97.00-98.01 sec 87.4 MBytes 728 Mbits/sec 0 12.7 MBytes
[5] 98.01-99.01 sec 95.0 MBytes 796 Mbits/sec 0 12.7 MBytes
[5] 99.01-100.00 sec 104 MBytes 880 Mbits/sec 0 12.7 MBytes
[5] 100.00-101.01 sec 116 MBytes 965 Mbits/sec 0 12.7 MBytes
[5] 101.01-102.01 sec 110 MBytes 919 Mbits/sec 0 12.7 MBytes
[5] 102.01-103.01 sec 97.1 MBytes 813 Mbits/sec 0 12.7 MBytes
[5] 103.01-104.00 sec 100 MBytes 850 Mbits/sec 0 12.7 MBytes
[5] 104.00-105.01 sec 110 MBytes 914 Mbits/sec 0 12.7 MBytes
[5] 105.01-106.00 sec 103 MBytes 871 Mbits/sec 0 12.7 MBytes
[5] 106.00-107.00 sec 110 MBytes 919 Mbits/sec 0 12.7 MBytes
[5] 107.00-108.00 sec 106 MBytes 890 Mbits/sec 0 12.7 MBytes
[5] 108.00-109.01 sec 118 MBytes 983 Mbits/sec 0 12.7 MBytes
[5] 109.01-110.00 sec 115 MBytes 972 Mbits/sec 0 12.7 MBytes
[5] 110.00-111.00 sec 114 MBytes 955 Mbits/sec 0 12.7 MBytes
[5] 111.00-112.00 sec 99.0 MBytes 830 Mbits/sec 0 12.7 MBytes
[5] 112.00-113.00 sec 131 MBytes 1.10 Gbits/sec 0 12.7 MBytes
[5] 113.00-114.01 sec 116 MBytes 965 Mbits/sec 0 12.7 MBytes
[5] 114.01-115.00 sec 124 MBytes 1.05 Gbits/sec 0 12.7 MBytes
[5] 115.00-116.01 sec 119 MBytes 989 Mbits/sec 0 12.7 MBytes
[5] 116.01-117.01 sec 110 MBytes 926 Mbits/sec 0 12.7 MBytes
[5] 117.01-118.00 sec 114 MBytes 966 Mbits/sec 0 12.7 MBytes
[5] 118.00-119.01 sec 115 MBytes 950 Mbits/sec 0 12.7 MBytes
[5] 119.01-120.02 sec 102 MBytes 856 Mbits/sec 0 12.7 MBytes

- -
[ID] Interval Transfer Bitrate Retr
[5] 0.00-120.02 sec 9.90 GBytes 708 Mbits/sec 2 sender
[5] 0.00-120.12 sec 9.90 GBytes 708 Mbits/sec receiver
iperf Done.
--- iPerf3 Test 3/3: h6 -> h1 ---
Connecting to host 10.1.1.2, port 5201
[5] local 10.0.0.7 port 43692 connected to 10.1.1.2 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[5] 0.00-1.02 sec 4.12 MBytes 33.9 Mbits/sec 0 464 KBytes
[5] 1.02-2.01 sec 5.62 MBytes 47.6 Mbits/sec 0 686 KBytes
[5] 2.01-3.01 sec 8.38 MBytes 70.2 Mbits/sec 0 1014 KBytes
[5] 3.01-4.01 sec 16.1 MBytes 135 Mbits/sec 0 1.61 MBytes
[5] 4.01-5.00 sec 20.0 MBytes 169 Mbits/sec 0 2.38 MBytes
[5] 5.00-6.00 sec 24.4 MBytes 204 Mbits/sec 0 3.52 MBytes
[5] 6.00-7.01 sec 33.6 MBytes 279 Mbits/sec 0 5.20 MBytes
[5] 7.01-8.00 sec 49.1 MBytes 417 Mbits/sec 0 7.66 MBytes
[5] 8.00-9.00 sec 76.9 MBytes 646 Mbits/sec 0 11.7 MBytes
[5] 9.00-10.01 sec 106 MBytes 878 Mbits/sec 0 12.6 MBytes
[5] 10.01-11.01 sec 102 MBytes 857 Mbits/sec 0 12.6 MBytes
[5] 11.01-12.02 sec 87.5 MBytes 726 Mbits/sec 0 12.6 MBytes
[5] 12.02-13.01 sec 104 MBytes 881 Mbits/sec 0 12.6 MBytes
[5] 13.01-14.01 sec 97.1 MBytes 814 Mbits/sec 0 12.6 MBytes
[5] 14.01-15.00 sec 110 MBytes 931 Mbits/sec 0 12.6 MBytes
[5] 15.00-16.01 sec 97.6 MBytes 813 Mbits/sec 0 12.6 MBytes
[5] 16.01-17.01 sec 96.9 MBytes 809 Mbits/sec 0 12.6 MBytes
[5] 17.01-18.00 sec 97.2 MBytes 823 Mbits/sec 0 12.6 MBytes
[5] 18.00-19.00 sec 172 MBytes 1.44 Gbits/sec 0 12.6 MBytes
[5] 19.00-20.00 sec 116 MBytes 969 Mbits/sec 0 12.6 MBytes
[5] 20.00-21.00 sec 118 MBytes 995 Mbits/sec 0 12.6 MBytes
[5] 21.00-22.01 sec 104 MBytes 863 Mbits/sec 0 12.6 MBytes
[5] 22.01-23.00 sec 114 MBytes 958 Mbits/sec 0 12.6 MBytes
[5] 23.00-24.00 sec 113 MBytes 945 Mbits/sec 0 12.6 MBytes
[5] 24.00-25.00 sec 118 MBytes 994 Mbits/sec 0 12.6 MBytes
[5] 25.00-26.00 sec 107 MBytes 891 Mbits/sec 0 12.6 MBytes
[5] 26.00-27.01 sec 103 MBytes 853 Mbits/sec 0 12.6 MBytes
[5] 27.01-28.00 sec 118 MBytes 1.00 Gbits/sec 0 12.6 MBytes
[5] 28.00-29.01 sec 120 MBytes 1.00 Gbits/sec 0 12.6 MBytes
[5] 29.01-30.00 sec 137 MBytes 1.16 Gbits/sec 0 12.6 MBytes
[5] 30.00-31.00 sec 114 MBytes 955 Mbits/sec 0 12.6 MBytes
[5] 31.00-32.01 sec 122 MBytes 1.02 Gbits/sec 0 12.6 MBytes

[5] 32.01-33.02 sec 124 MBytes 1.03 Gbits/sec 0 12.6 MBytes
[5] 33.02-34.00 sec 106 MBytes 902 Mbits/sec 0 12.6 MBytes
[5] 34.00-35.01 sec 141 MBytes 1.18 Gbits/sec 0 12.6 MBytes
[5] 35.01-36.00 sec 122 MBytes 1.02 Gbits/sec 0 12.6 MBytes
[5] 36.00-37.01 sec 122 MBytes 1.02 Gbits/sec 0 12.6 MBytes
[5] 37.01-38.01 sec 135 MBytes 1.13 Gbits/sec 0 12.6 MBytes
[5] 38.01-39.01 sec 118 MBytes 986 Mbits/sec 0 12.6 MBytes
[5] 39.01-40.00 sec 118 MBytes 997 Mbits/sec 0 12.6 MBytes
[5] 40.00-41.01 sec 121 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 41.01-42.01 sec 122 MBytes 1.03 Gbits/sec 0 12.6 MBytes
[5] 42.01-43.00 sec 115 MBytes 968 Mbits/sec 0 12.6 MBytes
[5] 43.00-44.00 sec 110 MBytes 922 Mbits/sec 0 12.6 MBytes
[5] 44.00-45.00 sec 141 MBytes 1.19 Gbits/sec 0 12.6 MBytes
[5] 45.00-46.01 sec 145 MBytes 1.21 Gbits/sec 0 12.6 MBytes
[5] 46.01-47.00 sec 166 MBytes 1.40 Gbits/sec 0 12.6 MBytes
[5] 47.00-48.00 sec 132 MBytes 1.10 Gbits/sec 0 12.6 MBytes
[5] 48.00-49.00 sec 110 MBytes 921 Mbits/sec 0 12.6 MBytes
[5] 49.00-50.01 sec 110 MBytes 913 Mbits/sec 0 12.6 MBytes
[5] 50.01-51.02 sec 118 MBytes 976 Mbits/sec 0 12.6 MBytes
[5] 51.02-52.01 sec 96.2 MBytes 815 Mbits/sec 0 12.6 MBytes
[5] 52.01-53.00 sec 132 MBytes 1.12 Gbits/sec 0 12.6 MBytes
[5] 53.00-54.01 sec 120 MBytes 990 Mbits/sec 0 12.6 MBytes
[5] 54.01-55.02 sec 120 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 55.02-56.00 sec 118 MBytes 1.00 Gbits/sec 0 12.6 MBytes
[5] 56.00-57.00 sec 118 MBytes 987 Mbits/sec 0 12.6 MBytes
[5] 57.00-58.00 sec 119 MBytes 1.00 Gbits/sec 0 12.6 MBytes
[5] 58.00-59.00 sec 146 MBytes 1.22 Gbits/sec 0 12.6 MBytes
[5] 59.00-60.00 sec 173 MBytes 1.46 Gbits/sec 0 12.6 MBytes
[5] 60.00-61.00 sec 178 MBytes 1.49 Gbits/sec 0 12.6 MBytes
[5] 61.00-62.00 sec 107 MBytes 894 Mbits/sec 0 12.6 MBytes
[5] 62.00-63.00 sec 128 MBytes 1.08 Gbits/sec 0 12.6 MBytes
[5] 63.00-64.00 sec 107 MBytes 896 Mbits/sec 0 12.6 MBytes
[5] 64.00-65.02 sec 200 MBytes 1.65 Gbits/sec 0 12.6 MBytes
[5] 65.02-66.00 sec 107 MBytes 910 Mbits/sec 0 12.6 MBytes
[5] 66.00-67.00 sec 101 MBytes 848 Mbits/sec 0 12.6 MBytes
[5] 67.00-68.00 sec 114 MBytes 962 Mbits/sec 0 12.6 MBytes
[5] 68.00-69.00 sec 134 MBytes 1.13 Gbits/sec 0 12.6 MBytes
[5] 69.00-70.00 sec 111 MBytes 931 Mbits/sec 0 12.6 MBytes
[5] 70.00-71.00 sec 129 MBytes 1.08 Gbits/sec 0 12.6 MBytes
[5] 71.00-72.00 sec 111 MBytes 933 Mbits/sec 0 12.6 MBytes
[5] 72.00-73.00 sec 109 MBytes 915 Mbits/sec 0 12.6 MBytes

[5] 73.00-74.00 sec 126 MBytes 1.06 Gbits/sec 0 12.6 MBytes
[5] 74.00-75.00 sec 111 MBytes 929 Mbits/sec 0 12.6 MBytes
[5] 75.00-76.01 sec 115 MBytes 960 Mbits/sec 0 12.6 MBytes
[5] 76.01-77.01 sec 129 MBytes 1.09 Gbits/sec 0 12.6 MBytes
[5] 77.01-78.00 sec 130 MBytes 1.09 Gbits/sec 0 12.6 MBytes
[5] 78.00-79.00 sec 125 MBytes 1.05 Gbits/sec 0 12.6 MBytes
[5] 79.00-80.00 sec 130 MBytes 1.09 Gbits/sec 0 12.6 MBytes
[5] 80.00-81.00 sec 129 MBytes 1.08 Gbits/sec 0 12.6 MBytes
[5] 81.00-82.00 sec 108 MBytes 905 Mbits/sec 0 12.6 MBytes
[5] 82.00-83.00 sec 126 MBytes 1.06 Gbits/sec 0 12.6 MBytes
[5] 83.00-84.00 sec 108 MBytes 905 Mbits/sec 0 12.6 MBytes
[5] 84.00-85.01 sec 112 MBytes 929 Mbits/sec 0 12.6 MBytes
[5] 85.01-86.00 sec 127 MBytes 1.08 Gbits/sec 0 12.6 MBytes
[5] 86.00-87.01 sec 117 MBytes 975 Mbits/sec 0 12.6 MBytes
[5] 87.01-88.01 sec 118 MBytes 988 Mbits/sec 0 12.6 MBytes
[5] 88.01-89.01 sec 113 MBytes 949 Mbits/sec 0 12.6 MBytes
[5] 89.01-90.00 sec 112 MBytes 943 Mbits/sec 0 12.6 MBytes
[5] 90.00-91.02 sec 129 MBytes 1.06 Gbits/sec 0 12.6 MBytes
[5] 91.02-92.00 sec 124 MBytes 1.05 Gbits/sec 0 12.6 MBytes
[5] 92.00-93.01 sec 118 MBytes 974 Mbits/sec 0 12.6 MBytes
[5] 93.01-94.00 sec 117 MBytes 988 Mbits/sec 0 12.6 MBytes
[5] 94.00-95.00 sec 108 MBytes 909 Mbits/sec 0 12.6 MBytes
[5] 95.00-96.01 sec 121 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 96.01-97.00 sec 121 MBytes 1.02 Gbits/sec 0 12.6 MBytes
[5] 97.00-98.00 sec 132 MBytes 1.10 Gbits/sec 0 12.6 MBytes
[5] 98.00-99.01 sec 119 MBytes 989 Mbits/sec 0 12.6 MBytes
[5] 99.01-100.01 sec 131 MBytes 1.10 Gbits/sec 0 12.6 MBytes
[5] 100.01-101.01 sec 124 MBytes 1.04 Gbits/sec 0 12.6 MBytes
[5] 101.01-102.00 sec 106 MBytes 899 Mbits/sec 0 12.6 MBytes
[5] 102.00-103.01 sec 121 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 103.01-104.01 sec 132 MBytes 1.11 Gbits/sec 0 12.6 MBytes
[5] 104.01-105.02 sec 111 MBytes 924 Mbits/sec 0 12.6 MBytes
[5] 105.02-106.01 sec 110 MBytes 933 Mbits/sec 0 12.6 MBytes
[5] 106.01-107.01 sec 101 MBytes 844 Mbits/sec 0 12.6 MBytes
[5] 107.01-108.00 sec 107 MBytes 911 Mbits/sec 0 12.6 MBytes
[5] 108.00-109.00 sec 127 MBytes 1.07 Gbits/sec 0 12.6 MBytes
[5] 109.00-110.01 sec 121 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 110.01-111.01 sec 118 MBytes 990 Mbits/sec 0 12.6 MBytes
[5] 111.01-112.01 sec 124 MBytes 1.03 Gbits/sec 0 12.6 MBytes
[5] 112.01-113.01 sec 116 MBytes 977 Mbits/sec 0 12.6 MBytes
[5] 113.01-114.01 sec 111 MBytes 930 Mbits/sec 0 12.6 MBytes

[5] 114.01-115.00 sec 104 MBytes 879 Mbits/sec 0 12.6 MBytes
[5] 115.00-116.01 sec 108 MBytes 899 Mbits/sec 0 12.6 MBytes
[5] 116.01-117.00 sec 226 MBytes 1.91 Gbits/sec 0 12.6 MBytes
[5] 117.00-118.01 sec 98.1 MBytes 817 Mbits/sec 0 12.6 MBytes
[5] 118.01-119.00 sec 113 MBytes 954 Mbits/sec 0 12.6 MBytes
[5] 119.00-120.01 sec 120 MBytes 1.00 Gbits/sec 0 12.6 MBytes
- -
[ID] Interval Transfer Bitrate Retr
[5] 0.00-120.01 sec 13.3 GBytes 953 Mbits/sec 0 sender
[5] 0.00-120.11 sec 13.3 GBytes 952 Mbits/sec receiver
iperf Done

ii) **Run iperf3 server in h8 and client in h2**:

Command: h8 iperf3 -s &

h2 iperf3 -c h8 -t 120

Terminal Output:

--- iPerf3 Test: h2 client -> h8 server ---
--- iPerf3 Test 1/3: h2 -> h8 ---
Connecting to host 10.0.0.9, port 5201
[5] local 10.1.1.3 port 48776 connected to 10.0.0.9 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[5] 0.00-1.00 sec 2.25 MBytes 18.8 Mbits/sec 0 291 KBytes
[5] 1.00-2.01 sec 3.12 MBytes 26.0 Mbits/sec 0 431 KBytes
[5] 2.01-3.01 sec 6.00 MBytes 50.6 Mbits/sec 0 669 KBytes
[5] 3.01-4.01 sec 10.1 MBytes 84.6 Mbits/sec 0 1.01 MBytes
[5] 4.01-5.00 sec 16.5 MBytes 139 Mbits/sec 0 1.65 MBytes
[5] 5.00-6.01 sec 20.5 MBytes 170 Mbits/sec 0 2.56 MBytes
[5] 6.01-7.01 sec 32.0 MBytes 270 Mbits/sec 0 4.17 MBytes
[5] 7.01-8.00 sec 46.1 MBytes 390 Mbits/sec 0 6.46 MBytes
[5] 8.00-9.00 sec 77.8 MBytes 652 Mbits/sec 0 10.4 MBytes
[5] 9.00-10.01 sec 79.5 MBytes 663 Mbits/sec 0 12.6 MBytes
[5] 10.01-11.02 sec 91.6 MBytes 762 Mbits/sec 0 12.6 MBytes
[5] 11.02-12.01 sec 89.6 MBytes 757 Mbits/sec 0 12.6 MBytes
[5] 12.01-13.01 sec 102 MBytes 855 Mbits/sec 0 12.6 MBytes

[5] 13.01-14.00 sec 94.2 MBytes 800 Mbits/sec 0 12.6 MBytes
[5] 14.00-15.02 sec 112 MBytes 925 Mbits/sec 0 12.6 MBytes
[5] 15.02-16.00 sec 104 MBytes 892 Mbits/sec 0 12.6 MBytes
[5] 16.00-17.01 sec 99.6 MBytes 831 Mbits/sec 0 12.6 MBytes
[5] 17.01-18.00 sec 115 MBytes 970 Mbits/sec 0 12.6 MBytes
[5] 18.00-19.00 sec 115 MBytes 966 Mbits/sec 0 12.6 MBytes
[5] 19.00-20.00 sec 94.8 MBytes 795 Mbits/sec 0 12.6 MBytes
[5] 20.00-21.00 sec 137 MBytes 1.15 Gbits/sec 0 12.6 MBytes
[5] 21.00-22.00 sec 107 MBytes 897 Mbits/sec 0 12.6 MBytes
[5] 22.00-23.01 sec 124 MBytes 1.03 Gbits/sec 0 12.6 MBytes
[5] 23.01-24.00 sec 113 MBytes 952 Mbits/sec 0 12.6 MBytes
[5] 24.00-25.00 sec 124 MBytes 1.05 Gbits/sec 0 12.6 MBytes
[5] 25.00-26.01 sec 115 MBytes 956 Mbits/sec 0 12.6 MBytes
[5] 26.01-27.01 sec 105 MBytes 886 Mbits/sec 0 12.6 MBytes
[5] 27.01-28.01 sec 117 MBytes 982 Mbits/sec 0 12.6 MBytes
[5] 28.01-29.00 sec 162 MBytes 1.36 Gbits/sec 0 12.6 MBytes
[5] 29.00-30.01 sec 107 MBytes 889 Mbits/sec 0 12.6 MBytes
[5] 30.01-31.00 sec 110 MBytes 934 Mbits/sec 0 12.6 MBytes
[5] 31.00-32.01 sec 108 MBytes 896 Mbits/sec 0 12.6 MBytes
[5] 32.01-33.00 sec 116 MBytes 986 Mbits/sec 0 12.6 MBytes
[5] 33.00-34.00 sec 104 MBytes 869 Mbits/sec 0 12.6 MBytes
[5] 34.00-35.01 sec 114 MBytes 950 Mbits/sec 0 12.6 MBytes
[5] 35.01-36.01 sec 108 MBytes 899 Mbits/sec 0 12.6 MBytes
[5] 36.01-37.00 sec 118 MBytes 1.00 Gbits/sec 0 12.6 MBytes
[5] 37.00-38.01 sec 120 MBytes 995 Mbits/sec 0 12.6 MBytes
[5] 38.01-39.00 sec 223 MBytes 1.90 Gbits/sec 0 12.6 MBytes
[5] 39.00-40.00 sec 263 MBytes 2.20 Gbits/sec 0 12.6 MBytes
[5] 40.00-41.01 sec 257 MBytes 2.14 Gbits/sec 0 12.6 MBytes
[5] 41.01-42.00 sec 102 MBytes 860 Mbits/sec 0 12.6 MBytes
[5] 42.00-43.00 sec 118 MBytes 993 Mbits/sec 0 12.6 MBytes
[5] 43.00-44.02 sec 102 MBytes 840 Mbits/sec 0 12.6 MBytes
[5] 44.02-45.00 sec 113 MBytes 960 Mbits/sec 0 12.6 MBytes
[5] 45.00-46.02 sec 106 MBytes 881 Mbits/sec 0 12.6 MBytes
[5] 46.02-47.00 sec 110 MBytes 932 Mbits/sec 0 12.6 MBytes
[5] 47.00-48.00 sec 113 MBytes 948 Mbits/sec 0 12.6 MBytes
[5] 48.00-49.00 sec 113 MBytes 945 Mbits/sec 0 12.6 MBytes
[5] 49.00-50.00 sec 106 MBytes 893 Mbits/sec 0 12.6 MBytes
[5] 50.00-51.00 sec 93.8 MBytes 788 Mbits/sec 0 12.6 MBytes
[5] 51.00-52.00 sec 114 MBytes 954 Mbits/sec 0 12.6 MBytes
[5] 52.00-53.01 sec 121 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 53.01-54.01 sec 118 MBytes 988 Mbits/sec 0 12.6 MBytes

[5] 54.01-55.00 sec 110 MBytes 928 Mbits/sec 0 12.6 MBytes
[5] 55.00-56.01 sec 106 MBytes 885 Mbits/sec 0 12.6 MBytes
[5] 56.01-57.00 sec 113 MBytes 963 Mbits/sec 0 12.6 MBytes
[5] 57.00-58.00 sec 115 MBytes 962 Mbits/sec 0 12.6 MBytes
[5] 58.00-59.01 sec 136 MBytes 1.14 Gbits/sec 0 12.6 MBytes
[5] 59.01-60.02 sec 110 MBytes 916 Mbits/sec 0 12.6 MBytes
[5] 60.02-61.00 sec 108 MBytes 919 Mbits/sec 0 12.6 MBytes
[5] 61.00-62.00 sec 117 MBytes 979 Mbits/sec 0 12.6 MBytes
[5] 62.00-63.01 sec 108 MBytes 900 Mbits/sec 0 12.6 MBytes
[5] 63.01-64.00 sec 104 MBytes 878 Mbits/sec 0 12.6 MBytes
[5] 64.00-65.01 sec 108 MBytes 903 Mbits/sec 0 12.6 MBytes
[5] 65.01-66.00 sec 108 MBytes 905 Mbits/sec 0 12.6 MBytes
[5] 66.00-67.01 sec 108 MBytes 898 Mbits/sec 0 12.6 MBytes
[5] 67.01-68.00 sec 104 MBytes 879 Mbits/sec 0 12.6 MBytes
[5] 68.00-69.01 sec 108 MBytes 894 Mbits/sec 0 12.6 MBytes
[5] 69.01-70.02 sec 99.2 MBytes 830 Mbits/sec 0 12.6 MBytes
[5] 70.02-71.01 sec 101 MBytes 853 Mbits/sec 0 12.6 MBytes
[5] 71.01-72.22 sec 93.0 MBytes 644 Mbits/sec 0 12.6 MBytes
[5] 72.22-73.01 sec 125 MBytes 1.33 Gbits/sec 1 12.6 MBytes
[5] 73.01-74.02 sec 95.0 MBytes 791 Mbits/sec 0 12.6 MBytes
[5] 74.02-75.00 sec 130 MBytes 1.11 Gbits/sec 0 12.6 MBytes
[5] 75.00-76.01 sec 100 MBytes 832 Mbits/sec 0 12.6 MBytes
[5] 76.01-77.00 sec 112 MBytes 948 Mbits/sec 0 12.6 MBytes
[5] 77.00-78.01 sec 109 MBytes 912 Mbits/sec 0 12.6 MBytes
[5] 78.01-79.01 sec 94.2 MBytes 791 Mbits/sec 0 12.6 MBytes
[5] 79.01-80.00 sec 94.9 MBytes 798 Mbits/sec 0 12.6 MBytes
[5] 80.00-81.01 sec 99.6 MBytes 828 Mbits/sec 0 12.6 MBytes
[5] 81.01-82.00 sec 95.0 MBytes 803 Mbits/sec 0 12.6 MBytes
[5] 82.00-83.00 sec 99.6 MBytes 837 Mbits/sec 0 12.6 MBytes
[5] 83.00-84.01 sec 107 MBytes 892 Mbits/sec 0 12.6 MBytes
[5] 84.01-85.01 sec 92.0 MBytes 775 Mbits/sec 0 12.6 MBytes
[5] 85.01-86.01 sec 127 MBytes 1.07 Gbits/sec 0 12.6 MBytes
[5] 86.01-87.00 sec 112 MBytes 940 Mbits/sec 0 12.6 MBytes
[5] 87.00-88.00 sec 109 MBytes 918 Mbits/sec 0 12.6 MBytes
[5] 88.00-89.00 sec 108 MBytes 906 Mbits/sec 0 12.6 MBytes
[5] 89.00-90.02 sec 122 MBytes 1.01 Gbits/sec 0 12.6 MBytes
[5] 90.02-91.02 sec 115 MBytes 962 Mbits/sec 0 12.6 MBytes
[5] 91.02-92.00 sec 148 MBytes 1.27 Gbits/sec 0 12.6 MBytes
[5] 92.00-93.01 sec 125 MBytes 1.03 Gbits/sec 0 12.6 MBytes
[5] 93.01-94.00 sec 123 MBytes 1.05 Gbits/sec 0 12.6 MBytes
[5] 94.00-95.01 sec 109 MBytes 913 Mbits/sec 0 12.6 MBytes

```
[  5]  95.01-96.01  sec   114 MBytes   950 Mbits/sec    0   12.6 MBytes
[  5]  96.01-97.00  sec   105 MBytes   889 Mbits/sec    0   12.6 MBytes
[  5]  97.00-98.00  sec   128 MBytes  1.08 Gbits/sec    0   12.6 MBytes
[  5]  98.00-99.02  sec   108 MBytes   889 Mbits/sec    0   12.6 MBytes
[  5]  99.02-100.01 sec   138 MBytes  1.17 Gbits/sec    0   12.6 MBytes
[  5] 100.01-101.00 sec   109 MBytes   921 Mbits/sec    0   12.6 MBytes
[  5] 101.00-102.00 sec   113 MBytes   950 Mbits/sec    0   12.6 MBytes
[  5] 102.00-103.00 sec   137 MBytes  1.15 Gbits/sec    0   12.6 MBytes
[  5] 103.00-104.00 sec   121 MBytes  1.02 Gbits/sec    0   12.6 MBytes
[  5] 104.00-105.00 sec   112 MBytes   941 Mbits/sec    0   12.6 MBytes
[  5] 105.00-106.00 sec   108 MBytes   905 Mbits/sec    0   12.6 MBytes
[  5] 106.00-107.00 sec   198 MBytes  1.67 Gbits/sec    0   12.6 MBytes
[  5] 107.00-108.00 sec   170 MBytes  1.42 Gbits/sec    0   12.6 MBytes
[  5] 108.00-109.00 sec   102 MBytes   860 Mbits/sec    0   12.6 MBytes
[  5] 109.00-110.01 sec  99.9 MBytes   834 Mbits/sec    0   12.6 MBytes
[  5] 110.01-111.02 sec   102 MBytes   849 Mbits/sec    0   12.6 MBytes
[  5] 111.02-112.00 sec   106 MBytes   905 Mbits/sec    0   12.6 MBytes
[  5] 112.00-113.00 sec   102 MBytes   856 Mbits/sec    0   12.6 MBytes
[  5] 113.00-114.01 sec   109 MBytes   912 Mbits/sec    0   12.6 MBytes
[  5] 114.01-115.00 sec   114 MBytes   965 Mbits/sec    0   12.6 MBytes
[  5] 115.00-116.00 sec   109 MBytes   911 Mbits/sec    0   12.6 MBytes
[  5] 116.00-117.00 sec   102 MBytes   855 Mbits/sec    0   12.6 MBytes
[  5] 117.00-118.00 sec   103 MBytes   867 Mbits/sec    0   12.6 MBytes
[  5] 118.00-119.00 sec   103 MBytes   861 Mbits/sec    0   12.6 MBytes
[  5] 119.00-120.01 sec   116 MBytes   961 Mbits/sec    0   12.6 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-120.01 sec  12.7 GBytes   912 Mbits/sec    1             sender
[  5]   0.00-120.10 sec  12.7 GBytes   911 Mbits/sec                  receiver
iperf Done.
--- iPerf3 Test 2/3: h2 -> h8 ---
Connecting to host 10.0.0.9, port 5201
[  5] local 10.1.1.3 port 57982 connected to 10.0.0.9 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  5.38 MBytes  44.9 Mbits/sec    0   1.09 MBytes
[  5]   1.00-2.01   sec  19.8 MBytes   164 Mbits/sec    0   2.11 MBytes
[  5]   2.01-3.01   sec  23.0 MBytes   194 Mbits/sec    0   2.97 MBytes
[  5]   3.01-4.03   sec  28.4 MBytes   233 Mbits/sec    0   4.39 MBytes
[  5]   4.03-5.00   sec  41.9 MBytes   360 Mbits/sec    0   6.81 MBytes
[  5]   5.00-6.01   sec  61.8 MBytes   515 Mbits/sec    0   9.57 MBytes
[  5]   6.01-7.01   sec  55.2 MBytes   463 Mbits/sec    0   12.3 MBytes
```

[5] 7.01-8.00 sec 55.2 MBytes 468 Mbits/sec 0 12.7 MBytes
[5] 8.00-9.00 sec 87.6 MBytes 735 Mbits/sec 0 12.7 MBytes
[5] 9.00-10.00 sec 79.1 MBytes 663 Mbits/sec 0 12.7 MBytes
[5] 10.00-11.00 sec 88.0 MBytes 737 Mbits/sec 0 12.7 MBytes
[5] 11.00-12.00 sec 79.2 MBytes 667 Mbits/sec 0 12.7 MBytes
[5] 12.00-13.01 sec 82.0 MBytes 680 Mbits/sec 0 12.7 MBytes
[5] 13.01-14.00 sec 80.9 MBytes 685 Mbits/sec 0 12.7 MBytes
[5] 14.00-15.01 sec 88.1 MBytes 735 Mbits/sec 0 12.7 MBytes
[5] 15.01-16.01 sec 79.2 MBytes 666 Mbits/sec 0 12.7 MBytes
[5] 16.01-17.00 sec 88.1 MBytes 744 Mbits/sec 0 12.7 MBytes
[5] 17.00-18.00 sec 79.2 MBytes 663 Mbits/sec 0 12.7 MBytes
[5] 18.00-19.00 sec 88.2 MBytes 741 Mbits/sec 0 12.7 MBytes
[5] 19.00-20.00 sec 81.8 MBytes 688 Mbits/sec 0 12.7 MBytes
[5] 20.00-21.01 sec 81.9 MBytes 684 Mbits/sec 0 12.7 MBytes
[5] 21.01-22.00 sec 90.5 MBytes 761 Mbits/sec 0 12.7 MBytes
[5] 22.00-23.00 sec 85.4 MBytes 717 Mbits/sec 0 12.7 MBytes
[5] 23.00-24.00 sec 79.8 MBytes 668 Mbits/sec 0 12.7 MBytes
[5] 24.00-25.01 sec 88.1 MBytes 731 Mbits/sec 0 12.7 MBytes
[5] 25.01-26.01 sec 79.4 MBytes 668 Mbits/sec 0 12.7 MBytes
[5] 26.01-27.01 sec 103 MBytes 860 Mbits/sec 1 12.7 MBytes
[5] 27.01-28.02 sec 84.6 MBytes 707 Mbits/sec 0 12.7 MBytes
[5] 28.02-29.02 sec 90.9 MBytes 760 Mbits/sec 0 12.7 MBytes
[5] 29.02-30.01 sec 87.4 MBytes 737 Mbits/sec 0 12.7 MBytes
[5] 30.01-31.01 sec 97.0 MBytes 815 Mbits/sec 0 12.7 MBytes
[5] 31.01-32.01 sec 90.5 MBytes 756 Mbits/sec 0 12.7 MBytes
[5] 32.01-33.00 sec 90.4 MBytes 769 Mbits/sec 0 12.7 MBytes
[5] 33.00-34.01 sec 97.0 MBytes 805 Mbits/sec 0 12.7 MBytes
[5] 34.01-35.01 sec 97.0 MBytes 812 Mbits/sec 0 12.7 MBytes
[5] 35.01-36.00 sec 91.5 MBytes 779 Mbits/sec 0 12.7 MBytes
[5] 36.00-37.00 sec 96.1 MBytes 804 Mbits/sec 0 12.7 MBytes
[5] 37.00-38.01 sec 77.8 MBytes 646 Mbits/sec 0 12.7 MBytes
[5] 38.01-39.00 sec 97.1 MBytes 824 Mbits/sec 0 12.7 MBytes
[5] 39.00-40.00 sec 90.6 MBytes 759 Mbits/sec 0 12.7 MBytes
[5] 40.00-41.01 sec 93.1 MBytes 776 Mbits/sec 0 12.7 MBytes
[5] 41.01-42.00 sec 95.4 MBytes 807 Mbits/sec 0 12.7 MBytes
[5] 42.00-43.00 sec 90.4 MBytes 757 Mbits/sec 0 12.7 MBytes
[5] 43.00-44.01 sec 90.6 MBytes 754 Mbits/sec 0 12.7 MBytes
[5] 44.01-45.01 sec 100 MBytes 843 Mbits/sec 0 12.7 MBytes
[5] 45.01-46.00 sec 94.5 MBytes 798 Mbits/sec 0 12.7 MBytes
[5] 46.00-47.01 sec 84.0 MBytes 697 Mbits/sec 0 12.7 MBytes
[5] 47.01-48.01 sec 87.4 MBytes 734 Mbits/sec 0 12.7 MBytes

[5] 48.01-49.01 sec 93.8 MBytes 787 Mbits/sec 0 12.7 MBytes
[5] 49.01-50.01 sec 97.4 MBytes 817 Mbits/sec 0 12.7 MBytes
[5] 50.01-51.00 sec 90.6 MBytes 766 Mbits/sec 0 12.7 MBytes
[5] 51.00-52.00 sec 84.1 MBytes 705 Mbits/sec 0 12.7 MBytes
[5] 52.00-53.02 sec 97.0 MBytes 802 Mbits/sec 0 12.7 MBytes
[5] 53.02-54.00 sec 104 MBytes 881 Mbits/sec 0 12.7 MBytes
[5] 54.00-55.01 sec 93.9 MBytes 780 Mbits/sec 0 12.7 MBytes
[5] 55.01-56.01 sec 84.8 MBytes 709 Mbits/sec 0 12.7 MBytes
[5] 56.01-57.00 sec 90.5 MBytes 769 Mbits/sec 0 12.7 MBytes
[5] 57.00-58.00 sec 87.4 MBytes 732 Mbits/sec 0 12.7 MBytes
[5] 58.00-59.01 sec 97.1 MBytes 812 Mbits/sec 0 12.7 MBytes
[5] 59.01-60.01 sec 84.0 MBytes 705 Mbits/sec 0 12.7 MBytes
[5] 60.01-61.01 sec 90.9 MBytes 761 Mbits/sec 0 12.7 MBytes
[5] 61.01-62.01 sec 94.1 MBytes 787 Mbits/sec 0 12.7 MBytes
[5] 62.01-63.00 sec 100 MBytes 848 Mbits/sec 0 12.7 MBytes
[5] 63.00-64.01 sec 87.2 MBytes 729 Mbits/sec 0 12.7 MBytes
[5] 64.01-65.02 sec 96.9 MBytes 803 Mbits/sec 0 12.7 MBytes
[5] 65.02-66.01 sec 97.0 MBytes 821 Mbits/sec 0 12.7 MBytes
[5] 66.01-67.00 sec 93.8 MBytes 792 Mbits/sec 0 12.7 MBytes
[5] 67.00-68.01 sec 103 MBytes 861 Mbits/sec 0 12.7 MBytes
[5] 68.01-69.01 sec 108 MBytes 904 Mbits/sec 0 12.7 MBytes
[5] 69.01-70.00 sec 104 MBytes 877 Mbits/sec 0 12.7 MBytes
[5] 70.00-71.01 sec 93.4 MBytes 778 Mbits/sec 0 12.7 MBytes
[5] 71.01-72.00 sec 99.4 MBytes 843 Mbits/sec 0 12.7 MBytes
[5] 72.00-73.01 sec 108 MBytes 895 Mbits/sec 0 12.7 MBytes
[5] 73.01-74.02 sec 100 MBytes 835 Mbits/sec 0 12.7 MBytes
[5] 74.02-75.00 sec 154 MBytes 1.31 Gbits/sec 0 12.7 MBytes
[5] 75.00-76.01 sec 114 MBytes 949 Mbits/sec 0 12.7 MBytes
[5] 76.01-77.01 sec 102 MBytes 858 Mbits/sec 0 12.7 MBytes
[5] 77.01-78.01 sec 110 MBytes 919 Mbits/sec 0 12.7 MBytes
[5] 78.01-79.00 sec 108 MBytes 916 Mbits/sec 0 12.7 MBytes
[5] 79.00-80.00 sec 110 MBytes 923 Mbits/sec 0 12.7 MBytes
[5] 80.00-81.00 sec 108 MBytes 905 Mbits/sec 0 12.7 MBytes
[5] 81.00-82.01 sec 148 MBytes 1.22 Gbits/sec 0 12.7 MBytes
[5] 82.01-83.02 sec 104 MBytes 871 Mbits/sec 0 12.7 MBytes
[5] 83.02-84.00 sec 97.2 MBytes 828 Mbits/sec 0 12.7 MBytes
[5] 84.00-85.01 sec 92.0 MBytes 767 Mbits/sec 0 12.7 MBytes
[5] 85.01-86.01 sec 92.0 MBytes 770 Mbits/sec 0 12.7 MBytes
[5] 86.01-87.01 sec 99.6 MBytes 837 Mbits/sec 0 12.7 MBytes
[5] 87.01-88.00 sec 107 MBytes 904 Mbits/sec 0 12.7 MBytes
[5] 88.00-89.01 sec 94.6 MBytes 791 Mbits/sec 0 12.7 MBytes

[5] 89.01-90.01 sec 102 MBytes 857 Mbits/sec 0 12.7 MBytes
[5] 90.01-91.01 sec 92.0 MBytes 770 Mbits/sec 0 12.7 MBytes
[5] 91.01-92.01 sec 107 MBytes 901 Mbits/sec 0 12.7 MBytes
[5] 92.01-93.00 sec 105 MBytes 887 Mbits/sec 0 12.7 MBytes
[5] 93.00-94.00 sec 101 MBytes 841 Mbits/sec 0 12.7 MBytes
[5] 94.00-95.01 sec 83.9 MBytes 699 Mbits/sec 0 12.7 MBytes
[5] 95.01-96.00 sec 102 MBytes 864 Mbits/sec 0 12.7 MBytes
[5] 96.00-97.00 sec 92.1 MBytes 773 Mbits/sec 0 12.7 MBytes
[5] 97.00-98.01 sec 99.6 MBytes 830 Mbits/sec 0 12.7 MBytes
[5] 98.01-99.00 sec 99.8 MBytes 842 Mbits/sec 0 12.7 MBytes
[5] 99.00-100.00 sec 94.5 MBytes 792 Mbits/sec 0 12.7 MBytes
[5] 100.00-101.01 sec 99.8 MBytes 827 Mbits/sec 0 12.7 MBytes
[5] 101.01-102.01 sec 104 MBytes 875 Mbits/sec 0 12.7 MBytes
[5] 102.01-103.01 sec 110 MBytes 921 Mbits/sec 0 12.7 MBytes
[5] 103.01-104.01 sec 108 MBytes 902 Mbits/sec 0 12.7 MBytes
[5] 104.01-105.00 sec 108 MBytes 912 Mbits/sec 0 12.7 MBytes
[5] 105.00-106.00 sec 95.0 MBytes 794 Mbits/sec 0 12.7 MBytes
[5] 106.00-107.01 sec 88.5 MBytes 740 Mbits/sec 0 12.7 MBytes
[5] 107.01-108.01 sec 124 MBytes 1.03 Gbits/sec 0 12.7 MBytes
[5] 108.01-109.00 sec 115 MBytes 977 Mbits/sec 0 12.7 MBytes
[5] 109.00-110.00 sec 106 MBytes 890 Mbits/sec 0 12.7 MBytes
[5] 110.00-111.01 sec 107 MBytes 891 Mbits/sec 0 12.7 MBytes
[5] 111.01-112.00 sec 104 MBytes 876 Mbits/sec 0 12.7 MBytes
[5] 112.00-113.01 sec 127 MBytes 1.06 Gbits/sec 0 12.7 MBytes
[5] 113.01-114.01 sec 136 MBytes 1.14 Gbits/sec 0 12.7 MBytes
[5] 114.01-115.00 sec 108 MBytes 914 Mbits/sec 0 12.7 MBytes
[5] 115.00-116.01 sec 101 MBytes 838 Mbits/sec 0 12.7 MBytes
[5] 116.01-117.00 sec 101 MBytes 862 Mbits/sec 0 12.7 MBytes
[5] 117.00-118.00 sec 108 MBytes 904 Mbits/sec 0 12.7 MBytes
[5] 118.00-119.00 sec 112 MBytes 940 Mbits/sec 0 12.7 MBytes
[5] 119.00-120.01 sec 120 MBytes 1.01 Gbits/sec 0 12.7 MBytes
- -
[ID] Interval Transfer Bitrate Retr
[5] 0.00-120.01 sec 10.9 GBytes 783 Mbits/sec 1 sender
[5] 0.00-120.10 sec 10.9 GBytes 783 Mbits/sec receiver
iperf Done.
--- iPerf3 Test 3/3: h2 -> h8 ---
Connecting to host 10.0.0.9, port 5201
[5] local 10.1.1.3 port 60808 connected to 10.0.0.9 port 5201
[ID] Interval Transfer Bitrate Retr Cwnd
[5] 0.00-1.00 sec 1.38 MBytes 11.5 Mbits/sec 0 240 KBytes

[5] 1.00-2.00 sec 3.00 MBytes 25.2 Mbits/sec 0 355 KBytes
[5] 2.00-3.00 sec 4.12 MBytes 34.6 Mbits/sec 0 525 KBytes
[5] 3.00-4.02 sec 6.00 MBytes 49.7 Mbits/sec 0 775 KBytes
[5] 4.02-5.01 sec 8.88 MBytes 74.5 Mbits/sec 0 1.07 MBytes
[5] 5.01-6.00 sec 11.0 MBytes 93.6 Mbits/sec 0 1.57 MBytes
[5] 6.00-7.01 sec 16.1 MBytes 134 Mbits/sec 0 2.21 MBytes
[5] 7.01-8.00 sec 23.4 MBytes 197 Mbits/sec 0 3.27 MBytes
[5] 8.00-9.00 sec 29.9 MBytes 251 Mbits/sec 0 4.61 MBytes
[5] 9.00-10.02 sec 44.0 MBytes 363 Mbits/sec 0 7.14 MBytes
[5] 10.02-11.01 sec 62.1 MBytes 523 Mbits/sec 0 9.91 MBytes
[5] 11.01-12.01 sec 55.2 MBytes 466 Mbits/sec 0 12.7 MBytes
[5] 12.01-13.02 sec 55.4 MBytes 461 Mbits/sec 0 12.7 MBytes
[5] 13.02-14.01 sec 48.2 MBytes 408 Mbits/sec 0 12.7 MBytes
[5] 14.01-15.02 sec 62.2 MBytes 516 Mbits/sec 0 12.7 MBytes
[5] 15.02-16.02 sec 55.2 MBytes 464 Mbits/sec 0 12.7 MBytes
[5] 16.02-17.00 sec 55.2 MBytes 471 Mbits/sec 0 12.7 MBytes
[5] 17.00-18.01 sec 62.2 MBytes 516 Mbits/sec 0 12.7 MBytes
[5] 18.01-19.02 sec 62.2 MBytes 520 Mbits/sec 0 12.7 MBytes
[5] 19.02-20.00 sec 62.1 MBytes 529 Mbits/sec 0 12.7 MBytes
[5] 20.00-21.02 sec 75.0 MBytes 619 Mbits/sec 0 12.7 MBytes
[5] 21.02-22.00 sec 79.1 MBytes 674 Mbits/sec 0 12.7 MBytes
[5] 22.00-23.01 sec 79.2 MBytes 663 Mbits/sec 0 12.7 MBytes
[5] 23.01-24.00 sec 88.2 MBytes 745 Mbits/sec 0 12.7 MBytes
[5] 24.00-25.02 sec 89.4 MBytes 737 Mbits/sec 0 12.7 MBytes
[5] 25.02-26.01 sec 88.5 MBytes 748 Mbits/sec 0 12.7 MBytes
[5] 26.01-27.00 sec 79.8 MBytes 676 Mbits/sec 0 12.7 MBytes
[5] 27.00-28.00 sec 81.0 MBytes 680 Mbits/sec 0 12.7 MBytes
[5] 28.00-29.00 sec 86.6 MBytes 724 Mbits/sec 0 12.7 MBytes
[5] 29.00-30.01 sec 70.4 MBytes 585 Mbits/sec 0 12.7 MBytes
[5] 30.01-31.00 sec 79.4 MBytes 673 Mbits/sec 0 12.7 MBytes
[5] 31.00-32.02 sec 88.0 MBytes 725 Mbits/sec 0 12.7 MBytes
[5] 32.02-33.01 sec 88.4 MBytes 749 Mbits/sec 0 12.7 MBytes
[5] 33.01-34.00 sec 79.5 MBytes 670 Mbits/sec 0 12.7 MBytes
[5] 34.00-35.00 sec 70.5 MBytes 593 Mbits/sec 0 12.7 MBytes
[5] 35.00-36.01 sec 79.4 MBytes 658 Mbits/sec 0 12.7 MBytes
[5] 36.01-37.01 sec 79.2 MBytes 665 Mbits/sec 0 12.7 MBytes
[5] 37.01-38.00 sec 88.4 MBytes 748 Mbits/sec 0 12.7 MBytes
[5] 38.00-39.00 sec 75.8 MBytes 636 Mbits/sec 0 12.7 MBytes
[5] 39.00-40.00 sec 89.1 MBytes 749 Mbits/sec 0 12.7 MBytes
[5] 40.00-41.01 sec 79.6 MBytes 664 Mbits/sec 0 12.7 MBytes
[5] 41.01-42.00 sec 79.5 MBytes 671 Mbits/sec 0 12.7 MBytes

[5] 42.00-43.01 sec 79.9 MBytes 664 Mbits/sec 0 12.7 MBytes
[5] 43.01-44.01 sec 88.0 MBytes 741 Mbits/sec 0 12.7 MBytes
[5] 44.01-45.00 sec 84.9 MBytes 715 Mbits/sec 0 12.7 MBytes
[5] 45.00-46.01 sec 74.9 MBytes 621 Mbits/sec 0 12.7 MBytes
[5] 46.01-47.00 sec 87.8 MBytes 744 Mbits/sec 0 12.7 MBytes
[5] 47.00-48.00 sec 79.9 MBytes 670 Mbits/sec 0 12.7 MBytes
[5] 48.00-49.01 sec 70.6 MBytes 584 Mbits/sec 0 12.7 MBytes
[5] 49.01-50.01 sec 91.2 MBytes 766 Mbits/sec 0 12.7 MBytes
[5] 50.01-51.01 sec 83.4 MBytes 701 Mbits/sec 0 12.7 MBytes
[5] 51.01-52.00 sec 93.0 MBytes 788 Mbits/sec 0 12.7 MBytes
[5] 52.00-53.01 sec 93.1 MBytes 776 Mbits/sec 0 12.7 MBytes
[5] 53.01-54.01 sec 81.8 MBytes 684 Mbits/sec 0 12.7 MBytes
[5] 54.01-55.00 sec 90.6 MBytes 768 Mbits/sec 0 12.7 MBytes
[5] 55.00-56.01 sec 84.0 MBytes 701 Mbits/sec 0 12.7 MBytes
[5] 56.01-57.01 sec 101 MBytes 846 Mbits/sec 0 12.7 MBytes
[5] 57.01-58.00 sec 90.6 MBytes 762 Mbits/sec 0 12.7 MBytes
[5] 58.00-59.00 sec 90.5 MBytes 758 Mbits/sec 0 12.7 MBytes
[5] 59.00-60.00 sec 90.6 MBytes 763 Mbits/sec 0 12.7 MBytes
[5] 60.00-61.00 sec 97.0 MBytes 814 Mbits/sec 0 12.7 MBytes
[5] 61.00-62.01 sec 105 MBytes 869 Mbits/sec 0 12.7 MBytes
[5] 62.01-63.01 sec 108 MBytes 906 Mbits/sec 0 12.7 MBytes
[5] 63.01-64.01 sec 100 MBytes 838 Mbits/sec 0 12.7 MBytes
[5] 64.01-65.01 sec 115 MBytes 972 Mbits/sec 0 12.7 MBytes
[5] 65.01-66.01 sec 103 MBytes 858 Mbits/sec 0 12.7 MBytes
[5] 66.01-67.01 sec 110 MBytes 929 Mbits/sec 0 12.7 MBytes
[5] 67.01-68.02 sec 103 MBytes 861 Mbits/sec 0 12.7 MBytes
[5] 68.02-69.01 sec 110 MBytes 922 Mbits/sec 0 12.7 MBytes
[5] 69.01-70.01 sec 106 MBytes 896 Mbits/sec 0 12.7 MBytes
[5] 70.01-71.01 sec 97.9 MBytes 822 Mbits/sec 0 12.7 MBytes
[5] 71.01-72.01 sec 110 MBytes 912 Mbits/sec 0 12.7 MBytes
[5] 72.01-73.00 sec 110 MBytes 929 Mbits/sec 0 12.7 MBytes
[5] 73.00-74.01 sec 110 MBytes 914 Mbits/sec 0 12.7 MBytes
[5] 74.01-75.00 sec 104 MBytes 880 Mbits/sec 0 12.7 MBytes
[5] 75.00-76.00 sec 110 MBytes 921 Mbits/sec 0 12.7 MBytes
[5] 76.00-77.00 sec 104 MBytes 873 Mbits/sec 0 12.7 MBytes
[5] 77.00-78.01 sec 120 MBytes 1.01 Gbits/sec 0 12.7 MBytes
[5] 78.01-79.00 sec 98.8 MBytes 830 Mbits/sec 0 12.7 MBytes
[5] 79.00-80.01 sec 107 MBytes 895 Mbits/sec 0 12.7 MBytes
[5] 80.01-81.01 sec 107 MBytes 896 Mbits/sec 0 12.7 MBytes
[5] 81.01-82.00 sec 108 MBytes 914 Mbits/sec 0 12.7 MBytes
[5] 82.00-83.01 sec 116 MBytes 963 Mbits/sec 0 12.7 MBytes

```
[  5]  83.01-84.00  sec  97.5 MBytes   824 Mbits/sec    0   12.7 MBytes
[  5]  84.00-85.00  sec   120 MBytes  1.01 Gbits/sec    0   12.7 MBytes
[  5]  85.00-86.01  sec   101 MBytes   840 Mbits/sec    0   12.7 MBytes
[  5]  86.01-87.00  sec   113 MBytes   954 Mbits/sec    0   12.7 MBytes
[  5]  87.00-88.01  sec   108 MBytes   895 Mbits/sec    0   12.7 MBytes
[  5]  88.01-89.00  sec   104 MBytes   884 Mbits/sec    0   12.7 MBytes
[  5]  89.00-90.00  sec   110 MBytes   918 Mbits/sec    0   12.7 MBytes
[  5]  90.00-91.01  sec   113 MBytes   944 Mbits/sec    0   12.7 MBytes
[  5]  91.01-92.01  sec   108 MBytes   905 Mbits/sec    0   12.7 MBytes
[  5]  92.01-93.01  sec   102 MBytes   857 Mbits/sec    0   12.7 MBytes
[  5]  93.01-94.01  sec   113 MBytes   954 Mbits/sec    0   12.7 MBytes
[  5]  94.01-95.01  sec   115 MBytes   970 Mbits/sec    0   12.7 MBytes
[  5]  95.01-96.00  sec  97.4 MBytes   820 Mbits/sec    0   12.7 MBytes
[  5]  96.00-97.00  sec   110 MBytes   918 Mbits/sec    0   12.7 MBytes
[  5]  97.00-98.01  sec   108 MBytes   901 Mbits/sec    0   12.7 MBytes
[  5]  98.01-99.01  sec   108 MBytes   903 Mbits/sec    0   12.7 MBytes
[  5]  99.01-100.01 sec   140 MBytes  1.18 Gbits/sec    0   12.7 MBytes
[  5] 100.01-101.01 sec  99.1 MBytes   829 Mbits/sec    0   12.7 MBytes
[  5] 101.01-102.01 sec  99.5 MBytes   838 Mbits/sec    0   12.7 MBytes
[  5] 102.01-103.00 sec  99.6 MBytes   839 Mbits/sec    0   12.7 MBytes
[  5] 103.00-104.02 sec  99.6 MBytes   825 Mbits/sec    0   12.7 MBytes
[  5] 104.02-105.00 sec  98.9 MBytes   839 Mbits/sec    0   12.7 MBytes
[  5] 105.00-106.01 sec   108 MBytes   905 Mbits/sec    0   12.7 MBytes
[  5] 106.01-107.00 sec   112 MBytes   941 Mbits/sec    0   12.7 MBytes
[  5] 107.00-108.01 sec  99.8 MBytes   833 Mbits/sec    0   12.7 MBytes
[  5] 108.01-109.01 sec   118 MBytes   982 Mbits/sec    0   12.7 MBytes
[  5] 109.01-110.01 sec   102 MBytes   860 Mbits/sec    0   12.7 MBytes
[  5] 110.01-111.01 sec   114 MBytes   947 Mbits/sec    0   12.7 MBytes
[  5] 111.01-112.01 sec   116 MBytes   985 Mbits/sec    0   12.7 MBytes
[  5] 112.01-113.00 sec   105 MBytes   888 Mbits/sec    0   12.7 MBytes
[  5] 113.00-114.01 sec   110 MBytes   914 Mbits/sec    0   12.7 MBytes
[  5] 114.01-115.01 sec   113 MBytes   948 Mbits/sec    0   12.7 MBytes
[  5] 115.01-116.00 sec   139 MBytes  1.18 Gbits/sec    0   12.7 MBytes
[  5] 116.00-117.01 sec   112 MBytes   935 Mbits/sec    0   12.7 MBytes
[  5] 117.01-118.00 sec   122 MBytes  1.03 Gbits/sec    0   12.7 MBytes
[  5] 118.00-119.00 sec   107 MBytes   898 Mbits/sec    0   12.7 MBytes
[  5] 119.00-120.01 sec   107 MBytes   884 Mbits/sec    0   12.7 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-120.01 sec  10.3 GBytes   740 Mbits/sec    0             sender
[  5]   0.00-120.14 sec  10.3 GBytes   740 Mbits/sec                  receiver
```

iperf Done.

Routing tables can be seen using route -n

```
mininet> h9 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 h9-eth2
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 br0
172.16.10.0     0.0.0.0         255.255.255.0   U     0      0        0 h9-eth2
```

NAT Rules Summary

Rule Type	Rule Description	iptables Command
MASQUERADE	Outbound NAT for internal hosts	iptables -t nat -A POSTROUTING -o h9-eth0 -j MASQUERADE
DNAT	ICMP to H1	iptables -t nat -A PREROUTING -d 172.16.10.10 -p icmp --icmp-type echo-request -j DNAT --to 10.1.1.2
DNAT	ICMP to H2	iptables -t nat -A PREROUTING -d 172.16.10.10 -p icmp --icmp-type echo-request -j DNAT --to 10.1.1.3

Q3: Implementation of Distance Vector Routing (DV Routing)

Objective

The objective was to simulate the behavior of routers within a distributed network using an asynchronous distance vector algorithm. The system was implemented in C, where each node (router) is designed to update its routing table based on information received from its directly connected neighbors.

Problem Statement

The task involved implementing the core logic of a distance vector routing algorithm in a simulated environment consisting of four nodes (Node 0 to Node 3). Each node must:

- Maintain a **distance table** that stores the minimum known costs to reach every other node via each of its neighbors.
- Exchange routing information with **directly connected neighbors only**.
- Update its routing table whenever it receives a new distance vector from a neighbor.
- **Send updates to neighbors** only when the minimum known cost to any destination changes.
- Operate in an **asynchronous** manner—updates may be received in any order and at any time.

The environment simulates message passing through a function tolayer2() and allows tracing of internal events for debugging.

Distance Vector Routing

Distance vector routing is an algorithm in which routers share information about the distances (or costs) to each destination within a network. The key ideas are:

- **Local Knowledge:** Each node knows only its direct link costs.
- **Update Exchanges:** Nodes periodically send their distance vectors to neighboring nodes.
- **Bellman-Ford Principle:** Upon receiving an update from a neighbor, a router uses the Bellman-Ford equation to recalculate its best-known cost (or distance) to each destination. This is accomplished by examining if the path through the neighbor offers a lower cost than the current known cost.

  The general update formula is given by:

  $D_x(y) = \min\{c(x,v) + D_v(y)\}$, where $v \in$ neighbors(x)

  Here:

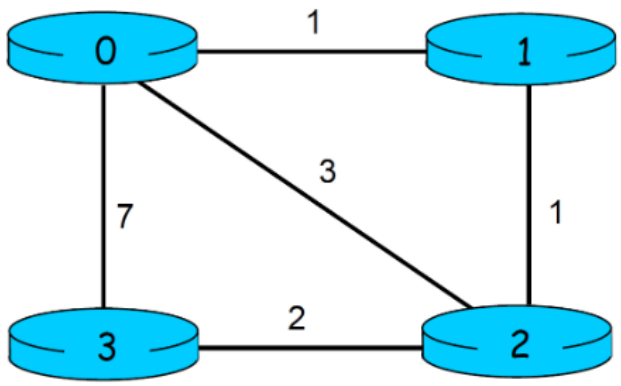
  - $D_x(y)$: cost from node x to destination y
  - c(x,v): cost from node x to neighbor v
  - $D_v(y)$: cost from neighbor v to destination y

In essence, node x examines the cost to reach each neighbor v, adds the cost reported by that neighbor v for reaching destination y, and then selects the smallest sum. If this sum is less than the current estimate at x, then the estimate is updated, and node x will inform its neighbors of the new route.

Network Topology and Node Interactions

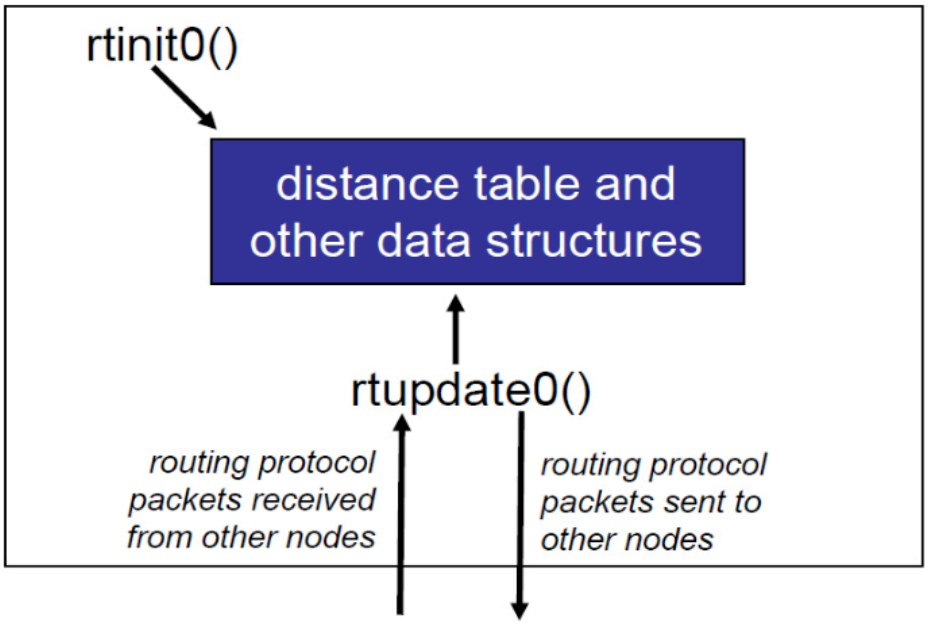
The Topology

The simulated topology includes four nodes (numbered 0–3) with bi-directional links.



Each node follows a similar structure, and nodes communicate only with their immediate (directly connected) neighbors using a shared function called `tolayer2()`. The network emulator also handles the delivery (in order, without loss) of packets between nodes at variable delays.

Node Communication



Each node implements two core procedures:

- **`rtinit*()`:** Called at the start of the simulation to initialize its distance table with direct link costs and then send its initial distance vector to neighbors.
- **`rtupdate*()`:** Called whenever a routing packet is received from a neighbor. This function updates the node's distance table and, if any changes occur in the minimum cost to a destination, sends updated cost information to its neighbors.

Optionally, nodes (like Node 0) also implement a **`linkhandler*()`** function that handles dynamic link cost changes during the simulation.

Software Design and Architecture

File Structure:

- distance_vector.c: Contains the simulator logic and the driver code.
- node0.c, node1.c, node2.c, node3.c: Node-specific routing logic files.

Data Structures:

```
extern struct rtpkt {
  int sourceid;       /* id of sending router sending this pkt */
  int destid;         /* id of router to which pkt being sent
                         (must be an immediate neighbor) */
  int mincost[4];    /* min cost to node 0 ... 3 */
  };
```

```
struct distance_table
{
  int costs[4][4];
} dt0;
```

Each cell `dt0.costs[i][j]` represents Node 0's computed cost to destination `i` when routed through neighbor `j`. Initially, the cost on the direct link is populated, and all other cells are set to a large number (`INFINITY`, defined as 999).

distance_vector.c:

We were given a support file named **`distance_vector.c`**, which contained key routines used by the individual node implementations (`node0.c`, `node1.c`, `node2.c`, and `node3.c`). This file facilitated the simulation of the network and included two important functions:

1. **`tolayer2(struct rtpkt pkt2send)`**
   This function simulates sending a packet from one node to another. The packet is structured using the `rtpkt` data type, which holds the source node, destination node, and the node's current distance vector.

2. **`printdt0()`**
   This function provides a formatted view of the distance table for node 0. Similar functions (`printdt1()`, `printdt2()`, `printdt3()`) exist for the other nodes, helping to visualize each node's view of the network

The file also contained a random number generation function called `jimsrand()` that returns a floating-point number between 0 and 1. This is used to standardize randomness across the simulation.

```
float jimsrand() {
    double mmm = __INT16_MAX__;  // Platform-dependent maximum int value
    float x = rand() / mmm;
```

```
    return x;
}
```

Originally, the variable `mmm` was assigned an arbitrary value, but we modified it to use `__INT16_MAX__` to ensure it worked correctly on our machine. This change made the function portable and prevented unexpected behavior due to incorrect scaling of the random values.

When the program runs, it prompts the user to enter a **trace value**, which determines the level of debug output printed during the simulation:

- A **trace value of 0** disables all output.
- **Values 1 or 2** display useful debug information, such as packet transmissions, routing updates, and distance table changes.
- **Values above 2** enable verbose internal messages used primarily for emulator debugging.

Helper Functions:

1. Minimum Calculation Functions

   A pair of helper functions compute the minimum value:

   ```
   int min_0 ( int a, int b ) {
       return (a < b) ? a : b;
   }
   int min_0_array ( int a[] ) {
       return min_0(min_0(min_0(a[0], a[1]), a[2]), a[3]);
   }
   ```

   These functions allow the program to determine the lowest cost from a set of values—for example, finding the best cost to a given destination across all available routes.

2. Recalculating and Tracking Costs

   The function `recalculate_min_costs_0()` updates an array `min_cost_track_0` that holds the minimum cost from Node 0 to each destination:

   ```
   void recalculate_min_costs_0(){
       for(int i=0;i<4;i++){
           min_cost_track_0[i] = min_0_array(dt0.costs[i]);
       }
   }
   ```

This function is crucial after any change in the distance table; it allows the node to decide whether updated information should be shared with its neighbors.

3. Packet Preparation and Sending

   The function `send_updated_pkt0()` prepares a routing packet for every node (except itself) and calls `tolayer2()`:

```
void send_updated_pkt0() {
    for (int i = 0; i < 4; i++) {
        pkt0[i].sourceid = 0;
        pkt0[i].destid = i;
        memcpy(pkt0[i].mincost, min_cost_track_0,
sizeof(min_cost_track_0));
    }
    for (int i = 0; i < 4; i++) {
        if (i != 0) {  // Avoid sending to self
            tolayer2(pkt0[i]);
            printf("Time %.3f: Node %d dispatches a packet to Node
%d carrying mincosts: [%d  %d  %d  %d]\n",
                   clocktime, pkt0[i].sourceid, pkt0[i].destid,
                   pkt0[i].mincost[0], pkt0[i].mincost[1],
                   pkt0[i].mincost[2], pkt0[i].mincost[3]);
        }
    }
}
```

   This function is called whenever a cost change is detected, ensuring that neighbors are kept up-to-date with Node 0's current best path estimates.

4. Evaluating Changes

   Before broadcasting new routing packets, the function `evaluate_and_send_pkt0()` compares the new minimum costs with the previous state:

```
void evaluate_and_send_pkt0() {
    int previous_min_costs[4];
    memcpy(previous_min_costs, min_cost_track_0,
sizeof(min_cost_track_0));

    int has_changed = 0;
    recalculate_min_costs_0();
    for (int i = 0; i < 4; i++) {
```

```
        if (previous_min_costs[i] != min_cost_track_0[i]) {
            has_changed = 1;
            break;
        }
    }
    if (has_changed) {
        send_updated_pkt0();
    } else {
        printf("\nNo change in minimum costs. Skipping packet
transmission.\n");
    }
}
```

This keeps the algorithm efficient by sending updates only when actual improvements occur.

Core Functions:

Each of the node files (`node0.c`, `node1.c`, `node2.c`, `node3.c`) defines three primary functions.

- ### rtinitX():

Purpose: Sets up the node's distance table and sends initial routing packets to neighbors.

Details:
- Initializes all costs in `distanceTable.costs` to `INFINITY` (999).
- Updates the table with known direct link costs from `connection_costs`.
- Sends out initial distance vectors using `tolayer2()` to all directly connected neighbors.

Example from node0.c:

```
int connection_costs0[4] = { 0,  1,  3, 7 };
```

This indicates Node 0 is directly connected to:

- Node 1 (cost 1)
- Node 2 (cost 3)
- Node 3 (cost 7)

- rtupdateX(struct rtpkt *rcvdpkt) (Packet Processing):

  Purpose: Called when a packet is received from a neighbor. It updates the distance table using the Bellman-Ford formula.

  Details:

  ` - **Receive and Log the Packet:** The function logs which neighbor sent the update along with the offered costs.

  - **Update the Table:** For each destination, Node 0 uses the cost to reach the sender (its neighbor) plus the neighbor's cost to the destination.

  - **Recalculation:** After updating, the node prints the updated table and calls `evaluate_and_send_pkt0()` to determine if neighbors must be informed of any cost changes.

  Example from node1.c:

```
for (int i = 0; i < 4; i++) {
        if (previous_min_costs[i] != min_cost_track_1[i]) {
            has_changed = 1;
        }
    }
```

- printdtX():
  - **Purpose**: Outputs the node's current distance table in a readable format.
  - Used to trace and debug routing decisions and updates.

- linkhandlerX()

  Purpose:
  - **Recalculate Costs:** When the cost to a directly connected neighbor changes, the function recalculates the costs to all destinations that use this neighbor.
  - **Propagation:** It then triggers an update to determine if updated routing packets should be sent.

Compilation and Execution:

The following command was used to compile the five files (node0.c, node1.c, node2.c, node3.c, distance_vector.c) :

```
gcc distance_vector.c node0.c node1.c node2.c node3.c -o distance_vector_final
```

An executable "distance_vector_final" was formed. Run the executable using the command:

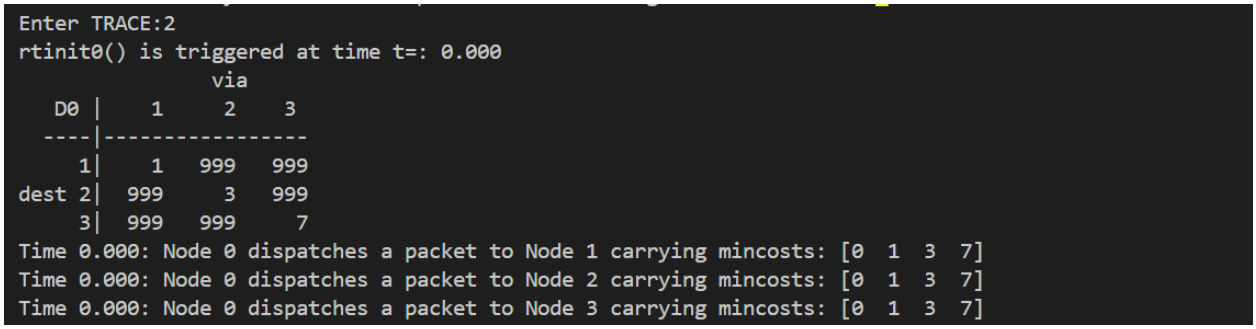
```
./distance_vector_final
```

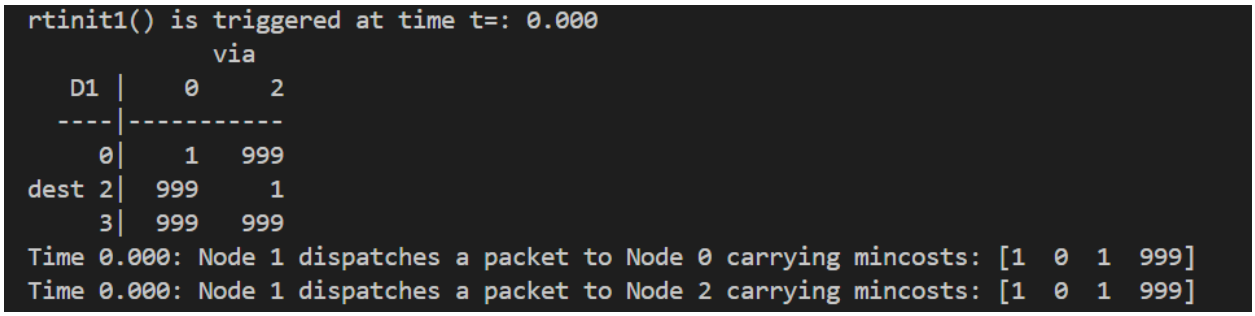
After selecting a trace value of 2, a detailed output was observed.

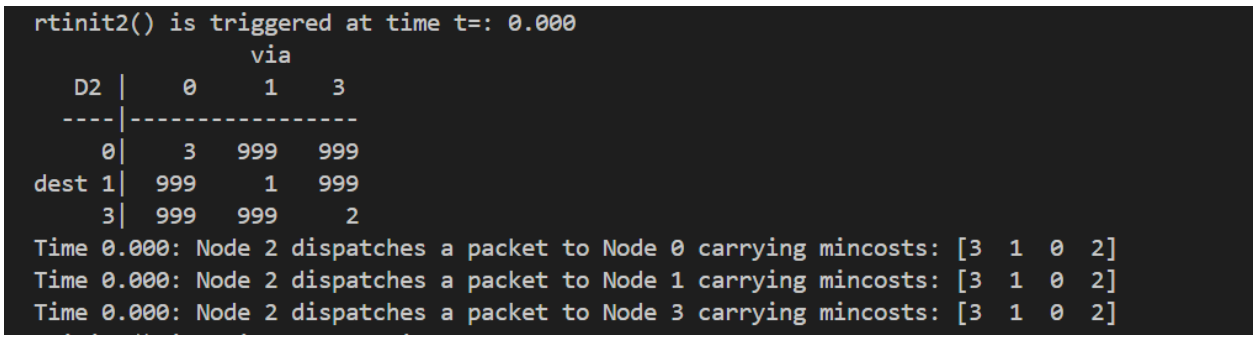
Program Output and Screenshots:

The program output was generated using TRACE = 2 for detailed logs. Sample logs are as follows:

1. Initial Distance table for each node:







```
rtinit3() is triggered at time t=: 0.000
            via
   D3 |    0     2
  ----|-----------
     0|    7   999
dest 1|  999   999
     2|  999     2
Time 0.000: Node 3 dispatches a packet to Node 0 carrying mincosts: [7  999  2  0]
Time 0.000: Node 3 dispatches a packet to Node 2 carrying mincosts: [7  999  2  0]
```

Each node starts by knowing the cost to its immediate neighbors and itself. All other distances are initialized to infinity. The rtinitX() function (where X = 0 to 3) is called for each node, and the initial distance table is printed.

2. Distance Table Updates Upon Receiving Packets:

```
MAIN: rcv event, t=0.094, at 1 src: 0, dest: 1, contents:   0   1   3   7
rtupdate1() executed at time t=: 0.094 as node 0 sent a packet containing (0  1  3  7)
            via
   D1 |    0     2
  ----|-----------
     0|    1   999
dest 2|    4     1
     3|    8   999
Time 0.094: Node 1 dispatches a packet to Node 0 carrying mincosts: [1  0  1  8]
Time 0.094: Node 1 dispatches a packet to Node 2 carrying mincosts: [1  0  1  8]
```

```
MAIN: rcv event, t=0.427, at 1 src: 2, dest: 1, contents:   3   1   0   2
rtupdate1() executed at time t=: 0.427 as node 2 sent a packet containing (3  1  0  2)
            via
   D1 |    0     2
  ----|-----------
     0|    1     4
dest 2|    4     1
     3|    8     3
Time 0.427: Node 1 dispatches a packet to Node 0 carrying mincosts: [1  0  1  3]
Time 0.427: Node 1 dispatches a packet to Node 2 carrying mincosts: [1  0  1  3]
```

```
MAIN: rcv event, t=0.998, at 0 src: 1, dest: 0, contents:   1   0   1 999
rtupdate0()  executed at time t = 0.998 as node 1 sent a packet containing (1  0  1  999)
               via
   D0 |    1     2    3
  ----|-----------------
     1|    1   999   999
dest 2|    2     3   999
     3|  999   999     7
Time 0.998: Node 0 dispatches a packet to Node 1 carrying mincosts: [0  1  2  7]
Time 0.998: Node 0 dispatches a packet to Node 2 carrying mincosts: [0  1  2  7]
Time 0.998: Node 0 dispatches a packet to Node 3 carrying mincosts: [0  1  2  7]
```

```
MAIN: rcv event, t=1.244, at 3 src: 0, dest: 3, contents:   0   1   3   7
rtupdate0()  executed at time t = 1.244 as node 0 sent a packet containing (0  1  3  7)
             via
   D3 |    0     2
  ----|-----------
     0|    7   999
dest 1|    8   999
     2|   10     2
Time 1.244: Node 3 dispatches a packet to Node 0 carrying mincosts: [7  8  2  0]
Time 1.244: Node 3 dispatches a packet to Node 2 carrying mincosts: [7  8  2  0]
```

When a node receives a distance vector from a neighbor, it evaluates whether a shorter path to a destination is possible through that neighbor. If a better route is found, it updates its distance table accordingly. These updates are printed and provide insight into how routing information is gradually propagated across the network.

3. No Update When Minimum Cost Remains the Same

```
MAIN: rcv event, t=2.489, at 0 src: 1, dest: 0, contents:   1   0   1   8
rtupdate0()  executed at time t = 2.489 as node 1 sent a packet containing (1  0  1  8)
             via
   D0 |    1     2    3
  ----|----------------
     1|    1     4   999
dest 2|    2     3     9
     3|    9     5     7

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=2.823, at 1 src: 0, dest: 1, contents:   0   1   2   7
rtupdate1() executed at time t=: 2.823 as node 0 sent a packet containing (0  1  2  7)
             via
   D1 |    0     2
  ----|-----------
     0|    1     4
dest 2|    3     1
     3|    8     3

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=3.780, at 3 src: 0, dest: 3, contents:   0   1   2   7
rtupdate0()  executed at time t = 3.780 as node 0 sent a packet containing (0  1  2  7)
             via
   D3 |    0     2
  ----|-----------
     0|    7     5
dest 1|    8     3
     2|    9     2

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=3.798, at 2 src: 3, dest: 2, contents:   7 999   2   0
rtupdate0()  executed at time t = 3.798 as node 3 sent a packet containing (7  999  2  0)
               via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2    9
dest 1|    4     1  999
     3|   10   999    2

No change in minimum costs. Skipping packet transmission.
```

In some cases, the received packet does not offer a better route. When this happens, the distance table remains unchanged and the node does not send out new packets. This prevents unnecessary communication.

4. Network Convergence – No Further Updates:

```
MAIN: rcv event, t=20004.307, at 0 src: 2, dest: 0, contents:   2   1   0   2
rtupdate0()  executed at time t = 20004.307 as node 2 sent a packet containing (2  1  0  2)
               via
   D0 |    1     2    3
  ----|-----------------
     1|    1     4   10
dest 2|    2     3    9
     3|    4     5    7

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=20002.922, at 1 src: 2, dest: 1, contents:   2   1   0   2
rtupdate1() executed at time t=: 20002.922 as node 2 sent a packet containing (2  1  0  2)
              via
   D1 |    0     2
  ----|-----------
     0|    1     3
dest 2|    3     1
     3|    5     3

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=20006.547, at 2 src: 3, dest: 2, contents:   4   3   2   0
rtupdate0()  executed at time t = 20006.547 as node 3 sent a packet containing (4  3  2  0)
               via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2    6
dest 1|    4     1    5
     3|    7     4    2

No change in minimum costs. Skipping packet transmission.
```

```
MAIN: rcv event, t=20004.629, at 3 src: 2, dest: 3, contents:   2   1   0   2
rtupdate0()  executed at time t = 20004.629 as node 2 sent a packet containing (2  1  0  2)
             via
   D3 |    0     2
  ----|-----------
     0|    7     4
dest 1|    8     3
     2|    9     2
Time 20004.629: Node 3 dispatches a packet to Node 0 carrying mincosts: [4  3  2  0]
Time 20004.629: Node 3 dispatches a packet to Node 2 carrying mincosts: [4  3  2  0]
```

After several rounds of message passing and updates, the network converges to a stable state. At this point, no node is able to find a shorter path via its neighbors, and the distance tables remain unchanged. This marks the successful completion of the simulation.

5. Final Converged Distance Tables:
   These are the final distance tables for each node established through the distributed and asynchronous distance vector protocol:

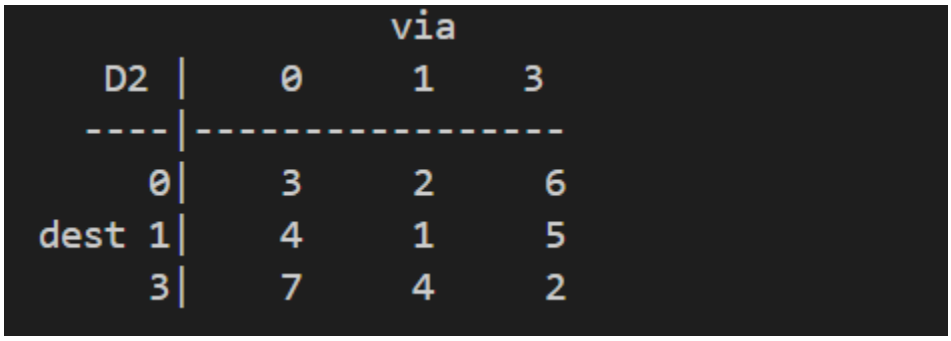
   For node 0:

```
                via
   D0 |    1     2     3
  ----|-----------------
     1|    1     4    10
dest 2|    2     3     9
     3|    4     5     7
```

   For node 1:

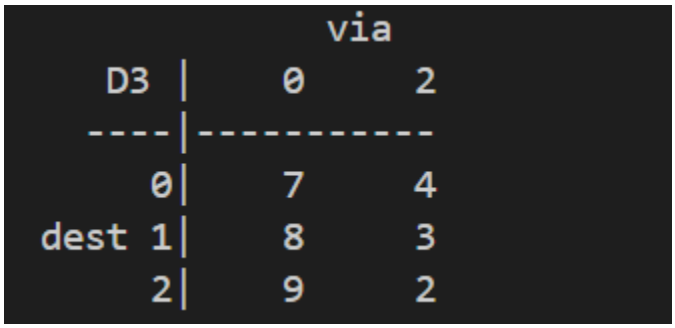
```
               via
   D1 |    0     2
  ----|-----------
     0|    1     3
dest 2|    3     1
     3|    5     3
```

For node 2:



	via		
D2	0	1	3
0	3	2	6
dest 1	4	1	5
3	7	4	2

For node 3:



	via	
D3	0	2
0	7	4
dest 1	8	3
2	9	2

Finally, the minimum cost from each node to all other nodes is:

- Node 0: 0, 1, 2, 4
- Node 1: 1, 0, 1, 3
- Node 2: 2, 1, 0, 2
- Node 3: 4, 3, 2, 0

Key Observations:

- **Modular Design:** Separating the logic into helper functions (`recalculate_min_costs_0()`, `evaluate_and_send_pkt0()`) not only makes debugging easier but also isolates individual responsibilities of the routing algorithm.

- **Dynamic Behavior:** The extra credit link handler (`linkhandler0()`) demonstrates an extension of the algorithm to handle dynamic link cost changes, reflecting real-world scenarios where network conditions may vary.

- **Efficient Updates:** By comparing previous minimum costs with the newly computed ones, the implementation sends update packets only when necessary—reducing

unnecessary network chatter.

- **Trace and Debug:** The detailed printouts (including the current simulation time and packet contents) are invaluable for validating the logic of asynchronous updates in a distributed environment.

Challenges Encountered:

- **Propagation Control**: Preventing infinite message propagation due to minor fluctuations.
- **Dynamic Updates**: Ensuring consistency during runtime link cost changes.
- **Asynchronous Messaging**: Accounting for unordered and delayed packet arrivals during updates.

References:

1. *Spanning Tree Protocol (STP) overview*. (2024, November 27). Cisco Meraki Documentation. https://documentation.meraki.com/MS/Port_and_VLAN_Configuration/Spanning_Tree_Protocol_(STP)_Overview
2. Contributors, M. P. (n.d.). *Mininet Walkthrough - Mininet*. https://mininet.org/walkthrough/
3. Jadhav, S. (2024, September 18). *Distance Vector Routing Algorithm*. Scaler Blog. https://www.scaler.in/distance-vector-routing-algorithm/