COMP3331 Project - Report

By: Hashimi-Mahmood Chau (z5242398)

Question 1

I have implemented the PTP protocol using Java version 11. Other than the two required files, i.e. *Sender.java* and *Receiver.java*, I have also made separate classes such as the following:

- *Globals.java* - contains global variables which are shared by the sending program and its associated threads
- *Helper.java* - contains static methods used by various classes to help with generic functions like finding the difference in time between two given time values, retransmitting packets, etc
- *Logger.java* - contains methods to write formatted lines to either Sender_log.txt or Receiver_log.txt, depending on the given OutputStream
- *Packet.java* - provides a template for representing individual packets, which assists in manipulating them for various purposes such as sending them, receiving them, and storing them in buffers (which are implemented as ArrayLists of Packet objects)
- *SenderReceiveThread.java* - implements a thread at the sender side that receives acknowledgement packets from the receiver
- *SenderSendThread.java* - implements a thread at the sender side that sends data packets to the receiver

Based on the list of required features given in the spec, I have successfully implemented the following:

- Three-way handshake for connection establishment
- Three-segment connection termination (FIN, FIN+ACK, ACK)
- Sender maintains a single timer for timeout operation
- Sender implements the simplified TCP sender in Figure 3.33 of the textbook
- Sender implements fast retransmit
- Sender has a buffer that contains MWS/MSS number of packets
- Receiver has a buffer for storing out-of-order packets
- PTP headers for each packet contain multiple fields including sequence numbers and acknowledgement numbers
- Receiver sends cumulative acknowledgements
- Receiver immediately acknowledges all segments received

- PTP packets contain at most MSS number of bytes of data
- MWS is implemented such that it does not include header sizes

In summary, I have implemented all the required features of PTP which were given in the assignment specification.

Question 2

Detailed diagram of PTP header:

<table>
<tr><td colspan="4">Sequence number</td></tr>
<tr><td colspan="4">Acknowledgement number</td></tr>
<tr><td>ACK flag</td><td>SYN flag</td><td>FIN flag</td><td>Maximum ...</td></tr>
<tr><td colspan="3">... Segment Size (MSS)</td><td>Maximum ...</td></tr>
<tr><td colspan="3">... Window Size (MWS)</td><td></td></tr>
</table>

Note that the width of this table is 4 bytes (32 bits) just like Figure 3.29 in the textbook.

The PTP segment header contains the following fields:

- The 4-byte (32-bit) sequence number - used in implementing a reliable data transfer service
- The 4-byte (32-bit) acknowledgement number - used in implementing a reliable data transfer service
- The 1-byte (8-bit) ACK flag field - indicates that the segment contains an acknowledgement for a segment that has been successfully received
- The 1-byte (8-bit) SYN flag field - used for connection setup
- The 1-byte (8-bit) FIN flag field - used for connection teardown
- The 4-byte (8-bit) MSS field - used to inform the receiver of the Maximum Segment Size value from the sender
- The 4-byte (8-bit) MWS field - used to inform the receiver of the Maximum Window Size value from the sender

Note that this header is 19-bytes in size.

Question 3

Part (a)

To determine a suitable value for timeout, we must consider a few key concepts. If we assume that the round trip time (RTT) is fairly consistent throughout the PTP "connection" (which we will call AveragedRTT), then we can use this RTT to find a suitable timeout interval. This interval should be, at least, AveragedRTT. If this isn't the case, the PTP timer would time out too quickly, leading to many unnecessary retransmissions. Additionally, the timeout interval should not be too much larger than AveragedRTT. If this occurs, then when a PTP segment is lost, this segment would not be retransmitted rapidly enough as the timeout interval is too large, causing larger delays in data transfer. Thus, the optimal timeout would be just slightly larger than the AveragedRTT.

To evaluate the AveragedRTT, several timeout values were tested using the following parameter settings: pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, and seed = 300. Figure 1 shows a sample of sent packets and their acknowledgements using the given parameters and a timeout value of 4 milliseconds (ms). Using timeout values of 40ms and 400ms, we get very similar results for the estimated RTT (which are not shown in the Appendix). As we can see in this Figure, the average amount of time between when a PTP data segment is sent and when an acknowledgement for that same segment is received (or its RTT) is approximately 100ms.

For example, in Figure 1, the packet with sequence number 101 is sent from the sender about 1293.726ms after the sender is started. Its acknowledgement packet, with acknowledgement number 151, is received at the sender approximately 1393.077ms after the sender is initiated. Subtracting 1293.726ms from 1393.077ms, we get 99.351ms which is very close to 100ms. Performing this same reasoning to the other packets in Figure 1, we can see from the sample that the RTT consistently stabilises at about 100ms.

Hence, this value of 100ms would be equivalent to our AveragedRTT in our explanation above. Thus, a good approximate timeout value would be slightly larger than this RTT. From this result, we arbitrarily chose a timeout value of 120ms as it is slightly higher than 100ms.

With this selected timeout value, an experiment was run by transferring the file test1.txt. A sample of the sequence of PTP packets observed at the receiver is shown in Figure 2. A sample of where dropping occurred is shown in Figure 4. A comparison of the total packets dropped compared with the total packets sent using pdrop = 0.1 is shown in Figure 6.

An additional experiment was run with pdrop = 0.3, transferring the same file. A sample of the sequence of PTP packets observed at the receiver is shown in Figure 3. A sample of where dropping occurred is shown in Figure 5. A comparison of the total packets dropped compared with the total packets sent using pdrop = 0.3 is shown in Figure 7.

We can see in Figure 4 that when given pdrop = 0.1, about 1 in every 10 packets is dropped. This is confirmed in Figure 6, where 66 packets are dropped but 656 packets are sent. In this instance, the actual drop rate is *(66/656) * 100% = 10.06%* to two decimal places.

Furthermore, we can see in Figure 5 that when given pdrop = 0.3, about 1 in every 3 packets is dropped. This is also confirmed in Figure 7, where 217 packets are dropped but 656 packets are sent. In this case, the actual drop rate is *(217/656) * 100% = 33.08%* to two decimal places.

Part (b)

If Tcurrent represents the timeout value that was chosen in part (a), then Tcurrent = 120ms. Setting pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, and seed = 300, three experiments were run with the following different timeout values:

- Tcurrent = 120ms
- 4 * Tcurrent = 4 * 120ms = 480ms
- Tcurrent / 4 = 120ms / 4 = 30ms.

The results are shown in the table below:

Timeout value (in milliseconds)	Number of PTP packets transmitted (including retransmissions)	Length of overall transfer (in milliseconds)
30	10453	1611488.872
120	10445	1052370.329
480	10459	2161145.142

Table 1: a comparison of the number of packets sent from the sender and the length of transfer using different timeout values

Figures 8, 9, and 10 show the actual data in Sender_log.txt that this table is based on.

Discussion of results

Using our reasoning at the beginning of part (a), we figured out that 120ms was a suitable timeout interval for sending packets. In Table 1, we can observe that this is true as using a timeout interval of 120ms results in the fastest overall transfer among the different timeout values. The reasons for this are similar to the reasons given at the start of part (a).

A timeout value of 30ms is too quick as it is much smaller than the AveragedRTT of 100ms. This means that timeouts occur even if the original transmission is on its way to the receiver, or its acknowledgement is on its way to the sender, resulting in unnecessary retransmissions.

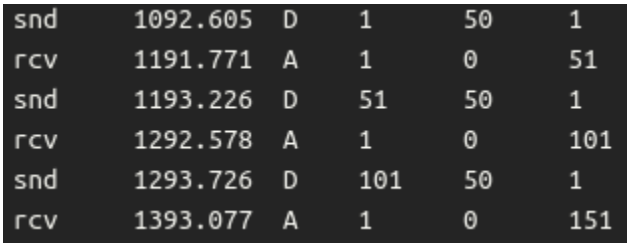
Ultimately, slower data transfer rates occur as a larger proportion of the PTP connection becomes “swamped” by the unnecessary retransmissions of data. This is observed in Table 1, where a timeout value of 30ms results in a data transfer time that is slower than when a timeout value of 120ms is used.

On the other hand, a timeout value of 480ms is too slow as it is much larger than the AveragedRTT of 100ms. This means that when a packet is lost, it takes a relatively long time for the PTP sender to detect such a packet loss due to the high timeout interval. As such, it takes longer to retransmit those packets which eventually leads to slower data transfer. This is seen in Table 1, where a timeout value of 480ms leads to the longest data transfer time.

Consequently, our chosen timeout value of 120ms is the most suitable, and optimal, of the three different timeout values.

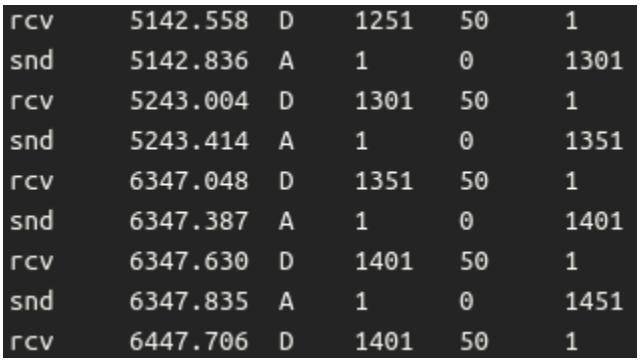
Note: you may realise from our reasoning that, even though a low timeout value of 30ms should produce the most retransmissions, and hence the most transmissions overall out of the three timeout values, it does not. This is because, while a high timeout value of 480ms may produce fewer retransmissions due to timeouts, it also sends out more original transmissions at once since it is less “sensitive” to such timeouts. If packet loss occurs, the receiver receives out-of-order packets, which leads to it sending duplicate acknowledgements; if the sender receives three of these, it uses fast retransmit to send a retransmission of the oldest unacknowledged packet. It is this feature of PTP that leads to the seemingly high number of transmitted packets observed when using a timeout value of 480ms.

Appendix



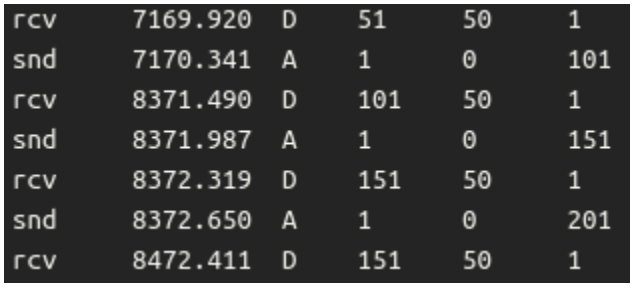
snd	1092.605	D	1	50	1
rcv	1191.771	A	1	0	51
snd	1193.226	D	51	50	1
rcv	1292.578	A	1	0	101
snd	1293.726	D	101	50	1
rcv	1393.077	A	1	0	151

Figure 1: a sample of sent packets and their acknowledgements in Sender_log.txt



rcv	5142.558	D	1251	50	1
snd	5142.836	A	1	0	1301
rcv	5243.004	D	1301	50	1
snd	5243.414	A	1	0	1351
rcv	6347.048	D	1351	50	1
snd	6347.387	A	1	0	1401
rcv	6347.630	D	1401	50	1
snd	6347.835	A	1	0	1451
rcv	6447.706	D	1401	50	1

Figure 2: a sample of packets received and acknowledgements sent in Receiver_log.txt using pdrop = 0.1



rcv	7169.920	D	51	50	1
snd	7170.341	A	1	0	101
rcv	8371.490	D	101	50	1
snd	8371.987	A	1	0	151
rcv	8372.319	D	151	50	1
snd	8372.650	A	1	0	201
rcv	8472.411	D	151	50	1

Figure 3: a sample of packets received and acknowledgements sent in Receiver_log.txt using pdrop = 0.3

```
drop    98800.010  D    32251  50     1
rcv     98805.692  A    1      0      32251
snd     98900.166  D    32251  50     1
snd     98900.284  D    32301  50     1
rcv     98905.825  A    1      0      32301
snd     99000.433  D    32301  50     1
snd     99000.520  D    32351  50     1
rcv     99005.998  A    1      0      32351
snd     99100.665  D    32351  50     1
snd     99100.783  D    32401  50     1
rcv     99106.164  A    1      0      32401
snd     99200.939  D    32401  50     1
snd     99201.051  D    32451  50     1
rcv     99206.365  A    1      0      32451
snd     99301.211  D    32451  50     1
drop    99301.305  D    32501  50     1
```

Figure 4: a sample of 1 in 10 packets dropping using pdrop = 0.1

```
drop    1519.740  D    251    50     1
rcv     1618.348  A    1      0      251
snd     1620.121  D    251    50     1
snd     1621.238  D    301    50     1
rcv     1718.930  A    1      0      301
snd     1721.600  D    301    50     1
drop    1721.937  D    351    50     1
rcv     1819.513  A    1      0      351
snd     1822.458  D    351    50     1
drop    1822.745  D    401    50     1
```

Figure 5: a sample of approximately 1 in 3 packets dropping using pdrop = 0.3

```
Number of Data Segments Sent (excluding retransmissions): 656
Number of (all) Packets Dropped (by the PL module): 66
```

Figure 6: a comparison of the total packets dropped compared with the total packets sent using pdrop = 0.1

```
Number of Data Segments Sent (excluding retransmissions): 656
Number of (all) Packets Dropped (by the PL module): 217
```

Figure 7: a comparison of the total packets dropped compared with the total packets sent using pdrop = 0.3

```
snd   1611488.872  A    262146  0      2
Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 479
Number of Retransmitted Segments: 5210
```

Figure 8: summary of stats using timeout = 30ms

```
snd   1052370.329  A    262146  0      2
Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 479
Number of Retransmitted Segments: 5202
```

Figure 9: summary of stats using timeout = 120ms

```
snd   2161145.142  A    262146  0      2
Amount of (original) Data Transferred (in bytes): 262144
Number of Data Segments Sent (excluding retransmissions): 5243
Number of (all) Packets Dropped (by the PL module): 479
Number of Retransmitted Segments: 5216
```

Figure 10: summary of stats using timeout = 480ms