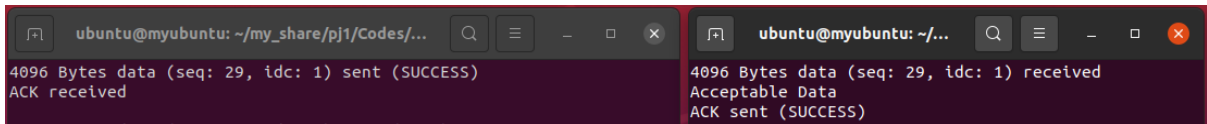


통신네트워크 Project 1

2018142059 김서영

- Data packet이 제대로 전송되고 ACK도 제대로 전송된 경우

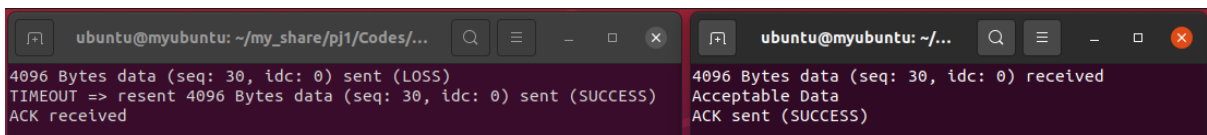


```
ubuntu@myubuntu: ~/my_share/pj1/Codes/...
4096 Bytes data (seq: 29, idc: 1) sent (SUCCESS)
ACK received

ubuntu@myubuntu: ~/...
4096 Bytes data (seq: 29, idc: 1) received
Acceptable Data
ACK sent (SUCCESS)
```

클라이언트가 sequence number가 29, indicator가 1인 데이터 패킷을 서버에 성공적으로 전송했다. 해당 data packet의 indicator값 1이 서버가 기다리던 값과 일치하기 때문에 서버가 data packet(seq#=29)를 accept하고, 이를 알리기 위해 ACK를 클라이언트에 다시 전송한다. 클라이언트는 이 ACK를 성공적으로 받아 이 data packet에 대한 전송이 마무리된다.

- Data packet이 Loss 되어 Timeout 이후 재전송된 경우

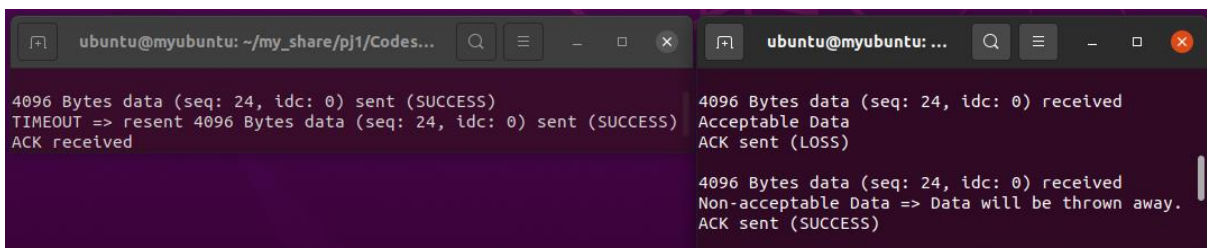


```
ubuntu@myubuntu: ~/my_share/pj1/Codes/...
4096 Bytes data (seq: 30, idc: 0) sent (LOSS)
TIMEOUT => resent 4096 Bytes data (seq: 30, idc: 0) sent (SUCCESS)
ACK received

ubuntu@myubuntu: ~/...
4096 Bytes data (seq: 30, idc: 0) received
Acceptable Data
ACK sent (SUCCESS)
```

클라이언트가 sequence number가 30, indicator가 0인 data packet을 서버에 전송하지만 loss된다. 따라서 서버는 받은 packet이 없으니 아무 일도 하지 않고, 클라이언트는 패킷 전송 시 시작한 timer가 timeout되어 해당 패킷을 재전송한다. 서버는 이를 받아 accept하고, 이에 대한 ACK를 다시 클라이언트에 다시 보내고 성공적으로 전송되어 sequence number가 30인 패킷에 대한 전송이 마무리된다.

- Data packet이 제대로 전송되었지만 ACK이 Loss되어 Data packet이 Timeout 이후 재전송되고 재전송 역시 Loss없이 제대로 전송되어 수신단에서 Data packet이 Duplicate 된 경우



```
ubuntu@myubuntu: ~/my_share/pj1/Codes/...
4096 Bytes data (seq: 24, idc: 0) sent (SUCCESS)
TIMEOUT => resent 4096 Bytes data (seq: 24, idc: 0) sent (SUCCESS)
ACK received

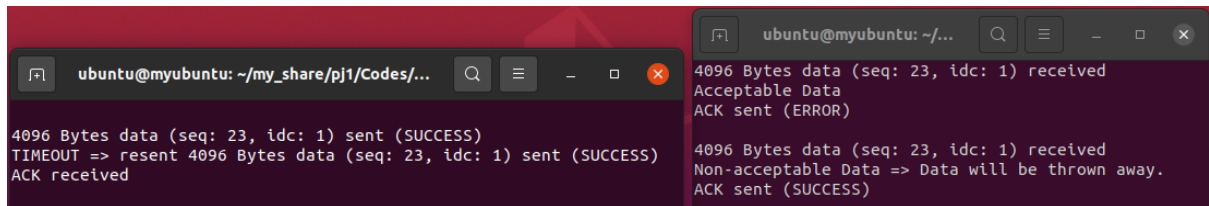
ubuntu@myubuntu: ...
4096 Bytes data (seq: 24, idc: 0) received
Acceptable Data
ACK sent (LOSS)

4096 Bytes data (seq: 24, idc: 0) received
Non-acceptable Data => Data will be thrown away.
ACK sent (SUCCESS)
```

클라이언트가 sequence number가 24인 패킷을 성공적으로 서버에 전송하고, 서버가 이를 accept 하였지만 그에 대한 표시로 클라이언트에 보낸 ACK가 loss되었다. 따라서 클라이언트에서는 패킷

이 전송된 것을 알지 못하고 패킷의 timer가 timeout되어 해당 패킷을 재전송하였다. 재전송된 패킷을 서버가 받았지만 이전과 같은 duplicate된 패킷이므로 서버가 이를 버린다.

- Data packet이 제대로 전송되었지만 ACK error가 발생하여 잘못된 indicator를 가진 ACK이 전송되어 잘못된 ACK으로 판단하여 Data packet에 대한 ACK을 기다리다 Data packet이 Timeout 이후 재전송되고 재전송 역시 Loss 없이 제대로 전송되어 수신단에서 Data packet이 Duplicate된 경우



```
ubuntu@myubuntu: ~/my_share/pj1/Codes/...
4096 Bytes data (seq: 23, idc: 1) sent (SUCCESS)
TIMEOUT => resent 4096 Bytes data (seq: 23, idc: 1) sent (SUCCESS)
ACK received

ubuntu@myubuntu: ~/...
4096 Bytes data (seq: 23, idc: 1) received
Acceptable Data
ACK sent (ERROR)

4096 Bytes data (seq: 23, idc: 1) received
Non-acceptable Data => Data will be thrown away.
ACK sent (SUCCESS)
```

클라이언트가 sequence number가 23인 데이터 패킷을 정상적으로 전송하여 서버가 이를 받았지만 ACK 전송 중 ERROR가 일어나 정상적으로 전송되었다는 것을 클라이언트가 알 수 없었고, ACK를 기다리다 timeout되어 패킷을 다시 보내고 정상적으로 전송되어 서버가 받았지만, 서버에게는 이전에 받은 패킷과 같은 duplicate 된 패킷이기 때문에 이를 버린다.

-stop and wait 방식의 특징

이 프로토콜에서는 하나의 패킷을 전송한 후 이에 대한 ACK를 받아야 다음 패킷의 전송을 시작하는 stop-and wait 방식이 이용되었다. 이 방법은 패킷 각각을 하나하나 전송하기 때문에 하나의 패킷 전송에 패킷이 가고, ACK가 오는 최소 2번의 RTT가 소요되어 시간이 많이 소요된다. 이 과정에서 패킷이 loss되거나 ACK loss/error가 발생하면 이에 대해 timeout될때까지의 시간과 재전송과정에 소요되는 시간이 추가적으로 소요되어 훨씬 많은 시간이 걸리게 된다.

-rdt 1.0과의 비교

rdt 1.0은 packet loss, ACK loss 및 error가 존재하지 않고 모든 패킷이 순서대로 전송되는 것을 가정한 rdt 방식이다. 이 프로젝트의 프로토콜은 rdt 1.0에 대해 sequence number를 추가하여 패킷의 순서를 명확히 하였고 (그러나 stop and wait 방식이기 때문에 전송 과정에서는 항상 순서대로 전송되어 sequence number는 이용되지 않는다), packet loss에 대응하기 위해 각 패킷마다 timer를 설정하도록 추가하여 loss 시 재전송 하도록 하였다. 또한, ACK의 loss나 error로 인해 이미 전송된 패킷이 재전송되어 duplicate 되는 것에 대응하기 위해 indicator 값을 패킷마다 추가하여 구분할 수 있도록 하였다.