Alex Ziegenhorn – Project Part B: Code Framework

This is all going to be in Python, with a separate Python program for both the server and the client. Both of the classes will be importing the **socket** Python library. I will specify which functions from the socket library will be used in the main section of the framework.

Client Framework

- **SendFileRequest()**
  - **Important Instantiations**
    - destAddress : String | Destination IP address of the server program.
    - destPort : Integer | Destination Port Number for the server program.
    - message : String | Filename being requested, as set by the user.
    - sock : Socket | Socket object (from socket library); specify UDP implementation with socket.SOCK_DGRAM
  - **Important Notes**
    - Create a socket object using the socket Python library.
    - Use sock.sendto(message, (destAddress, destPort)) to send the request to the server.

- **ReceiveSegment()**
  - **Important Instantiations**
    - sourceAddress : String | Destination IP address of the client program.
    - sourcePort : Integer | Destination Port Number for the client program.
    - message : String | Filename being requested, as set by the user.
    - sock : Socket | Socket object (from socket library); specify UDP implementation with socket.SOCK_DGRAM
  - **Important Notes**
    - This function activates as soon as the client send a file request to the server, so that we can receive it's ACK and any other subsequent messages.
    - We use sock.bind((sourceAddress, sourcePort)) to start listening on the listen port for packets.
    - Include an infinite while loop that continuously checks for/sends ACK's of incoming segments.
    - This will have implementations of the required algorithms to prevent/manage packet loss.

Server Framework

- **ReceiveRequest()**
  - **Important Instantiations**
    - files : Array | Contains all of the files in the server's local memory.
    - sourceAddress : String | Destination IP address of the server program.
    - sourcePort : Integer | Destination Port Number for the server program.
    - message : String | Filename being requested, as set by the user.
    - sock : Socket | Socket object (from socket library); specify UDP implementation with socket.SOCK_DGRAM

  - **Important Notes**
    - This functions waits until a segment is sent to it, and then checks the data of the segment to make sure that a file exists within the server's memory that matches the name requested. If file does not exist, send back a message that reflects this. If it does, then we initiate the file transfer process.
    - This function activates as soon as the program is run, so that it is ready to receive a file request as soon as possible.
    - We use sock.bind((sourceAddress, sourcePort)) to start listening on the listen port for packets.
    - Include an infinite while loop that continuously checks for/sends ACK's of incoming segments.
    - This will have implementations of the required algorithms to prevent/manage packet loss.

- **SplitFile()**
  - **Important Instantiations**
    - requestedFile – File | This is the requested file that needs to be split into bytes that are ready to be transmitted.
    - biteSized – Array/Data Structure/(Hash?) | This is the structure that will contain the split sections of the requestedFile. The splits should be stored in a way that makes it very easy to send using the socket library.
  - **Important Notes**
    - This method will have little to nothing to do with the socket library.
    - This method is solely for getting the requested file into manageable chunks.
- **SendSegment()**
  - **Important Instantiations**
    - destAddress : String | Destination IP address of the client program.
    - destPort : Integer | Destination Port Number for the client program.
    - biteSized : Array/Data Structure/(Hash?) | This is the structure that will contain the split sections of the requestedFile. The splits should be stored in a way that makes it very easy to send using the socket library.
    - segment : String? | Segment of the file that is currently being sent.
    - sock : Socket | Socket object (from socket library); specify UDP implementation with socket.SOCK_DGRAM
  - **Important Notes**
    - Create a socket object using the socket Python library.
    - Use sock.sendto(segment, (destAddress, destPort)) to send the segment to the client server.