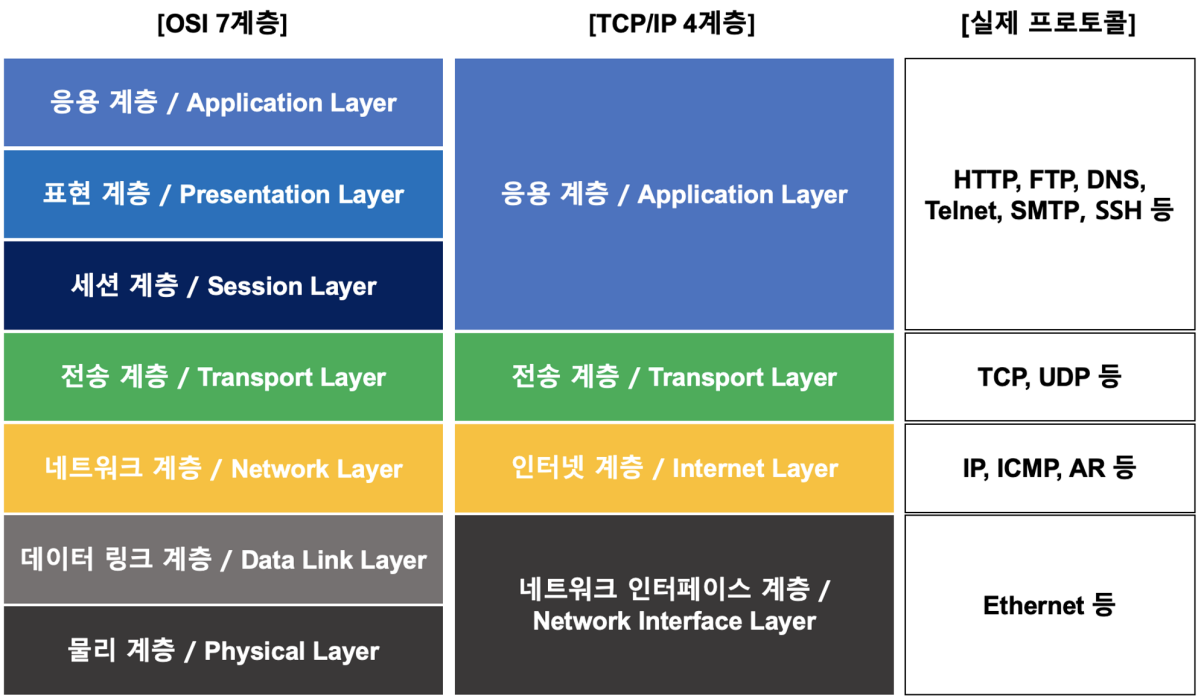
[Network] TCP/IP 4계층, TCP와 UDP, 3-way handshake

OSI 7계층, TCP/IP 4계층



OSI 7계층과 TCP/IP 4계층을 비교하기 전에 먼저 공통적인 부분부터 살펴보자면,

- OSI 7계층과 TCP/IP 4계층 모델에서 각 계층은 하위 계층의 기능을 이용하고, 상위 계층에게 기능을 제공한다.

    - ex) **HTTP는 TCP과 IP을 이용해서 작동한다.**
- 일반적으로 상위 계층의 프로토콜은 소프트웨어로, 하위 계층의 프로토콜은 하드웨어로 구현된다.
- 이렇게 프로토콜을 계층화하는 것의 장점은, 통신이 일어나는 과정을 단계별로 파악할 수 있고 문제를 작은 조각으로 나눠서(모듈화) 다른 계층에 영향이 없이 문제를 해결할 수 있다는 것이다.

OSI 7계층 (OSI 모델)

- 표준화된 네트워크 프로토콜을 설계하기 위해 만든 모델
    - But 실제로는 네트워크에서 사용되는 프로토콜들과 차이가 있어 널리 채택되지는 않음
- 네트워크의 구조와 흐름을 이해하기 쉽다.

TCP/IP 4계층 (TCP/IP 모델)

- 인터넷에서 통신을 하기 위한 프로토콜을 설명하기 위해 만든 모델
- 실제로 인터넷에서 사용되는 프로토콜들을 기반으로, 인터넷의 발전에 따라 계속해서 업데이트되어 왔기 때문에 실제로 사용되고 있는 프로토콜에 대한 특징을 파악하기 쉽다.

OSI 7계층과 TCP/IP 4계층 비교

- OSI 모델은 TCP/IP 모델의 응용 계층을 응용 계층, 표현 계층, 세션 계층 3개로 나눠서 표현
- OSI 모델은 TCP/IP 모델의 네트워크 인터페이스 계층(링크 계층)을 데이터 링크 계층, 물리 계층으로 나눠서 표현
- OSI 모델은 TCP/IP 모델의 인터넷 계층을 네트워크 계층이라고 부름

▼ cf) TCP/IP와 TCP, IP

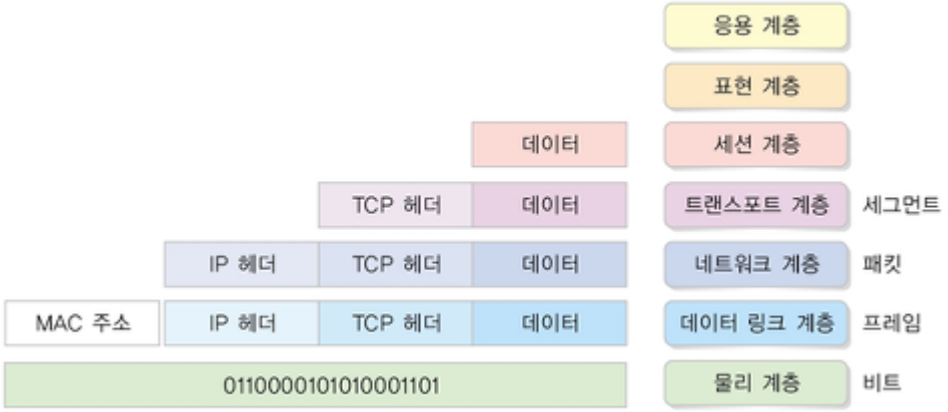
TCP/IP라는 것이 모호하게 느껴질 수 있다.

TCP, IP는 프로토콜이고, TCP/IP는 프로토콜 스위트이다.

> **프로토콜 스위트**: '여러 프로토콜들의 모음집'. 즉, 프로토콜들의 계층화된 구조를 말한다. 각 계층은 네트워크 커뮤니케이션에서 맡고 있는 역할에 따라 나누어져 있다.
> OSI 7계층, TCP/IP 4계층 모두 프로토콜 스위트이다.

TCP/IP 프로토콜 스위트는 네트워크 구조를 파악하거나 설계에 도움을 주기 위해 만들어진 "모델"이다. 이 모델의 중심적인 프로토콜이 TCP와 IP이며, 이 둘이 다른 계층의 프로토콜들과 긴밀하게 연결되어있기 때문에 프로토콜 스위트에 TCP/IP라는 이름이 붙여지게 된 것이다.

캡슐화 / 역캡슐화



- **캡슐화**: 상위 계층의 헤더와 데이터를 하위 계층의 데이터 부분에 포함시키고, 해당 계층의 헤더를 삽입하는 과정을 말함
- **비캡슐화**: 하위 계층에서 상위 계층으로 가며 각 계층의 헤더 부분을 제거하는 과정을 말함

▼ 사용자가 HTTP를 통해 웹 서버에 있는 데이터를 요청한다면?

이는 컴퓨터를 통해 다른 컴퓨터로 데이터를 요청하는 것이다.

(TCP/IP 4계층 기준) 응용 계층에서 전송 계층으로 사용자가 보내는 요청값들이 **캡슐화** 과정을 거쳐서 전달되고, 다시 링크 계층을 통해 해당 서버와 통신을 하고, 해당 서버의 링크 계층으로부터 어플리케이션까지 **비캡슐화** 과정을 거쳐 데이터가 전송된다.

▼ cf) PDU (Protocol Data Unit)

PDU는 네트워크의 어떠한 계층에서 계층으로 데이터가 전달될 때 한 덩어리의 단위를 말한다.

PDU는 제어 관련 정보들이 포함된 '헤더', 데이터를 의미하는 '페이로드'로 구성되어 있으며, 계층마다 부르는 명칭이 다르다.

- 응용 계층: 메세지
- 전송 계층: 세그먼트(TCP), 데이터그램(UDP)
- 인터넷 계층: 패킷
- 링크 계층: 프레임(데이터 링크 계층), 비트(물리 계층)

TCP vs UDP

TCP	UDP
신뢰성 보장 (3-way handeshake) (흐름제어, 혼잡제어, 오류 제어)	신뢰성 보장 X
속도 느림	속도 빠름
연결지향 (3-way handshake)	비연결성
전송 순서 보장	전송 순서 보장 X
신뢰성이 중요한 통신에 쓰임 ex) HTTP, File 전송 등	속도/실시간성이 중요한 통신에 쓰임 ex) 동영상 스트리밍 등

TCP: 연결형, 신뢰성 전송 프로토콜

- 연결지향적 서비스를 제공하기 위한 TCP의 통신 과정

  1) connection setup (TCP 연결 초기화): 3-way handshaking을 통해 두 호스트의 전송 계층 사이에 논리적 연결을 설립한다.

  2) data transfer (데이터 전송): 데이터 전송을 한다.

  3) connection termination (TCP 연결 종료): 데이터 전송 완료 시 4-way handshaking을 통해 연결을 해제한다.

- 신뢰성 있는 서비스를 제공하기 위해 TCP는 오류제어, 흐름제어, 혼잡제어 등을 통해 전체 스트림을 순서에 맞고 오류 없이, 또한 부분적인 손실이나 중복 없이 전송하는 것을 보장한다.
  - 흐름제어: 데이터를 보내는 속도와 데이터를 받는 속도의 균형을 맞추는 것

- 오류제어: 훼손된 segment의 감지 및 재전송, 손실된 segment의 재전송, 순서가 맞지 않게 도착한 segment를 정렬하고 중복 segment 감지 및 폐기를 한다. 이는 TCP header의 checksum, 확인응답, 타임-아웃 등을 통해 수행된다.

UDP: 비연결형, 비신뢰성 전송 프로토콜

- 비연결형: 논리적 연결을 설립하지 않고 datagram을 전송
- 흐름제어, 오류제어, 혼잡 제어를 제공하지 않는 간단한 프로토콜으로, 이러한 단순성으로 적은 양의 오버헤드를 갖기 때문에 작은 메시지를 보내거나 신뢰성을 크게 고려하지 않아도 되는 상황에서 사용한다.

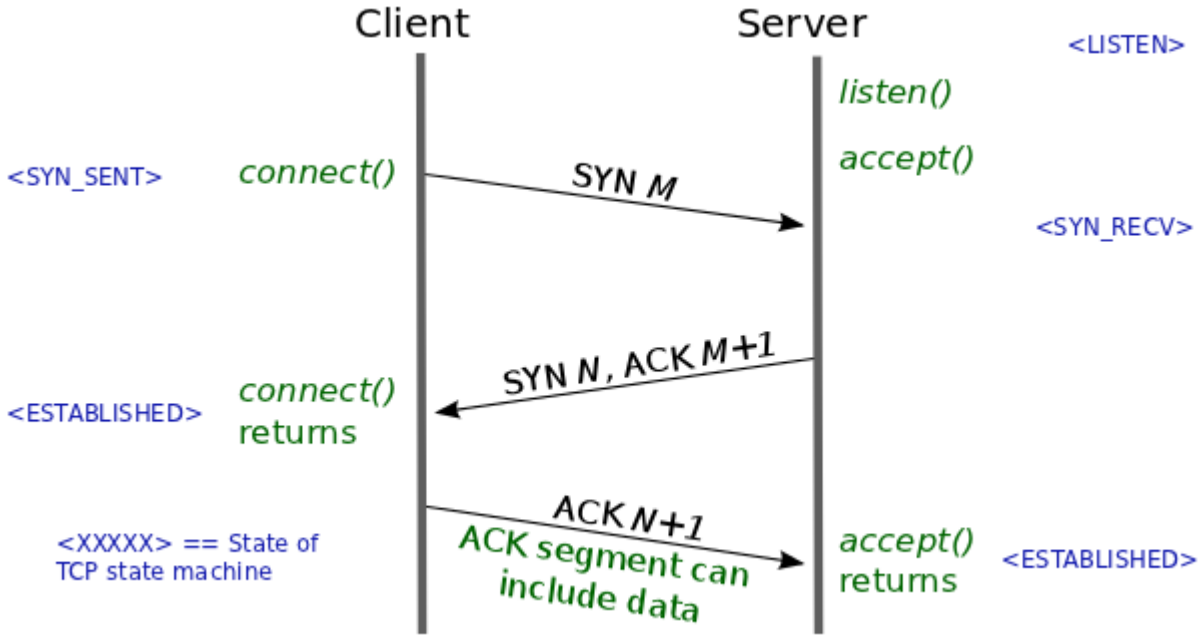
TCP의 3-way handshake

TCP는 신뢰성을 확보할 때 3-way handshake라는 작업을 진행한다.

3-way handeshake는 정확한 정보 전송을 위해 상대방 컴퓨터와 세션을 수립하는(연결을 하는) 과정이다.

3-way handshake를 간단히 표현하면 다음과 같다.

1. Client -> Server : 내 말 들려?
2. Server -> Client : 어 잘 들려! 내 말은 들려?
3. Client -> Server : 잘 들려!



위 그림처럼 클라이언트와 서버가 통신할 때 다음과 같은 세 단계의 과정을 거친다.

- cf 1) ISN: 연결 확인을 위해 보내는 임의의 시퀀스번호로, 장치마다 다를 수 있다.

▼ cf 2) 상태 설명

상태	설명
CLOSED	연결 수립을 시작하기 전의 기본 상태 (연결 없음)
LISTEN	포트가 열린 상태로 연결 요청 대기 중
SYN-SENT	SYN을 요청한 상태
SYN-RECEIVED	SYN 요청을 받고 상대방의 응답을 기다리는 중
ESTABLISHED	연결 수립이 완료된 상태, 서로 데이터를 교환할 수 있다.

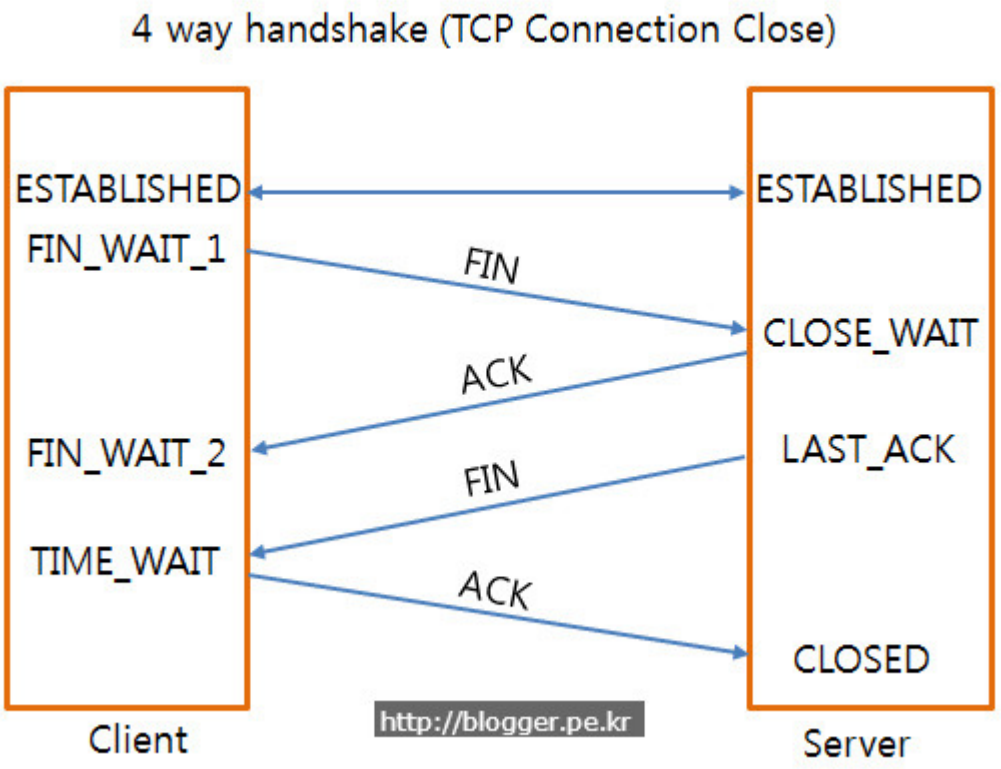
1) SYN 단계: 클라이언트는 서버에 접속을 요청하는 SYN(M) 패킷을 보낸다. 이 때, M은 클라이언트의 ISN이다.

2) SYN + ACK 단계: 서버는 클라이언트의 SYN을 수신하고, 클라이언트에 SYN(N)과 ACK(M+1)을 보낸다. 이 때, N은 서버의 ISN이며 ACK는 클라이언트의 ISN에 1을 더해준 값이다.

3) ACK 단계: 클라이언트는 서버가 보낸 패킷을 받고 ACK(N+1)을 서버에 보낸다. 이 때 ACK는 서버의 ISN에 1을 더해준 값이다.

TCP의 4-way handshake

4-way handeshake는 TCP 연결을 종료하는 과정이다.



1) 클라이언트가 연결을 닫으려고 할 때, FIN 세그먼트를 서버로 보낸다.

- 이 때, 클라이언트는 FIN_WAIT_1 상태로 들어가고 서버의 응답을 기다린다.

2) 서버는 클라이언트의 요청을 받고, 알겠다는 확인 메세지로 ACK라는 승인 세그먼트를 클라이언트로 보낸다.

- 이 때, 서버는 CLOSE_WAIT 상태로 들어간다. 클라이언트는 세그먼트를 받으면 FIN_WAIT_2 상태에 들어간다.

3) 데이터를 모두 보내고 통신이 끝났으면, 서버는 연결이 종료되었다는 의미로 FIN 세그먼트를 클라이언트로 보낸다.

4) 클라이언트는 서버로 다시 ACK(종료 메세지를 확인하였다는 의미)를 보낸다. 서버는 ACK를 받고 CLOSED 상태가 된다. 이후 클라이언트는 아직 서버로부터 받지 못한 데이터가 있을 것을 대비하여 일정 시간을 대기한 후(TIME_WAIT), 연결이 CLOSED되면 클라이언트와 서버의 모든 자원 연결이 해제된다.