**COMPUTER ENGINEERING**

COURSE CODE / NAME : CPE309 / DATA COMMUNICATION

ASSIGNMENT : Group Project (THREE Students per group)

*(Öğrenci grup olarak proje yapacaklardır.)*

DEADLINE : 31 December 2022 17:30 via MS TEAMS

*(Proje son teslim tarihi 31 Aralık 2021 tarihi 17:30 dur.)*

MARKS : 15 Points *(Proje 15 puan.)*

1. **OBJECTIVES**

This project aims to develop student’s knowledge and skills in socket programming using C++ programming language. A program for communication between clients through server will be developed. Multiclients (more than 1 client) can communicate with each other by using server simultaneously. The communication should be seen in consoles. TCP protocol will be used. (İstemciler arasında sunucu üzerinden iletişim için bir program geliştirilecektir. Çoklu istemciler (1'den fazla istemci), sunucuyu aynı anda kullanarak birbirleriyle iletişim kurabilir. İletişim konsollarda görülmelidir. TCP protokolü kullanılacaktır.

3. Thread functions should be used for the simultaneous communication. (Eş zamanlı iletişim için thread fonksiyonu kullanılacaktır.)

1. **DELIVERABLES**

Each group shall present their projects using their own computer and submit project source code files as a rar file via Teams. In addition, some questions about project will be asked during each presentation. (Öğrenciler projelerini kendi bilgisayarları ile sunacaklardır ve sunum dosyalarını rar dosyası olarak Teams’e yükleyeceklerdir. Sunum esnasında proje ile ilgili soru sorulacaktır.)

Same projects will not be accepted. Also, the final exam score of students who bring the same project will be reduced by 10 points. (Aynı projeyi getirenlerin projesi değerlendirmeye alınmayacaktır ve bu öğrencilerin final notlarından 10 puan düşülecektir.)

1. **PROJECT DESCRIPTION**

In this project, students are required to implement a multi-threaded chat room service. The system will have one (multi-threaded) chat server, and multiple chat clients. Socket interface is used to implement network communications. The C/C++ chat application you are going to build is a console application that is launched from the command line using TCP connection.

There can be multiple clients connect to a server and they can chat to each other. Only two users who are messaging can see each other’s message and other users cannot see these messages. It is a private chat between two users.

The application consists of two parts: Server and Client. Each part can run independently on separate computers.

Server manages the chat session. It maintains a list of the active clients and forwards incoming messages. The Server is multithreaded - communication with each client is through a separate thread. When Server is started, it starts to wait for client connections.

This thread continues forever to listen for clients. When a connection request is received, the server starts a thread to service the client. This is necessary for each client to have its own socket. When the thread is started, a Client object is created containing all the relevant info about each client and saved in a list.

The ServiceClient() thread is where all the work is done. ServiceClient() defines the implementation of the server side of the application protocol. The application protocol is the semantics of message exchange between the server and client. Firstly, the processing of a message does not depend on a previous message or any other context - messages can be received at any time and in any order without confusing the application. Second, the messages are fully self-describing, meaning all the data needed to process a message is contained within the message. Before the server forwards the message from the sender client to the receiver client, the server should randomly corrupt or not corrupt the message. If the receiver detects that the message is incorrect using error checking algorithms, it notifies the server that the message has an error by sending MERR command. The server recognizes four commands, CONN, MESG, MERR and GONE. CONN establishes a new client, sends a list of current clients, and notifies other clients a new person has joined the group. MESG sends a private message to the person designated. MERR sends last message again without an error if there is an error detected from receiver side. GONE removes an active client account and notifies all other members the client has left and kill the thread.

The server receives the incoming messages as ASCII strings. The '|' char is used as the separator between parts of the message. A message consists of a command, error checking bits and one or more other parameters required to process the message. The Server, visually, it just displays the active clients’ host ip and name.

The Client allows users to connect to the server and send and receive messages. After starting up, write your chat name and enter to Connect. The Server will respond with a list of current chat clients. After that you can send messages by typing the user name, “->” and message to Send. Only one name can be selected at a time. Then send the message. The server will get the message with a MESG code and an error checking bits.

There must be a method constructs and sends a CONN command to the Server. The CONN command contains the chatters name as well. The command is constructed using the '|' char as a separator. The Client then receives the list of chatters and adds them to the array.

After the Client has connected, it has to handle all the ins and outs. ReceiveChat() implements the client side of the application protocol.

When Client will begin, program starts writing all the chat and messages to a text file. The file name is built from the current date, time and user name. A sub directory called "logs" will be created to store the log files.

1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check