Michael Ghizzi
Rudy Alvarado
CPSC 471-02

1) What kinds of messages will be exchanged across the control channel?
TCP and FTP messages.
We can control the TCP messages by implementing a timeout, which will make it so if the server does not respond in time, the client will know to send another message to the server without restarting the program.
The FTP server and client will be mainly focused on uploading and downloading the text file to and from the client and server.

2) How should the other side respond to the messages?
The client and the server will accept the connection and listen for the client's commands.
The server will send an ACK to the client when it receives a command.
The server will print out a message indicating SUCCESS or FAILURE of a command which can also act as an acknowledgement that the command was transferred and it was run correctly.

3) What sizes/formats will the messages have?
The first 10 bytes of the message from client to server contain the file size and the rest contain the file data. If it does not have 10 digits, then we will append 0's to the left of the message until it has reached 10 digits. Once we have 10 digits on the message, we will then append the filename and contents of the file as well as convert it into an entire string.
Maximum number of bytes that can be transferred is 65536.
As can be seen from the snippet of code below:

```
# Get the size of the data read
# and convert it to string
dataSizeStr = str(len(fileData))

# Prepend 0's to the size string
# until the size is 10 bytes
while len(dataSizeStr) < 10:
    dataSizeStr = "0" + dataSizeStr

# Prepend the size of the data to the
# file data.
fileData = dataSizeStr + fileData
```

4) What message exchanges have to take place in order to setup a file transfer channel?
In order to setup a file transfer channel, we must setup a connection from the client to the server. We can do this by setting up a hostname and port number to establish a connection. The ephemeral port consists of both a host and port number. This port is created by the operating system when a program requests any available user port. The operating system selects the port number from a predefined range and releases the port after the related TCP connection terminates.

5) How will the receiving side know when to start/stop receiving the file?
We keep sending until the whole file is sent, and stop when the file is read.
If the file has been failed to send, an error message will follow.
If the user has entered the ‘quit’ option inside the menu(), then it will close connections with the server and quit the program.
As seen with the snippet of code below, we send the file until the entire file has been sent as a string of characters. The maximum number of bytes we can receive is 65536, anything above that and the receiving side will stop receiving the file.

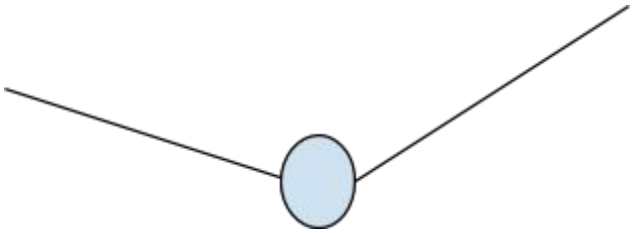
```
# Keep sending until all is sent
while True:

    # Read 65536 bytes of data
    fileData = fileObj.read(65536)

    # Make sure we did not hit EOF
    if fileData:


        # Get the size of the data read
        # and convert it to string
        dataSizeStr = str(len(fileData))

        # Prepend 0's to the size string
        # until the size is 10 bytes
        while len(dataSizeStr) < 10:
            dataSizeStr = "0" + dataSizeStr


        # Prepend the size of the data to the
        # file data.
        fileData = dataSizeStr + fileData

        # The number of bytes sent
        numSent = 0

        # Send the data!
        while len(fileData) > numSent:
            numSent += connSock.send(fileData[numSent:])

    # The file has been read. We are done
    else:
        break
```

6) How to avoid overflowing TCP buffers?
The TCP buffers received data because it does not know when the application is going to read the information and already has told the sender that it is ready to receive.
The application reads the data and it drops the data from the received window and increase the size, if the window does not exist then the sender must hold off sending until the receiver tells the sender it is okay to receive.
We can implement a timer to avoid overflowing of the TCP buffers, for example:
This timer will timeout for 30 seconds if the client cannot make a connection to the server.
The timer will timeout for 5 seconds if the server does not receive anything from the client.

7) You may want to use diagrams to model your protocol.
Client **Server**
while True:

1. Create Socket

2. Bind Socket



Connection has been made between Client and Server

While True:

if no connection is made:

Continue

else

socket.timeout(30)

Quit Program

3. List Menu() for User

If server receives nothing socket.timeout(5)

if server receives

continue

else

client resends user's choice

4. User enters choice

Example: user.choice('get')

5. Modify user's choice by appending 0's until 10 digits and making sure the filename and contents of the file are attached as one large string.

5. Open File.txt
   a. Read() → Save to string
   b. Calc_Bytes()
   c. SUCCESS/FAILURE

6. Back to Client to wait for new user's choice. ←