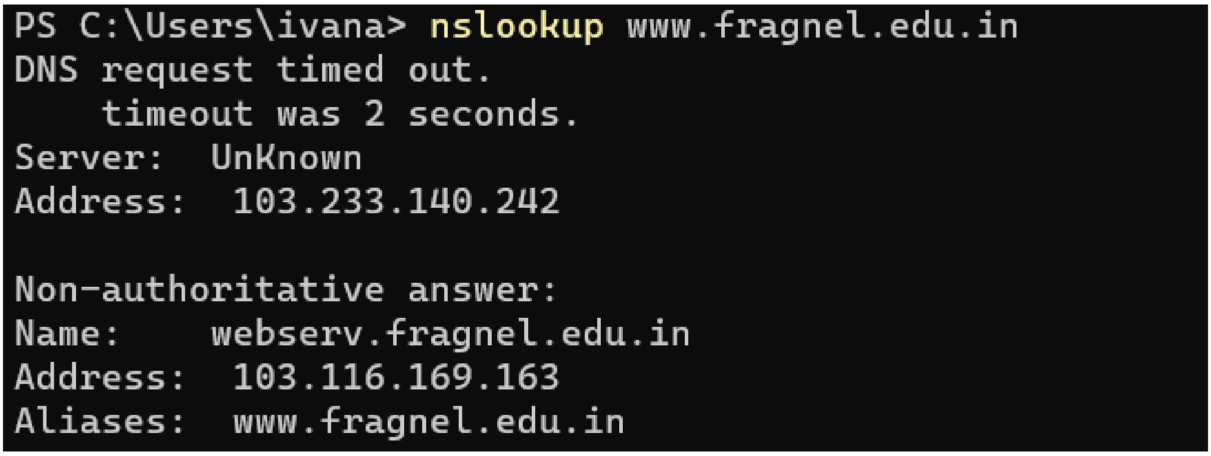
2. Use basic networking commands in Linux (ping, tracert, nslookup, netstat, ARP, RARP,ip, ifconfig , dig , route)

ifconfig

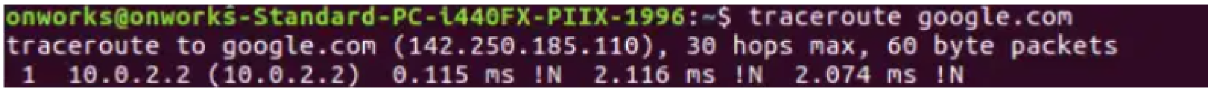
nslookup



```
PS C:\Users\ivana> nslookup www.fragnel.edu.in
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  103.233.140.242

Non-authoritative answer:
Name:    webserv.fragnel.edu.in
Address:  103.116.169.163
Aliases:  www.fragnel.edu.in
```

ping google.com
ping -c 132.412.412.124

traceroute google.com



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ traceroute google.com
traceroute to google.com (142.250.185.110), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.115 ms !N  2.116 ms !N  2.074 ms !N
```

netstat
netstat -a

arp -a
arp -- help

rarp -- help

ip address show
ip route show

dig google.com

route -n

Certainly! Here are some basic networking commands you can use in Linux:

1. **Ping**: The \`ping\` command is used to test network connectivity to a remote host by sending ICMP echo requests and waiting for ICMP echo replies.

Example:
```
ping google.com
```

2. **Traceroute (traceroute or tracepath)**: Traceroute is used to trace the route that packets take from your computer to a destination host, showing the intermediate hops.

Example:
```
traceroute google.com
```

3. **nslookup**: The `nslookup` command is used for querying DNS (Domain Name System) to obtain the IP address or DNS information of a domain.

Example:
```
nslookup google.com
```

4. **netstat**: The `netstat` command displays network statistics, active connections, routing tables, and interface information. However, it is deprecated, and it's recommended to use the `ss` command instead.

Example:
```
netstat -tuln
```

5. **ARP (Address Resolution Protocol)**: ARP is used to view and manipulate the ARP cache, which maps IP addresses to MAC addresses.

Example:
```
arp -a
```

6. **RARP (Reverse ARP)**: RARP is not commonly used anymore. It was used to resolve a MAC address to an IP address. Linux doesn't provide a built-in RARP command.

7. **ip**: The `ip` command is a powerful tool for configuring network interfaces, routes, tunnels, and more. It replaces the ifconfig and route commands.

Example:
```
ip address show
ip route show
```

~~~
```
~~~

8. **ifconfig**: Although deprecated, you can still use `ifconfig` for basic network interface configuration and information.

Example:
```
ifconfig -a
```

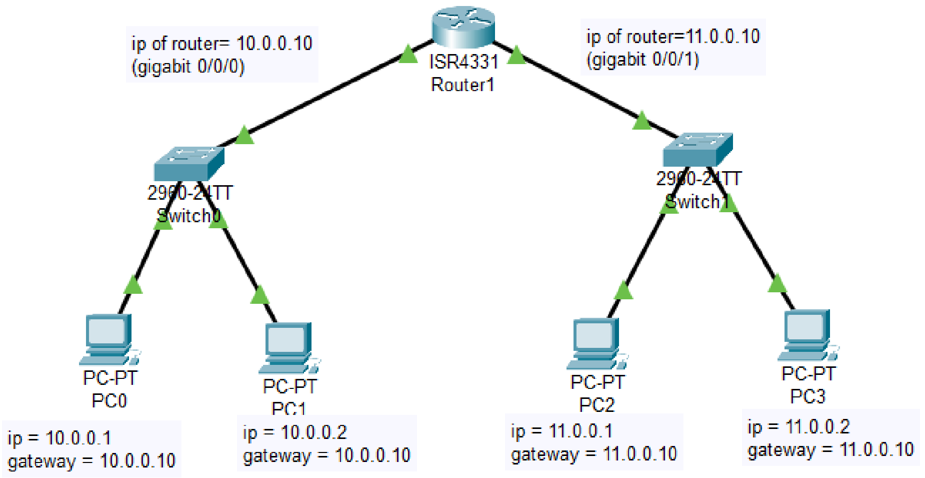
9. **dig**: The `dig` command is a versatile DNS query tool that provides detailed information about DNS records for a domain.

Example:
```
dig google.com
```

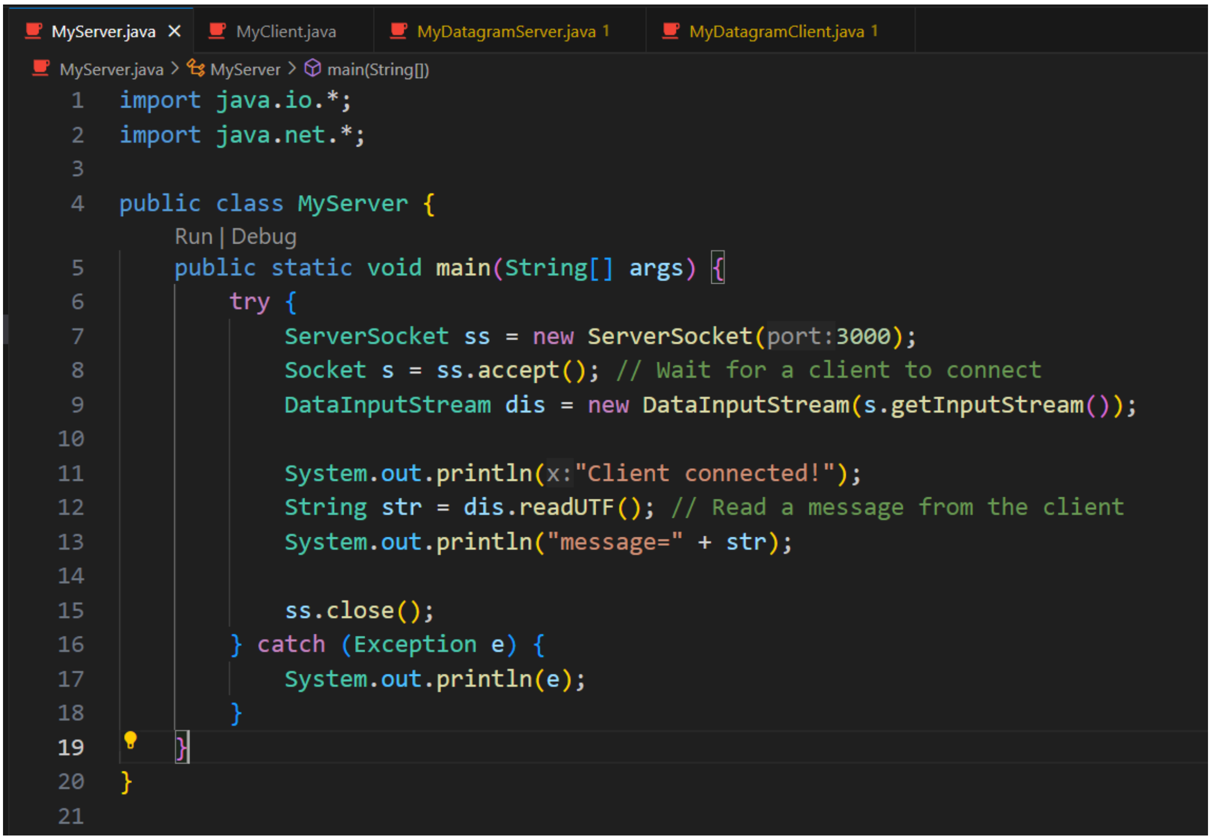
10. **route**: The `route` command is used to display or modify the kernel's IP routing table.

Example:
```
route -n
```

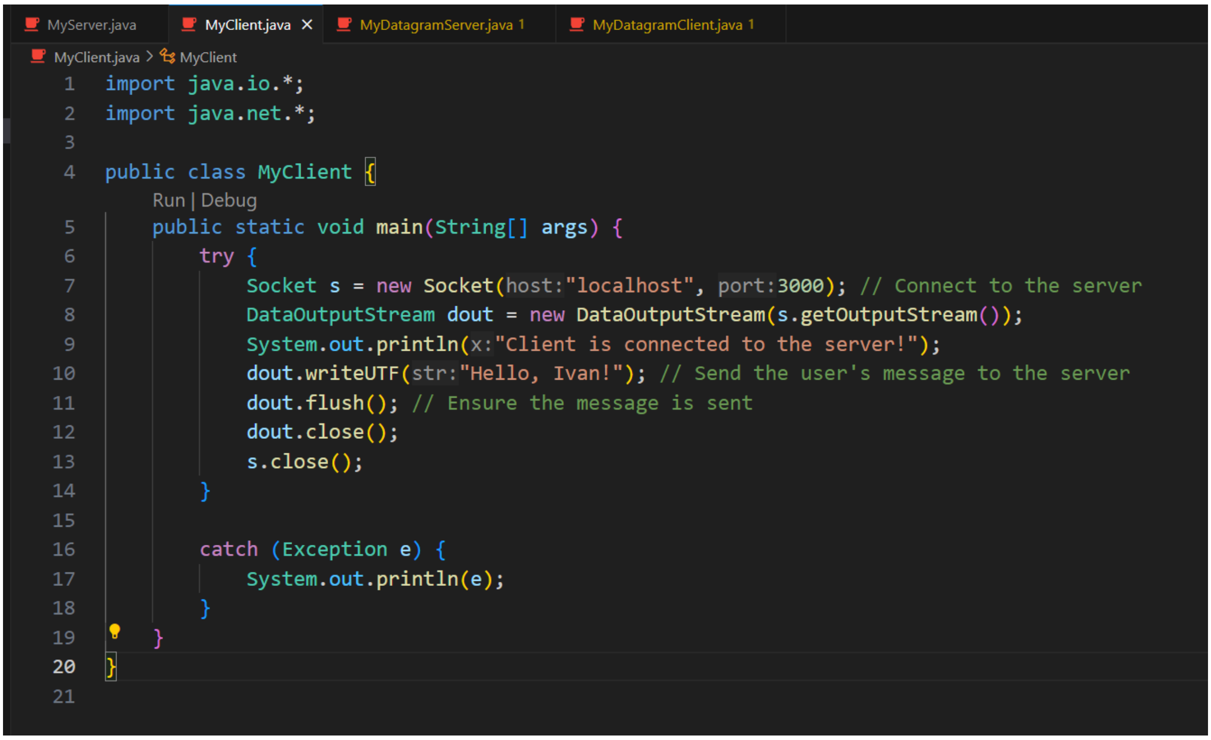
3. Build a simple network topology and configure it for static routing protocol using packet tracer. Setup a network and configure IP addressing , subnetting , masking



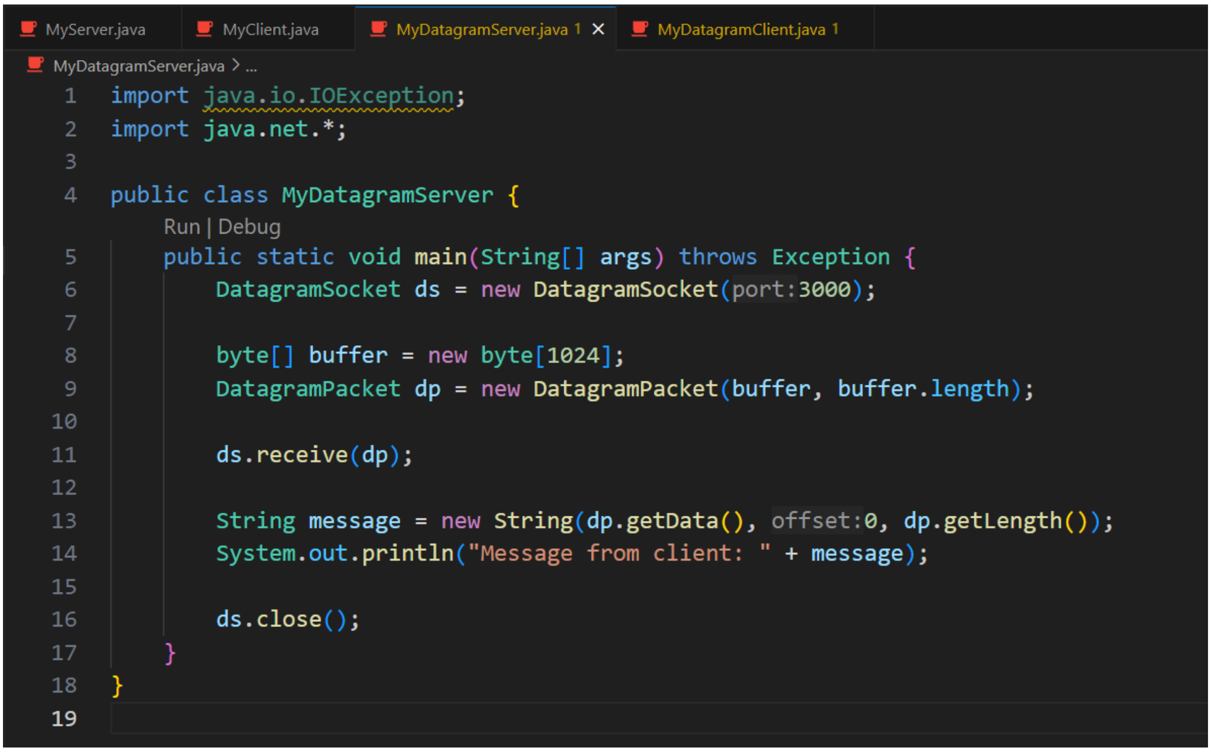
Q10) Socket Programming using TCP or UDP



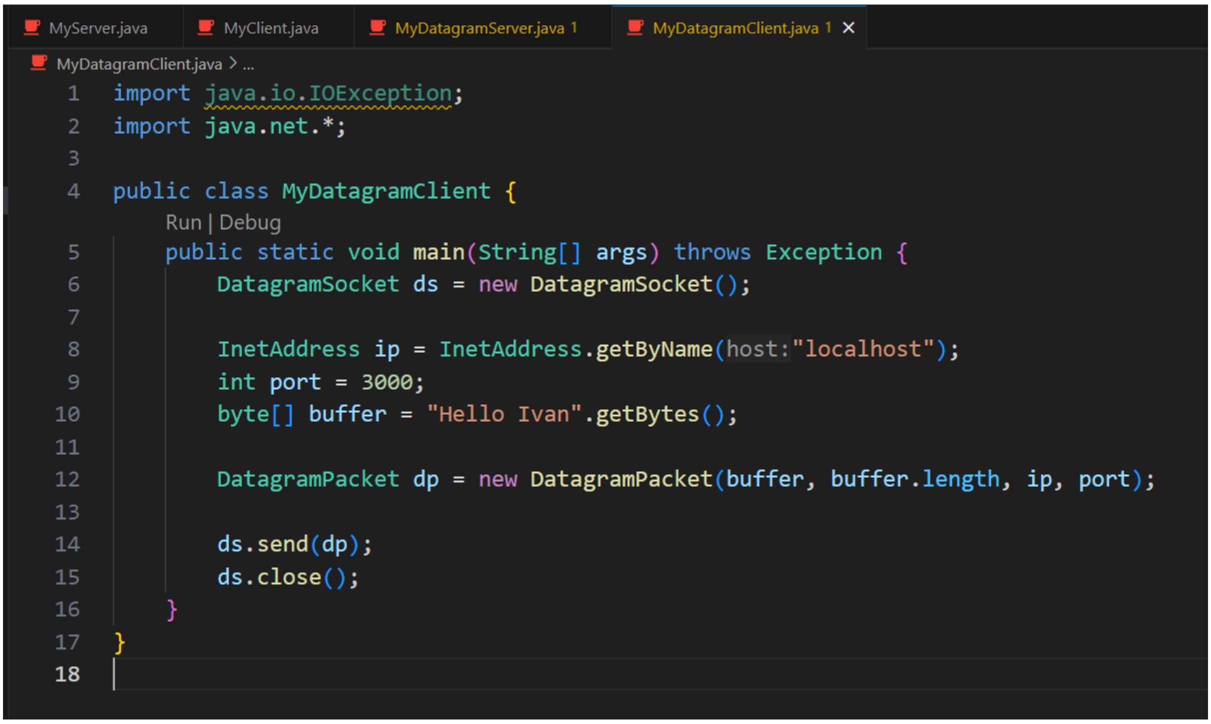
```
import java.io.*;
import java.net.*;

public class MyServer {
    Run | Debug
    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket(port:3000);
            Socket s = ss.accept(); // Wait for a client to connect
            DataInputStream dis = new DataInputStream(s.getInputStream());

            System.out.println(x:"Client connected!");
            String str = dis.readUTF(); // Read a message from the client
            System.out.println("message=" + str);

            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```



```
import java.io.*;
import java.net.*;

public class MyClient {
    Run | Debug
    public static void main(String[] args) {
        try {
            Socket s = new Socket(host:"localhost", port:3000); // Connect to the server
            DataOutputStream dout = new DataOutputStream(s.getOutputStream());
            System.out.println(x:"Client is connected to the server!");
            dout.writeUTF(str:"Hello, Ivan!"); // Send the user's message to the server
            dout.flush(); // Ensure the message is sent
            dout.close();
            s.close();
        }

        catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

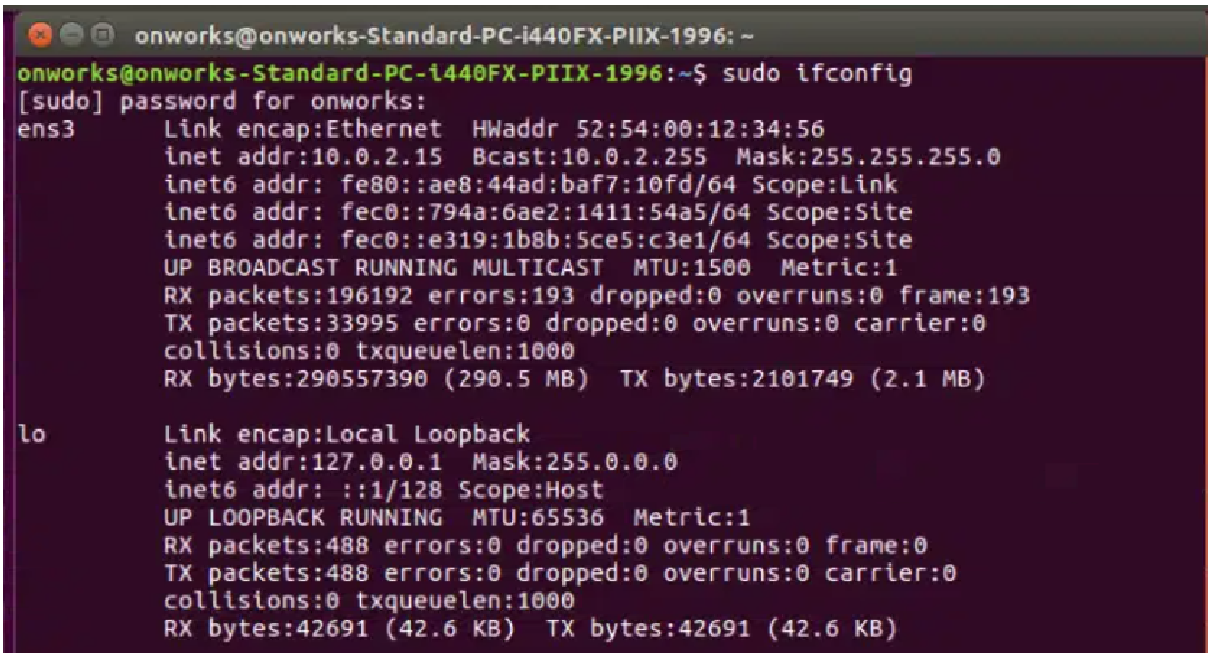


```java
import java.io.IOException;
import java.net.*;

public class MyDatagramServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(port:3000);

        byte[] buffer = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buffer, buffer.length);

        ds.receive(dp);

        String message = new String(dp.getData(), offset:0, dp.getLength());
        System.out.println("Message from client: " + message);

        ds.close();
    }
}

```



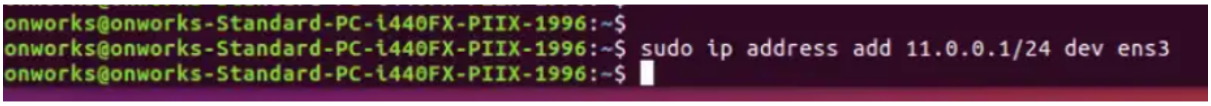
```java
import java.io.IOException;
import java.net.*;

public class MyDatagramClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();

        InetAddress ip = InetAddress.getByName(host:"localhost");
        int port = 3000;
        byte[] buffer = "Hello Ivan".getBytes();

        DatagramPacket dp = new DatagramPacket(buffer, buffer.length, ip, port);

        ds.send(dp);
        ds.close();
    }
}

```

Q8)



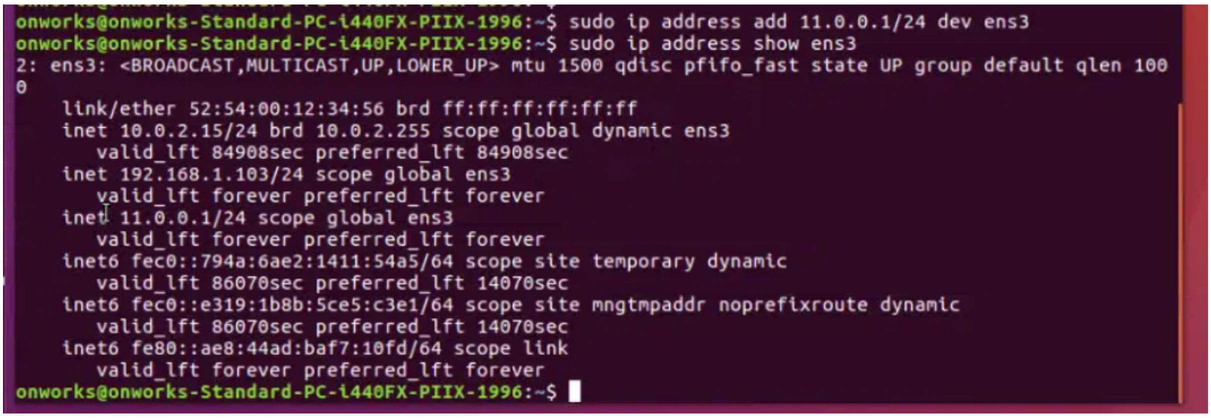
```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo ifconfig
[sudo] password for onworks:
ens3      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::ae8:44ad:baf7:10fd/64 Scope:Link
          inet6 addr: fec0::794a:6ae2:1411:54a5/64 Scope:Site
          inet6 addr: fec0::e319:1b8b:5ce5:c3e1/64 Scope:Site
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:196192 errors:193 dropped:0 overruns:0 frame:193
          TX packets:33995 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:290557390 (290.5 MB)  TX bytes:2101749 (2.1 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:488 errors:0 dropped:0 overruns:0 frame:0
          TX packets:488 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42691 (42.6 KB)  TX bytes:42691 (42.6 KB)
```

see device name = ens3
inet addr = 10.0.2.15



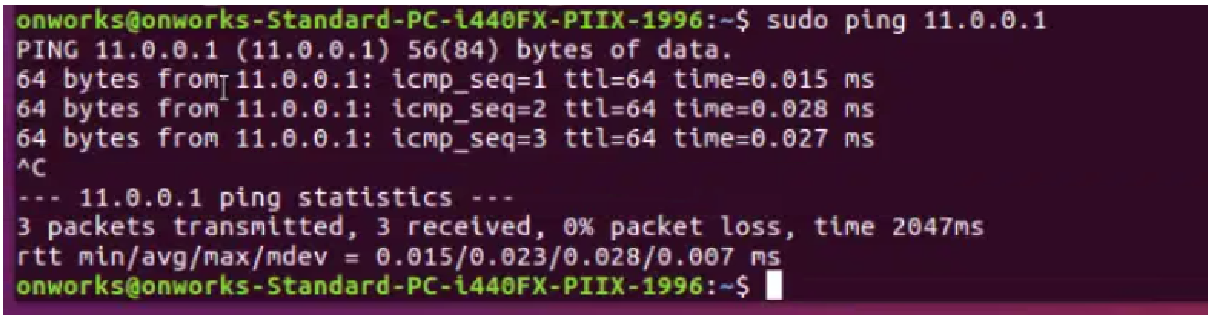
```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo ip address add 11.0.0.1/24 dev ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

sudo ip address add 11.0.0.1/24 dev ens3



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo ip address add 11.0.0.1/24 dev ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo ip address show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
0
    link/ether 52:54:00:12:34:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic ens3
       valid_lft 84908sec preferred_lft 84908sec
    inet 192.168.1.103/24 scope global ens3
       valid_lft forever preferred_lft forever
    inet 11.0.0.1/24 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fec0::794a:6ae2:1411:54a5/64 scope site temporary dynamic
       valid_lft 86070sec preferred_lft 14070sec
    inet6 fec0::e319:1b8b:5ce5:c3e1/64 scope site mngtmpaddr noprefixroute dynamic
       valid_lft 86070sec preferred_lft 14070sec
    inet6 fe80::ae8:44ad:baf7:10fd/64 scope link
       valid_lft forever preferred_lft forever
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

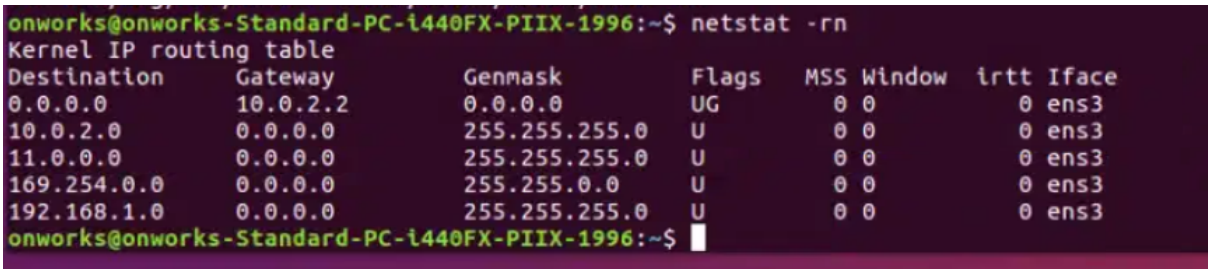
sudo ip address show ens3
(you can see the ip address added by you: inet 11.0.0.1/24 in the third line)



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo ping 11.0.0.1
PING 11.0.0.1 (11.0.0.1) 56(84) bytes of data.
64 bytes from 11.0.0.1: icmp_seq=1 ttl=64 time=0.015 ms
64 bytes from 11.0.0.1: icmp_seq=2 ttl=64 time=0.028 ms
64 bytes from 11.0.0.1: icmp_seq=3 ttl=64 time=0.027 ms
^C
--- 11.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.015/0.023/0.028/0.007 ms
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

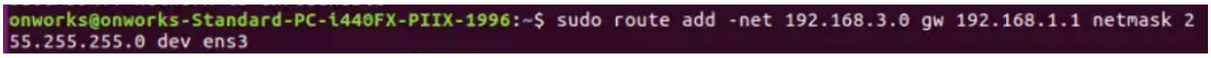
sudo ping 11.0.0.1
(ping the ip address added)

b. Using netstat and route commands in Linux: To view the current routing table, add and delete routes, and change the default gateway, you can use the following commands:



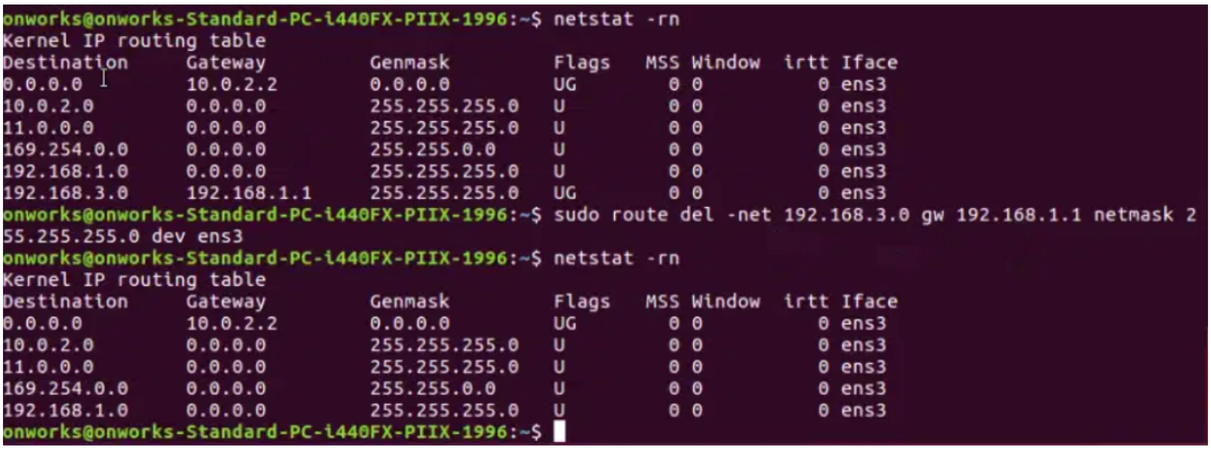
```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 ens3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
11.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 ens3
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

You can see the routing table with all the ip addresses



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo route add -net 192.168.3.0 gw 192.168.1.1 netmask 2
55.255.255.0 dev ens3
```

sudo route add -net 192.168.3.0 gw 192.168.1.1 netmask 255.255.255.0 dev ens3



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 ens3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
11.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 ens3
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 ens3
192.168.3.0     192.168.1.1     255.255.255.0   UG        0 0          0 ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo route del -net 192.168.3.0 gw 192.168.1.1 netmask 2
55.255.255.0 dev ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 ens3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
11.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 ens3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 ens3
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 ens3
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

sudo route del -net 192.168.3.0 gw 192.168.1.1 netmask 255.255.255.0 dev ens3

sudo route add default gw 192.168.1.1
sudo route del default gw 192.168.1.1

c. Perform packet filtering using IPtables in Linux:
To perform packet filtering using IPtables, you can follow these steps:

Enable IP forwarding in your Linux system. Open the sysctl.conf file:

```
sudo nano /etc/sysctl.conf
```

```
net.ipv4.ip_forward = 1
```

Save the file and apply the changes:

```
sudo sysctl -p
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
sudo iptables -A INPUT -j DROP
```

Use Wire shark to understand the operation of TCP/IP layers: • Ethernet Layer: Frame header, Frame size etc. Data Link Layer: MAC address, ARP (IP and MAC address binding) Network Layer: IP Packet (header, fragmentation), ICMP (Query and Echo) Transport Layer: TCP Ports, TCP handshake segments etc. Application Layer: DHCP, FTP, HTTP header formats how to do the above in wireshark, give a step by step tutorial

ChatGPT

Certainly! Here's a step-by-step tutorial on how to use Wireshark to understand the operation of TCP/IP layers for Ethernet, Data Link, Network, Transport, and Application layers:

Step 1: Install and Launch Wireshark

If you haven't already, download and install Wireshark from the official website (https://www.wireshark.org/). Launch Wireshark with administrator privileges to capture network traffic.

Step 2: Select Network Interface

- After launching Wireshark, you'll be presented with a list of available network interfaces. Select the one you want to capture traffic on and click "Start" to begin capturing.

Step 3: Ethernet Layer (Frame Header and Frame Size)

- In the Wireshark main window, you will see a live capture of network traffic. Each row represents a captured packet.
- Click on a packet in the list to view details about the Ethernet Layer.

- Expand the "Ethernet II" section to see the Frame header, including source and destination MAC addresses.
- You can find the Frame size information in the "Frame" section.

Step 4: Data Link Layer (MAC Address and ARP)

- Continue analyzing the same packet at the Data Link Layer.
- The MAC addresses are directly visible in the "Ethernet II" section.
- To view ARP traffic (IP and MAC address binding), apply an ARP display filter. Type in the display filter bar. This will show ARP requests and replies in the packet list.

Step 5: Network Layer (IP Packet and ICMP)

- Keep analyzing the same packet for the Network Layer.
- To view IP packets, apply an IP display filter by typing in the display filter bar. This will show IP packets.
- Click on an IP packet to inspect the IP header, which includes the source and destination IP addresses.
- If IP fragmentation occurs, Wireshark will indicate fragmentation in the IP header.
- To view ICMP packets (Query and Echo), apply an ICMP display filter by typing . You can then click on ICMP packets to inspect the details.

Step 6: Transport Layer (TCP Ports and Handshake)

- For TCP traffic analysis, apply a TCP display filter by typing in the display filter bar. This will display TCP packets.
- Click on a TCP packet to view the TCP header, which includes source and destination ports, sequence numbers, acknowledgment numbers, and flags.
- To analyze the TCP handshake, look for the initial SYN, SYN-ACK, and ACK segments within the TCP packets.

Step 7: Application Layer (DHCP, FTP, HTTP Header Formats)

- Wireshark can decode various application layer protocols.
- For DHCP, apply a DHCP display filter by typing to capture DHCP packets. You can then analyze DHCP packet details, including IP address leases.
- For FTP, apply an FTP display filter by typing . This will capture FTP traffic, and you can see FTP commands and responses.
- For HTTP, apply an HTTP display filter by typing . This will allow you to capture and analyze HTTP header formats, requests, and responses.

By following these steps, you can use Wireshark to explore and understand the operation of TCP/IP layers in a captured network traffic trace.