<인터넷 프로토콜>

작성자_2018320161_송대선

작성일_05_21

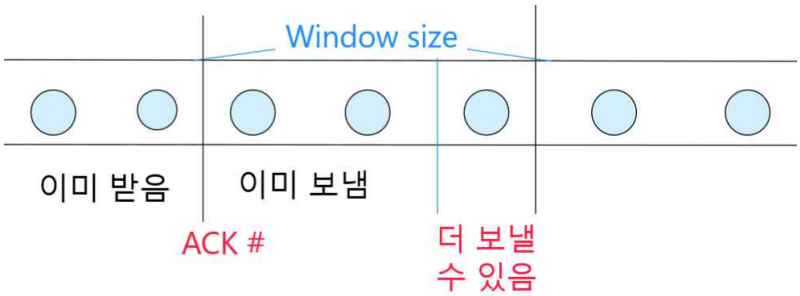
TCP Segment에는
1. ACK number
2. Window size
두 개가 같이 온다.

A->B로 가는 connection은
1. A의 ISN,
2. B의 Window size,
3. 공통의 MSS가 필요하다.

B->A로 가는 connection은
1. B의 ISN,
2. A의 Window size,
3. 공통의 MSS가 필요하다.

두 Connection은 정말 별도의 것이다.

window size만큼 무조건 다 보낼 수 있는 것이 아니다.



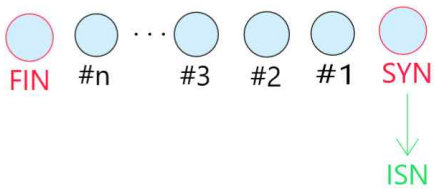
-window size-
요즘은 window size를 늘리기도 하고, 줄이기도 한다.
1. 만일 socket buffer가 계속 가득 차 있으면? -> Socket buffer와 window size를 늘린다.
2. 만일 socket buffer가 계속 거의 비어 있으면? -> Socket buffer와 window size를 줄인다.

-TCP Pseudo header-
IP와 data의 관계가 일치하는 지의 여부를 본다.

-urgent ponter-
urgent ponter는 "내 뒤에 급한 것이 와요"라는 의미이다.
-> "급한 것"의 의미는 응용이 이해할 수 있다. (TCP는 모른다.)
ex) 다운로드를 받다가 창을 지워버리면 data flush를 해야 하는데, 그런 것들이다.
Flag urgent ponter=0, urgent ponter값은 무의미(쓰레기 값)
Flag urgent ponter=1, urgent ponter값은 유의미

-TCP 3-way handshake-
1. SYN(->) : Connection을 맺자!
2. ACK(<-) + SYN(<-) : 1.의 SYN에 대한 ACK, 그리고 상대도 Connection을 맺기를 원함
3. ACK(->) : 알겠다.

SYN Segment -> 3개의 값을 전달한다.
ISN는 SYN Segment의 Sequence Number이다.



ISN은 첫 번째 data byte의 Sequence number이다. -> False
"0"= ISN, "x"=ISN+x으로 처리한다.

TCP는 Data는 재전송하지만, ACK를 재전송하지 않는다.

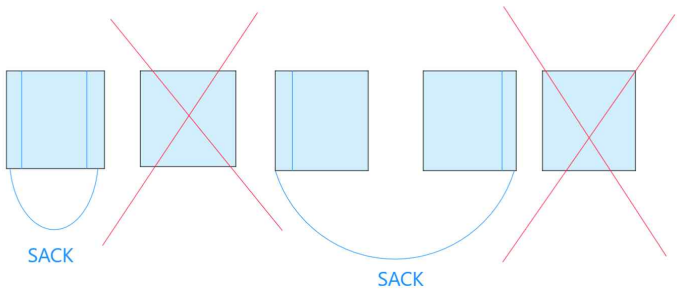
-FIN-
FIN=1인 4개의 Segment가 필요하다.
1. FIN(->) : A->B Connection을 끊겠다.
2. ACK(<-) : 1. FIN에 대한 B의 응답
---
3. FIN(<-) : B->A Connection을 끊겠다.
4. ACK(->) : 3. FIN에 대한 A의 응답

TCP는 한쪽만 끊을 수도 있다. 그러나 그렇게 굳이 할 이유는 없다.
half-close: TCP 연결에서 A->B 또는 B->A 둘 중 하나가 끊어진 경우
half-open: 두 TCP 중 하나가 그냥 집에 가버림. -> 이미 하나가 죽었는데 그걸 모른다.

-Timeout of connection establishment-
SYN을 보냈는데, 상대가 없으면? -> SYN에 대한 ACK가 오지 않는다.
아니면 SYN이 오다가 죽거나, ACK가 오다가 죽으면?
-> 연결될 때까지 SYN을 보내본다.
사실 SYN은 처음 보내보는 Segment라서 얼마의 시간을 소요해서 ACK가 올지를 모름
-> 3초 기다려 본다.
3초 후 안 오면 다시 보내 본다.
-> 6초 기다려 본다.
안 오면 -> 12초
안 오면 -> 24초

사용자 입장에서는 3초, 9초, 21초, 45초, ...이렇게 기다린다.
-> exponential backoff의 이유?
-> Network이 너무 복잡해서 오다가 죽는 것일 수도 있으니 그냥 배려하는 차원에서 늦게 보내는 것이다.

-Selective ACK-
tcp.options==sack



sequence Number는 4Byte이다.
SACK에는 두 sequence Number에 대한 정보->8Byte
그리고 이 option이 SACK라는 정보
이 option의 길이에 대한 정보(Parsing을 위함)
가 필요하기 때문에 SACK의 길이는 8B를 초과한다.

TCP Header의 길이는 고정 20B, 가변 40B이기 때문에
8*4=32
최대 4개까지 가능하다.

SACK Permitted: SYN Segment에서 “04 02”하고 SACK를 쓰겠다는 말을 해줘야 된다.
Sender가 써도 되냐고 물어보고, Receiver가 써도 된다고 대답을 해야 사용가능하다.

-Window Scale Option-
원래 TCP에는 window size에 대한 부분이 16bit만큼 있어서 최대 65535 Bytes만큼 쓸 수 있다. -> 그 이상의 window size를 쓰기 위함이다.

Window Scale Option에 쓰인 수를 제곱해서 window size에 곱해준다.
ex) 10이라고 쓰면, 2^{10}을 window size에 곱해서 생각해라
window scale option=0으로 답변이 오면, '난 그걸 이해하는데 사용하지는 않을꺼야'라는 의미이다.
window scale option에 대한 언급조차 없으면 '난 그거 이해 못 하니까 쓰지마'라는 의미이다.

-Timestamp Option-
data 재전송을 위한 시간을 정해야하는데 그걸 정하려면 왕복시간을 측정해야한다.
완복 시간을 측정하는데 타이머를 사용한다.
패킷이 떠날 때 타이머가 돌기 시작하고, ACK로 패킷이 전송되었다는 것을 알면 타이머를 엄춘다.
-> 원래 타이머는 이미 하나가 돌고 있으면, 다른 패킷에 대해서는 시간을 측정하지 않는다.
왜?
-> 시간 계산을 하려면 패킷 하나당 interrupt가 최소 두 번은 일어난다.
-> 이게 옛날에는 너무 힘든 일이었다. 그러나, 요즘은 아니다.

Timestamp Option에 걸린 시간을 찍어서 보낸다.

-Path MTU Discovery-
ICMP가 그 MSS가 fragment될 뻔 했다고 이야기 하면서 더 작은 MSS를 알려준다.