네트워크 개론 Part 13
: TCP(Transmission Control Protocol)의 이해 I

글. 조인준 KBS 미디어기술연구소 차장

지난 편에서 전송계층 프로토콜 중 가장 널리 사용되는 프로토콜인 TCP(Transmission Control Protocol)와 UDP(User Datagram Protocol)의 개요와 간략한 특징을 소개드렸습니다.
이번 편에서는 TCP의 동작에 관한 좀 더 세부적 설명을 통해 어떤 특징들이 있는지 살펴보도록 하겠습니다. 이를 위해 [그림 1]의 TCP 헤더 구조와 이 헤더에 포함된 정보들이 의미하는 바를 알아보겠습니다.

TCP 헤더

	BYTE 1								BYTE 2								BYTE 3								BYTE 4							
❶	Source Port																Destination Port															
❷	Sequence Number																															
❸	Acknowledgement Number																															
❹	Data Offset				Reserved				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
❺	Checksum																	Urgent Pointer														
❻	Options... (0 ~ 40 Bytes)																															

그림 1. TCP 헤더 구조

[그림 1]은 TCP 헤더의 구조이며, TCP는 옵션블록([그림 1]의 ⑥)을 제외하고 20바이트로 된 헤더를 사용합니다.

• Source Port

[그림 1] ①번 행의 Source Port는 데이터를 보내는 포트 번호입니다. 이전에 포트 번호는 0~65,535 사이의 값을 가질 수 있다고 전해드렸는데, 보시면 2바이트(16비트)로 되어 있어서 0~65,535 사이의 값을 가질 수 있습니다.

• Destination Port

[그림 1] ①번 행의 Destination Port는 데이터를 받는 포트 번호입니다. Source Port와 같이 2바이트(16비트)로 되어 있어서 0~65,535 사이의 값을 가질 수 있습니다.

• Sequence Number

[그림 1] ②번 행의 Sequence Number는 전송하는 데이터의 순서를 의미하며, 4바이트(32비트)로 되어 있어서 0~4,294,967,295 사이의 값을 가질 수 있습니다. Sequence Number를 통해 수신 디바이스는 패킷으로 분리되어 들어온 데이터들의 순서를 맞춰서 원래 보낸 순서대로 데이터를 재조립할 수 있습니다. Sequence Number가 최대값인 4,294,967,295를 넘어가면 0부터 다시 시작됩니다.

• Acknowledgement Number

[그림 1] ③번 행의 Acknowledgement Number는 Sequence Number와 마찬가지로 4바이트(32비트)로 되어 있어서 0~4,294,967,295 사이의 값을 가질 수 있고 최대값을 넘어가면 0에서 다시 시작되며 데이터를 수신한 측이 데이터를 송신하는 측으로부터 기대하는 다음 Sequence Number를 의미합니다. TCP 연결의 생성과 종료를 위한 핸드셰이크(Handshake)와 연결 생성 후의 데이터 송수신에서 Acknowledgement Number가 생성되는 방식에 차이가 있으며, 이는 추후 설명하도록 하겠습니다.

• Data Offset

[그림 1] ④번 행의 Data Offset은 4비트로 된 값이며, 워드(1워드 = 4바이트 = 32비트) 단위로 환산한 헤더의 크기를 나타냅니다. 4비트라서 0~15까지의 값을 가질 수 있으나 TCP 헤더의 기본 크기가 20바이트(5워드)이기 때문에 최소값으로 5를 가지며 최대값인 15를 가지면 최대 40바이트인 ⑥번 행의 Option 블록까지 헤더에 포함할 수 있습니다.

• Reserved

[그림 1] ④번 행의 Reserved는 말 그대로 미래의 사용을 위해 남겨둔 3비트 데이터 영역입니다.

• Flags

[그림 1] ④번 행의 NS부터 FIN까지의 1비트로 된 9개 플래그들은 TCP 연결 및 데이터 관리 등을 위해 사용됩니다.

- NS, CWR, ECE : 네트워크 혼잡 상황을 알리기 위한 ECN(Explicit Congestion Notification)을 위해 사용
- URG(Urgent) : [그림 1] ⑤번 행의 Urgent Pointer의 값을 참조하여 우선적으로 처리해야 할 데이터가 있다는 것을 알림
- ACK(Acknowledgement) : TCP 연결을 설정하거나 해제하기 위한 핸드셰이크 및 데이터 수신 확인에 사용하는 플래그이며, 이 값이 1이면 [그림 1] ③번 행의 Acknowledgement Number에 값이 있음을 알림
- PSH(Push) : 데이터를 처리하는 응용프로그램에 수신된 데이터를 최대한 빠르게 전달하도록 알림
- RST(Reset) : 연결을 리셋하기 위한 플래그이며, 1로 설정된 TCP 세그먼트를 수신하는 경우 해당 디바이스는 즉시 연결을 끊음
- SYN(Synchronize) : 연결을 설정하기 위한 핸드셰이크에 사용하는 플래그
- FIN(Synchronize) : 연결 종료를 위한 요청을 나타내는 플래그

• Window Size

[그림 1] ④번 행의 Window Size는 16비트 값으로 0~65,535 사이의 값을 갖습니다. Window Size 값을 통해 매번 전송한 패킷에 대해 수신 확인을 받지 않고, 설정한 윈도우 크기만큼 수신 확인 없이 데이터를 전송할 수 있는 데이터양을 설정합니다.

• Chcecksum

[그림 1] ⑤번 행의 Checksum은 16비트로 되어있으며 데이터의 전송 과정 중에 발생할 수 있는 오류를 검출하기 위한 값입니다.

• **Urgent Pointer**

[그림 1] ⑤번 행의 Urgent Pointer는 16비트로 되어있으며 URG 플래그가 1일 때, 긴급히 처리되어야 할 데이터의 마지막 바이트 위치를 가리킵니다.

• **Options**

[그림 1] ⑥번 행의 Options 부분은 20바이트~60바이트가 될 수 있는 TCP 헤더에서 기본 20바이트 헤더를 제외한 추가적 헤더 부분을 나타내며 최대 40바이트가 될 수 있습니다.

Three-Way Handshake

TCP를 사용하여 데이터를 주고받기 위해서는 우선 Three-Way Handshake를 통해서 논리적 연결을 확보해야 합니다. TCP가 연결 지향의 프로토콜이라고 불리는 이유가 이 Handshake에 있습니다. 그럼 TCP에서 연결을 시작하기 위한 Three-Way Handshake 절차에 대해 설명 드리겠습니다.

Host A가 클라이언트로서 서버인 Host B에 연결을 시도하는 상황을 가정하겠습니다. 연결을 시작하기 위해 Host A는 [그림 1] TCP 헤더의 Source Port에 자신의 포트번호 5000과 자신의 요청에 응답해줄 Host B의 포트번호 80을 Destination Port에 적고, SYN 플래그를 1로 한 다음 Sequence Number를 랜덤으로 발생시켜 Host B에 보냅니다. [그림 2]의 예에서는 임의의 값 100을 사용하고 있습니다.

Host A가 보낸 패킷을 받은 Host B는 TCP 헤더의 Source Port에 자신의 포트번호 80을 적고, Host A의 포트번호 5000을 Destination Port에 적은 다음, SYN 및 ACK 두 개의 플래그를 모두 1로 한 다음 랜덤으로 발생시킨 Sequence Number 300 및 Host A가 보낸 패킷에 데이터는 없었지만 1바이트(Phantom Byte)를 받은 걸로 가정하여 Host A가 보낸 Sequence Number 100에 1을 더한 Acknowledge Number 101을 보냅니다. 이 패킷을 받은 Host A는 Host B의 Acknowledge Number 101에 대응하는 Sequence Number 101 및 실제로 받은 데이터는 없었지만 1바이트(Phantom Byte)를 받은 걸로 가정하여 Host B가 보낸 Sequence Number 300에 1을 더한 Acknowledge Number 301을 적어 Host B에 보내고 이로써 연결이 성립합니다. 여기서 한 가지 주의할 것은 마지막에 Host A가 보내는 Sequence Number는 랜덤으로 발생된 번호가 아니며 Host B가 보낸 Acknowledge Number(Host B가 Host A에 기대하는 다음 Sequence Number)와 같은 값이 됩니다. 이는 Three-Way Handshake에서 초기 Sequence Number는 랜덤으로 발생되지만 이후의 Sequence Number는 데이터를 수신한 디바이스가 다음의 Sequence Number로 기대하는 Acknowledge Number를 따라서 Sequence Number를 맞춰가는 원칙을 따른 것입니다.

또한, Acknowledge Number들도 이전에 Host들이 보낸 Sequence Number에 맞춰 실제로 받지는 않았지만 가상의 1바이트(Phantom Byte)를 받았다고 가정하고 그다음의 번호가 되는 Sequence Number가 되는 Sequence Number+1을 Acknowledge Number로 보낸 것입니다. [그림 2]에서는 설명을 위해 Sequence Number와 Acknowledge Number에 숫자를 사용하였지만 숫자 옆에 X와 Y로 조금 더 일반화 시킨 경우를 표시했으니 참고하시기 바랍니다.

Four-Way Handshake

TCP는 연결이 성립된 후 데이터를 모두 주고받고 나면 연결을 종료합니다. TCP의 연결종료 과정은 [그림 3]과 같은 Four-Way Handshake를 통해 이루어집니다. [그림 3]과 같이 Host A가 먼저 연결을 종료하고자 한다면 TCP 헤더의 FIN과 ACK 플래그를 1로 하여 Host B로 패킷을 보냅니다. 이때 [그림 3]의 Sequence Number 100과 Acknowledge Number 300은 이전 데이터를 주고받을 때의 번호들을

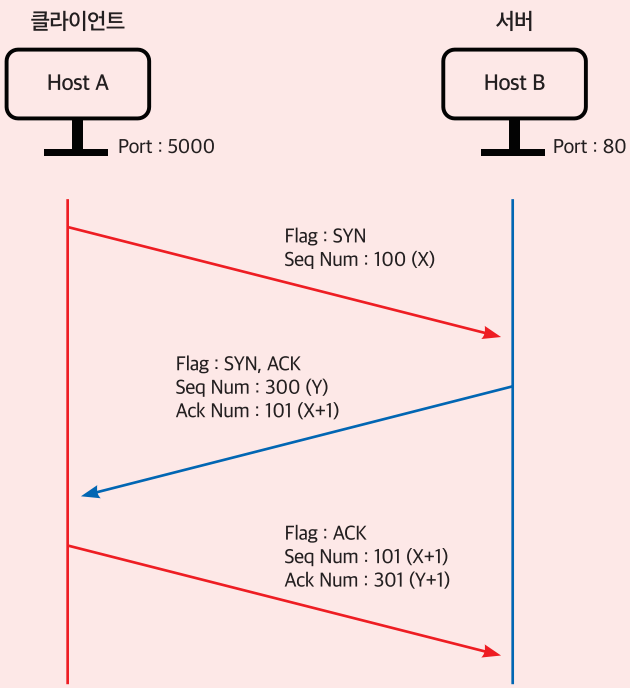

그림 2. 연결 생성을 위한 Three-Way Handshake

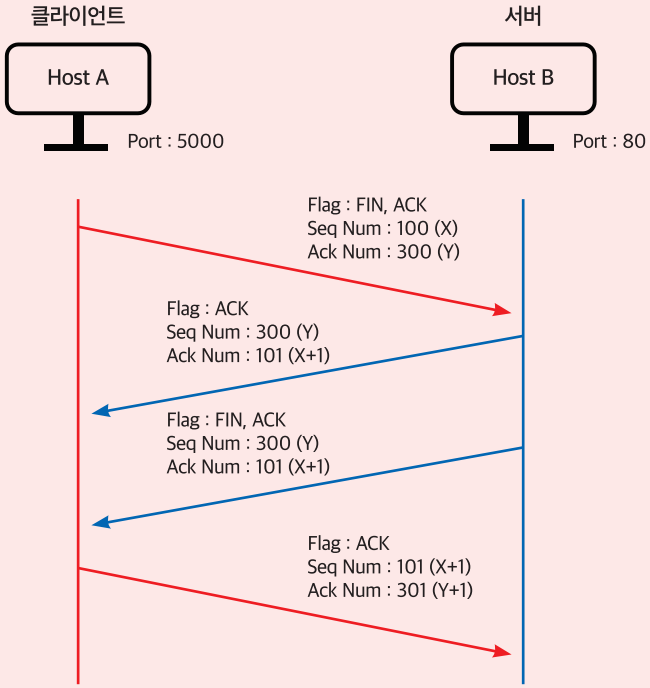

그림 3. 연결 종료를 위한 Four-Way Handshake

이어서 사용합니다. 연결 해지 요청을 받은 Host B는 ACK 플래그를 1로 하고 Host A의 Acknowledge Number 300에 맞춰 Sequence Number는 300으로, 그리고 실제 Host A가 보낸 패킷에 데이터는 없었지만 1바이트(Phantom Byte)를 받은 걸로 가정하여 Host A가 보낸 Sequence Number 100에 1을 더한 Acknowledge Number 101을 보냅니다. 이후 Host B는 다시 FIN과 ACK 플래그를 1로 하여 Host A로 패킷을 보내며, 이때 Sequence Number와 Acknowledge Number는 그사이 진행된 데이터 교환이 아무것도 없으므로 이전과 같은 Sequence Number 300과 Acknowledge Number 101을 보냅니다. Host B가 보낸 패킷을 받은 Host A는 Host B가 보낸 Acknowledge Number 101에 맞춘 Sequence Number 101과 Host B가 보낸 패킷에 데이터는 없었지만 1바이트를 받은 걸로 가정(Phantom Byte)하여 Host B가 보낸 Sequence Number 300에 1을 더한 Acknowledge Number 301을 ACK 플래그가 1로 세팅된 패킷으로 보냅니다.

[그림 3]에서도 [그림 2]와 같이 Sequence Number와 Acknowledge Number에 숫자를 사용하였지만 숫자 옆에 X와 Y로 조금 더 일반화 시킨 경우를 표시했으니 참고하시기 바랍니다. 지금까지 설명 드린 바와 같이 네 번의 패킷 교환으로 TCP 연결은 종료가 되며 다시 연결을 시작하려면 [그림 2]와 같은 Three-way Handshake로 다시 시작해야 합니다.

여기까지 TCP를 좀 더 상세하게 이해하기 위해 TCP 헤더의 내용과 연결 시작 및 종료를 위한 Three-way Handshake 및 Four-way Handshake에 대해 설명 드렸습니다.

다음 편에서는 실제로 TCP를 통해 데이터를 주고받는 과정에 대한 이야기로 다시 찾아뵙겠습니다.