네트워크 개론 Part 20 : TCP Transmission Control Protocol의 이해 8

조인준
KBS 미디어기술연구소
차장

지난 편에서는 안정적 전송을 보장하는 TCP의 특징인 흐름제어(Flow Control), 오류제어(Error Control), 혼잡제어(Congestion Control) 중 그 마지막인 혼잡제어에 관해 다루었습니다. 혼잡제어는 아래 세 단계로 나누어 볼 수 있으며 현재까지 설명된 내용은 ① Slow Start와 ② Congestion Avoidance입니다. 이번 편에서는 ③ Congestion Detection 즉, 혼잡 검출에 관한 이야기와 함께 혼잡제어 관련 내용을 종합하도록 하겠습니다.

① Slow Start　② Congestion Avoidance (혼잡 회피)　③ Congestion Detection (혼잡 검출)

Congestion Detection 단계에서는 송신 디바이스가 전송된 데이터의 유실 및 네트워크의 혼잡 여부를 판단하고 이에 따라 혼잡 윈도우 사이즈를 조절하는 혼잡제어에 관한 조치를 하게 됩니다. 송신 디바이스가 전송된 데이터의 유실을 유추하는 방법은 아래 두 가지입니다.

① Time Out (제한시간 경과)
- Time Out은 송신 데이터에 대한 수신 디바이스의 ACK 메시지가 일정 시간 안에 송신 디바이스에 도착하지 않으면 송신 디바이스가 네트워크 혼잡으로 인해 데이터가 유실된 것으로 판단
- 네트워크에 혼잡이 발생했을 가능성이 높음

② 3 Duplicate Acknowledgements (3회 중복 승인)
- 수신 디바이스가 수신되지 않은 데이터에 대한 ACK 메시지(사실상 요청 메시지)를 중복하여 보내고 송신 디바이스가 같은 데이터에 대한 ACK 메시지를 3회 이상 수신했을 경우 네트워크 혼잡으로 데이터가 유실된 것으로 판단

3 Duplicate Acknowledgements는 네트워크 혼잡으로 인해 손실된 데이터의 빠른 복구를 위한 Fast Retransmit(빠른 재전송) 알고리즘에서 사용하는 혼잡 검출 방식입니다. [그림 1]을 통해 3 Duplicate Acknowledgements에 대해 자세히 알아보겠습니다. [그림 1]과 같이 송신 디바이스가 보내는 세그먼트(Segment : TCP, UDP 등 전송계층에 해당하는 헤더와 데이터 모두를 포함하는 용어)에 대해 수신 디바이스는 ACK 메시지를 보내게 됩니다. ACK 메시지의 헤더 안에는 어떤 세그먼트에 대한 ACK 메시지인지 식별할 수 있는 데이터가 있습니다. [그림 1]에서 송신 디바이스는 세그먼트 1, 2, 3, 4, 5, 6을 순차적으로 발송하였지만, 세그먼트 2는 네트워크 혼잡으로 인해 유실되어 전송되지 못합니다. 세그먼트 1을 받은 수신 디바이스는 이에 대한 ACK 1을 발송합니다. ACK 1은 세그먼트 1을 잘 받았다는 회신인 동시에 세그먼트 1을 잘 받았으니 세그먼트 2를 보내라는 요청으로도 여길 수 있습니다. 하지만 네트워크 혼잡으로 세그먼트 2가 유실되어서 수신하지 못했으므로 세그먼트 3, 4, 5, 6 등에 대해 세그먼트 2를 요청하는 ACK인 ACK 1 을 [그림 1]과 같이 중복해서 발송합니다.

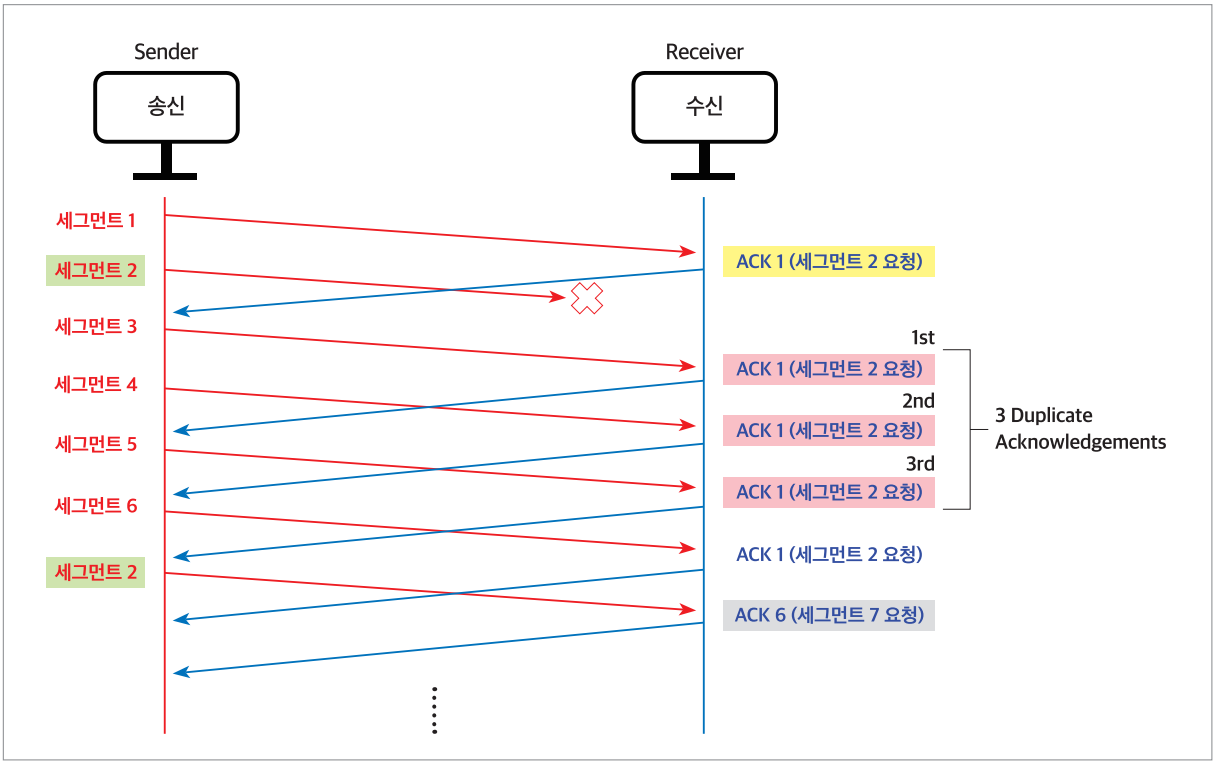

그림 1. 3 Duplicate Acknowledgements

[그림 1]에서 노란색 바탕으로 표시된 최초의 ACK 1에 이어 분홍색 바탕으로 표시된 중복된 세 개의 ACK 1을 받게 되면 송신 디바이스는 세그먼트 2를 재송신합니다. 세그먼트 2가 이상 없이 수신 디바이스에 도착하면 수신 디바이스는 세그먼트 6까지 현재 잘 받았으므로 세그먼트 6을 잘 받았고 세그먼트 7을 전송하라는 의미의 ACK 6을 보냅니다. 이상이 3 Duplicate Acknowledgements를 이용한 Fast Retransmit이 동작하는 방식입니다. 인터넷에서 자료를 검색하다 보니 3 Duplicate Acknowledgements를 세는 방법에 있어서 [그림 1] 노란색 바탕의 ACK 1 부터 세 개의 ACK 1을 세는 방법과 ACK 1 이후에 최초로 중복되어 보내진 ACK 1부터 세 개의 ACK 1을 세는 방법이 혼재하여 이 중 무엇이 맞는지 조사하다가 해외 유명 대학에서 올린 자료들을 참고해서 최초의 ACK 1 이후 중복되어 보내진 세 개의 ACK 1을 3 Duplicate Acknowledgements로 판단하는 것이 맞는 것 같아 [그림 1]과 같이 표시하였습니다. 그렇다면 이쯤에서 독자 여러분에게 하나의 질문이 떠오를 것 같습니다.

왜 하나도 둘도 아닌 세 개의 중복 ACK 메시지를 사용하지?

"왜 3일까?"라는 강한 궁금증에 C군은 또다시 열심히 인터넷 검색을 했고 아래와 같은 대답을 찾을 수 있었습니다. Fast Retransmit에 대해 정의하고 있는 표준문서인 RFC 2001에 아래와 같은 이야기가 있다고 합니다.

> Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost.

표 1. RFC 2001 중 3 Duplicate Acknowledgements 관련 내용

설명하면 TCP 방식에서 ACK 메시지를 받은 송신 디바이스는 중복된 ACK 메시지가 세그먼트의 유실로 인한 것인지 아니면 전송 과정에서의 세그먼트 도달 순서가 바뀌어 뒤의 세그먼트들이 먼저 도착했기 때문인지 알 수가 없으므로 중복된 ACK 메시지를 받았다고 해서 바로 재전송하는 것보다 한두 개 더 기다리다 보면 새로운 ACK 메시지가 도착하는 경우가 있으니 일단 바로 반응하지는 않는 것이 좋다고 본 것 같습니다. 하지만 세 개 이상의 중복된 ACK 메시지가 도착한다면 이것은 네트워크 혼잡으로 인해 세그먼트가 유실되었을 가능성이 농후한 상황이므로 세 개의 중복 ACK 메시지 이후 혼잡을 판단하는 것이 바람직하다는 이야기 같습니다. 왜 굳이 세 개까지 기다릴까에 대한 궁금증이 해소되었나요?

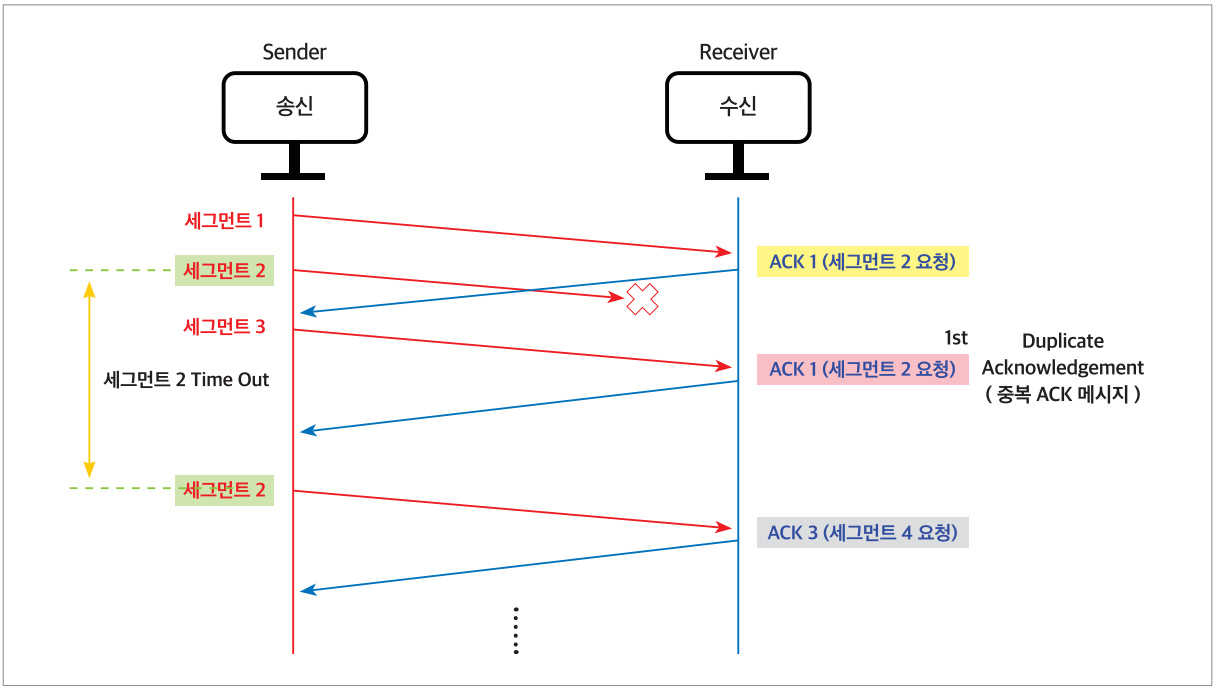

그림 2. 윈도우 사이즈가 작은 경우 Time Out의 필요성

지금까지 보면 3 Duplicate Acknowledgements 방식만 사용해도 충분할 것 같은데 왜 Time Out 같은 답답한 방식을 혼잡 검출에 같이 사용하는지 의아한 생각도 들 것 같습니다. 왜 Time Out 방식도 같이 사용할 수밖에 없는지 [그림 2]와 같은 아주 극단적인 예를 통해 그 이유를 알려드리겠습니다. [그림 2]는 작은 윈도우 사이즈를 사용하는 경우입니다. 윈도우 사이즈가 작으면 세 개 이상의 중복 ACK 메시지를 발생시킬 정도로 많은 세그먼트를 연이어 전송하지 않을 수 있습니다. 이 때문에 [그림 2]에서 ACK 1 메시지의 중복 ACK 메시지는 하나만 발생한 채로 유실된 세그먼트 2의 전송이 지체되는 순간이 발생할 수 있습니다. 이때는 Time Out으로 세그먼트 2를 재전송하는 것이 유실된 세그먼트를 더 빨리 재전송하는 방법이 됩니다. 이와 같은 이유로 Time Out과 3 Duplicate Acknowledgements가 혼용되고 있다고 합니다.

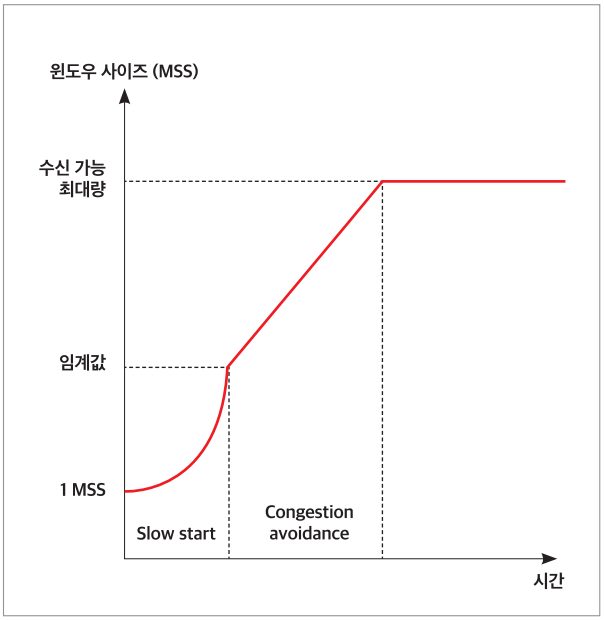

그림 3. 혼잡 윈도우 사이즈 증가 방식

지금까지 설명 드린 Slow Start, Congestion Avoidance, Congestion Detection을 모두 종합하여 상황에 따라 [그림 3]의 혼잡 윈도우 사이즈(지난 편에서 소개)를 변화시켜 나가는 혼잡제어를 수행하게 됩니다. 혼잡제어 방식에는 여러 가지가 있지만 가장 많이 알려진 방식인 TCP Tahoe와 TCP Reno에 대해 간략한 설명을 드리겠습니다.

• TCP Tahoe

[그림 4]에 표시된 TCP Tahoe 혼잡제어 방식에서는 Time Out이나 3 Duplicate Acknowledgements 발생 시 유실된 패킷을 재송신한 후에 [그림 3]의 Slow Start & Congestion Avoidance를 새롭게 시작합니다. 새롭게 시작하는 Slow Start & Congestion Avoidance에서의 Slow Start 임계값은 Time Out이나 3 Duplicate Acknowledgements가 발생한 시점의 윈도우 사이즈의 절반으로 설정([그림 4]에서 녹색과 파란색 화살표 옆에 1/2로 표시)되며, 1 MSS(Maximum Segment Size, 지난 편에서 설명)부터 Slow Start를 시작합니다.

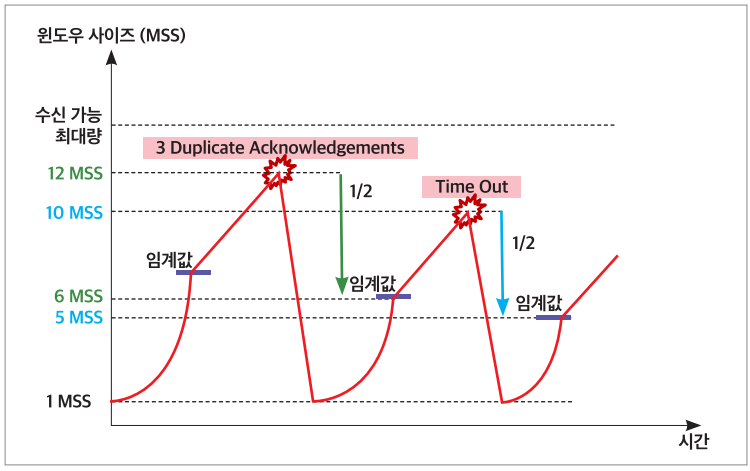

그림 4. TCP Tahoe 방식에서의 윈도우 사이즈 변화

• TCP Reno

[그림 5]의 TCP Reno 혼잡제어 방식은 TCP Tahoe와 달리 Time Out 발생과 3 Duplicate Acknowledgements 발생 후의 대처 방식이 다릅니다. 3 Duplicate Acknowledgements가 발생한 경우 Time Out 발생보다는 네트워크의 혼잡 확률이 낮다고 판단하여 Slow Start를 생략하고 곧바로 Congestion Avoidance 단계를 진행합니다. Time Out 발생 시에는 네트워크 혼잡 확률이 높다고 판단하여 TCP Tahoe처럼 1 MSS부터 Slow Start를 다시 시작합니다. Slow Start 임계값을 정하는 방식은 TCP Tahoe와 동일합니다.

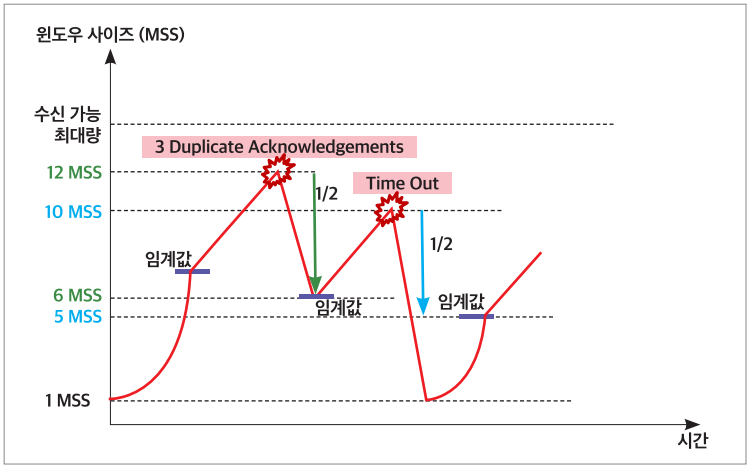

그림 5. TCP Reno 방식에서의 윈도우 사이즈 변화

지금까지의 내용으로 TCP에 관한 전반적 설명을 모두 마쳤습니다. 다음 편에서는 TCP와 UDP에 관한 속설과 사실을 비교하여 전송계층 관련 내용을 종합 정리해보겠습니다.

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다. 본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.