

**School of Engineering & Design**

**Electronic and Computer Engineering**

**Diego Humberto Puentes Caceres**

**Student ID 0829734**

**MSc Data Communication Systems (EE5506)**

**Evaluating UDP Performance in File Transfer**

**Objectives - Aims - Scope.**

* **To obtain a deep understanding of Java Socket programming using UDP.**
* **To establish UDP Client/Server connection and file transfer by using Java Socket programming.**
* **To experiment and analyse how UDP works when transferring data using different packet sizes.**
* **To analyse and understand the main features involved when using Java programming and the Graphical User Interface (GUI).**

**Table of Contents**

# 1.0 Introduction

# 2.0 Task 1

# 3.0 Task 2

# 4.0 Conclusions

# 5.0 References

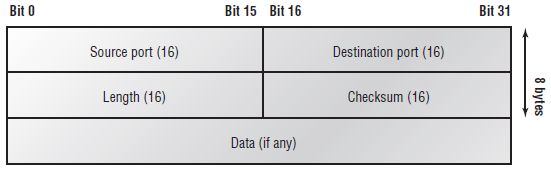
1. **Introduction**

**User Datagram Protocol (UDP)**

UDP is a protocol which produces the same result that TCP but using fewer features and less network resources. Therefore, UDP does not utilize much bandwidth on a network and it does an excellent job when transporting unreliable data.

Principally, when running large networks, it sends intermittent messages, flow of updates and alerts. Consequently, UDP will definitely the best option to take. However, UDP is a connectionless protocol at the same time it does not worry about in which order the data arrives at the destination, it does not contact the destination side before sending data to it. In other words, we use UDP for faster transfers and TCP for reliability.

The figure below shows us the simplicity of a UDP segment. There are not sequence number, window size or acknowledge number.



**Figure 1.4 UDP format, [1]**

**Source port** Port of the application in the end host.

**Destination port** Port number of the application in the destination host.

**Length** Size of UDP header and UDP data.

**Checksum** Checksum of the UDP header and UDP data.

**Data** Amount of information to be transmitted in addition with the upper layer headers.

**Advantages and disadvantages between TCP and UDP.**

|  |  |
| --- | --- |
| TCP | UDP |
| Reliable | Unreliable |
| Sequenced | Unsequenced |
| Connection-oriented | Connectionless |
| Virtual circuit | Low transparency |
| Acknowledgments | No acknowledgment |
| Flow control | No flow control |

**Table 1.0 Advantages and disadvantages TCP vs. UDP**

**UDP Socket performance**

UDP is an uncomplicated, unreliable datagram protocol used to work with the SOCK\_DGRAM socket kind used in the internet protocol field. UDP sockets are connectionless and we can set a connection between two computers in order to send and receive data. Moreover, UDP sockets do not have the global uniqueness property, in other words, there are no connection-oriented and the UDP socket refers to just the local computer and only require a receive buffer.

UDP and TCP address formats are similar. In fact, UDP require a port identifier plus the normal internet address composition. UDP port cannot be connected to a TCP port because the UDP port domain is different or separate from the TCP port. The default send buffer size for UDP sockets is 65535 bytes and for the receiver part is 2147483647 bytes[2].

A datagram socket is the sending or receiving point for a packet or data delivery. All packets sent or received on a datagram socket are individually addressed and routed. When we send multiple packets from one computer to another may be these data can be routed in a different way, and may arrive in any order. UDP broadcasts sends are commonly enabled on a DatagramSocket. In other to obtain broadcast packets a DatagramSocket should be bound to the wildcard address or to a more specific address.

Example

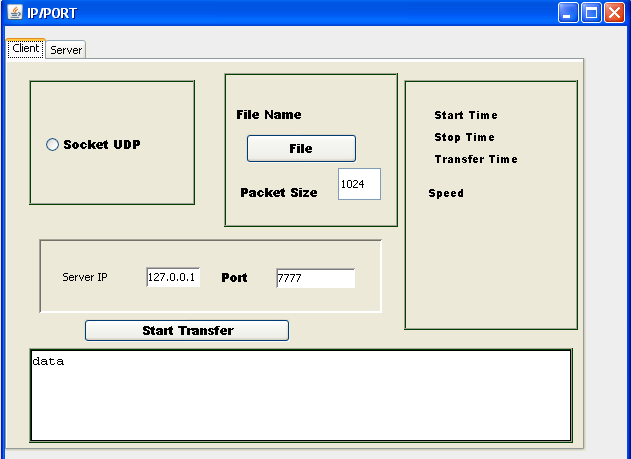
**DatagramSocket s = new DatagramSocket(null); s.bind(new InetSocketAddress(8888));**

**Which is equivalent to: DatagramSocket s = new DatagramSocket(8888); Both cases will create a DatagramSocket able to receive broadcasts on UDP port 8888.**

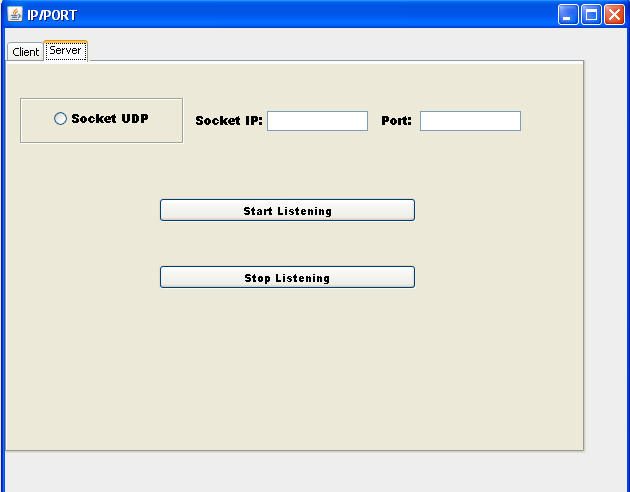
In addition to the UDP socket class, there is a UDP broadcast class and it can be set to send messages to an entire subnet rather than to an individual peer socket. Moreover, UDP sockets are used for creating real-time media streaming protocols and full duplex messaging services by using a pair of sockets together; one of them to send and the other one to receive data.

**2.0 Task 1: A Graphical User Interface (GUI) Design**

The UDP based Client/Server codes we extended for file transfer using different packet sizes. A GUI interface for Client and Server is shown:



**Figure 2.0 Client Interface**



**Figure 2.1 Server Interface**

I created a program using Java software in order to obtain the Client and Server interfaces and to send any request or messages for UDP server by the UDP client[1]. For this we need the destination IP address and destination port number. If we know both, then we send messages to UDP server. The sending process has been defined as follows:

**Description of program:**

This is a layout with the help of this program, which has destination IP address, destination port number. The IP address specify local host and destination port number takes a specific port number where you want to send the file. All files chosen in the main area and finally click the send Transfer button, files are sent to the specific destination.

There some important instruction we need to analyse in order to understand the behaviour of the Java code:

**getByName(Stringsocket):**This method identify the IP address of the host machine (localhost) which is entered in string of this method.

**getBytes():**This method returns a sequence of bytes of the string.

**DatagramPacket**(byte[] message, int message\_length, InetAddress address, int pnum):  
This is the constructor of DatagramPacket class. The DatagramPacket class extends the Object and shows the datagam packet. It creates a datagram packet for sending files on the specified host machine with the help of specified port number. The length of datagram packet depends upon file size.

**socket.send(DatagramPacketpacket):**It can be used to send the datagarm packet to specified socket.

**DatagramSocket(int port):**DatagramSocket is the constructor of **DatagramSocket** class. This class extends **Object**. It creates the datagram socket and takes only integer  [port number](http://www.roseindia.net/java/example/java/net/udp/udp-server.shtml" \t "_top)

[http://kona.kontera.com/javascript/lib/imgs/grey_loader.gif](http://www.roseindia.net/java/example/java/net/udp/udp-server.shtml" \t "_top)

**DatagramPacket(byte[] buffer, int buffer\_length):**This constructor provided by the **DatagramPacket** class. This class extends the Object and imports the *java.net.\**. It constructs a DatagramPacket to the receiving packet length. It takes the buffer size and buffer length. Where the connectionless services are provided by the user then use it.

**receive(DatagramPacket packet):**The datagram packet received by this method from the specified socket.

**getAddress():**Above method returns the IP address of the object of **InetAddress** class, where the datagram packet can be send or receive.

**getPort():**This method returns the port number of the host machine. Where the datagram packets can be send or receive.

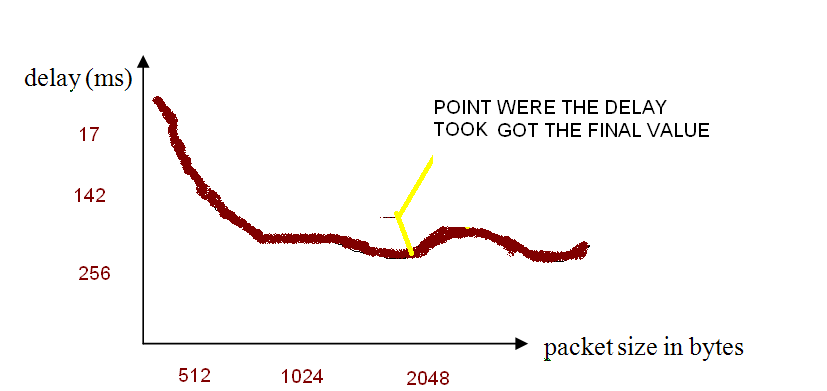
**socket.close():**This method closes the datagram socket that means you cannot be received any messages.

When you will run the program then appears a graphical interface on the screen. Which has destination IP ([Internet protocol](http://www.roseindia.net/java/example/java/net/udp/udp-client-send-receive.shtml)) address (localhost),[3] destination [port number](http://www.roseindia.net/java/example/java/net/udp/udp-client-send-receive.shtml) (PORT), and Transfer file. All information or messages are written by client after that you clicked on the Start Transfer button then files to be send into the UDP server and it also sends a message to UDP client. Therefore UDP client can identify that your sending information are received or not by the UDP server.

**3.0 Task 2: Evaluating UDP Performance**

I plotted a figure to show UDP performance in data transfer using different sizes of packets.

The Client code and Server code run on two machines connected by a network. In order to explain how packet sizes could affect delay.



**Figure 3.0 UDP performances in data transfer**

The graph show us the UDP performance when changing the packet sizes (521K, 1024K and 2048K) and I noticed a high decrement in the delay when we change de size of the file and also a new behaviour after 1500K where UDP file delay started to increase again.

1. **Conclusions**

UDP is very useful in some circumstances, especially in speed and network management. Its speed is faster than TCP protocol. UDP transports the data periodically and with no guarantee. So the updated data can be sending after a certain period, which not only recovers the lost data but also it updates network. While in the TCP it is very difficult to recover the lost data at the congestion stage. It is also not suitable to carry network management due to its transmitting methodology.

UDP sockets do not have the global uniqueness property, in other words, there are no connection-oriented and the UDP socket refers to just the local computer and only require a receive buffer. UDP sockets are used for creating real-time media streaming protocols and full duplex messaging services by using a pair of sockets together; one of them to send and the other one to receive data.

1. **References**

[1] http://www.networld.com/documentation/docs/manuals/TCPIP.pdf

[2].http://java.sun.com/j2se/1.4.2/docs/api/java/net/DatagramSocket.html

[3]. http://www.cisco.ncnu.edu.tw/~yshan/intro\_net\_code.pdf

.