

Project: TCP Socket Programming

By Josue Palacios

Introduction: For this project, I learned how to build a client and server applications that are able communicate with each other using sockets. Sockets are door-like pathways between the application process and the transport protocol. There are two kinds of sockets: TCP and UDP. Of the two, I used TCP, which is a more reliable datagram and has an in-order byte-stream transfer between client and server.

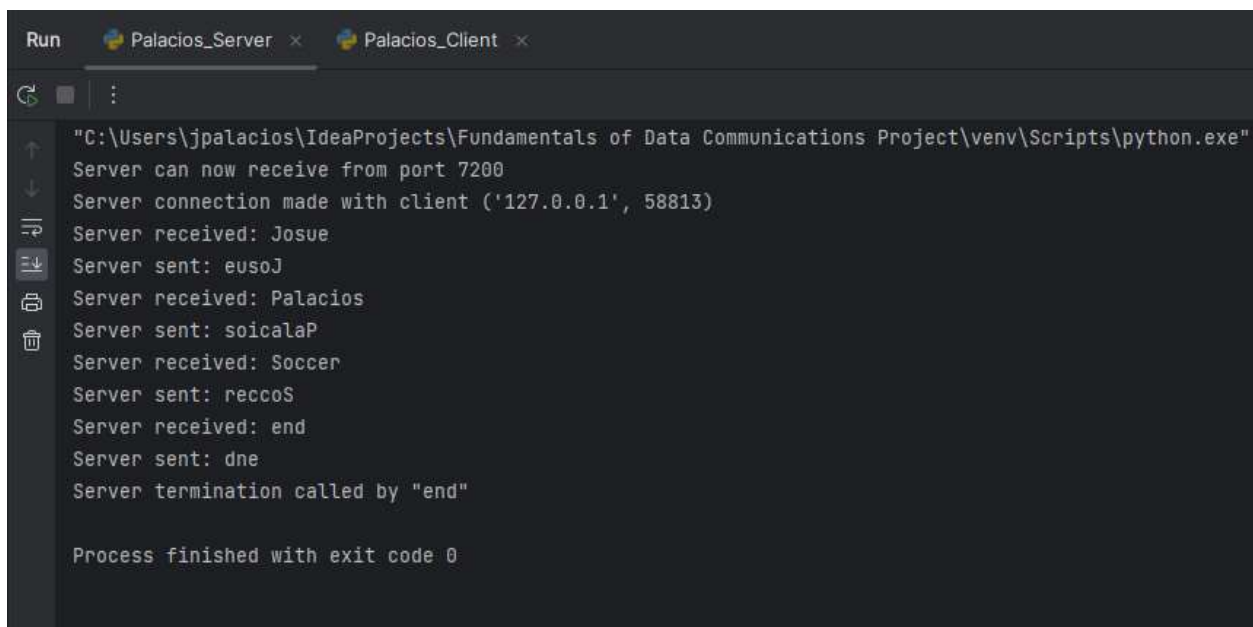
In socket programming with TCP, we will first create the connection. For this, the client needs to contact the server, which must already be up and running. The client's contact with the server is done by making a TCP socket, specifying the IP address, and then creating a port number of the server process. Once the client has made the socket, the client's TCP establishes a connection to the server's TCP. However, the server will need to have already made the socket that will accept the client's contact. This socket is made by the server when it is contacted by the client. Once this is done, the connection with the server and the client is set. At this point, a message can be sent between the client and the server.

For the project, a reverse-echo server and clients that run on different terminals are needed.

Reverse Echo Server: The reverse echo server's task is to receive a message from a client over the TCP socket and send the message back to the client in reverse order. The program will be terminated upon receiving "end" from the client and will send "dne" before then. First, connections must be allowed via python function. Then the port (7200) and IP addresses accepted (0.0.0.0) are set. In this case, 0.0.0.0 is used so that connections can be accepted from any IP. Once this is done, the server is attached to the port and IP address of the client. The server has also been set to only allow one connection at a time. With the preparations done, the server sends a message to the terminal that the server can now receive from the client's port. In my code, the new socket for communicating with the client and the IP address of the client are accepted by the server socket. The terminal will then output that the connection has been made with the client while displaying the client's IP. The client socket is then used to communicate with the server. A loop within the client socket is made so that the server will continuously receive data from the client, decode the data to a string, print what is sent by the client, reverse the string, turn data back to bytes, send reversed string to the client, print reversed message, and repeat until the server received "end" by the client where it will then terminate.

Echo Client: The echo client's task is to prompt the user to send a message, take in a message from the user, and send the message out to the connected server. From there, the message will be reversed once it returns from the server and will output the message to the user. This process will repeat until the user inputs "end" and receives "dne" from the server. Just as before, to begin the program, connections need to be allowed via Python's built-in functions. Once that is done, the hostname "localhost" and port "7200" are taken in. Then, the client is connected to the server's IP and port. This is followed by the terminal communicating that the client connection has been made with the server along with the host name and server's port. Then a loop is made where the terminal will continuously ask the user to enter a message, where it will then be encoded into bytes, sent to the server, await the server's response, decode the string, and then print the reversed response from the server. This loop is ended once the user inputs "end" and receives "dne" from the server.

Server Output on Terminal 1:

A screenshot of a terminal window with two tabs: 'Palacios_Server' and 'Palacios_Client'. The terminal shows the following output:

```
"C:\Users\jpalacios\IdeaProjects\Fundamentals of Data Communications Project\venv\Scripts\python.exe"  
Server can now receive from port 7200  
Server connection made with client ('127.0.0.1', 58813)  
Server received: Josue  
Server sent: eusoJ  
Server received: Palacios  
Server sent: soicalaP  
Server received: Soccer  
Server sent: reccoS  
Server received: end  
Server sent: dne  
Server termination called by "end"  
  
Process finished with exit code 0
```

Client Output on Terminal 2:

```
Run Palacios_Server x Palacios_Client x
"C:\Users\jpalacios\IdeaProjects\Fundamentals of Data Communications Project\venv\Scripts\python.exe"
Client connection made with server at localhost : 7200
Client enter message: Josue
Client received: eusoJ
Client enter message: Palacios
Client received: soicalaP
Client enter message: Soccer
Client received: reccoS
Client enter message: end
Client received: dne
Client termination called by message "end" and response "dne"

Process finished with exit code 0
```