CS-331 COMPUTER NETWORKS
ASSIGNMENT 3

Github link: https://github.com/Siri-gangannagudem/CN_Assignment3
TASK 1:

In this task, we have first built a mininet topology as given in the figure.

The implemented network had the following configuration:

- 4 switches (s1, s2, s3, s4) connected in a loop
- 8 hosts (h1-h8) with 2 hosts connected to each switch
- Switch-to-switch links with 7ms delay
- Host-to-switch links with 5ms delay
- Had loops - s1 ↔ s2 ↔ s3 ↔ s4 ↔ s1, s1 ↔ s3 (additional link creating a loop)

We initialized networks with proper host and switch configuration, static MAC address assignment and ARP entries to facilitate communication.

a. Initially, when we run the ping commands between the hosts it gave 100% packet loss

```
--- PART A: Testing Ping Commands (Before STP) ---

Test 1: Ping h1 from h3
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2045ms


Test 2: Ping h7 from h5
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.

--- 10.0.0.8 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms


Test 3: Ping h2 from h8
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2057ms
```

The ping command can fail because the ping packets get stuck in a cycle, preventing them from reaching their intended destination. Essentially, the network devices (routers or switches) keep forwarding the ping packets back and forth within the loop, rather than delivering them to the target IP address. It is inherently unable to handle loops, as they lead to several critical issues. Broadcast storms can occur, where broadcast frames circulate indefinitely, consuming excessive bandwidth and degrading network performance. MAC address table instability arises as switches constantly update their forwarding tables with conflicting information. Finally, multiple frame delivery may result in duplicate packets reaching their destinations, leading to data inconsistency.

```
The network is now fixed using STP, and ping tests should be successful.

You can now run manual tests in the Mininet CLI
For example:
  h3 ping -c 3 10.0.0.2
  h5 ping -c 3 10.0.0.8
  h8 ping -c 3 10.0.0.3

Type 'exit' to quit Mininet
*** Starting CLI:
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=95.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=84.9 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=55.0 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 55.032/78.487/95.571/17.152 ms
mininet>
```

b. We used implementing Spanning Tree protocol on all switches to fix this problem without changing the topology. STP solves this by dynamically identifying and disabling redundant paths in the network, thus forming a loop-free logical topology even when the physical topology contains loops.
A 60-second waiting period allowed STP to converge properly, establishing a stable spanning tree.
STP is enabled:

```
for s in [s1, s2, s3, s4]:
    s.cmd('ovs-vsctl set bridge {} stp_enable=true'.format(s.name))
    if s.name == 's1':
        s.cmd('ovs-vsctl set bridge {} other_config:stp-priority=0x1000'.format(s.name))
```

```
--- PART B: Enabling STP to fix network loops ---
Waiting for STP to converge (60 seconds)...

--- STP Status on Each Switch ---

Switch s1 STP status:
other_config        : {datapath-id="0000000000000001", disable-in-band="true", dp-desc=s1, stp-priority="0x1000"}
rstp_enable         : false
rstp_status         : {}
status              : {stp_bridge_id="8000.52c44ef81346", stp_designated_root="8000.4ed52c82014e", stp_root_path_cost="2"}
stp_enable          : true

true

 port  VLAN  MAC                Age

2025-04-12T16:32:48Z|00001|vconn|WARN|unix:/var/run/openvswitch/s1.mgmt: version negotiation failed (we support version 0x01, peer supports version 0x04)
ovs-ofctl: s1: failed to connect to socket (Broken pipe)


Switch s2 STP status:
rstp_enable         : false
rstp_status         : {}
status              : {stp_bridge_id="8000.8e4b366eaa41", stp_designated_root="8000.4ed52c82014e", stp_root_path_cost="2"}
stp_enable          : true

true

 port  VLAN  MAC                Age

2025-04-12T16:32:48Z|00001|vconn|WARN|unix:/var/run/openvswitch/s2.mgmt: version negotiation failed (we support version 0x01, peer supports version 0x04)
ovs-ofctl: s2: failed to connect to socket (Broken pipe)


Switch s3 STP status:
```

```
Type 'exit' to quit Mininet
*** Starting CLI:
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=95.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=84.9 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=55.0 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 55.032/78.487/95.571/17.152 ms
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=76.1 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=40.8 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=38.7 ms

--- 10.0.0.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 38.712/51.853/76.054/17.133 ms
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=86.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=64.2 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=84.0 ms

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 64.219/78.370/86.852/10.072 ms
mininet>
```

1. For ping h1 from h3,

I ran the command 3 times:

```
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=67.6 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=93.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=76.3 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 67.602/79.161/93.600/10.807 ms
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=58.4 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=65.2 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=80.5 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 58.432/68.037/80.484/9.224 ms
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=53.0 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=105 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=141 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 52.983/99.871/141.294/36.259 ms
mininet>
```

Each command sends 3 packets, the average RTT in each test are 79.161ms, 68.037ms, 99.871ms

For h3 to h1, h3 → s2 → s1 → h1

h3–s2: 5 ms

s2–s1: 7 ms

s1–h1: 5 ms

Forward = 5+7+5 = 17ms, expected RTT = 17+17 = 34ms

But in our tests, they are higher due to STP convergence delay.

2. Ping h7 from h5

```
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=45.3 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=46.8 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=45.3 ms

--- 10.0.0.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 45.286/45.816/46.832/0.718 ms
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=38.6 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=225 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=258 ms

--- 10.0.0.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2013ms
rtt min/avg/max/mdev = 38.586/173.857/258.070/96.603 ms
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=37.5 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=37.4 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=44.7 ms

--- 10.0.0.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2014ms
rtt min/avg/max/mdev = 37.368/39.870/44.716/3.427 ms
mininet>
```

Each command sends 3 packets, the average RTT in each test are 45.286ms, 173.857ms, 39.870ms

For h5 to h7, h5 → s3 → s4 → h7

h5–s3: 5 ms , s3–s4: 7 ms , s4–h7: 5 ms

Forward = 5+7+5 = 17ms, expected RTT = 17+17 = 34ms

But in our tests, they are higher due to STP convergence delay.

3. Ping h2 from h8

```
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=59.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=65.8 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=73.7 ms

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 59.935/66.477/73.720/5.649 ms
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=54.2 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=92.0 ms

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 54.225/94.809/138.227/34.352 ms
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=53.2 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=56.0 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=52.5 ms

--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 52.526/53.910/55.987/1.495 ms
mininet>
```

Each command sends 3 packets, the average RTT in each test are 66.477ms, 94.809ms, 53.910ms

For h8 to h2, h8 → s4 → s1 → h2
h8–s4: 5 ms
s4–s1: 7 ms
s1–h2: 5 ms
Forward = 5+7+5 = 17ms, expected RTT = 17+17 = 34ms
But in our tests, they are higher due to STP convergence delay.

TASK 2:
Added a new host h9 to the topology with IP address 172.16.10.10 serving as NAT gateway for internal hosts h1 (10.1.1.2) and h2 (10.1.1.3)

a. Output:

```
a) Test communication to an external host from an internal host:
i) Ping to h5 from h1
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
From 10.1.1.2 icmp_seq=1 Destination Host Unreachable
From 10.1.1.2 icmp_seq=2 Destination Host Unreachable
From 10.1.1.2 icmp_seq=3 Destination Host Unreachable
From 10.1.1.2 icmp_seq=4 Destination Host Unreachable

--- 10.0.0.6 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3076ms
pipe 4

ii) Ping to h3 from h2
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
From 10.1.1.3 icmp_seq=1 Destination Host Unreachable
From 10.1.1.3 icmp_seq=2 Destination Host Unreachable
From 10.1.1.3 icmp_seq=3 Destination Host Unreachable
From 10.1.1.3 icmp_seq=4 Destination Host Unreachable

--- 10.0.0.4 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3058ms
pipe 4
```

Both the ping failed likely due to the missing firewall rules or NAT configuration. IP forwarding was likely not enabled on H9

b. Output:

```
b) Test communication to an internal host from an external host:
i) Ping to h1 from h8
PING 172.16.10.11 (172.16.10.11) 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
64 bytes from 172.16.10.11: icmp_seq=4 ttl=63 time=168 ms

--- 172.16.10.11 ping statistics ---
4 packets transmitted, 1 received, +3 errors, 75% packet loss, time 3055ms
rtt min/avg/max/mdev = 167.912/167.912/167.912/0.000 ms, pipe 4

ii) Ping to h2 from h6
PING 172.16.10.12 (172.16.10.12) 56(84) bytes of data.
64 bytes from 172.16.10.12: icmp_seq=1 ttl=63 time=138 ms
64 bytes from 172.16.10.12: icmp_seq=2 ttl=63 time=75.8 ms
64 bytes from 172.16.10.12: icmp_seq=3 ttl=63 time=55.4 ms
64 bytes from 172.16.10.12: icmp_seq=4 ttl=63 time=67.5 ms

--- 172.16.10.12 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 55.379/84.170/138.020/31.924 ms
```

The inconsistent results (75% vs 0% packet loss) suggest partial or inconsistent DNAT rule implementation. Successful pings prove that the public IPs (172.16.10.11 for h1 and 172.16.10.12 for h2) were properly assigned

c. Iperf output:

```
c) Iperf tests: 3 tests of 120s each.
i) Run iperf3 server in h1 and iperf3 client in h6
Connecting to host 172.16.10.11, port 5201
[  5] local 10.0.0.7 port 38346 connected to 172.16.10.11 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  1.86 MBytes  15.6 Mbits/sec    0    161 KBytes
[  5]   1.00-2.00   sec  3.36 MBytes  28.1 Mbits/sec    2    212 KBytes
[  5]   2.00-3.00   sec  3.42 MBytes  28.7 Mbits/sec    0    223 KBytes
[  5]   3.00-4.01   sec  4.16 MBytes  34.8 Mbits/sec    0    238 KBytes
[  5]   4.01-5.00   sec  3.36 MBytes  28.2 Mbits/sec    0    249 KBytes
[  5]   5.00-6.00   sec  4.04 MBytes  33.8 Mbits/sec    0    260 KBytes
[  5]   6.00-7.01   sec  3.73 MBytes  31.1 Mbits/sec    0    301 KBytes
[  5]   7.01-8.00   sec  4.16 MBytes  35.2 Mbits/sec    0    363 KBytes
[  5]   8.00-9.00   sec  4.91 MBytes  41.2 Mbits/sec    0    454 KBytes
[  5]   9.00-10.00  sec  7.33 MBytes  61.5 Mbits/sec    0    560 KBytes
[  5]  10.00-11.01  sec  9.92 MBytes  82.6 Mbits/sec    0    706 KBytes
[  5]  11.01-12.00  sec  11.2 MBytes  94.9 Mbits/sec    0    875 KBytes
[  5]  12.00-13.01  sec  16.2 MBytes   135 Mbits/sec    0   1.06 MBytes
[  5]  13.01-14.00  sec  17.5 MBytes   148 Mbits/sec    0   1.29 MBytes
[  5]  14.00-15.00  sec  21.2 MBytes   178 Mbits/sec    0   1.57 MBytes
[  5]  15.00-16.00  sec  27.5 MBytes   231 Mbits/sec    0   1.90 MBytes
[  5]  16.00-17.01  sec  27.5 MBytes   229 Mbits/sec    0   2.26 MBytes
```

```
[  5] 116.00-117.00 sec  80.0 MBytes   670 Mbits/sec    0   8.02 MBytes
[  5] 117.00-118.00 sec  88.8 MBytes   746 Mbits/sec    0   8.02 MBytes
[  5] 118.00-119.01 sec  85.0 MBytes   709 Mbits/sec    0   8.02 MBytes
[  5] 119.01-120.00 sec  97.5 MBytes   823 Mbits/sec    0   8.02 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-120.00 sec  10.5 GBytes   750 Mbits/sec    2             sender
[  5]   0.00-120.06 sec  10.5 GBytes   749 Mbits/sec                  receiver

iperf Done.

ii) Run iperf3 server in h8 and iperf3 client in h2
Connecting to host 10.0.0.9, port 5201
[  5] local 10.1.1.3 port 37182 connected to 10.0.0.9 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  27.8 MBytes   233 Mbits/sec    0   6.25 MBytes
[  5]   1.00-2.00   sec  82.5 MBytes   692 Mbits/sec    0   8.34 MBytes
[  5]   2.00-3.00   sec  98.8 MBytes   828 Mbits/sec    0   8.34 MBytes
[  5]   3.00-4.00   sec  87.5 MBytes   732 Mbits/sec    0   8.34 MBytes
[  5]   4.00-5.00   sec  86.2 MBytes   724 Mbits/sec    0   8.34 MBytes
[  5]   5.00-6.00   sec  92.5 MBytes   778 Mbits/sec    0   8.34 MBytes
[  5]   6.00-7.00   sec  91.2 MBytes   766 Mbits/sec    0   8.34 MBytes
[  5]   7.00-8.00   sec  97.5 MBytes   818 Mbits/sec    0   8.34 MBytes
[  5]   8.00-9.00   sec  90.0 MBytes   755 Mbits/sec    0   8.34 MBytes
```

```
[  5] 102.00-103.00 sec  83.8 MBytes   700 Mbits/sec    0   8.34 MBytes
[  5] 103.00-104.00 sec  55.0 MBytes   463 Mbits/sec    0   8.34 MBytes
[  5] 104.00-105.01 sec  60.0 MBytes   501 Mbits/sec    0   8.34 MBytes
[  5] 105.01-106.01 sec  52.5 MBytes   439 Mbits/sec    0   8.34 MBytes
[  5] 106.01-107.00 sec  58.8 MBytes   495 Mbits/sec    0   8.34 MBytes
[  5] 107.00-108.00 sec  53.8 MBytes   452 Mbits/sec    0   8.34 MBytes
[  5] 108.00-109.01 sec  62.5 MBytes   522 Mbits/sec    0   8.34 MBytes
[  5] 109.01-110.01 sec  57.5 MBytes   481 Mbits/sec    0   8.34 MBytes
[  5] 110.01-111.00 sec  70.0 MBytes   592 Mbits/sec    0   8.34 MBytes
[  5] 111.00-112.00 sec  67.5 MBytes   567 Mbits/sec    0   8.34 MBytes
[  5] 112.00-113.00 sec  81.2 MBytes   680 Mbits/sec    0   8.34 MBytes
[  5] 113.00-114.00 sec  62.5 MBytes   525 Mbits/sec    0   8.34 MBytes
[  5] 114.00-115.00 sec  72.5 MBytes   607 Mbits/sec    0   8.34 MBytes
[  5] 115.00-116.00 sec  71.2 MBytes   596 Mbits/sec    0   8.34 MBytes
[  5] 116.00-117.00 sec  73.8 MBytes   621 Mbits/sec    0   8.34 MBytes
[  5] 117.00-118.00 sec  72.5 MBytes   608 Mbits/sec    0   8.34 MBytes
[  5] 118.00-119.00 sec  80.0 MBytes   672 Mbits/sec    0   8.34 MBytes
[  5] 119.00-120.00 sec  83.8 MBytes   700 Mbits/sec    0   8.34 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-120.00 sec  9.11 GBytes   652 Mbits/sec   45             sender
[  5]   0.00-120.10 sec  9.11 GBytes   651 Mbits/sec                  receiver

iperf Done.
```

For both iperf tests:

- For test i (h6→h1): Average throughput was ~750 Mbits/sec with only 2 retransmissions
- For test ii (h2→h8): Average throughput was ~652 Mbits/sec with 45 retransmissions

Expected NAT rules:

- DNAT (Destination NAT) rules were partially working to forward incoming connections from external hosts to internal hosts (172.16.10.11→10.1.1.2 and 172.16.10.12→10.1.1.3)
- SNAT (Source NAT) rules for outgoing connections were missing or improperly configured since internal hosts couldn't reach external hosts

The NAT implementation was partially successful, allowing external hosts to reach internal hosts (especially h2), but failing to provide outbound connectivity from the internal network. The primary issues were incomplete NAT configuration, particularly missing SNAT rules for outbound traffic, and possibly disabled IP forwarding.

TASK 3:

The network consists of four nodes (0, 1, 2, and 3) connected with the following direct link costs:
Node 0: {0, 1, 3, 7} - Connected to all other nodes
Node 1: {1, 0, 1, ∞} - Connected to nodes 0 and 2, but not to node 3
Node 2: {3, 1, 0, 2} - Connected to all other nodes
Node 3: {7, ∞, 2, 0} - Connected to nodes 0 and 2, but not to node 1

````
The network topology can be visualized as:

```
    1
   / \
  /   \
 0-----2
  \   /
   \ /
    3
```
````

After initialization, each node only knows its direct link costs.
The network converges to a stable state after several rounds of updates.
The simulation also tests how the network responds to link cost changes.

Command used for running: gcc distance_vector.c node0.c node1.c node2.c node3.c -o output
>> ./output

We need to choose from trace 0 to 3:

Output for trace 0:

```
PS C:\Users\VENU GOPALROA\Desktop\TERM 2 MATERIALS\3-2\Computer Networks\A3\Mario\umass\cmpsci453\PA2> gcc distance_vector.c
 node0.c node1.c node2.c node3.c -o output
>> ./output
Enter TRACE:0
                via
   D0 |    1     2    3
  ----|-----------------
     1|    1   999   999
dest 2|  999     3   999
     3|  999   999     7
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999   999
dest 2|  999     1   999
     3|  999   999   999
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3   999   999
dest 1|  999     1   999
     3|  999   999     2
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999   999
dest 1|  999   999   999
     2|  999   999     2
                via
```

```
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2     6
dest 1|    4     1     5
     3|    7     4     2
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999     4
dest 1|    8   999     3
     2|    9   999     2

Simulator terminated at t=20001.978516, no packets in medium
```

The network took [X] iterations to reach a stable state. This is expected as the diameter of the network is small.

Path Selection:
The final routing tables show that:
- For node 0 to reach node 3, the optimal path is directly through node 3 with cost 7.
- For node 1 to reach node 3, the optimal path is through node 2 with a total cost of 3.
- For node 2 to reach node 1, the optimal path is directly through node 1 with cost 1.
- For node 3 to reach node 1, the optimal path is through node 2 with a total cost of 3.

Response to Link Changes:
When the cost of the link between nodes 0 and 1 was increased to 20:
- Node 0 started routing traffic to node 1 through node 2, resulting in a new cost of 4.
- Node 1 continued to route directly to node 0 but with the increased cost of 20.
- The change propagated through the network, affecting routes to other destinations.

When the link cost was restored to 1:
- The routing tables readjusted, returning to the optimal paths.

Output for trace 1:

```
PS C:\Users\VENU GOPALROA\Desktop\TERM 2 MATERIALS\3-2\Computer Networks\A3\Mario\umass\cmpsci453\PA2> gcc
ode0.c node1.c node2.c node3.c -o output
>> ./output
Enter TRACE:1
                via
   D0 |    1     2    3
  ----|-----------------
     1|    1   999   999
dest 2|  999     3   999
     3|  999   999     7
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999   999
dest 2|  999     1   999
     3|  999   999   999
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3   999   999
dest 1|  999     1   999
     3|  999   999     2
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999   999
dest 1|  999   999   999
     2|  999   999     2
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999   999
dest 2|    4     1   999
     3|    8   999   999
```

```
             via
   D0 |    1     2    3
  ----|-----------------
     1|    1     4    10
dest 2|    2     3     9
     3|    4     5     7
             via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
             via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2     6
dest 1|    4     1     5
     3|    7     4     2
             via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
             via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999     4
dest 1|    8   999     3
     2|    9   999     2

Simulator terminated at t=20001.978516, no packets in medium
```

Path selection:

With TRACE=1, the routing decisions are more visible, and we can confirm the following optimal paths:
- Node 0 routes to node 3 directly with cost 7, as shown in node 0's distance table where costs[3][3] = 7.
- Node 1 routes to node 3 through node 2 (costs[3][2] = 3), which can be traced as: Node 1 → Node 2 → Node 3 with a cost of 1+2=3.
- Node 2 routes to node 1 directly with cost 1, as confirmed by costs[1][1] = 1 in node 2's table.
- Node 3 routes to node 1 through node 2, as costs[1][2] = 3 in node 3's table, representing the path: Node 3 → Node 2 → Node 1 with a cost of 2+1=3.

Additionally, with TRACE=1, we can see each update propagate through the network, showing:

- Node 0 uses direct paths to nodes 1 and 3, but reaches node 2 through a direct path with cost 3.
- Node 1 uses a direct path to node 0 with cost 1, and to node 2 with cost 1.
- Node 2 uses direct paths to all its connected neighbors.
- Node 3 reaches node 0 directly with cost 7, but must reach node 1 through node 2.

With TRACE=1, we can see more detailed effects of the link cost change at t=10000:
- Immediately after the cost change, node 0 updates its cost to node 1:
costs[1][1] increases from 1 to 20.
- Within a few updates, node 0 discovers a better path to node 1 through node 2:
costs[1][2] = 4 (cost to node 2 = 3, then cost from node 2 to node 1 = 1, total = 4).
- Node 1 updates its direct cost to node 0: costs[0][0] increases from 1 to 20.
- Node 1 does not find a better alternative route to node 0, as all paths must eventually go through the expensive link.
- The impact ripples through the network, causing updates to routes that previously went through the 0-1 link.
- Nodes 2 and 3 adjust their routing tables based on the updates received from nodes 0 and 1.
- Node 2 continues to route to node 1 directly with cost 1 and to node 0 directly with cost 3.
- Node 3's path to node 1 remains through node 2 with cost 3, unaffected by the 0-1 link change.

When the link cost was restored to 1 at t=20000:
- The routing tables quickly readjust, with node 0's costs[1][1] returning to 1.
- Node 0 starts routing to node 1 directly again.
- Node 1's costs[0][0] returns to 1, restoring the direct path to node 0.
- The network converges again to the original optimal paths within a few update rounds.

Output for trace 2:

```
PS C:\Users\VENU GOPALROA\Desktop\TERM 2 MATERIALS\3-2\Computer Networks\A3\Mario\
c node1.c node2.c node3.c -o output
>> ./output
Enter TRACE:2
                via
   D0 |    1     2    3
  ----|-----------------
     1|    1   999  999
dest 2|  999     3  999
     3|  999   999    7
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999  999
dest 2|  999     1  999
     3|  999   999  999
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3   999  999
dest 1|  999     1  999
     3|  999   999    2
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999  999
dest 1|  999   999  999
     2|  999   999    2
MAIN: rcv event, t=0.094, at 1 src: 0, dest: 1, contents:   0   1   3   7
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999  999
dest 2|    4     1  999
     3|    8   999  999
MAIN: rcv event, t=0.427, at 1 src: 2, dest: 1, contents:   3   1   0   2
```

```
MAIN: rcv event, t=20000.000, at 7143521                via
   D0 |    1     2    3
  ----|-----------------
     1|    1     4    10
dest 2|    2     3     9
     3|    4     5     7
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
MAIN: rcv event, t=20000.191, at 2 src: 0, dest: 2, contents:   0   1   2   4
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2     6
dest 1|    4     1     5
     3|    7     4     2
MAIN: rcv event, t=20000.992, at 1 src: 0, dest: 1, contents:   0   1   2   4
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
MAIN: rcv event, t=20001.979, at 3 src: 0, dest: 3, contents:   0   1   2   4
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999     4
dest 1|    8   999     3
     2|    9   999     2

Simulator terminated at t=20001.978516, no packets in medium
```

With TRACE=2, we get even more detailed logging that includes packet contents and precise event timings:

- The exact mincost vector in each packet becomes visible, showing exactly what information each node is sending to its neighbors.
- We can see the scheduling of packet arrivals with precise timestamps (e.g., "scheduling arrival on other side at time X").
- Internal packet processing becomes visible, showing how each node interprets the received information.
- The output confirms that the network convergence process follows the distributed Bellman-Ford algorithm properly.
- For link cost changes, we can observe the precise sequence of packets generated, including:
  - Immediate notification to directly affected nodes (0 and 1)
  - The cascade of updates as indirect routes adjust
  - The exact timing of convergence after the link cost changes

For example, when the link cost changes at t=10000:

- We see the event generation for link change
- The immediate updates to nodes 0 and 1's distance tables
- The exact sequence of packets with timestamps as the changes propagate
- The complete contents of each update packet (sourceid, destid, and all mincost[] values)

Output for Trace 3:

```
    TOLAYER2: scheduling arrival on other side
    TOLAYER2: source: 3, dest: 2
              costs:7  999  2  0
    TOLAYER2: scheduling arrival on other side
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999   999
dest 1|  999   999   999
     2|  999   999     2
MAIN: rcv event, t=0.094, at 1 src: 0, dest: 1, contents:   0   1   3   7
    TOLAYER2: source: 1, dest: 0
              costs:1  0  1  8
    TOLAYER2: scheduling arrival on other side
    TOLAYER2: source: 1, dest: 2
              costs:1  0  1  8
    TOLAYER2: scheduling arrival on other side
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1   999   999
dest 2|    4     1   999
     3|    8   999   999
MAIN: rcv event, t=0.427, at 1 src: 2, dest: 1, contents:   3   1   0   2
    TOLAYER2: source: 1, dest: 0
              costs:1  0  1  3
    TOLAYER2: scheduling arrival on other side
    TOLAYER2: source: 1, dest: 2
              costs:1  0  1  3
    TOLAYER2: scheduling arrival on other side
                via
```

```
MAIN: rcv event, t=20000.191, at 2 src: 0, dest: 2, contents:   0   1   2   4
                via
   D2 |    0     1    3
  ----|-----------------
     0|    3     2     6
dest 1|    4     1     5
     3|    7     4     2
MAIN: rcv event, t=20000.992, at 1 src: 0, dest: 1, contents:   0   1   2   4
                via
   D1 |    0     2    3
  ----|-----------------
     0|    1     3   999
dest 2|    1     1   999
     3|    3     3   999
MAIN: rcv event, t=20001.979, at 3 src: 0, dest: 3, contents:   0   1   2   4
                via
   D3 |    0     1    2
  ----|-----------------
     0|    7   999     4
dest 1|    8   999     3
     2|    9   999     2

Simulator terminated at t=20001.978516, no packets in medium
```

With TRACE=3, we get the maximum level of detail about the internal operations of the simulation:
- Full event list management becomes visible, showing how events are inserted, processed, and removed from the event queue.
- The precise timing decisions for packet deliveries become transparent, demonstrating how the simulator prevents packet reordering.
- Memory management for packets and events becomes visible.
- The simulation's random number generation is fully exposed, allowing verification of its statistical properties.
- The complete life cycle of each event from creation to execution to cleanup is traceable.

It is valuable for debugging and verifying the correctness of the simulation framework itself, beyond just the routing algorithm implementation. It allows us to confirm that packets are being properly created, scheduled, delivered, and processed according to the simulation model's specifications.

For example, when the link cost changes at t=10000:
- The event insertion details show exactly how the link change event is scheduled
- We can see verification that all nodes are properly initialized before events begin processing
- Each event's complete memory lifecycle becomes visible

With the higher level of detail in TRACE=1 and TRACE=2, we can analyze the exact sequence of updates for each packet:
- When the simulation begins, we see each node sending its initial distance vector to its neighbors.
- As updates arrive, we can track how information propagates, with the most recent information overriding older data.

- We observe the count-to-infinity problem being mitigated through proper handling of updates.
- The output clearly demonstrates how the Bellman-Ford equation is applied at each step to compute shortest paths.