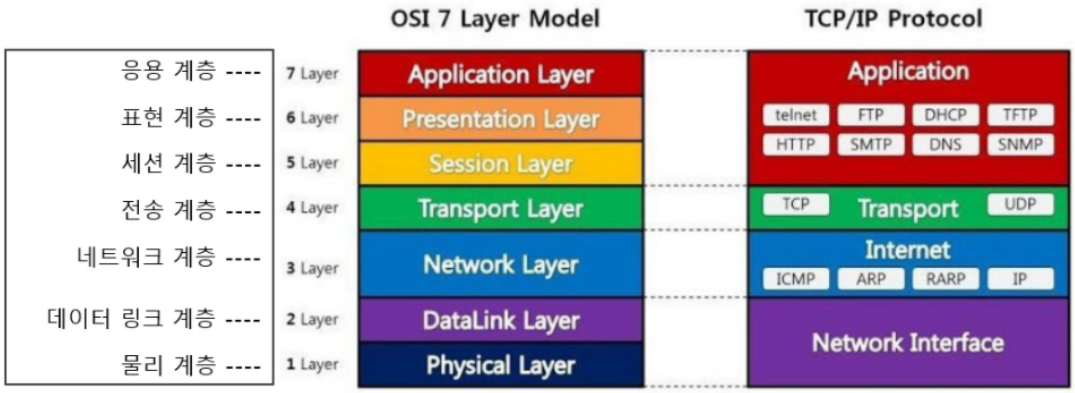
Computer_Network_02_HDLC(1)

> High-level Data Link Control
>
> 컴퓨터가 일대일 혹은 일대다로 연결된 환경에 데이터의 송수신 기능을 제공

11.1 Framing

- 데이터 링크 지원 통신 프로토콜(2가지)
  - Character-oriented protocol (Byte-oriented protocol) 문자 지향 프로토콜
  - Bit-oriented protocol 비트 지향 프로토콜

- 데이터 링크 계층



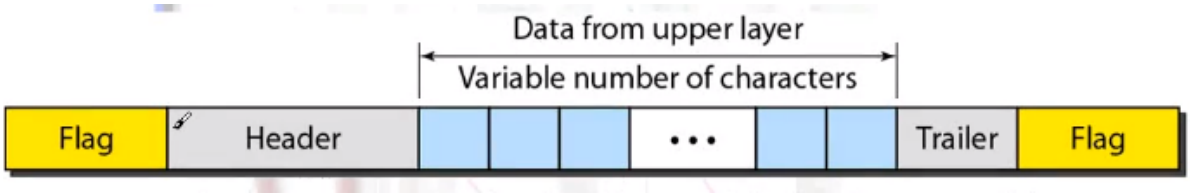
  - OSI 7계층 모델 중 2번째 계층
  - 잡음이 있는 인접 노드 간의 물리적인 회선을, 상위 네트워크 계층이 신뢰적으로 사용할 수 있도록 전송 에러 없는 통신 채널로 변환시키는 계층

- Message
  - 전송하는 데이터
  - Message는 하나의 큰 덩어리 => 전송 시 분할
  - 분할된 덩어리를 **Frame** 이라 한다.
  - Message = Frame들의 집합

- Frame
  - 데이터링크 계층에서의 데이터 단위
  - Frame도 하나의 덩어리 = Field의 집합
  - Frame을 byte들의 나열로 보는 것 => Byte-oriented protocol
  - Frame을 bit들의 나열로 보는 것 => Bit-oriented protocol

1. Character Oriented Protocol (Byte Oriented Protocol)

- Frame을 **byte의 나열**로 보는 프로토콜
- 1 character(byte) = 8 bit
- 8 bit 보다 적은 field는 없다 => 최소 단위 = 1 byte(= 8 bit)



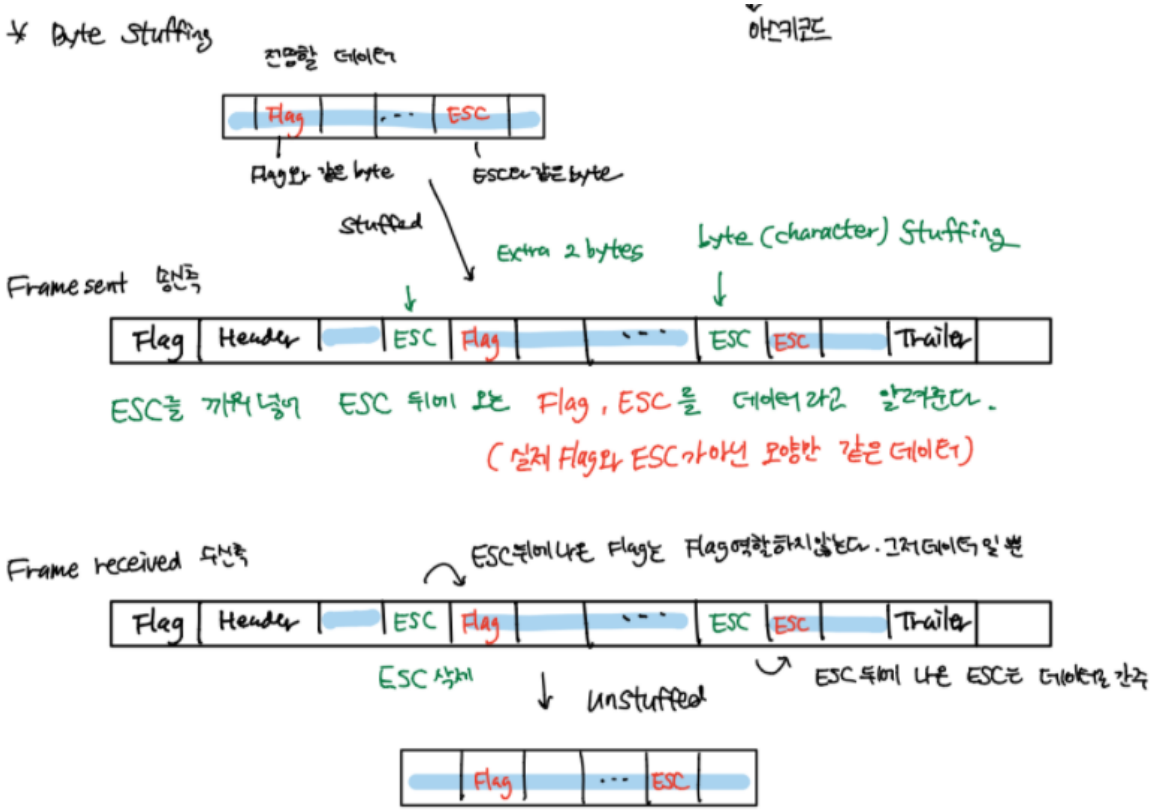
- Flag의 역할: Frame의 시작과 끝을 알려준다
- 앞에 오는 Flag를 opening flag, 뒤에 오는 Flag를 closing flag라 한다
- Flag의 크기: 1 ~ 2 byte
- Header: opening flag 뒤에 오는 것으로 데이터 외 추가적인 정보를 담고 있다 (수신측의 주소와 순서 등)
- Trailer: closing flag 앞에 오는 것으로 데이터 외 추가적인 정보를 담고 있다 (에러 정보)
- 실제 데이터(Payload) = Flag, Header, Trailer를 뺀 나머지

문제점

- 실제 데이터에 Flag와 같은 바이트가 들어가는 경우 (같은 비트 배열)
- 이 경우 Flag 위치를 잘못 읽을 수 있다
- 이를 Transparency (투명성 문제)라 한다

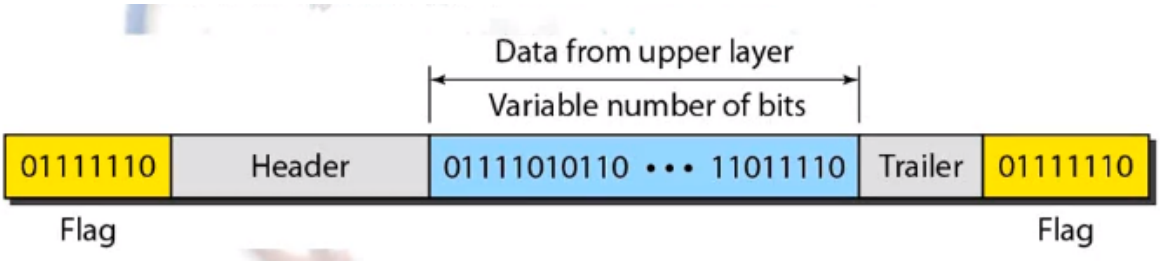
해결방안

- 특정 byte를 끼워 넣는다 (Byte Stuffing or Character Stuffing)
- 특정 byte(1 byte)를 **ESC(escape character)**라 한다.



- 송신측에서 전송할 데이터에 Flag나 ESC와 같은 byte가 존재할 경우 **해당 byte앞에 ESC를 삽입**
- 수신측에서 ESC다음 오는 byte는 데이터로 인식

2. Bit Oriented Protocol



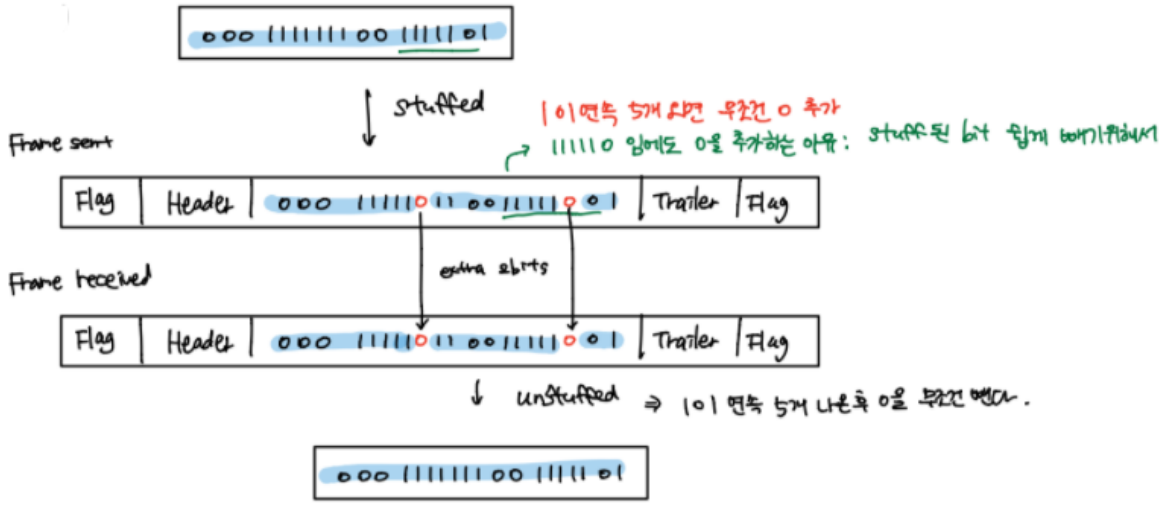
- Frame을 **Bit의 나열**로 본다
- 최소 단위는 1 bit
- Flag는 모두 `01111110`
- Header에는 3 bit로 된 Frame Number존재 000 ~ 111까지 0번 ~ 7번 까지 존재
- Frame Number로 몇 번째 Frame인지 알려준다
- Flag, Header, Trailer를 제외한 부분이 실제 데이터
- Flag, Header, Trailer의 역할은 Byte Oriented Protocol에서의 역할과 같다

문제점

- Byte Oriented Protocol과 마찬가지로 Transparency 문제가 있다
- 데이터 중에 `01111110`과 같은 비트 배열이 있을 경우 발생

해결방안

- Bit Stuffing: bit 1개 끼워 넣기



- 송신측은 **1이 연속 5개** 오면 **무조건 0을 뒤에 추가**한다
- 수신측은 **1이 연속 5개** 오고 **뒤에 나타난 0을 삭제**한다
- `0111110`는 Flag `01111110`과 다르지만 0을 추가하는 이유는 stuff된 bit를 쉽게 빼기 위해서이다 (로직 간단하게)

11.2 Flow and Error Control

Flow Control 흐름 제어

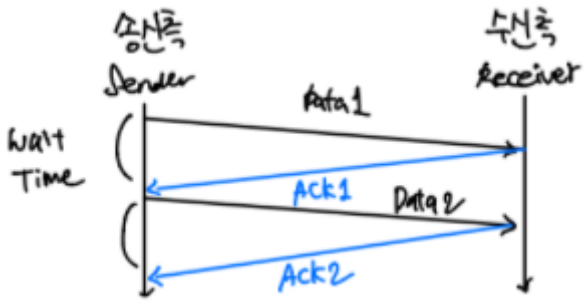
- 데이터 양 조정
- 수신측에서 **ACK**로 얼만큼 보내야 할지 조정
- **제어권은 수신측에 있다** => 송신측 성능이 좋아도 수신측 성능이 떨어지면 빠르게 받을 수 없음
- 수신측에서 잘 받았으면 **ACK**를 보내고 그렇지 않으면 **NAK(NACK)**를 보낸다
- Flow Control 종류 (3가지)
  - XON/XOFF: Start or Stop Transmission
  - Stop and Wait: 한번에 하나의 Frame 전송
  - Sliding Window: 한번에 여러개의 Frame 전송

1. XON/XOFF

- XON: 전송 가능 / XOFF: 전송 불가
- Serial Terminal에서 사용
- Serial Terminal => 프린터, 디스플레이 장치와 같이 1 byte씩 보내는 컴퓨터
- 예시)
  - PC에서 프린터로 데이터 전송 => 프린터 메모리(버퍼)에 쌓임
  - 프린터 메모리가 일정 수준 이상 채워지면 PC에 XOFF 1 byte 전송
  - XOFF 받은 PC는 데이터 전송 중단, 프린터는 메모리에서 데이터 꺼내 사용하여 계속 작동

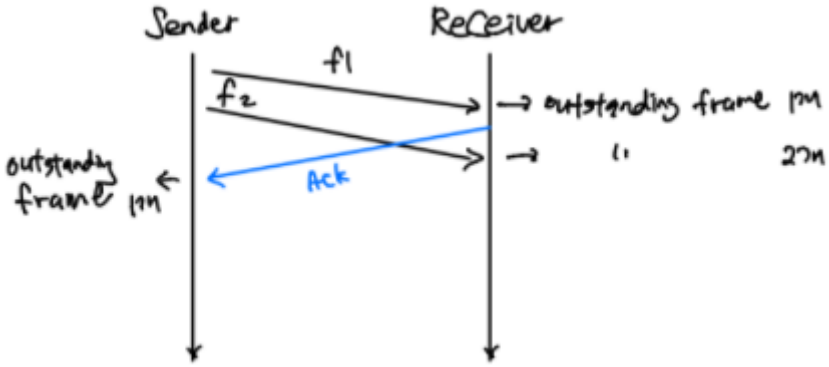
- 프린터 메모리에 데이터가 일정 수준 이하로 떨어지면 XON을 보내 다시 전송하도록 한다

2. Stop and Wait



- 한번에 하나의 Frame을 전송
- 하나의 Frame을 전송 후 바로 다음 Frame을 전송할 수 없다 (Stop)
- 수신측에서 보낸 ACK를 받은 후 다음 Frame 전송 가능, 그 때까지 대기 (Wait)
- 장점: 간단함
- 단점: 비효율적 (기다려야함)

3. Sliding Window



- Window Flow Control에서 몇 개의 Frame을 한번에 전송할지 결정
- Window Size: 한번에 보낼 수 있는 최대 Frame의 수
- 현재 ACK 받은 Frame 몇개인지 세어서 Control (Window Size - ACK 수 만큼 더 보낼 수 있다)
- Outstanding Frame: ACK를 받지 못한 Frame
- Window Size와 최대 Outstading Frame은 같은 크기를 갖는다

Error Control

- Error Detection (에러 검출) + Error Correction (에러 복구)
- Error Control 3가지
  - 에러 무시 => 복구하지 않음 (ex: 전화 잡음)
  - Forward Error Correction (FEC): 수신측에서 복구 (해밍코드 사용)

    - Automatic Repeat Request (ARQ): 에러가 발생한 Frame(NAK 받을 경우) 재전송 요청
- Error
    - Lost Frame => 프레임 없어짐, 도착 X
    - Damaged Frame => 프레임은 도착했으나 인식불가 (몇몇 bit들이 깨짐)
- Duplicated Frame: 중복 Frame, 받은 Frame을 또 보냄