CSC3002F Networks Assignment 2022
Socket programming project

Department of Computer Science
University of Cape Town, South Africa

February, 2022

This assignment is on networked applications where you are required to develop a Python-based client-server chat application in groups of three students. This document describes the context/requirements, what to submit, and some basic information about socket programming.

Chat Application Requirements

There are a range of chat applications with a variety of features. Some are intended for group communication, such as the IRC, whereas others, such as Facebook Chat, are typically designed for a 1:1 private chat. A few others, such as Skype and WhatsApp, are typically used for both 1:1 and group chat. There are also others that are ideal for broadcast messaging in group communication (Vula chat is one example of this). All these 'chat apps' are supported by different protocols and can be distinguished based on their features and interaction interfaces.

In this assignment, you are required to design an application layer protocol and use it to implement in Python a chat application **that uses UDP at the transport layer**. Your application layer protocol must support the client-server architecture, and implement a server that should manage the interaction between clients. The chat application must allow multiple pairs/groups of users to exchange messages in real-time. As this is a network application, the different clients and the server should be able to run on different hosts over a network. The client interface may have either a GUI or CLI, with appropriate user menus. Additional features may include user authentication, and the ability for a user to retrieve historical messages from the server (e.g if a user was offline when the messages were sent to them should be able to retrieve those messages when they come online).

Protocol Requirements

An important step in protocol design involves specifying requirements and constraints of the protocol, such as whether real-time interaction is expected, and reliability, i.e., if we need to verify/check that every message is delivered correctly. Since this application will be using UDP as its transport layer protocol (which provides unreliable service), the application layer protocol needs to be designed with its own mechanism for achieving reliability. Users of the application need confirmation that their messages have been received by either the server or destination client (similar to WhatsApps single or double tick).

You should design the application layer protocol to verify that messages are correctly received (error detection). For example, you could use a hash function on both the sender and receiver side to verify correctness of messages. The protocol should also verify/confirm that all messages sent are received (loss detection). You need to design a mechanism for exchanging acknowledgement passages between clients and servers.

Protocol Design

Protocol design includes specification for the pattern of communication, messagestructure, as well communications rules. In this assignment, the pattern of communication will be client-server, meaning that a server will be responsible for the overall control and coordination of communication between clients.

Messages in an application protocol could be either text, consisting of readable character strings (e.g 'CHAT', or 'JOIN', etc), or binary format, where messages are blocks of structured binary data (eg. where '1000001' is used to mean 'CHATt'). Text-based protocols have the advantage of being human readable, hence providing for easier understanding, monitoring and testing. **You are required to use text-based messaging (ASCII) in this application protocol design** (Hint: look at the HTTP protocol)**.**

The protocol design also specifies the acceptable sequence of messages at every stage of communication. For example, a protocol may dictate that message exchange will only occur after a communication session has been set up. This requires clearly specifying messages and reactions for every communication scenario. Three types of messages can be defined:

- Command messages define the different stages of communication between parties, such as the initiation or termination of a session.
- Control messages manage the dialogue between parties, such as message acknowledgements and retransmission requests.
- Data transfer messages are used to carry the data that is exchanged between parties.

The message structure comprises at least the header and body. The header, whose structure must be known to both the sender and receiver, may contain fields that describe the actual data in the message. Some of the fields/information contained in the header might include the message type, the command, recipient information, and message sequence information. The header generally has fixed size and contains clues that should help the receiver to understand the rest of the message.

What to submit:

You will be required to submit the following (1-3 as a single zip folder):

1. Your code with proper inline documentation (comments)
2. A report (5 pages maximum) on the design and functionality of your chat application.
3. The report needs to include:
    a. A list of features with an explanation for their inclusion.
    b. A protocol specification, detailing the message formats and structure. You are required to include sequence diagram(s).
    c. Screenshots of the chat application revealing its features. To personalise the screenshot(s), you have to mention all three student numbers of your group in a chat of which you take the screenshot.
4. Oral presentation to be scheduled with the TA and Tutors