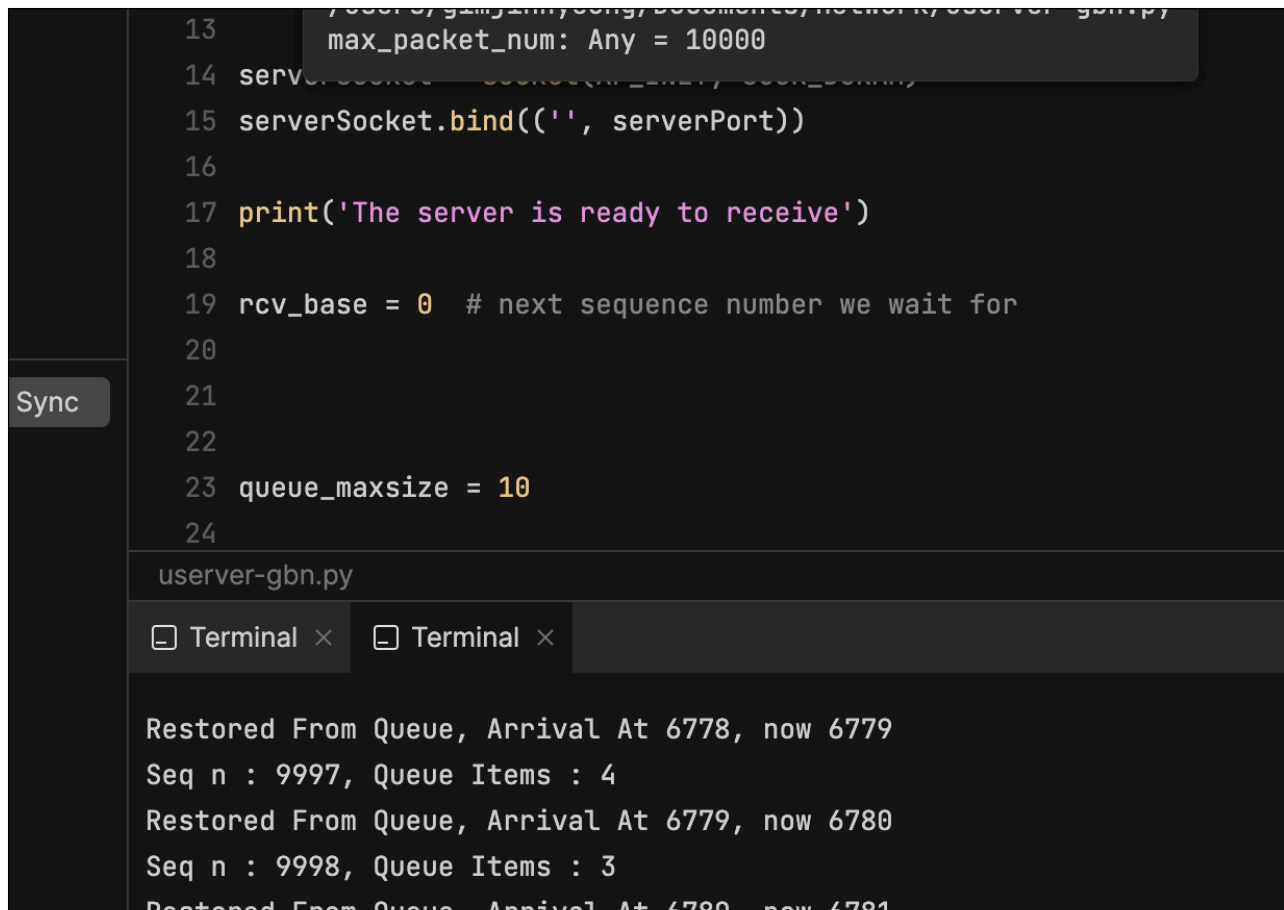


# Computer Network PA1

## UDP ACKING, QUEUEING SIMULATOR

2018310522 김진녕



```
13 serverSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP, socket.MSG_WAITFORDATA)
14 serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
15 serverSocket.bind(('', serverPort))
16
17 print('The server is ready to receive')
18
19 rcv_base = 0 # next sequence number we wait for
20
21
22
23 queue_maxsize = 10
24
```

userver-gbn.py

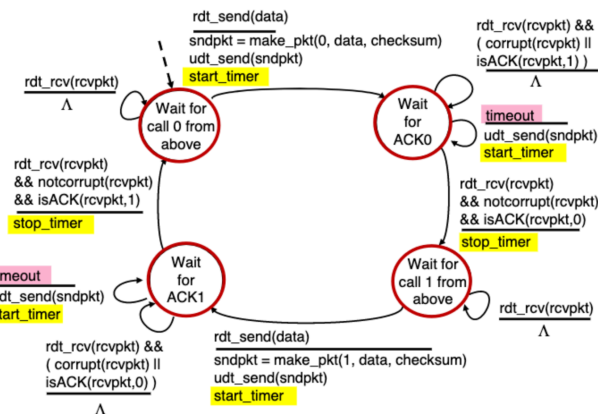
Terminal × Terminal ×

```
Restored From Queue, Arrival At 6778, now 6779
Seq n : 9997, Queue Items : 4
Restored From Queue, Arrival At 6779, now 6780
Seq n : 9998, Queue Items : 3
Restored From Queue, Arrival At 6780, now 6781
```

## 소개

이번 프로젝트는, UDP 기반의 통신에서 TCP의 ACK 프로세스를 모방하는 시뮬레이션을 구축하는 것입니다. 프로젝트는 두 개의 응용프로그램으로 이루어집니다. 클라이언트, 서버 응용프로그램이 소켓 통신을 통해 서로 메시지를 주고 받으면서 위 과정을 수행합니다.

클라이언트는 모든 ack를 전달받을때 까지 서버에 패킷을 전송합니다. 서버는 수신받은 패킷에 대해 ack를 반환합니다. 이를 바탕으로 client는 패킷 전송에 대해 congestion을 control합니다.



이미지 - RDT 3.0의 동작 방식

## 부가 기능

위 프로젝트는 단일 컴퓨터의 동일한 가상 환경에서 서버와 클라이언트를 동시에 실행하기 때문에, 실제로 timeout, packet drop 등의 통신 오류가 발생할 염려가 없습니다. 이상적인 시뮬레이션 구현을 위해, timeout, packet loss, queueing 등의 동작을 시뮬레이션에 가상으로 구현하도록 합니다.

### Packet Loss(Bit Error) 구현

실제 통신에서는 낮은 확률로 패킷이 링크를 타고 다음 라우터로 전달이 안되거나, 전달은 됐는데 데이터가 변조되어 읽을 수 없는 경우가 발생합니다. 이를 구현하기 위해, 랜덤한 낮은 확률로 클라이언트가 패킷을 보내지 않도록 합니다.

### Timeout 구현

클라이언트가 전송한 패킷이 일정 시간이 지나도 ack를 받지 못하면 timeout을 일으킵니다. Timeout interval 값을 통해, 돌아온 패킷이 interval을 넘길 경우 timeout을 raise할 수 있습니다.

## Queuing Delay 구현

서버에서 실제 큐 처럼 동작할 수 있도록 패킷 큐를 만듭니다. 서버가 수신한 패킷은 우선 사이즈가 제한된 큐에 삽입됩니다. 서버는 큐를 읽는 thread를 만듭니다. 해당 thread는 임의로 지정한 queue delay만큼 sleep 했다가, 큐의 첫번째 데이터를 읽어 패킷을 처리합니다.

Queuing delay의 존재로 패킷이 timeout loss를 야기할 수 있으며, queue의 크기가 제한되어있기 때문에, packet drop이 구현될 수 있습니다.

## Congestion Controll 구현

Client는 ack 수신을 바탕으로 timeout interval, window size를 지정할 수 있습니다.

window는 윈도우 1회에 대해 정상적으로 수신할 경우 window size를 1 증가시킵니다. 중간에 loss가 발생하면 window size를 1/2로 줄입니다.

Client는 평균 rtt를 구하여, 이를 기반으로 timeout interval을 지정하여, timeout까지 기다리는 시간을 능동적으로 조절할 수 있습니다.

## 로그

```
SEQ 9790 | WIN 39 | TO ITV 0.022300
timeout detected: 9798
timeout interval: 0.01747317777341483
timeout ratio : 0.011486
retransmission: 9800
SEQ 9800 | WIN 21 | TO ITV 0.019758
SEQ 9800 | WIN 27 | TO ITV 0.020547
SEQ 9860 | WIN 30 | TO ITV 0.039230
```

클라이언트는 매 ack 수신마다, 수신한 ack의 seq #를 출력하며, 이를 바탕으로 현재 클라이언트의 congestion controll 상황을 출력합니다. 이는 window size, timeout interval을 포함합니다.

timeout이 발생하면, 발생한 패킷의 seq와, timeout 발생 빈도를 출력합니다.

```
SEQ 9990 | WIN 38 | TO ITV 0.028071
done
Mean RTT : 0.010021
MAX WIN SIZE : 39
AVG WIN SIZE : 23.723791
```

최종 클라이언트 보고서입니다. 평균 RTT, 최대 window 크기, 평균 window 크기를 출력합니다.