ASSIGNMENT REPORT

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Under the Guidance of:

Prof. Aaditeshwar Seth
Assot. Professor

CSE Department

Submitted by:

Karan Agrawal (2023MCS2479)
Annanya Mehta (2023MCS2484)

INDIAN INSTITUTE OF TECHNOLOGY, DELHI

Hauz Khas, New Delhi -110016, India

Detailed Report on the Provided Python Code: Client-Server Communication

Introduction

This report provides a comprehensive analysis of a Python script that serves as a client in a client-server communication system. The primary objective of this script is to interact with a remote server using the User Datagram Protocol (UDP). The client's responsibilities include retrieving data from the server in smaller segments, assembling these segments into a complete dataset, calculating an MD5 hash for integrity checking, and subsequently submitting both the hash and data back to the server. Additionally, this script offers a feature to plot request and reply times along with corresponding offsets using the Matplotlib library. To understand this script better, we will break it down into various sections.

Code Overview

Server Information

The script begins by defining essential server information, including the server's IP address (SERVER_HOST), port number (SERVER_PORT), and a timeout duration for socket operations (timeOut). These parameters are crucial for establishing a connection with the server.

Server Host: "127.0.0.1"
Server Port: 9801
Timeout: 0.03 seconds
Timing and Offset Lists
The script initializes several lists to record and analyze timestamps and offsets during client-server communication:

Lists for Request and Reply Time & Offset

This section initializes lists that will be used to record and analyze timestamps and offsets during the client's communication with the server. These lists are crucial for the subsequent plotting of request and reply times.

Request Time: Records timestamps when a request is initiated.
Reply Time: Records timestamps for server reply reception.
Request Offset: Records the offset when a request is made to track the requested data segment's position.
Reply Offset: Records the offset when a reply is received to analyze each segment's position in the final dataset.

These lists play a crucial role in understanding and visualizing the timing and sequencing of client-server interaction

Change Timeout Function

The change_timeOut function dynamically adjusts the timeout duration based on the client's recent interactions with the server. It calculates the timeout value by considering the round-trip time (RTT) and the request number. This adaptive approach ensures that the client can handle the server's responses within a reasonable timeframe.

Server Request Function

The server_request function is the core of the client's communication with the server. It is responsible for sending requests to the server and receiving responses using UDP sockets. Additionally, it records timestamps and offsets for request and reply operations. The function's main steps are:

Sending a request message to the specified server using a UDP socket.
Recording timestamps, request time, and reply time.
Handling timeouts and adapting the timeout duration if required.
Returning the server's response.
This function is the workhorse behind the client's communication with the server.

Connect Server Function

The connect_server function orchestrates the complete communication process between the client and the server. This function includes the following key steps:

Initializing a UDP socket for communication.
Sending an initial "SendSize" command to the server, which informs the server to expect data transfer.
Receiving the server's response, which contains information about the size of the data to be retrieved.

Calculating the number of requests required to retrieve the entire dataset and initializing a data list to store the segments.
Requesting data segments from the server in a loop, ensuring each segment is received correctly.

Assembling the received segments into a complete dataset.
Computing an MD5 hash of the complete dataset to ensure data integrity.
Submitting the computed hash and the data to the server using a "Submit" command.

Utilizing Matplotlib to create a plot that displays the request and reply times along with offsets, providing a visual representation of the communication timeline.
The connect_server function is the heart of the client's interaction with the server, responsible for data retrieval, assembly, integrity checking, and submission.

Script Execution
The script's execution starts with the main function, which serves as the entry point. The conditional block if __name__ == '__main': ensures that the main function is executed when the script is run. This function, in turn, initiates the client-server communication process.

In conclusion, this Python script effectively functions as a client for a client-server communication system using UDP. It manages data retrieval, ensures data integrity, and provides a visual representation of communication timings. It consists of various functions that facilitate interaction with the server and is initiated by running the main function.

Running Time Graphs