1. Capturing Ping and Traceroute PDUs Using Wireshark

Aim

To capture and analyze **Ping** and **Traceroute** PDUs using Wireshark and study ICMP packet behavior in an IP network.

Algorithm

1. Start Wireshark and select the active network interface.
2. Begin packet capture.
3. Execute the ping command to a known host.
4. Stop capture and apply ICMP filter.
5. Restart capture and execute tracert / traceroute.
6. Analyze ICMP Echo Request, Echo Reply, and Time Exceeded packets.

---

2. HTTP Web Client Using TCP Sockets

Aim

To design and implement a simple **HTTP web client** using **TCP sockets** to retrieve a webpage from a web server.

Algorithm

1. Create a TCP socket.
2. Connect the socket to the web server using port 80.
3. Send an HTTP GET request to the server.
4. Receive the HTTP response from the server.
5. Display the response and close the socket.

---

3. Echo Client and Server (TCP-Based Chat Application)

Aim

To implement a **TCP-based echo client and server** that exchanges messages between two hosts.

Algorithm

1. Create a TCP socket on the server side.
2. Bind the socket to an IP address and port.

3. Listen for incoming client connections.
4. Accept client connection and receive data.
5. Send the same data back to the client (echo).
6. Close the connection.

---

4. File Transfer Using TCP Sockets

Aim

To transfer a file from a server to a client using **TCP socket communication**.

Algorithm

1. Create and bind a TCP socket on the server.
2. Listen for client connection.
3. Accept client request.
4. Open the file in binary mode.
5. Send file data to the client.
6. Client receives data and stores it in a file.
7. Close the connection.

---

5. DNS Resolver Using UDP Sockets

Aim

To implement a **DNS resolver** using **UDP sockets** to obtain domain name resolution.

Algorithm

1. Create a UDP socket.
2. Construct a DNS query packet.
3. Send the query to a DNS server on port 53.
4. Receive the DNS response packet.
5. Display the received response.
6. Close the socket.

---

6. TCP vs UDP Communication

Aim

To compare **TCP and UDP protocols** by implementing basic client-server communication using sockets.

Algorithm

1. Create TCP client and server sockets.
2. Establish connection and exchange data using TCP.
3. Create UDP client and server sockets.
4. Send and receive datagrams using UDP.
5. Observe reliability and connection behavior.
6. Compare the performance characteristics of TCP and UDP.