2016311739 김연재

Report

1. Development Environment information

Version of operating system: window10

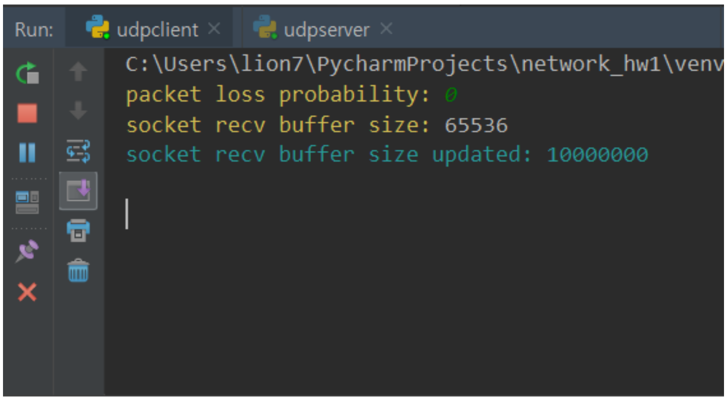
Languages: python

Compiler/interpreter version: python 3.6

2. How to run sender and receiver programs.

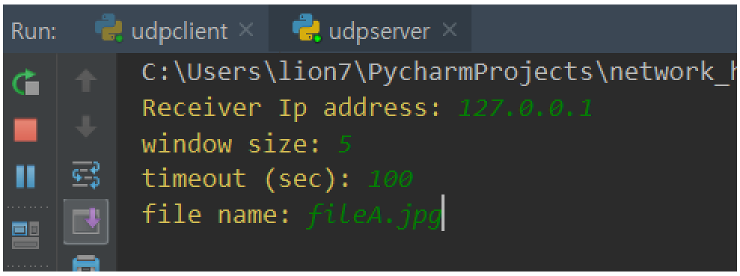
Please follow the order to run safely

1. Run "udpclient.py" in directory 'PA3/client', first!!
2. Type input for "udpclient.py"



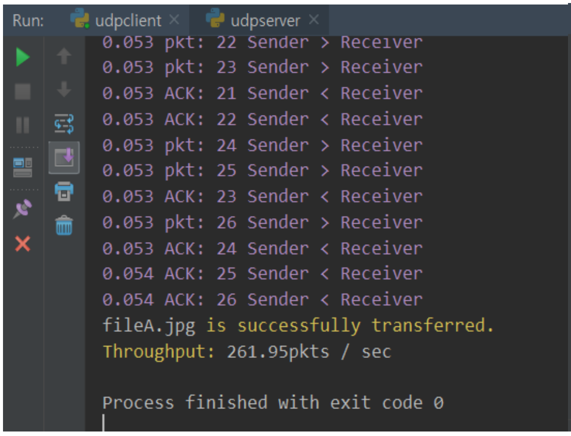
```
C:\Users\lion7\PycharmProjects\network_hw1\venv
packet loss probability: 0
socket recv buffer size: 65536
socket recv buffer size updated: 10000000
```

3. Next, run "udpserver.py" in directory 'PA3'
4. Type input for "udpserver.py"

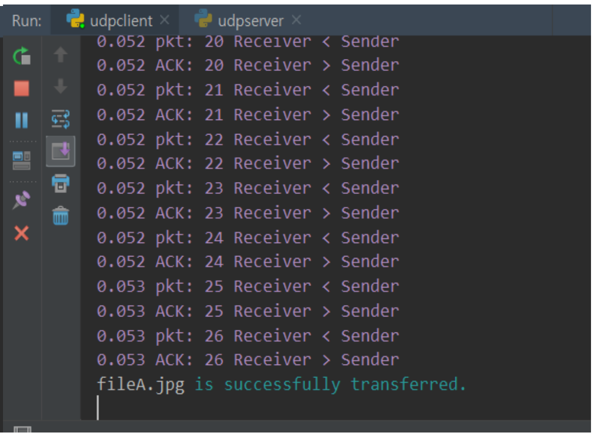


```
C:\Users\lion7\PycharmProjects\network_
Receiver Ip address: 127.0.0.1
window size: 5
timeout (sec): 100
file name: fileA.jpg
```

5. See file transmission



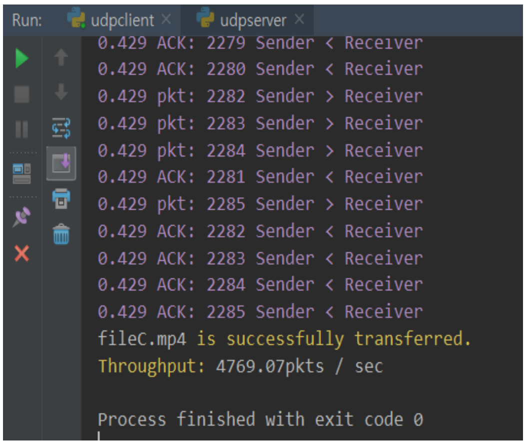
```
0.053 pkt: 22 Sender > Receiver
0.053 pkt: 23 Sender > Receiver
0.053 ACK: 21 Sender < Receiver
0.053 ACK: 22 Sender < Receiver
0.053 pkt: 24 Sender > Receiver
0.053 pkt: 25 Sender > Receiver
0.053 ACK: 23 Sender < Receiver
0.053 pkt: 26 Sender > Receiver
0.053 ACK: 24 Sender < Receiver
0.054 ACK: 25 Sender < Receiver
0.054 ACK: 26 Sender < Receiver
fileA.jpg is successfully transferred.
Throughput: 261.95pkts / sec

Process finished with exit code 0
```



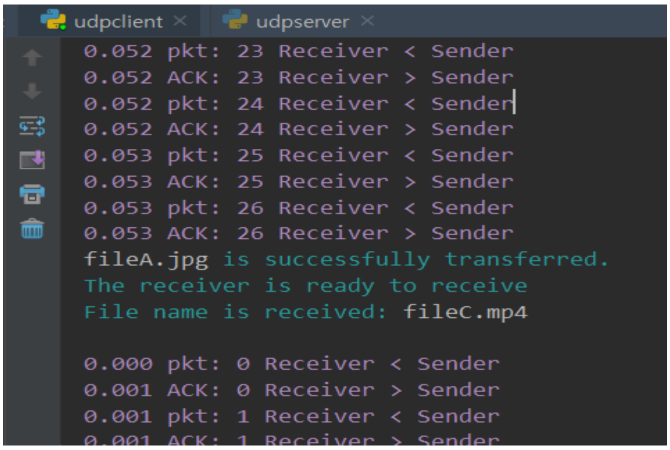
```
0.052 pkt: 20 Receiver < Sender
0.052 ACK: 20 Receiver > Sender
0.052 pkt: 21 Receiver < Sender
0.052 ACK: 21 Receiver > Sender
0.052 pkt: 22 Receiver < Sender
0.052 ACK: 22 Receiver > Sender
0.052 pkt: 23 Receiver < Sender
0.052 ACK: 23 Receiver > Sender
0.052 pkt: 24 Receiver < Sender
0.052 ACK: 24 Receiver > Sender
0.053 pkt: 25 Receiver < Sender
0.053 ACK: 25 Receiver > Sender
0.053 pkt: 26 Receiver < Sender
0.053 ACK: 26 Receiver > Sender
fileA.jpg is successfully transferred.
```

[left: sender side, right: receiver side]

6. If you want to send another file after the file transmission, run “udpserver.py” again.



```
0.429 ACK: 2279 Sender < Receiver
0.429 ACK: 2280 Sender < Receiver
0.429 pkt: 2282 Sender > Receiver
0.429 pkt: 2283 Sender > Receiver
0.429 pkt: 2284 Sender > Receiver
0.429 ACK: 2281 Sender < Receiver
0.429 pkt: 2285 Sender > Receiver
0.429 ACK: 2282 Sender < Receiver
0.429 ACK: 2283 Sender < Receiver
0.429 ACK: 2284 Sender < Receiver
0.429 ACK: 2285 Sender < Receiver
fileC.mp4 is successfully transferred.
Throughput: 4769.07pkts / sec

Process finished with exit code 0
```

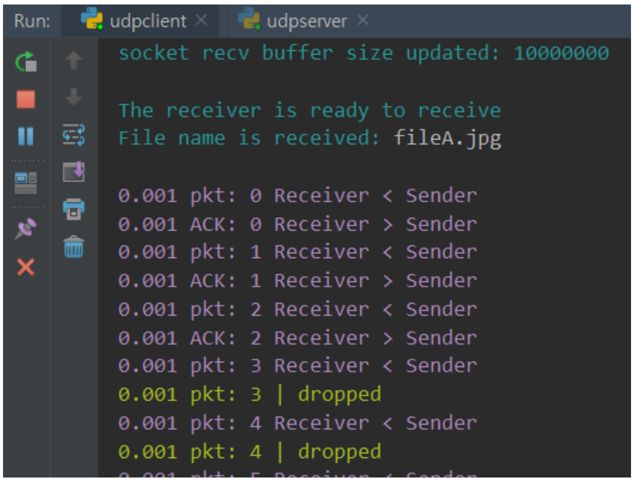


```
0.052 pkt: 23 Receiver < Sender
0.052 ACK: 23 Receiver > Sender
0.052 pkt: 24 Receiver < Sender
0.052 ACK: 24 Receiver > Sender
0.053 pkt: 25 Receiver < Sender
0.053 ACK: 25 Receiver > Sender
0.053 pkt: 26 Receiver < Sender
0.053 ACK: 26 Receiver > Sender
fileA.jpg is successfully transferred.
The receiver is ready to receive
File name is received: fileC.mp4

0.000 pkt: 0 Receiver < Sender
0.001 ACK: 0 Receiver > Sender
0.001 pkt: 1 Receiver < Sender
0.001 ACK: 1 Receiver > Sender
```

[left: receiver side(continue), right: sender side(new)]

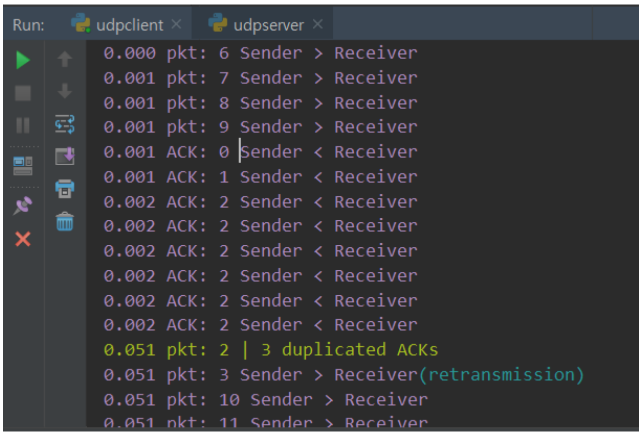
3. Screen shot for several scenarios

1. receiver: dropped



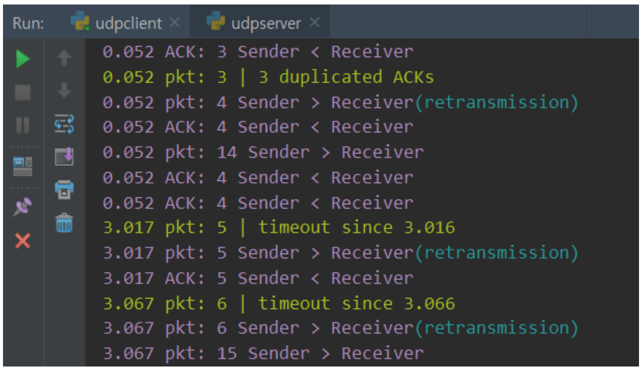
```
socket recv buffer size updated: 10000000

The receiver is ready to receive
File name is received: fileA.jpg

0.001 pkt: 0 Receiver < Sender
0.001 ACK: 0 Receiver > Sender
0.001 pkt: 1 Receiver < Sender
0.001 ACK: 1 Receiver > Sender
0.001 pkt: 2 Receiver < Sender
0.001 ACK: 2 Receiver > Sender
0.001 pkt: 3 Receiver < Sender
0.001 pkt: 3 | dropped
0.001 pkt: 4 Receiver < Sender
0.001 pkt: 4 | dropped
0.001 pkt: 5 Receiver < Sender
```

2. sender: 3 duplicated acks



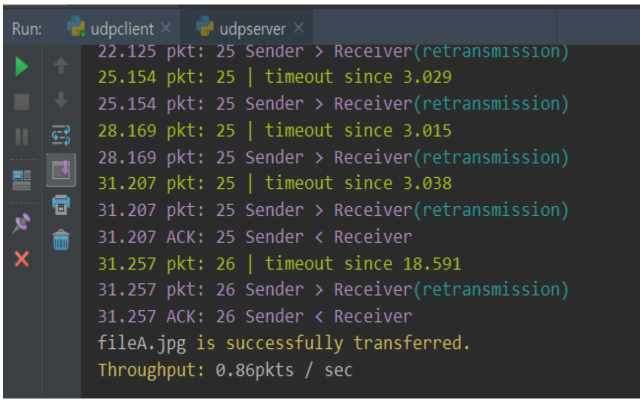
```
0.000 pkt: 6 Sender > Receiver
0.001 pkt: 7 Sender > Receiver
0.001 pkt: 8 Sender > Receiver
0.001 pkt: 9 Sender > Receiver
0.001 ACK: 0 Sender < Receiver
0.001 ACK: 1 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.002 ACK: 2 Sender < Receiver
0.051 pkt: 2 | 3 duplicated ACKs
0.051 pkt: 3 Sender > Receiver(retransmission)
0.051 pkt: 10 Sender > Receiver
0.051 pkt: 11 Sender > Receiver
```

3. sender: time out



```
0.052 ACK: 3 Sender < Receiver
0.052 pkt: 3 | 3 duplicated ACKs
0.052 pkt: 4 Sender > Receiver(retransmission)
0.052 ACK: 4 Sender < Receiver
0.052 pkt: 14 Sender > Receiver
0.052 ACK: 4 Sender < Receiver
0.052 ACK: 4 Sender < Receiver
3.017 pkt: 5 | timeout since 3.016
3.017 pkt: 5 Sender > Receiver(retransmission)
3.017 ACK: 5 Sender < Receiver
3.067 pkt: 6 | timeout since 3.066
3.067 pkt: 6 Sender > Receiver(retransmission)
3.067 pkt: 15 Sender > Receiver
```

4. sender: throughput



```
22.125 pkt: 25 Sender > Receiver(retransmission)
25.154 pkt: 25 | timeout since 3.029
25.154 pkt: 25 Sender > Receiver(retransmission)
28.169 pkt: 25 | timeout since 3.015
28.169 pkt: 25 Sender > Receiver(retransmission)
31.207 pkt: 25 | timeout since 3.038
31.207 pkt: 25 Sender > Receiver(retransmission)
31.207 ACK: 25 Sender < Receiver
31.257 pkt: 26 | timeout since 18.591
31.257 pkt: 26 Sender > Receiver(retransmission)
31.257 ACK: 26 Sender < Receiver
fileA.jpg is successfully transferred.
Throughput: 0.86pkts / sec
```

4. How to design program

1. udpserver.py – sender

- get input from user
- divide object file and store as packet in *packets* (list)
- send packet until receive all acks
- when sending packet, mark some information
  (e.g. whether the packet was sent, when the packet was sent)

- Following the information, catch 3 duplicated acks or timeout event
- If you catch one of those above events, retransmission occurs

2. udpclient.py- receive

   ack number는 마지막으로 제대로 받은 packet number로 구성했습니다

   아무것도 제대로 받지 못했을 때 ack를 보낸다면, -1을 보내는 것으로 구성했습니다.

   - get input from user
   - update receive buffer size, if needed
   - receive file information
     (e.g. file name, number of total packets)
   - receive file as packet
   - if the received packet is in order, send received packet number
   - if not, send last received packet number

3. color.py
   - to color prompt line

4. packet.py
   - data structure
   - It contains sequence number, data, flag for whether it was sent, time stamp to save when it was sent

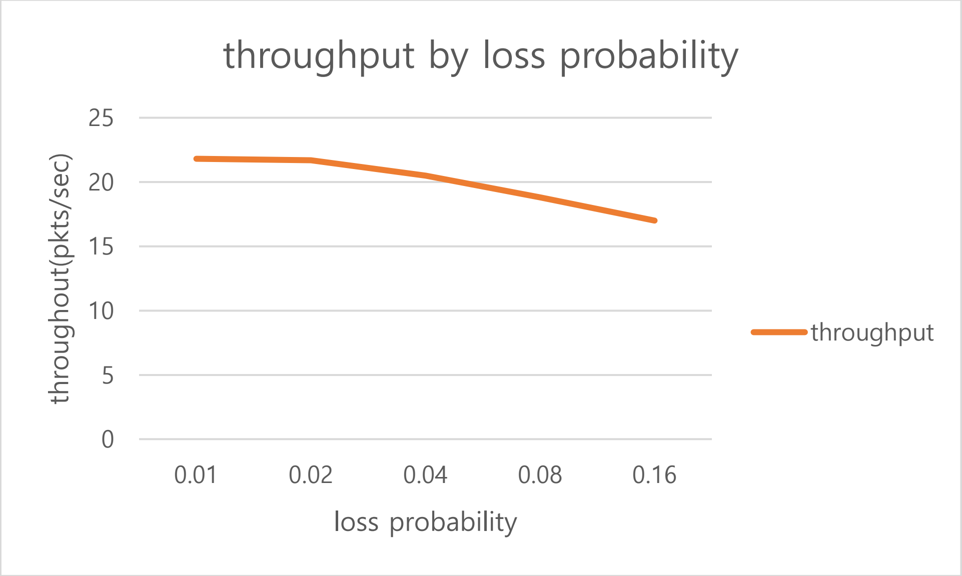
5. timer.py
   - unused

5. Show two graph for the experimentation results

experimented with fileC.mp4(2.23MB)

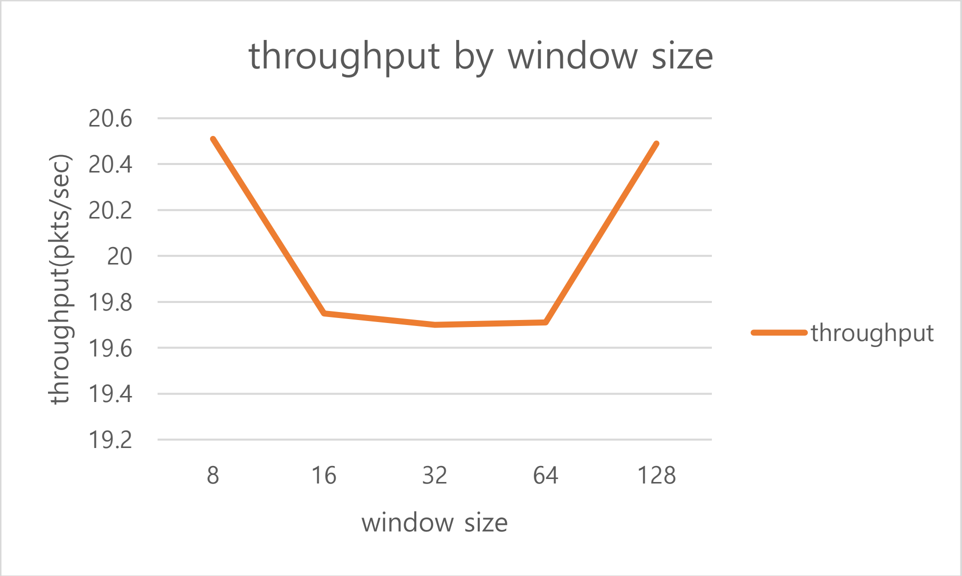
1. a goodput graph with different probabilities of packet loss

2016311739 김연재



Throughput is decreasing as loss probability grows

2. a goodput graph with different window sizes



The best window size would be 32 in above conditions.