**Assignment no:**

**Title:**  TCP and UDP programming.

**Problem Statement:**

Write a program to prepare TCP and UDP packets using header files and send the packets to destination machine in peer to peer mode. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

**Objective:**

To understand the format of TCP and UDP packets and perform programming accordingly.

**Software and Hardware requirement:**

Ubuntu 14.04 or later, Wireshark.

**Theory:**

**Networking Concepts**

The primary function of the TCP/IP is to provide a point to point communication mechanism. One process on one machine communicates with another process on another machine or within the same machine. This communication appears as two streams of data. One stream carries data from one process to the other, while the other carries data in the other direction. Each process can read the data that have been written by the other, and in normal conditions, the data received are the same, and in the same order, as when they are sent.

**Types of Network Programming**

Two general types are:

1. Connection-oriented programming.

2. Connectionless Programming.

* Connection-oriented Networking

The client and server have a communication link that is open and active from the time the application is executed until it is closed. Using Internet jargon, the Transmission control protocol os a connection oriented protocol. It is reliable connection – packets are guaranteed to arrive in the order they are sent.

* Connection-less Networking

In this type at each instance that packets are sent, they are transmitted individually. No link to the receiver is maintained after the packets arrive. The Internet equivalent is the User Datagram Protocol (UDP). Connectionless communication is faster but not reliable. Datagrams are used to implement a connectionless protocol, such as UDP.

**Client-Server Programming**

The most common model of network programming is referred to as client-server programming. The concept is simple: A client machine makes a request for information or sends a command to a server; in return, the server passes back the data or results of the command. Most often, the server only responds to clients; it does not initiate communication.

So the job of the server is to listen for a connection, and that’s performed by the special server object that we create. The job of the client is to try to make a connection to the server, and this is performed by the special client object we create. Once the connection is made, you will see that at the server and client ends, the connection is magically just turned into the IO Stream object, and from then onwards you can treat the connection as if you were reading and writing into the file.

**Socket Classes**

1. Socket
2. ServerSocket
3. DatagramSocket
4. MulticastSocket

* Socket

Socket object is the Java representation of a TCP connection. When a socket is created, a connection is opened to the specified destination.

Constructors:The Socket provides the programmer with four constructors. The address of the server may be specified as a string or an InetAddress, and the port number on the host to connect to. In each case, an optional Boolean parameter implements a connectionless socket if set to false.

Methods:The two most important methods are getInputStream() and getOutputStream(), which return stream objects that can be used to communicate through the socket. A close() method is provided to tell the underlying operating system to terminate the connection. Methods are also provided to retrieve information about the connection to the local host and remote port numbers and an integers representing the remote host.

* ServerSocket

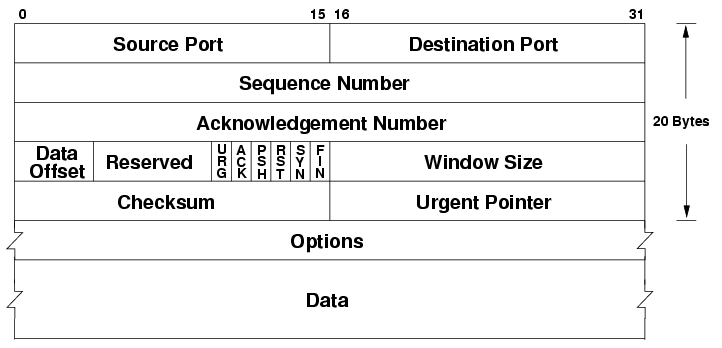
The ServerSocket represents a listening TCP connection. Once an incoming connection is requested, the ServerSocket object will return a Socket object representing the connection.

Constructors: The ServerSocket provides two constructors. Both take argument the local port number to listen for connection requests. A constructor is provided that also takes the maximum time to wait for a connection as a second argument.

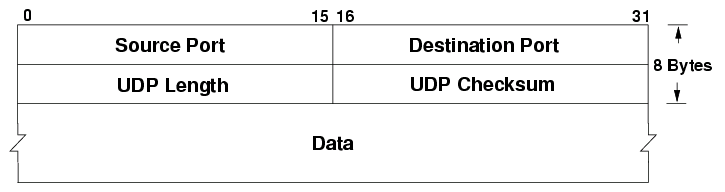
Methods: The most important method is accept(). It returns a Socket that is connected to the client. The close () method tells the operating system to stop listening for requests on the socket. Methods to retrieve the host name, the socket is listening on and the port number being listened to are also provided.

**Header Formats:**

**TCP Packet format**



**UDP Packet format**



Advantages of TCP

1. TCP always guarantees three things - your data reaches its destination, it reaches there in time and it reaches    there without duplication.

2. In TCP, since all the work is done by the operating system, so you just need to sit back and watch the show. Even the debugging is taken care of by your OS.

3. It automatically breaks up data into packets for you.

4. It is slower in functioning than UDP

Disadvantages of TCP

1. Since, all the work is being done by your OS, so if there are bugs in your OS, then you will face many problems like problems in surfing and downloading contents from the net.

2. TCP cannot be used for broadcast and multicast connections.

Advantages of UDP

1. Broadcast and multicast connections are available with UDP which is not the case with TCP.

2. It does not restrict you to connection based communication model

3. Much faster than TCP

Disadvantages of UDP

1. There are no guarantees with UDP. It is possible that a packet may not be delivered, or delivered twice, or delivered not in time.

2. You have to manually break the data into packets

**Wireshark**

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.You could think of a network packet analyzer as a measuring device used to examine what’s going on inside a network cable, just like a voltmeter is used by an electrician to examine what’s going on inside an electric cable (but at a higher level, of course).

The following are some of the many features Wireshark provides:

1. Available for UNIX and Windows.
2. Capture live packet data from a network interface.
3. Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
4. Import packets from text files containing hex dumps of packet data.
5. Display packets with very detailed protocol information.
6. Save packet data captured.
7. Export some or all packets in a number of capture file formats.

**Using Wireshark**

* Capturing Packets

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn’t necessary for now.

As soon as you click the interface’s name, you’ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled—it’s enabled by default—you’ll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the “Enable promiscuous mode on all interfaces” checkbox is activated at the bottom of this window.

Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.

* Color Coding

You’ll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.

* Sample Captures

If there’s nothing interesting on your own network to inspect, Wireshark’s wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.

* Filtering Packets

If you’re trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you’ll likely have a large amount of packets to sift through. That’s where Wireshark’s filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type “dns” and you’ll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

For more information on Wireshark’s display filtering language, read the Building display filter expressions page in the official Wireshark documentation.

Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

You’ll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.

Close the window and you’ll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

* Inspecting Packets

Click a packet to select it and you can dig down to view its details.

You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.

Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

**Conclusion –**

Hence, we have successfully implemented TCP and UDP header program using Java.