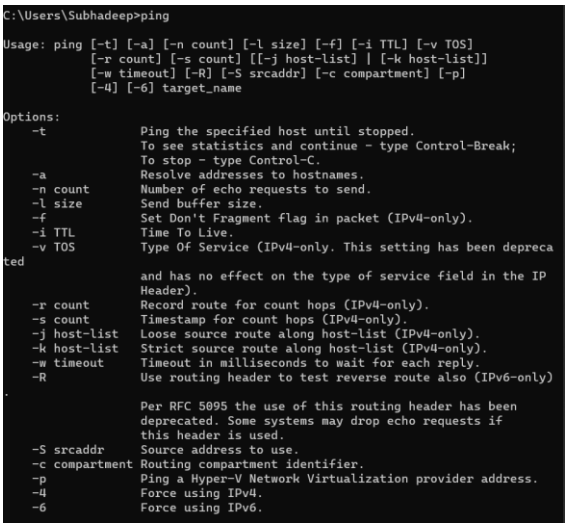
Assignment :1

Title: Study of different types of Networking Commands used for analyzing and troubleshooting.

Introduction: In the realm of networking, understanding and mastering various networking commands are essential for analyzing and troubleshooting network-related issues effectively. These commands provide insights into network configurations, connectivity, performance, and security, enabling network administrators and engineers to diagnose and resolve problems efficiently. From basic tools like Ping and Traceroute to advanced utilities such as Netstat and SSH, each command serves a specific purpose in dissecting network behavior and identifying potential issues.

Objective: The objective is to equip individuals with the knowledge and skills to efficiently diagnose and resolve network problems. By mastering these commands, professionals can identify connectivity issues, analyze configurations, monitor performance, investigate security incidents, and securely access remote systems.

1. Ping: Ping is a fundamental tool used to test the reachability of a host on a network, assess network connectivity, and measure the round-trip time for packets sent from the source to the destination.

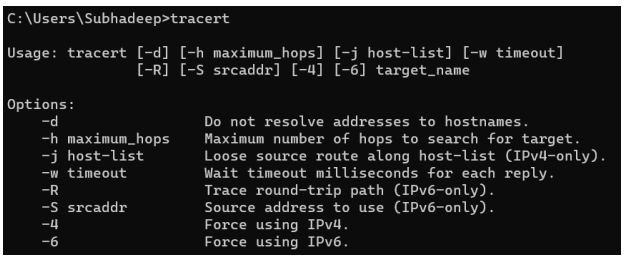


```
C:\Users\Subhadeep>ping

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been depreca
ted
                   and has no effect on the type of service field in the IP
                   Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only)
.
                   Per RFC 5095 the use of this routing header has been
                   deprecated. Some systems may drop echo requests if
                   this header is used.
    -S srcaddr     Source address to use.
    -c compartment Routing compartment identifier.
    -p             Ping a Hyper-V Network Virtualization provider address.
    -4             Force using IPv4.
    -6             Force using IPv6.
```

2. Traceroute (Tracert): Traceroute, also known as Tracert in Windows, is employed to determine the route packets take to reach a destination, identify network hops, and diagnose potential points of failure or latency issues along the path.

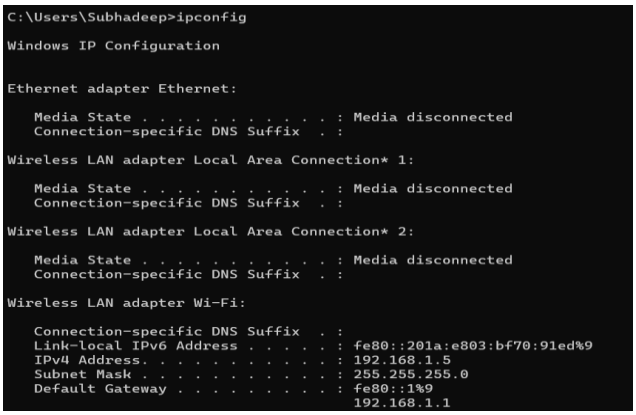


```
C:\Users\Subhadeep>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
               [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -d                 Do not resolve addresses to hostnames.
    -h maximum_hops    Maximum number of hops to search for target.
    -j host-list       Loose source route along host-list (IPv4-only).
    -w timeout         Wait timeout milliseconds for each reply.
    -R                 Trace round-trip path (IPv6-only).
    -S srcaddr         Source address to use (IPv6-only).
    -4                 Force using IPv4.
    -6                 Force using IPv6.
```

3. ifconfig / ipconfig: Ifconfig (on Unix-like systems) or ipconfig (on Windows) allows users to inspect the IP configuration of network interfaces, revealing crucial details such as IP addresses, subnet masks, and gateways, aiding in troubleshooting network connectivity problems.



```
C:\Users\Subhadeep>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::201a:e803:bf70:91ed%9
   IPv4 Address. . . . . . . . . . . : 192.168.1.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%9
                                       192.168.1.1
```

4. Netstat: Netstat facilitates the display of active network connections, routing tables, and interface statistics, enabling the identification of network services and associated ports for analysis and troubleshooting purposes.

```
C:\Users\Subhadeep>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:49670        DOPADELIX:49671        ESTABLISHED
  TCP    127.0.0.1:49671        DOPADELIX:49670        ESTABLISHED
  TCP    127.0.0.1:49672        DOPADELIX:49673        ESTABLISHED
  TCP    127.0.0.1:49673        DOPADELIX:49672        ESTABLISHED
  TCP    127.0.0.1:49765        DOPADELIX:65001        ESTABLISHED
  TCP    127.0.0.1:65001        DOPADELIX:49765        ESTABLISHED
```

5. Nslookup: Nslookup serves as a tool for querying DNS servers to resolve hostnames to IP addresses, troubleshoot DNS resolution issues, and retrieve DNS-related information.

```
C:\Users\Subhadeep>nslookup
Default Server:  UnKnown
Address:  192.168.1.1
```

6. Route: Route displays and modifies the IP routing table, assisting in determining routing paths for packets and troubleshooting routing and connectivity problems.

```
C:\Users\Subhadeep>route

Manipulates network routing tables.

ROUTE [-f] [-p] [-4|-6] command [destination]
                  [MASK netmask]  [gateway] [METRIC metric]  [IF interface]

  -f           Clears the routing tables of all gateway entries.  If this is
               used in conjunction with one of the commands, the tables are
               cleared prior to running the command.

  -p           When used with the ADD command, makes a route persistent acro
ss
               boots of the system. By default, routes are not preserved
               when the system is restarted. Ignored for all other commands,

               which always affect the appropriate persistent routes.

  -4           Force using IPv4.

  -6           Force using IPv6.

  command      One of these:
                 PRINT     Prints  a route
                 ADD       Adds    a route
                 DELETE    Deletes a route
                 CHANGE    Modifies an existing route
  destination  Specifies the host.
  MASK         Specifies that the next parameter is the 'netmask' value.
  netmask      Specifies a subnet mask value for this route entry.
               If not specified, it defaults to 255.255.255.255.
  gateway      Specifies gateway.
  interface    the interface number for the specified route.
  METRIC       specifies the metric, ie. cost for the destination.

All symbolic names used for destination are looked up in the network databas
e
file NETWORKS. The symbolic names for gateway are looked up in the host name
```

7. Arp: Arp enables the display and modification of the ARP cache, facilitating the mapping of IP addresses to MAC addresses and troubleshooting ARP-related issues.

```
C:\Users\Subhadeep>arp

Displays and modifies the IP-to-Physical address translation tables used by
address resolution protocol (ARP).

ARP -s inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -a [inet_addr] [-N if_addr] [-v]

  -a            Displays current ARP entries by interrogating the current
                protocol data.  If inet_addr is specified, the IP and Physic
al
                addresses for only the specified computer are displayed.  If
                more than one network interface uses ARP, entries for each A
RP
                table are displayed.
  -g            Same as -a.
  -v            Displays current ARP entries in verbose mode.  All invalid
                entries and entries on the loop-back interface will be shown
.
  inet_addr     Specifies an internet address.
  -N if_addr    Displays the ARP entries for the network interface specified
                by if_addr.
  -d            Deletes the host specified by inet_addr. inet_addr may be
                wildcarded with * to delete all hosts.
  -s            Adds the host and associates the Internet address inet_addr
                with the Physical address eth_addr.  The Physical address is
                given as 6 hexadecimal bytes separated by hyphens. The entry
                is permanent.
  eth_addr      Specifies a physical address.
  if_addr       If present, this specifies the Internet address of the
                interface whose address translation table should be modified
.
                If not present, the first applicable interface will be used.
Example:
  > arp -s 157.55.85.212   00-aa-00-62-c6-09  .... Adds a static entry.
  > arp -a                                    .... Displays the arp table.
```

8. SSH (Secure Shell): SSH (Secure Shell) provides secure access to remote systems, enabling users to execute commands securely and transfer files securely between hosts.

```
C:\Users\Subhadeep>ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command]
```

9. SCP (Secure Copy): SCP (Secure Copy) allows for secure file transfers between local and remote hosts over SSH, ensuring data integrity and confidentiality.

```
C:\Users\Subhadeep>scp
usage: scp [-346ABCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-J destination] [-l limit] [-o ssh_option] [-P port]
           [-S program] source ... target
```

10. Wget (or Curl): Wget or Curl enables the downloading of files from web servers over HTTP/HTTPS, testing web server availability, and retrieving content from URLs for scripting or automation tasks.

```
C:\Users\Subhadeep>wget
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

11. Hostname: The hostname command, used in computing, displays or sets the system's name, crucial for network identification and communication settings, facilitating efficient network operations and system configuration.

```
C:\Users\Subhadeep>hostname
DOPADELIX
```

Conclusion: Mastering networking commands is essential for effective network management. With these tools, professionals can swiftly diagnose issues, optimize performance, and ensure the security of network infrastructure, contributing to smoother operations and enhanced efficiency.

Assignment : 2

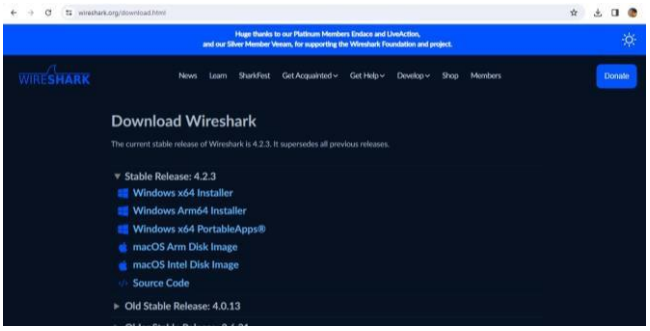
Title: Write the steps of Network packet tracing using Wireshark.

Introduction: Wireshark is a powerful network protocol analyzer that enables real-time examination and troubleshooting of network traffic. This guide will help you download Wireshark and utilize its capabilities for packet tracing, providing insights into network behavior.

Objective: This guide aims to provide you with the knowledge and skills to efficiently download Wireshark, configure it to capture network traffic from different interfaces, analyze captured packets to understand communication patterns, apply filters for focused analysis, and save captured data for future reference. By mastering these skills, you'll gain the ability to troubleshoot network issues, optimize performance, and enhance network security effectively.

Download Wireshark:

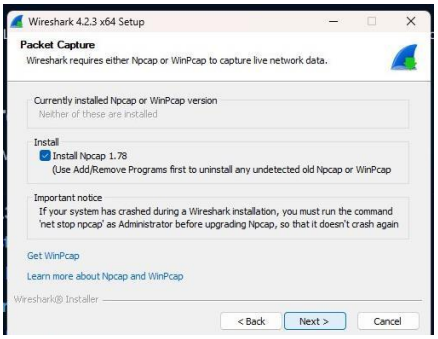
Visit the Wireshark website: Go to the official Wireshark website by typing the following URL into your web browser's address bar: https://www.wireshark.org/.

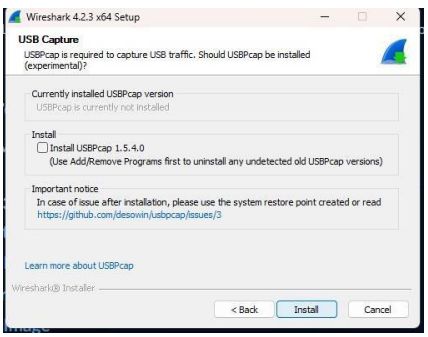


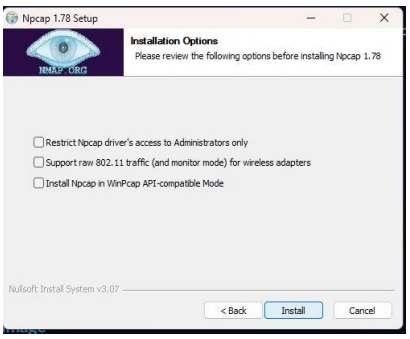
Navigate to the Download section: Once you're on the Wireshark website's homepage, look for the "Download" section. This is usually prominently displayed on the main page.

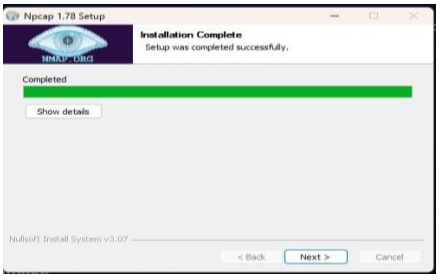
Choose your operating system: Wireshark is available for various operating systems including Windows, macOS, and Linux. Click on the appropriate download link for your operating system.

Download the installer: Click on the download link provided for your operating system. This will start the download of the Wireshark installer file. The file size may vary depending on your operating system and the version of Wireshark you are downloading









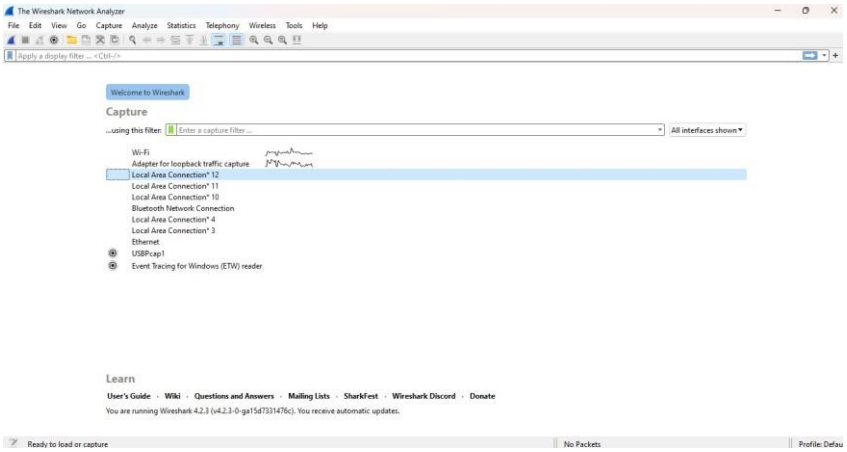
Verify the download (optional): It's a good practice to verify the integrity of the downloaded file, especially for security reasons. Some websites provide checksums or digital signatures for this purpose. Check the Wireshark website for any provided verification instructions.

Run the installer: Once the download is complete, locate the downloaded installer file on your system (usually in your Downloads folder) and double-click on it to run the installer.

Follow the installation wizard: The Wireshark installation wizard will guide you through the installation process. Follow the on-screen instructions, such as accepting the license agreement, choosing the installation directory, and selecting additional components if prompted.

Complete the installation: After you've configured the installation settings, proceed with the installation process by clicking on the "Install" or "Next" button. The installer will then proceed to install Wireshark on your system.

Launch Wireshark: After installation, launch Wireshark from the desktop shortcut or application menu.



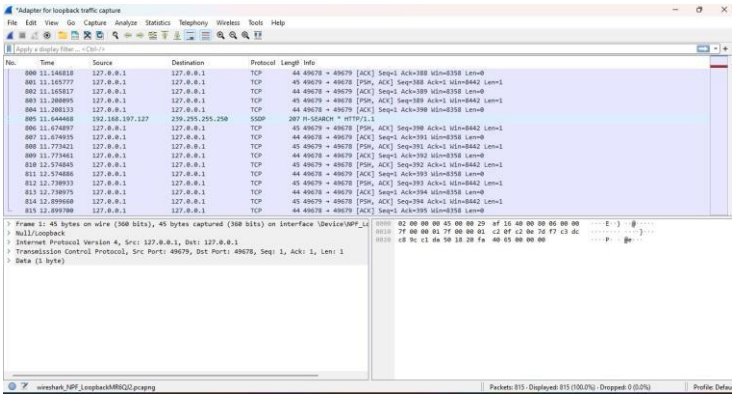
Select Network Interface:

Upon launching, Wireshark will display a list of available network interfaces.

Choose the network interface through which you want to capture packets. This could be your Wi-Fi adapter, Ethernet adapter, etc.

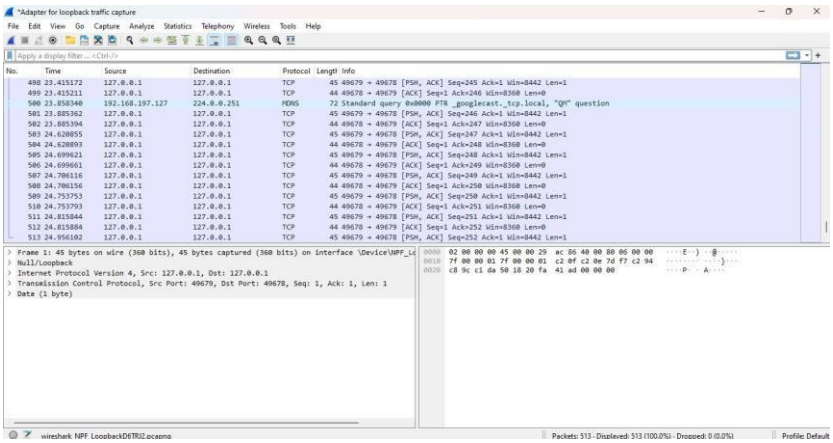
Start Capturing Packets: Once the desired interface is selected, click on the "Start" or "Capture" button to begin

capturing packets. Wireshark will start displaying captured packets in real-time.



Analyze Packets: You can analyze captured packets by examining various details such as source and destination IP addresses, protocols, packet size, etc. Use Wireshark's filter options to focus on specific types of packets or communication.

Stop Capturing Packets: When you've captured enough data or want to stop capturing, click on the "Stop" or "Capture" button.



Save Captured Packets: You can save the captured packets for later analysis by going to "File" > "Save As" and choosing a file format (e.g., pcap).

Analyze Saved Packets: If you've saved the captured packets, you can reopen them in Wireshark later for further analysis.

Additional Features: Explore Wireshark's various features such as protocol analysis, statistics generation, and packet decryption for encrypted traffic.

Learn and Experiment: Wireshark is a powerful tool with many features. Take some time to explore its capabilities, experiment with different settings, and learn how to interpret packet data effectively.

Conclusion: Wireshark is an essential tool for network analysis, offering valuable insights into network communication. By mastering its features, you can effectively troubleshoot issues, optimize performance, and enhance security. Embrace Wireshark to unlock the full potential of your network infrastructure.

Assignment: 3

Title: Write a programming to implement Single server Socket Programming in Java.

Introduction: Socket programming is a fundamental aspect of network communication, allowing applications to communicate across a network using the Internet Protocol (IP). In this paradigm, sockets serve as the endpoints for communication, facilitating the exchange of data between different processes, either on the same machine or across the network. By utilizing sockets, developers can create versatile and powerful networked applications, ranging from simple client-server interactions to complex distributed systems.

Objective: The primary objective of socket programming is to enable communication between processes over a network. This involves establishing connections, sending and receiving data, and handling various network protocols. By mastering socket programming, developers aim to create efficient, reliable, and scalable network applications that can fulfill a wide range of requirements, including real-time data exchange, remote procedure calls, and distributed computing tasks.

Code:

//Creating Server

```
import java.io.*;

import java.net.*;

public class MyServer1 {

public static void main(String[] args){

try{

ServerSocket ss=new ServerSocket(6666);

Socket s=ss.accept();//establishes connection

DataInputStream dis=new DataInputStream(s.getInputStream());

String  str=(String)dis.readUTF();

System.out.println("message= "+str);

ss.close();

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

} }
```

//Creating Client

```
import java.io.*;

import java.net.*;

public class MyClient1 {

public static void main(String[] args) {
```

```
try{

Socket s=new Socket("localhost",6666);

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

dout.writeUTF("Hello Server");

dout.flush();

dout.close();

s.close();

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

} }
```

Output:

Server Output

```
C:\Users\Subhadeep>cd documents

C:\Users\Subhadeep\Documents>javac MyServer1.java

C:\Users\Subhadeep\Documents>java MyServer1
message= Hello Server

C:\Users\Subhadeep\Documents>
```

Client Output

```
C:\Users\Subhadeep>cd documents

C:\Users\Subhadeep\Documents>javac MyClient1.java

C:\Users\Subhadeep\Documents>java MyClient1

C:\Users\Subhadeep\Documents>
```

Conclusion: Socket programming plays a crucial role in modern networked applications, empowering developers to create dynamic and interconnected systems. By leveraging sockets, applications can communicate seamlessly across networks, enabling functionalities such as file transfer, remote command execution, and real-time data streaming. Understanding socket programming principles equips developers with the tools to build robust and scalable networked solutions, facilitating efficient data exchange and collaboration in today's interconnected world.

Assignment: 4

Title: Write a programming to implement Multithreading Socket programming Java

Introduction:

Multithreaded socket programming is an advanced technique in network communication that utilizes multiple threads to handle concurrent connections between clients and servers. In this approach, each client connection is managed by a separate thread, allowing the server to handle multiple clients simultaneously. Multithreaded socket programming enhances the scalability and responsiveness of networked applications by efficiently utilizing system resources and enabling parallel processing of client requests.

Objective:

The primary objective of multithreaded socket programming is to improve the performance and scalability of networked applications by leveraging the concurrency capabilities of modern operating systems. By employing multiple threads, developers aim to maximize the utilization of system resources and enhance the responsiveness of servers to client requests. The objective is to design efficient and scalable networked systems that can handle a large number of concurrent connections while maintaining high performance and reliability.

Code:

```
// Server class

import java.io.*;

import java.net.*;

 class Server1 {

   public static void main(String[] args) {

      ServerSocket server = null;

   try {

    server = new ServerSocket(1234);

          server.setReuseAddress(true);

      while (true) {

     Socket client = server.accept();

  System.out.println("New client connected"+ client.getInetAddress() .getHostAddress());

  ClientHandler clientSock = new ClientHandler(client);

  new Thread(clientSock).start();

        } }

    catch (IOException e) {

   e.printStackTrace();   }

      finally {
```

```
        if (server != null) {
            try {
                server.close();  }
            catch (IOException e) {
                e.printStackTrace();
            } } } }
    private static class ClientHandler implements Runnable {
    private final Socket clientSocket;
  public ClientHandler(Socket socket)  {
    this.clientSocket = socket;  }
   public void run() {
    PrintWriter out = null;
        BufferedReader in = null;
        try {
   out = new PrintWriter(clientSocket.getOutputStream(), true);
   in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    String line;
            while ((line = in.readLine()) != null) {
   System.out.printf(" Sent from the client: %s\n",line);
                out.println(line);
            }  }
         catch (IOException e) {
            e.printStackTrace();  }
        finally {
            try {
                if (out != null) {
                    out.close(); }
                if (in != null) {
                    in.close();
                    clientSocket.close();
```

```
            } }
            catch (IOException e) {
               e.printStackTrace();
            } } } } }
// Client class
import java.io.*;
import java.net.*;
import java.util.*;
class Client1 {
public static void main(String[] args) {
   try (Socket socket = new Socket("localhost", 1234)) {
      PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
      BufferedReader in= new BufferedReader(new InputStreamReader(socket.getInputStream()));
      Scanner sc = new Scanner(System.in);
          String line = null;
       while (!"exit".equalsIgnoreCase(line)) {
          line = sc.nextLine();
         out.println(line);
              out.flush();
         System.out.println("Server replied "+ in.readLine());   }
          sc.close();  }
        catch (IOException e) {
          e.printStackTrace();
       } } }
```

Output:

Server Output

```
C:\Users\Subhadeep\Documents>javac Server1.java

C:\Users\Subhadeep\Documents>java Server1
New client connected127.0.0.1
New client connected127.0.0.1
 Sent from the client: hi this is subhadeep
 Sent from the client: hi this is subz
```

Client Output

```
C:\Users\Subhadeep>cd documents

C:\Users\Subhadeep\Documents>javac Client1.java

C:\Users\Subhadeep\Documents>java Client1
hi this is subz
Server replied hi this is subz
```

```
C:\Users\Subhadeep\Documents>javac Client1.java

C:\Users\Subhadeep\Documents>java Client1
hi this is subhadeep
Server replied hi this is subhadeep
```

Conclusion:

Multithreaded socket programming offers a powerful solution for building high-performance networked applications capable of handling concurrent connections efficiently. By utilizing multiple threads, developers can create responsive and scalable server architectures that can accommodate a large number of clients simultaneously. Multithreaded socket programming enables the development of robust and scalable networked systems that can meet the demands of modern applications, ranging from web servers to real-time communication platforms.