EE444 Introduction to Computer Networks

HW1 – Socket Programming

Introduction

In this homework, a simple network system consisting of 3 nodes, namely Client, Proxy, and Server, implemented using python programming language. Aim of this homework is to write various processes which communicate through TCP sockets. Topology which is defining this system can be seen in Figure 1.

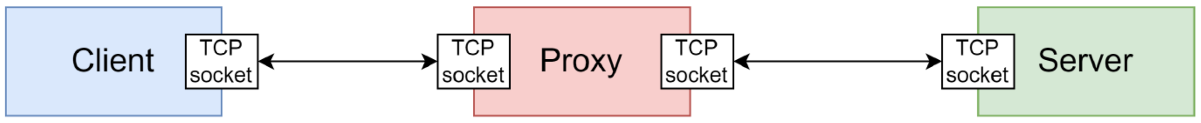


Figure 1

Network applications can be developed using number of different programming languages, and they can be running on different types of devices and operating systems. There are several communication protocols which are running on Internet Protocol (IP) defined for this type of applications to transfer data and messages between them, and Transmission Control Protocol (TCP) is one of them. TCP is a ~~stateless~~ protocol and state diagram for TCP sockets can be seen in Figure 2.

stateful

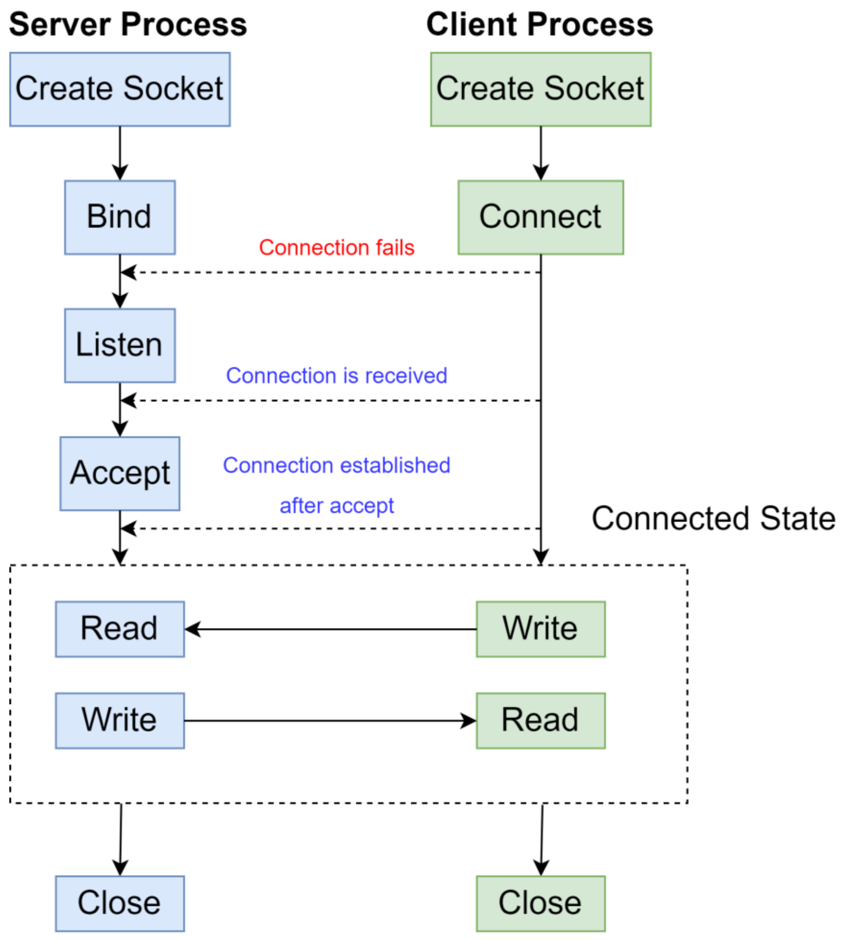


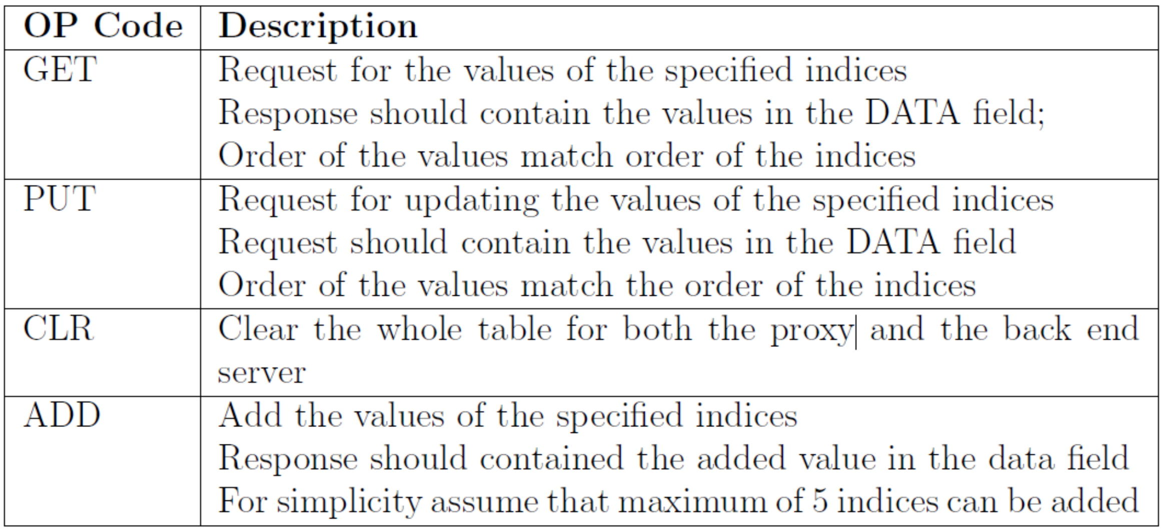
Figure 2

Although it is asked to implement server process in MATLAB, due to local development environment problems faced during MATLAB development, server process is also implemented in python. Different storage table structure is formed and alterations in some function structure is made to being consisting with MATLAB development syntax and approach in arrays and data structures compared to proxy process implementation.

All functionality and reasoning behind the functions and variables defined in these three programs are explained with further detail in the source code as comments, being consistent with the function and variables names used throughout the source codes. Implementation details which are thought to be worth mentioning are written in this report. Ordinary functions which were expected to be implemented are implemented in each of the processes as required but, only extra features and/or functionalities are mentioned in this report.

Base functionality expected from the solution can be seen in Table 1.

Table 1



OP Code	Description
GET	Request for the values of the specified indices Response should contain the values in the DATA field; Order of the values match order of the indices
PUT	Request for updating the values of the specified indices Request should contain the values in the DATA field Order of the values match the order of the indices
CLR	Clear the whole table for both the proxy and the back end server
ADD	Add the values of the specified indices Response should contained the added value in the data field For simplicity assume that maximum of 5 indices can be added

Implementation

Client process is implemented in the python file with the name “Client_process.py”. It establishes a connection with the proxy process and perform operations by communicating with only the proxy process on port 8081 of proxy process’s host. A cli is developed for the client interface and user is prompted with available operations and required parameters for the desired operation. In every step validity of the user input is checked, using error handling user is warned and prompted for other trials within an infinite while loop.

Backend server process is implemented in the python file with the name “Server_process.py” and chosen to be run on port 8080. Server process accepts connections through to port 8080 of its host and listens for the incoming messages. Performs necessary operations and respond to proxy process with required data and format. There is a list of 10 elements to be stored in the server process as the server table. Implementation is initialized with arbitrarily assigned 10 numbers. Implementation could be started with a list of 10 ‘None’ elements. When cleared, the list is reset to that format.

Proxy server process is implemented in the python file with the name "Proxy_process.py" and chosen to be run on port 8081. Proxy process established a connection with the server process through port 8080 of server process host and accepts connections through port 8081 of its own host. Proxy process listens for the incoming messages from the client process and performs necessary operations and send response to client process. Proxy process is assumed to be able store half of the size of the server table (5 elements). After every get operation, 5 data retrieved the very last are stored in the proxy table. If client process request to get data which are stored in the proxy table, proxy process respond to client **without reaching the server process**. If one or more of the indexes requested by the client does not exist in the proxy table, proxy process asks the server process for **only the missing data.** After successfully retrieving the missing data, it updates its table and append those new indices and data to message to be sent to the client process.

An additional field of **CODE=XXX** is implemented in the solution. HTTP status code convention is followed for implementation of successful (200), not found (404) and in case of a server error (500) in proxy or server process status codes are made. Using this status code convention both in the proxy-server and the client-proxy communication, response handling made consistently using this CODE field in the transmitted message. After decomposing proxy's response message in client, user is informed with appropriate forms of console outputs.

If one or more of the requested indices for GET or ADD operations by the client are not yet defined in server table (server is reached by the proxy only if it is necessary), server table responds to proxy process with an error code of 404 (Not Found), and proxy responds with the same code to client.

Functional programming paradigm is tried to be followed during the development of all the processes. Global variables to be shared across all a process's functions are implemented. Every task to be formed for a specific purpose is handled in a function named consistently with its task. A further application of this system can be implemented by further collecting some common functions to decomposing and composing messages using the global variables in all those processes and using enumerated response status across all processes would be helpful for consistency.