NETWORKING TRAINING - MODULE 7 & 8

Akash S, embedUR Systems

Qn 1:

Try Test-Connection and nslookup commands for below websites www.google.com, www.facebook.com www.amazon.com, www.github.com, www.cisco.com

nslookup:

- nslookup - **Name Server Lookup**
- It is a command-line tool used to query Domain Name System (DNS) servers to obtain domain name or IP address mappings.

What does nslookup do?

- Resolves a domain name to its corresponding IP address.
- Queries specific DNS servers for domain information.
- We can diagnose DNS related network issues.

How it works?

- When we type a domain name, nslookup contacts the system's DNS resolver.
- DNS resolver queries appropriate DNS server to get IP address
- If the queried server is non-authoritative, it forwards the request to the authoritative DNS server.
- Finally the domain name and its mapped IP is shown

```
akash@akash:~$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.183.196
Name:   www.google.com
Address: 2404:6800:4009:826::2004
```

```
akash@akash:~$ nslookup www.facebook.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
www.facebook.com        canonical name = star-
mini.c10r.facebook.com.
Name:   star-mini.c10r.facebook.com
Address: 163.70.138.35
Name:   star-mini.c10r.facebook.com
Address: 2a03:2880:f184:81:face:b00c:0:25de
```

```
akash@akash:~$ nslookup www.github.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
www.github.com  canonical name = github.com.
Name:   github.com
Address: 20.207.73.82
```

```
akash@akash:~$ nslookup www.amazon.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
www.amazon.com  canonical name = tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com        canonical name = d3ag4hukkh62yn.cloudfront.net.
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 18.161.217.215
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:a00:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:8a00:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:0:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:1400:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:d000:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:9200:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:dc00:7:49a5:5fd4:b121
Name:   d3ag4hukkh62yn.cloudfront.net
Address: 2600:9000:24d9:8400:7:49a5:5fd4:b121
```

```
akash@akash:~$ nslookup www.cisco.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
www.cisco.com   canonical name = www.cisco.com.akadns.net.
www.cisco.com.akadns.net        canonical name = wwwds.cisco.com.edgekey.net.
wwwds.cisco.com.edgekey.net     canonical name = wwwds.cisco.com.edgekey.net.globalredir.akadns.net.
wwwds.cisco.com.edgekey.net.globalredir.akadns.net    canonical name = e2867.dsca.akamaiedge.net.
Name:   e2867.dsca.akamaiedge.net
Address: 23.63.219.126
Name:   e2867.dsca.akamaiedge.net
Address: 2600:140f:4:e8e::b33
Name:   e2867.dsca.akamaiedge.net
Address: 2600:140f:4:eb1::b33
```

Qn 2:

. Use Wireshark to capture and analyze DNS, TCP, UDP traffic and packet header, packet flow, options and flags

Capturing DNS Traffic:

1

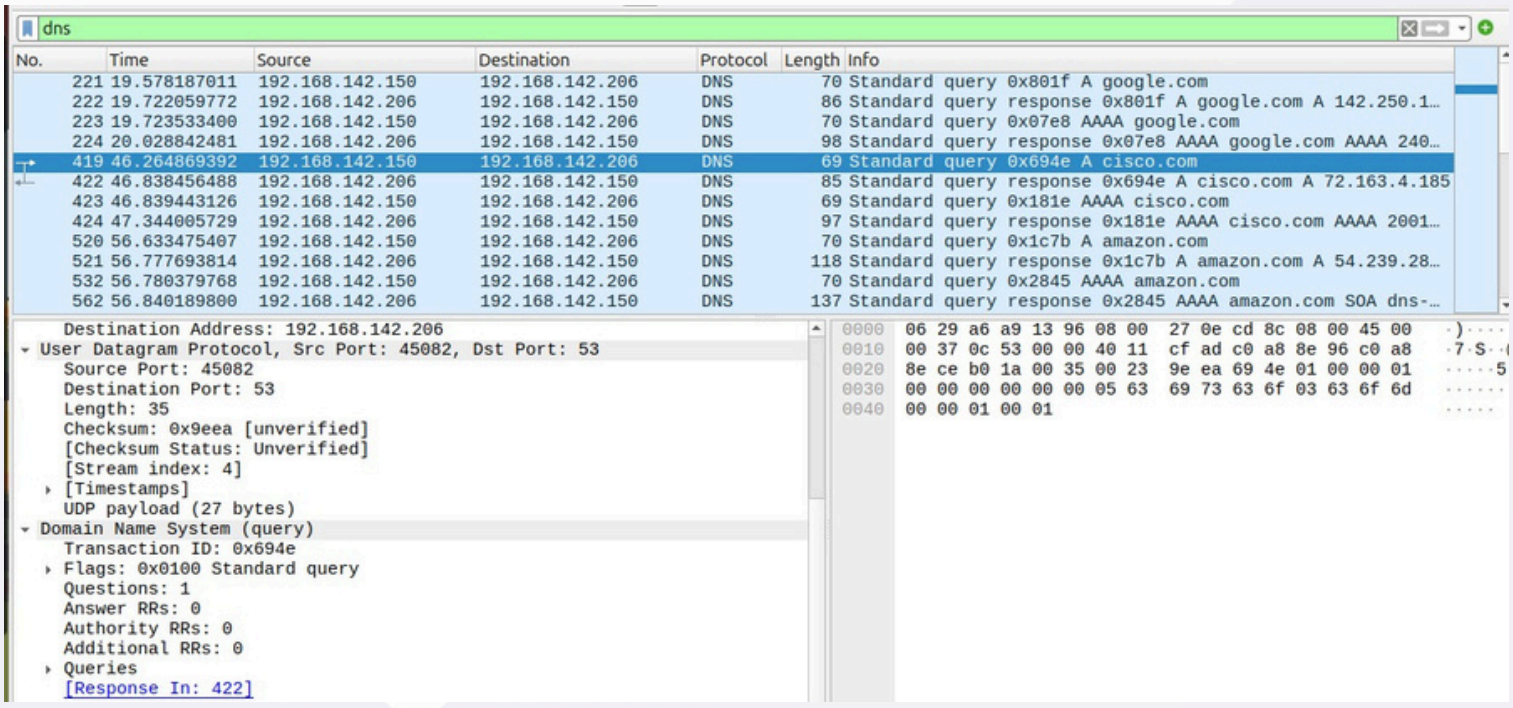
- By running this command, we can capture DNS traffic
  - **nslookup embedur.ai**
- In wireshark, we can filter the traffic with the keyword “**dns**”

```
akash@akash:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.199.142
Name:   google.com
Address: 2404:6800:4009:828::200e
```

google.com

```
akash@akash:~$ nslookup cisco.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   cisco.com
Address: 72.163.4.185
Name:   cisco.com
Address: 2001:420:1101:1::185
```

cisco.com



dns traffic in wireshark

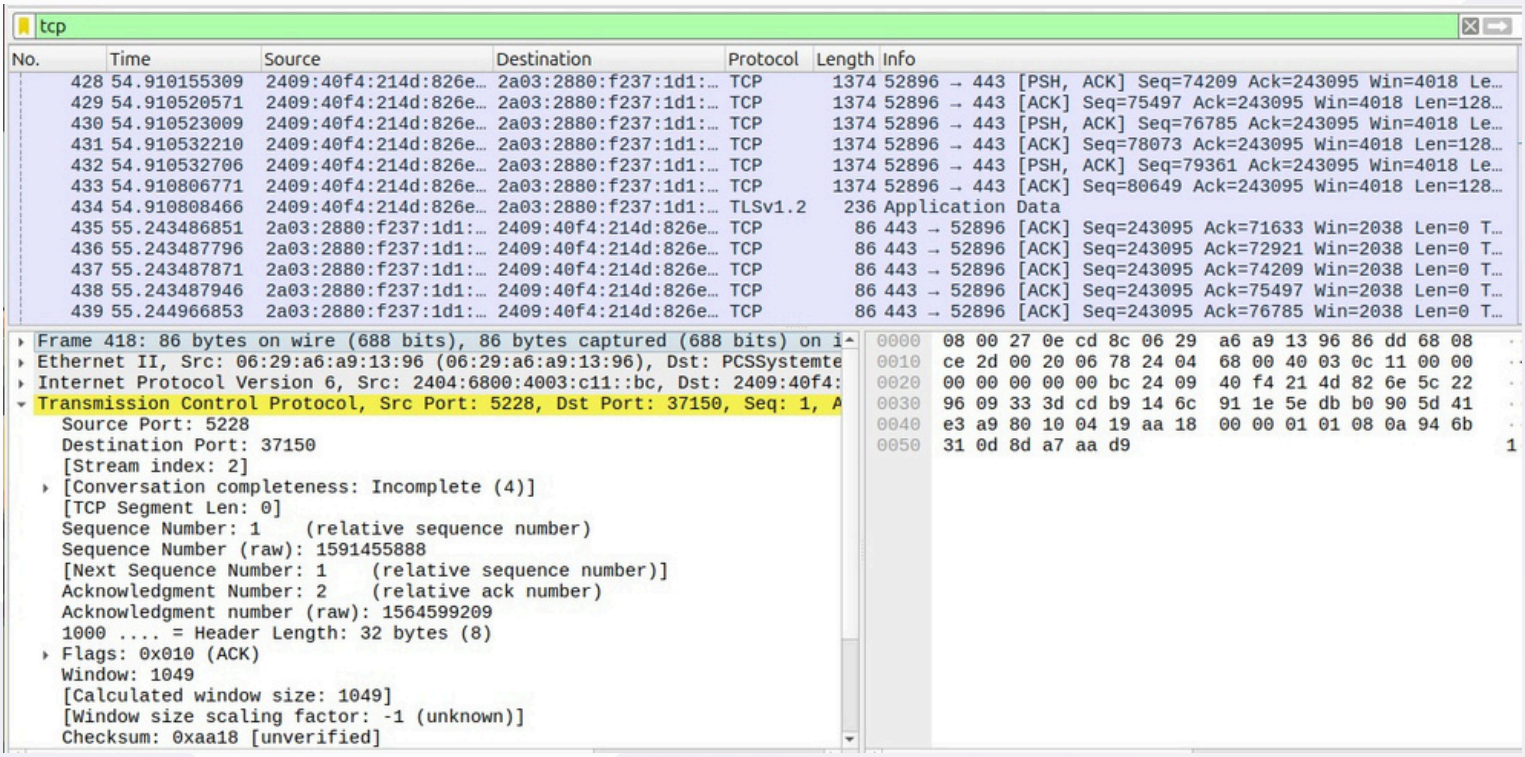
Capturing TCP Traffic:

2

- By running this command, we can capture TCP traffic
  - curl http://embedur.ai
- We can also capture tcp traffic by running any website in the machine
- In wireshark, we can filter the traffic with the keyword **“tcp”**

```
akash@akash:~$ curl http://google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

curl command



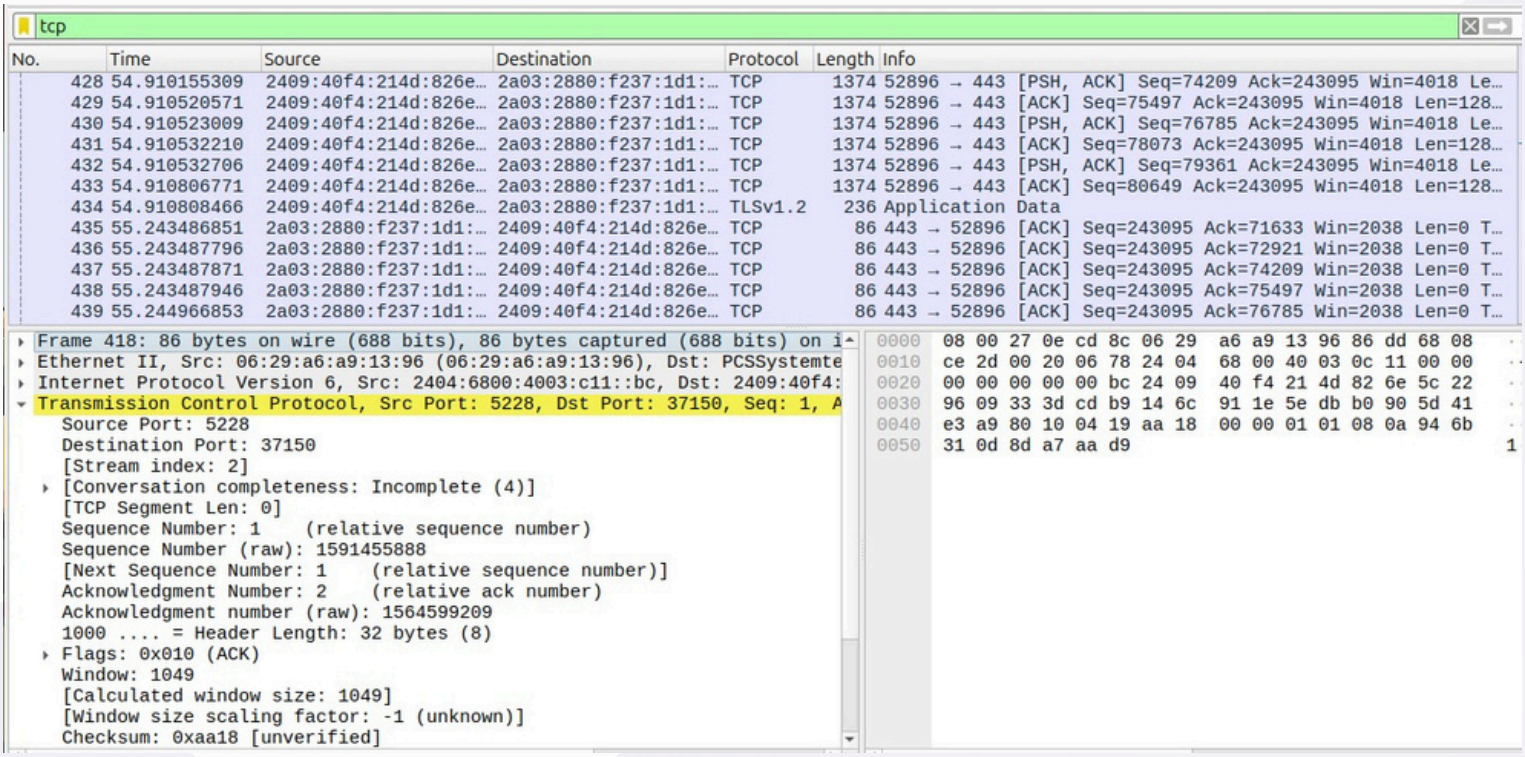
tcp traffic in wireshark

Capturing UDP Traffic:

- By running this command, we can capture TCP traffic
  - **dig embedur.ai**
- In wireshark, we can filter the traffic with the keyword **“udp”**

```
akash@akash:~$ curl http://google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

dig command



udp traffic in wireshark

Qn 3:

Explore traceroute/tracert for different websites eg:google.com and analyse the parameters in the output and explore different options for traceroute command.

Traceroute:

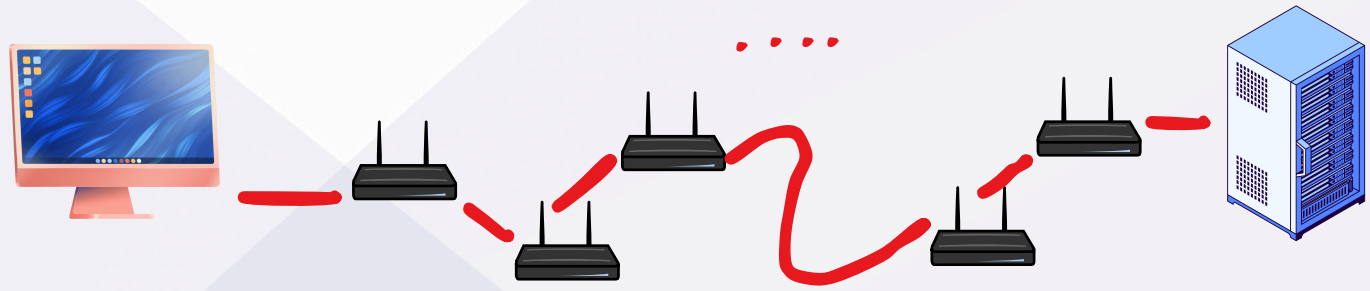
- Traceroute is a network diagnostic tool used to track the path that packets take from a source system to a destination across an IP network.
- It helps identify network congestion, routing issues, and unreachable nodes.

How traceroute works:

- While sending, the TTL value increases from 1
- Upon reaching each and every router, the TTL value is decremented by 1
    - When TTL=0, packet is dropped
- traceroute identifies time taken by each router to respond
- Once target responds, full path is recorded

Options in traceroute:

- -I : ICMP
- -T : TCP
- -p 23 : Destination Port number
- -m 35 : Maximum hops needed
- -w <time> : Timeout for each response
- -4 : IPv4
- -6 : IPv6



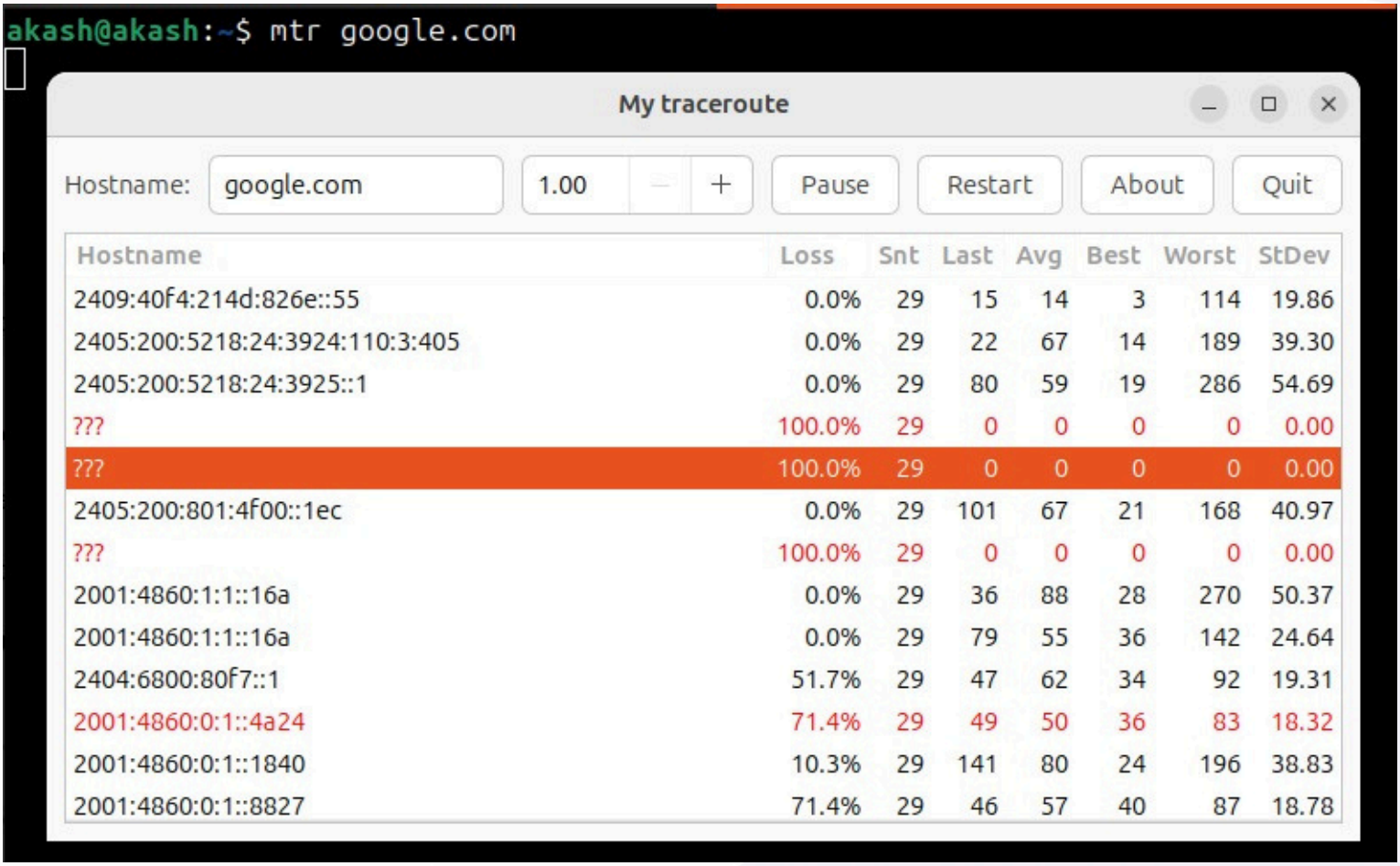
traceroute:

```
akash@akash:~$ sudo traceroute -I embedur.ai
traceroute to embedur.ai (162.159.136.54), 30 hops max, 60 byte packets
 1  _gateway (192.168.142.206)  3.011 ms  2.980 ms  2.973 ms
 2  192.0.0.1 (192.0.0.1)  5.740 ms  5.735 ms *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  162.159.136.54 (162.159.136.54)  48.023 ms  48.019 ms  48.014 ms
akash@akash:~$
```

```
akash@akash:~$ sudo traceroute -T -p 443 google.com
traceroute to google.com (142.250.205.78), 30 hops max, 60 byte packets
 1  _gateway (192.168.142.206)  4.163 ms  5.318 ms  5.283 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  pnmaaa-ar-in-f14.1e100.net (142.250.205.78)  42.822 ms  77.444 ms  49.889 ms
```

```
akash@akash:~$ traceroute www.akashn.com
traceroute to www.akashn.com (76.76.21.142), 30 hops max, 60 byte packets
 1  _gateway (192.168.142.206)  3.735 ms  3.666 ms  5.109 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
```

Detailed traceroute analysis using mtr:



Hostname	Loss	Snt	Last	Avg	Best	Worst	StDev
2409:40f4:214d:826e::55	0.0%	29	15	14	3	114	19.86
2405:200:5218:24:3924:110:3:405	0.0%	29	22	67	14	189	39.30
2405:200:5218:24:3925::1	0.0%	29	80	59	19	286	54.69
???	100.0%	29	0	0	0	0	0.00
???	100.0%	29	0	0	0	0	0.00
2405:200:801:4f00::1ec	0.0%	29	101	67	21	168	40.97
???	100.0%	29	0	0	0	0	0.00
2001:4860:1:1::16a	0.0%	29	36	88	28	270	50.37
2001:4860:1:1::16a	0.0%	29	79	55	36	142	24.64
2404:6800:80f7::1	51.7%	29	47	62	34	92	19.31
2001:4860:0:1::4a24	71.4%	29	49	50	36	83	18.32
2001:4860:0:1::1840	10.3%	29	141	80	24	196	38.83
2001:4860:0:1::8827	71.4%	29	46	57	40	87	18.78

Qn 4 - 10:

Network Topology in Cisco Packet Tracer

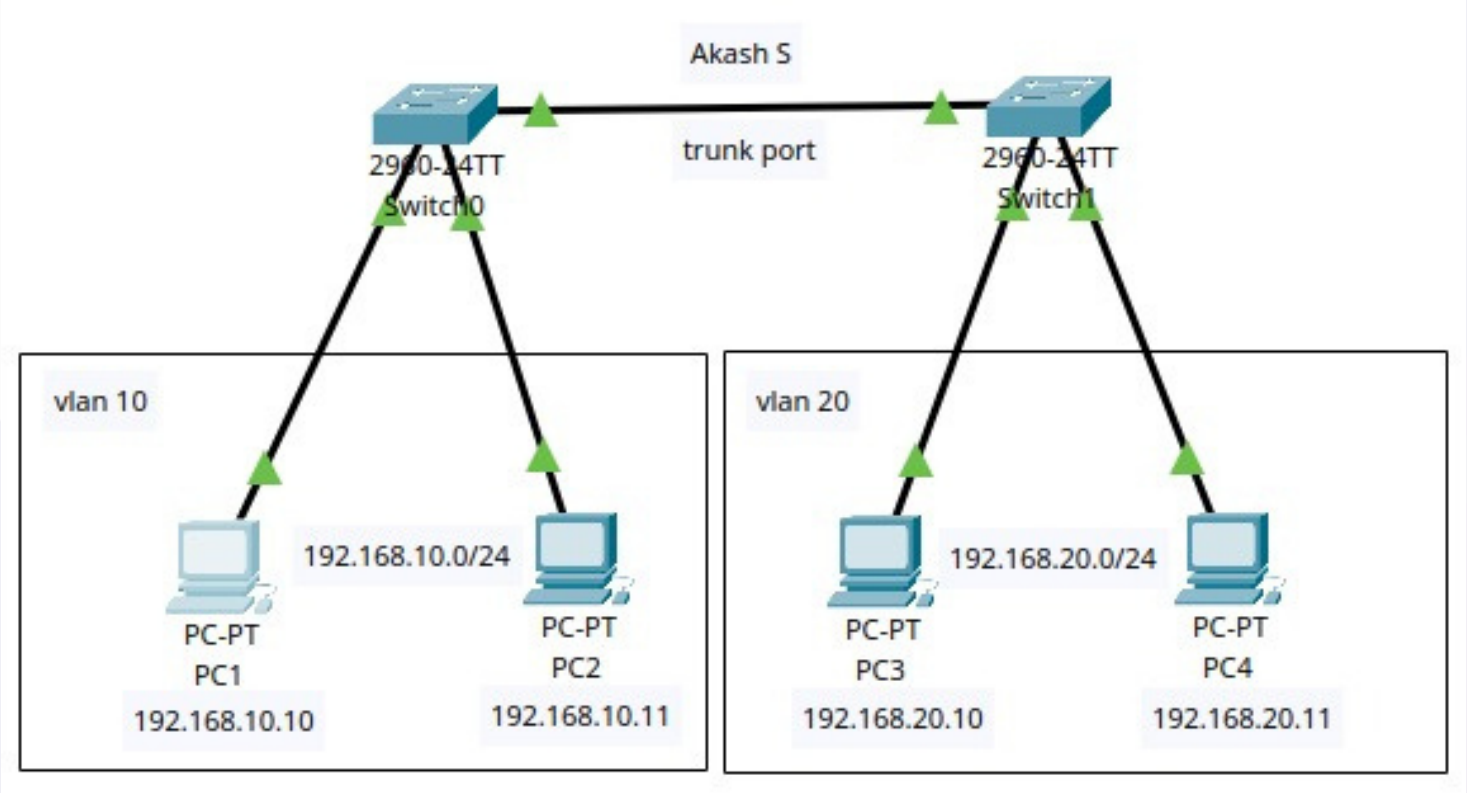
- VLAN, SSH/Telnet, Troubleshooting, Inter VLAN Routing

VLAN:

- VLANs (Virtual Local Area Networks)
- VLANs segment a network to separate traffic logically rather than physically.
- Devices in the same VLAN can communicate, but different VLANs need a Layer 3 device (router or Layer 3 switch) for communication.

Trunk Port:

- A trunk port is a switch port that carries multiple VLANs over a single physical link between network devices like switches or routers.
- It uses **802.1Q encapsulation** to tag VLAN traffic to differentiate different vlan when they travel across the trunk.



Pinging from PC1 to PC2:

```
C:\>ping 192.168.10.11

Pinging 192.168.10.11 with 32 bytes of data:

Reply from 192.168.10.11: bytes=32 time<1ms TTL=128
Reply from 192.168.10.11: bytes=32 time<1ms TTL=128
Reply from 192.168.10.11: bytes=32 time<1ms TTL=128
Reply from 192.168.10.11: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.10.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

There ‘s a problem!

- Pinging works only inside the same vlan
- When I pinged from PC 1 to PC2, it worked
- When I pinged from PC3 to PC4, it worked
- But when I pinged from PC1 to PC3 or PC4, it didn’t work

PC1

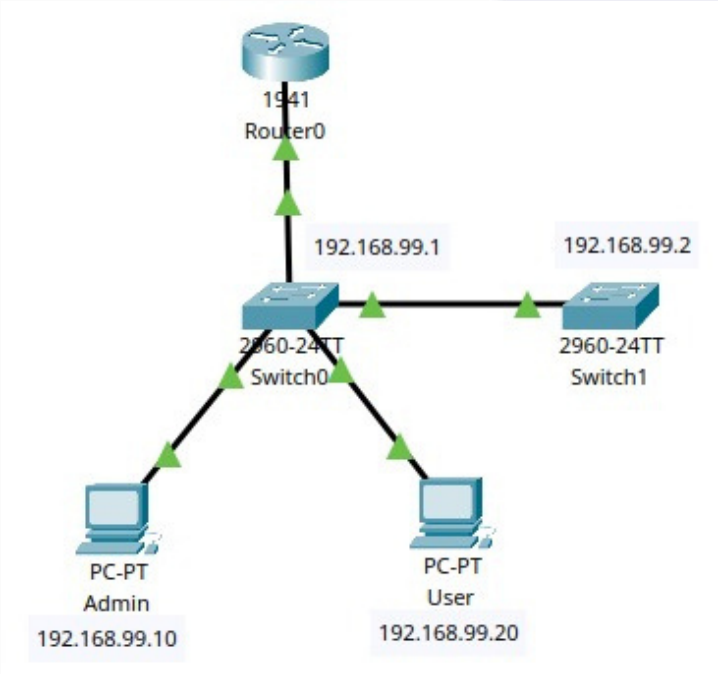
Physical | Config | Desktop | Programming | Attributes

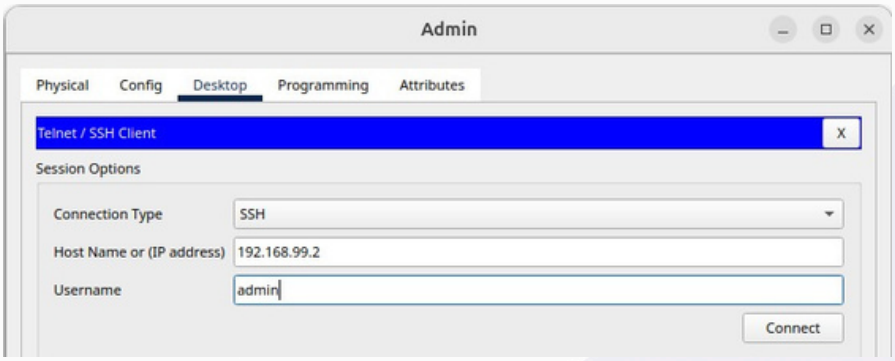
Command Prompt

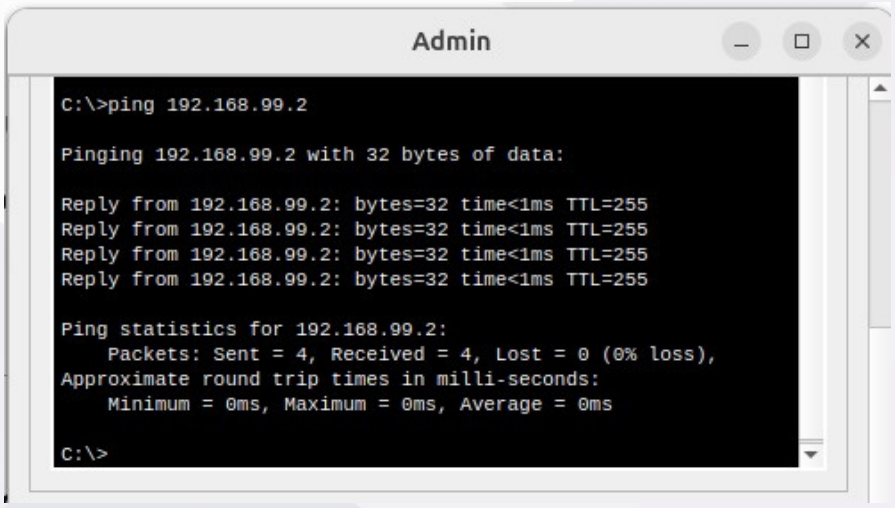
```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.20.11

Pinging 192.168.20.11 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.20.11:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

Testing SSH/Telnet:

- I configured a network by connecting two PCs for remote access
    - PC1: **192.168.99.10**
    - PC2: **192.168.99.20**
- I connected two switches and a router
- When I tried connected to the host machine using ssh, it worked
    - **ssh akash@192.168.99.2**

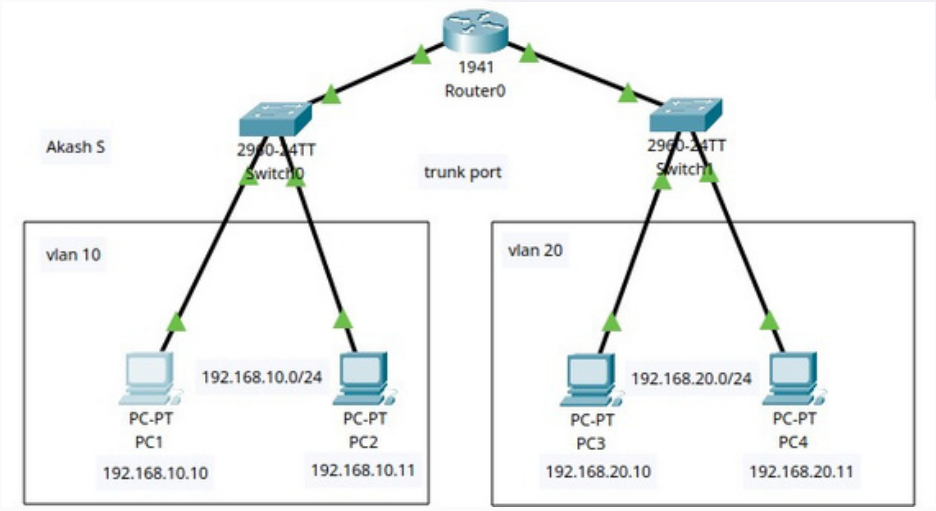


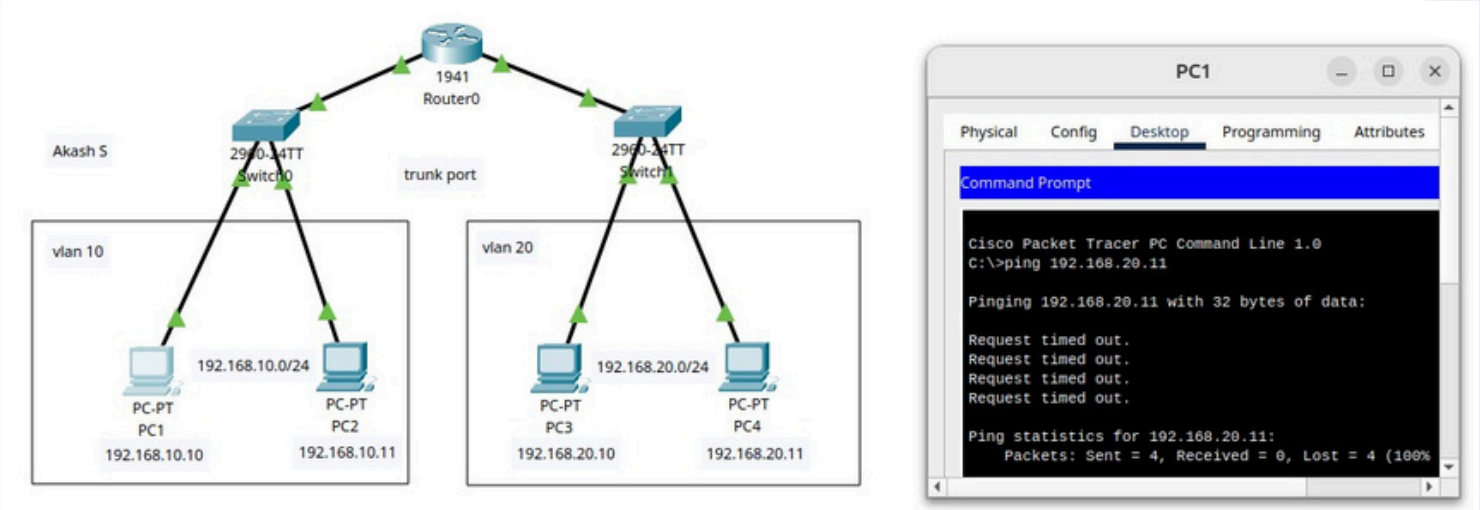


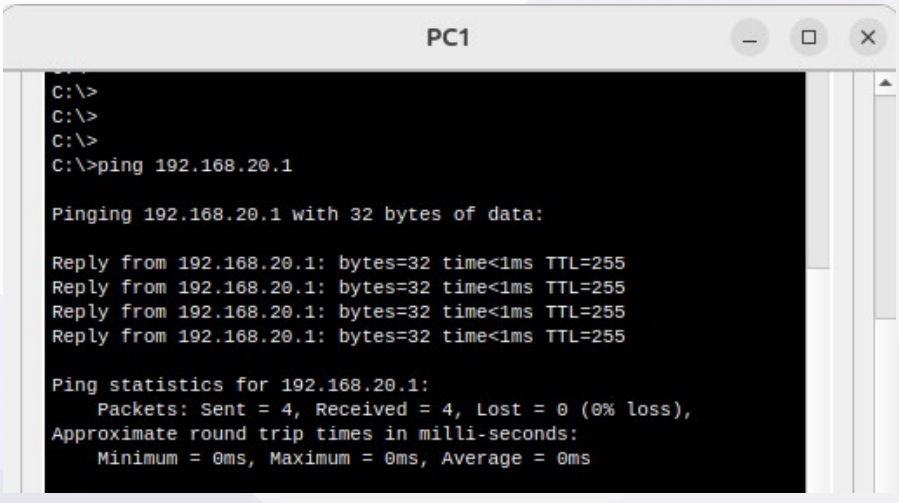


Troubleshooting Inter-VLAN pinging:

- When I tried pinging from different VLAN to a different VLAN, it failed.
- What could be the reason?
- After several attempts, I realized that VLANs need routers to connect across different VLANs.
- So I connected a router connecting two switches.







Qn 11:

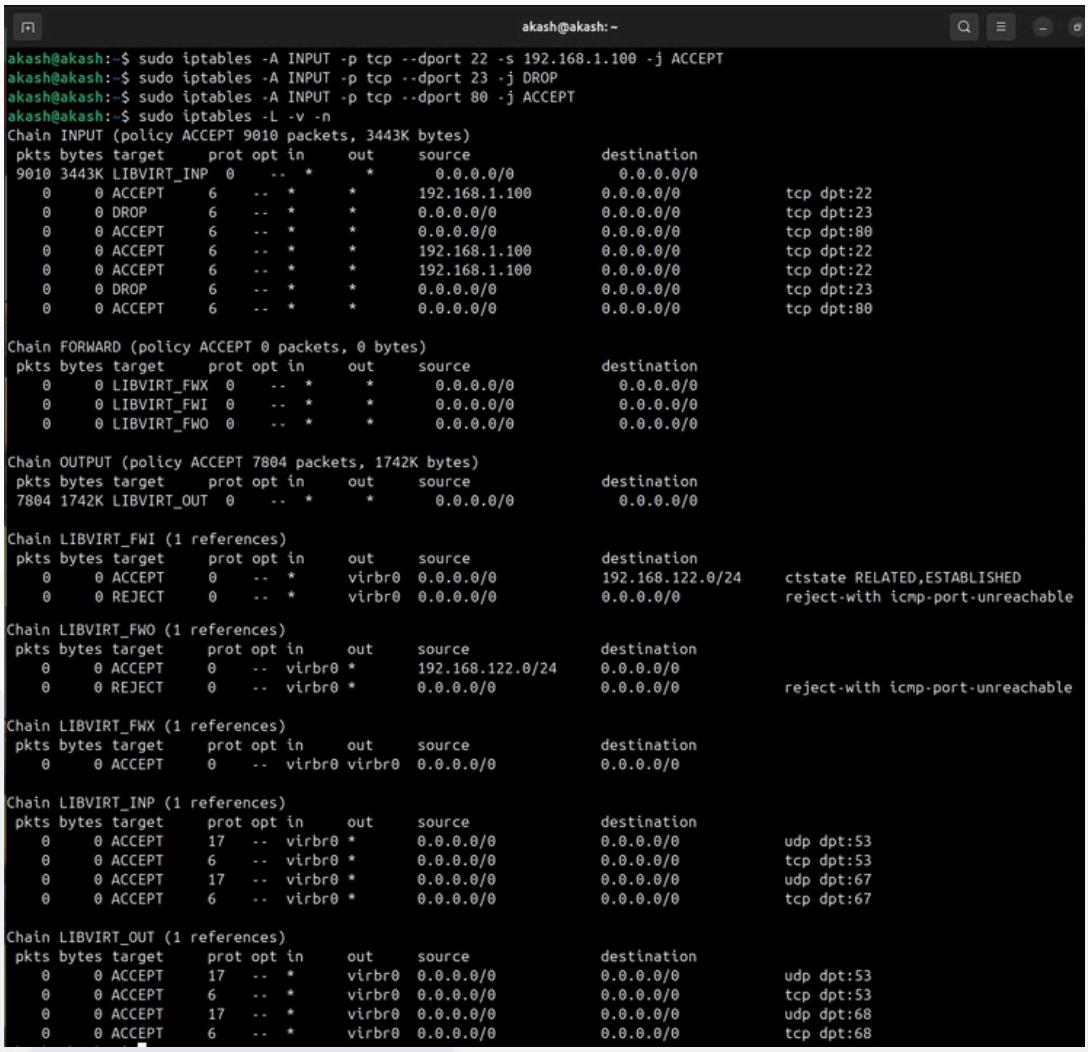
Implement ACLs to restrict traffic based on source and destination ports. Test rules by simulating legitimate and unauthorized traffic.

ACL:

- **Access Control Lists** are used to filter network traffic based on rules. They can control which packets are allowed or denied based on parameters such as source/destination IP addresses, protocols, and ports. This helps in enforcing security policies and managing traffic efficiently.

Steps I followed:

- I created an extended ACL
- I blocked SSH (port 22)
- I allowed HTTP (port 80)
- I also blocked ICMP ping requests to check how it works.



Testing how it works:

```
akash@akash:~$ ssh akash@192.168.29.220
akash@192.168.29.220's password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.11.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

17 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

8 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Sat Mar  1 13:50:50 2025 from 192.168.29.112
```

```
akash@akash:~$ sudo iptables -A INPUT -p tcp --dport 80 -j DROP
[sudo] password for akash:
akash@akash:~$ curl http://192.168.29.220
curl: (7) Failed to connect to 192.168.29.220 port 80 after 0 ms: Couldn't
 connect to server
akash@akash:~$
```

Two main commands:

- **ACCEPT:** To allow the port
- **DROP:** To block the port

Takeaway:

- The above captures explain how the access is denied or controlled based on the specific port or ip address
- It is useful when we want to allow only limited IPs or Ports in an organization

Qn 12 & 13:

Configure a standard Access Control List (ACL) on a router to permit traffic from a specific IP range. Test connectivity to verify the ACL is working as intended.

Create an extended ACL to block specific applications, such as HTTP or FTP traffic. Test the ACL rules by attempting to access blocked services.

Permit IP Range in ACL:

Steps I followed:

- I tried this task by setting a network topology in the Cisco Packet Tracer.
- I decided to allow traffic only from 192.168.1.0/24 and block all other sources.
- Since standard ACLs filter only based on source IP, I had to ensure correct placement.

Configuring ACL on router:

- I used these commands,
    - **access-list 10 permit 192.168.1.0 0.0.0.255**
    - **access-list 10 deny any**

Applying ACL to interface:

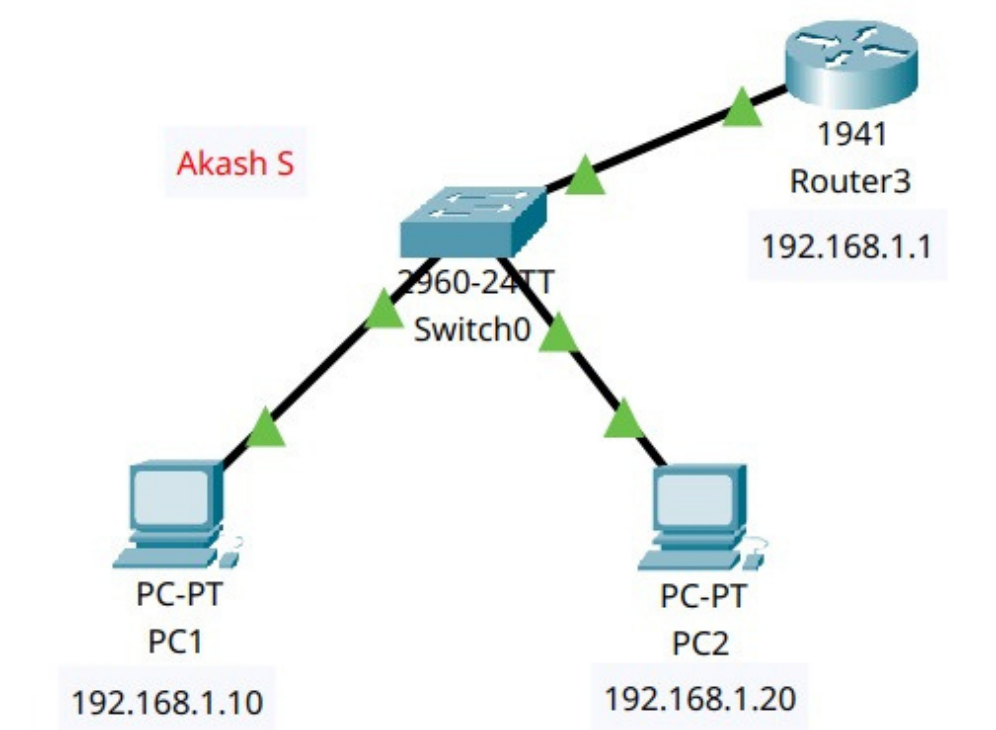
- These commands are used to apply ACL to interface
    - **interface GigabitEthernet0/0**
    - **ip access-group 10 in**
    - **exit**

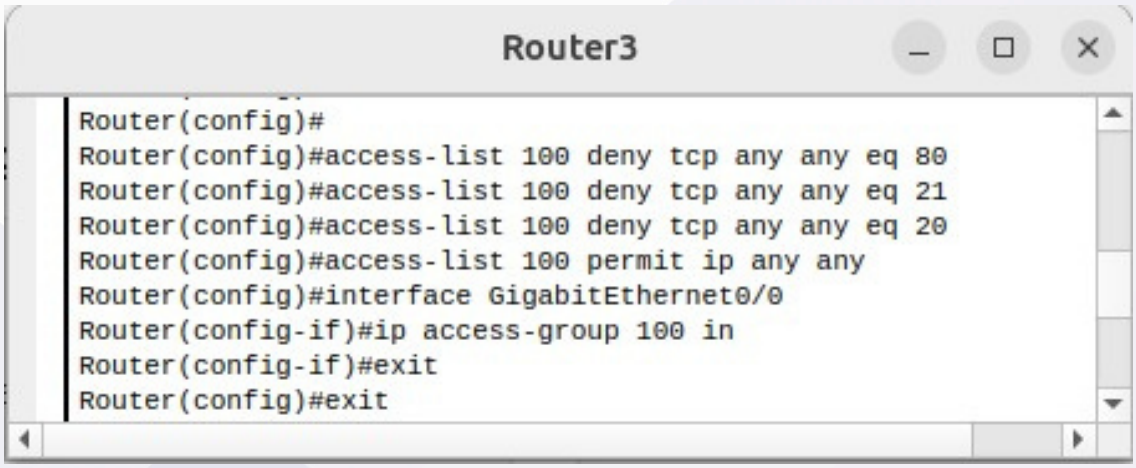
Testing:

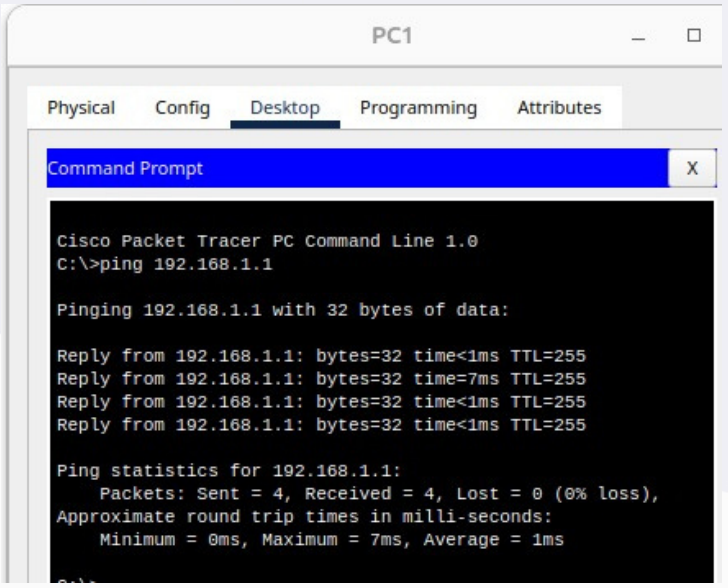
- A PC from the 192.168.1.0/24 range successfully accessed the router.
- A PC from a different subnet was unable to connect.
- Ping tests from an unauthorized subnet failed, confirming that the ACL was applied correctly.

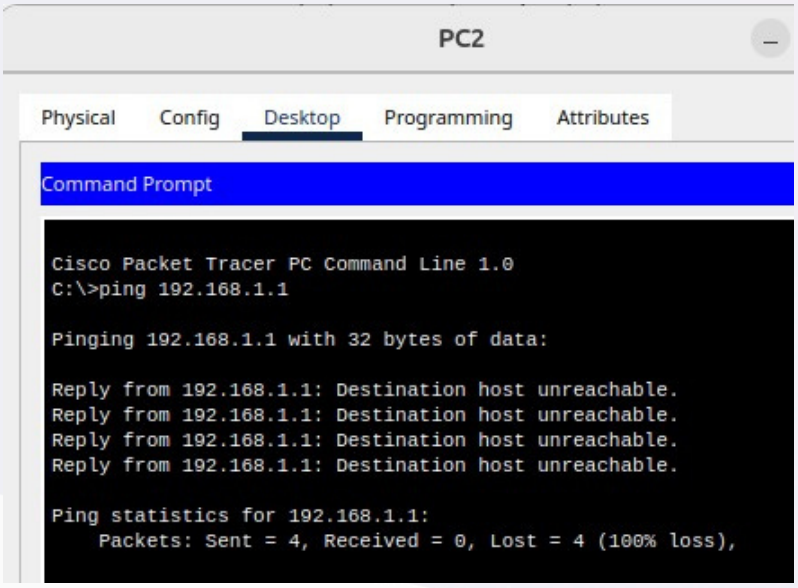
Attempt 1:

- I created a simple network to test first.
- Added two PCs, a switch and a router.





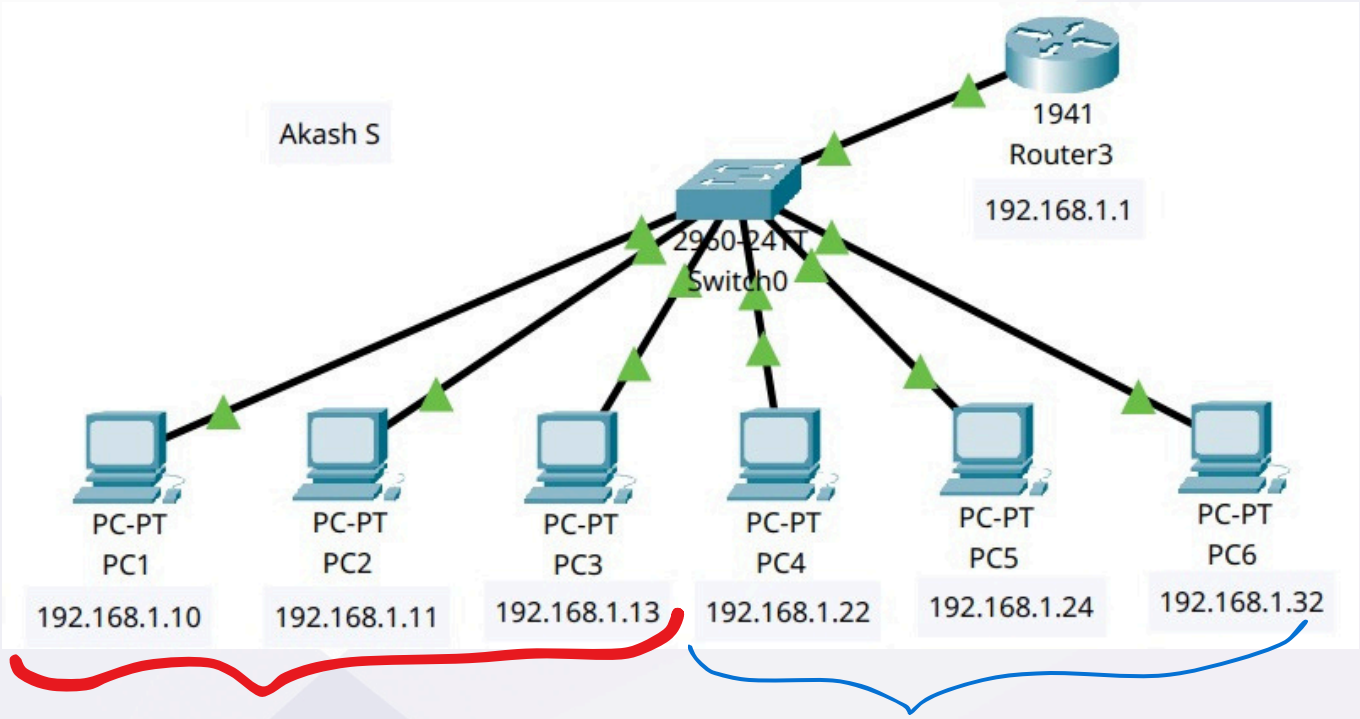




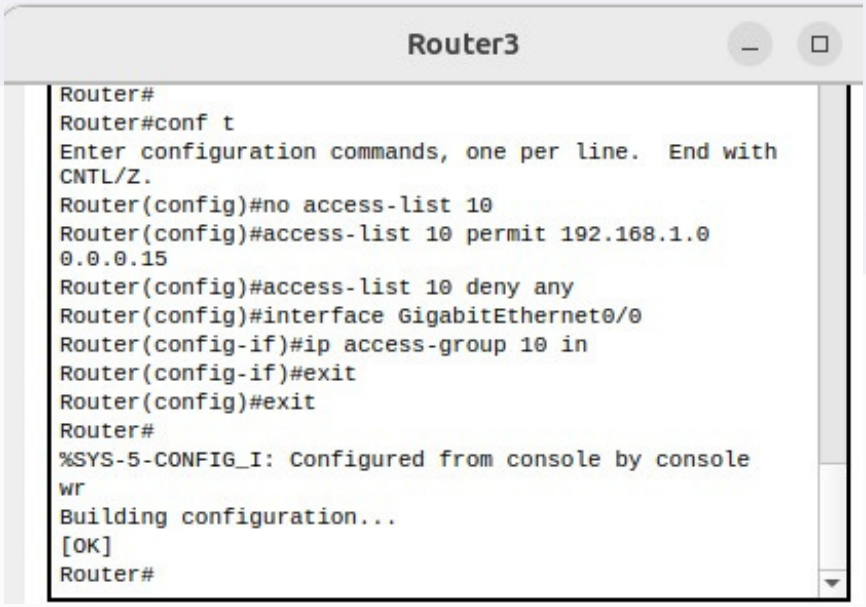
- For testing purpose, I created a simple network to allow and check
- Finally, the test network worked
- So, implementing the actual network with multiple devices

Actual Network:

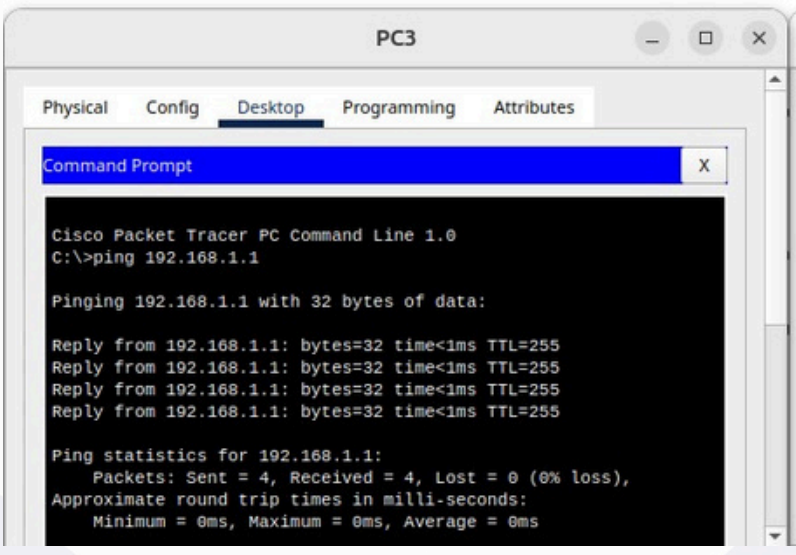
- I constructed a network with 6 PCs, and I mentioned the network as **/28** so it allowed the range 0-15
- I used a wildcard mask to allow only first 15 and block the rest
- I configured this on the router



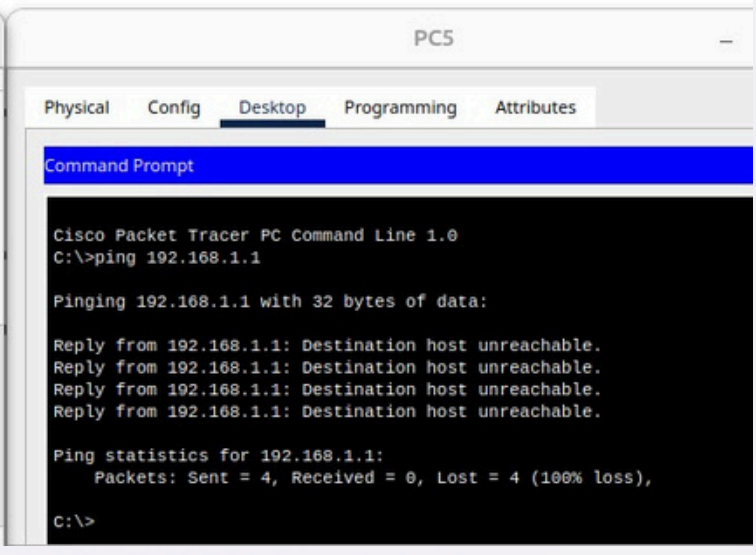
Router Configuration:



- After masking with wildcard mask, the allowed and denied PCs are
  - PC1, PC2 & PC3 - range: **192.168.1.0 - 192.168.1.15**



PC3 **success!**

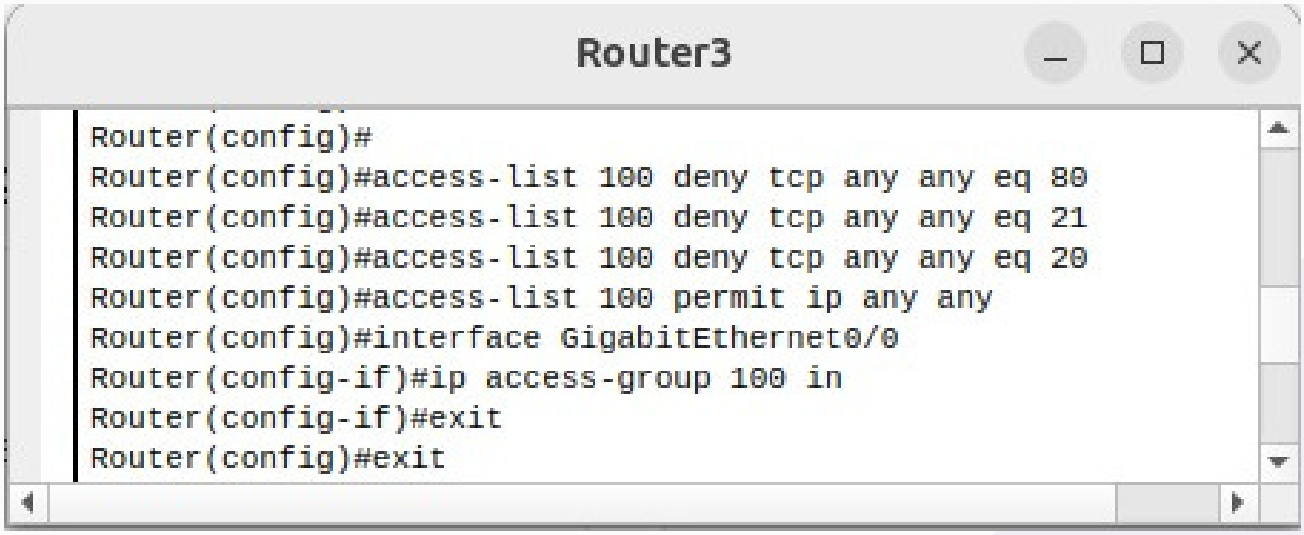


PC5 **failed!**

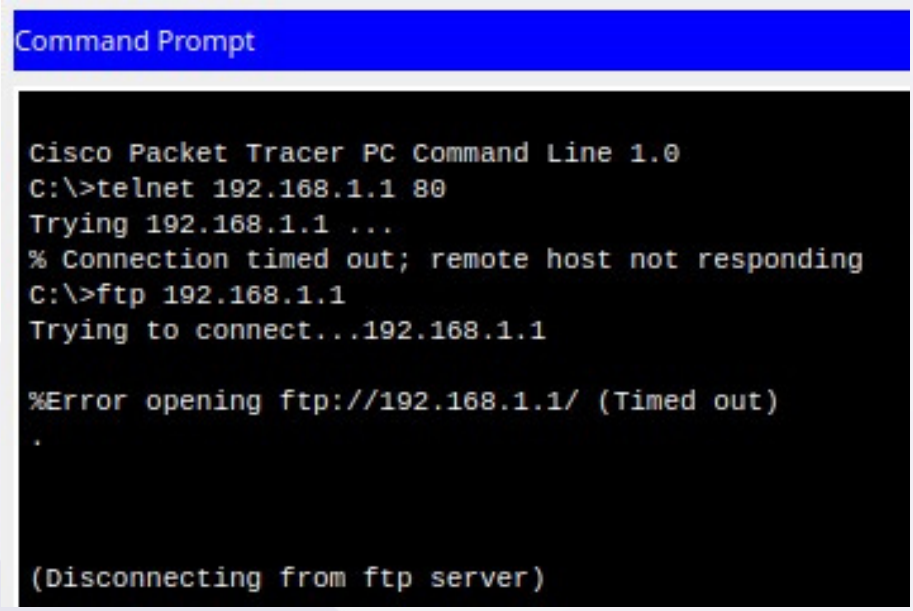
13 Permissions:

- Allowing the HTTP Port:
  - **access-list 100 deny tcp any any eq 80**
- Allowing the FTP Port - Data Transfer:
  - **access-list 100 deny tcp any any eq 20**
- Allowing the FTP Port - Control Channel:
  - **access-list 100 deny tcp any any eq 21**

Router Configuration:



Results after updating the permission:

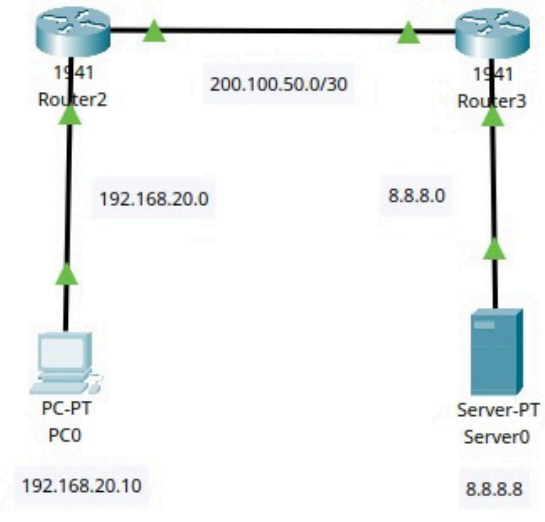


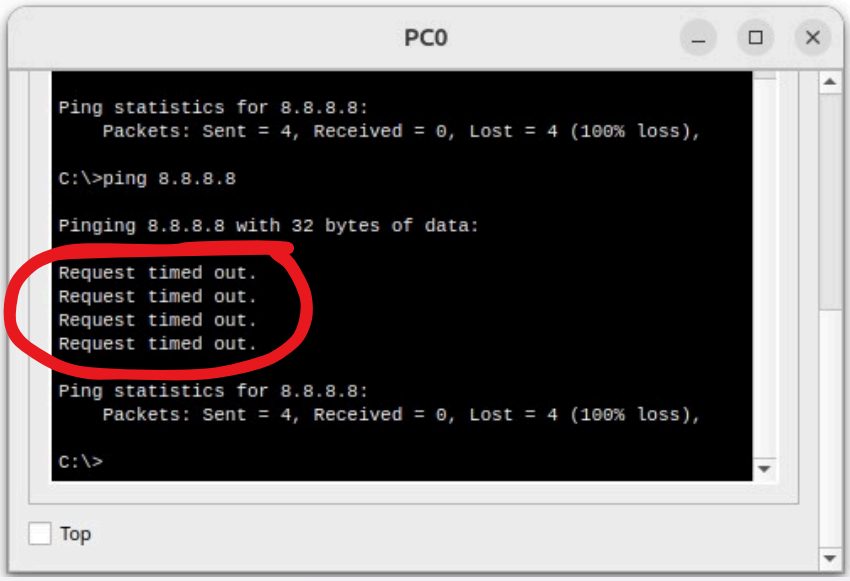
Qn 14:

Network Address Translation Task

NAT configuration:

- I started by creating a small network with a PC, two routers and a server
- Here are the IP configurations:
    - PC: **192.168.20.10**
    - Server: **8.8.8.8**
- I allowed the routers to trunk
- I configured NAT inside and NAT outside
- Router inside IP:
    - PC - Router1: **192.168.20.1**
    - Router1 - Router 2: **200.100.50.1**
- Router outside IP:
    - Router 1 - Router2: **200.100.50.2**
    - Router2 - Server: **8.8.8.1**
- I tried several attempts to solve this and configured the final network.
- I can’t access the server





Actual Network:

- There was a problem in between as I couldn’t reach Router 2 from **PC1**
- I tried debugging through Router’s CLI and PC1's Command prompt
- Configured
  - **IP NAT Inside**
  - **IP NAT Outside**

```
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface GigabitEthernet0/1
Router(config-if)#ip nat inside
Router(config-if)#exit
Router(config)#
Router(config)#
Router(config)#
Router(config)#interface GigabitEthernet0/0
Router(config-if)#ip nat outside
Router(config-if)#exit
Router(config)#
Router(config)#
Router(config)#ip nat inside source ?
  list    Specify access list describing local addresses
  static  Specify static local->global mapping
Router(config)#ip nat inside source ?
  list    Specify access list describing local addresses
  static  Specify static local->global mapping
Router(config)#ip nat inside source static 192.168.10.10
% Incomplete command.
Router(config)#ip nat inside source static 192.168.10.10 10.0.0.1
Router(config)#
```

Router’s CLI

```
C:\>
C:\>
C:\>
C:\>
C:\>ping 10.0.0.2

Pinging 10.0.0.2 with 32 bytes of data:

Reply from 10.0.0.2: bytes=32 time<1ms TTL=254
Reply from 10.0.0.2: bytes=32 time<1ms TTL=254
Reply from 10.0.0.2: bytes=32 time<1ms TTL=254
Reply from 10.0.0.2: bytes=32 time<1ms TTL=254

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

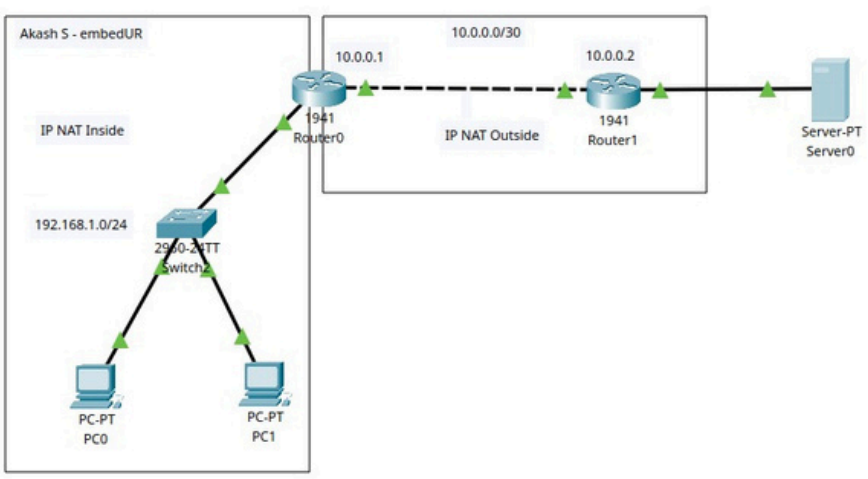
Top

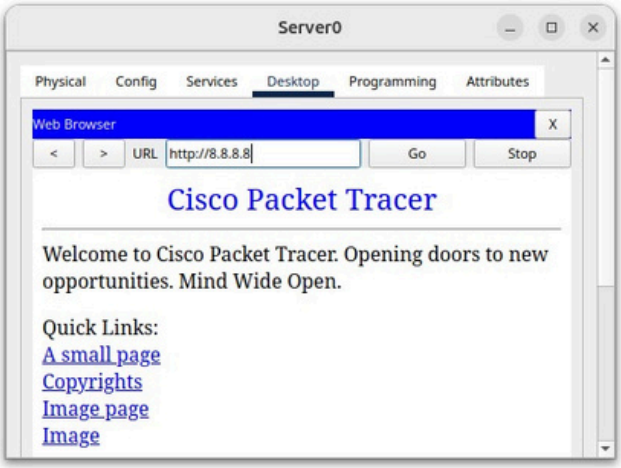
```
Router#
Router#
Router#
Router#
Router#
Router#
NAT: s=192.168.10.10->10.0.0.1, d=10.0.0.2 [30]

NAT*: s=10.0.0.2, d=10.0.0.1->192.168.10.10 [13]

NAT: s=192.168.10.10->10.0.0.1, d=10.0.0.2 [31]

NAT*: s=10.0.0.2, d=10.0.0.1->192.168.10.10 [14]

NAT: s=192.168.10.10->10.0.0.1, d=10.0.0.2 [32]

NAT*: s=10.0.0.2, d=10.0.0.1->192.168.10.10 [15]

NAT: s=192.168.10.10->10.0.0.1, d=10.0.0.2 [33]

NAT*: s=10.0.0.2, d=10.0.0.1->192.168.10.10 [16]
```

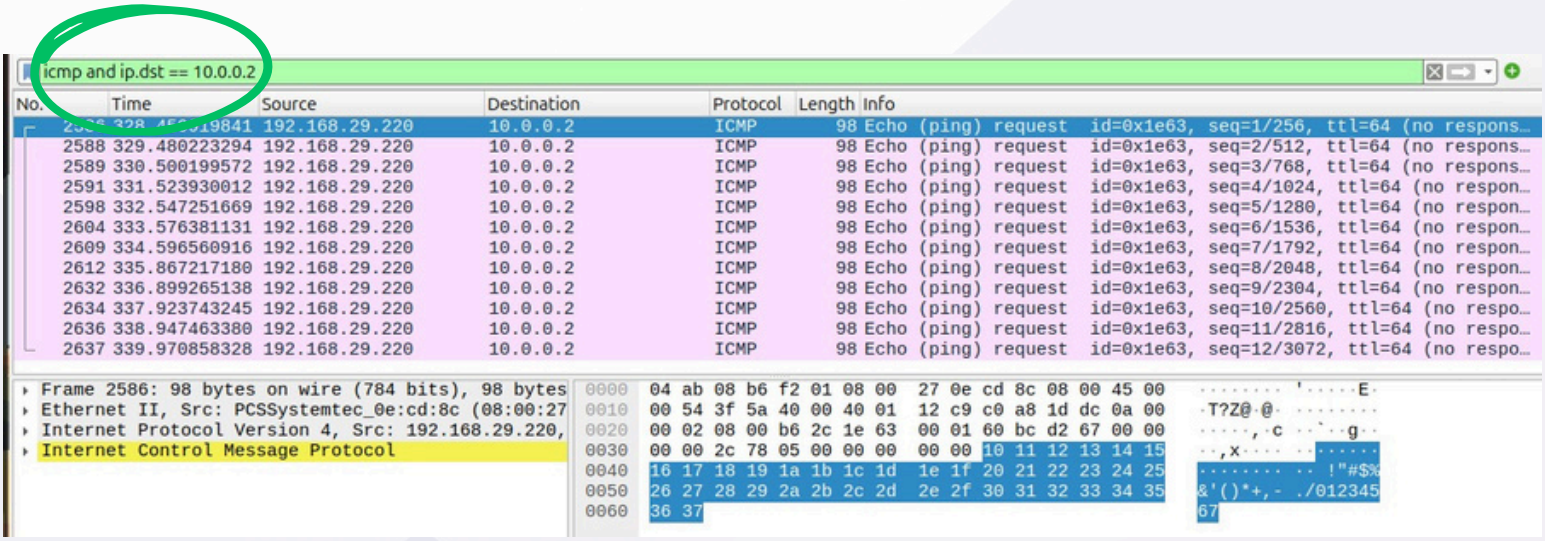
Results:

- I can now connect to the Web Server
- Google.com / 8.8.8.8

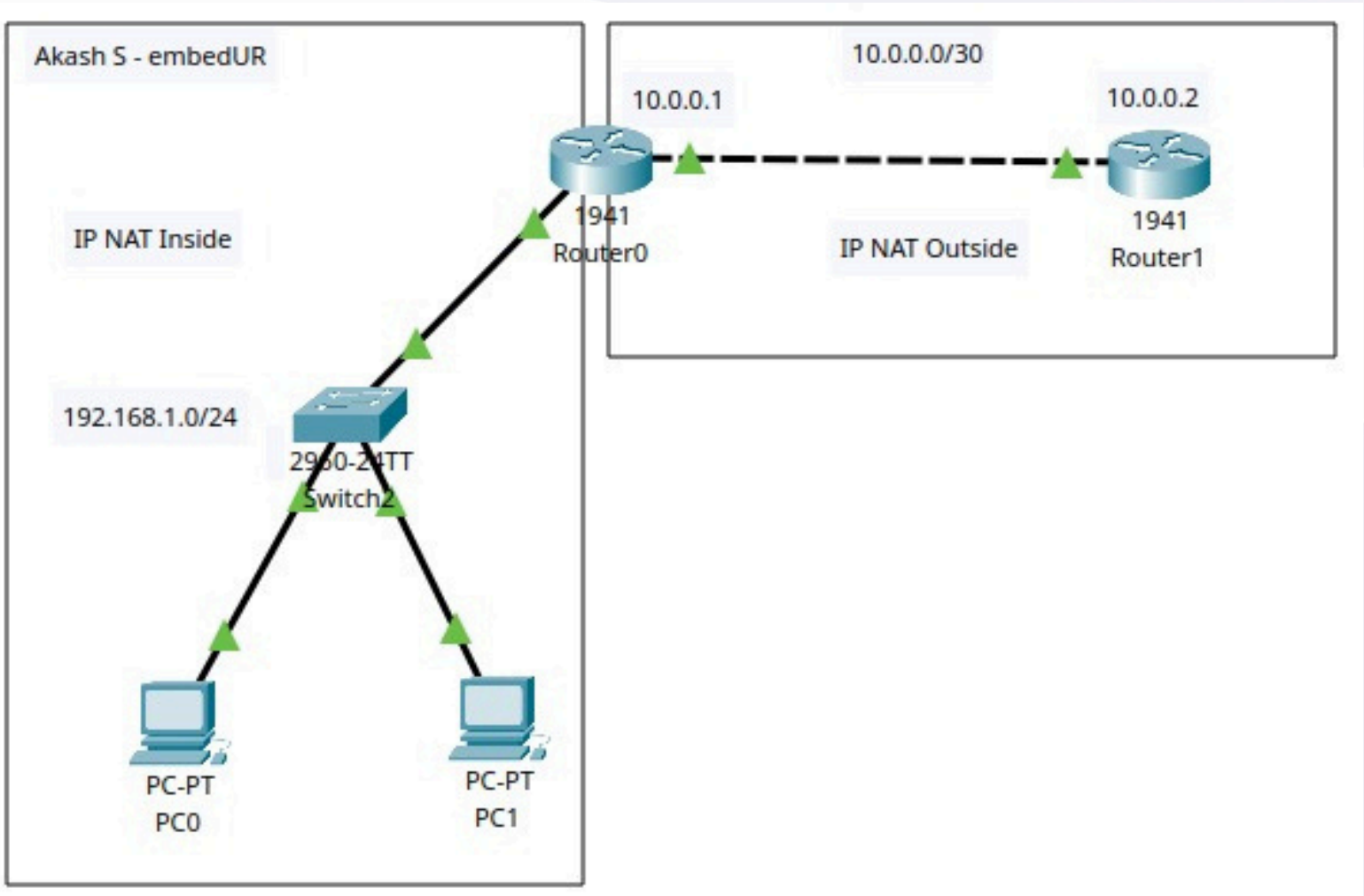
Fire	Last Status	Source	Destination	Type	Color	Time(sec)
	Successful	PC0	Router1	ICMP		0.000
	Successful	PC1	Router1	ICMP		0.000
	Successful	PC0	Router0	ICMP		0.000
	Successful	PC1	Router0	ICMP		0.000







Network Topology:



NAT inside and NAT outside

Qn 15:

Download iperf in laptop/phone and make sure they are in same network. Try different iperf commands with tcp, udp, birectional, reverse, multicast, parallel options and analyze the bandwidth and rate of transmission, delay, jitter etc

iperf:

- Iperf is a network performance testing tool that measures bandwidth, latency, jitter, and packet loss for TCP and UDP connections. It helps analyze network speed and performance under different conditions.

How iperf works?

- iPerf works in a client-server model. One device runs as a server (listening for connections), while another acts as a client (sending data).
- It supports TCP and UDP testing, bidirectional traffic, reverse mode, and parallel streams to simulate real-world network conditions.
- In my case, I tested everything within VirtualBox using localhost (127.0.0.1) instead of two separate devices.

Commands I used:

- Starting iperf3 server:
  - **iperf3 -s**
- Running a TCP client mode:
  - **iperf3 -c 127.0.0.1**
- Running a UDP server:
  - **iperf3 -c 127.0.0.1 -u -b 10M**

- Bidirectional Test:
  - **iperf3 -c 127.0.0.1 --bidir**
- Reverse Test:
  - **iperf3 -c 127.0.0.1 -R**
- Parallel Streams Test:
  - **iperf3 -c 127.0.0.1 -P 5**

Results:

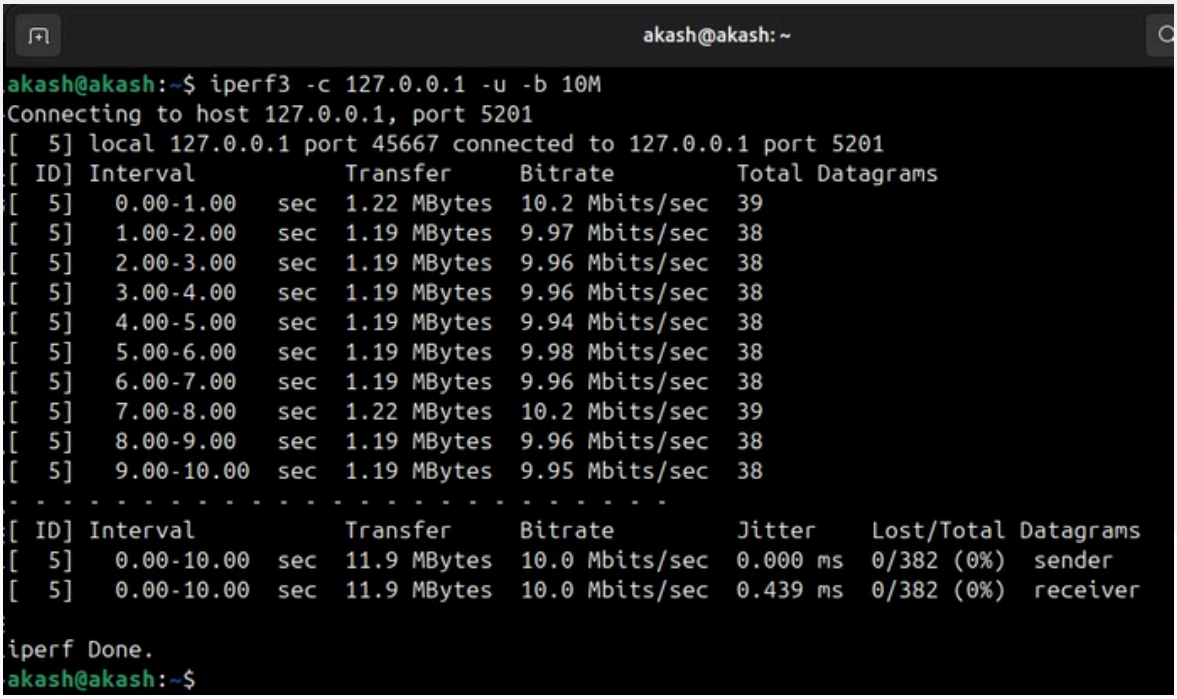
TCP Client

```
akash@akash: ~
akash@akash:~$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201 (test #1)
-----------------------------------------------------------
Accepted connection from 127.0.0.1, port 45098
[  5] local 127.0.0.1 port 5201 connected to 127.0.0.1 port 45108
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  4.55 GBytes  39.0 Gbits/sec
[  5]   1.00-2.00   sec  4.63 GBytes  39.7 Gbits/sec
[  5]   2.00-3.00   sec  4.94 GBytes  42.4 Gbits/sec
[  5]   3.00-4.00   sec  4.09 GBytes  35.1 Gbits/sec
[  5]   4.00-5.00   sec  4.24 GBytes  36.4 Gbits/sec
[  5]   5.00-6.00   sec  3.70 GBytes  31.8 Gbits/sec
[  5]   6.00-7.00   sec  2.74 GBytes  23.5 Gbits/sec
[  5]   7.00-8.02   sec  1.65 GBytes  14.0 Gbits/sec
[  5]   8.02-9.00   sec  1.59 GBytes  13.8 Gbits/sec
[  5]   9.00-10.00  sec  2.64 GBytes  22.7 Gbits/sec
[  5]  10.00-10.00  sec  8.38 MBytes  25.3 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.00  sec  34.8 GBytes  29.9 Gbits/sec
-----------------------------------------------------------
```

Terminal

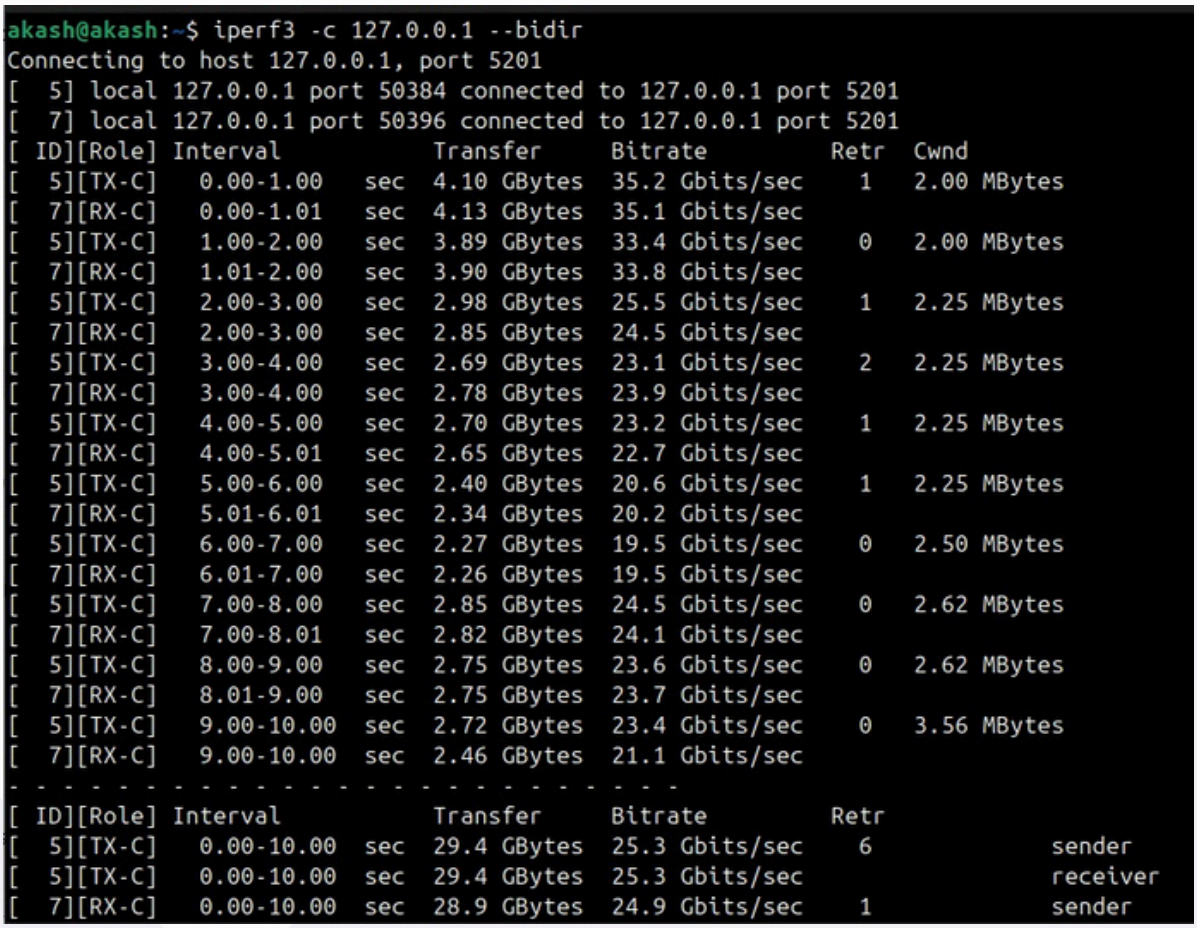
```
akash@akash: ~
akash@akash:~$ iperf3 -c 127.0.0.1
Connecting to host 127.0.0.1, port 5201
[  5] local 127.0.0.1 port 58942 connected to 127.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.01   sec  3.11 GBytes  26.6 Gbits/sec    0   2.50 MBytes
[  5]   1.01-2.00   sec  2.43 GBytes  20.9 Gbits/sec    1   2.75 MBytes
[  5]   2.00-3.00   sec  2.54 GBytes  21.8 Gbits/sec    0   2.75 MBytes
[  5]   3.00-4.01   sec  2.89 GBytes  24.7 Gbits/sec    0   2.75 MBytes
[  5]   4.01-5.00   sec  3.19 GBytes  27.6 Gbits/sec    0   2.75 MBytes
[  5]   5.00-6.01   sec  2.40 GBytes  20.5 Gbits/sec    0   2.75 MBytes
[  5]   6.01-7.00   sec  2.45 GBytes  21.2 Gbits/sec    0   2.87 MBytes
[  5]   7.00-8.00   sec  1.66 GBytes  14.3 Gbits/sec    1   3.18 MBytes
[  5]   8.00-9.00   sec  1.58 GBytes  13.6 Gbits/sec    0   3.43 MBytes
[  5]   9.00-10.00  sec  1.80 GBytes  15.5 Gbits/sec    0   3.81 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  24.2 GBytes  20.8 Gbits/sec    2             sender
[  5]   0.00-10.00  sec  24.2 GBytes  20.8 Gbits/sec                  receiver

iperf Done.
akash@akash:~$
```

UDP Server:



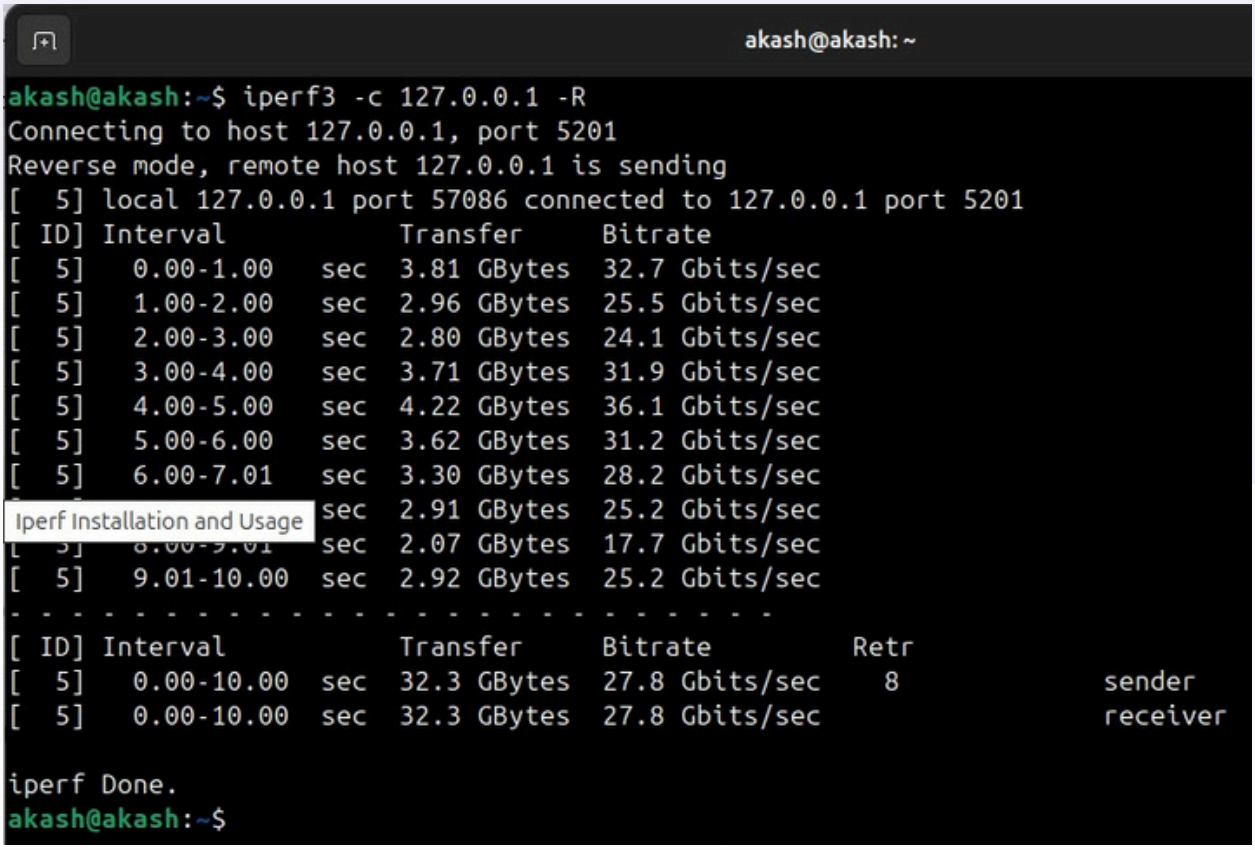
```
akash@akash: ~
akash@akash:~$ iperf3 -c 127.0.0.1 -u -b 10M
Connecting to host 127.0.0.1, port 5201
[  5] local 127.0.0.1 port 45667 connected to 127.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  1.22 MBytes  10.2 Mbits/sec  39
[  5]   1.00-2.00   sec  1.19 MBytes  9.97 Mbits/sec  38
[  5]   2.00-3.00   sec  1.19 MBytes  9.96 Mbits/sec  38
[  5]   3.00-4.00   sec  1.19 MBytes  9.96 Mbits/sec  38
[  5]   4.00-5.00   sec  1.19 MBytes  9.94 Mbits/sec  38
[  5]   5.00-6.00   sec  1.19 MBytes  9.98 Mbits/sec  38
[  5]   6.00-7.00   sec  1.19 MBytes  9.96 Mbits/sec  38
[  5]   7.00-8.00   sec  1.22 MBytes  10.2 Mbits/sec  39
[  5]   8.00-9.00   sec  1.19 MBytes  9.96 Mbits/sec  38
[  5]   9.00-10.00  sec  1.19 MBytes  9.95 Mbits/sec  38
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.00  sec  11.9 MBytes  10.0 Mbits/sec  0.000 ms  0/382 (0%)  sender
[  5]   0.00-10.00  sec  11.9 MBytes  10.0 Mbits/sec  0.439 ms  0/382 (0%)  receiver

iperf Done.
akash@akash:~$
```

Bidirectional:

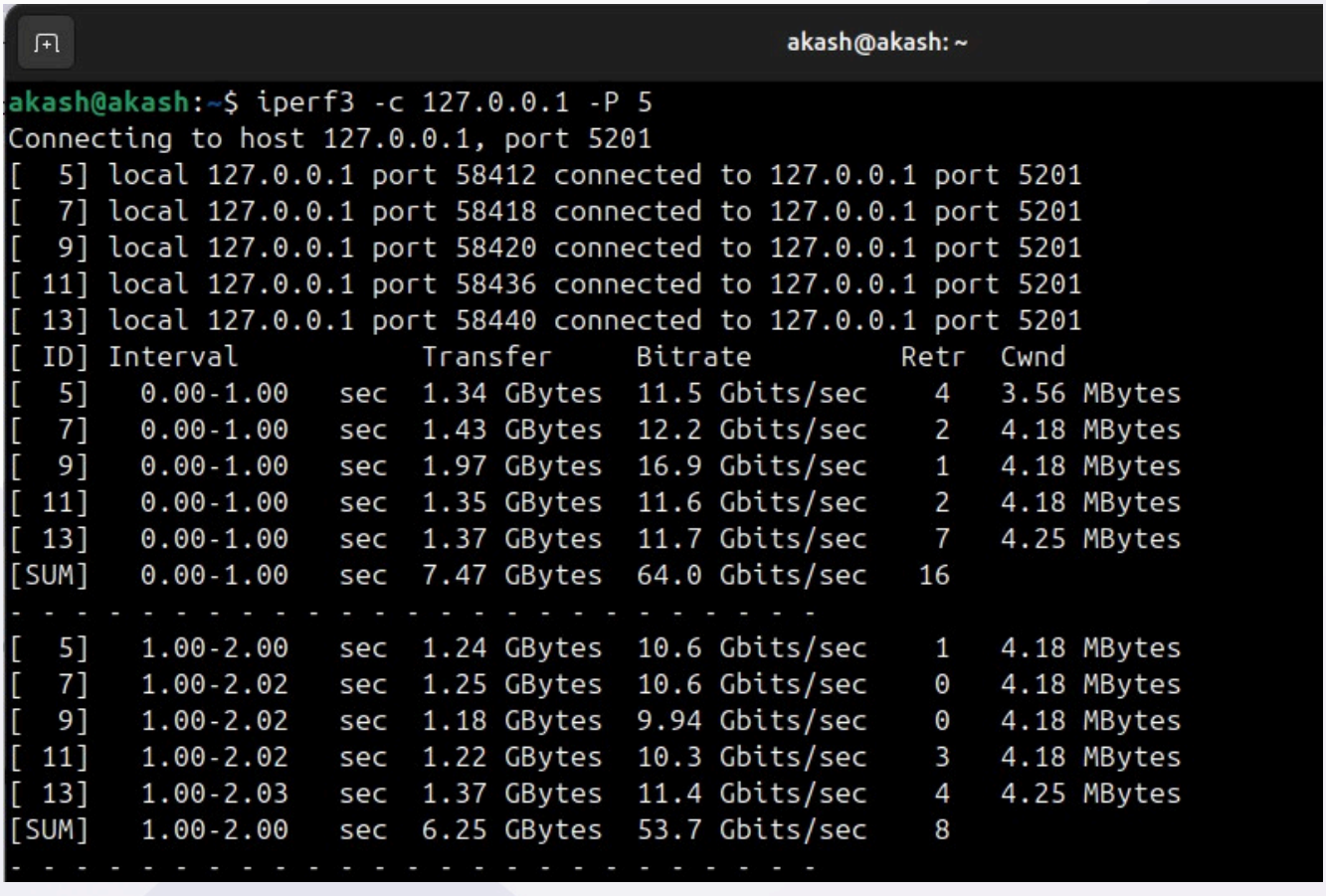


```
akash@akash:~$ iperf3 -c 127.0.0.1 --bidir
Connecting to host 127.0.0.1, port 5201
[  5] local 127.0.0.1 port 50384 connected to 127.0.0.1 port 5201
[  7] local 127.0.0.1 port 50396 connected to 127.0.0.1 port 5201
[ ID][Role] Interval           Transfer     Bitrate         Retr  Cwnd
[  5][TX-C]   0.00-1.00   sec  4.10 GBytes  35.2 Gbits/sec    1   2.00 MBytes
[  7][RX-C]   0.00-1.01   sec  4.13 GBytes  35.1 Gbits/sec
[  5][TX-C]   1.00-2.00   sec  3.89 GBytes  33.4 Gbits/sec    0   2.00 MBytes
[  7][RX-C]   1.01-2.00   sec  3.90 GBytes  33.8 Gbits/sec
[  5][TX-C]   2.00-3.00   sec  2.98 GBytes  25.5 Gbits/sec    1   2.25 MBytes
[  7][RX-C]   2.00-3.00   sec  2.85 GBytes  24.5 Gbits/sec
[  5][TX-C]   3.00-4.00   sec  2.69 GBytes  23.1 Gbits/sec    2   2.25 MBytes
[  7][RX-C]   3.00-4.00   sec  2.78 GBytes  23.9 Gbits/sec
[  5][TX-C]   4.00-5.00   sec  2.70 GBytes  23.2 Gbits/sec    1   2.25 MBytes
[  7][RX-C]   4.00-5.01   sec  2.65 GBytes  22.7 Gbits/sec
[  5][TX-C]   5.00-6.00   sec  2.40 GBytes  20.6 Gbits/sec    1   2.25 MBytes
[  7][RX-C]   5.01-6.01   sec  2.34 GBytes  20.2 Gbits/sec
[  5][TX-C]   6.00-7.00   sec  2.27 GBytes  19.5 Gbits/sec    0   2.50 MBytes
[  7][RX-C]   6.01-7.00   sec  2.26 GBytes  19.5 Gbits/sec
[  5][TX-C]   7.00-8.00   sec  2.85 GBytes  24.5 Gbits/sec    0   2.62 MBytes
[  7][RX-C]   7.00-8.01   sec  2.82 GBytes  24.1 Gbits/sec
[  5][TX-C]   8.00-9.00   sec  2.75 GBytes  23.6 Gbits/sec    0   2.62 MBytes
[  7][RX-C]   8.01-9.00   sec  2.75 GBytes  23.7 Gbits/sec
[  5][TX-C]   9.00-10.00  sec  2.72 GBytes  23.4 Gbits/sec    0   3.56 MBytes
[  7][RX-C]   9.00-10.00  sec  2.46 GBytes  21.1 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID][Role] Interval           Transfer     Bitrate         Retr
[  5][TX-C]   0.00-10.00  sec  29.4 GBytes  25.3 Gbits/sec    6             sender
[  5][TX-C]   0.00-10.00  sec  29.4 GBytes  25.3 Gbits/sec                  receiver
[  7][RX-C]   0.00-10.00  sec  28.9 GBytes  24.9 Gbits/sec    1             sender
```

Reverse Test:



Parallel Stream Test:



THE END

AKASH S