EECE 4830 - Network Design Project Phase One: Design Document

By: Kalvin McCallum

Table of Contents

[1. Overview 3](#_Toc114318898)

[2. Purpose 4](#_Toc114318899)

[3. Code Explanation 5](#_Toc114318900)

[3.1 Software Architecture Figures: 5](#_Toc114318901)

[3.2 Software Terminal Output Figures: 14](#_Toc114318902)

# Overview

As part of the Network Design class, a programming project will form part of the overall final grade for each student. This project is broken down into five phases, of which the first phase will be completed on an individual basis and the last four phases will be completed as a group. The project focuses on the overall concepts around the TCP/IP stack which has five layers. These layers are known as the application, transport, network, link, and physical layers. During phase one of the project, each student must individually implement the standard user datagram protocol (UDP) sockets. This design document details the architecture and protocols utilized to implement phase one of the project.

The following table illustrates the software tools and languages used to execute phase one of the Network Design project:

|  |  |  |
| --- | --- | --- |
| **Project Phase** | **Software Tools** | **Software Languages** |
| Phase 1 – User Datagram Protocol Client & Server | * Linux Operating System or Oracle VM Virtual Box or Cygwin * Notepad++ or Visual Studio or Vim | C++ |

# Purpose

The purpose of phase one is to design and program a standard user datagram protocol (UDP) client and server sockets. Then, using a unique message string, such as “HELLO”, a message must be sent from the client to the server. The message string must then be echoed back from the server to the client. Different port numbers will be utilized to allow the server and client to exist on the same host. Furthermore, bidirectional communication between the server and client must be facilitated.

The deliverable requirements for phase one of the project are listed below as software requirements:

* 1. The program *SHALL* deliver messages bidirectionally between the User Datagram Protocol server and client.
  2. The program *SHALL* send a message from the User Datagram Protocol client to the server.
  3. The program *SHALL* echo the message from the User Datagram Protocol server back to the client.
  4. The program *SHALL* utilize a single host for the User Datagram Protocol server and client.
  5. The program *SHALL* respectively utilize unique port numbers for the User Datagram Protocol server and client when using a single host.

The following files were used to implement phase one of the User Datagram Protocol client server project:

**1. Makefile** - the Makefile contains a list of Linux commands that are used to build the software program. The purpose of this file is to simplify and automate the setup process of the program.

**2. Socket.h & Socket.cpp** - the purpose of these files is to implement the raw socket API calls. The Socket.h file contains function definitions used in the Socket.cpp file. The Socket.cpp file is the main socket file used to implement the socket class.

**3. SocketException.h** - the purpose of this file is to implement a mechanism that can handle exceptions when running the UDP server and client. This header file will prevent the UDP server and client program from terminating when an exception occurs.

**4. simple\_server\_main.cpp** - the purpose of this file is to facilitate and implement the main functionalities and executables of the UDP server. This is the main server file of the UDP server client program.

**5. ServerSocket.h & ServerSocket.cpp** - the purpose of these files is to implement the server socket API calls. The ServerSocket.h file contains function definitions used in the ServerSocket.cpp file. The ServerSocket.cpp file is the main server socket file used to implement the ServerSocket class.

**6. simple\_client\_main.cpp** - the purpose of this file is to facilitate and implement the main functionalities and executables of the UDP client. This is the main client file of the UDP server client program.

**7. ClientSocket.h & ClientSocket.**cpp - the purpose of these files is to implement the client socket API calls. The ClientSocket.h file contains function definitions used in the ClientSocket.cpp file. The ClientSocket.cpp file is the main client socket file used to implement the ClientSocket class.

The execution of these files ensures that the requirements of phase one are accurately and efficiently implemented.

# Code Explanation

The following screenshots capture the architecture and output of the software used to implement phase one of the User Datagram Protocol client server program.

## Software Architecture Figures:

*Figure 1*: Make File

Text

Description automatically generated

*Figure 1* illustrates the Makefile which contains a list of Linux commands that are used to build the software program. The purpose of this file is to simplify and automate the setup process of the program.

*Figure 2A*: Socket Header File

Text

Description automatically generated

*Figure 2A* illustrates the socket header file which defines the functions used for the socket main file. Together these two files make up the socket class which is utilized in the User Datagram Protocol (UDP) program.

*Figure 2B*: Socket Main File Part 1

Text

Description automatically generated

*Figure 2B* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which the socket is implemented as well as the implementation of closing the socket.

*Figure 2C*: Socket Main File Part 2

Text

Description automatically generated

*Figure 2C* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which creating a socket is implemented.

*Figure 2D*: Socket Main File Part 3

Text

Description automatically generated

*Figure 2D* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which socket binding to ports occurs.

*Figure 2E*: Socket Main File Part 4

Text

Description automatically generated

*Figure 2E* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which listening for connection between sockets occurs.

*Figure 2F*: Socket Main File Part 5

Text

Description automatically generated

*Figure 2F* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which acceptance between sockets as well as sending data between sockets occurs.

*Figure 2G*: Socket Main File Part 6

Text

Description automatically generated

*Figure 2G* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which receiving data between sockets occurs.

*Figure 2H*: Socket Main File Part 7

Text

Description automatically generated

*Figure 2H* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which connecting sockets occurs.

*Figure 2I*: Socket Main File Part 8

Text

Description automatically generated

*Figure 2I* illustrates the Socket main file which utilizes the function definitions created in the Socket header file to implement the functionalities of the User Datagram Protocol (UDP) Socket class. This particular screenshot presents the mechanisms by which non-blocking for sockets occurs.

*Figure 3*: Socket Exception Header File

Text

Description automatically generated

*Figure 3* illustrates the socket exception file which defines the functions used to handle any exceptions experienced will running the User Datagram Protocol client server program. This file helps ensure that when an exception occurs the program does not simply terminate. Rather it will allow the program to continue running until the desired functions occur.

*Figure 4A*: Server Socket Header File

Text

Description automatically generated

*Figure 4A* illustrates the server socket header file which defines the functions used for the server socket main file. Together these two files make up the ServerSocket class which is utilized in the User Datagram Protocol (UDP) program.

*Figure 4B*: Server Socket Main File Part 1

Text

Description automatically generated

*Figure 4B* illustrates the first part of the server socket main file which utilizes the functions definitions created in the server socket header file to implement the functionalities of the User Datagram Protocol (UDP) ServerSocket class. Additionally, this screenshot captures the mechanisms by which the create, bind, and listen objects are constructed within the ServerSocket class.

*Figure 4C*: Server Socket Main File Part 2

Text

Description automatically generated

*Figure 4C* illustrates the first part of the server socket main file which utilizes the functions definitions created in the server socket header file to implement the functionalities of the User Datagram Protocol (UDP) ServerSocket class. Additionally, this screenshot captures the mechanisms by which the send, receive, and accept objects are constructed within the ServerSocket class.

*Figure 5A*: Client Socket Header File

Text

Description automatically generated

*Figure 5A* illustrates the client socket header file which defines the functions used for the ClientSocket.cpp file. Together these two files make up the ClientSocket class which is utilized in the User Datagram Protocol (UDP) program.

*Figure 5B*: Client Socket Main File Part 1

Text

Description automatically generated

*Figure 5B* illustrates the first part of the client socket main file which utilizes the functions definitions created in the client socket header file to implement the functionalities of the User Datagram Protocol (UDP) ClientSocket class. Additionally, this screenshot captures the mechanisms by which the create and connect objects are constructed within the ClientSocket class.

*Figure 5C*: Client Socket Main File Part 2

Text

Description automatically generated

*Figure 5C* illustrates the second part of the client socket main file which utilizes the functions definitions created in the client socket header file to implement the functionalities of the User Datagram Protocol (UDP) ClientSocket class. Additionally, this screenshot captures the mechanisms by which the send and receive objects are constructed within the ClientSocket class.

### Software Terminal Output Figures:

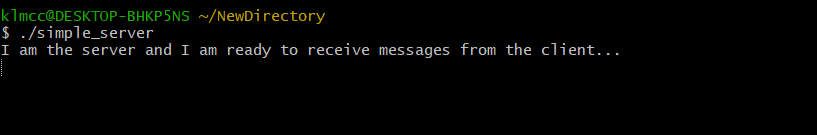
*Figure 6*: Make Command Terminal Output

Shape

Description automatically generated with medium confidence

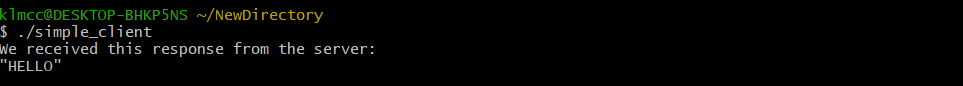
*Figure 6* illustrates the terminal output that will occur once you have successfully run the make file in the Linux command window.

*Figure 7*: Server Terminal Output



*Figure 7* illustrates the terminal output that will occur once you have successfully initiated the User Datagram Protocol server application program. This output indicates to the user that the server has successfully been constructed and is listening for messages from the client.

*Figure 8*: Client Terminal Output



*Figure 8* illustrates the terminal output that will occur once you have successfully initiated the User Datagram Protocol client application program. This output indicates to the user that a sting “HELLO” was sent from the server and was echoed back to the client.