# UDP Client

Coursework Assignment IWT

M G S Dassanayake (12712980)

# Table of Contents

User Datagram Protocol ..... 2

PHP Socket Programing ..... 3

UDP Client Application ..... 4

Bibliography: ..... 6

Test Results Transcript 1 ..... 7

Test Results Transcript 2 ..... 8

Test Results Transcript 3 ..... 10

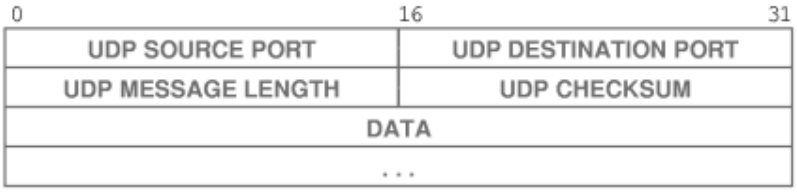
PHP Implementations (Coding) ..... 11

# User Datagram Protocol

User Datagram protocol is an unreliableand a connectionless protocol. It does not guarantee reliability or the correct sequencing of data. Datagrams may go missing without notice, or arrive in a different order from the one in which they were sent

**UDP Datagram Format**

UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection. UDP transmits *segments* consisting of an 8-byte header followed by the payload. The format is shown below




**UDP Packet Structure**

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits). The use of two of those is optional in IPv4 (pink background in table). In IPv6 only the source port is optional (see below).

**Source port number**

This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.

**Destination port number**

This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.

**Length**

A field that specifies the length in bytes of the entire datagram: header and data. The minimum length is 8 bytes since that's the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes (65,535 – 8 byte UDP header – 20 byte IP header).

**Checksum**

The checksum field is used for error-checking of the header *and* data. If no checksum is generated by the transmitter, the field uses the value all-zeros. This field is not optional for IPv6.

# PHP Socket Programing

PHP comes with a comprehensive socket programming API which has all the functions you need to create socket based client – server application.

Functions I have used in my program

| | |
|---|---|
| socket_create() | Create a socket (endpoint for communication) |
| socket_connect() | Initiates a connection on a socket |
| socket_sendto() | Sends a message to a socket, whether it is connected or not |
| socket_recvfrom() | Receives data from a socket whether or not it is connection-oriented |
| socket_close($sock) | Closes a socket resource |

# UDP Client Application

To run the UDP application the user need to execute the client.php file with two arguments in the command prompt.

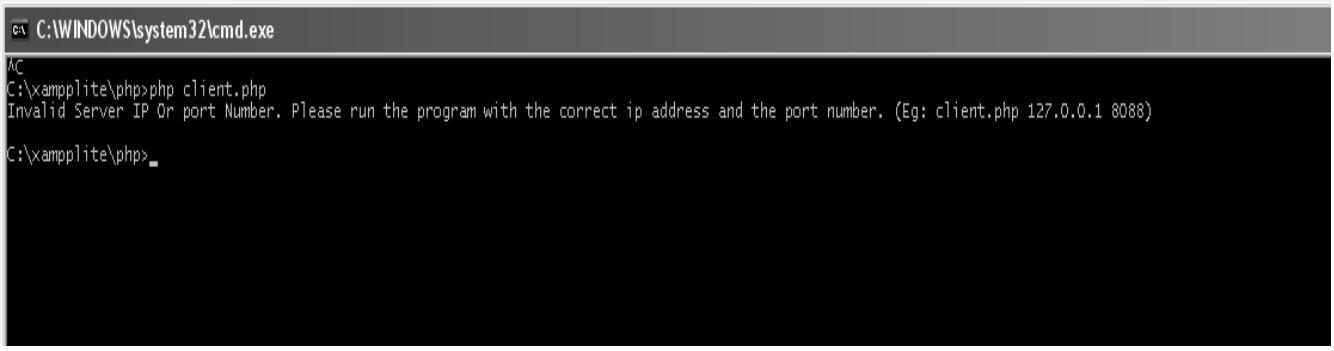
Argument 1: Server IP

Argumnet 2: Port Number

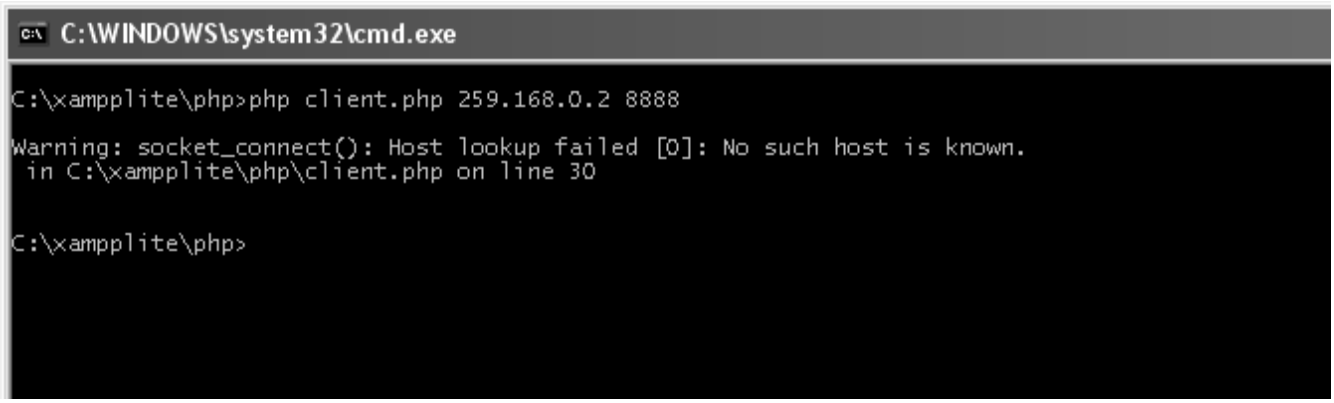
$php>php client.php <server_ip>  <server_port>

(Eg: $php>php client.php 192.168.0.2 8888)


If the user run the php file without any arguments the program will display an error message.




If the IP address is not valid it will display the following error message



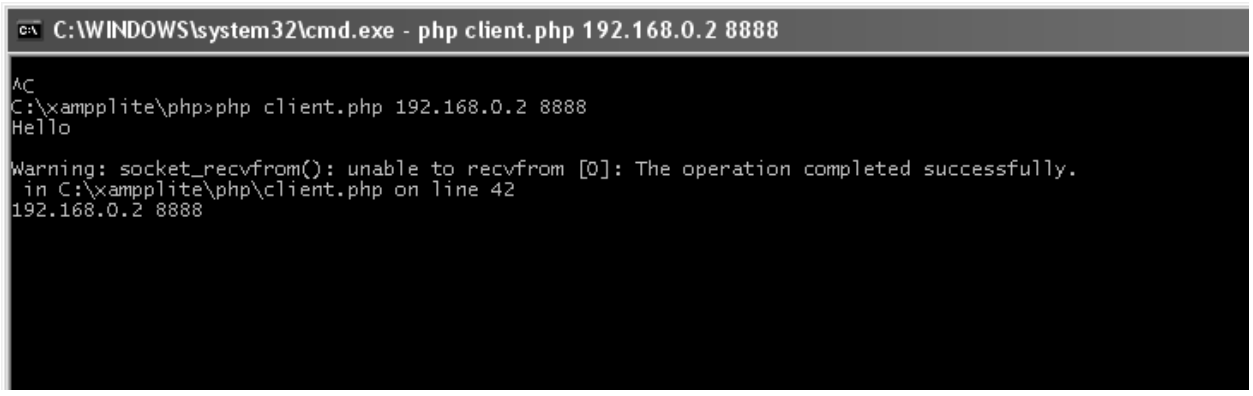

On my system I have installed xampplite which is a software bundle which includes PHP 5.3.1, Apache, MySQL and many other services.

For the development and for testing purposes I have used the PHP interpreter came with the xampplite budle.

Once you run the application it will wait for a keyboard input. (As shown in the above figure)

If the server is running on the given IP and the Port number it will display the input text with the server IP and the port number from the datagram received back from the server. (Test results are provided in the Appendix)

If the server is not running or if the program is not able to connect to the server it will display the following warning message




To exit the application type the word ‘quit’ instead of the message.

# Bibliography:


http://php.net/manual/en/function.socket-recvfrom.php

http://penguin.dcs.bbk.ac.uk/academic/networks/transport-layer/udp/index.php

http://en.wikipedia.org/wiki/User_Datagram_Protocol

http://www.php.net/manual/en/ref.sockets.php

http://www.apachefriends.org/en/xampp-windows.html

## *Test Results Transcript 1*

Name: James Pearson-Kirk (07th January 2010)

Server IPAddress: 86.14.117.160

Server Port Number: 2555

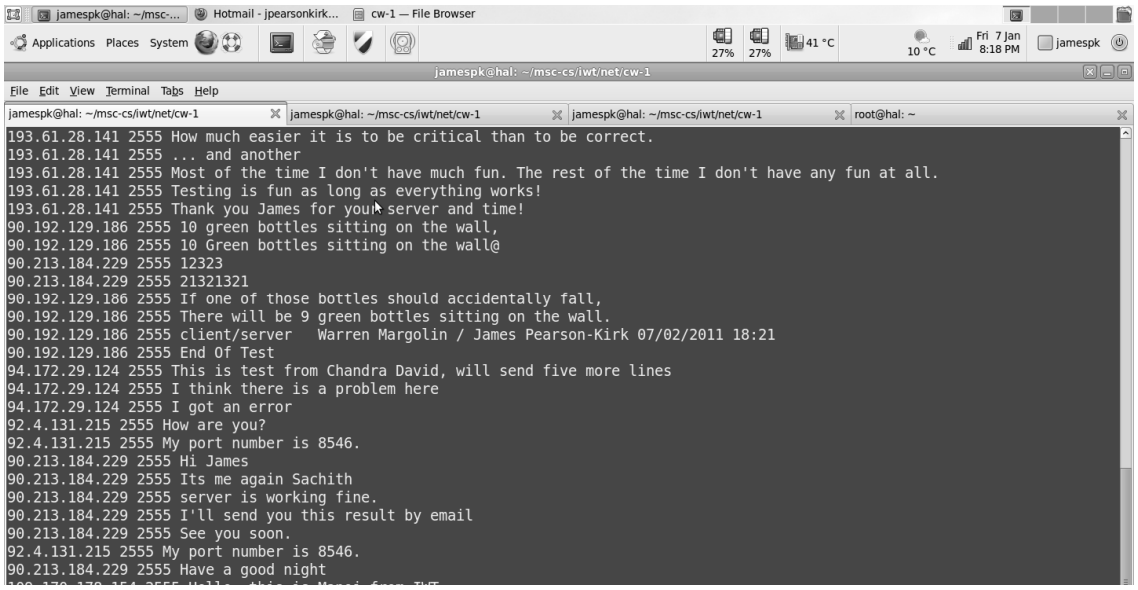
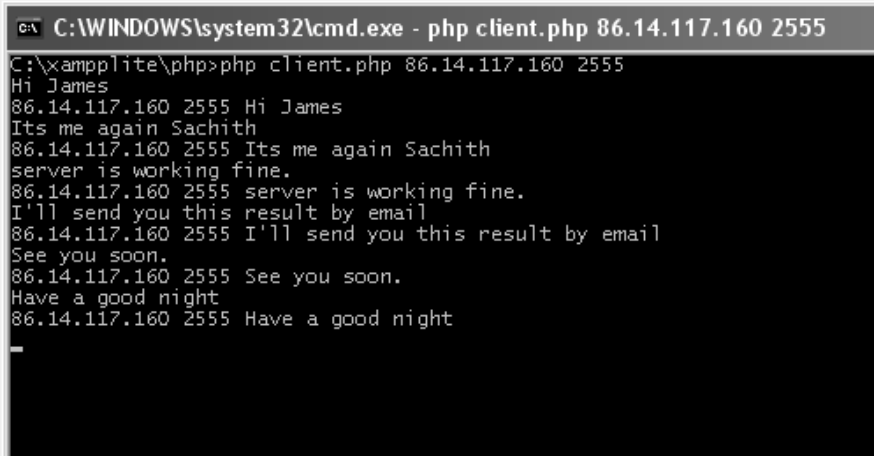



Figure 1: UDP Server

Client IP Address: 90.213.184.229

Client Port : 2555 (This was number received from the server as the port)



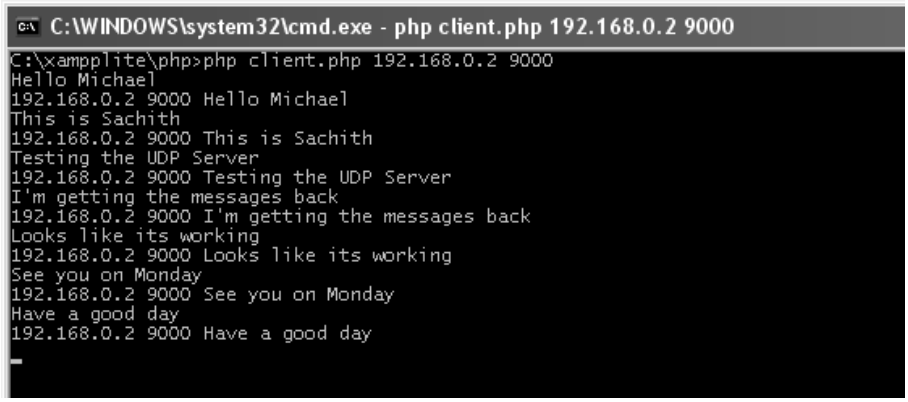

UDP Client

## *Test Results Transcript 2*

Name: Michael Sauter (08th January 2010)

Server IPAddress: 192.168.0.2

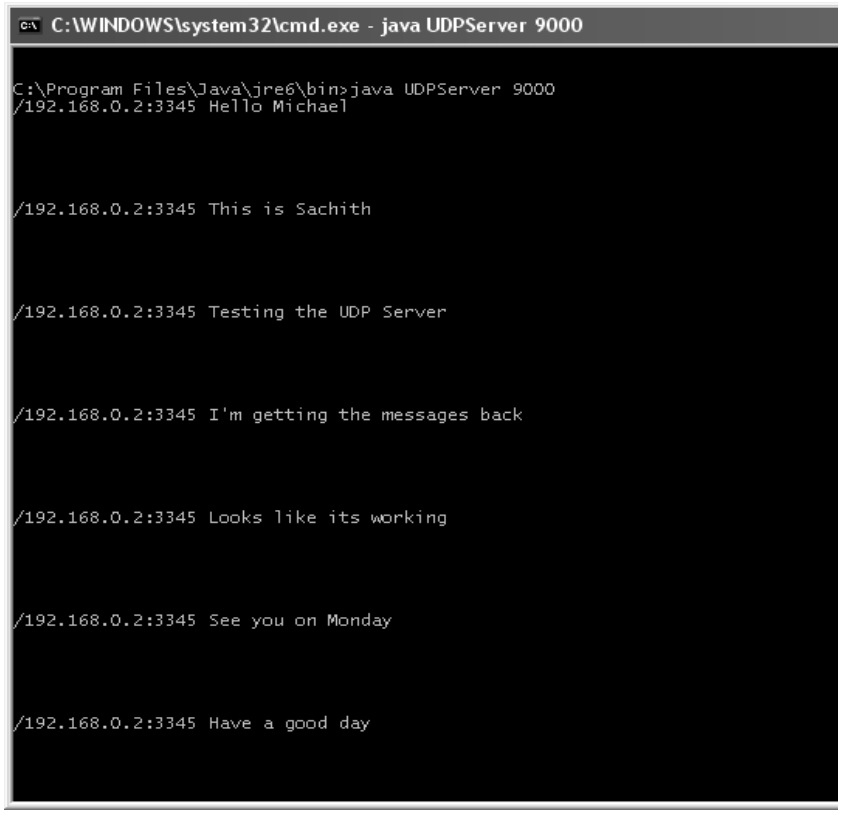
Server Port Number: 9000




UDP Client

Client IP Address: 192.168.0.2

Client Port: 3345

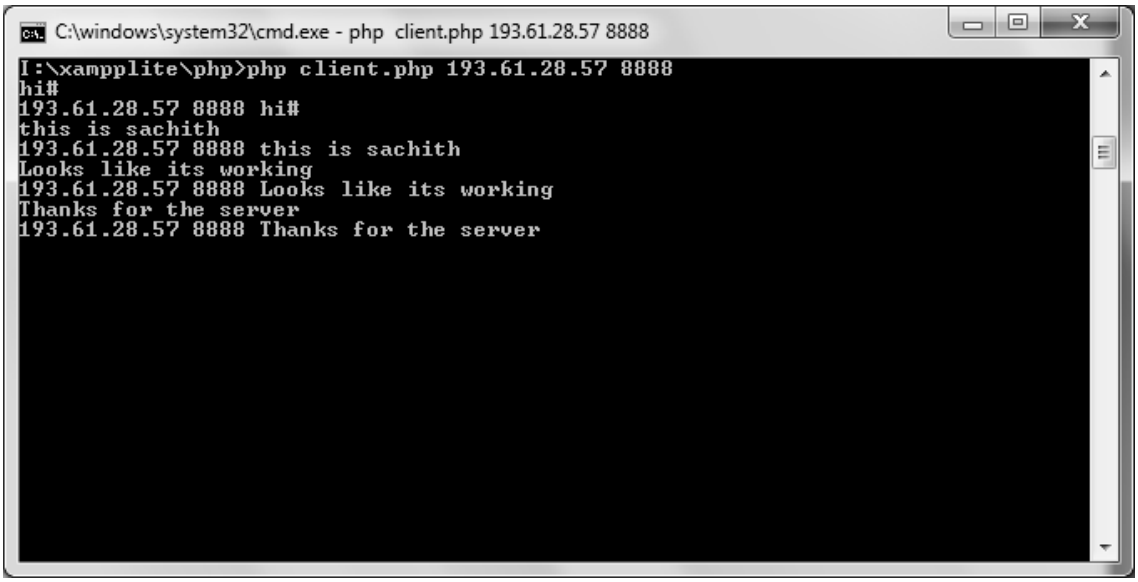



UDP Server

The same server was also tested on a lab computer and the results as follows.

Server IPAddress: 193.61.28.57 (10th January 2010)

Server Port Number: 8888



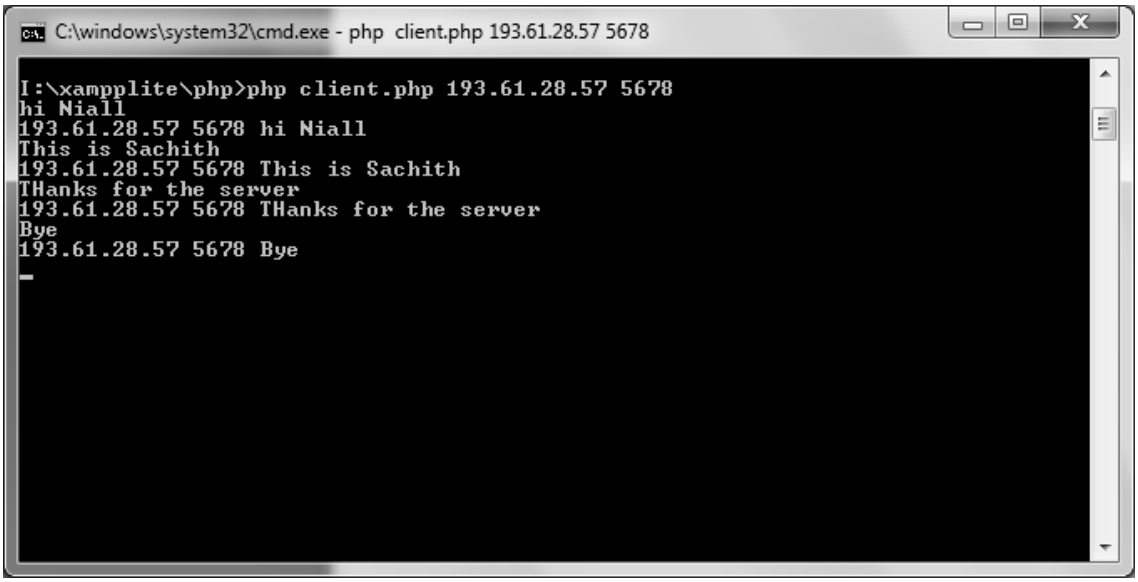

UDP Client

## *Test Results Transcript 3*

Name: Niall Gallagher (12th January 2010)

Server IPAddress: 193.61.28.57

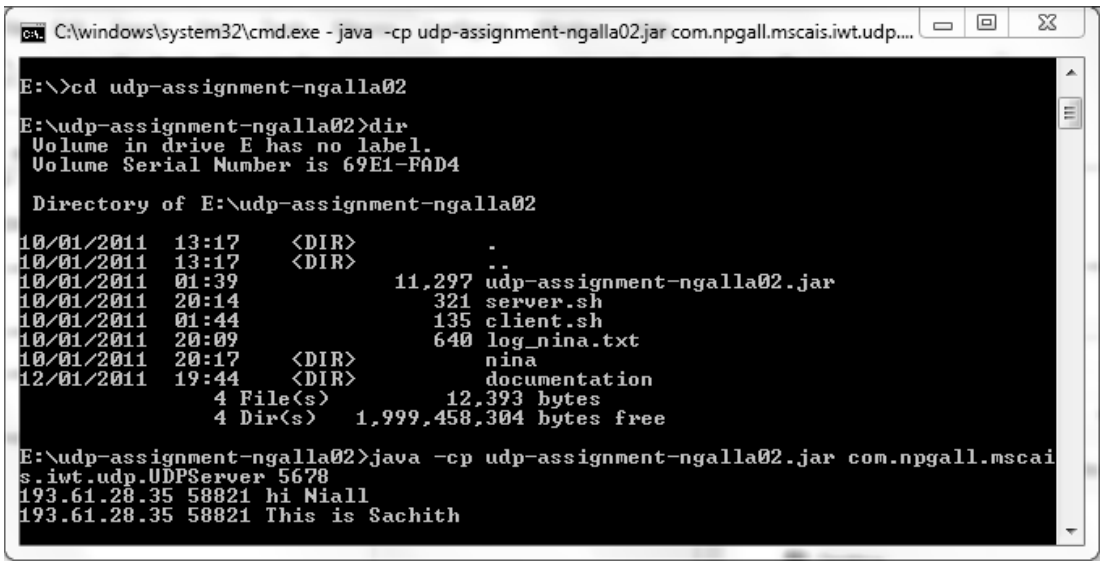
Server Port Number: 5678




UDP Client

Client IP Address: 193.61.28.35

Port Number: 58821




UDP Server

## *PHP Implementations (Coding)*

```
<?php
/*  Coursework Assignment IWT
        UDP Client
        M G S Dassanayake (12712980)
        References: Example #1 http://php.net/manual/en/function.socket-recvfrom.php
*/
$srvIP = $argv[1];   // Remote Server IP address (Argument 1)
$srvPort = $argv[2]; // Remote Server Port (Argument 2)
if ($argv[1] == "" || $argv[2] == "")
{
        echo "Invalid Server IP Or port Number. Please run the program with the correct ip address and the port number. (Eg: client.php 127.0.0.1 8088)";
}
else
{
        //Creates the UDP Socket
        $sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
        if(!$sock)
        {
          echo 'Socket_create failed: '.socket_strerror(socket_last_error())."\n";
        }
        else
        {
                // Socket Connect
                $result = socket_connect($sock, $srvIP, $srvPort);
                if($result)
                {
```

```
                        do{
                                // Keyboard Input
                                $msg = trim(fgets(STDIN));
                                // Check if the user wants to close the application
                                if ($msg!= "quit")
                                {
                                        // Length of the input
                                        $len = strlen($msg);
                                        // Sends the input message to the server
                                        socket_sendto($sock, $msg, $len, 0, $srvIP, $srvPort);

                                        // Receives the message from the server
                                        socket_recvfrom($sock, $buf, 2048, 0, $srvIP, $srvPort);
                                        //Shows the message received from the server including the ip address and
the port
                                        echo $srvIP . " " . $srvPort. " " . trim($buf) . PHP_EOL;

                                        //Clears the buffer
                                        $buf = "";
                                }
                        }while($msg!= "quit"); // Exit the application when the user type 'quit'
                }

        }
        //Close the socket
        socket_close($sock);
}

?>
```