CHAPTER 1

INTRODUCTION

1.1 BRIEF INTRODUCTION :

In today's era of the internet majority of the communications happens through the client-server architecture model. All online transactions such as browsing the web, shopping on e-commerce portals, online banking transactions, etc are made possible through communication between the client i.e. the user of a service and the server. This process of communication is not as simple as explained above.

Clients and servers exchange messages in a request–response messaging pattern. The client initiates a request to the server and the server returns a response. This exchange of messages is an example of inter-process communication. To start the actual communication between the client and server, first a connection has to be established between the client and server through an interface called socket. Once the connection is set up, the client and server exchange messages through this socket. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol.

The exchange of messages between the client and server is in the form of data packets . Bytes are grouped together to form a data packet. Apart from the actual message the data packets also contain additional information such as source and destination address, protocol and its version, error control information, etc.

1.2 MOTIVATION :

Simulation of any process helps in better understanding of it. This project will demonstrate the internal working of a client-server architecture model. It would help in visualizing how connection between a client and server is established. This system can be used for educational purposes to teach and explain the working of client and server and network programming.

1.3 SCOPE :

The existing system of displaying a simulated system of client server simulations has always been flawed because of its inability to display a simulated GUI. It has always been unable to convey a convincing display about the packet transfer and the receiving party and also to know where the packets and data is being sent to the server.

Our system covers the existing in the way that only the necessary and essential transfers are noted and also the actual details of the packets being sent are noted. The details about what is in the packets is not noted down and only the simulations of the working packets is clearly shown. This will help a person understand what is happening in a network without understanding the technical details in the network. This system can be used for educational purposes to teach and explain the working of client and server and network programming.

1.4 STATEMENT OF PROBLEM :

The aim of the simulation is to provide a graphical interface to a user to show how exactly client server simulation in a network happens. This should allow the user to understand how exactly communication happens between client server in a network. So the user should be able see how the commands are sent from client to server and how the responses happen. The user should understand how the packets in a network travel from the client to the server. The various functions such as retrieve, store and list should also be visualised in real time.

1.5 LIMITATION OF PROBLEM:

Our system provides insight into data being sent across client and server and how exactly communication happens. While the packet transfer to the different components of the client and server is shown, this does not show what exactly is contained inside the packets and how the packets are routed. It does not explain how the data is encapsulated inside a data packet or what exactly the commands are and the internet protocol used is also left unexplained for now.

CHAPTER 2

LITERATURE SURVEY

2.1 COMPUTER GRAPHICS :

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results. Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

2.2 OPENGL INTERFACE :

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.

2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.

3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

2.3 OPENGL OVERVIEW :

- OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. It is streamlined. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- It is a state machine. At any moment during the execution of a program there is a current model transformation.
- It is a rendering pipeline. The rendering pipeline consists of the following steps:
    - Defines objects mathematically.
    - Arranges objects in space relative to a viewpoint.
    - Calculates the color of the objects.
    - Rasterizes the objects.

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

OpenGL (open graphics library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. OpenGL was developed by silicon graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games.

OpenGL serves two main purpose:

- To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
- To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full OpenGL, feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- Rasterized points, lines and polygons are basic primitives.
- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping, Alpha, Blending.

2.4 CLIENT – SERVER ARCHITECTURE :

Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an application programming interface (API). The API is an abstraction layer for accessing a service. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

A server may receive requests from many distinct clients in a short period of time. A computer can only perform a limited number of tasks at any moment, and relies on a scheduling system to prioritize incoming requests from clients to accommodate them. To prevent abuse and maximize availability, server software may limit the availability to clients. Denial of service attacks are designed to exploit a server's obligation to process requests by overloading it with excessive request rates.

The client–server model does not dictate that server-hosts must have more resources than client-hosts. Rather, it enables any general-purpose computer to extend its capabilities by using the shared resources of other hosts. Centralized computing, however, specifically allocates a large amount of resources to a small number of computers. The more computation is offloaded from client-hosts to the central computers, the simpler the client-hosts can be.

CHAPTER 3

REQUIREMENTS SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS :

Functional requirements are the necessary requirements for functioning of a software or a system. Functional requirements required for this system are –

1. **Processor :** Intel Core i3 or higher.
2. **Memory :** 1GB or more.
3. **Operating System :** Windows / Linux / MacOS X.
4. **IDE :** Code Blocks.
5. **Compiler :** GNU / GCC Compiler.
6. **Graphics Library :** OpenGL.

Various methods such as **establish_connecton, send_data, disconnect_connection,** etc are used for the implementation of the system.

3.2 NON – FUNCTIONAL REQUIREMENTS :

Non – Functional requirements support the functioning of the system. They have a check on the entire system as a whole. Non – functional requirements of the system are -

1. **Reliability :** How reliable is the system to deliver the system its service.
2. **Safety :** The ability of the system to perform without failure.
3. **Availability :** The ability of the system to render its services at all times whenever requested by the user.
4. **Dependability :** This is a property of a software / system that establishes a confidence about the system to the user.
5. **Security :** The system is secure such that no outside attacker can intercept the system.

CHAPTER 4

SYSTEM ANALYSIS

4.1 CLIENT SERVER COMMUNICATION

In general, a service is an abstraction of computer resources and a client does not have to be concerned with how the server performs while fulfilling the request and delivering the response. The client only has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service.

Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an application programming interface (API).

The API is an abstraction layer for accessing a service. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

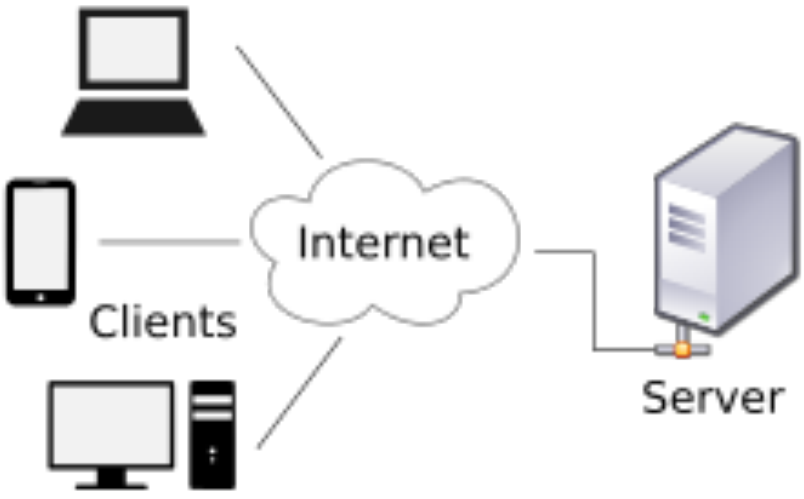


Fig 1 – Client Server Model

When a bank customer accesses online banking services with a web browser (the client), the client initiates a request to the bank's web server. The customer's login credentials may be stored in a database, and the web server accesses the database server as a client. An application server interprets the returned data by applying the bank's business logic, and provides the output to the web server. Finally, the web server returns the result to the client web browser for display.

In each step of this sequence of client–server message exchanges, a computer processes a request and returns data. This is the request-response messaging pattern. When all the requests are met, the sequence is complete and the web browser presents the data to the customer.

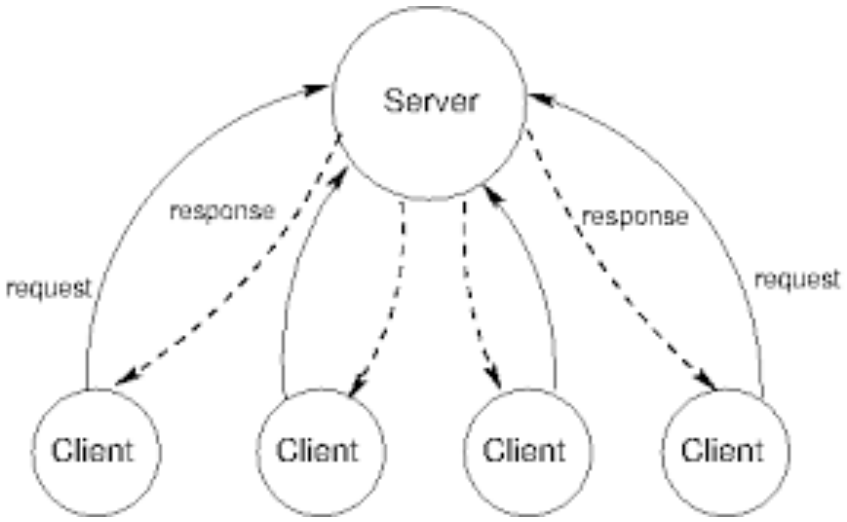
This example illustrates a design pattern applicable to the client–server model: separation of concerns.

The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

CHAPTER 5

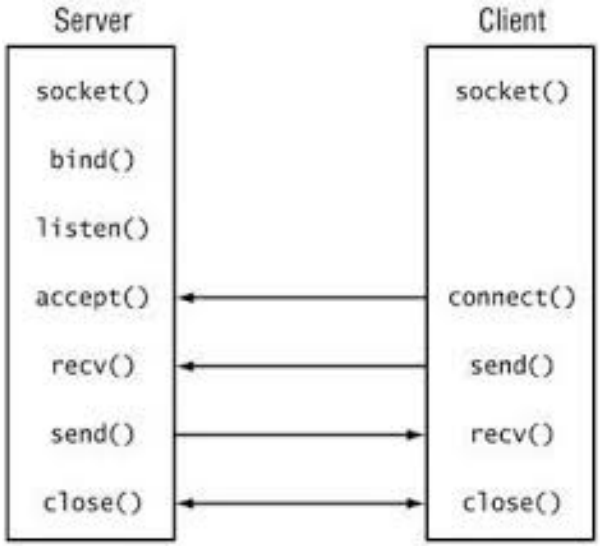
SYSTEM DESIGN

5.1 ARCHITECTURE DESIGN :



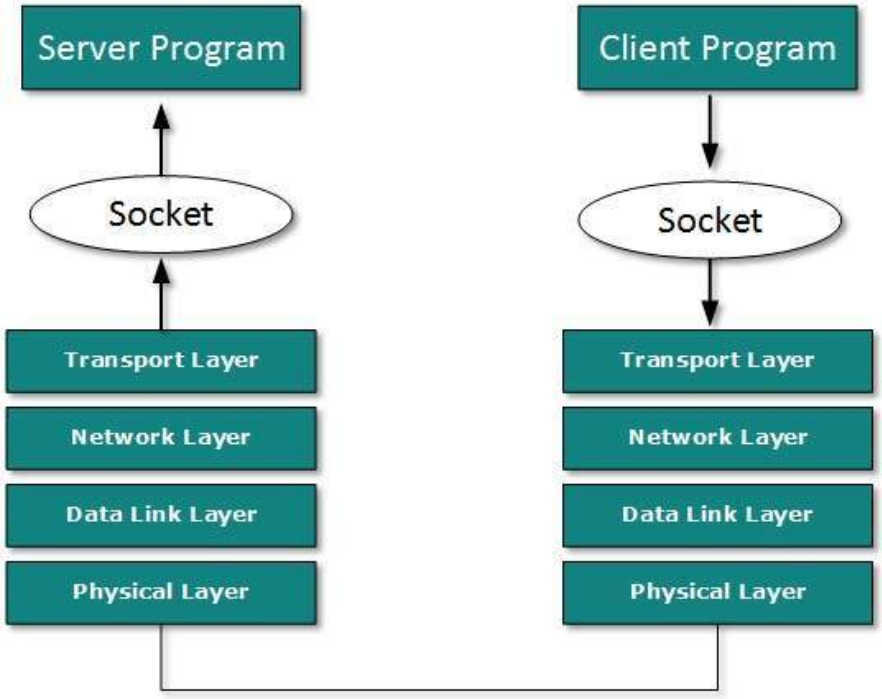
The system consists of a client and server which send commands over a network and a resulting handshaking occurs. After a handshake there is an exchange of packets that occurs.

5.2 COMPONENT DESIGN :



The client and server consist of several functions before handshaking, these functions determine the connections and the commands sent between the client and server.

5.3 BEHAVIOURAL DESIGN :



The client server typically follows the TCP/IP model where the socket is above the transport layer in both client and server. This helps in sending commands over a network.

CHAPTER 6

IMPLEMENTATION

This project is implemented using C++ as the programming language.

The graphics library used is OpenGL.

Integrated Development Environment (IDE) used is Code Blocks (Version 13.12).

OpenGL :

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

Code Blocks :

Code Blocks is a free, open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the Graphical User Interface(GUI) toolkit. Using a plugin architecture, its capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards C, C++ and Fotran.

Client Server Simulation :

START -

The client sends a command to the server, the server in turn responds to the client with a response, i.e, the acknowledgement. This means that a three-way handshake has taken place between the client and server. This only tells us that client and server know and authenticate each other.Now the client send the data or files to the server in the form of data packets through the network.

RETRIVE –

The client sends a retrieve command (RETR) to the server to retrieve or fetch a data or file it wants. The server responds back to the client by sending the data packet requested by it.

STORE –

When the client wants to store some data in the server it sends STR command to the server. This lets the server know that the data it receives has to be saved. Then the client sends the data to be stored as a data packet to the server, the server on receiving it saves it as it knows it has to be saved.

LIST –

When the client wants to know to information about its data stored in the server, it sends a LIST command to the server. The server responds with the list data packets of the client that it has been stored.

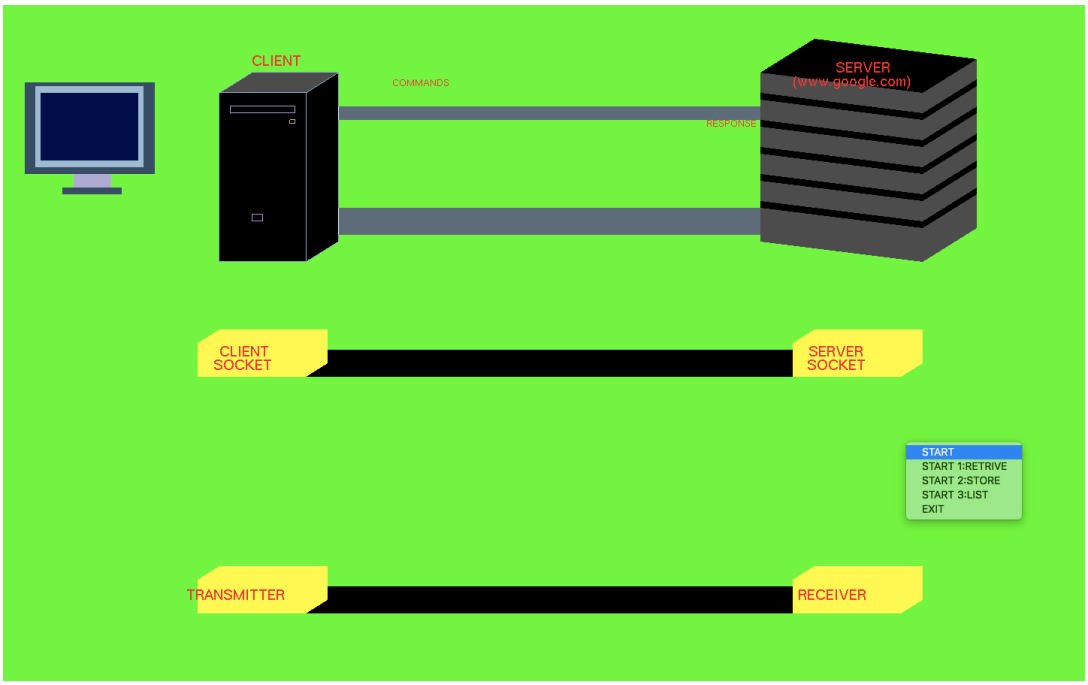
EXIT –

This exits and closes the client – server simulation running.

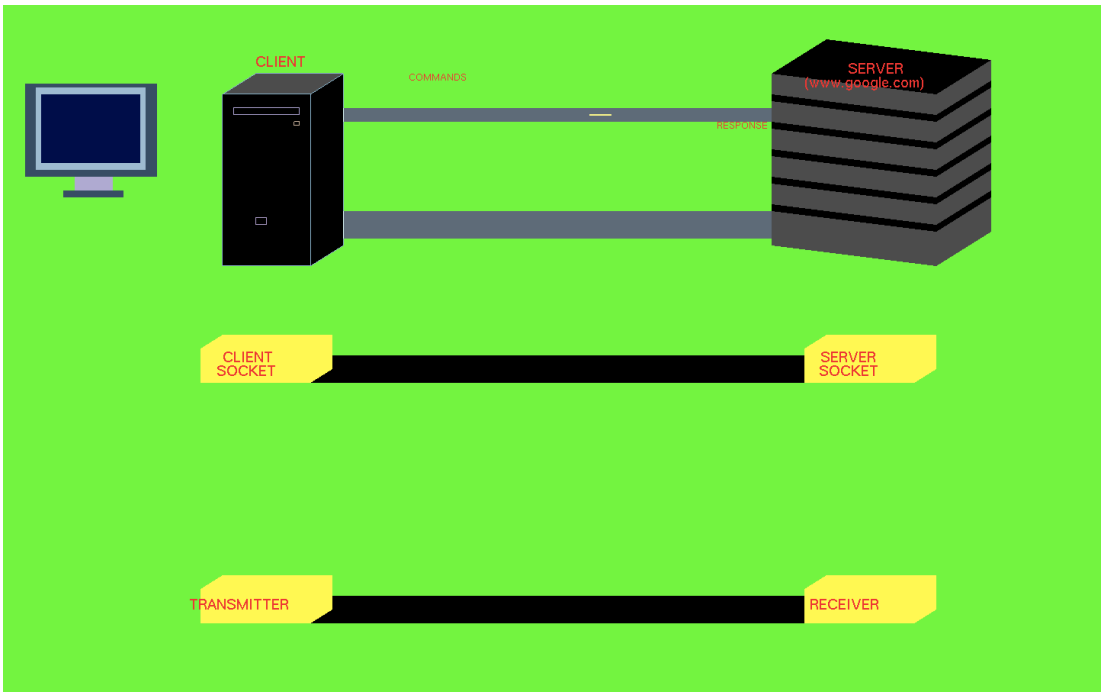
CHAPTER 7

INTERPRETATION OF RESULTS

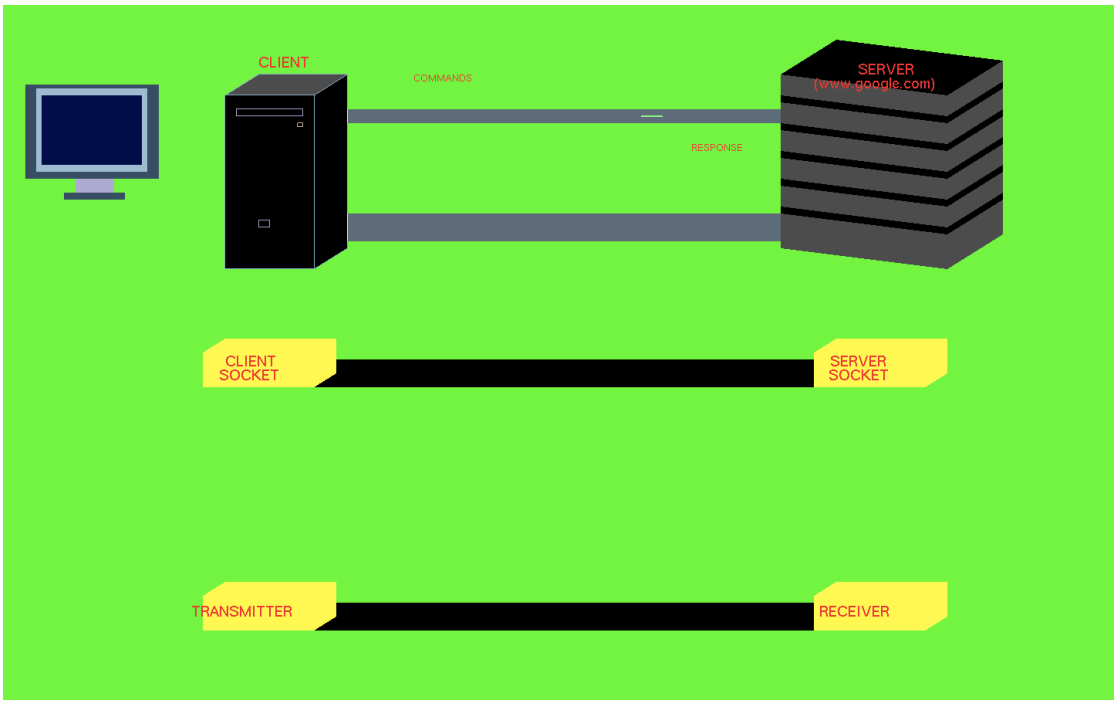
SCREENSHOTS :



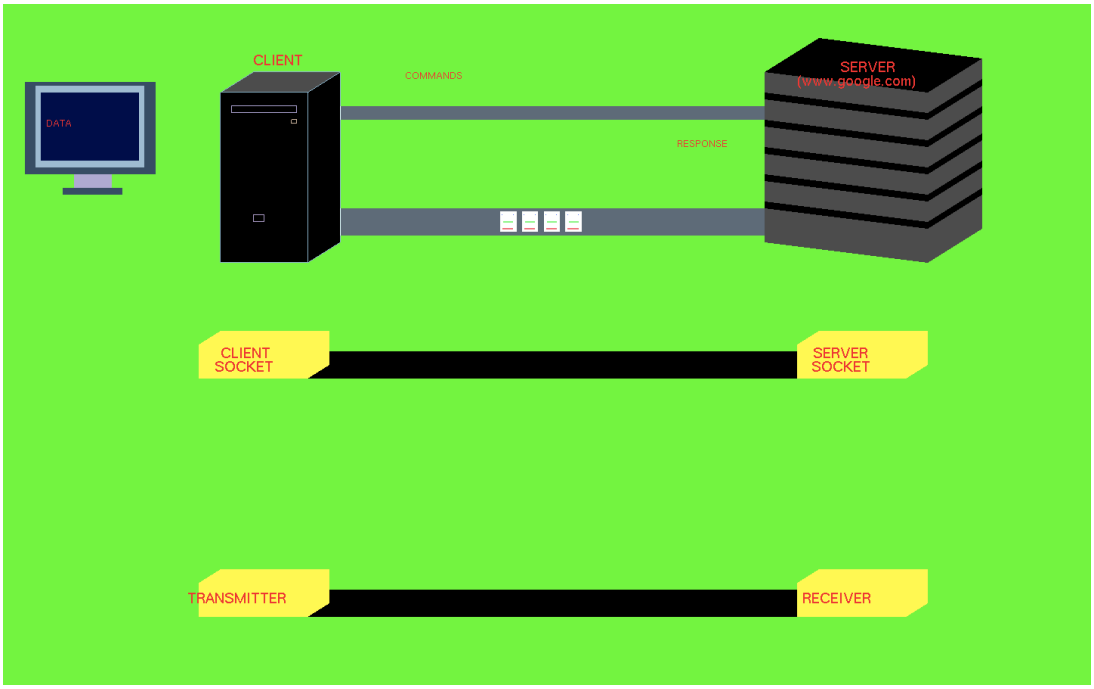
Idle status of client and server architecture.



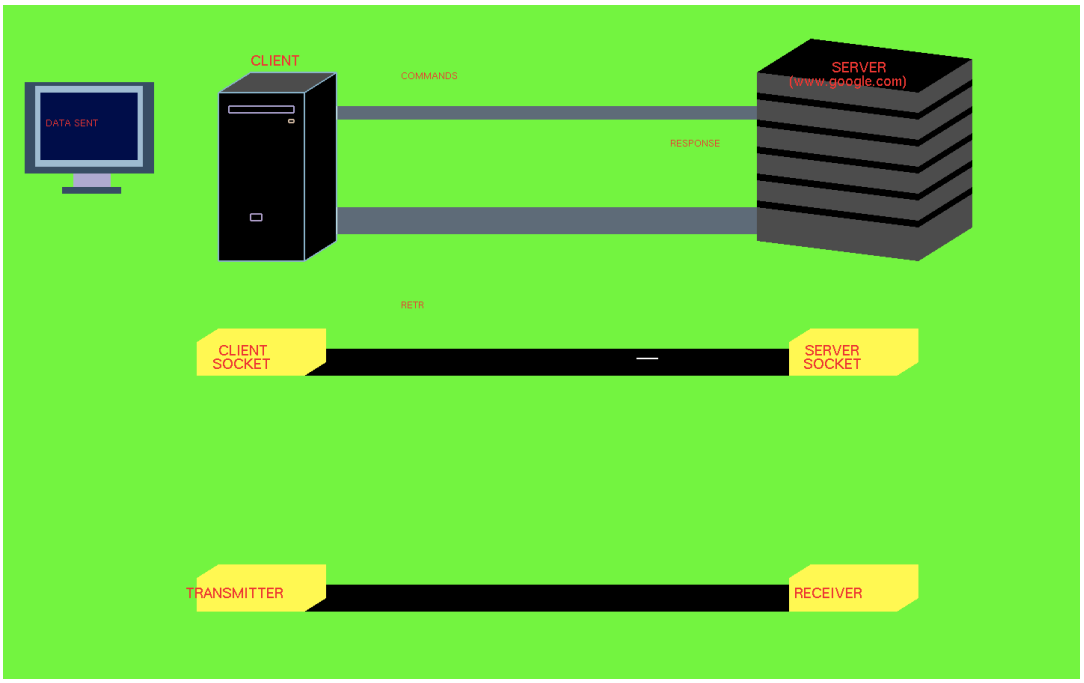
Client sends command to server for authentication and three - way handshake



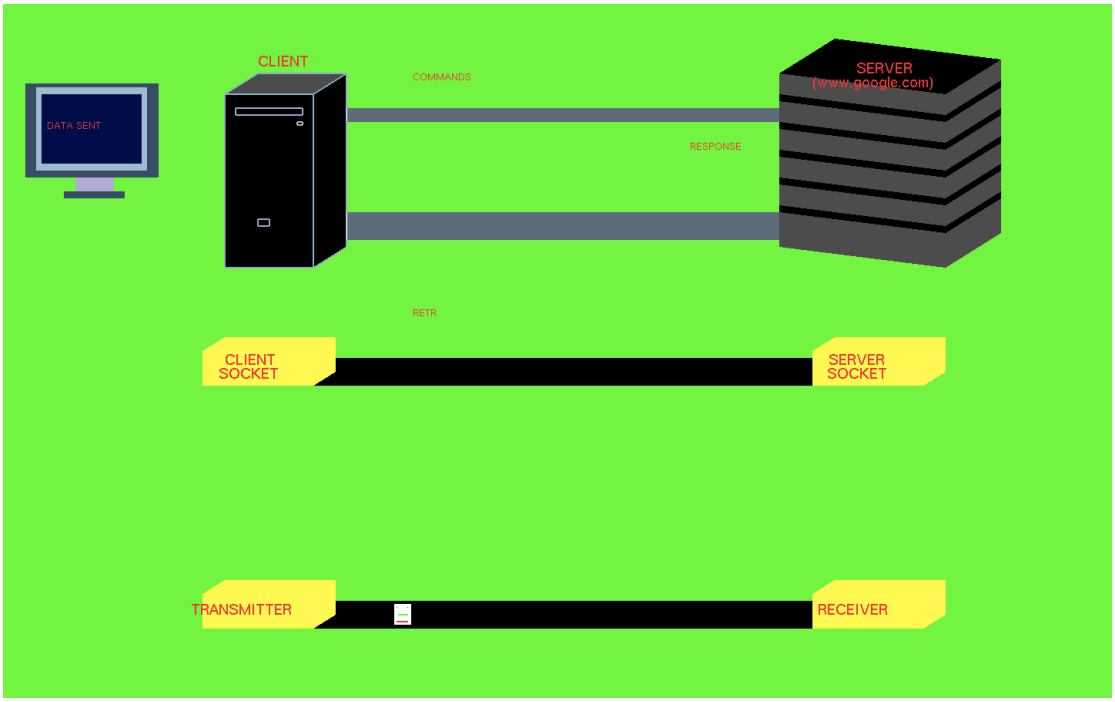
Server sends a response to client after authenticating it.



Data packets sent to server from client after handshaking.



Request sent from client socket to server socket for data packet retrieval.



Server responds to the client with requested packet.

CHAPTER 8
CONCLUSION

The users of this client server simulation will be able to view content as it takes place in the client – server network. The fully functional system as per the functional requirements is ready for deployment. It will be used share study materials at the institutional level and later will be made open for public to use. Testing of the application is done to ensure its integrity and reliability.

The simulation serves to provide a simulation where the users can see the data packets as they move between the client and server within a given network. This can be used for educational purposes where the students can learn about what exactly happens in TCP communication. This also saves work for the professor or faculty who have an interest or work in computer networks in general. The simulation allows users to visualize the many aspects of network communication including the different functions such retrieve, store and list which cannot be explained very easily to first timers.

A graphical representation with different colors used to explain the various aspects of the components and their working saves several hours of time and effort spent in explaining the concepts and improves the overall lecture instead of explaining it on a board or paper. This helps the user or student understand networks and how the PC connects to the internet in general.

CHAPTER 9

FUTURE ENHANCEMENTS

1. Visualization of content inside the data packets.
2. Explanation of routing through the network using different network devices such as router, hub, gateway, etc.
3. Connectivity in the network.
4. Visualization of further concepts such as peer – to – peer.
5. Visualization of protocols used in networks.

CHAPTER 10
REFERENCES

1. https://www.opengl.org/
2. https://stackoverflow.com/
3. https://www.britannica.com/technology/client-server-architecture
4. https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_Introduction.html
5. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd Edition, Pearson Education,201.
6. Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition Pearson Education, 2008.