Problem Statement

Create a client and server in python

1. The server should generate data (Use a library such as faker to generate user information data or numpy to generate random arrays) and send it to the clients connected to it one message at a time. It should also print any message from any of its clients whenever it is received.
2. The client connected to the server should collect the data obtained from the server in blocks of arbitrary block size of your choosing (say 5 messages received) and then send a response back to the server saying 'Client <client_name> received <n> block in <t> time', where n has to be the sequence number of the block that the client received and time t taken for the client to receive the entire message block.

Features

1. The server should be able to handle multiple concurrent connections.
2. The client has a mechanism to re-establish the connection with the server if the server stops for some reason and then restarts, instead of just exiting from the program.
3. The client should log the blocks received in a text file or the console, along with the timestamps of each message in the block. (The timestamps are used to calculate the amount of time it takes to receive each block)

Bonus Points for

1. Analyzing the difference in the time taken for different block sizes and higher number of clients connected to the server.
2. Allowing for a graceful disconnection of a client from the server, such as removing the client from the list of connected clients to the server.
3. Allowing the feature for the client to specify the kind of data to be received from the server, such as being able to choose between random user information data or random arrays.

Pointers

1. First, create a simple client and server where the server sends messages to the client which receives all of them and logs them, and then you can move on to collecting the messages in blocks and sending back responses from the client to the server.
2. Timestamp each message the client receives, calculate the total time taken to receive one entire block, and log the blocks along with the timestamps. Look up the ***time*** module for python.

3. Add the feature of handling multiple concurrent connections in the server. Look up concurrency and concurrent tasks in ***asyncio***, and find a way to send messages to all the connected clients instead of just one. Make multiple copies of the same client file and run them all together to get multiple clients trying to connect to the server.
4. Add the feature of re-establishing the connection with the server on the client side. This involves error handling and subsequent action taken in case of an error.

Note

You can use any module of your choosing and you do not need to use only the ones mentioned here, as long as the final application has the same functionality and features.

Resources

Slides,
https://docs.google.com/presentation/d/1HURfT-pSG9SbyhgMyVy-F5rnBIG-PNhC7sAmoHWlRxc/edit?usp=sharing
Code,
https://github.com/ChinmayaSharma-hue/websockets-demos