1. Appendix

Name: Ping Fan
zNumber: z3491689

2. STP protocol implementation

Sender:

In the Sender part, we need to keep send a window's all segments at every time, at the same time we need to open a listener (thread) to keep listen Ack package send from the Receiver Side. Also another thread named Resender (in charge of retransmit the dropped packets). The detail are in the flow chart below

Receiver:

Receiver have a buffer to buffer all received segments' data. I use a variable named **SendAck** to record what's the next segment I want to receive from Sender. e.g. the Sender sends 2,3,4,5 and 3 is lost. So even though the 4, 5 have arrived Receiver successfully, the Receiver will send Ack segment with AckNum = 3. When 3 is received, it will update the SendAck = 6, because 4, 5 have already in the buffer.

Connection establish:

Three-way hand shaking

The Sender send a SYN then waiting for receiver's SYN&ACK.

Lastly Sender send a SYN, and the connection is established

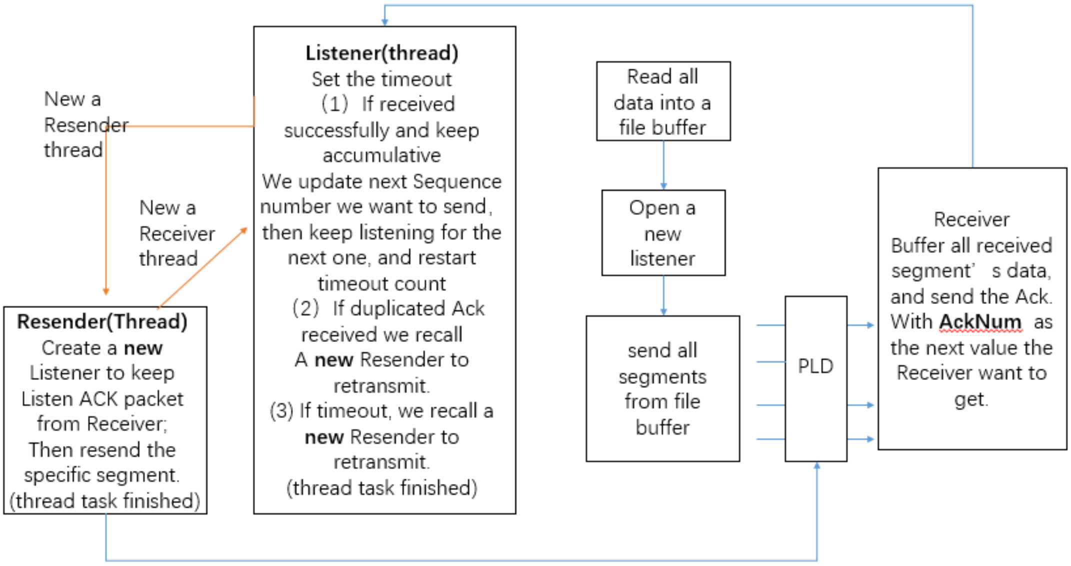
Connection closure:

The sender sends a Fin to inform the receiver

Then the Receiver send a Ack and a Fin

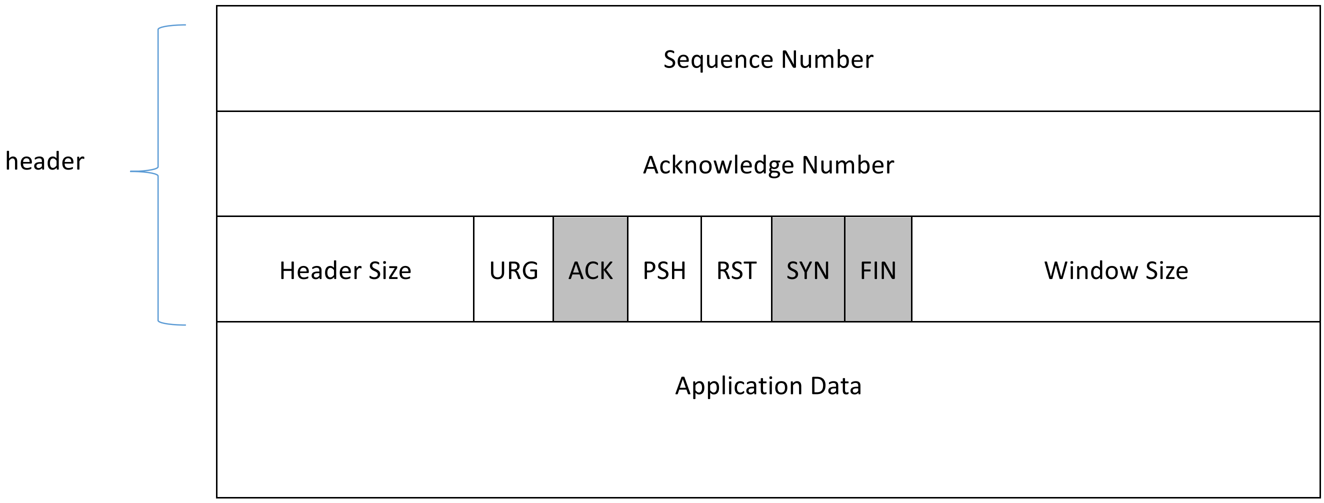
Lastly the Sender send a Ack and wait for a while then closed (In the waiting time I write the log file so I do not set a wait clock, just use the write log file's time as the wait time).

The flow chart below has list most feature in the programming.



New a Resender thread
New a Receiver thread
Resender(Thread)
Create a new Listener to keep Listen ACK packet from Receiver;
Then resend the specific segment.
(thread task finished)
Listener(thread)
Set the timeout
(1) If received successfully and keep accumulative
We update next Sequence number we want to send, then keep listening for the next one, and restart timeout count
(2) If duplicated Ack received we recall A new Resender to retransmit.
(3) If timeout, we recall a new Resender to retransmit.
(thread task finished)
Read all data into a file buffer
Open a new listener
send all segments from file buffer
PLD
Receiver
Buffer all received segment' s data, and send the Ack. With AckNum as the next value the Receiver want to get.

3. STP Header Structure



Overall 12bytes:

(1) Sequence Number: occupy 4 bytes

(2) Acknowledge Number: occupy 4 bytes

(3) Header Size: occupy 1 byte, because Header Size is 12 bytes, 1 byte is enough to store the value

(4) All flag together: occupy 1 byte, low 6 bits [URG, ACK, PSH, RST, SYN, FIN]

We only use ACK, SYN and Fin (Ack# valid, setup and teardown) in STP. e.g. If only ACK is set, the byte value is 1<<4; if ACK and SYN are set, the byte value is 1<<4 + 1<<1

(5) Window size: occupy 2 bytes, used for flowing control

In the programming

Class Segment has some methods could be used to facilitate the operation according the header:

private member:

Sequence Number
Acknowledge Number
Header Size
ACK flag
SYN flag
FIN flag
Window Size

Constructor: assign each private member with corresponding value.

Method:

createHeader() could transfer private members into bytes and store them into a byte array.

4. Question Answer

(a)

Firstly, timeout value must greater than RTT, secondly the probability of drop is small only 0.1. I check the RTT by looking at the time from Sender sends a SYN until it received a SYN & ACK.

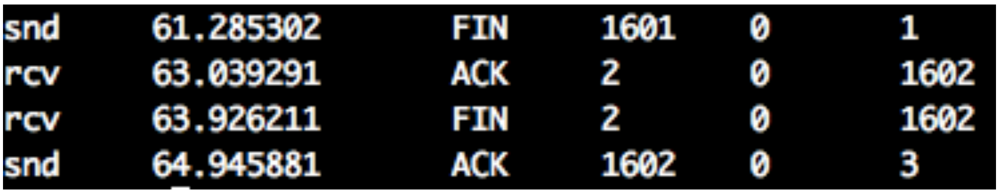
So, I estimate the timeout is no need to be very large.

(1) timeout = 8

java Sender 127.0.0.1 2222 test1.txt 500 50 9 0.1 300

java Receiver 2222 res1.txt

(complete data please check the Sender_log.txt)



```
snd     61.285302     FIN    1601    0      1
rcv     63.039291     ACK    2       0      1602
rcv     63.926211     FIN    2       0      1602
snd     64.945881     ACK    1602    0      3
```

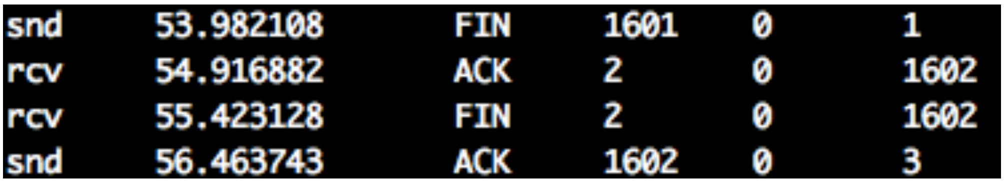
we can find before 61ms, the file has already transmitted completely.

(2) timeout = 10

java Sender 127.0.0.1 2222 test1.txt 500 50 10 0.1 300

java Receiver 2222 res1.txt

(complete data please check the Sender_log.txt)



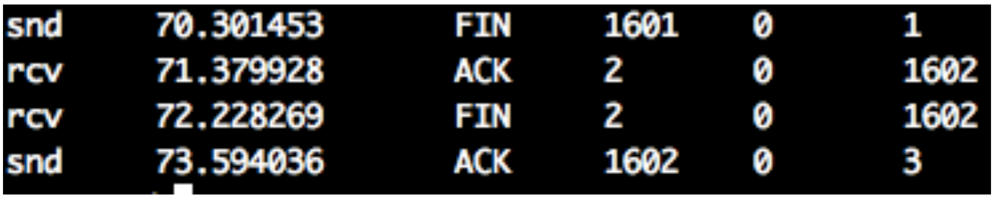
```
snd     53.982108     FIN    1601    0      1
rcv     54.916882     ACK    2       0      1602
rcv     55.423128     FIN    2       0      1602
snd     56.463743     ACK    1602    0      3
```

we can find before 53ms, the file has already transmitted completely.

(3) timeout = 11

java Sender 127.0.0.1 2222 test1.txt 500 50 11 0.1 300

java Receiver 2222 res1.txt



```
snd     70.301453     FIN    1601    0      1
rcv     71.379928     ACK    2       0      1602
rcv     72.228269     FIN    2       0      1602
snd     73.594036     ACK    1602    0      3
```

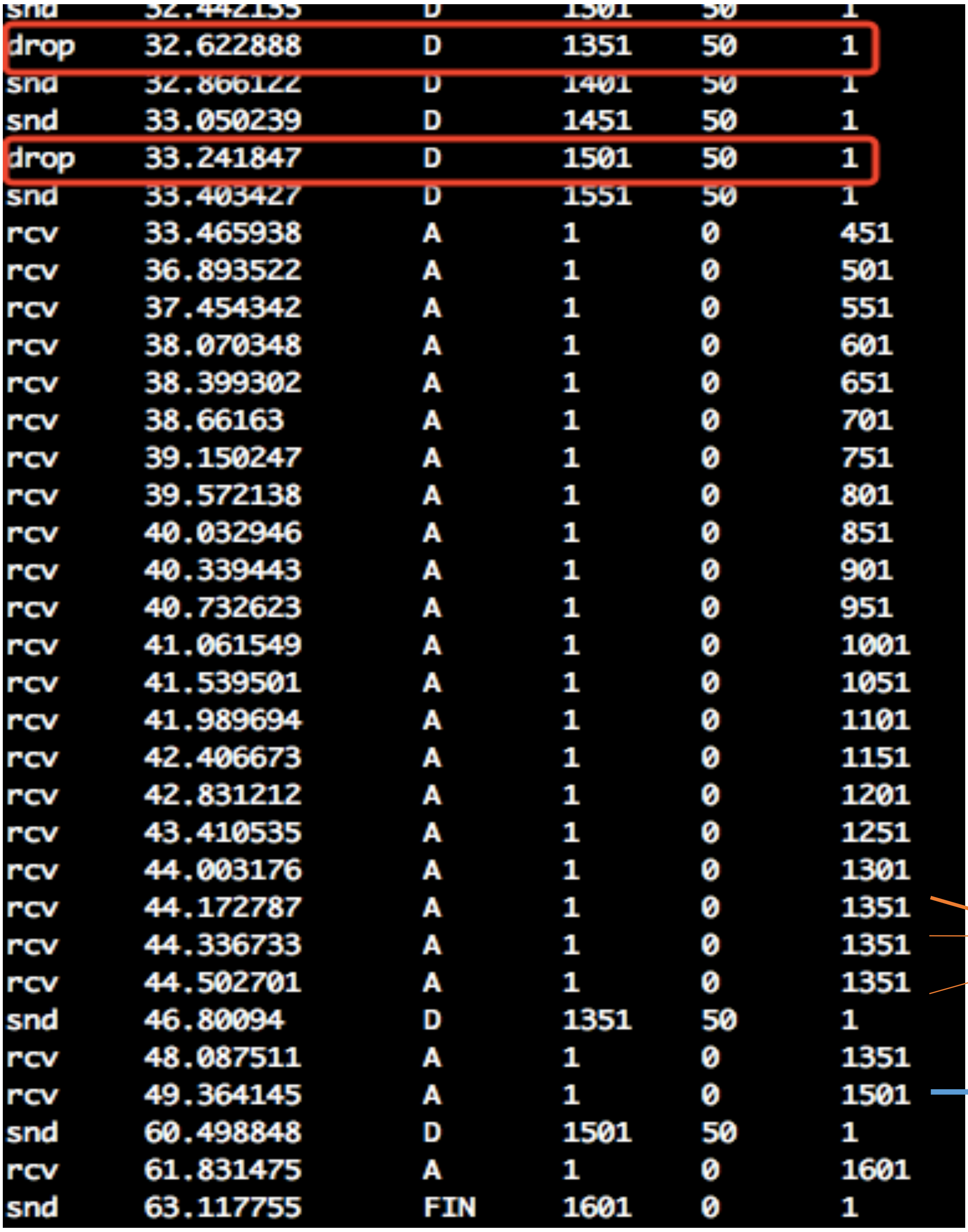
we can find before 70ms, the file has already transmitted completely.

So, **the most suitable timeout = 10**;

Experiment:

the Send_log.txt of timeout = 10, pdrop = 0.1, MWS = 500 bytes, MSS = 50 bytes, seed = 300

(complete log please check the Sender_log.txt, below is partial)



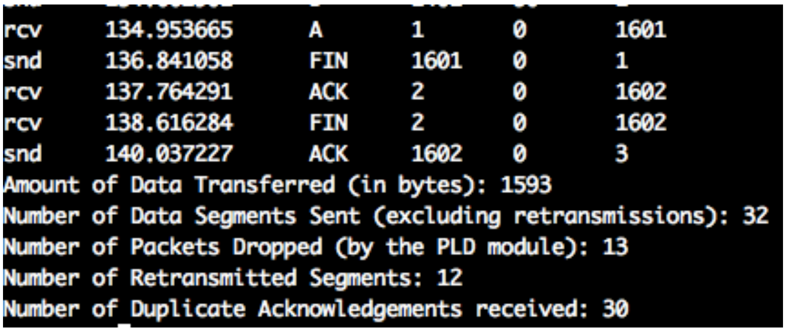
```
snd     32.442135     D      1301    50     1
drop    32.622888     D      1351    50     1
snd     32.866122     D      1401    50     1
snd     33.050239     D      1451    50     1
drop    33.241847     D      1501    50     1
snd     33.403427     D      1551    50     1
rcv     33.465938     A      1       0      451
rcv     36.893522     A      1       0      501
rcv     37.454342     A      1       0      551
rcv     38.070348     A      1       0      601
rcv     38.399302     A      1       0      651
rcv     38.66163      A      1       0      701
rcv     39.150247     A      1       0      751
rcv     39.572138     A      1       0      801
rcv     40.032946     A      1       0      851
rcv     40.339443     A      1       0      901
rcv     40.732623     A      1       0      951
rcv     41.061549     A      1       0      1001
rcv     41.539501     A      1       0      1051
rcv     41.989694     A      1       0      1101
rcv     42.406673     A      1       0      1151
rcv     42.831212     A      1       0      1201
rcv     43.410535     A      1       0      1251
rcv     44.003176     A      1       0      1301
rcv     44.172787     A      1       0      1351
rcv     44.336733     A      1       0      1351
rcv     44.502701     A      1       0      1351
snd     46.80094      D      1351    50     1
rcv     48.087511     A      1       0      1351
rcv     49.364145     A      1       0      1501
snd     60.498848     D      1501    50     1
rcv     61.831475     A      1       0      1601
snd     63.117755     FIN    1601    0      1
```

duplicate Acks

timeout so resend

Additional experiment:

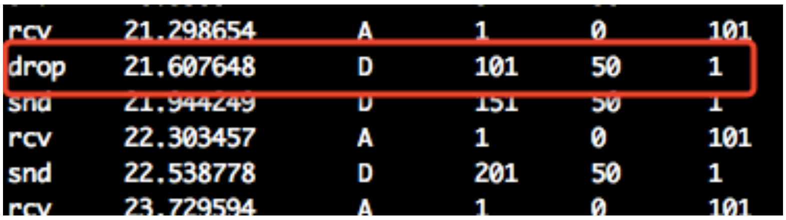
(complete log please check the Sender_log.txt, below is partial)



```
rcv     134.953665      A       1       0       1601
snd     136.841058      FIN     1601    0       1
rcv     137.764291      ACK     2       0       1602
rcv     138.616284      FIN     2       0       1602
snd     140.037227      ACK     1602    0       3
Amount of Data Transferred (in bytes): 1593
Number of Data Segments Sent (excluding retransmissions): 32
Number of Packets Dropped (by the PLD module): 13
Number of Retransmitted Segments: 12
Number of Duplicate Acknowledgements received: 30
```

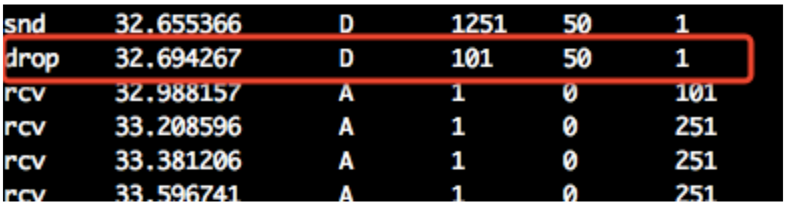
we could find there are 13 Packets dropped, and retransmitted segment 12, which means there one segment dropped twice.

first time:



```
rcv     21.298654       A       1       0       101
drop    21.607648       D       101     50      1
snd     21.944249       D       151     50      1
rcv     22.303457       A       1       0       101
snd     22.538778       D       201     50      1
rcv     23.729594       A       1       0       101
```

second time



```
snd     32.655366       D       1251    50      1
drop    32.694267       D       101     50      1
rcv     32.988157       A       1       0       101
rcv     33.208596       A       1       0       251
rcv     33.381206       A       1       0       251
rcv     33.596741       A       1       0       251
```

And the pdrop increase , the chance of retransmit also increase. So, the overall time increases.

(b)

java Sender 127.0.0.1 2222 test2.txt 500 50 10 0.1 300

	Tcurrent = 10	4 * Tcurrent = 40	Tcurrent / 4 = 2
package number	40 + 5 = 45	40 + 5 = 45	40 + 5 = 45
Overall time	38.205509	46.550078	33.780733

Because the MSS, seed and file not changed, so STP package number is unchanged. In terms of overall time, if a segment is dropped, and received 3 same ACKs, the Sender would reset the timer and resend the segment again:

if time out interval is very small (Tcuurent/4) Sender could hard to get 3 duplicates, but resend it after time out;

if time out interval is large (4 * Tcurrent), Sender could easier to get 3 duplicates and resend. But if Sender didn't get 3 duplicates, the segment will wait the time out finished, then it could be resend and timer will be reset.

So, if time out is too small, it would frequently resend the lost segment, which wastes the bandwidth. Reversely, if time out is large, it would waste time. So using time out and fast retransmit together could find a balanced way in this situation. Even though it may not be the best one, but it could fit most situations.