<TCP (흐름제어/혼잡제어)>

- TCP 통신이란?

네트워크 통신에서 신뢰적인 연결 방식이다. TCP는 기본적으로 unreliable network에서, reliable network를 보장할 수 있도록 하는 프로토콜이다. network congestion avoidance algorithm을 사용한다.

- reliable network를 보장한다는 것은 4가지 문제점

1) 손실 : packet이 손실될 수 있는 문제
2) 순서바뀜 : packet의 순서가 바뀌는 문제
3) Congestion : 네트워크가 혼잡한 문제
4) Overload : receiver가 overload 되는 문제

- 흐름제어/혼잡제어란?

흐름제어는 (endsystem 대 endsystem) 즉 송신 측과 수신 측의 데이터 처리 속도 차이를 해결하기 위한 기법이다. Flow Control은 receiver가 packet을 지나치게 많이 받지 않도록 조절하는 것이다. 기본 개념은 receiver가 sender에게 현재 자신의 상태를 feedback 한다는 점이다.

- 전송의 전체 과정

1) Application layer : sender application layer가 socket에 data를 쓴다
2) Transport layer : data를 segment에 감싸고 network layer에 넘김. segment 가 receiving node로 전송이 되는데 sender는 send buffer에 data를 저장하고, receiver는 receive buffer에 data를 저장
3) application에서 준비가 되면 이 buffer에 있는 것을 읽기 시작합니다. 흐름제어(flow control)의 핵심은 이 receiver buffer가 넘치지 않게 하는 것
4) receiver는 RWND(Receive Window) : receive buffer의 남은 공간을 확보

1. 흐름제어 (Flow Control)

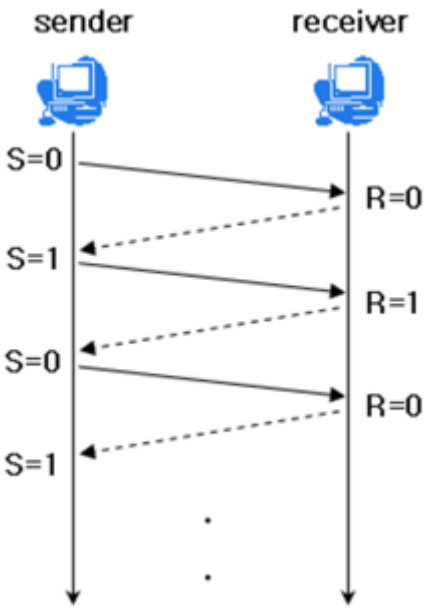
수신 측이 송신 측보다 데이터 처리 속도가 빠르면 문제없지만, 송신 측의 속도가 빠를 경우 문제가 생김

수신 측에서 제한된 저장 용량을 초과한 이후에 도착하는 데이터는 손실될 수 있으며, 만약 손실된다면 불필요하게 응답과 데이터 전송이 송/수신 측 간에 빈번히 발생

이러한 위험을 줄이기 위해 송신 측의 데이터 전송량을 수신 측에 따라 조절

해결방법

1) **Stop and Wait :** 매번 전송한 패킷에 대해 확인 응답을 받아야만 그 다음 패킷을 전송하는 방법

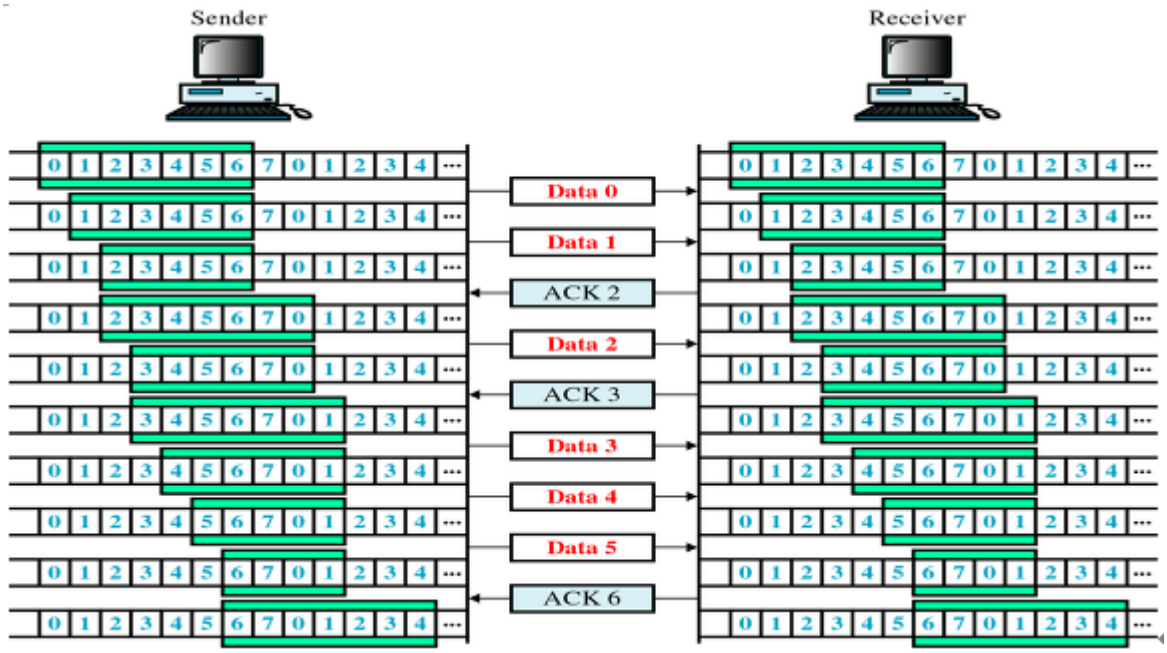


2) **Sliding Window (Go Back N ARQ)**

수신 측에서 설정한 윈도우 크기만큼 송신 측에서 확인응답없이 세그먼트를 전송할 수 있게 하여 데이터 흐름을 동적으로 조절하는 제어기법

목적: 전송은 되었지만, ACK를 받지 못한 byte의 숫자를 파악하기 위해 사용하는 protocol

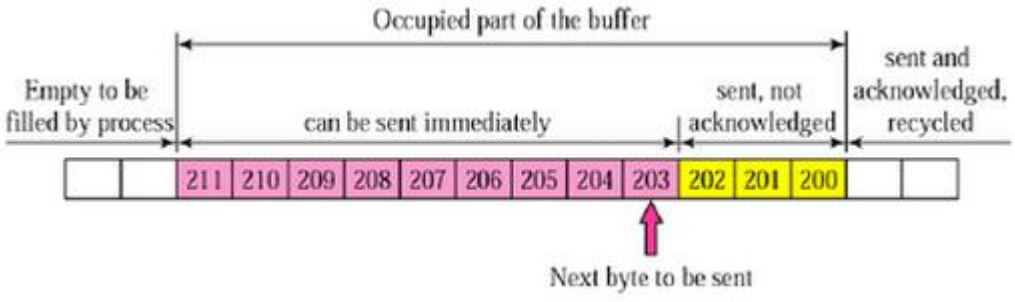
동작방식: 먼저 윈도우에 포함되는 모든 패킷을 전송하고, 그 패킷들의 전달이 확인 되는대로 이 윈도우를 옆으로 옮김으로써 그 다음 패킷들을 전송



Window: TCP/IP를 사용하는 모든 호스트들은 송신하기 위한 것과 수신하기 위한 2개의 Window를 가지고 있다. 호스트들은 실제 데이터를 보내기 전에 '3 way handshaking'을 통해 수신 호스트의 receive window size에 자신의 send window size를 맞추게 된다.

3) 세부구조

송신 버퍼

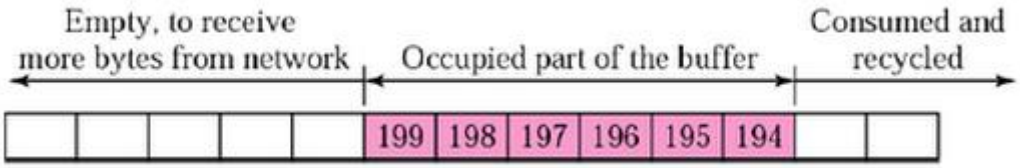


200 이전의 바이트는 이미 전송되었고, 확인응답을 받은 상태

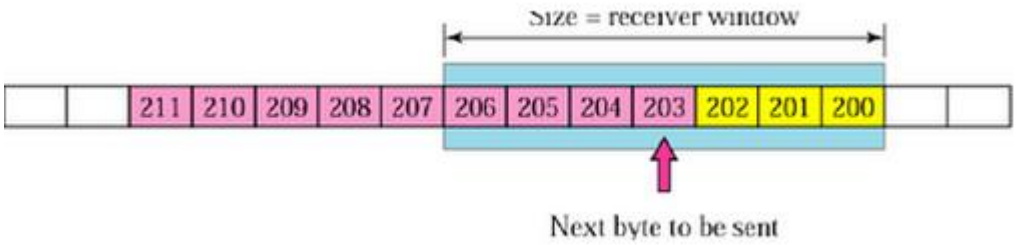
200 ~ 202 바이트는 전송되었으나 확인응답을 받지 못한 상태

203 ~ 211 바이트는 아직 전송이 되지 않은 상태

수신 윈도우

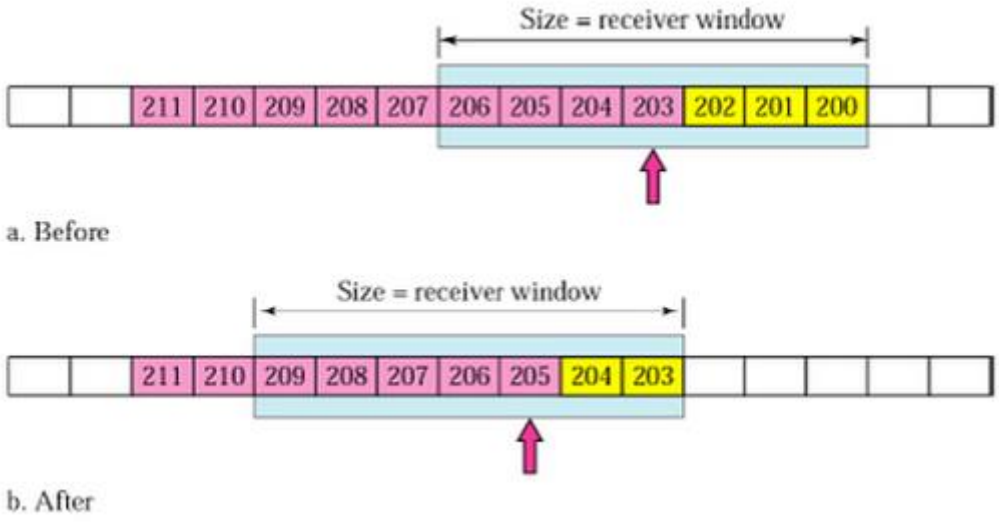


송신 윈도우



수신 윈도우보다 작거나 같은 크기로 송신 윈도우를 지정하게 되면 흐름제어가 가능하다.

송신 윈도우 이동



Before : 203 ~ 204를 전송하면 수신측 에서는 확인 응답 203을 보내고, 송신측은 이를 받아 after 상태와 같이 수신 윈도우를 203 ~ 209 범위로 이동

after : 205 ~ 209가 전송 가능한 상태

Selected Repeat

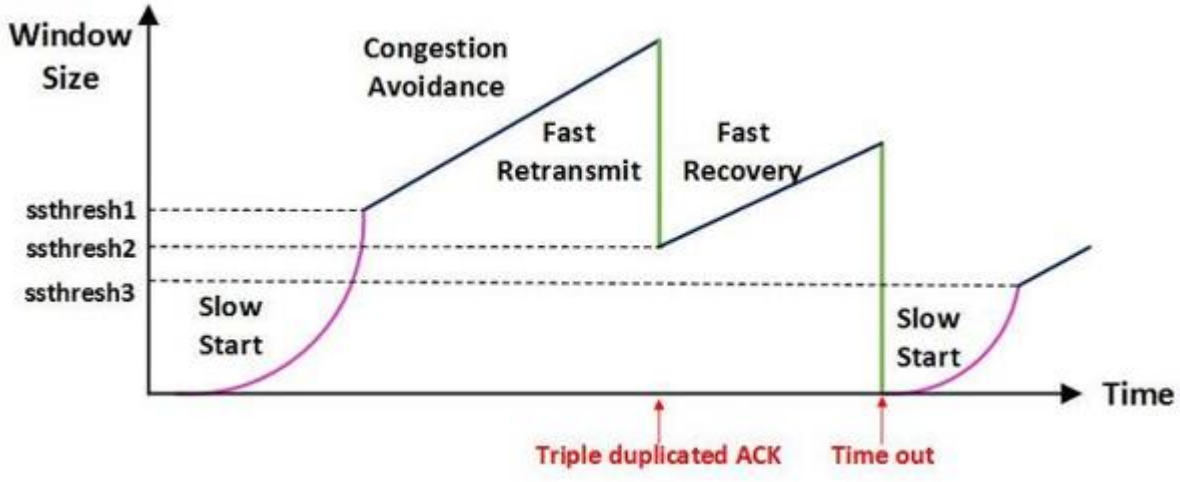
2. 혼잡제어 (Congestion Control)

송신 측의 데이터는 지역망이나 인터넷으로 연결된 대형 네트워크를 통해 전달된다. 만약 한 라우터에 데이터가 몰릴 경우, 자신에게 온 데이터를 모두 처리할 수 없게 된다. 이런 경우 호스트들은 또 다시 재전송을 하게 되고 결국 혼잡만 가중시켜 오버플로우나 데이터 손실을 발생시키게 된다. 따라서 이러한 네트워크의 혼잡을 피하기 위해 송신 측에서 보내는 데이터의 전송속도를 강제로 줄이게 되는데, 이러한 작업을 **혼잡제어**라고 한다.

또한 네트워크 내에 패킷의 수가 과도하게 증가하는 현상을 혼잡이라 하며, 혼잡 현상을 방지하거나 제거하는 기능을 혼잡제어라고 한다.

흐름제어가 송신측과 수신측 사이의 전송속도를 다루는데 반해, 혼잡제어는 호스트와 라우터를 포함한 보다 넓은 관점에서 전송 문제를 다루게 된다.

해결 방법



1) AIMD(Additive Increase / Multiplicative Decrease)

처음에 패킷을 하나씩 보내고 이것이 문제없이 도착하면 window 크기(단위 시간 내에 보내는 패킷의 수)를 1씩 증가시켜가며 전송하는 방법

패킷 전송에 실패하거나 일정 시간을 넘으면 패킷의 보내는 속도를 절반으로 줄인다.

공평한 방식으로, 여러 호스트가 한 네트워크를 공유하고 있으면 나중에 진입하는 쪽이 처음에는 불리하지만, 시간이 흐르면 평형상태로 수렴하게 되는 특징이 있다.

문제점은 초기에 네트워크의 높은 대역폭을 사용하지 못하여 오랜 시간이 걸리게 되고, 네트워크가 혼잡해지는 상황을 미리 감지하지 못한다. 즉, 네트워크가 혼잡해지고 나서야 대역폭을 줄이는 방식이다.

2) Slow Start (느린 시작)

AIMD 방식이 네트워크의 수용량 주변에서는 효율적으로 작동하지만, 처음에 전송 속도를 올리는 데 시간이 오래 걸리는 단점이 존재했다.

Slow Start 방식은 AIMD와 마찬가지로 패킷을 하나씩 보내면서 시작하고, 패킷이 문제없이 도착하면 각각의 ACK 패킷마다 window size를 1씩 늘려준다. 즉, 한 주기가 지나면 window size가 2배로 된다.

전송속도는 AIMD에 반해 지수 함수 꼴로 증가한다. 대신에 혼잡 현상이 발생하면 window size를 1로 떨어뜨리게 된다.

처음에는 네트워크의 수용량을 예상할 수 있는 정보가 없지만, 한번 혼잡 현상이 발생하고 나면 네트워크의 수용량을 어느 정도 예상할 수 있다.

그러므로 혼잡 현상이 발생하였던 window size의 절반까지는 이전처럼 지수 함수 꼴로 창 크기를 증가시키고 그 이후부터는 완만하게 1씩 증가시킨다.

3) Fast Retransmit (빠른 재전송)

빠른 재전송은 TCP의 혼잡 조절에 추가된 정책이다.

패킷을 받는 쪽에서 먼저 도착해야할 패킷이 도착하지 않고 다음 패킷이 도착한 경우에도 ACK 패킷을 보내게 된다.

단, 순서대로 잘 도착한 마지막 패킷의 다음 패킷의 순번을 ACK 패킷에 실어서 보내게 되므로, 중간에 하나가 손실되게 되면 송신 측에서는 순번이 중복된 ACK 패킷을 받게 된다. 이것을 감지하는 순간 문제가 되는 순번의 패킷을 재전송 해줄 수 있다.

중복된 순번의 패킷을 3개 받으면 재전송을 하게 된다. 약간 혼잡한 상황이 일어난 것이므로 혼잡을 감지하고 window size를 줄이게 된다.

4) Fast Recovery (빠른 회복)

혼잡한 상태가 되면 window size를 1로 줄이지 않고 반으로 줄이고 선형증가시키는 방법이다. 이 정책까지 적용하면 혼잡 상황을 한번 겪고 나서부터는 순수한 AIMD 방식으로 동작하게 된다.

Reference

- https://www.brianstorti.com/tcp-flow-control/
- https://www.brianstorti.com/tcp-flow-control/