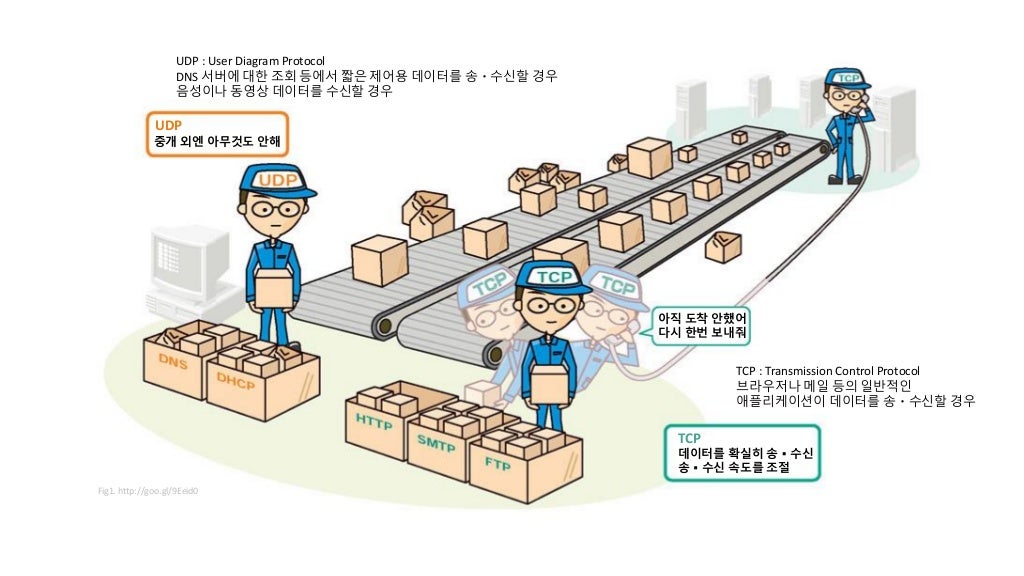
**1. 전송계층**

TCP와 UDP는 TCP/IP의 전송계층에서 사용되는 프로토콜이다. **전송계층은 IP에 의해 전달되는 패킷의 오류를 검사하고 재전송 요구 등의 제어를 담당**하는 계층이다.

**2. TCP vs UDP**

TCP는 Transmission Control Protocol의 약자이고, UDP는 User Datagram Protocol의 약자이다. 두 프로토콜은 모두 [패킷](https://velog.io/@hidaehyunlee/TCP-%EC%99%80-UDP-%EC%9D%98-%EC%B0%A8%EC%9D%B4)을 한 컴퓨터에서 다른 컴퓨터로 전달해주는 IP 프로토콜을 기반으로 구현되어 있지만, 서로 다른 특징을 가지고 있다.

**신뢰성이 요구되는 애플리케이션에서는 TCP를 사용**하고 **간단한 데이터를 빠른 속도로 전송하고자 하는 애플리케이션에서는 UDP를 사용**한다.



| **TCP** | **UDP** |
| --- | --- |
| Connection-oriented protocol (연결지향형 프로토콜) | Connection-less protocol (비 연결지향형 프로토콜) |
| Connection by **byte** stream (바이트 스트림을 통한 연결) | Connection by **message** stream (메세지 스트림을 통한 연결) |
| Congestion / Flow control (혼잡제어, 흐름제어) | NO Congestion / Flow control (혼잡제어와 흐름제어 지원 X) |
| Ordered, Lower speed (순서 보장, 상대적으로 느림) | Not ordered, Higer speed (순서 보장되지 않음, 상대적으로 빠름) |
| Reliable data transmission (신뢰성 있는 데이터 전송 - 안정적) | Unreliable data transmission (데이터 전송 보장 X) |
| TCP packet : Segment (세그먼트 TCP 패킷) | UDP packet : Datagram (데이터그램 UDP 패킷) |
| HTTP, Email, File transfer 에서 사용 | DNS, Broadcasting (도메인, 실시간 동영상 서비스에서 사용) |

**3. TCP (Transmission Control Protocol)**

TCP는 네트워크 계층 중 전송 계층에서 사용하는 프로토콜로서, 장치들 사이에 논리적인 접속을 성립(establish)하기 위하여 연결을 설정하여 **신뢰성을 보장하는 연결형 서비스** 이다. TCP는 네트워크에 연결된 컴퓨터에서 실행되는 프로그램 간에 **일련의 옥텟(데이터, 메세지, 세그먼트라는 블록 단위)를 안정적으로, 순서대로, 에러없이 교환**할 수 있게 한다.

**3.1. TCP의 특징**

**연결형 서비스**

연결형 서비스로 가상 회선 방식을 제공한다.

* 3-way handshaking 과정을 통해 연결을 설정
* 4-way handshaking 을 통해 연결을 해제.

**흐름제어(Flow control)**

데이터 처리 속도를 조절하여 수신자의 버퍼 오버플로우를 방지

* 송신하는 곳에서 감당이 안되게 많은 데이터를 빠르게 보내 수신하는 곳에서 문제가 일어나는 것을 막는다.
* 수신자가 윈도우크기(Window Size) 값을 통해 수신량을 정할 수 있다.

**혼잡제어(Congestion control)**

네트워크 내의 패킷 수가 넘치게 증가하지 않도록 방지

* 정보의 소통량이 과다하면 패킷을 조금만 전송하여 혼잡 붕괴 현상이 일어나는 것을 막는다.

**신뢰성이 높은 전송(Reliable transmission)**

* Dupack-based retransmission
  + 정상적인 상황에서는 ACK 값이 연속적으로 전송되어야 한다.
  + 그러나 ACK값이 중복으로 올 경우 패킷 이상을 감지하고 재전송을 요청한다.
* Timeout-based retransmission
  + 일정시간동안 ACK 값이 수신을 못할 경우 재전송을 요청한다.

**전이중, 점대점 방식**

* **전이중 (Full-Duplex)**  
  전송이 양방향으로 동시에 일어날 수 있다.
* **점대점 (Point to Point)**  
  각 연결이 정확히 2개의 종단점을 가지고 있다.

=> 멀티캐스팅이나 브로드캐스팅을 지원하지 않는다.

**3.2. TCP Header 정보**

테이블이(가) 표시된 사진

자동 생성된 설명

응용 계층으로부터 데이터를 받은 TCP는 헤더를 추가한 후에 이를 IP로 보낸다. 헤더에는 아래 표와 같은 정보가 포함된다.

| **필 드** | **내 용** | **크기** |
| --- | --- | --- |
| 송수신자의 포트 번호 | TCP로 연결되는 가상 회선 양단의 송수신 프로세스에 할당되는 **포트 주소** | 16 |
| 시퀀스 번호(Sequence Number) | 송신자가 지정하는 순서 번호, **전송되는 바이트 수**를 기준으로 증가. SYN = 1 : 초기 시퀀스 번호가 된다. ACK 번호는 이 값에 1을 더한 값. SYN = 0 : 현재 세션의 이 세그먼트 데이터의 최초 바이트 값의 누적 시퀀스 번호 | 32 |
| 응답 번호(ACK Number) | 수신 프로세스가 제대로 **수신한 바이트의 수**를 응답하기 위해 사용. | 32 |
| 데이터 오프셋(Data Offset) | TCP 세그먼트의 시작 위치를 기준으로 **데이터의 시작 위치**를 표현(TCP 헤더의 크기) | 4 |
| 예약 필드(Reserved) | 사용을 하지 않지만 나중을 위한 예약 필드이며 0으로 채워져야한다. | 6 |
| 제어 비트(Flag Bit) | SYN, ACK, FIN 등의 제어 번호 -> 아래 추가 설명 참조 | 6 |
| 윈도우 크기(Window) | **수신 윈도우의 버퍼 크기**를 지정할 때 사용. 0이면 송신 프로세스의 전송 중지 | 16 |
| 체크섬(Checksum) | TCP 세그먼트에 포함되는 프로토콜 헤더와 **데이터에 대한 오류 검출** 용도 | 16 |
| 긴급 위치(Urgent Pointer) | 긴급 데이터를 처리하기 위함, URG 플래그 비트가 지정된 경우에만 유효 | 16 |

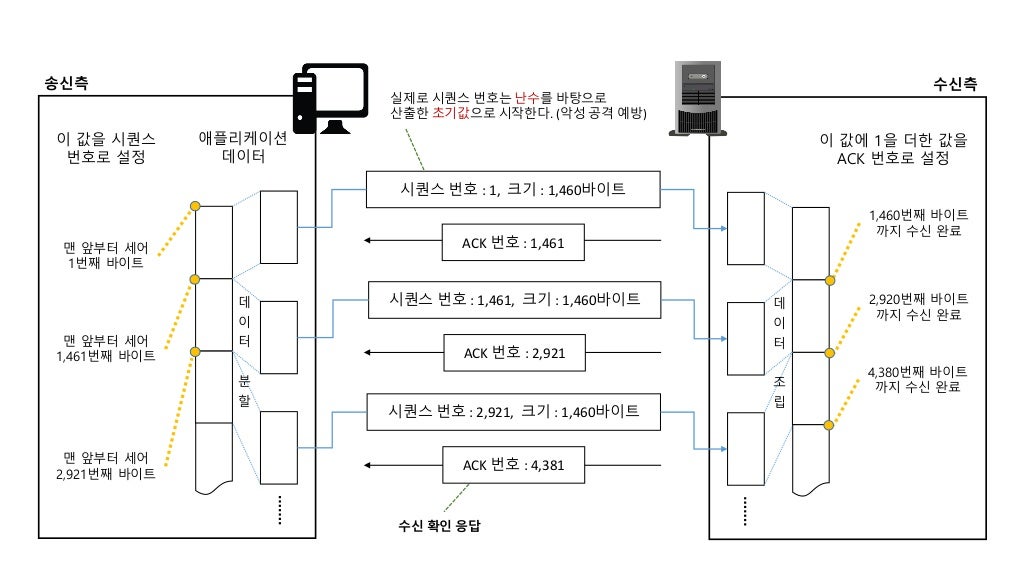
**제어 비트(Flag Bit) 정보**

| **종류** | **내 용** |
| --- | --- |
| URG | 긴급 위치를 필드가 유효한지 설정 |
| ACK | 응답 번호 필드가 유효한지 설정. 클라이언트가 보낸 최초의 SYN 패킷 이후에 전송되는 모든 패킷은 이 플래그가 설정되어야 한다. 자세한 내용은 아래 추가 설명 참조 |
| PSH | 수신 애플리케이션에 버퍼링된 데이터를 상위 계층에 즉시 전달할 때 |
| RST | 연결의 리셋이나 유효하지 않은 세그먼트에 대한 응답용 |
| SYN | 연결 설정 요구. 동기화 시퀀스 번호. 양쪽이 보낸 최초의 패킷에만 이 플래그가 설정되어 있어야 한다. |
| FIN | 더 이상 전송할 데이터가 없을 때 연결 종료 의사 표시 |

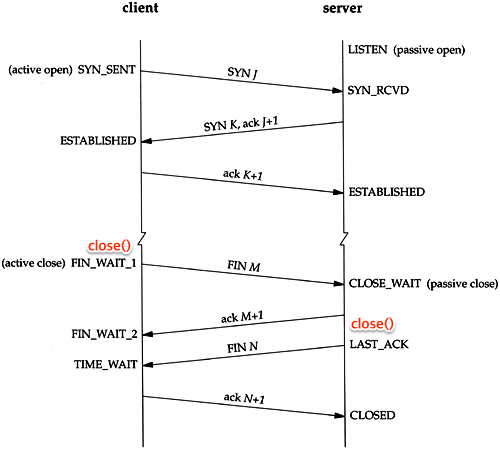
**ACK 제어비트**

* ACK는 송신측에 대하여 **수신측에서 긍정 응답**으로 보내지는 전송 제어용 캐릭터
* ACK 번호를 사용하여 패킷이 도착했는지 확인한다.

-> 송신한 패킷이 제대로 도착하지 않았으면 **재송신**을 요구한다.



**3.3. TCP의 연결 및 해제 과정**



**TCP Connection (3-way handshake)**

1. 먼저 open()을 실행한 클라이언트가 SYN을 보내고 SYN\_SENT 상태로 대기한다.
2. 서버는 SYN\_RCVD 상태로 바꾸고 SYN과 응답 ACK를 보낸다.
3. SYN과 응답 ACK을 받은 클라이언트는 ESTABLISHED 상태로 변경하고 서버에게 응답 ACK를 보낸다.
4. 응답 ACK를 받은 서버는 ESTABLISHED 상태로 변경한다.

**TCP Disconnection (4-way handshake)**

1. 먼저 close()를 실행한 클라이언트가 FIN을 보내고 FIN\_WAIT1 상태로 대기한다.
2. 서버는 CLOSE\_WAIT으로 바꾸고 응답 ACK를 전달한다. 동시에 해당 포트에 연결되어 있는 어플리케이션에게 close()를 요청한다.
3. ACK를 받은 클라이언트는 상태를 FIN\_WAIT2로 변경한다.
4. close() 요청을 받은 서버 어플리케이션은 종료 프로세스를 진행하고 FIN을 클라이언트에 보내 LAST\_ACK 상태로 바꾼다.
5. FIN을 받은 클라이언트는 ACK를 서버에 다시 전송하고 TIME\_WAIT으로 상태를 바꾼다. TIME\_WAIT에서 일정 시간이 지나면 CLOSED된다. ACK를 받은 서버도 포트를 CLOSED로 닫는다.

**주의**

* 반드시 서버만 CLOSE\_WAIT 상태를 갖는 것은 아니다.
* 서버가 먼저 종료하겠다고 FIN을 보낼 수 있고, 이런 경우 서버가 FIN\_WAIT1 상태가 됩니다.
* 누가 먼저 close를 요청하느냐에 따라 상태가 달라질 수 있다.

**4. UPD Header 정보**

응용 계층으로부터 데이터 받은 UDP도 UDP 헤더를 추가한 후에 이를 IP로 보낸다.

| **필 드** | **크 기** | **내 용** |
| --- | --- | --- |
| 송신자의 포트 번호 | 16 | 데이터를 보내는 애플리케이션의 포트 번호 |
| 수신자의 포트 번호 | 16 | 데이터를 받을 애플리케이션의 포트 번호 |
| 데이터의 길이 | 16 | UDP 헤더와 데이터의 총 길이 |
| 체크섬(Checksum) | 16 | 데이터 오류 검사에 사용 |

TCP 헤더와 다르게 UDP 헤더에는 포함된 정보가 부실한 느낌마저 든다.  
UDP는 수신자가 데이터를 받는지 마는지 관심이 없기 때문이다. 즉, 신뢰성을 보장해주지 않지만 간단하고 속도가 빠른 것이 특징이다.

**5. 정리**

**공통점**

| **TCP(Transfer Control Protocol) | UDP(User Datagram Protocol)** |
| --- |
| 포트 번호를 이용하여 주소를 지정 |
| 데이터 오류 검사를 위한 체크섬 존재 |

**차이점**

| **TCP(Transfer Control Protocol)** | **UDP(User Datagram Protocol)** |
| --- | --- |
| 연결이 성공해야 통신 가능(연결형 프로토콜) | 비연결형 프로토콜(연결 없이 통신이 가능) |
| 데이터의 경계를 구분하지 않음(Byte-Stream Service) | 데이터의 경계를 구분함(Datagram Service) |
| 신뢰성 있는 데이터 전송(데이터의 재전송 존재) | 비신뢰성 있는 데이터 전송(데이터의 재전송 없음) |
| 일 대 일(Unicast) 통신 | 일 대 일, 일 대 다(Broadcast), 다 대 다(Multicast) 통신 |