네트워크 개론 Part 14
: TCP(Transmission Control Protocol)의 이해 2

글. 조인준 KBS 미디어기술연구소 차장

지난 편에서는 TCP(Transmission Control Protocol)의 연결 설정 및 해제를 위한 핸드셰이크(Handshake)에 대해 다루었습니다. TCP의 핸드셰이크는 물리적 연결이 아닌 논리적 연결의 성립과 해제를 위한 것이라는 것을 다시 한번 상기시켜 드리며, 이번 편부터는 연결 성립 이후 데이터 전송이 이루어지는 방식에 관한 설명을 이어나가겠습니다.

송신 측과 수신 측 호스트 디바이스가 TCP로 핸드셰이크를 통해 논리적 연결을 설정하면 데이터를 주고받을 준비가 완료됩니다. 이제부터는 실제로 필요한 데이터를 보내거나 받으면 되므로 송신 측은 보낼 데이터를 보내고, 수신 측은 그냥 받으면 모든 것이 문제없이 잘 될 것 같은 느낌이 듭니다.
그런데 현실은 그렇지가 않습니다. 송수신 과정에 어떤 문제가 발생할 수 있고, 이런 문제 극복을 위해 어떤 방식이 고안되었는지 그 흐름을 알아보기 위해 이상적인 상황에서만 가능한 단순 송수신 방식부터 현실적 문제를 고려해가며 단계적으로 발전하여 TCP에서 주로 사용되고 있는 송수신 방식까지 차례대로 알아보겠습니다.

단순 송수신

[그림 1]은 데이터를 보내는 Sender인 호스트 A가 데이터를 수신하는 Receiver인 호스트 B에 지속해서 데이터 패킷을 보내고, 호스트 B는 수신한 데이터를 일단 수신 버퍼에 받아 두고 처리하는 상황을 가정한 것입니다.

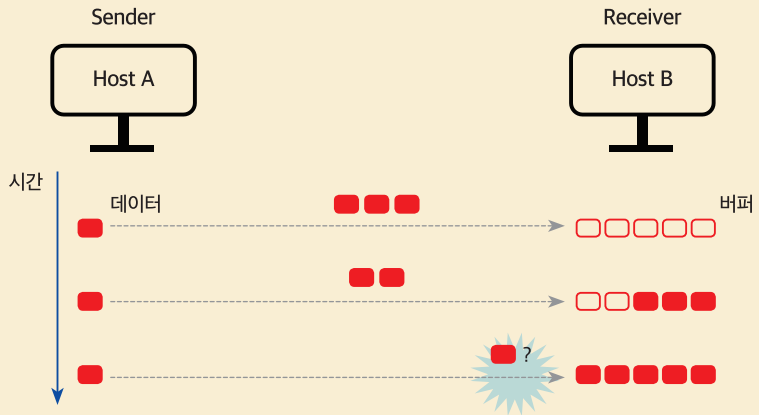

그림 1. 단순 송수신 방식과 수신 버퍼

Host B는 데이터를 일단 수신 버퍼에 받아 두고 하나씩 빼서 처리하기 때문에, 처리가 시작되지 않은 데이터는 수신 버퍼의 자리를 그대로 차지하고 있습니다. 하지만 Host B의 상황을 전혀 알 리가 없는 Host A는 계속해서 데이터를 보낼 것이고, Host B가 데이터를 수신 버퍼에서 빼내어 처리하는 속도보다 Host A가 더 빨리 데이터를 보내게 되면 [그림 1]의 하단처럼 Host B의 수신 버퍼가 모두 차버려서 Host A가 보낸 데이터를 더 이상 받을 수 없는 상황이 발생할 수 있습니다. 이런 단순 송수신의 문제점을 해결하기 위해서는 수신 측 호스트가 데이터를 수신할 수 있는 상태인지 아닌지에 관한 최소한의 정보를 알 수 있어야 합니다. 이를 위해 수신 측 호스트가 준비되었는지 송신 측 호스트가 알 수 있도록 고안된 방법이 Stop & Wait 방식입니다.

Stop & Wait

[그림 2]는 Stop & Wait 방식의 대략적인 동작을 보여주고 있습니다. Stop & Wait에서 데이터를 보내는 Sender인 호스트 A는 데이터를 수신하는 Receiver인 호스트 B에 적당한 양의 데이터 패킷을 보냅니다. 호스트 B는 수신한 데이터를 처리하여 추가로 데이터를 받을 수 있는 상황이 되면 ACK(Acknowledgement) 메시지를 Host A에 보냅니다. ACK 메시지를 받은 Host A는 Host B가 데이터를 수신할 준비가 되어있다는 것을 알았으니 적당한 양의 데이터 패킷을 추가로 보낸 후 다시 데이터 패킷을 보내기 전까지 Host B의 ACK 메시지를 기다립니다. Stop & Wait 방식은 이렇게 수신 측 호스트가 수신할 준비가 되어있는지 ACK 메시지를 통해 확인하며 데이터를 보내므로 단순 송수신과 같이 수신 버퍼가 모두 소진되어 수신할 수 없는 상황이 발생하지 않습니다.

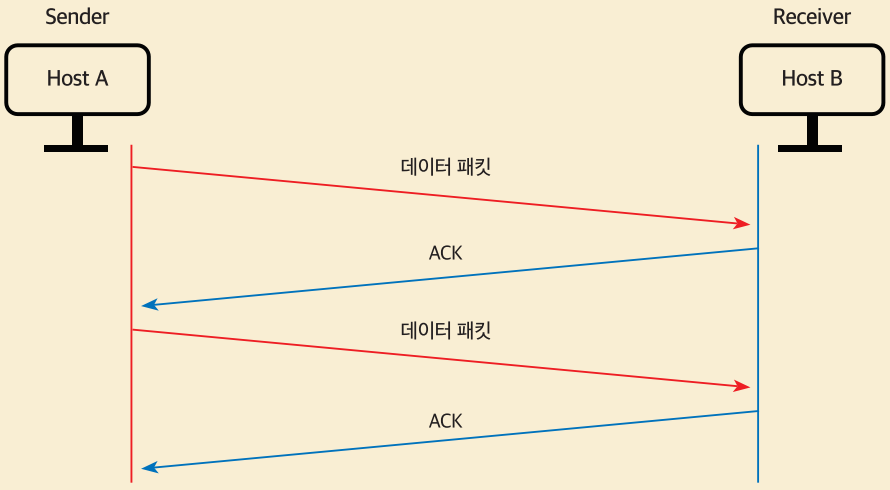

그림 2. Stop & Wait 방식

여기서 한 가지 질문을 해보겠습니다. 현실에서 Stop & Wait 방식이 실제로 동작할까요? 호스트와 호스트 사이에 완벽한 데이터 전송이 보장되는 이상적인 네트워크에서라면 Stop & Wait 방식이 동작할 것입니다. 하지만 복잡하게 연결된 현실의 네트워크에서는 [그림 3]과 같이 호스트와 호스트 사이에 데이터가 완벽히 전달되지 않을 수도 있습니다.

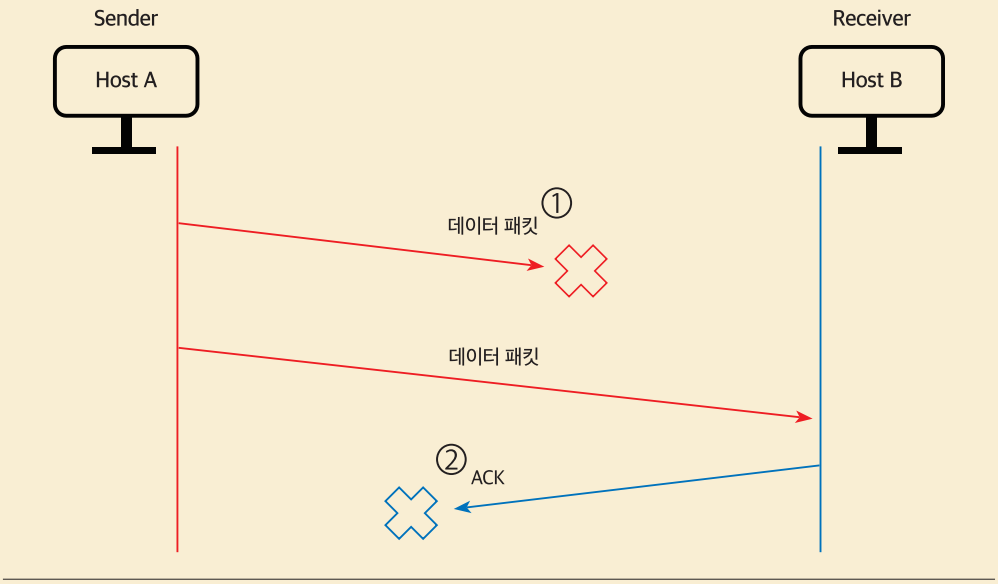

그림 3. 현실 네트워크에서 발생할 수 있는 Stop & Wait 전송오류

Stop & Wait 방식을 전송오류 가능성이 있는 현실의 복잡한 네트워크에서 사용한다면 [그림 3]의 ①로 표시된 부분과 같이 수신 측 호스트에 데이터 패킷이 도달하지 않은 경우와 ②로 표시된 부분과 같이 수신 측에서 데이터 패킷을 받아서 처리 후에 수신 가능 상태가 되어서 ACK 메시지를 보냈으나, ACK 메시지가 송신 측 호스트에 도달하지 못하는 두 가지 경우가 있을 수 있습니다. 두 경우 모두 송신 측의 호스트는 ACK 메시지를 받지 못하므로 무한히 ACK 메시지를 기다리면서 더 이상의 데이터 송수신이 이루어지지 않습니다. 이렇게 이상적인 네트워크에서만 동작이 가능한 Stop & Wait 방식을 현실의 네트워크에 맞게 보완한 방식들이 있으며, Stop & Wait ARQ, Go-Back-N ARQ, Selective Repeat ARQ 등이 그것입니다.

Stop & Wait ARQ (Automatic Repeat ReQuest)

Stop & Wait ARQ는 Stop & Wait의 기본구조를 따르지만, 타임아웃과 시퀀스 번호를 적용하여 네트워크 오류로 데이터 패킷이나 ACK 메시지 전송이 이루어지지 않았을 때 무한히 기다리는 상황에 빠지지 않도록 고안된 방식입니다. 그렇다면 타임아웃과 시퀀스 번호를 통해 어떻게 현실의 문제를 극복하였는지 몇 가지 케이스를 구분하여 설명하겠습니다.

• 전송오류 없음

[그림 4]는 송신 측 호스트가 '데이터 패킷 1'을 보낸 후 수신 측 호스트에서 보낸 ACK 메시지가 타임아웃 이전에 도착한 상황을 나타냅니다. 전송에 아무 문제가 없는 상황이므로 송신 측 호스트는 차례로 데이터 패킷들을 송신하고 이에 대한 ACK를 타임아웃 전에 받아서 순조롭게 데이터 송수신을 진행합니다. 전송 자체가 타임아웃 안에서 오류 없이 진행되고 있으므로 타임아웃에 따른 조치나 동작이 없어서 Stop & Wait과 차이가 없어 보이는 상태입니다.

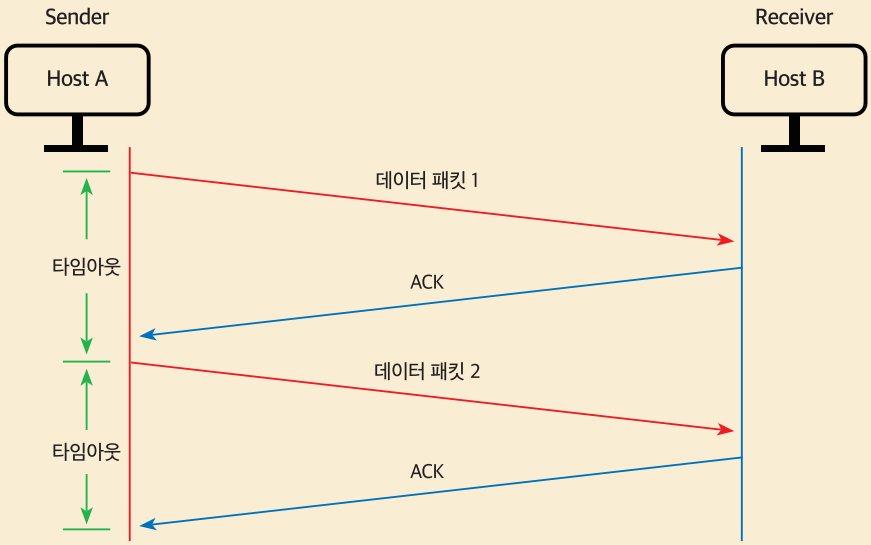

그림 4. 전송오류가 없을 때의 Stop & Wait ARQ

• 데이터 패킷 전송 실패

[그림 5]는 '데이터 패킷 1'의 전송이 실패한 경우입니다. 이때 수신 측 Host B는 데이터를 받지 못했으므로 ACK 메시지를 보내지 않고, 타임아웃까지 ACK를 받지 못한 Host A는 무언가 전송 과정에 문제가 발생했음을 알 수 있으므로 '데이터 패킷 1'을 다시 보내게 됩니다. 다행히 두 번째 보낸 '데이터 패킷 1'은 정상적으로 전송이 되어 Host B가 ACK 메시지를 보내고, 이 ACK 메시지를 받은 Host A는 그다음의 데이터 패킷들을 순차적으로 보내게 됩니다.

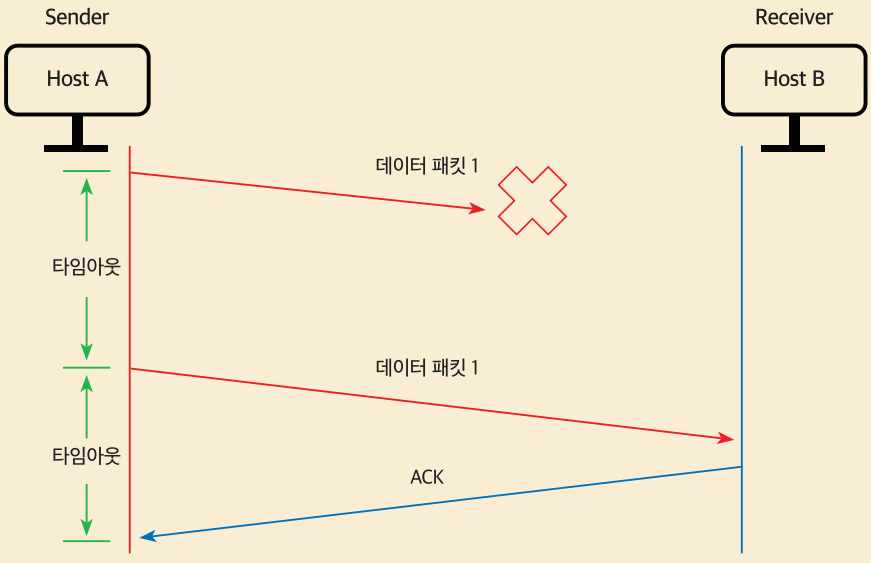

그림 5. Stop & Wait ARQ와 데이터 패킷 전송오류

• ACK 메시지 전송 실패

[그림 6]은 '데이터 패킷 1'의 전송은 문제없이 이루어졌으나 이를 수신한 Host B의 ACK 메시지의 전송 과정에 문제가 생겨 Host A가 이를 수신하지 못한 상황입니다. ACK 메시지를 수신하지 못한 Host A 입장에서는 '데이터 패킷 1'의 전송이 잘못된 건지 ACK 메시지 전송이 잘못된 건지 알 수 없으므로 '데이터 패킷 1'을 다시 보냅니다. 이후 다시 보낸 '데이터 패킷 1'의 전송이 잘 이루어져 이에 대한 ACK 메시지를 Host A가 타임아웃 이내에 수신하면 다음 데이터 전송으로 진행됩니다. 그런데 여기서 한 가지 질문이 떠오릅니다. Host B는 '데이터 패킷 1'을 두 번 받았고, 이를 모른 채로 데이터를 이어 붙인다면 전체 데이터는 잘못된 것이 됩니다. 송신된 데이터와 수신된 데이터가 동일하려면 두 번 수신된 '데이터 패킷 1' 중 하나를 버려야 합니다.

Host B가 같은 데이터를 두 번 받았는지 알 수 있는 방법이 있을까요? 시퀀스 번호를 이용하면 중복으로 수신된 데이터 패킷의 존재를 알 수 있습니다. Host A는 큰 데이터를 조각조각 송신하면서 각 조각의 송신 순서에 따라 다른 시퀀스 번호를 패킷에 포함하여 송신합니다. Host B는 '데이터 패킷 1'을 두 번 수신하였지만, 패킷 안의 시퀀스 번호를 조회하면 같은 시퀀스 번호가 두 번 나오므로 중복되는 시퀀스 번호를 갖는 패킷 중 하나만 사용하고 나머지는 버리게 됩니다. 이로써 중복으로 수신하더라도 전체 데이터를 온전히 재구성할 수 있습니다. 시퀀스 번호에 관해서는 이후 TCP에 관한 설명을 심화하면서 좀 더 구체적으로 설명할 기회가 있을 것입니다.

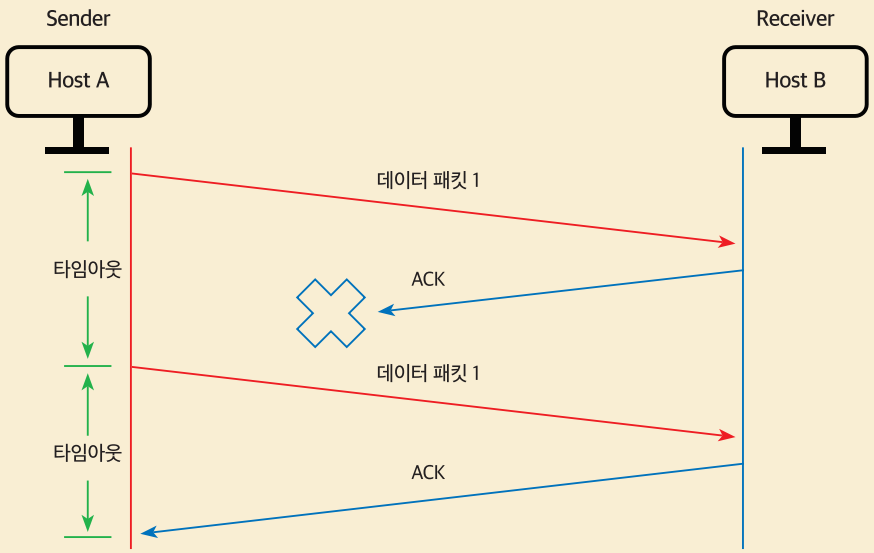

그림 6. Stop & Wait ARQ와 ACK 메시지 전송오류

• 타임아웃 발생 (전송에는 문제없음)

[그림 7]은 데이터 패킷과 ACK 메시지 전송 모두에 문제는 없지만 타임아웃이 충분히 길지 않아서 ACK를 수신하기 전에 전송 실패가 발생한 것으로 오인하여 데이터 패킷을 다시 보내는 경우입니다. 앞서 중복으로 수신된 데이터 패킷이 존재하는 경우처럼 Host B는 시퀀스 번호를 통해 중복으로 송신된 데이터를 구분하여 버리는 방법 등으로 중복 제거 처리를 합니다. 또한 ACK 메시지에는 시퀀스 번호와 연계

된 응답 번호(Acknowledgement Number)가 포함되어 있어서 Host A가 어떤 시퀀스 번호를 갖는 데이터 패킷에 대응하는 ACK 메시지인지 구분이 가능합니다. 같은 시퀀스 번호를 갖는 '데이터 패킷 1'을 두 번 보내면, 각각에 대한 ACK 메시지도 같은 응답 번호를 가지고 있으므로, 동일 ACK 메시지를 두 번 받은 Host A는 자신의 타임아웃이 짧다는 것을 알 수 있고 적절한 조치를 할 여지가 있습니다.

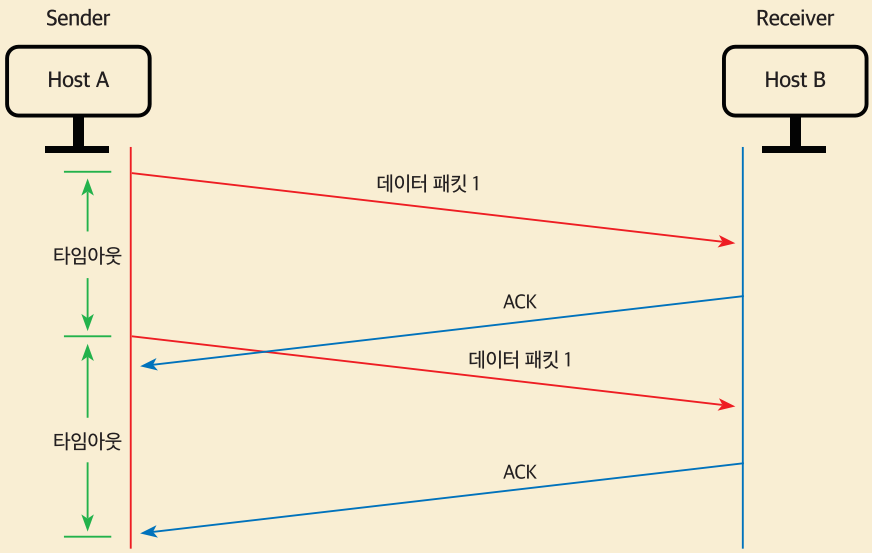

그림 7. Stop & Wait ARQ에서의 전송오류 없는 타임아웃 발생

이로써 Stop & Wait ARQ 방식까지 간단히 알아보았습니다.
다음 편에서는 Stop & Wait ARQ가 갖는 전송 비효율 해결을 위해 전송 속도를 높인 Go-Back-N ARQ 및 Selective Repeat ARQ 방식에 관해 설명 드리겠습니다.