Lu Lu

Program 3 Report

Build Script

Build Server file:

```
[lulu1031@csslab2 P2]$ g++ UdpSocket.cpp Timer.cpp hw2.cpp -o hw2
```

Build Client file:

```
[lulu1031@csslab3 P2]$ g++ UdpSocket.cpp Timer.cpp hw2.cpp -o hw2
```

(Make sure to run server first)

Run Server: `[lulu1031@csslab2 P2]$ ./hw2`

Run Client: `[lulu1031@csslab3 P2]$ ./hw2 csslab2 > data`

Port Used: 75118

Test1(Unreliable):

The Client will send 2000 messages to the Server unreliable. The elapsed time may vary.

Avg: 345946 usec

UDP unreliable is the fastest test of three. Messages may drop through the transmitting, but it was not detected. Base UDP is frequently used in streaming or live videos as it only sending messages from one to the other and do not require feedback.

```
[lulu1031@csslab2 P2]$ ./hw2
Choose a testcase
   1: unreliable test
   2: stop-and-wait test
   3: sliding windows
--> 1
20000 messages received
finished
```

```
1 20000 messages sent.
2 Elasped time = 568252
3 finished
4
```

```
20000 messages sent.
Elasped time = 237867
finished

```

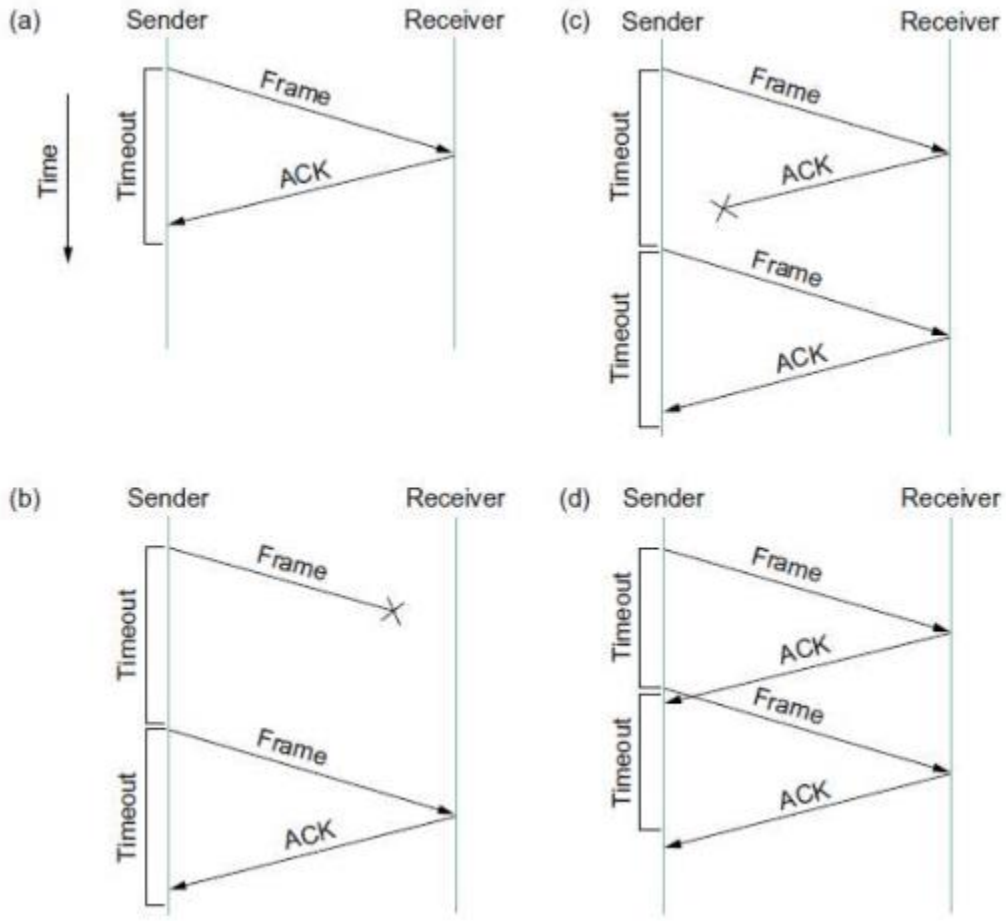
```
20000 messages sent.
Elasped time = 231720
finished

```

Test2(Stop and Wait):

Retransmits is happening in stop and wait. It receiver does not receive expected message in an amount of time with sequence, it will ask for retransmits.

Algorithm:



Sender will send frame to the receiver and start timer, once receiver receive the frame, it will send back ACK to the sender. For b), if frame dropped in transmitting, receiver won't receive it and unable to send ACK back, sender will retransmit frame again after a period of time(timeout).

For c), if ACK get lost in transmitting, sender won't receive an ACK, it will assume the packets lost in transmitting and retransmit the frame to the receiver. For d), if Sender didn't receive ACK back in timeout period, it will assume packet lost and resend it, receiver will send ACK regarding to the new Frame.

Avg: 2056433 usec

Stop and Wait is slower than unreliable, but it can detect if packets dropped or not. If one missing, the sender will notice and resend it. Even it is slower, it can make sure that receiver receives all the packets.

Then con for this is that sender must wait for a time period and do nothing, it is not efficiency enough for network communicating. It is reliable and making sure that nothing will be missed by both sides.

```
[lulu1031@csslab2 P2]$ ./hw2
Choose a testcase
   1: unreliable test
   2: stop-and-wait test
   3: sliding windows
--> 2
finished
```

```
1 Elasped time = 2006888
2 retransmits = 9
3 finished
4
```

```
Elasped time = 1983160
retransmits = 1
finished
```

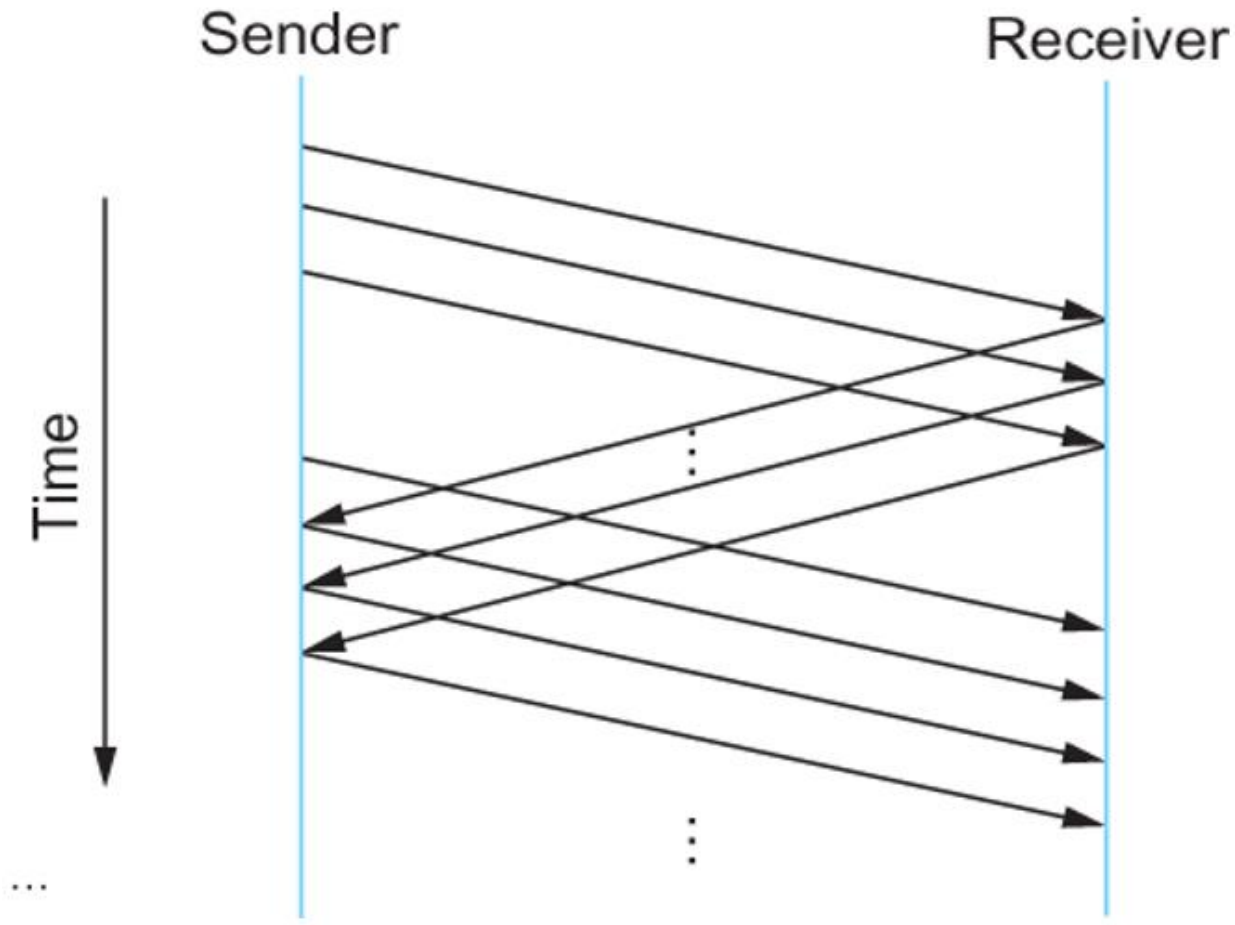
```
1 Elasped time = 2120403
2 retransmits = 27
3 finished
4
```

```
Elasped time = 2115284
retransmits = 3
finished
```

Test3(Sliding Window):

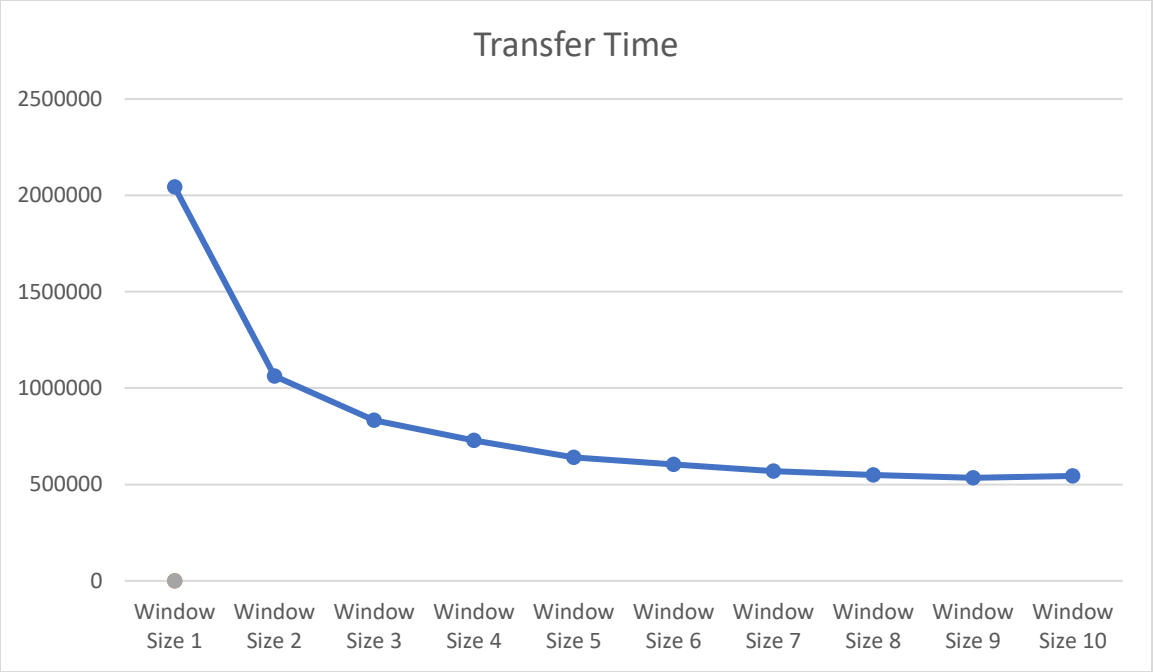
The client will continue sending messages to the server in a windows size frame no matter received ACK or not. But after timeout, sender didn't receive ACK back, it will retransmit the packets. It is much more efficiently than Stop and Wait as the sending continuing sending message while waiting.

Algorithm:



Average elapsed time for each window size:

Window Size(1)	Window Size(2)	Window Size(3)	Window Size(4)	Window Size(5)	Window Size(6)	Window Size(7)	Window Size(8)	Window Size(9)	Window Size(10)
2042897	1061708	832891	727855	641313	604375	569983	550060	534644	544920



The bigger window size, the elapsed time is slower in a slope like graph above. By comparing for three tests, sliding window will be the most efficiency way in transmitting data. Window size in 3,4 is preferred as the more window size increases, the slope of decrease is going slower. Retransmit rate in Stop and wait are averagely high, while in sliding window, window size between 2-4 is the lowest size, while the bigger size window may have larger rate in retransmits.

If communicating without feedback, unreliable transmitting will be fasted way to send data, otherwise, sliding window can increase the efficiency in communicating.

```
[lulu1031@csslab2 P2]$ ./hw2
Choose a testcase
   1: unreliable test
   2: stop-and-wait test
   3: sliding windows
--> 3
finished
```

```
Window size = 1 Elasped time = 2008966
retransmits = 0
Window size = 2 Elasped time = 1037359
retransmits = 2
Window size = 3 Elasped time = 833363
retransmits = 0
Window size = 4 Elasped time = 700104
retransmits = 5
Window size = 5 Elasped time = 623777
retransmits = 0
Window size = 6 Elasped time = 620215
retransmits = 72
```

```
Window size = 1 Elasped time = 2076777
retransmits = 0
Window size = 2 Elasped time = 1037308
retransmits = 0
Window size = 3 Elasped time = 810329
retransmits = 9
Window size = 4 Elasped time = 688443
retransmits = 0
Window size = 5 Elasped time = 634526
retransmits = 11
Window size = 6 Elasped time = 605396
retransmits = 18
```

```
Window size = 1 Elasped time = 2153879
retransmits = 14
Window size = 2 Elasped time = 1137165
retransmits = 2
Window size = 3 Elasped time = 840000
retransmits = 9
Window size = 4 Elasped time = 763974
retransmits = 8
```

```
Window size = 1 Elasped time = 1984399
retransmits = 2
Window size = 2 Elasped time = 1057819
retransmits = 0
Window size = 3 Elasped time = 843187
retransmits = 3
Window size = 4 Elasped time = 762100
retransmits = 0
Window size = 5 Elasped time = 658102
retransmits = 15
Window size = 6 Elasped time = 587516
retransmits = 0
Window size = 7 Elasped time = 569983
retransmits = 0
Window size = 8 Elasped time = 550060
retransmits = 0
Window size = 9 Elasped time = 534644
retransmits = 0
Window size = 10 Elasped time = 544920
retransmits = 60
```

```
Window size = 1 Elasped time = 1990465
retransmits = 3
Window size = 2 Elasped time = 1038889
retransmits = 2
Window size = 3 Elasped time = 837579
retransmits = 0
Window size = 4 Elasped time = 724655
retransmits = 28
Window size = 5 Elasped time = 648850
retransmits = 10
```