CS Study

Network - OSI 7계층 PDU

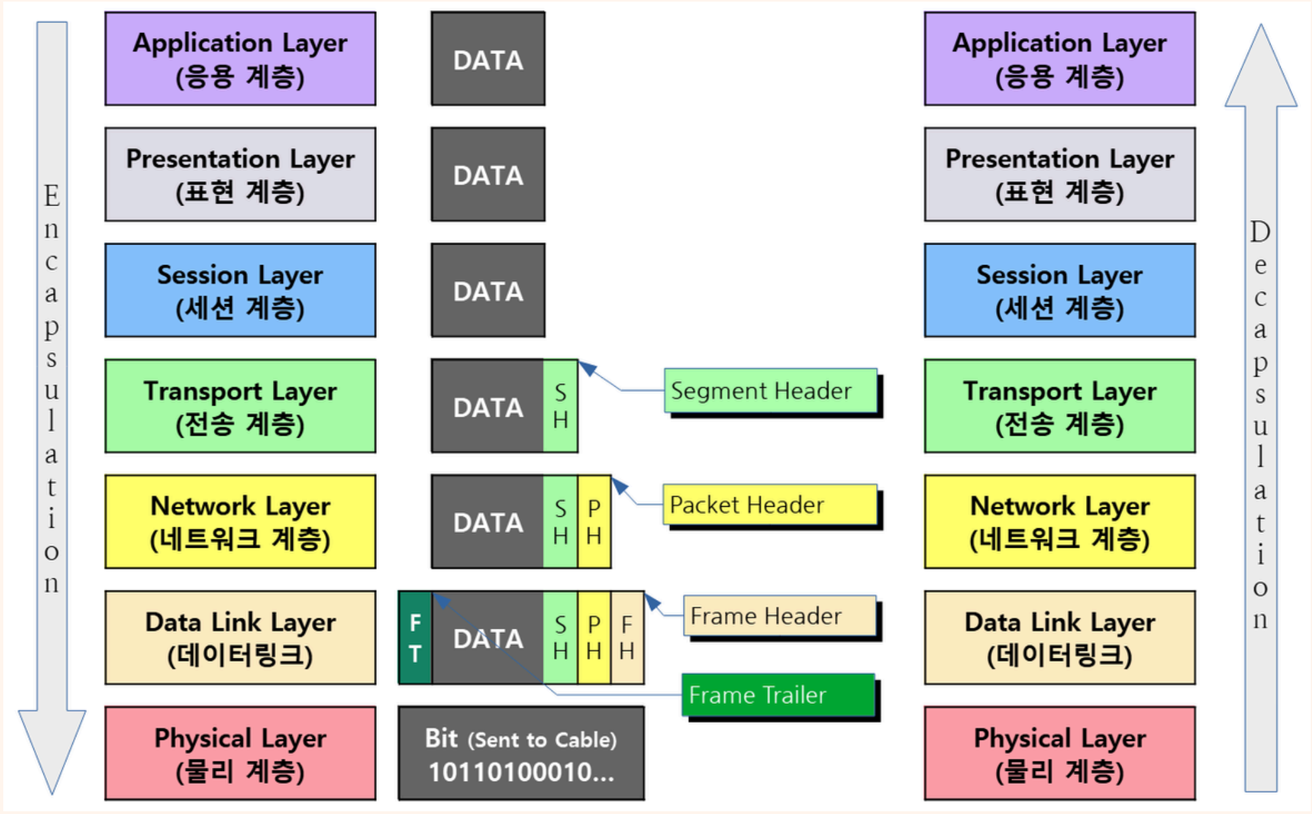
목차

목차

OSI 7계층 전송 흐름

OSI 7계층 전송 흐름

- 송신자에서 데이터가 각 계층별로 **캡슐화 되어 헤더가 차례로 추가됨**
- 물리 계층에서 **신호로 전환되어 전송**됨
- 수신자에서 데이터가 각 계층별로 **역캡슐화 되어 헤더가 차례로 제거됨**

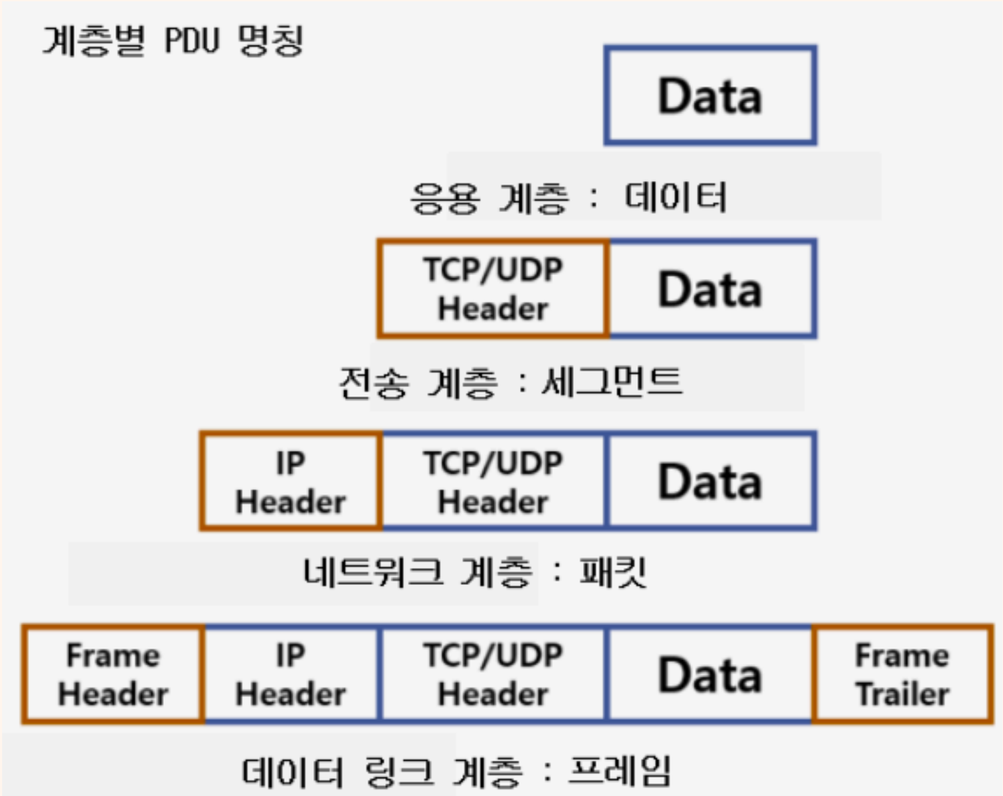


OSI 7계층 전송 흐름

각 계층 PDU (Protocol Data Unit)

각 계층은 **데이터 전송 단위**를 가지고 있음

- 응용, 표현, 세션: **데이터(Data)**
- 전송: **세그먼트(Segment), 데이터그램(Datagram)**
- 네트워크: **패킷(Packet)**
- 데이터 링크: **프레임(Frame)**
- 물리: **비트(Bit)**



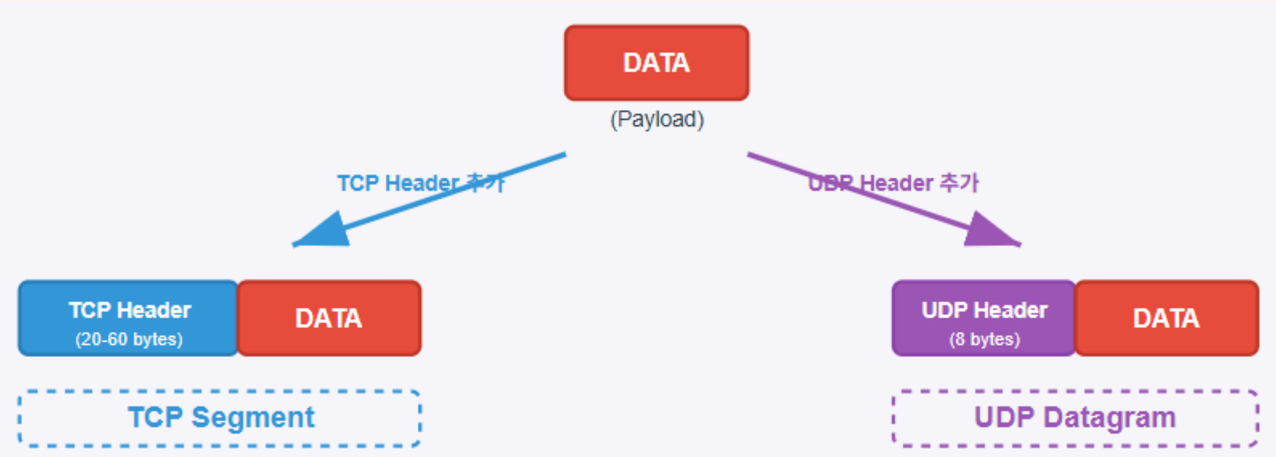
목차

Segment / Datagram

구조

4계층 PDU는 **사용 프로토콜에 따라 두 가지**로 나뉨

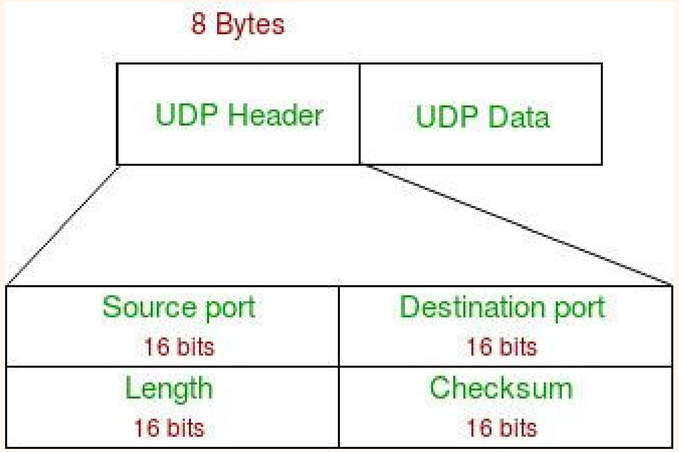
- **세그먼트 (Segment)**:
  - **TCP 프로토콜**에서 사용하는 데이터 단위
- **데이터그램 (Datagram)**:
  - **UDP 프로토콜**에서 사용하는 데이터 단위

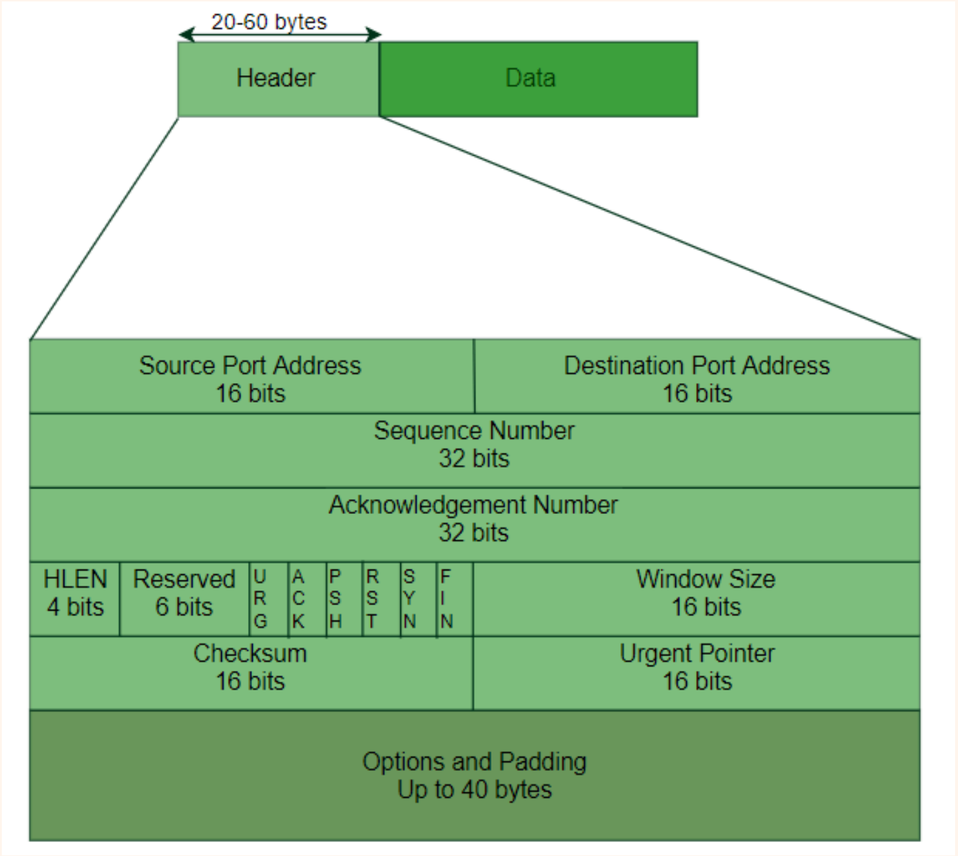


Segment / Datagram

Segment / Datagram 구조

각각 **TCP/UDP 헤더 + 데이터**로 나뉨
데이터는 **상위 계층에서 내려온 데이터**

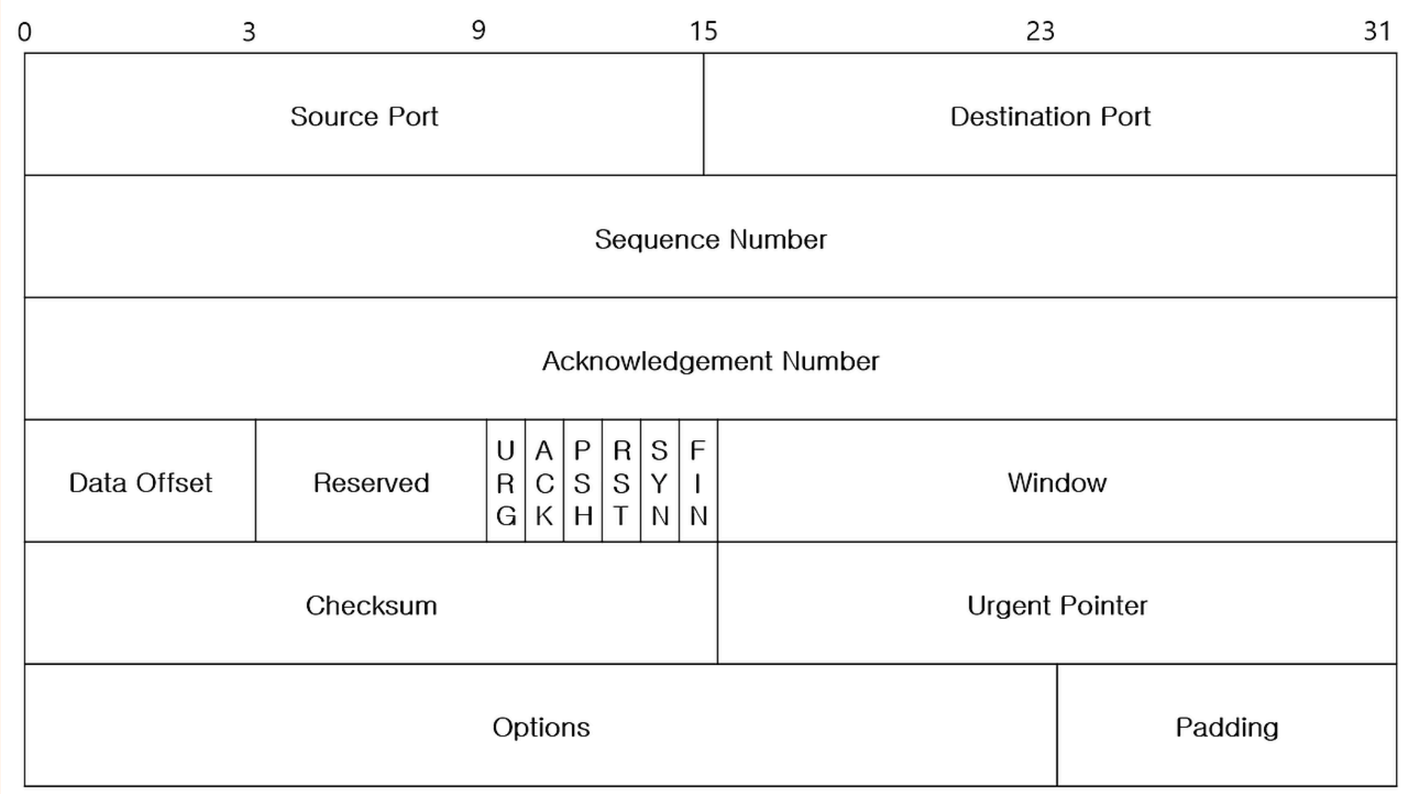




Segment / Datagram

TCP 헤더

기본적으로 **20바이트**
옵션 붙으면 **최대 60 바이트**

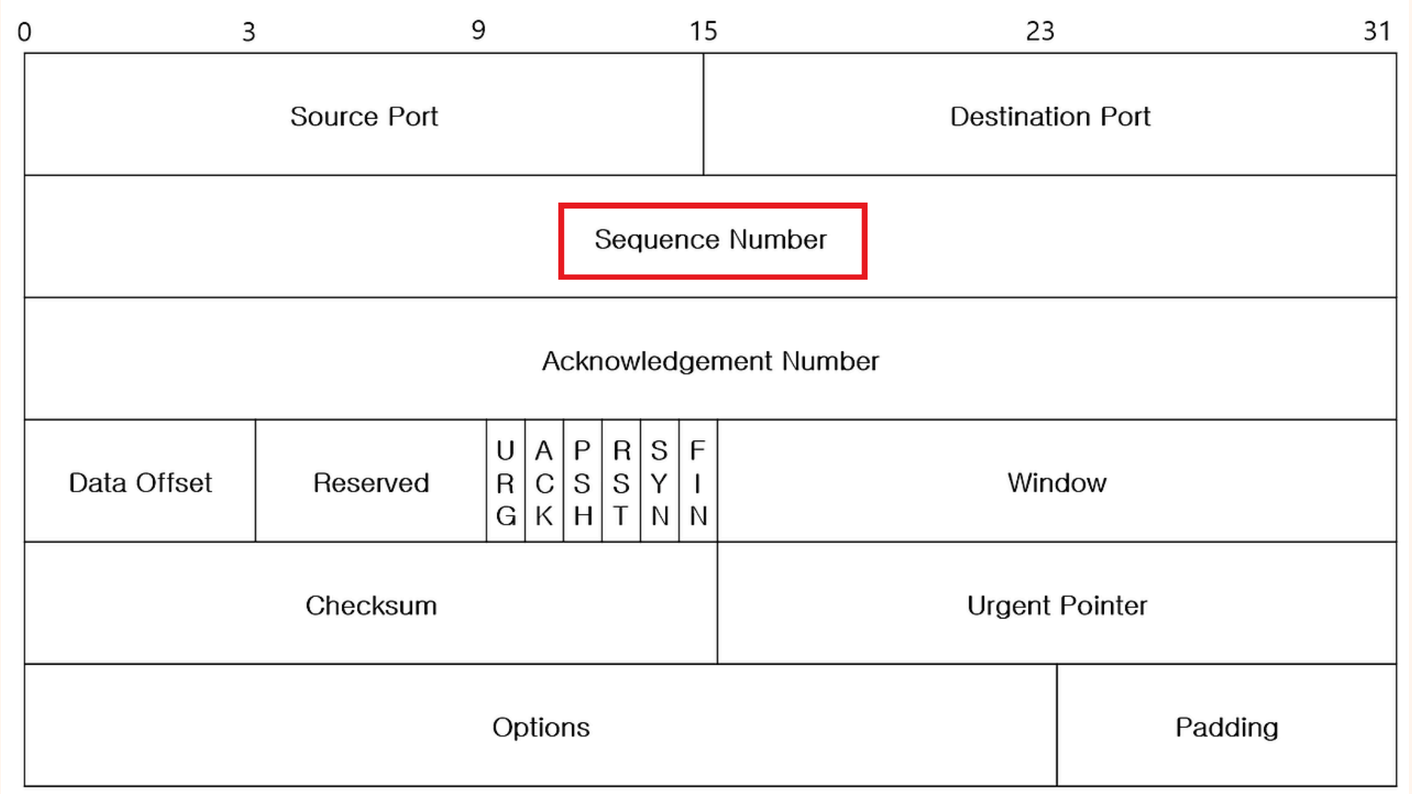


Segment / Datagram

TCP 헤더

Sequence Number

- 각 **segment 식별 ID** 아님
- 전체 데이터 흐름에서 **몇번째 바이트부터 시작하는지 알려주는** 주소

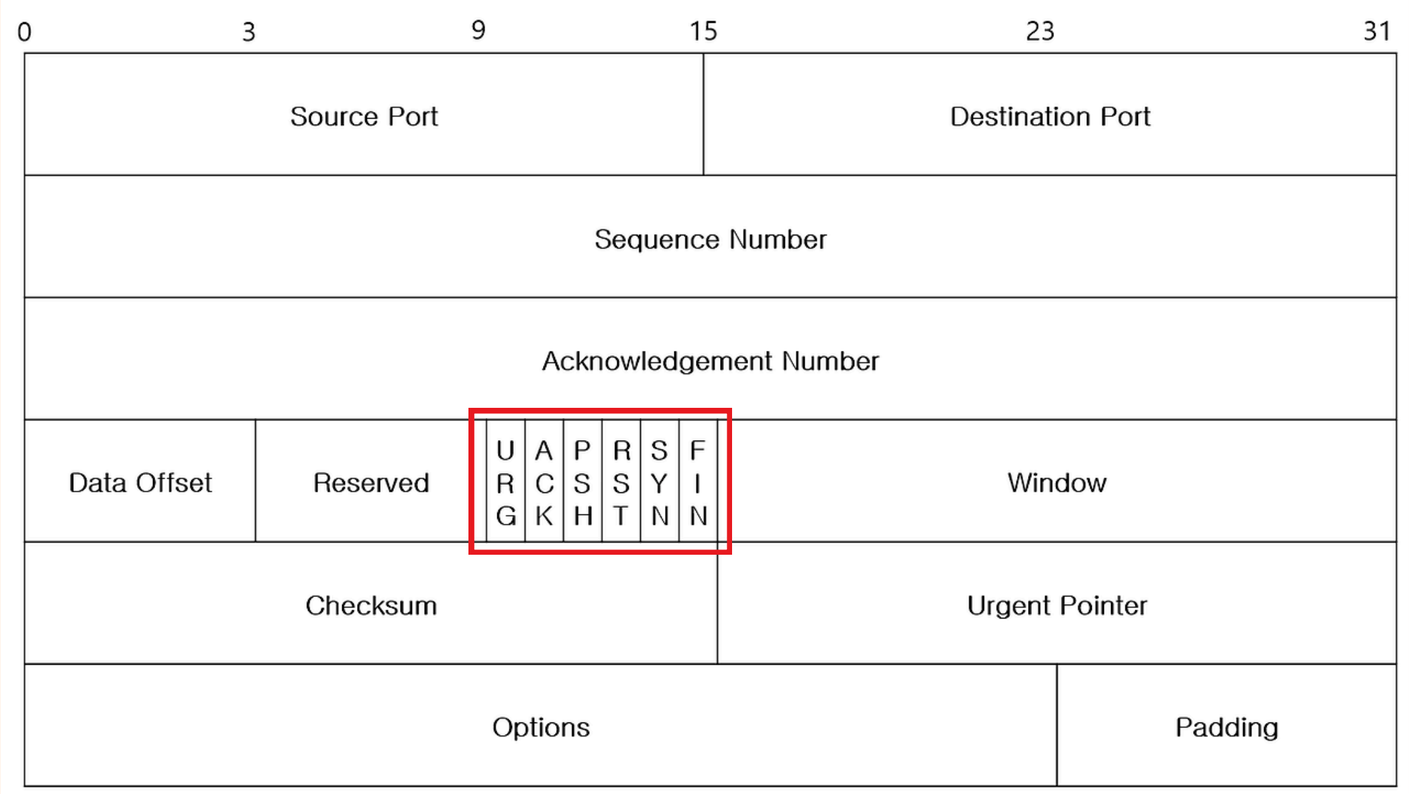


Segment / Datagram

TCP 헤더

Control Flags

- URG (Urgent)
- ACK (Acknowledgment)
- PSH (Push)
- RST (Reset)
- SYN (Synchronize)
- FIN (Finish)



Segment / Datagram

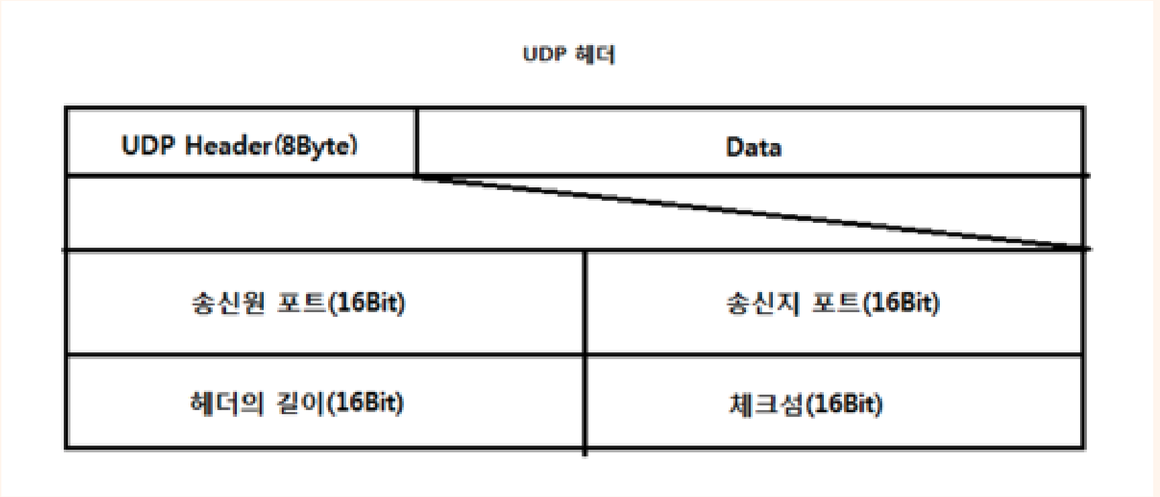
UDP 헤더

단순함과 속도 위해 설계

최소한의 정보만 있음

8바이트의 고정된 크기

2바이트의 **4개 필드로 구성**

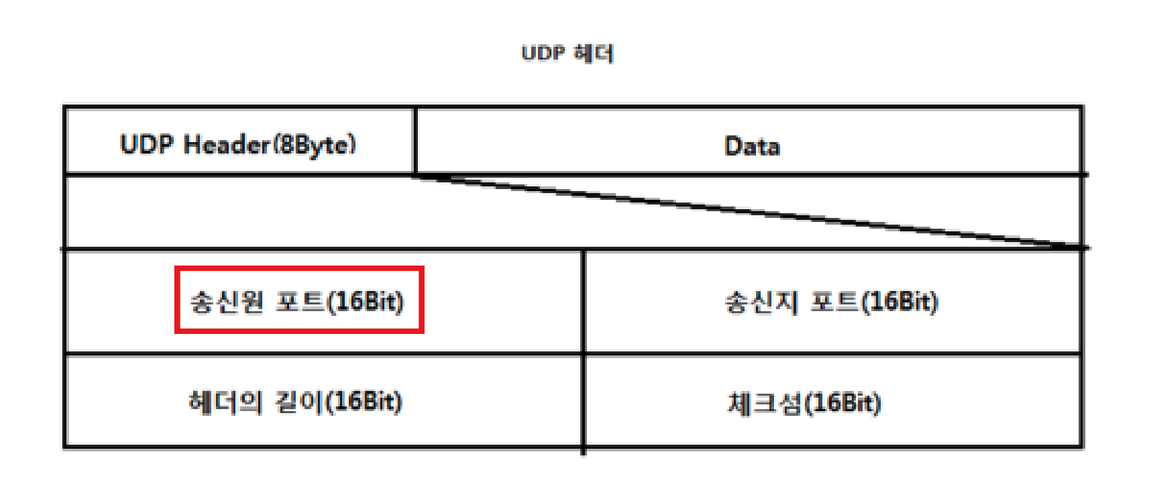


Segment / Datagram

UDP 헤더

Source Port

- **응답 받을 필요 없는 경우** 0으로 설정
- **TCP는** 양방향 통신 위해 **반드시 Source Port 필요**

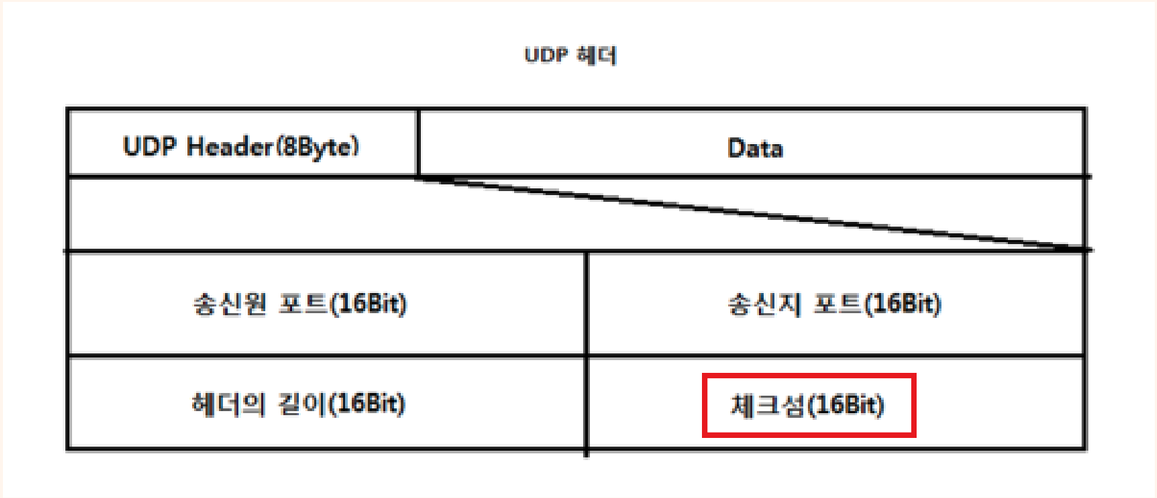


Segment / Datagram

UDP 헤더

Checksum

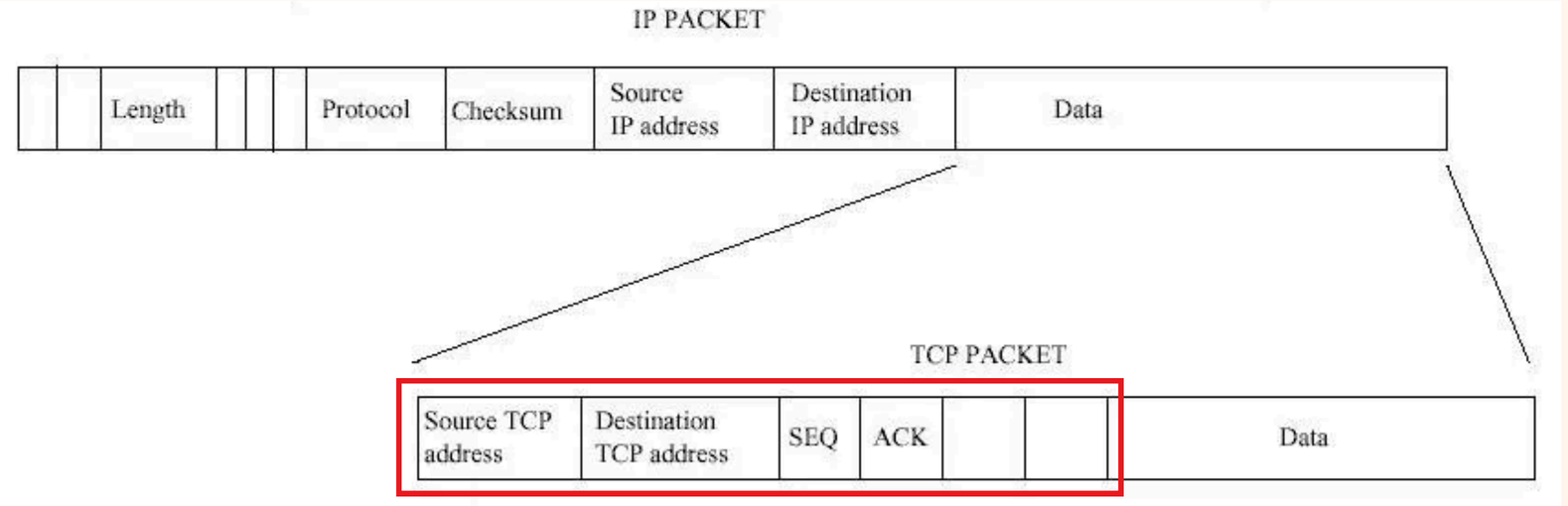
- **IPv4**에서는 **선택 사항**
- **IPv6**에서는 **필수**
- 사용하지 않을 경우 **모든 비트 0으로 채움**



목차

Packet

구조



3계층에서 사용하는 PDU

각각 **IP 헤더 + 데이터**로 나뉨

데이터는 **상위 계층에서 내려온 segment, datagram**로 통째로 들어감

[IP헤더 + TCP/UDP 헤더 + 데이터]

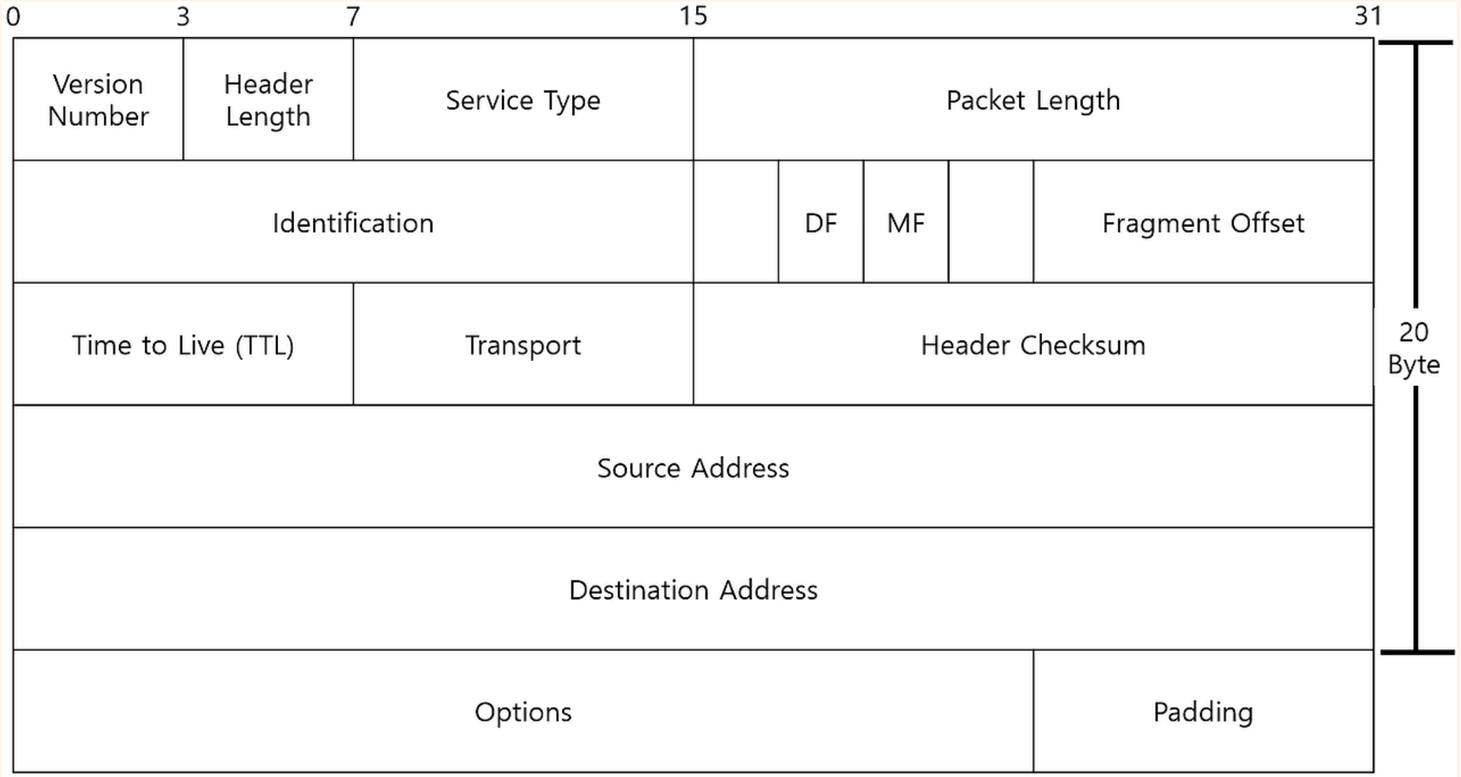
Packet

IP 헤더

IP 헤더 **기본적으로 20 바이트**

옵션 추가되어 **최대 60 바이트**

IPv4 기준



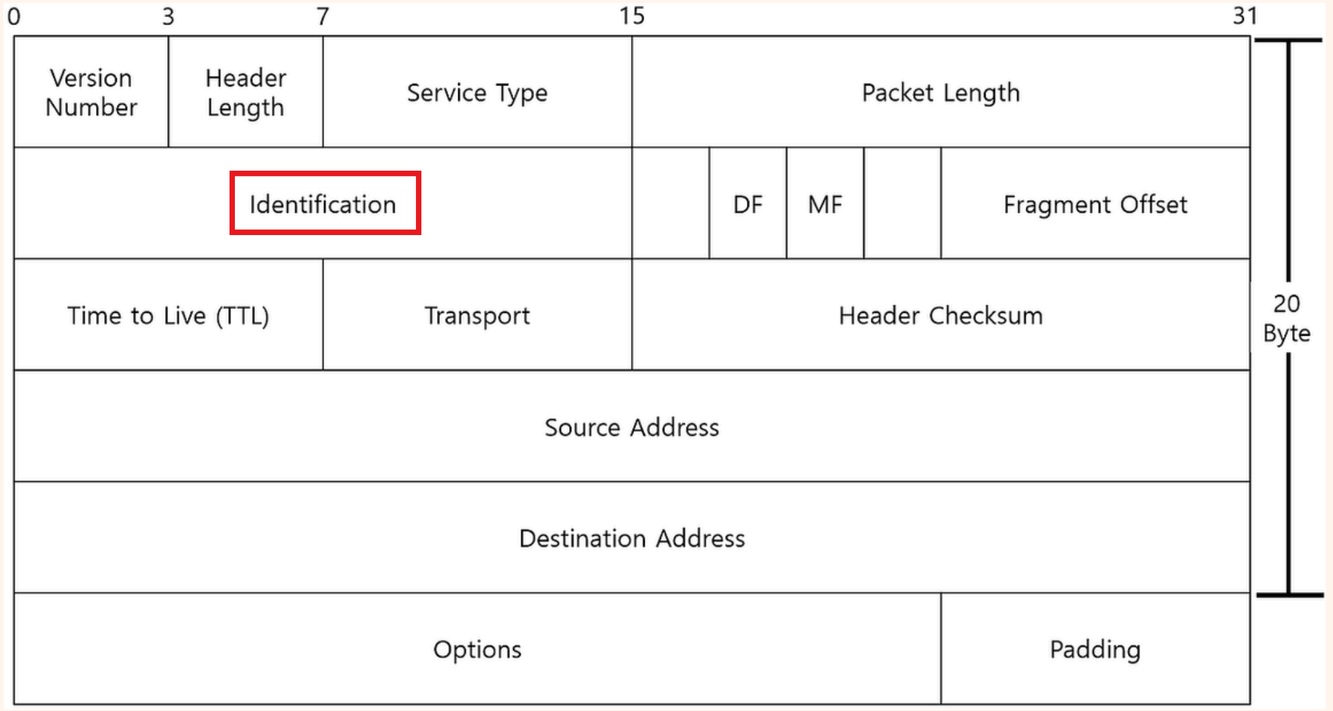
Packet

식별자 (Identification)

패킷 **여러 조각으로 나뉘었을 때** 사용

"이 조각들은 **원래 하나의 패킷**이었다"는 것을 알려주는 **고유한 ID 번호**

패킷 크기가 **네트워크의 MTU (Maximum Transmission Unit)보다 큰 경우** 패킷 나눔



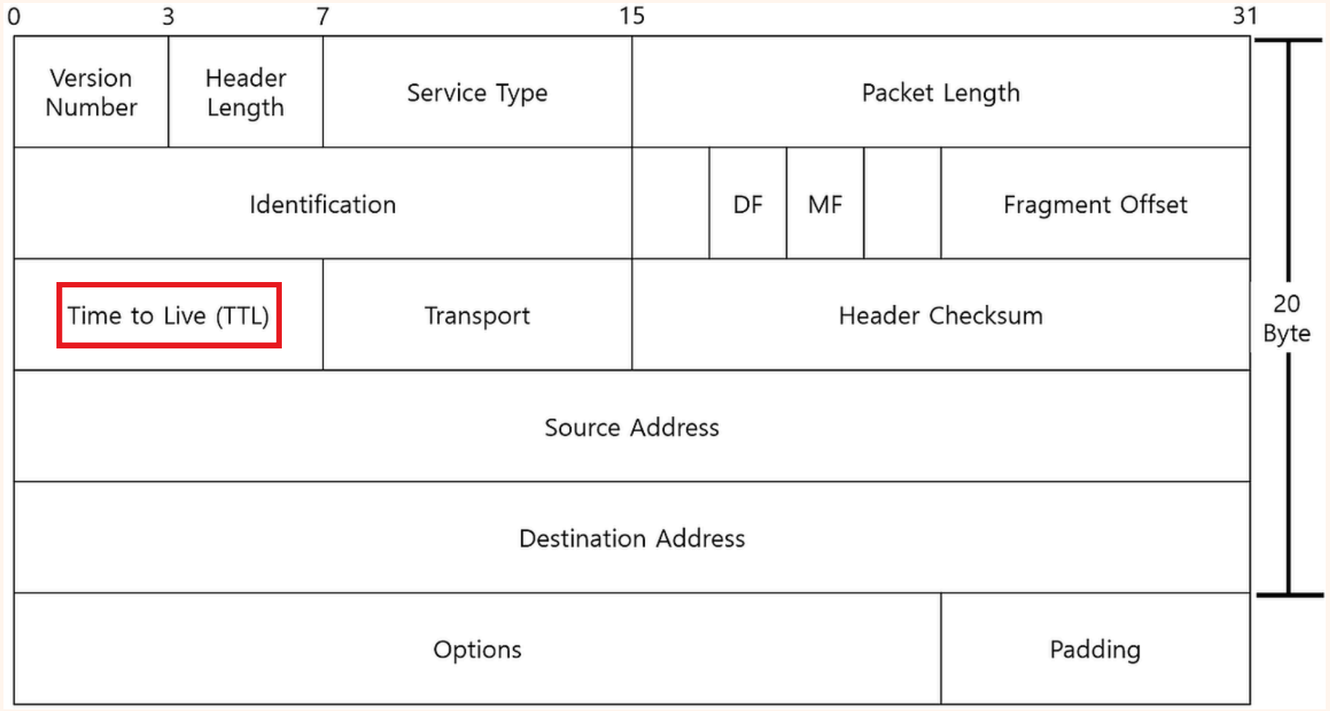
Packet

생존 시간 (TTL)

패킷 인터넷 상에서 **무한정 떠도는 것 방지**

패킷이 **라우터 하나 거칠 때마다 값 1씩 감소**

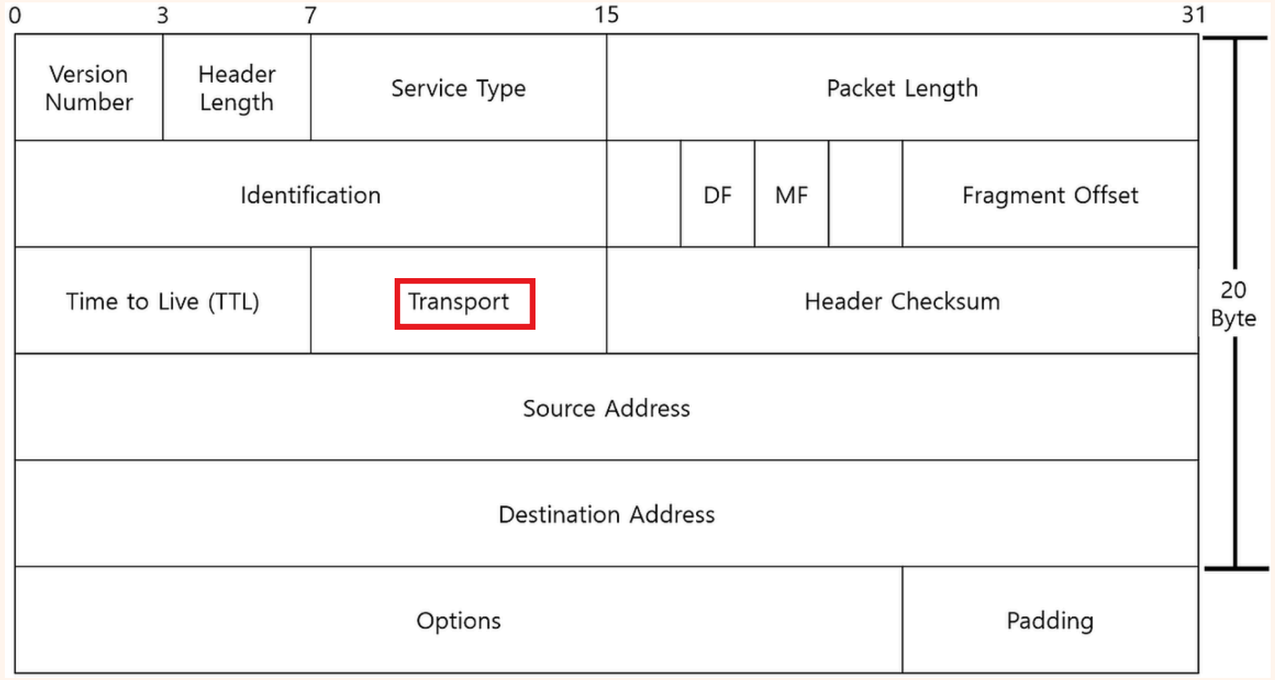
0이 되면 해당 패킷은 소멸



Packet

프로토콜 (Protocol)

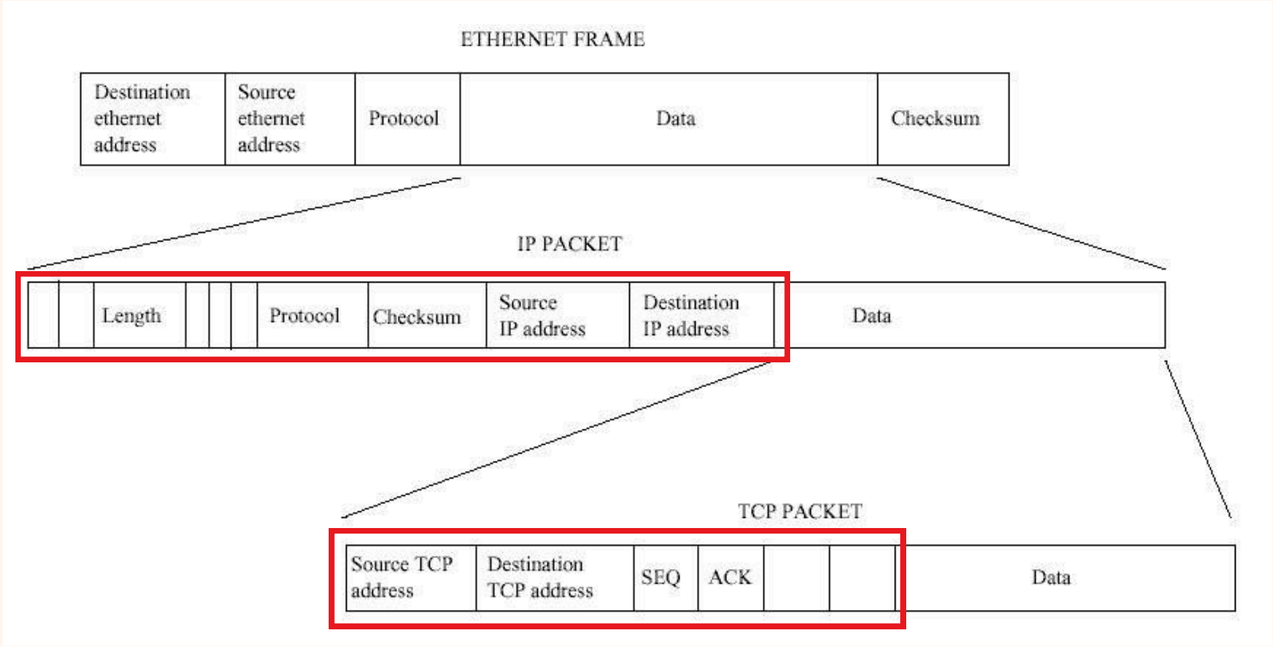
- 그림에서는 Transport
- **데이터에 어떤 프로토콜 데이터** 있는지 알려줌
- **값이 6이면** payload는 **TCP segment**
- **값이 17이면** payload는 **UDP datagram**
- 수신 측 이 값 보고 데이터 **4계층의 TCP 또는 UDP에게 전달**



목차

Frame

구조



2계층에서 사용하는 PDU

데이터의 **앞에 헤더, 뒷에 트레일러** 붙음

프레임은 각각 **Ethernet 헤더 + 데이터 + 트레일러**

데이터는 **상위 계층에서 내려온 packet**

[Ethernet헤더+IP헤더+TCP/UDP헤더+데이터+트레일러]

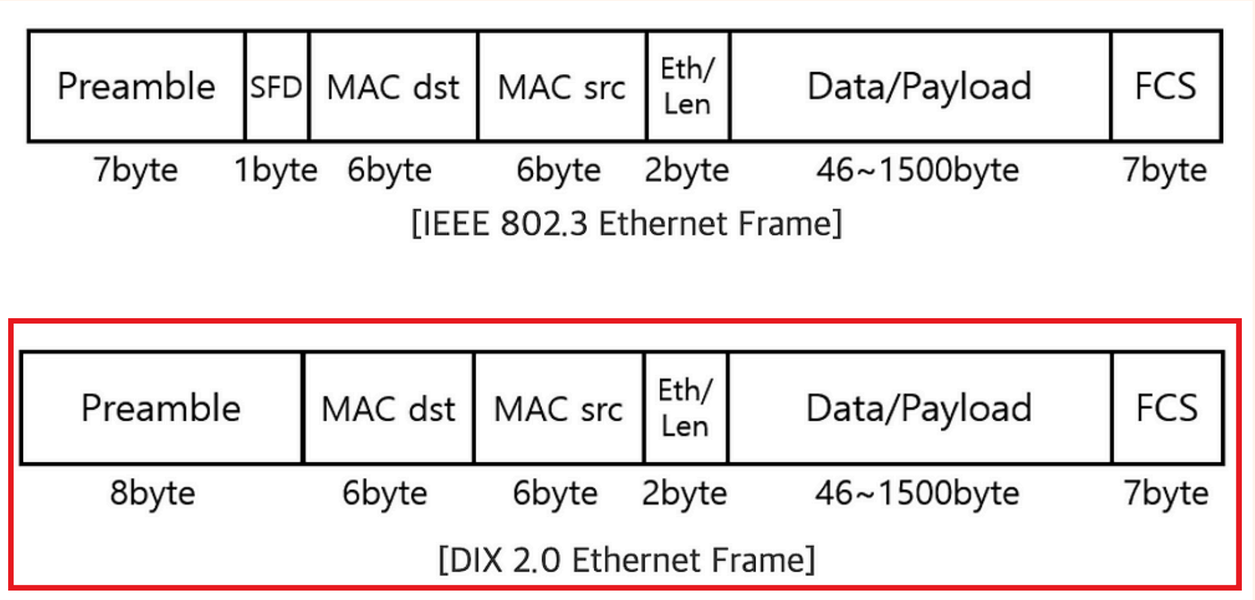
Frame

Ethernet 헤더

LAN 내에서 데이터 전달하기 위한 주소 포함

MAC 주소 이용해 정확한 주소 지정

Ethernet 헤더는 **총 14 바이트**



Frame

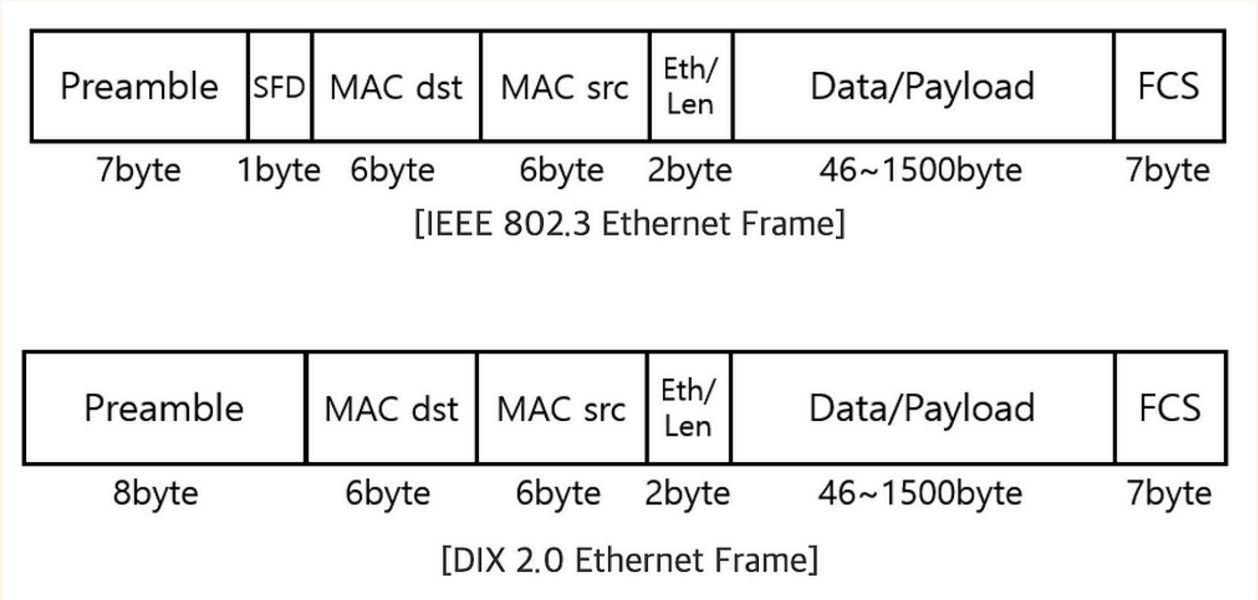
Ethernet 헤더

구성

- 목적지, 출발지 **MAC 주소**
- EtherType (**데이터 프로토콜**)

헤더는 X, Frame 앞에 붙음

- Preamle
- SFD

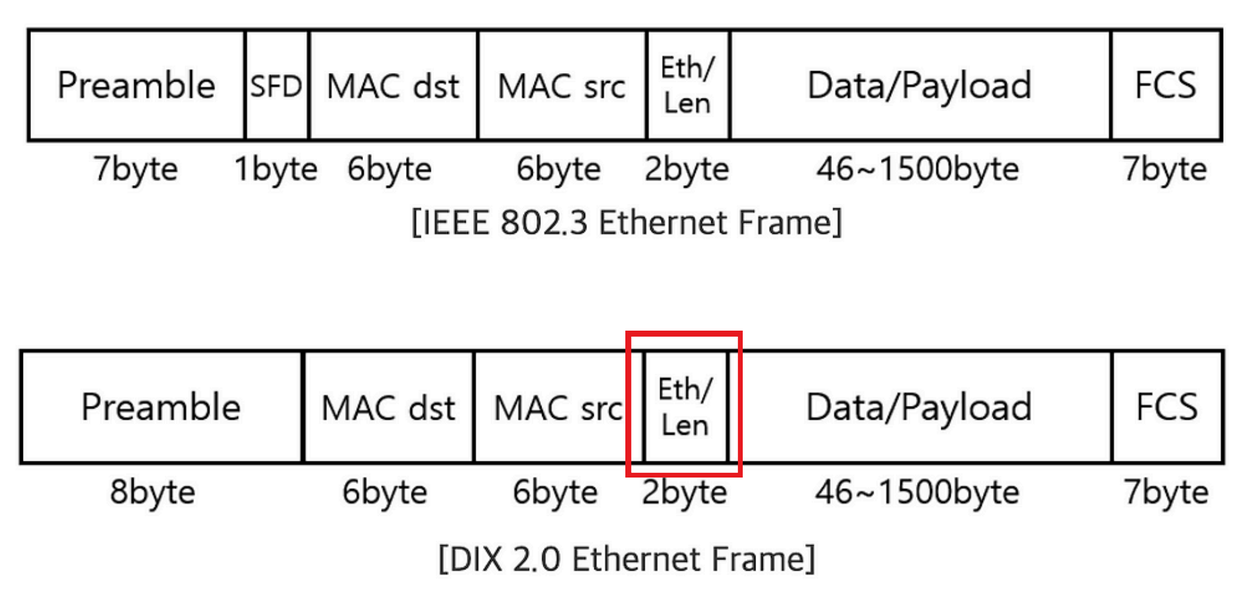


Frame

Ethernet 헤더

EtherType

- Payload = 데이터에 어떤 **3계층 프로토콜 패킷**인지 알려주는 것
- **프로토콜 예시**
  - IPv4 - 2048
  - IPv6 - 34525
  - ARP - 2054



Frame

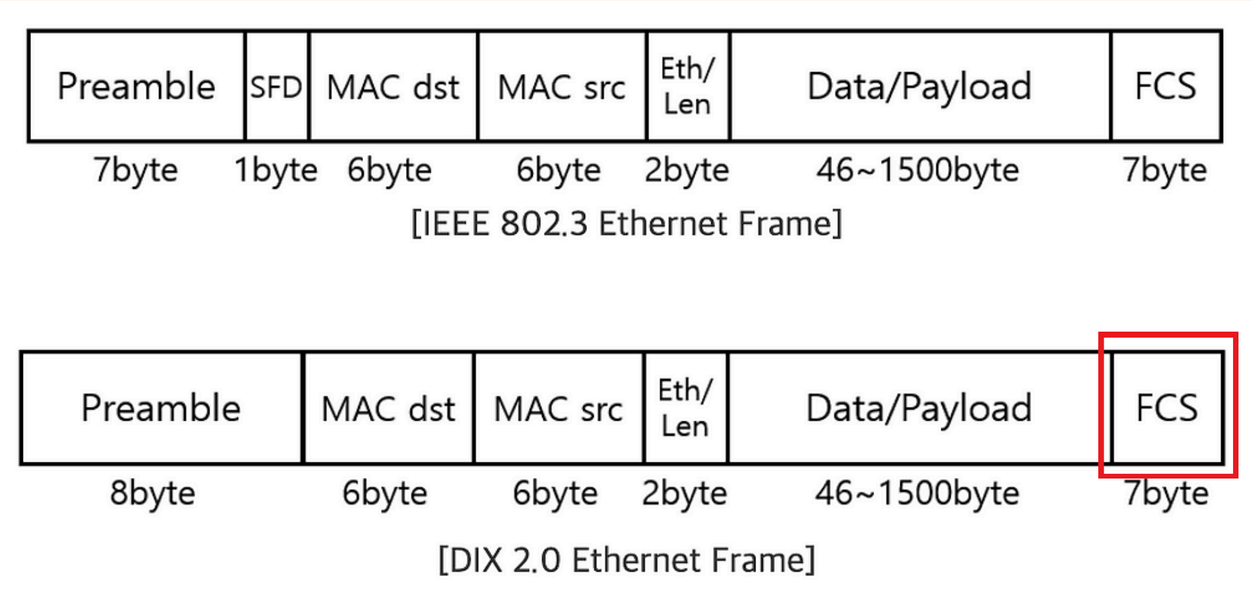
Ethernet 트레일러

데이터 뒤에 붙음

보통 **4바이트 크기** 가짐

오류 검출 역할

FCS라는 오류 검출 코드가 들어감



Frame

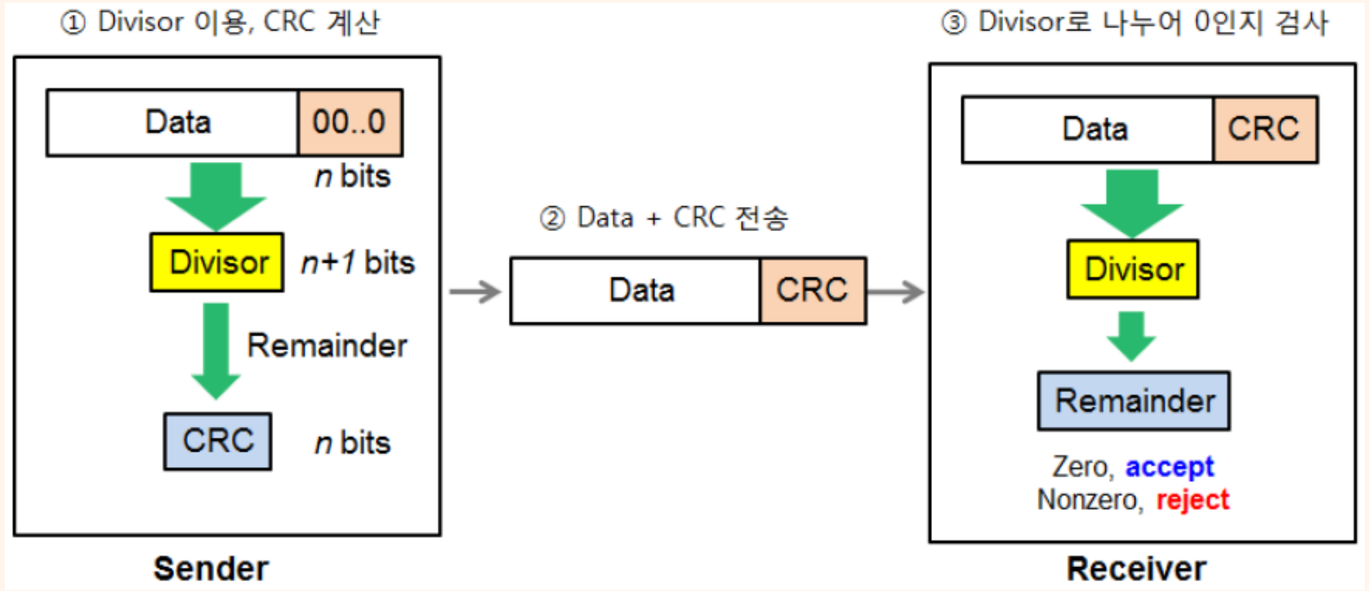
Ethernet 트레일러

송신측

- **헤더+데이터 → CRC 알고리즘 → FCS 만듬**
- FCS를 **트레일러에 붙여 전송**

수신측

- **다시 FCS 계산**
- 계산한 FCS와 트레일러의 **FCS 비교해 오류 검출**



Quiz

1번 문제

UDP 헤더와 마찬가지로 **TCP 헤더도**

Source Port에 0을 넣어도 된다.

1번 문제

TCP는 **양방향 통신 위해** 반드시 source port **필요함**

또 UDP와 달리 **TCP에서는 0번 port는 예약된 port**

2번 문제

패킷은 절대 **나눠지지 않은 PDU**이다.

2번 문제

패킷은 **네트워크의 MTU보다 크기 큰 경우** 나눠짐

이 경우 **Identification 필드로 하나의 패킷임을 확인**함

3번 문제

프레임 뒤에 붙는 **트레일러의 역할**은?

1. 악의적 공격 방어

2. 프레임 끝 알리기

3. 데이터 오류 검출

4. 옵션 제공

3번 문제

프레임 뒤에 붙는 **트레일러의 역할**은 **오류 검출**

CRC 알고리즘으로 **FCS 만들어 송수신측이 비교**

1. 악의적 공격 방어
2. 프레임 끝 알리기
3. **데이터 오류 검출**
4. 옵션 제공