TCP/IP 흐름제어와 혼잡제어

흐름 제어

데이터링크 레이어의 디자인 이슈이다. 송신측에서 수신측으로의 데이터 흐름을 적절하게 제어하는 것을 처리하는 것을 말한다.

- 송신자가 데이터를 빠르게 보내면 수신자도 그것을 받아 처리할 수 있도록 하는 것이다.
- 서로 다른 속도로 동작하는 주체들이 커뮤니케이션 할 수 있게 한다.
- 수신자에 로드가 많이 걸리거나 송신에 비해 프로세싱 파워가 모자라면 일어난다.

리시버에게 확인 응답을 받기 전에 보낼 수 있는 데이터 양을 조정하는 방식으로 이루어진다. 즉, 송신자에게 보낼 수 있는 데이터 양을 알려주는 것이다.

수신자가 센더에게 피드백을 보내는 방식(feedback based flow control), 프로토콜에 빌트인 매커니즘으로 전체 전송률을 제한하는 방식(rate based flow control) 등으로 분류 가능하다.

Stop-and-Wait Flow Control

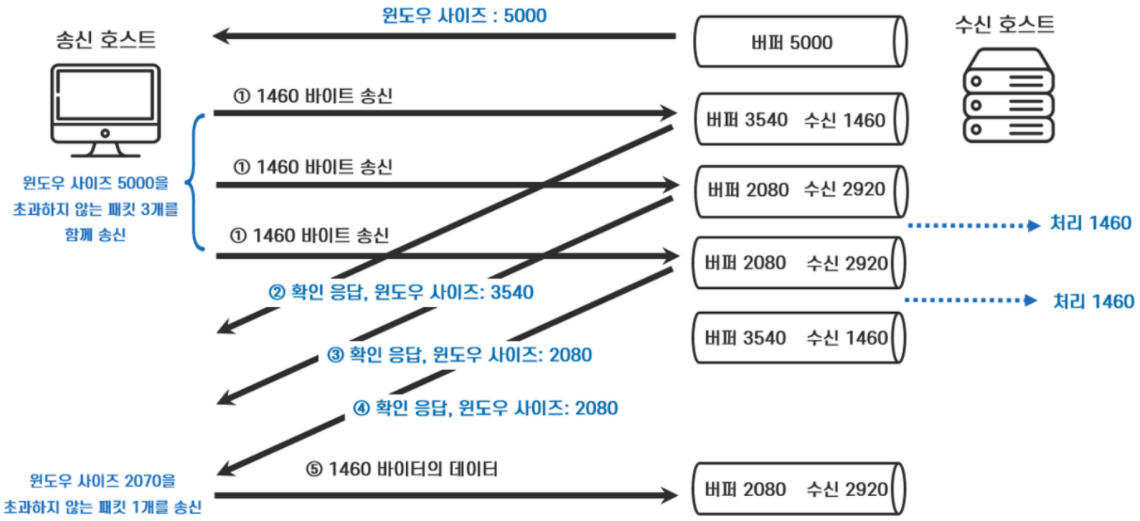
수신자는 다음 데이터 프레임을 수신할 준비가 되어있음을 송신자에게 알려준다. 승인을 받으면 발신자가 데이터프레임을 전송한다. 한 번에 한 프레임만 전송할 수 있어서 비효율적이다. 응답이 돌아오는 동안 아무 일도 하지 않으므로 속도가 느리다.

- 장점: 간단하고 정확함
- 단점: 느리다, 한번에 한 패킷만 보낼 수. 있다., 비효율적이고 전송프로세스를 느리게 만들수있음.

Sliding Window Flow Control

매번 응답을 기다리는 대신, 윈도우에 포함되는 패킷을 연속해서 전송하고 패킷들의 전달이 확인되는 대로 윈도우를 옆으로 옮겨서 다음 패킷을 전송하는 방법

윈도우 크기는 핸드셰이킹 과정에서 수신측의 TCP헤더에 담아서 보낸다. 이후 수신자의 버퍼 크기에 따라 변한다.



동작방식: 프레임을 전송하면 양쪽의 윈도우 크기가 줄어들고, ACK 프레임이 되돌아오면 양쪽 윈도우의 크기가 늘어난다. 각각 왼쪽경계, 오른쪽 경계가 오른쪽으로 이동하는 방식이다.

혼잡제어 (Congestion Control)

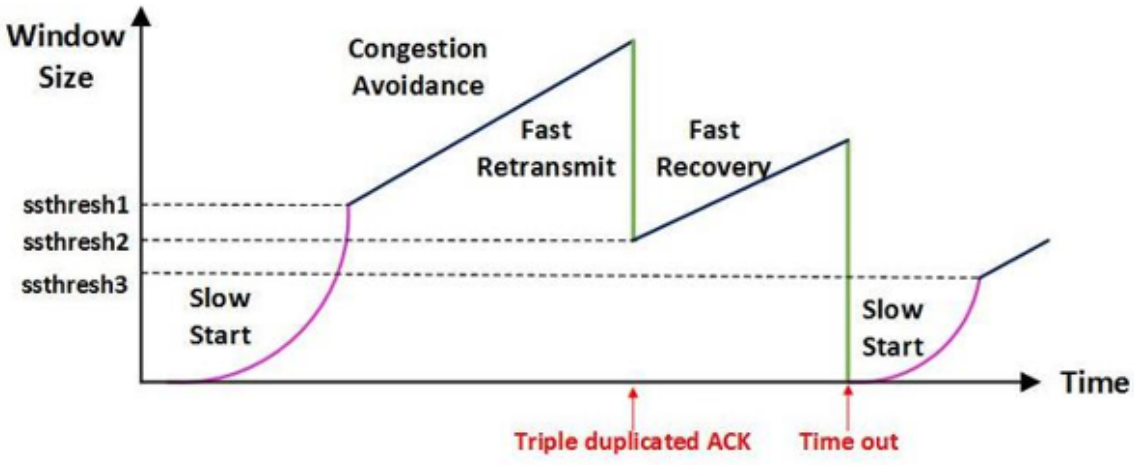
송신측의 데이터가 수신측으로 이동하는 과정에서, 대형 네트워크를 거치게 된다. 그 중 한 라우터에 데이터가 몰리게 될 경우, 데이터는 모든 데이터를 처리할 수 없게 된다. 따라서 이런 경우 호스트들은 재전송을 하게 되고 결국 혼잡만 가중되게 된다. 이것은 오버풀로우나 데이터 손실을 야기한다.

따라서 송신측에서는 데이터 전송속도를 강제로 줄여야 하는데 이러한 작업을 혼잡제어라고 한다

또한, 네트워크 내의 패킷 수가 과도하게 증가하는 현상을 혼잡이라고 하고 이를 방지 또는 제거하는 기능을 혼잡 제어라고 한다.

흐름제어는 송수신측의 전송속도를 다루지만, 혼잡제어는 호스트와 라우터를 포함한 넓은 범위의 전송 문제를 다루게 된다.

- **#해결 방법**



  - **#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.

- **#Slow Start (느린 시작)**
  - AIMD 방식이 네트워크의 수용량 주변에서는 효율적으로 작동하지만, 처음에 전송 속도를 올리는데 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.
- **#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.
- **#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.