Name: Farica Mago, UCID:30111924, Assignment 4

How did you implement retransmissions? Explain how you setup the timer and when you start and stop the transmission timer.

To implement the retransmissions, I created the TimeoutHandler.java class in which extends TimerTask just like we do when we use threads.

```
class TimeoutHandler extends TimerTask
```

I overrided the function run() which will be automatically invoked to retransmit the packet when the timeout happens in the StopWaitFtp.java class.

```
@Override
   public void run()
   {try {
           System.out.println("timeout");
           clientSocket.send(packet);
           System.out.println("retx        "+seqNum);
       } catch (IOException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
       }}
```

To retransmit the packets, I passed the necessary arguments in the constructor of TimeoutHandler.java.

```
public TimeoutHandler(DatagramSocket clientSocket, DatagramPacket packet, int seqNum)
```

In the StopWaitFtp.java class, I created the object of the class Timer called timer, outside the while loop, so that we can close it afterwards when we close the sockets, and it does cause and infinite loop.

```
Timer timer = new Timer();
```

I have created the while loop which reads all the content of the file that is to be transmitted chunk by chunk. Inside this first while loop, I have created an object of TimerTask and with the help of this timertask, I have called timer.scheduleAtFixedRate() using appropriate arguments and this is when the timer starts.

```
TimerTask timertask = new TimeoutHandler(clientSocket, packet, seqNum);
timer.scheduleAtFixedRate(timertask,this.timeout,this.timeout);
```

After the inner while loop, I have cancelled timtertask, this is where the timer stops, when the appropriate ACK is received outside the while loop.

```
timertask.cancel();
```

The inner while loop stops if we want to jump to the next sequence number to be sent. It means we want to stop the previous timer and start a new timer.

```
while(seqNum == ackNum){clientSocket.receive(packet);
                    FtpSegment segment2 = new FtpSegment(packet);
                    ackNum = segment2.getSeqNum();
                }
```

What happens in your implementation if a timeout occurs at the same time that an ACK arrives from the server. Is there going to be a race condition in your program in that case? Explain your answer.

If timeout occurs at the same time the ACK arrives from the server, then the program will still retransmit the packet for which the timeout happened, and the server will drop that packet. The sequence number will be updated accordingly and the next packet will be sent.