Assignment One

Spring 2023 CPSC 471-01 Network Communications

Instructor: Mutalip Kurban

Due Date: 4/29/2023

Kevin Ortiz: keortiz@csu.fullerton.edu

Ben Martinez: benmrtnz27@gmail.com

Anthony Maida: amaida@csu.fullerton.edu

Ethan Bockler: ethanbockler@gmail.com

Anna Chiu: anna.chiu@csu.fullerton.edu

Introduction:

The following protocol will be designed for a simplified FTP server and client using pythons built in socket programming API. The client shall connect to the server and support uploading and downloading of files to/from the server.

Control Channel Messages:

The types of messages that go across the control channel are the commands. There are four commands that are implemented which are ‘get’, ‘put’, ‘ls’ and ‘quit’. Get and put both require a file name to be followed after it. The server will then send the ephemeral port to the client on the control channel to be used for a data channel. The client will connect to this port and either send or receive data.

Responding to Messages:

Server responds to messages from the client with either a “SUCCESS” and then sends the required information or “FAILURE” and will not send anything. If `get` is sent, the server will respond with the requested data. If ‘put’ is sent, the server will be receiving data from the client. If ‘ls’ is sent, the server will read all the files in its directory and send it to the client. If ‘quit’ is sent, the control channel gets closed and the server will wait for another connection.

Messages Format:

The size of the message is initially 10 bytes for the header which says the size of the actual data so the server can determine the amount of data being received. The format for get is ‘get <filename>’, put is ‘put <filename>, ‘ls’ and ‘quit’.

File Transfer Channel Setup:

The client will send a command to the server, which includes ‘get’, ‘put’ or ‘ls’ (quit does not require a data channel). The server will then create a socket and bind the socket by doing ‘serverSock.bind(('', 0))’. This will select an available port to bind to. We then find what the port number is and send this to the client. The server will listen and wait for a connection, and once the client connects we have a data channel. All necessary data will be transferred through the channel and once everything is complete, the data socket will close (control channel is still connected). The quit command will close the control channel.

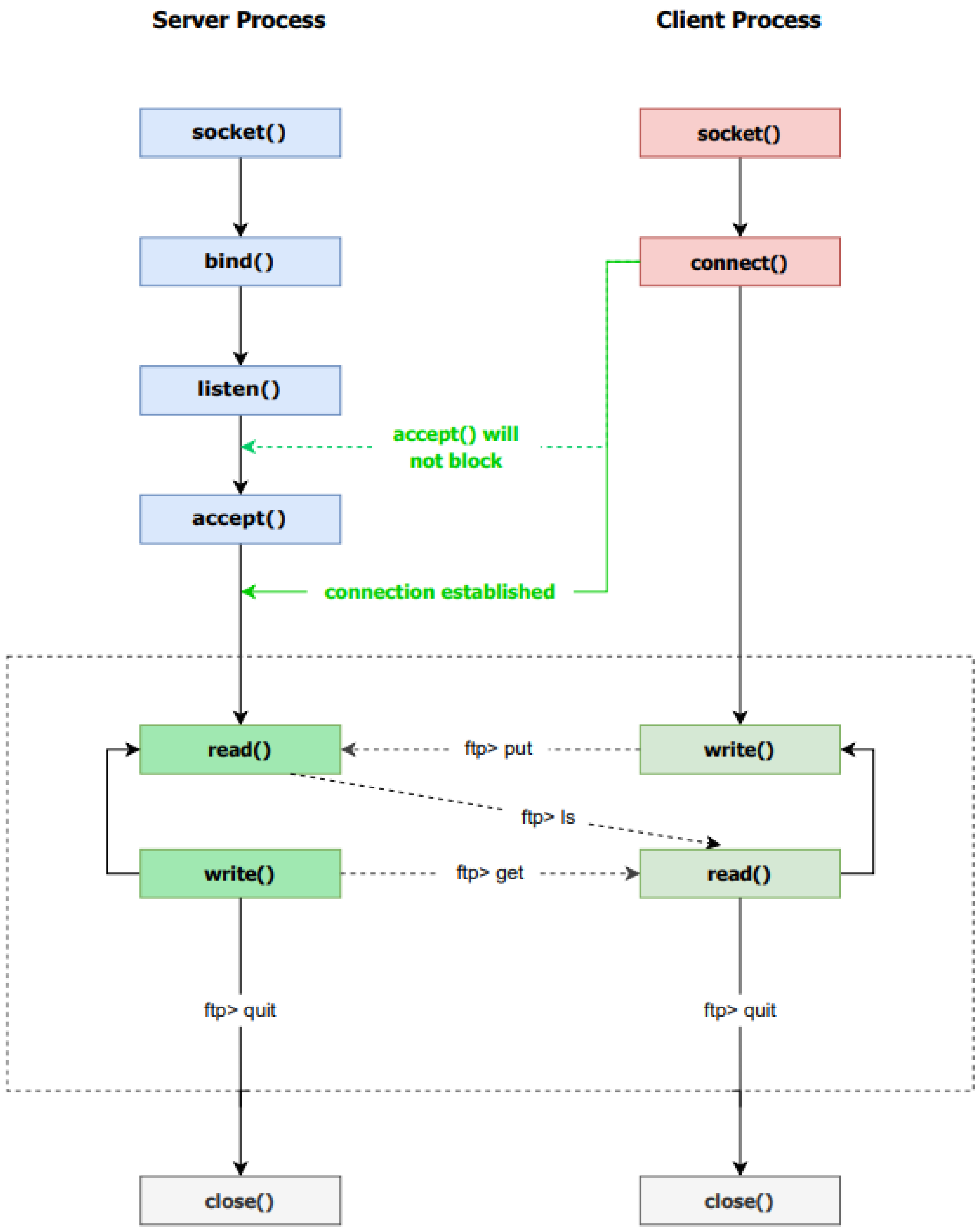
Start/Stop Receiving File:

First the client needs to be connected to the socket that the server is binded to. After this, the server is continuously listening to the client for a message. It will know when to start receiving or sending a file when the client sends a command such as ‘put’, ‘get’ or ‘ls’. The server will then create and send an ephemeral port to the client to connect to. The client will connect and then the server will send or receive data depending on the clients command. The receiving end knows when to stop receiving the file by looking at the header and getting the size of the data.

Overflowing TCP Buffer:

We will be reading and sending data between the client and server in 65,536 byte chunks to avoid overflowing.

Diagrams:


Server Process
Client Process
socket()
bind()
listen()
accept()
socket()
connect()
accept() will not block
connection established
read()
write()
write()
read()
ftp> put
ftp> ls
ftp> get
ftp> quit
ftp> quit
close()
close()