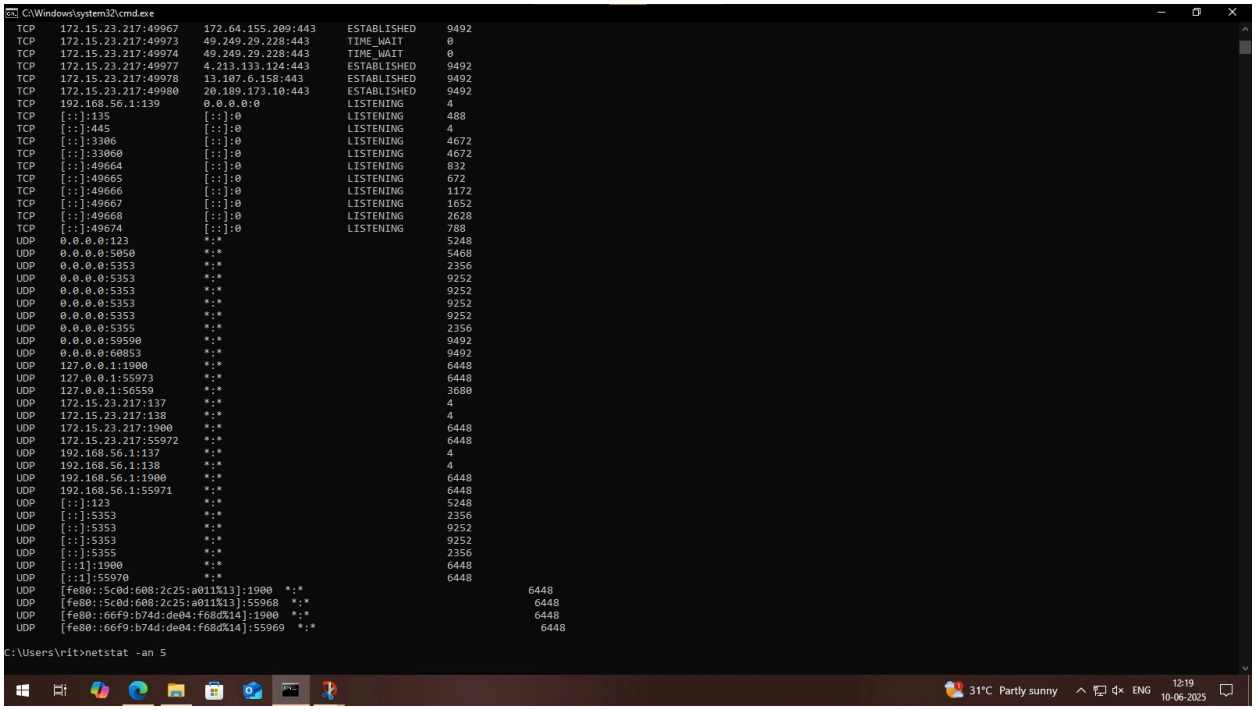
1. Tcpdump:

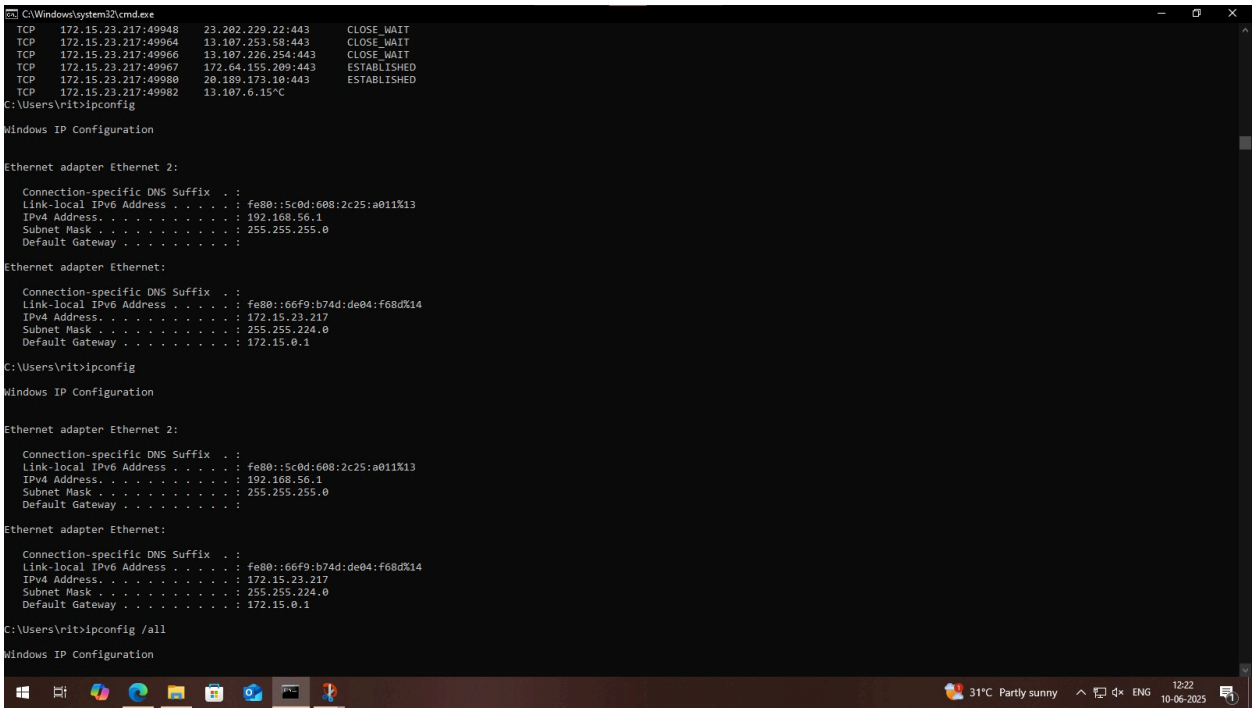
```
susel:~ # tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
20:39:28.014065 IP 192.168.198.1.netbios-ns > 192.168.198.255.netbios-ns: NBT U
DP PACKET(137): QUERY; REQUEST; BROADCAST
20:39:28.014840 IP 192.168.198.128.56851 > 192.168.198.2.domain: 18867+ PTR? 25
5.198.168.192.in-addr.arpa. (46)
20:39:28.027418 IP 192.168.198.1.49733 > 224.0.0.252.llmnr: UDP, length 22
20:39:28.027850 IP 192.168.198.128.50611 > lhr14s24-in-f19.1e100.net.https: P 2
912329209:2912329246(37) ack 1375935787 win 18760
20:39:28.034322 IP lhr14s24-in-f19.1e100.net.https > 192.168.198.128.50611: . a
ck 37 win 64240
20:39:28.037196 IP6 fe80::2cfe:5154:6c0d:fafd.65460 > ff02::1:3.llmnr: UDP, len
gth 22
20:39:28.039057 IP 192.168.198.1.65460 > 224.0.0.252.llmnr: UDP, length 22
20:39:28.051576 IP 192.168.198.2.domain > 192.168.198.128.56851: 18867 NXDomain
 0/1/0 (95)
20:39:28.051744 IP 192.168.198.128.35496 > 192.168.198.2.domain: 58919+ PTR? 1.
198.168.192.in-addr.arpa. (44)
20:39:28.077704 IP 192.168.198.2.domain > 192.168.198.128.35496: 58919 NXDomain
 0/1/0 (93)
20:39:28.077903 IP 192.168.198.128.56215 > 192.168.198.2.domain: 59223+ PTR? 2.
198.168.192.in-addr.arpa. (44)
20:39:28.103262 IP 192.168.198.2.domain > 192.168.198.128.56215: 59223 NXDomain
 0/1/0 (93)
```

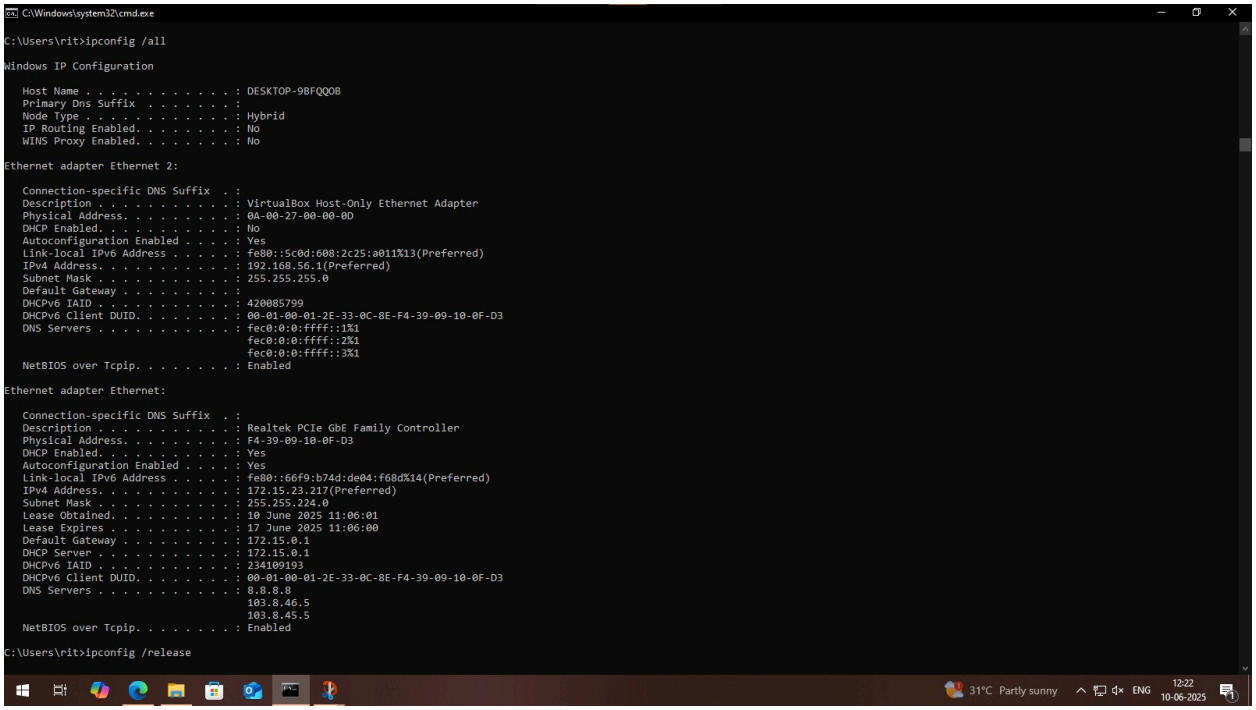
2. Netstat:

```
C:\Windows\system32\cmd.exe
C:\Users\rit>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:49669        DESKTOP-9BFQQOB:49670  ESTABLISHED
  TCP    127.0.0.1:49670        DESKTOP-9BFQQOB:49669  ESTABLISHED
  TCP    127.0.0.1:49671        DESKTOP-9BFQQOB:49672  ESTABLISHED
  TCP    127.0.0.1:49672        DESKTOP-9BFQQOB:49671  ESTABLISHED
  TCP    172.15.23.217:49675    4.213.25.240:https     ESTABLISHED
  TCP    172.15.23.217:49832    whatsapp-cdn-shv-03-maa3:https  ESTABLISHED
  TCP    172.15.23.217:49852    172.64.155.209:https   ESTABLISHED
  TCP    172.15.23.217:49869    1:https                ESTABLISHED
  TCP    172.15.23.217:49896    0:https                TIME_WAIT
  TCP    172.15.23.217:49914    1:https                ESTABLISHED
  TCP    172.15.23.217:49932    a23-38-59-250:http     ESTABLISHED
  TCP    172.15.23.217:49938    a23-202-229-5:https    CLOSE_WAIT
  TCP    172.15.23.217:49948    a23-202-229-22:https   CLOSE_WAIT
  TCP    172.15.23.217:49964    13.107.253.58:https    CLOSE_WAIT
  TCP    172.15.23.217:49966    13.107.226.254:https   CLOSE_WAIT
  TCP    172.15.23.217:49967    172.64.155.209:https   ESTABLISHED
  TCP    172.15.23.217:49973    static-228:https       TIME_WAIT
  TCP    172.15.23.217:49974    static-228:https       TIME_WAIT
  TCP    172.15.23.217:49976    150.171.28.11:https    ESTABLISHED
  TCP    172.15.23.217:49977    4.213.133.124:https    ESTABLISHED
  TCP    172.15.23.217:49978    bingforbusiness:https  ESTABLISHED
  TCP    172.15.23.217:49979    ec2-3-233-158-25:https  ESTABLISHED
  TCP    172.15.23.217:49980    20.189.173.10:https    SYN_SENT

C:\Users\rit>netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       488
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       4672
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING       5468
  TCP    0.0.0.0:33060          0.0.0.0:0              LISTENING       4672
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       832
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       672
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1172
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       1652
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING       2628
  TCP    0.0.0.0:49674          0.0.0.0:0              LISTENING       788
  TCP    127.0.0.1:49669        127.0.0.1:49670        ESTABLISHED     4672
  TCP    127.0.0.1:49670        127.0.0.1:49669        ESTABLISHED     4672
  TCP    127.0.0.1:49671        127.0.0.1:49672        ESTABLISHED     4672
  TCP    127.0.0.1:49672        127.0.0.1:49671        ESTABLISHED     4672
  TCP    172.15.23.217:139      0.0.0.0:0              LISTENING       4
  TCP    172.15.23.217:49675    4.213.25.240:443       ESTABLISHED     3132
  TCP    172.15.23.217:49832    57.144.213.32:443      ESTABLISHED     9492
```

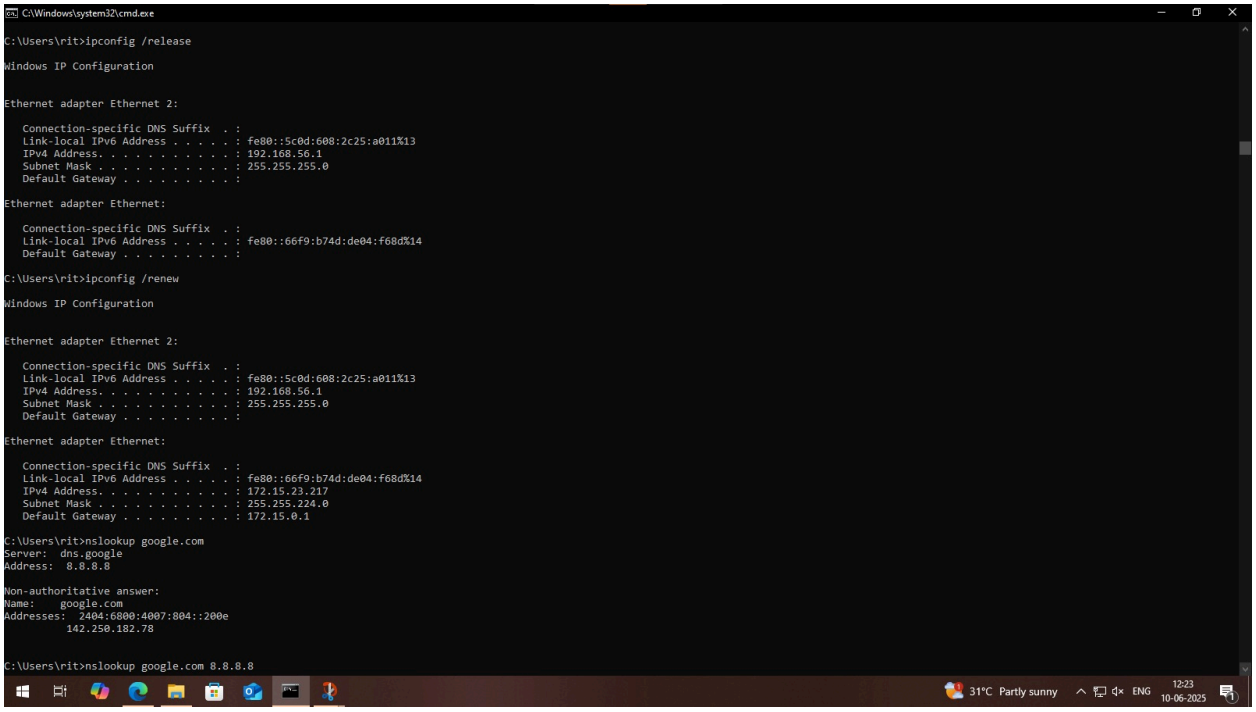

C:\Windows\system32\cmd.exe
TCP 172.15.23.217:49967 172.64.155.209:443 ESTABLISHED 9492
TCP 172.15.23.217:49973 49.249.29.228:443 TIME_WAIT 0
TCP 172.15.23.217:49974 49.249.29.228:443 TIME_WAIT 0
TCP 172.15.23.217:49977 4.213.133.124:443 ESTABLISHED 9492
TCP 172.15.23.217:49978 13.107.6.158:443 ESTABLISHED 9492
TCP 172.15.23.217:49980 20.189.173.10:443 ESTABLISHED 9492
TCP 192.168.56.1:139 0.0.0.0:0 LISTENING 4
TCP [::]:135 [::]:0 LISTENING 488
TCP [::]:445 [::]:0 LISTENING 4
TCP [::]:3306 [::]:0 LISTENING 4672
TCP [::]:33060 [::]:0 LISTENING 4672
TCP [::]:49664 [::]:0 LISTENING 832
TCP [::]:49665 [::]:0 LISTENING 672
TCP [::]:49666 [::]:0 LISTENING 1172
TCP [::]:49667 [::]:0 LISTENING 1652
TCP [::]:49668 [::]:0 LISTENING 2628
TCP [::]:49674 [::]:0 LISTENING 788
UDP 0.0.0.0:123 *:* 5248
UDP 0.0.0.0:5050 *:* 5468
UDP 0.0.0.0:5353 *:* 2356
UDP 0.0.0.0:5353 *:* 9252
UDP 0.0.0.0:5353 *:* 9252
UDP 0.0.0.0:5353 *:* 9252
UDP 0.0.0.0:5353 *:* 9252
UDP 0.0.0.0:5355 *:* 2356
UDP 0.0.0.0:59590 *:* 9492
UDP 0.0.0.0:60053 *:* 9492
UDP 127.0.0.1:1900 *:* 6448
UDP 127.0.0.1:55973 *:* 6448
UDP 127.0.0.1:56559 *:* 3680
UDP 172.15.23.217:137 *:* 4
UDP 172.15.23.217:138 *:* 4
UDP 172.15.23.217:1900 *:* 6448
UDP 172.15.23.217:55972 *:* 6448
UDP 192.168.56.1:137 *:* 4
UDP 192.168.56.1:138 *:* 4
UDP 192.168.56.1:1900 *:* 6448
UDP 192.168.56.1:55971 *:* 6448
UDP [::]:123 *:* 5248
UDP [::]:5353 *:* 2356
UDP [::]:5353 *:* 9252
UDP [::]:5353 *:* 9252
UDP [::]:5355 *:* 2356
UDP [::1]:1900 *:* 6448
UDP [::1]:55970 *:* 6448
UDP [fe80::5c0d:608:2c25:a011%13]:1900 *:* 6448
UDP [fe80::5c0d:608:2c25:a011%13]:55968 *:* 6448
UDP [fe80::66f9:b74d:de04:f68d%14]:1900 *:* 6448
UDP [fe80::66f9:b74d:de04:f68d%14]:55969 *:* 6448
C:\Users\rit>netstat -an 5
31°C Partly sunny
ENG
12:19
10-06-2025

3. ipconfig

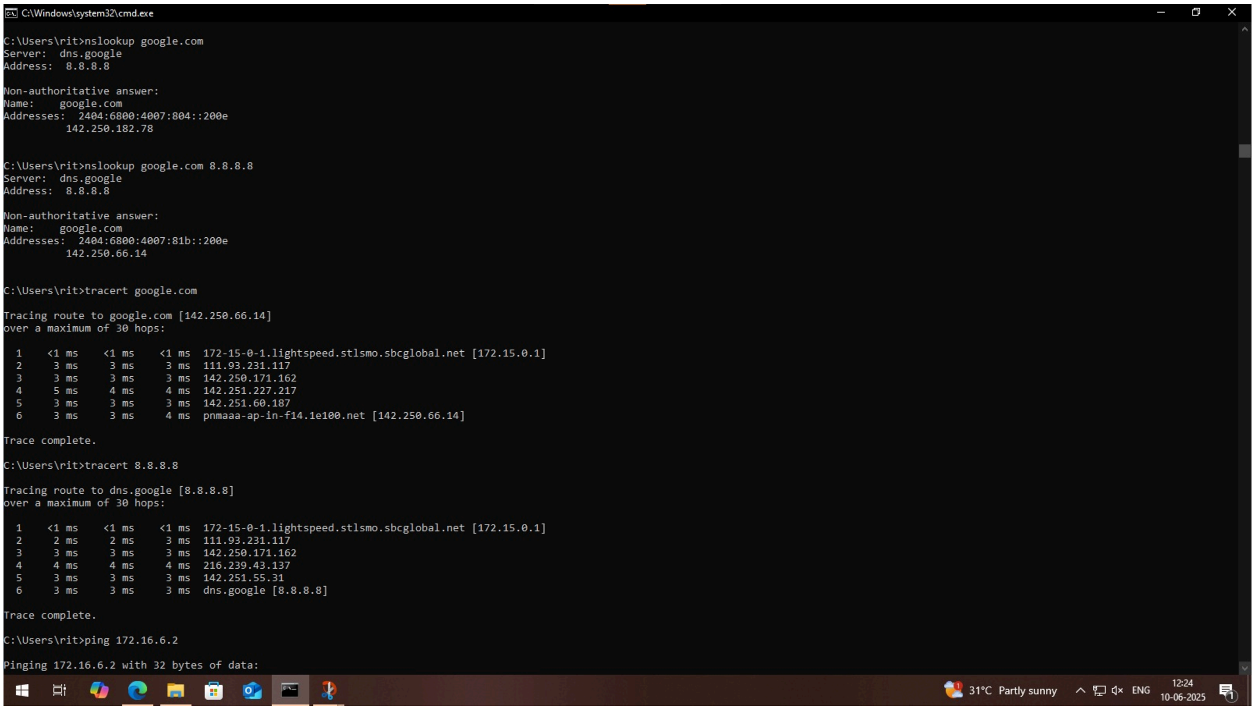

C:\Windows\system32\cmd.exe
TCP 172.15.23.217:49948 23.202.229.22:443 CLOSE_WAIT
TCP 172.15.23.217:49964 13.107.253.58:443 CLOSE_WAIT
TCP 172.15.23.217:49966 13.107.226.254:443 CLOSE_WAIT
TCP 172.15.23.217:49967 172.64.155.209:443 ESTABLISHED
TCP 172.15.23.217:49980 20.189.173.10:443 ESTABLISHED
TCP 172.15.23.217:49982 13.107.6.15^C
C:\Users\rit>ipconfig
Windows IP Configuration
Ethernet adapter Ethernet 2:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::5c0d:608:2c25:a011%13
IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::66f9:b74d:de04:f68d%14
IPv4 Address. . . . . : 172.15.23.217
Subnet Mask . . . . . : 255.255.224.0
Default Gateway . . . . . : 172.15.0.1
C:\Users\rit>ipconfig
Windows IP Configuration
Ethernet adapter Ethernet 2:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::5c0d:608:2c25:a011%13
IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::66f9:b74d:de04:f68d%14
IPv4 Address. . . . . : 172.15.23.217
Subnet Mask . . . . . : 255.255.224.0
Default Gateway . . . . . : 172.15.0.1
C:\Users\rit>ipconfig /all
Windows IP Configuration
31°C Partly sunny
ENG
12:22
10-06-2025


C:\Windows\system32\cmd.exe
C:\Users\rit>ipconfig /all
Windows IP Configuration
Host Name . . . . . : DESKTOP-9BFQQOB
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
Ethernet adapter Ethernet 2:
Connection-specific DNS Suffix . :
Description . . . . . : VirtualBox Host-Only Ethernet Adapter
Physical Address. . . . . : 0A-00-27-00-00-0D
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::5c0d:608:2c25:a011%13(Preferred)
IPv4 Address. . . . . : 192.168.56.1(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 420085799
DHCPv6 Client DUID. . . . . : 00-01-00-01-2E-33-0C-8E-F4-39-09-10-0F-D3
DNS Servers . . . . . : fec0:0:0:ffff::1%1
fec0:0:0:ffff::2%1
fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : F4-39-09-10-0F-D3
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::66f9:b74d:de04:f68d%14(Preferred)
IPv4 Address. . . . . : 172.15.23.217(Preferred)
Subnet Mask . . . . . : 255.255.224.0
Lease Obtained. . . . . : 10 June 2025 11:06:01
Lease Expires . . . . . : 17 June 2025 11:06:00
Default Gateway . . . . . : 172.15.0.1
DHCP Server . . . . . : 172.15.0.1
DHCPv6 IAID . . . . . : 234109193
DHCPv6 Client DUID. . . . . : 00-01-00-01-2E-33-0C-8E-F4-39-09-10-0F-D3
DNS Servers . . . . . : 8.8.8.8
103.8.46.5
103.8.45.5
NetBIOS over Tcpip. . . . . : Enabled
C:\Users\rit>ipconfig /release
31°C Partly sunny
ENG
12:22
10-06-2025

4. Nslookup

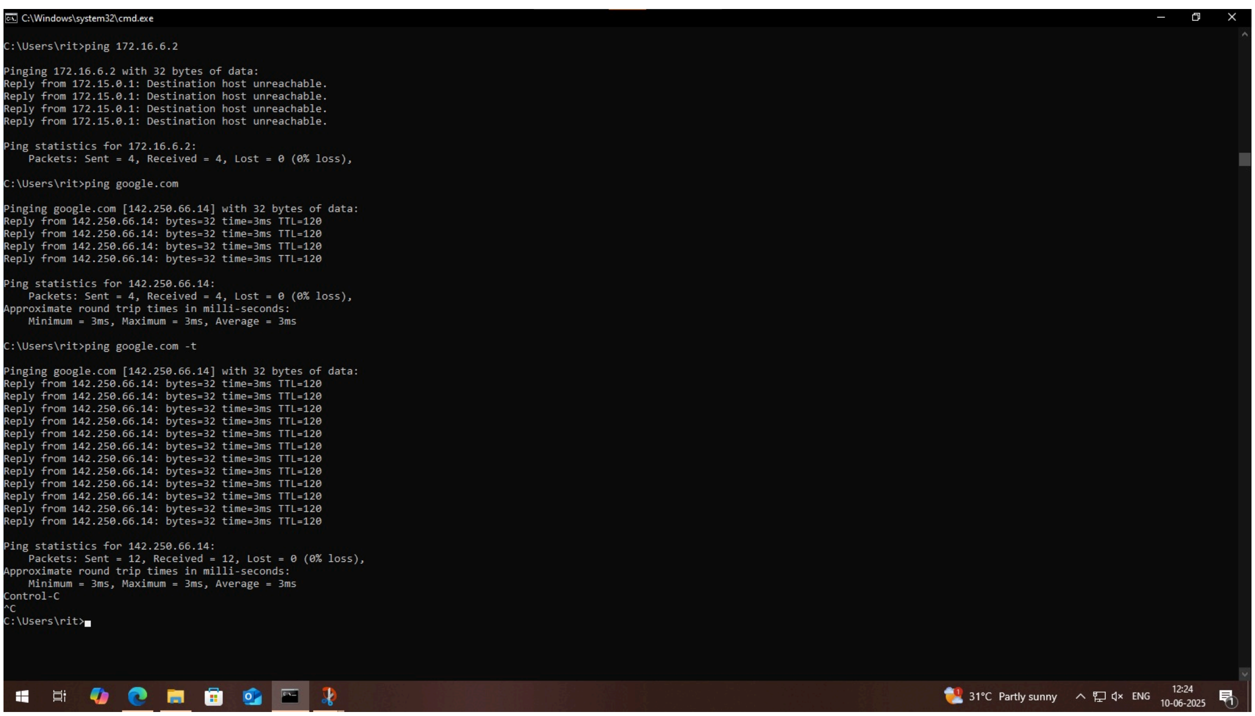

C:\Windows\system32\cmd.exe
C:\Users\rit>ipconfig /release
Windows IP Configuration
Ethernet adapter Ethernet 2:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::5c0d:608:2c25:a011%13
IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::66f9:b74d:de04:f68d%14
Default Gateway . . . . . :
C:\Users\rit>ipconfig /renew
Windows IP Configuration
Ethernet adapter Ethernet 2:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::5c0d:608:2c25:a011%13
IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter Ethernet:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::66f9:b74d:de04:f68d%14
IPv4 Address. . . . . : 172.15.23.217
Subnet Mask . . . . . : 255.255.224.0
Default Gateway . . . . . : 172.15.0.1
C:\Users\rit>nslookup google.com
Server: dns.google
Address: 8.8.8.8
Non-authoritative answer:
Name: google.com
Addresses: 2404:6800:4007:804::200e
142.250.182.78
C:\Users\rit>nslookup google.com 8.8.8.8
31°C Partly sunny
ENG
12:23
10-06-2025

5. Trace route:



```
C:\Users\rit>nslookup google.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4007:804::200e
          142.250.182.78


C:\Users\rit>nslookup google.com 8.8.8.8
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4007:81b::200e
          142.250.66.14


C:\Users\rit>tracert google.com

Tracing route to google.com [142.250.66.14]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172-15-0-1.lightspeed.stlsmo.sbcglobal.net [172.15.0.1]
  2     3 ms     3 ms     3 ms  111.93.231.117
  3     3 ms     3 ms     3 ms  142.250.171.162
  4     5 ms     4 ms     4 ms  142.251.227.217
  5     3 ms     3 ms     3 ms  142.251.60.187
  6     3 ms     3 ms     4 ms  pnmaaa-ap-in-f14.1e100.net [142.250.66.14]

Trace complete.

C:\Users\rit>tracert 8.8.8.8

Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  172-15-0-1.lightspeed.stlsmo.sbcglobal.net [172.15.0.1]
  2     2 ms     2 ms     3 ms  111.93.231.117
  3     3 ms     3 ms     3 ms  142.250.171.162
  4     4 ms     4 ms     4 ms  216.239.43.137
  5     3 ms     3 ms     3 ms  142.251.55.31
  6     3 ms     3 ms     3 ms  dns.google [8.8.8.8]

Trace complete.

C:\Users\rit>ping 172.16.6.2

Pinging 172.16.6.2 with 32 bytes of data:
```


6. Ping:



```
C:\Users\rit>ping 172.16.6.2

Pinging 172.16.6.2 with 32 bytes of data:
Reply from 172.15.0.1: Destination host unreachable.
Reply from 172.15.0.1: Destination host unreachable.
Reply from 172.15.0.1: Destination host unreachable.
Reply from 172.15.0.1: Destination host unreachable.

Ping statistics for 172.16.6.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Users\rit>ping google.com

Pinging google.com [142.250.66.14] with 32 bytes of data:
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120

Ping statistics for 142.250.66.14:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 3ms, Average = 3ms

C:\Users\rit>ping google.com -t

Pinging google.com [142.250.66.14] with 32 bytes of data:
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120
Reply from 142.250.66.14: bytes=32 time=3ms TTL=120

Ping statistics for 142.250.66.14:
    Packets: Sent = 12, Received = 12, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 3ms, Average = 3ms
Control-C
^C
C:\Users\rit>
```

PROGRAM:

Client

```
import javax.swing.*;
import java.net.*;
import java.awt.image.*;
import javax.imageio.ImageIO;
import java.io.*;
public class Client {
  public static void main(String[] args) {
     Socket soc = null;
     BufferedImage img = null;
     try {
        soc = new Socket("localhost", 4000);
        System.out.println("Client is running.");
        // Read image from disk (your actual path)
        System.out.println("Reading image from disk.");
        img = ImageIO.read(new File("C:\\Users\\JOHIT
A\\OneDrive\\Pictures\\Screenshot_19-10-2024_23854_www.bing.com.jpg"));
        // Check if image was loaded successfully
        if (img == null) {
           System.out.println("Could not read image. Check the path or file format.");
           soc.close();
           return;
        }
        // Convert image to byte array
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ImageIO.write(img, "jpg", baos);
        baos.flush();
        byte[] bytes = baos.toByteArray();
        baos.close();
        // Send image to server
        System.out.println("Sending image to server.");
        OutputStream out = soc.getOutputStream();
        DataOutputStream dos = new DataOutputStream(out);
        dos.writeInt(bytes.length);       // Send length of image
        dos.write(bytes, 0, bytes.length); // Send image bytes
        System.out.println("Image sent to server.");
        // Close streams
        dos.close();
        out.close();
```

```
        } catch (Exception e) {
            e.printStackTrace(); // Show detailed error
        } finally {
            try {
                if (soc != null) {
                    soc.close();
                }
            } catch (IOException e) {
                System.out.println("Error closing socket: " + e.getMessage());
            }
        }
    }
}
```

Server

```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.ImageIO;
import javax.swing.*;
public class Server {
    public static void main(String[] args) {
        ServerSocket server = null;
        Socket socket = null;
        try {
            server = new ServerSocket(4000);
            System.out.println("Server waiting for image...");
            socket = server.accept();  // Wait for client
            System.out.println("Client connected.");
            // Receive image data
            InputStream in = socket.getInputStream();
            DataInputStream dis = new DataInputStream(in);
            int len = dis.readInt();  // Receive length of image
            System.out.println("Image Size: " + len / 1024 + " KB");
            byte[] data = new byte[len];
            dis.readFully(data);  // Receive image bytes
            dis.close();
            in.close();
            // Convert byte array back to image
            InputStream ian = new ByteArrayInputStream(data);
```

```
            BufferedImage bImage = ImageIO.read(ian);
            if (bImage == null) {
                System.out.println("Failed to decode received image.");
                return;
            }
            // Display image using JFrame
            JFrame f = new JFrame("Server - Received Image");
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JLabel l = new JLabel(new ImageIcon(bImage));
            f.add(l);
            f.pack();
            f.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace(); // Show detailed error
        } finally {
            try {
                if (socket != null)
                    socket.close();
                if (server != null)
                    server.close();
            } catch (IOException e) {
                System.out.println("Error closing socket: " + e.getMessage());
            }
        }
    }
}
```

OUTPUT:

Server waiting for image...
Client connected.
Image Size: 6 KB

PROGRAM:

EchoServer.java

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

public class EServer {
   public static void main(String[] args) {
      ServerSocket serverSocket = null;
      Socket clientSocket = null;
      DataInputStream input = null;
      PrintStream output = null;

      try {
         serverSocket = new ServerSocket(9000);
         System.out.println("Server started. Waiting for client...");

         clientSocket = serverSocket.accept();
         System.out.println("Client connected.");

         input = new DataInputStream(clientSocket.getInputStream());
         output = new PrintStream(clientSocket.getOutputStream());

         String line;
         while (true) {
            line = input.readLine();
            if (line == null || line.equalsIgnoreCase("exit")) {
               System.out.println("Connection closed by client.");
               break;
            }
            System.out.println("Received from client: " + line);
            output.println(line);  // Echo back to client
         }
      } catch (IOException e) {
         System.out.println("Error: " + e);
      } finally {
         try {
            if (input != null) input.close();
            if (output != null) output.close();
            if (clientSocket != null) clientSocket.close();
            if (serverSocket != null) serverSocket.close();
```

```
            } catch (IOException e) {
                System.out.println("Error closing resources: " + e);
            }
        }
    }
}
```

EClient.java

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

public class EClient {
    public static void main(String[] args) {
        Socket socket = null;
        DataInputStream userInput = null;
        DataInputStream serverInput = null;
        PrintStream output = null;

        try {
            InetAddress ia = InetAddress.getLocalHost();
            socket = new Socket(ia, 9000);
            System.out.println("Connected to server.");

            userInput = new DataInputStream(System.in);              // Keyboard input
            serverInput = new DataInputStream(socket.getInputStream()); // Server response
            output = new PrintStream(socket.getOutputStream());      // Sending to server

            String line;
            while (true) {
                System.out.print("Client: ");
                line = userInput.readLine();
                if (line.equalsIgnoreCase("exit")) {
                    System.out.println("Exiting client...");
                    break;
                }
                output.println(line);
                String response = serverInput.readLine();
                System.out.println("Server: " + response);
            }
        } catch (IOException e) {
```

```
            System.out.println("Error: " + e);
        } finally {
            try {
                if (userInput != null) userInput.close();
                if (serverInput != null) serverInput.close();
                if (output != null) output.close();
                if (socket != null) socket.close();
            } catch (IOException e) {
                System.out.println("Error closing resources: " + e);
            }
        }
    }
}
```

OUTPUT:

```
Server started. Waiting for client...
Client connected
Connected to server.
Client: heyy
Server: heyy
Client: how r u?
Server: how r u?
Client:exit
Received from client: heyy
Received from client: how r u?
Connection closed by client.
```

PROGRAM

Chatserver.java:

```
import java.net.*;
import java.io.*;
public class chatserver {
  public static void main(String[] args) throws Exception {
     ServerSocket serverSocket = new ServerSocket(2000);
     System.out.println("Server is running. Waiting for client...");
     Socket socket = serverSocket.accept();
     System.out.println("Client connected.");
     BufferedReader clientInput = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
     PrintStream clientOutput = new PrintStream(socket.getOutputStream());
     BufferedReader serverInput = new BufferedReader(new InputStreamReader(System.in));
     String message;
     while (true) {
        message = clientInput.readLine();
        if (message == null) break;
        if (message.equalsIgnoreCase("END")) {
           clientOutput.println("BYE");
           System.out.println("Client ended the chat.");
           break;
        }
        System.out.println("Client: " + message);
        System.out.print("Server: ");
        message = serverInput.readLine();
        clientOutput.println(message);
     }
     // Close all resources
     serverSocket.close();
     socket.close();
     clientInput.close();
     clientOutput.close();
     serverInput.close();
  }
}
```

Chatclient.java

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

```
public class chatclient {
  public static void main(String[] args) throws Exception {
     Socket socket = new Socket("127.0.0.1", 2000);
     System.out.println("Connected to server.");
     BufferedReader serverInput = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
     PrintStream serverOutput = new PrintStream(socket.getOutputStream());
     BufferedReader clientInput = new BufferedReader(new InputStreamReader(System.in));
     String message;
     while (true) {
        System.out.print("Client: ");
        message = clientInput.readLine();
        serverOutput.println(message);
        message = serverInput.readLine();
        System.out.println("Server: " + message);
        if (message.equalsIgnoreCase("BYE"))
           break;
     }
     // Close all resources
     socket.close();
     serverInput.close();
     serverOutput.close();
     clientInput.close();
  }
}
```

OUTPUT:

Server is running. Waiting for client...
Connected to server.
Client: hii
Server: heyy
Client: how r u?
Server: fine
Client: byee

PROGRAM

File Server :

```
import java.io.*;
import java.net.*;
public class FileServer {
  public static void main(String[] args) {
     ServerSocket serverSocket = null;
     Socket socket = null;
     BufferedInputStream bis = null;
     OutputStream os = null;
     try {
        serverSocket = new ServerSocket(5000);
        System.out.println("Server started. Waiting for client...");
        socket = serverSocket.accept();
        System.out.println("Client connected. Sending file...");
        // Use your actual file path here
        File file = new File("C:\\Users\\JOHIT A\\Downloads\\LAB MANUAL.pdf");
        if (!file.exists()) {
           System.out.println("❌ File not found. Please check the path.");
           return;
        }
        FileInputStream fis = new FileInputStream(file);
        bis = new BufferedInputStream(fis);
        os = socket.getOutputStream();
        byte[] buffer = new byte[10000];
        int bytesRead;
        long totalSent = 0;
        long fileSize = file.length();
        while ((bytesRead = bis.read(buffer)) != -1) {
           os.write(buffer, 0, bytesRead);
           totalSent += bytesRead;
           System.out.println("Sending file... " + (totalSent * 100) / fileSize + "% complete");
        }
        os.flush();
        System.out.println("✅ File sent successfully!");
     } catch (IOException e) {
        System.out.println("Server Error: " + e.getMessage());
     } finally {
        try {
           if (bis != null) bis.close();
```

```
            if (os != null) os.close();
            if (socket != null) socket.close();
            if (serverSocket != null) serverSocket.close();
        } catch (IOException e) {
            System.out.println("Error closing: " + e.getMessage());
        }
    }
  }
}
```

File Client:

```
import java.io.*;
import java.net.*;
public class FileClient {
  public static void main(String[] args) {
    Socket socket = null;
    InputStream is = null;
    BufferedOutputStream bos = null;
    try {
      socket = new Socket(InetAddress.getByName("localhost"), 5000);
      System.out.println("Connected to server. Receiving file...");
      is = socket.getInputStream();
      // Save file as a copy
      FileOutputStream fos = new FileOutputStream("C:\\Users\\JOHIT A\\Downloads\\LAB
MANUAL - RECEIVED.pdf");
      bos = new BufferedOutputStream(fos);
      byte[] buffer = new byte[10000];
      int bytesRead;
      while ((bytesRead = is.read(buffer)) != -1) {
        bos.write(buffer, 0, bytesRead);
      }
      bos.flush();
      System.out.println("✅ File received and saved successfully!");
    } catch (IOException e) {
      System.out.println("Client Error: " + e.getMessage());
    } finally {
      try {
```

```
            if (bos != null) bos.close();
            if (is != null) is.close();
            if (socket != null) socket.close();
        } catch (IOException e) {
            System.out.println("Error closing: " + e.getMessage());
        }
    }
  }
}
```

OUTPUT:

Server started. Waiting for client...
✅ File received and saved successfully!

PROGRAM

DNS Server

```
import java.io.*;
import java.net.*;
public class udpdnsserver {
  private static int indexOf(String[] array, String str) {
     str = str.trim();
     for (int i = 0; i < array.length; i++) {
        if (array[i].equalsIgnoreCase(str))
           return i;
     }
     return -1;
  }
  public static void main(String arg[]) throws IOException {
     String[] hosts = {"yahoo.com", "gmail.com", "cricinfo.com", "facebook.com"};
     String[] ips = {"68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16"};
     DatagramSocket serverSocket = new DatagramSocket(1362);
     System.out.println("DNS Server is running on port 1362... Press Ctrl+C to stop.");
     while (true) {
        byte[] receiveData = new byte[1024];
        DatagramPacket recvPacket = new DatagramPacket(receiveData, receiveData.length);
        serverSocket.receive(recvPacket);
        String receivedHost = new String(recvPacket.getData(), 0, recvPacket.getLength()).trim();
        InetAddress clientAddress = recvPacket.getAddress();
        int clientPort = recvPacket.getPort();
        System.out.println("Request for host: " + receivedHost);
        String ipResponse;
        int index = indexOf(hosts, receivedHost);
        ipResponse = (index != -1) ? ips[index] : "Host Not Found";
        byte[] sendData = ipResponse.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
clientAddress, clientPort);
        serverSocket.send(sendPacket);
     }
  }
}
```

UPD DNS CLIENT

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

```
public class udpdnsclient {
  public static void main(String args[]) throws IOException {
     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
     DatagramSocket clientSocket = new DatagramSocket();
     InetAddress serverAddress;
     if (args.length == 0)
        serverAddress = InetAddress.getLocalHost(); // default to localhost
     else
        serverAddress = InetAddress.getByName(args[0]);
     byte[] sendData = new byte[1024];
     byte[] receiveData = new byte[1024];
     int port = 1362;
     System.out.print("Enter the hostname: ");
     String host = br.readLine();
     sendData = host.getBytes();
     DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
serverAddress, port);
     clientSocket.send(sendPacket);
     DatagramPacket recvPacket = new DatagramPacket(receiveData, receiveData.length);
     clientSocket.receive(recvPacket);
     String response = new String(recvPacket.getData(), 0, recvPacket.getLength());
     System.out.println("IP Address: " + response);
     clientSocket.close();
  }
}
```

OUTPUT:

```
DNS Server is running on port 1362... Press Ctrl+C to stop.
Enter the hostname: yahoo.com
🌐 IP Address: 68.180.206.184
```