QUESTION 1 – DISCUSSION OF STP PROTOCOL IMPLEMENTATION

To complete the STP Protocol, I have used Java and have 4 classes; Sender, Receiver, Packet and Log. Sender and Receiver both create packets and logs. Sender, the server in a client-server interaction, sends a file to the Receiver (i.e. the client) to thus create a unidirectional transfer of data between the two.

Sender

Sender accepts 8 input arguments from the user to send files to the Receiver. They are:

- Receiver host IP
- Receiver port
- The file to be transferred
- Maximum window size in bytes
- Maximum segment size in bytes
- Timeout in milliseconds
- The probability that the STP data segment will be dropped
- A seed value to generate a random number

The sender will then create a socket to send and receive STP segments through and initiate a three-way handshake.

Following this, the Sender will begin to transfer the file to the Receiver, breaking up the file into *mss* (maximum segment size) bytes and inserting this data into a Packet. The Sender will initially send a window of packets to the Receiver then begin to receive and instantaneously send new Packets through the specified socket to the Receiver, ensuring that the last byte sent minus the last byte that has not been acknowledged is less than or equal to the *mws* (maximum window size). That is $last\ byte\ sent - last\ byte\ acked \leq mws$. Throughout this process, the window that the Sender maintains contains packets that have yet to be acknowledged by the Receiver. If the Sender receives 3 duplicates of an ACK or times out, it will send the base of the window to complete a fast retransmit.

Before the file is transmitted, to simulate packet loss, the created packets are sent to a PLD method which will either drop or send files through comparing the *pdrop* (probability segment will be dropped) with a randomly generated number. Thus, all datagrams have probability *pdrop* of being dropped.

Upon completion, the Sender will complete a four-segment connection termination, closing the connection once the file has been fully transmitted.

All of the Sender's actions are sent to the Log to output.

Receiver

Receiver accepts 2 arguments: the receiver port and the file to send the transferred file to. The receiver responds to the handshake initiation from the Sender with a SYN+ACK, and upon receiving the final ACK of handshake, prepares to output the transferred file. Upon receiving a datagram, the Receiver will immediately generate and send an ACK. If datagrams arrive out of order, the Receiver will save these to a buffer to later reorder and sort once it receives the expected file and remove the correctly ordered packets from the buffer.Finally, once receiving the FIN segment from the Sender, the receiver will respond with FIN, ACK. (Implementation similar to Assignment specification section 3.5.3, sending FA).
All of the Receiver's actions are sent to the Log to output and the Receiver writes the transferred file once it has been reliably received.

Packet

This class contains the datagram packet's header and data (See Question 2 for more detail). For implementation in Java, the class contains Booleans to represent header segments, contains an ACK number and Sequence number, and maintains a byte array as the data segment.

Log

This class is used to create the logs for the Receiver and Sender, it creates and writes to a file *file.txt* (file name determined when an instance of the class is created).

Improvements and comments on implementation
The greatest area for error that stems from my implementation is a potentially incorrect timeout. If I had more time to do this assessment, I would work more with understanding and using threads and concurrency to implement the timeout so that when the base packet sends, a thread timer would begin and end when either the sender receives 3 duplicate ACKs or it times out. The impact of this issue can be seen in Question 3b where theoretically, the smaller the timeout, the more packets would be sent as a result of a large window and small timeout period.

QUESTION 2 – STP HEADER

Sequence Number		
Acknowledgement Number		
SYN	ACK	FIN
Data		

Size: (32 + *mss*)bytes

The STP Packet header contains the sequence number, and acknowledgement numbers as set during its time of creation. Furthermore, it contains SYN, ACK and FIN segments to allow the Sender/Receiver to know the packet type. To know that it is a Data packet, Sender/Receiver check that data is not of size 0. Data can be up to *mss* bytes large.

As it is implemented in Java, the SYN, ACK and FIN flags are all booleans (8 bytes each), the Sequence and Acknowledge numbers ints (4 bytes each) and Data *mss* bytes large.

QUESTION 3 – EXPERIMENTS

A) OBSERVATIONS AND EXPERIMENTING WITH TIMEOUT VALUES– TEXT1.TXT

To determine a suitable timeout value, there are a range of factors that need to be considered, but primarily the probability of packet dropping. The greater the probability of a packet dropping, the greater the number of packets dropped. Thus, a faster timeout would be preferable to ensure a faster total transmission time.

See Appendix A for experiment and results, with dropping/retransmitted packets indicated in red.

B) OBSERVATIONS AND EXPERIMENTING WITH TIMEOUT VALUES – TEST2.TXT

	Total Packets Sent (including retransmitted)	Total Time of Transfer (milliseconds)
Tcurrent = 500 milliseconds	46	3693
4 * Tcurrent = 2000 milliseconds	49	5477
Tcurrent/4 = 125 milliseconds	44	3686

As expected, the greater the timeout, the greater the total time of transfer.
As mentioned in Question 1, due to my implementation, we can see that the trend within the Total Packets Sent is not as expected. Instead, I would expect that the greater the timeout value, the smaller the number of packets sent due to spurious retransmissions.

APPENDIX

APPENDIX A

Results from Receiver after using pdrop = 0.1, mws =500 bytes, mss = 50 bytes, seed = 300

- Timeout = 100 milliseconds

```
Act    Time(ms)      Seg    Seq    Data   ACK
-----------------------------------------------
rcv    937.0         A      432    0      402
rcv    982.0         D      432    50     402
rcv    1005.0        D      482    50     402
rcv    1034.0        D      532    50     402
rcv    1062.0        D      582    50     402
rcv    1098.0        D      632    50     402
rcv    1131.0        D      682    50     402
rcv    1167.0        D      732    50     402
rcv    1198.0        D      782    50     402
rcv    1226.0        D      832    50     402
rcv    1295.0        D      882    50     402
rcv    1333.0        D      932    50     402
rcv    1383.0        D      982    50     402
rcv    1412.0        D      1032   50     402
rcv    1494.0        D      1082   50     402
rcv    1599.0        D      1132   50     402
rcv    1638.0        D      1182   50     402
rcv    1698.0        D      1232   50     402
rcv    1795.0        D      1282   50     402
rcv    1843.0        D      1332   50     402
rcv    1888.0        D      1382   50     402
rcv    1997.0        D      1432   50     402
rcv    2138.0        D      1482   50     402
rcv    2204.0        D      1532   50     402
rcv    2274.0        D      1582   50     402
rcv    2321.0        D      1632   50     402
rcv    2368.0        D      1682   50     402
rcv    2427.0        D      1732   50     402
rcv    2578.0        D      1832   50     402
rcv    2746.0        D      1882   50     402
rcv    2854.0        D      1982   43     402
rcv    2982.0        D      1782   50     402
rcv    3129.0        D      1932   50     402
rcv    3167.0        F      2025   0      402
rcv    3292.0        A      2026   0      403
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received: 32
Total Time: 2355ms
```

- Timeout = 500 milliseconds

```
Act    Time(ms)      Seg    Seq    Data   ACK
-----------------------------------------------
rcv    3766.0        A      625    0      279
rcv    3809.0        D      625    50     279
rcv    3835.0        D      675    50     279
rcv    3861.0        D      725    50     279
rcv    3892.0        D      775    50     279
rcv    3914.0        D      825    50     279
rcv    3949.0        D      875    50     279
rcv    3980.0        D      925    50     279
rcv    4009.0        D      975    50     279
rcv    4047.0        D      1025   50     279
rcv    4075.0        D      1075   50     279
rcv    4100.0        D      1125   50     279
rcv    4205.0        D      1175   50     279
rcv    4247.0        D      1225   50     279
```

```
rcv    4278.0        D      1275   50     279
rcv    4315.0        D      1325   50     279
rcv    4355.0        D      1375   50     279
rcv    4390.0        D      1425   50     279
rcv    4424.0        D      1475   50     279
rcv    4462.0        D      1525   50     279
rcv    4498.0        D      1575   50     279
rcv    4526.0        D      1625   50     279
rcv    4562.0        D      1675   50     279
rcv    4607.0        D      1725   50     279
rcv    4642.0        D      1775   50     279
rcv    4673.0        D      1825   50     279
rcv    4732.0        D      1875   50     279
rcv    4805.0        D      1925   50     279
rcv    4921.0        D      2025   50     279
rcv    5005.0        D      2075   50     279
rcv    5078.0        D      2175   43     279
rcv    5231.0        D      1975   50     279
rcv    5777.0        D      2125   50     279
rcv    5802.0        F      2218   0      279
rcv    5853.0        A      2219   0      280
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received: 32
Total Time: 2142ms
```

- Timeout=1000 milliseconds

```
Act    Time(ms)Seg   Seq    Data   ACK
----------------------------------------
rcv    6654.0 A      621    0      664
rcv    6702.0 D      621    50     664
rcv    6733.0 D      671    50     664
rcv    6759.0 D      721    50     664
rcv    6788.0 D      771    50     664
rcv    6815.0 D      821    50     664
rcv    6846.0 D      871    50     664
rcv    6877.0 D      921    50     664
rcv    6913.0 D      971    50     664
rcv    6944.0 D      1021   50     664
rcv    6973.0 D      1071   50     664
rcv    7000.0 D      1121   50     664
rcv    7034.0 D      1171   50     664
rcv    7067.0 D      1221   50     664
rcv    7109.0 D      1271   50     664
rcv    7147.0 D      1321   50     664
rcv    7188.0 D      1371   50     664
rcv    7235.0 D      1421   50     664
rcv    7279.0 D      1471   50     664
rcv    7318.0 D      1521   50     664
rcv    7353.0 D      1571   50     664
rcv    7388.0 D      1621   50     664
rcv    7446.0 D      1671   50     664
rcv    7570.0 D      1721   50     664
rcv    7606.0 D      1771   50     664
rcv    7679.0 D      1821   50     664
rcv    7732.0 D      1871   50     664
rcv    7827.0 D      1921   50     664
rcv    7991.0 D      2021   50     664
rcv    8049.0 D      2071   50     664
rcv    8132.0 D      2171   43     664
rcv    8253.0 D      1971   50     664
rcv    9309.0 D      2121   50     664
rcv    9331.0 F      2214   0      664
rcv    9395.0 A      2215   0      665
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received: 32
Total Time: 2741ms
```

Results from Receiver after using pdrop = 0.3, mws =500 bytes, mss = 50 bytes, seed = 300, timeout=500 milliseconds

```
Act    Time(ms) Seg   Seq    Data   ACK
--------------------------------------
rcv    3183.0 A      473    0      529
rcv    3219.0 D      473    50     529
rcv    3251.0 D      523    50     529
rcv    3277.0 D      623    50     529
rcv    3301.0 D      673    50     529
rcv    3348.0 D      773    50     529
rcv    3401.0 D      923    50     529
rcv    3433.0 D      973    50     529
rcv    3473.0 D      1023   50     529
rcv    3566.0 D      573    50     529
rcv    3682.0 D      1073   50     529
rcv    4215.0 D      723    50     529
rcv    4279.0 D      1123   50     529
rcv    4823.0 D      823    50     529
rcv    4863.0 D      1173   50     529
rcv    5408.0 D      873    50     529
rcv    5970.0 D      1223   50     529
rcv    6002.0 D      1273   50     529
rcv    6052.0 D      1323   50     529
rcv    6095.0 D      1373   50     529
rcv    6137.0 D      1423   50     529
rcv    6184.0 D      1473   50     529
rcv    6230.0 D      1523   50     529
rcv    6791.0 D      1573   50     529
rcv    6830.0 D      1623   50     529
rcv    7914.0 D      1673   50     529
rcv    7964.0 D      1723   50     529
rcv    8009.0 D      1773   50     529
rcv    8058.0 D      1823   50     529
rcv    8094.0 D      1873   50     529
rcv    8144.0 D      1923   50     529
rcv    8188.0 D      1973   50     529
rcv    8233.0 D      2023   43     529
rcv    8260.0 F      2066   0      529
rcv    8315.0 A      2067   0      530
Amount of (original) Data Received (in bytes): 1593
Number of (original) Data Segments Received: 32
Total time: 5132ms
```