31341

Assignment B4

Problem Statement: Write a program using TCP/UDP sockets for wired network to implement

a. Peer to Peer chat

b. Multi-user chat

Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode

Objectives:

1. To understand TCP sockets
2. To understand UDP sockets

Outcomes:

Students will be able to understand the concepts of TCP and UDP sockets and use appropriate classes and methods to implement a P2P and multiuser chat program.

H/W and S/W requirements:

64 bit Linux based OS

Wireshark Packet Analyzer Tool

Theory:

Client-Server communication

A device running a program that makes a request for services is called a client. A device which offers the requested services to one or more clients is called a server. The media for communication between the both can be a wired or a wireless network. These networking services are provided by the transport layer, which comprises two kinds of protocols - TCP (Transport Control Protocol) and UDP (User Datagram

31341

Protocol).

TCP is a connection-oriented protocol that provides a reliable flow of data between multiple devices. HTTP, FTP and Telnet make use of TCP.

UDP is a protocol that sends independent packets of data, called Datagrams, with no guarantee of arrival or sequencing. Clock server and ping make use of UDP.

Sockets and Socket-based communication

Sockets provide an interface for programming networks at the transport layer. Socket-based communication is independent of a programming language used for implementing it.
A server runs on a specific device and has a socket that is bound to a specific port. The server listens to the socket for a client o make a connection request. If the server accepts the connection, the server gets a socket bound to a different port. It needs a new socket and a different port number so that it can continue to listen to the original socket for connection requests while serving the connected client.

TCP Socket Programming in Java
The two key classes from the java.net package used in creating the server and client programs are:
1. ServerSocket:
This class implements server sockets. A server socket waits for the request to come in over the network. It performs some operation based on that request and returns a result to the requester.
Constructor used:
ServerSocket (int port) - Creates a server socket bound to a specific port
Methods used:
Socket accept( ) - Listens for a connection to be made to this socket and accepts it.

31341

.

2. Socket:

This class implemenets client sockets, i.e. an endpoint to communicate between two machines.

Constructor used:

Socket (String host, int port) - Creates a stream socket and connects it to the specified port number on the named host. Used on the client side.

Methods used:

a. InputStream getInputStream( ) - Returns an input stream for this socket

b. OutputStram getOutputStream( ) - Returns an output stream for this socket

c. void close( ) - Closes this socket

UDP Socket Programming in Java

The three key classes from the java.net package used in creating the server and client programs are:

1. DatagramPacket

This class represents a datagram packet. Datagram packets are used to implement a connectionless packet delivery service.

Constructors used:

a. DatagramPacket(byte[ ]buf, int len):

Constructs a DatagramPacket for receiving packets of length len.

b. DatagramPacket(byte [ ]buf, int len, InetAddress addr, int port):

Constructs a DatagramPacket for sending packets of length len to a specified port number on the specified host.

Methods used:

a. InetAddress getAddress( ) - Returns the IP address of the machine to which this datagram is being sent or from which it was received.

b. int getPort( ) - Returns the port number of the remote host to which the

31341

datagram is send or from which it was received.
c. int getLength( ) - Returns the length of the data sent or received

2. DatagramSocket
This class represents a socket for sending and receiving datagram packets.
Constructors used:
DatagramSocket( ) - Constructs a datagram socket and binds it to any available port on the local host machine

Methods used:
a. void send(DatagramPacket p) - Sends a datagram packet from this socket.
b. void receive(DatagramPacket p) - Receives a datagram packet from this socket.

3. MulticastSocket
The multicast datagram socket class is useful for sending and receiving IP multicast packets. A MulticastSocket is a (UDP) DatagramSocket, with additional capabilities for joining "groups" of other multicast hosts on the internet.
A multicast group is specified by a class D IP address and by a standard UDP port number. Class D IP addresses are in the range 224.0.0.0 to 239.255.255.255, inclusive. The address 224.0.0.0 is reserved and should not be used.

Constructors used:
MulticastSocket (int port) - Creates a multicast socket and binds it to a specific port.
Methods used:
a. void joinGroup(InetAddress mcastaddr) - Joins a multicast group
b. void leaveGroup(InetAddress mcastaddr) - Leave a multicast group
c. void setTimeToLive (int ttl) - Set the default time-to-live for multicast packets sent out on this MulticastSocket in order to control the scope of the multicasts.

Conclusion: We were able to successfully implement a peer-to-peer and a multi-user chat