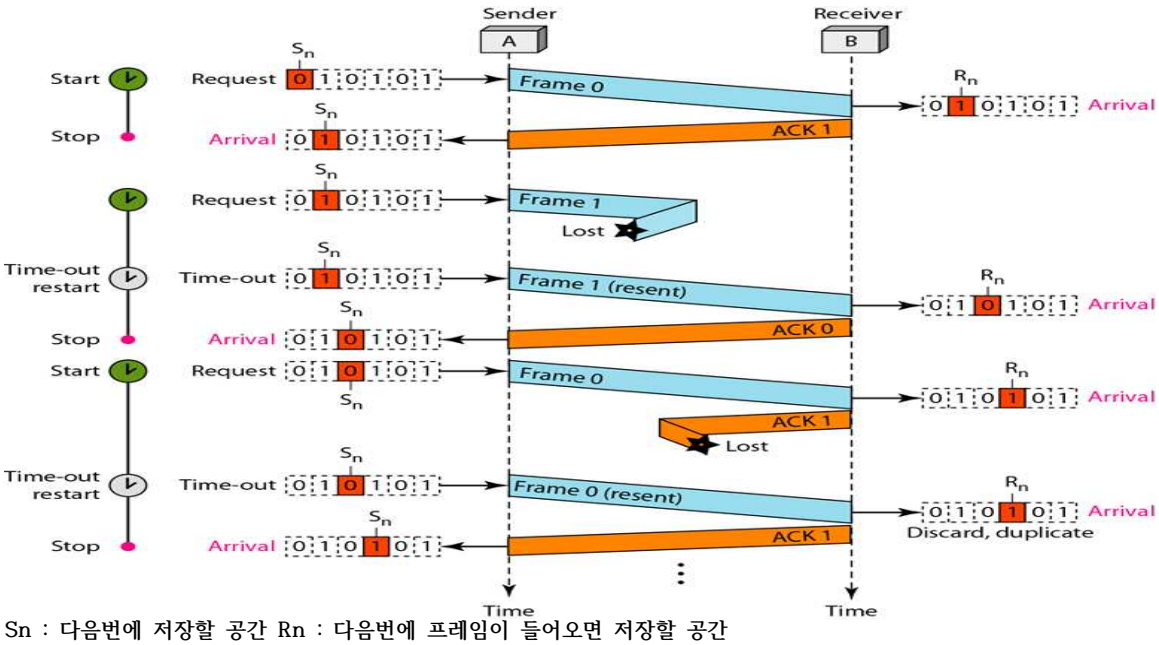
ARQ(Automatic ReQuest(자동 재전송 요구)) : TCP와 관계없이 독립적으로 발전한 기술.
노드와 노드 간의 일대일 전송.

Stop-and-wait ARQ



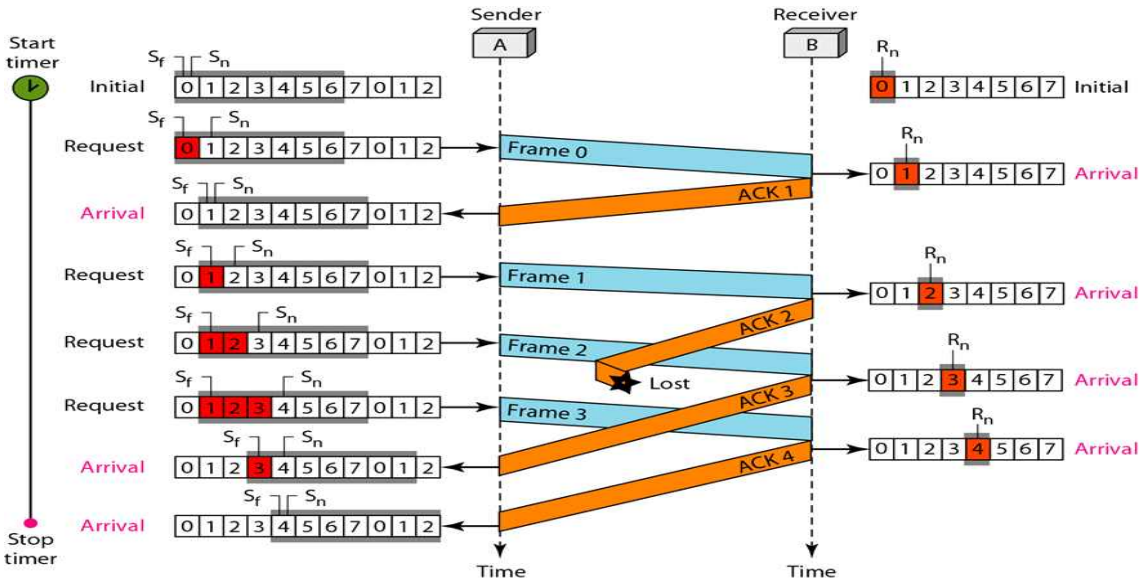
Sn : 다음번에 저장할 공간 Rn : 다음번에 프레임이 들어오면 저장할 공간
transmission delay : frame(segment)전송을 시작된 시점과 전송을 완료한 시간
propagation delay : sender에서 receiver까지 전송되는 시간
Frame 전송에 에러가 발생하는 경우 : Receiver가 ACK 를 보내지 않는다.
Sender은 적당한 시간을 기다렸다가 ACK가 와야 할 시간이 되었는데 안 오면 오류가 발생한 것으로 상정하고 frame을 재전송한다.(timer 이용)
Frame말고 ACK 전송에 에러가 발생하는 경우 ; Frame을 정상적으로 보냇다고 쳐도 ACK가 정상적으로 오지 않았기에 Frame 재전송.
근데 여기서 문제점은 Receiver은 재전송된 패킷과 아까 패킷이 서로 다른 패킷이라고 생각한다. 그래서 application message가 똑같은 것이 두 번 간다. 이런 경우 오류 발생

->**해결방법** : frame에 일련번호(ID)를 붙인다. 이걸로 통해서 재전송임을(같은 패킷임을)알 수 있다.
->ID는 몇 개를 사용해야 할까? - 2개면 된다.(위에서 ACK 전송에 오류가 난 경우에서 0번이 두 번 오면 ACK가 깨졌다는 사실을 인지할 수 있다. 두 번째 패킷은 자연스럽게 폐기되었다.)
<위의 그림> Sn:0 Rn:0 -> Frame0전송, Rn=0벡터에 Frame저장후 Rn증가(Sn:0 Rn:1) -> ACK1전송, ACK가 도착하지 않았으므로 Frame0 재전송(Sn:0 Rn:1) -> Rn=1이므로 Frame0이 와도 폐기, ACK1전송(Sn:0 Rn:1) -> ACK1이 도착했으므로 Sn증가(Sn:1 Rn:1)

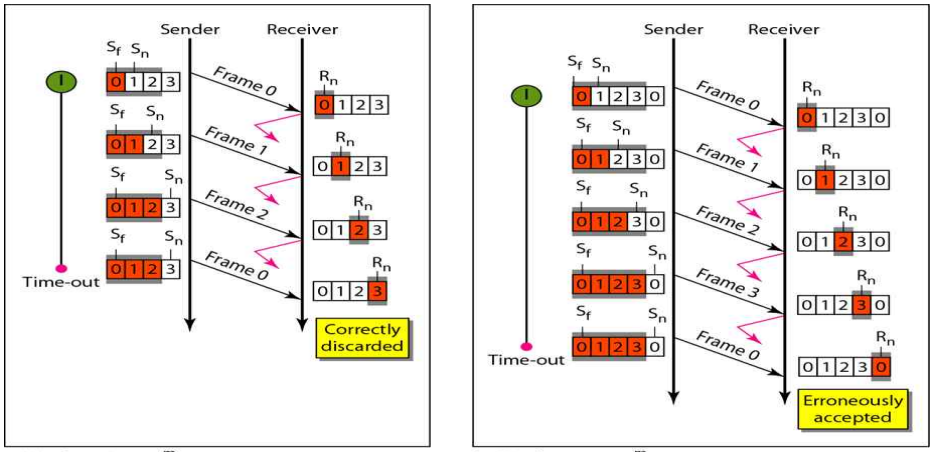
메모리 공간은 버퍼로 확보. 포인터로 버퍼를 관리. 포인터를 이용해 데이터를 쓰거나 읽음.
포인터(Sn : 다음번에 저장할 공간 Rn : 다음번에 프레임이 들어오면 저장할 공간)
Stop-and-Wait ARQ : 프레임을 하나씩 보내 ACK를 받는 것은 비효율적이다. propagation

delay가 크다면 한세월이 걸린다.
->해결방법 : 여러 개 프레임(4개정도)를 한꺼번에 보내고 ACK를 한번에 받음 : Pipelining
Frame을 구분하는 일련번호(ID)는 frame 2개를 연속으로 보낸다고 치면 최소 3개 필요

Go-Back-N ARQ



(Sn : 다음번에 저장할 공간 Rn : 다음번에 프레임이 들어오면 저장할 공간,
Sf : ACK를 기다리고 있는 frame들 중 첫 번째)
다음 시스템은 7개까지 연속으로 보내는 시스템 여러 개의 Frame을 동시에 보내고 싶어라.
Frame 1,2,3을 거의 동시에(굉장히 빠른 시간 안에) ACK를 받지 않고 보내는 상황
Frame1의 ACK가 깨지고 Frame2, Frame3 ACK 잘 받음.
Frame1을 재전송을 해야 하나? -> 안 해도 된다. ACK3이 왔으니까 앞에 꺼 잘 받았다는 의미이다. 설사 ACK2가 깨졌어도 커버 가능
Sn(next point to write) Application이 쓰고(write) 싶으면 여기다 써라. 다음 번에 저장할 공간. Stop-and-wait와 다른 점은 ACK를 받지 않았어도 증가시킨다.
윈도우 상황을 보면 Sender의 상황파악 가능.



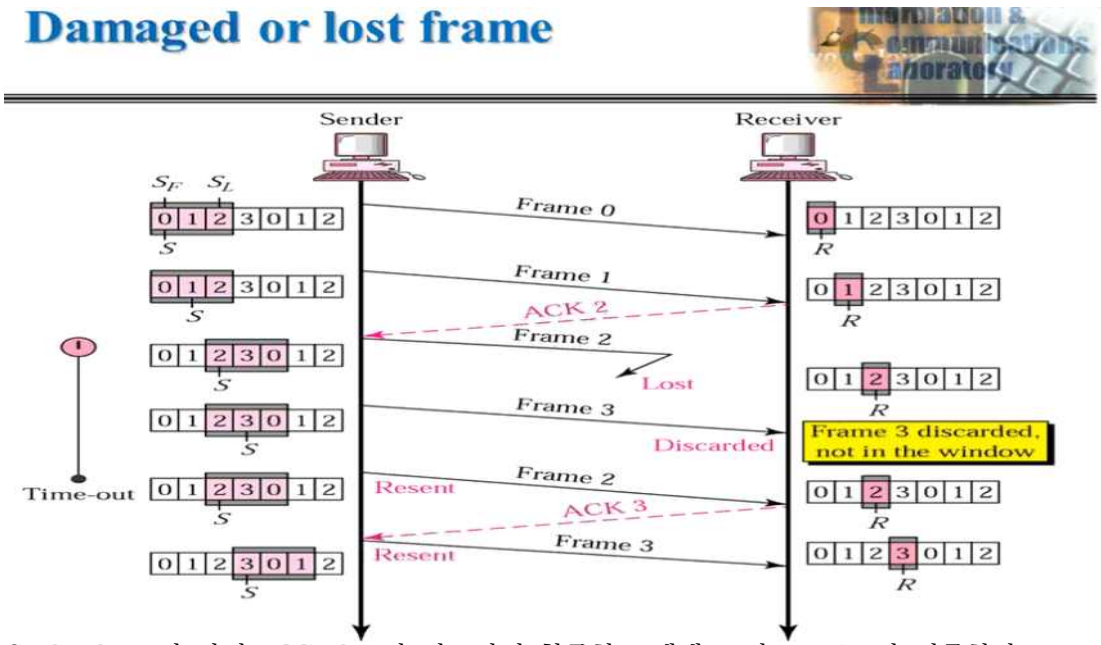
a. Window size < 2^m b. Window size = 2^m

Window size는 몇으로 해야 하는가?(까만 부분)

2^m －1 size(m : sequence number의 bit)

2^m인 경우 오른쪽 경우처럼 reciever가 에러 난 건지도 모름

(참고로 n개를 연속해서 보낼 때의 ID는 n+1개가 필요하다.)



Go back n 의 단점 : Window가 바보같이 활동함 : 제대로 된 receive만 기록한다.

Frame2가 오류가 발생 했다고 가정하자.

Frame3은 일단 보낸다. 근데 Frame 3을 보내도 receiver는 ACK를 보내지 않는다.

(Rn=2일 때 Frame2가 도착하지 않았으므로 Rn2를 증가시키지 않고 ACK를 보내지 않는다.

Rn=2인데 Frame3을 보내면 ACK를 보내지 않음.)

그러면 2번부터 다시 재전송하고 3번을 재전송한다. 이래야 receive pointer가 제대로 작동한다. 그래서 Go back n은 잘 사용하지 않는다.(2번으로 go back한다는 의미)

이 경우는 Go-back-2