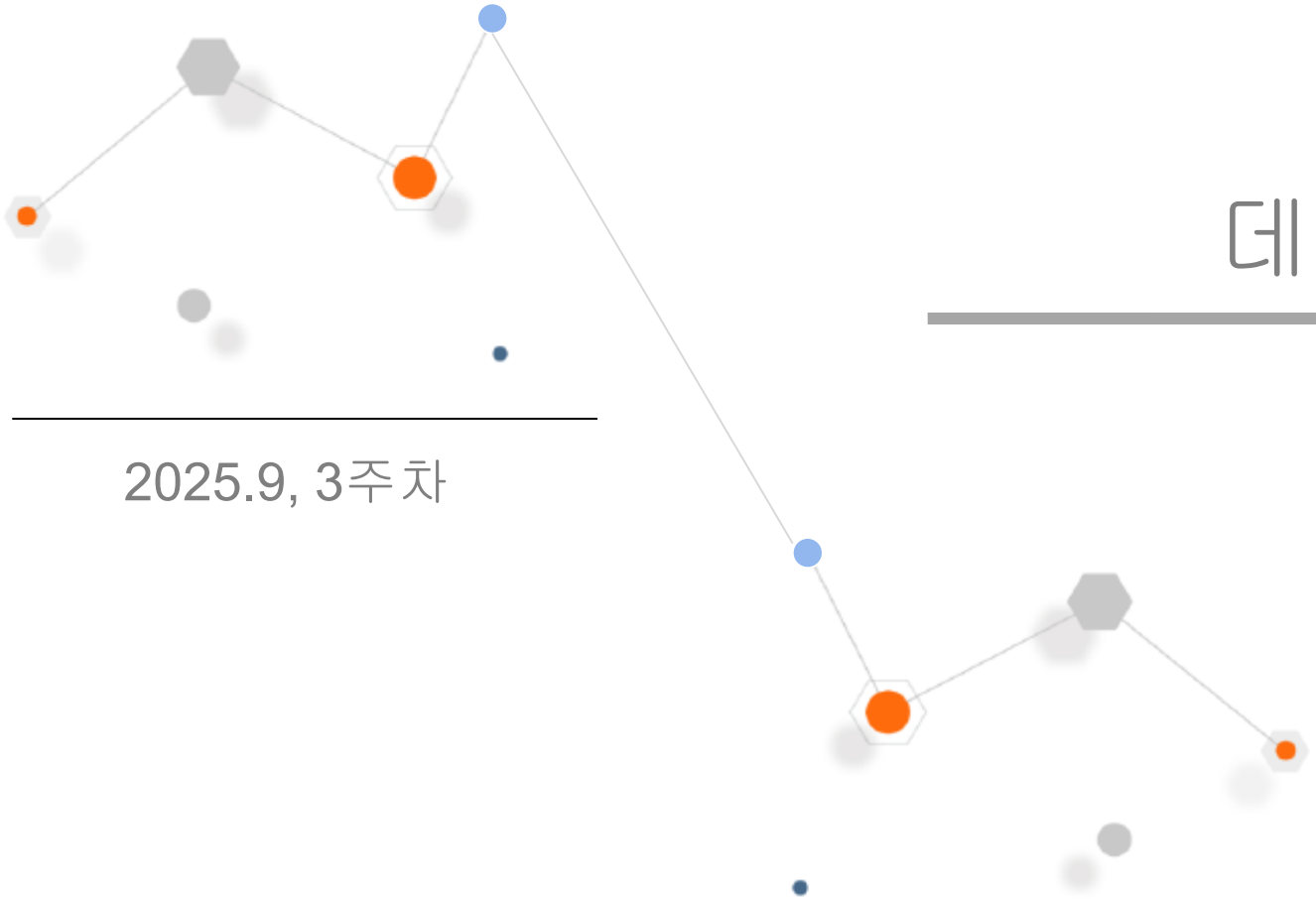


데이터 통신

컴퓨터공학과 이병문

2025.9, 3주차

강의일정

bmlee made

01 강의/일정소개, 평가소개

02 개요 및 물리계층

03 데이터링크 계층 1

04 데이터링크 계층 2

05 무선통신

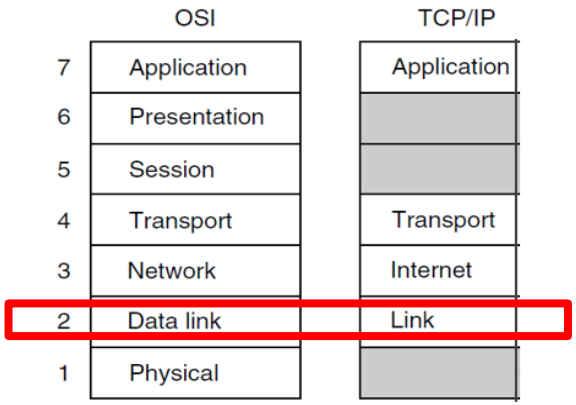
06 네트워크 계층 1

07 네트워크 계층 2

08 중간고사

데이터링크 계층

- 데이터링크계층 개요
- 오류제어
- 흐름제어
- 프레임 생성/관리
- 접근제어 및 링크제어관리



(9.25(목) Quiz 1 시험)- 4,5반

(10.3(목) 개천절, 웹엑스보강) – **5반**

(10.6(월) 추석, 웹엑스보강) – **4반**

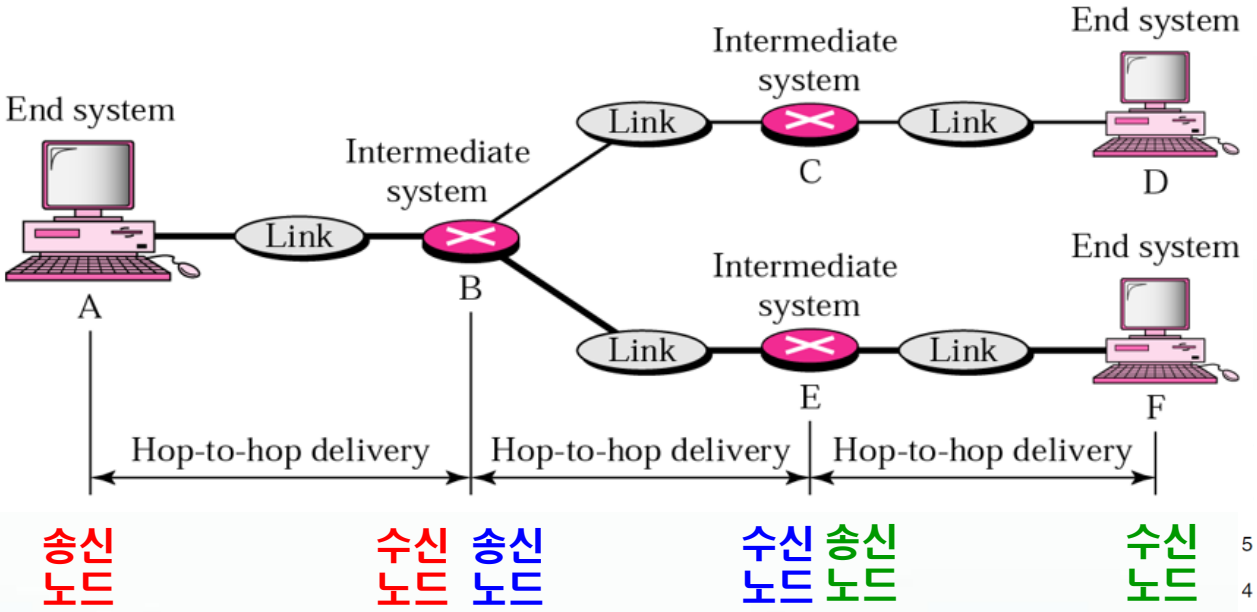
(10.9(목) 한글날, 웹엑스보강) – **4, 5반**

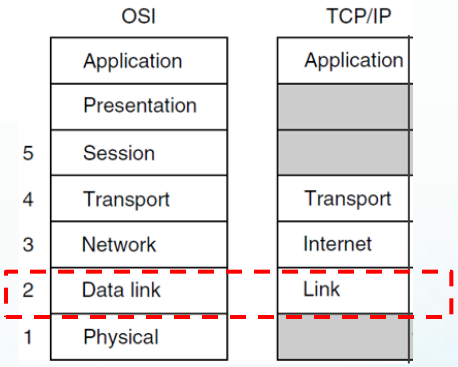
데이터링크 계층

■ 데이터링크 계층 개요

☑ 데이터링크계층

- "노드와 노드 간의 오류가 없는 데이터전송" 을 하기 위한 목표로 전송 규격을 정의
- 상위계층(네트워크계층)에서 오류없는 물리계층처럼 보이도록 역할을 함

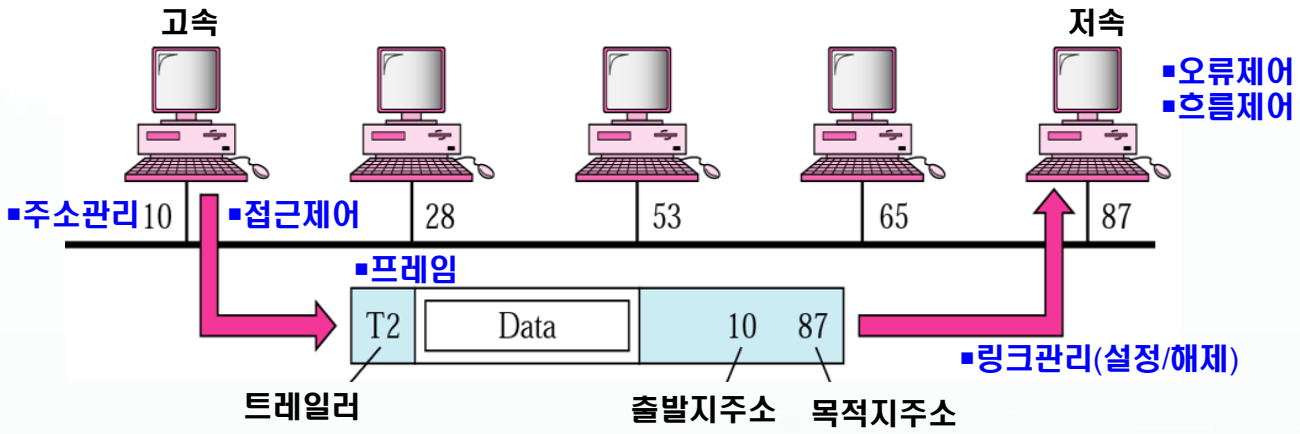




데이터링크 계층

☑ 데이터링크계층의 핵심기능

- 주소관리 …… 물리주소체계를 정의하고, 송신노드와 수신노드의 물리주소를 지정
- 접근제어/링크제어 …… 물리계층의 특성과 구조에 맞게 접근을 제어하고 링크를 관리하는 기능
- 프레임생성/관리 …… 데이터 전송을 위해, 데이터를 캡슐화한 프레임형식으로의 생성/관리기능
- 오류제어 …… 수신노드에서 오류를 탐지하고 복구하는 기능
- 흐름제어 …… 수신처리 속도를 고려하여 수신기가 송신기의 전송속도를 제어하는 기능



"프레임"(Frame) 이란? 데이터링크계층에서 전송되는 전송데이터단위를 말함
즉, 물리계층에서는 "신호 " 를 전송하지만, 데이터계층에서는 "프레임 " 을 전송한다.

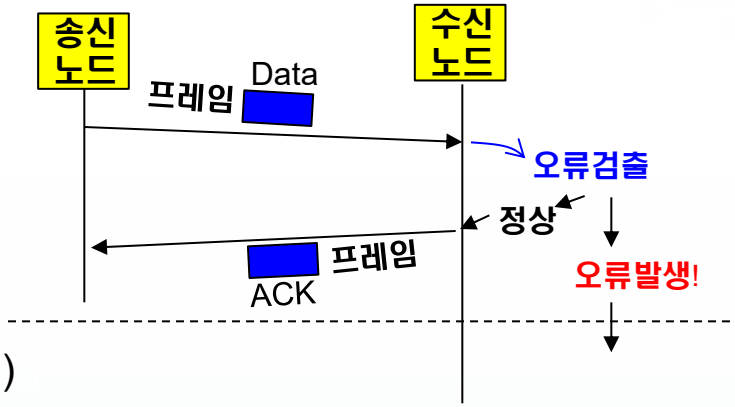
데이터링크 계층

■ 오류제어

- 오류검출
- 오류복구(오류정정)
  - 전방향오류정정(FEC)
  - 역방향오류정정(REC)

☑ 오류의 원인

- 물리계층의 전송장애로 오류발생
  (신호감쇄, 왜곡, 잡음, 신호간섭 등)
- 송신노드 또는 수신노드의 장애(고장)
- 전송매체의 절단, 파괴등의 장애
- 저속 전송매체(예, UTP1)로 데이터를 빠른속도로 무리하게 송신하는 경우
  (즉, 전송매체의 성능에 적합하지 않게 사용하는 경우)



☑ 오류유형

- single bit error
- burst errors

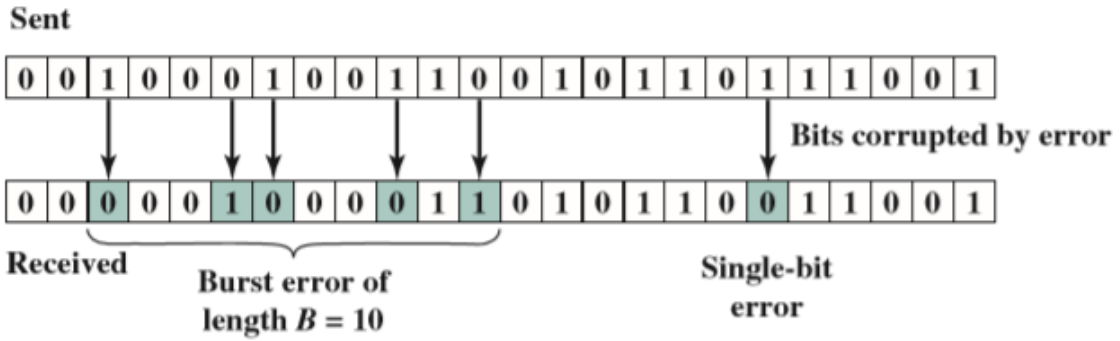


Figure 6.1 Burst and Single-Bit Errors

데이터링크 계층

■ 오류제어

- 오류검출
- 오류복구(오류정정)
  - 전방향오류정정(FEC)
  - 역방향오류정정(REC)

☑ 오류검출

- Parity비트 검출법
- 2차원 parity비트 검출법
- Checksums 검출법
- Internet Checksums 검출법
- CRC (Cyclic Redundancy Checks)

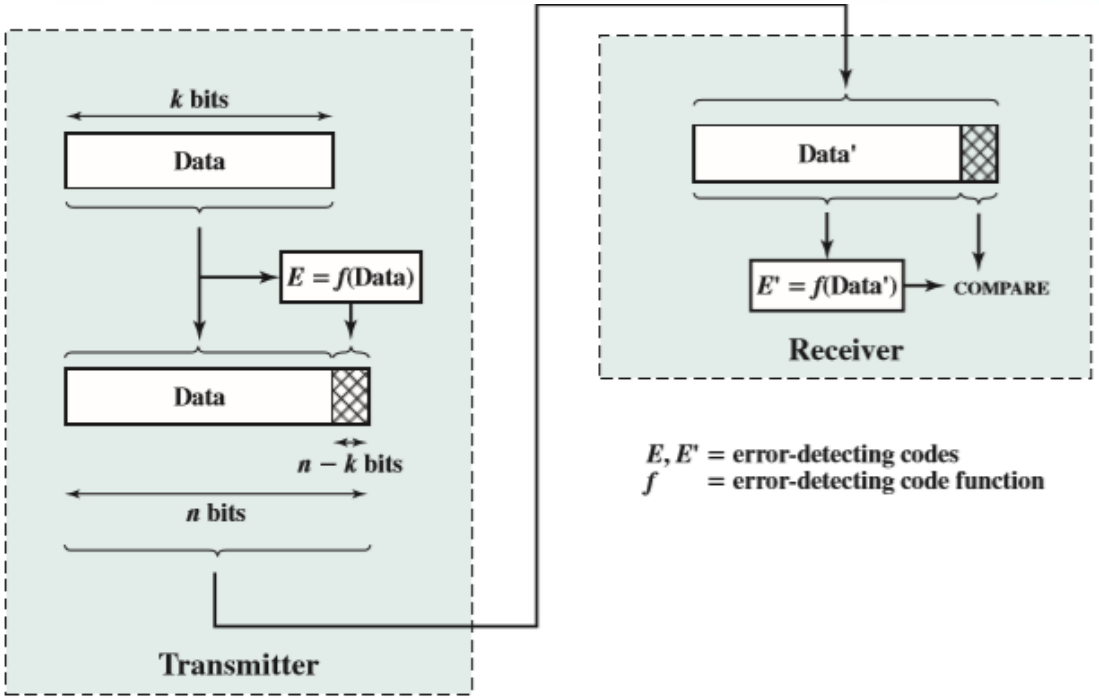


Figure 6.2 Error-Detection Process

☑ 오류복구

- 전방향 오류정정

  Hamming 코드기법,

  binary convolutional codes,

  Reed-Solomon codes ,…

- 역방향 오류정정

  NAK ……. 재전송 요청후 다시 수신해서 정정

  Timeout ……. 응답을 못 받으면 다시 전송해 정정

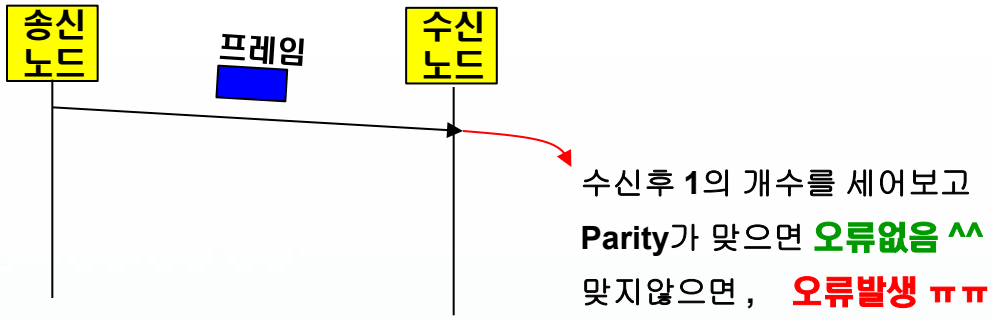
데이터링크 계층

■ 오류검출

- Parity bit
- Two dimensional parity bit
- Checksums
- Internet Checksums
- CRCs

☑ **Parity bit 검출법**

- 데이터에서 1의 개수를 홀수 (또는 짝수)로 맞추어 전송!, 수신 후에 개수를 확인하는 기법
- even parity bit
- odd parity bit



- 예1) 데이터 "1010011"
  홀수(odd) Parity 경우
  10100111
  Parity bit

- 예2) 데이터 "1100010" , 짝수 Parity 라면, Parity bit를 포함한 전송할 데이터는 ? ( )

- 예3) ASCII문자 'B'을 전송 할 경우, Odd Parity bit 는 ? ( )

- 문제점: 2 비트이상 오류(burst error) 가 나면 검출 성능이 떨어짐

데이터링크 계층

■ 오류검출

☑ **2차원 Parity bit 검출법**

- 데이터를 (블록단위로) 분할 후, 행과 열에 각각 패리티 비트들을 추가하여 오류를 검출하는 방식
- 예시) 데이터 “1001001 1000100 1110101 0001010” 을 전송할 때

1 0 0 1 0 0 1	1
1 0 0 0 1 0 0	0
1 1 1 0 1 0 1	1
0 0 0 1 0 1 0	0
1 1 1 0 0 1 0	0

송신노드

수신노드

1 0 0 1 0 0 1	1
1 0 0 0 1 0 0	0
1 1 1 0 1 0 1	1
0 0 0 1 0 1 0	0
1 1 1 0 0 1 0	0

10010011 10001000 11101011 00010100 11100100

▲오류없음

1 0 0 1 0 0 1	1
1 0 0 0 1 0 0	0
1 0 1 0 1 0 1	1
0 0 0 1 0 1 0	0
1 1 1 0 0 1 0	0

▲1비트 오류 (검출, 정정)

1 0 0 1 0 0 1	1
1 0 0 0 1 0 0	0
1 0 1 0 1 1 1	1
0 0 0 1 0 1 0	0
1 1 1 0 0 1 0	0

▲2비트 오류 (검출)

1 1 0 1 0 1 1	1
1 0 0 0 1 0 0	0
1 0 1 0 1 1 1	1
0 0 0 1 0 1 0	0
1 1 1 0 0 1 0	0

▲4비트 오류 (검출불가능)

데이터링크 계층

■ 오류검출

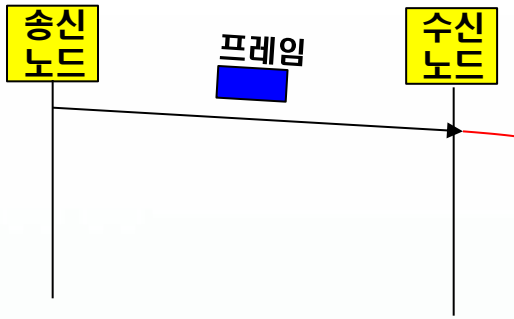
☑ **Checksum (1 byte) 검출법**

- 데이터를 1 Byte (블록)단위로 XOR를 하여 Checksum 을 생성한후 전송, 수신 후에 받은 데이터들을 동일하게 XOR하여 만든 것과 수신한 Checksum Byte를 비교/확인하는 방법

- 예1) 데이터가 아래 같다면 (즉, 4 Byte라면)

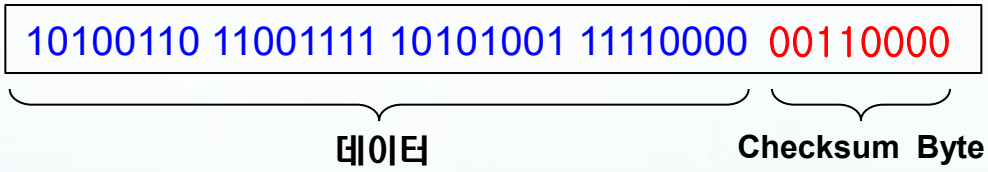
"10100110 11001111 10101001 11110000"

```
  10100110
  11001111
  10101001
  11110000   ( XOR
-------------
  00110000
```



수신 후 데이터 부분을 **XOR**하여 **Byte**를 **Checksum**을 생성하고, **Checksum** 필드값과 비교하여 동일하면 **오류없음** 틀리면, **오류발생(검출)**

(프레임 내)의 전송데이터



■ 오류검출

☑ **Internet Checksum 검출법 (2 byte 단위)**

- ones-complement addition 방식으로 계산
- IP, TCP, UDP 에서 사용하는 오류검출방식

ones-complement addition

0011	1101
+1100	+1011
1111	11000
	+ 1
	1001

- 예, 10 octets

00 01 F2 03 F4 F5 F6 F7 00 00

송신 컴퓨터

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Ones complement of the result	220D

1의보수

00 01 F2 03 F4 F5 F6 F7 22 0D

수신 컴퓨터

Partial sum	0001 F203 F204
Partial sum	F204 F4F5 1E6F9
Carry	E6F9 1 E6FA
Partial sum	E6FA F6F7 1DDF1
Carry	DDF1 1 DDF2
Partial sum	DDF2 220D FFFF

오류없음 !!

데이터링크 계층

■ 오류검출

☑ **CRC (Cyclic Redundancy Check) 검출법**

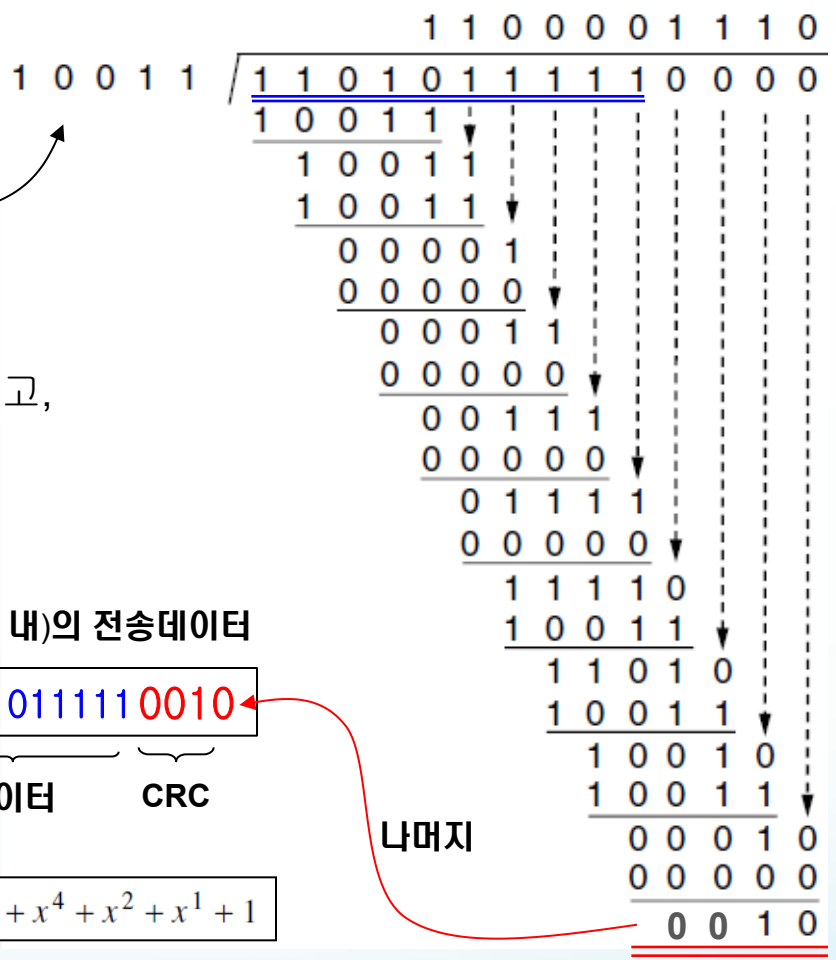
- 다항식의 원리를 이용한 방법
- 예) 4차 다항식 ($X^4 + X^3 + X^2 + X^1 + X^0$)

계수 추출

CRC 다항식: $X^4 + X + 1$

Generator : 10011

실제데이터: 1101011111



- 수신 후 데이터 부분을 나누어 CRC를 생성하고, CRC 필드값과 비교하여 동일하면 오류없음
  틀리면, 오류발생

- 검출능력이 좋아서 대부분의 LAN(Ethernet)에서 사용되는 방법
- CRC-16, CRC-32, CRC-12

(프레임 내)의 전송데이터

1101011111 0010

데이터 CRC

나머지

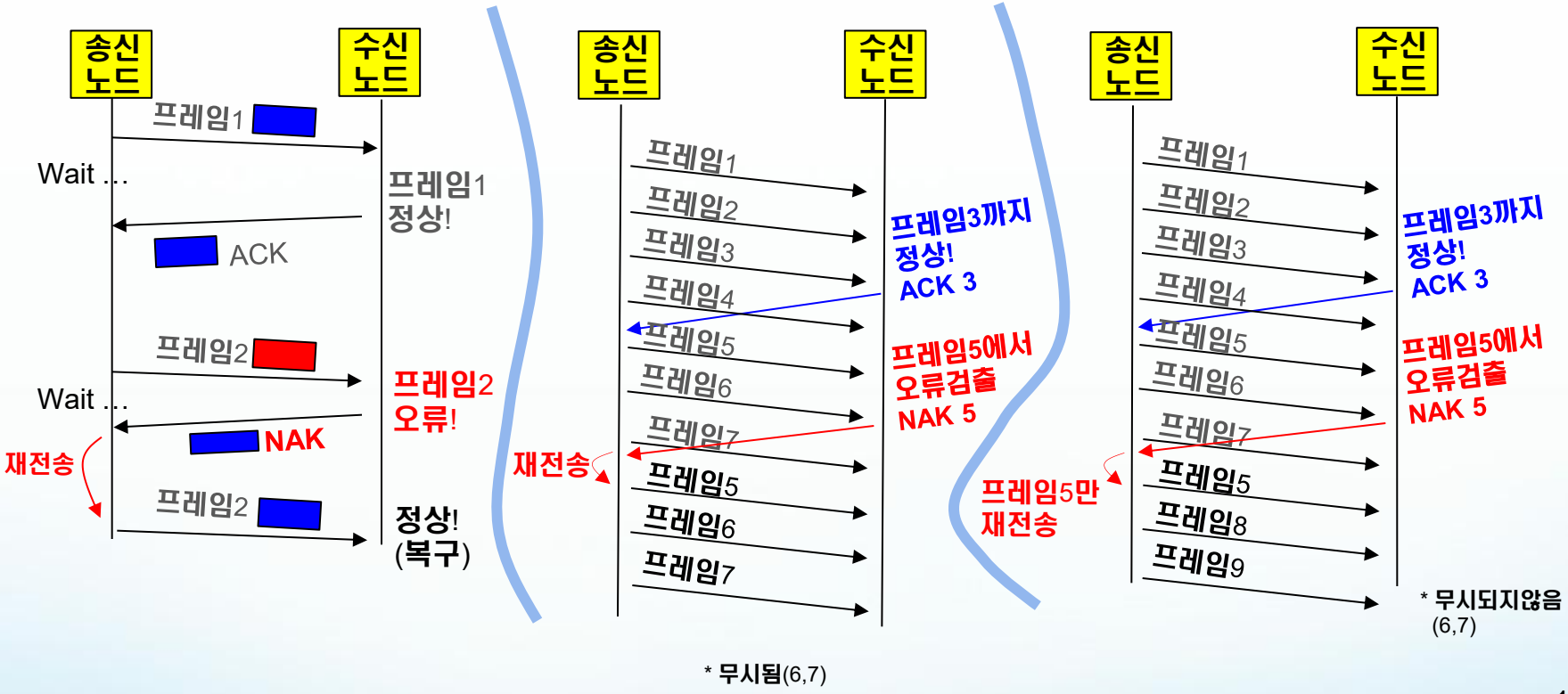
$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

데이터링크 계층

■ 오류복구/정정 — 전방향 오류정정 / 역방향 오류정정

☑ **역방향 오류정정 …** 재전송으로 오류를 해결하는 방법

- Stop-and-wait ARQ …. 프레임 1개를 전송할 때마다, ACK를 기다림
- Go-back-N ARQ ……. 프레임을 계속보내다가, NAK가 온 이후의 모든 프레임을 재전송
- Selective-reject ARQ …. 프레임을 계속보내다가, NAK가 온 프레임만 재전송



데이터링크 계층

■ 오류복구/정정 — 전방향 오류정정 / 역방향 오류정정

☑ **전방향 오류정정** … 재전송 하지 않고 오류를 해결하는 방법

- (전송 프레임내에 부가정보를 추가하여) **수신된 프레임만을 가지고 오류정정을 할 수 있는 방식**

- Hamming codes
- Binary convolution codes (IEEE 802.11a)
- Reed-Solomon codes
- Low-Density Parity Check
- …

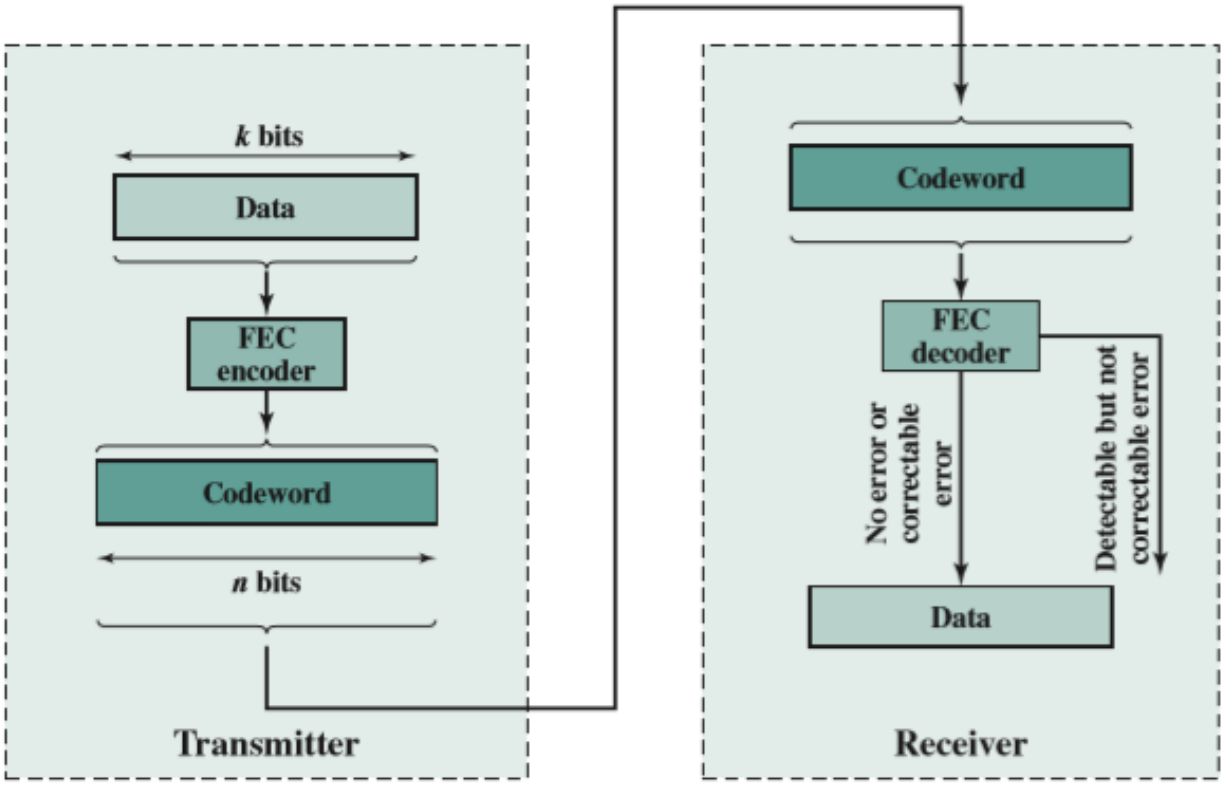


Figure 6.8 Error-Correction Process

데이터링크 계층

☑ 해밍코드(Hamming code)를 이용한 오류정정법

- 송신할 프레임 데이터: **00101110**

- 해밍코드로 Parity bit를 생성하는 과정

비트위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
원본 데이터			0		0	1	0		1	1	1	0
P_1 영역	0		0		0		0		1		1	
P_2 영역		1	0			1	0			1	1	
P_4 영역				1	0	1	0					0
P_8 영역								1	1	1	1	0
생성된 코드	0	1	0	1	0	1	0	1	1	1	1	0

생성된 패리티

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$
$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

☑ 해밍코드(Hamming code)를 이용한 오류정정법

- 해밍코드로 오류를 검출하고 오류를 정정하는 과정

수신된 프레임 데이터: 010111011110

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
0	1	0	1	1	1	0	1	1	1	1	0

☞ 패리티들을 포함하여 검사

$$P_1 = P_1 \oplus D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$P_2 = P_2 \oplus D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$P_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$
$$P_8 = P_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

0101 = 5

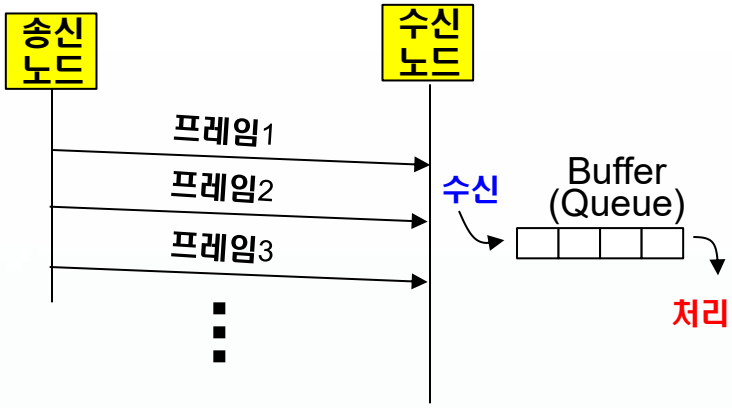
- 모든 패리티가 0이면 오류가 없는 것이고, 그렇지 않으면 오류가 발생한 것이다.
- 결과가 0101이므로 오류가 있으며, 이것을 10진수로 바꾸면 5가 된다.
  (즉, 수신된 데이터 0101**1**1011110 에서, (앞에서) 5번째 비트 1이 오류가 발생한 것이므로 0101**0**1011110으로 바꾸어 주면 오류가 정정됨)

데이터링크 계층

■ 흐름제어(Flow Control)
- Feedback-based Flow Control
  - Stop & Wait Flow Control
  - Sliding Windows Flow Control
- Rate-based Flow Control

☑ 원인

- 송신노드와 수신노드의 처리성능(속도차)이 다름
- 송신속도가 너무 빨라서 수신노드가 처리하기 어려움
- 수신버퍼링이 늦어지면 프레임이 분실될 수 있음



☑ 해결방법

- 수신노드가 송신노드의 송신시점을 제어

☑ (피드백기반의) 흐름제어 종류

- Stop & Wait 흐름제어 기법

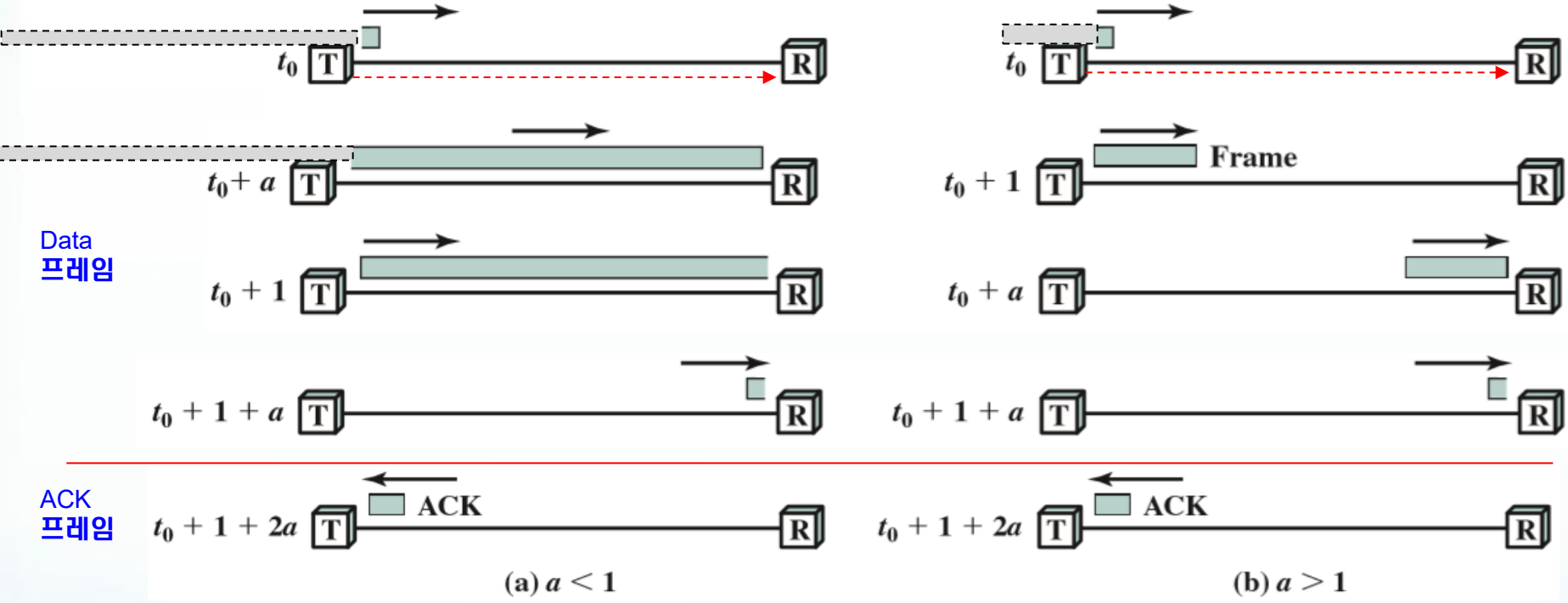
  > 오류제어(Stop & Wait ARQ)와 같은 방식, 단순한 제어방식, 성능 ▼

- Sliding Windows 흐름제어 기법

  > 정해진 양(Window)만큼 보내고 응답을 기다리는 방식. 즉, 정해진 양이상으로는 보내지 않음

☑ Stop & Wait 흐름제어 기법

- 전송프레임당 응답을 수신하는 흐름제어 기법
- 송신기는 수신기로부터 응답을 받기 전까지는 다음 프레임을 송신하지 않음
- 1 ; 프레임전송시간(transmission time), a : 프레임 전송지연시간(propagation time)



전송지연시간이
전송시간보다 작을 경우

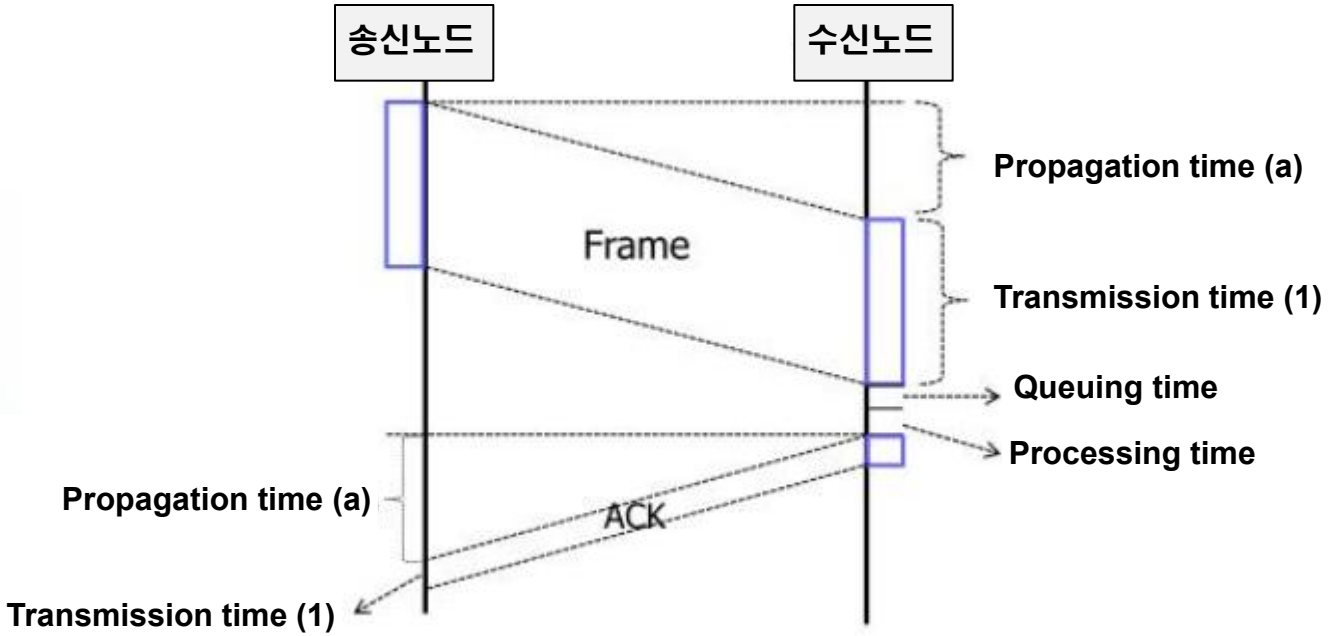
예시) a -> 100ms, 1 -> 200ms

전송지연시간이
전송시간보다 길 경우

예시) a -> 100ms, 1 -> 50ms

☑ Stop & Wait 흐름제어 기법

- 전송 소요시간, 처리시간



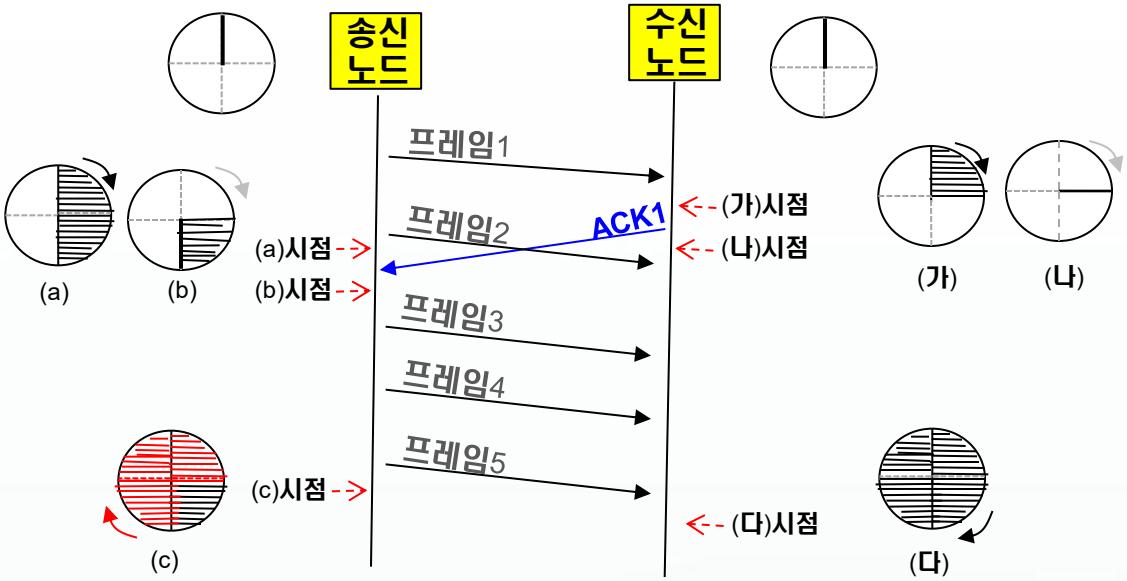
☑ Sliding Windows 흐름제어 기법

- Window size …. 응답없이 보낼 수 있는 프레임의 최대 개수

  예) Windows size = 4, 송신노드가 4개의 프레임을 최대로 보낼 수 있음

- Window 는 n-bit sequence counter 로 구현, 윈도우 범위: $0 \sim 2^n-1$

- 윈도우의 동작원리



- Window 범위 : 0~7 (3-bit), 0~63 (6-bit), 0~255 (8-bit), …

- 문제: 만약에 1-bit sequence counter 일 경우에는 ?

☑ Sliding Windows 흐름제어 기법

- 사례분석

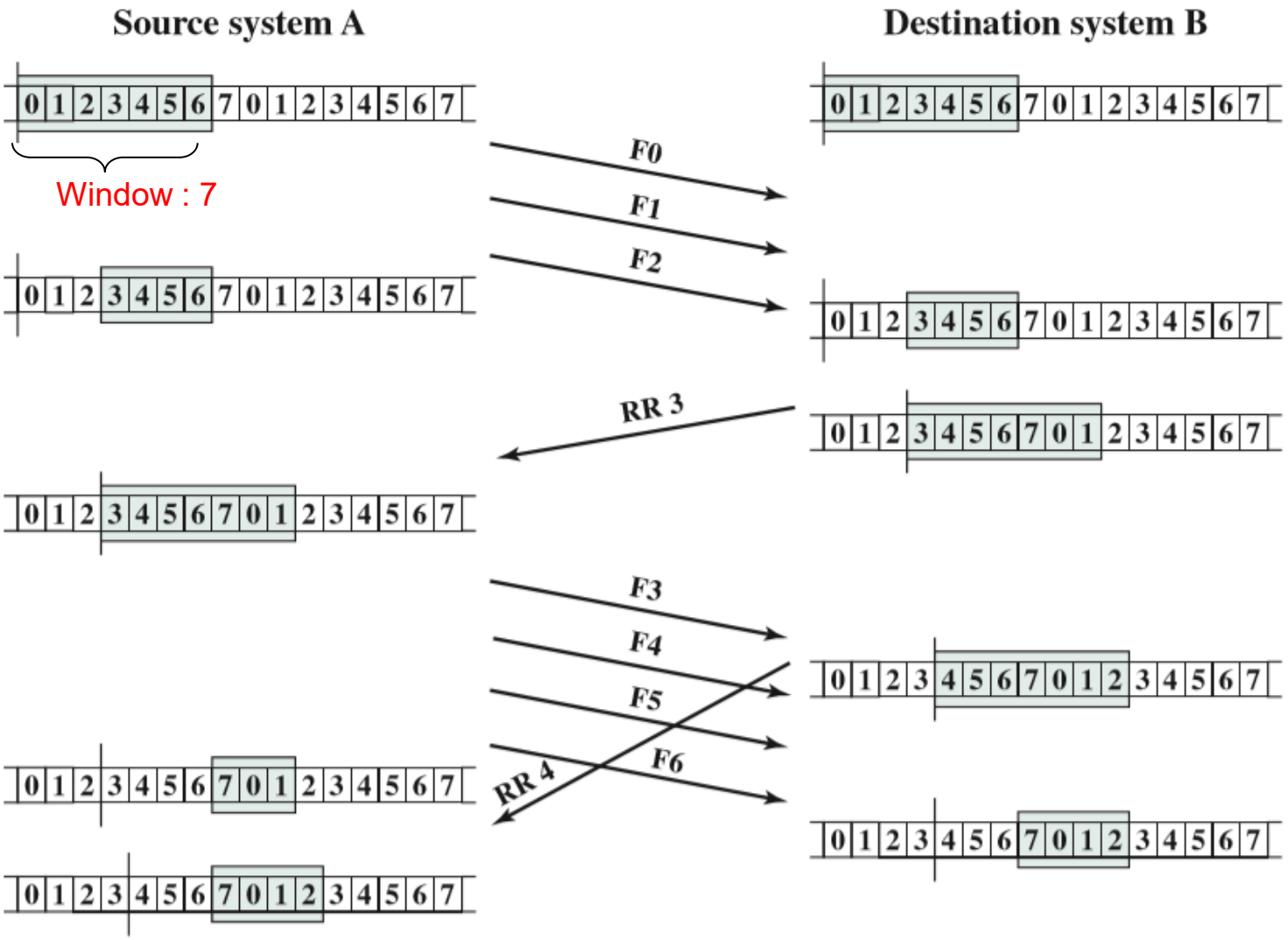
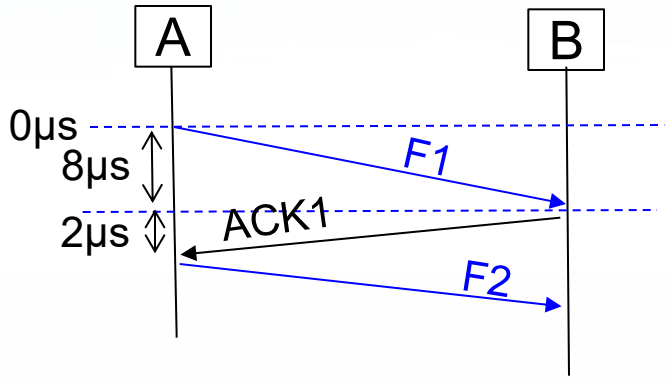
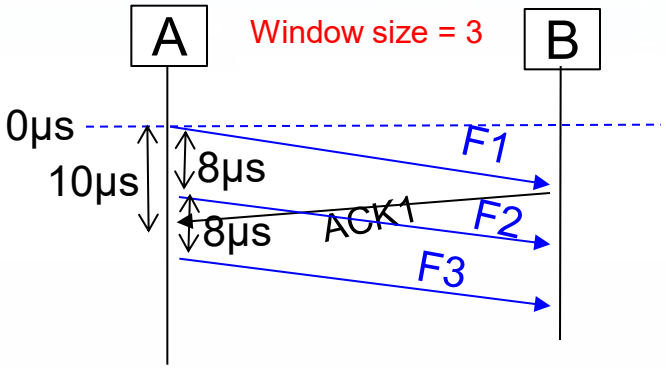


Figure 7.4 Example of a Sliding-Window Protocol

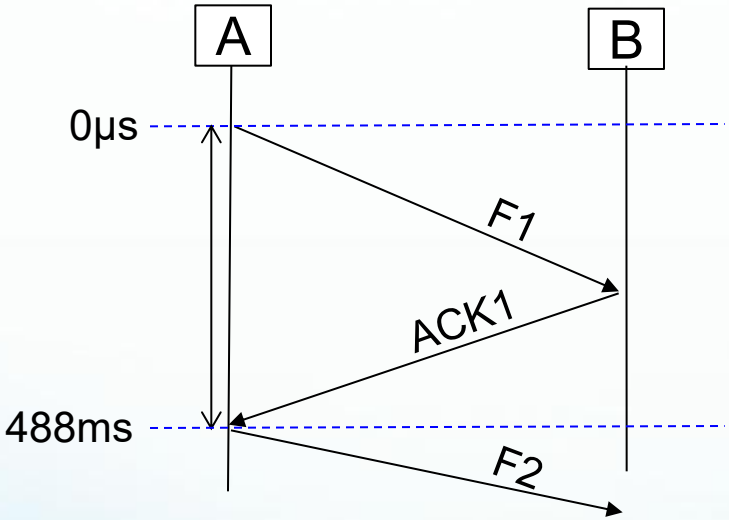
☑ 흐름제어 성능평가

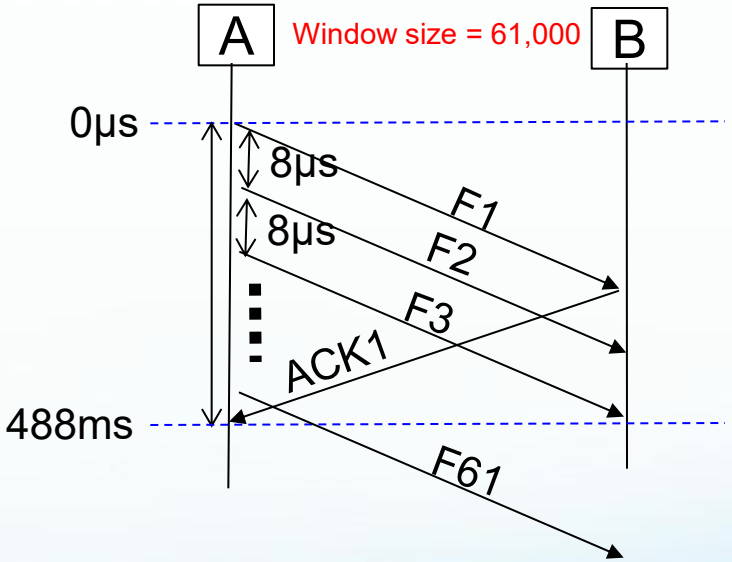
- 사례1) 데이터 프레임을 전송하는데 8μs 소요, ACK프레임까지 수신하는데 총 10μs 소요.





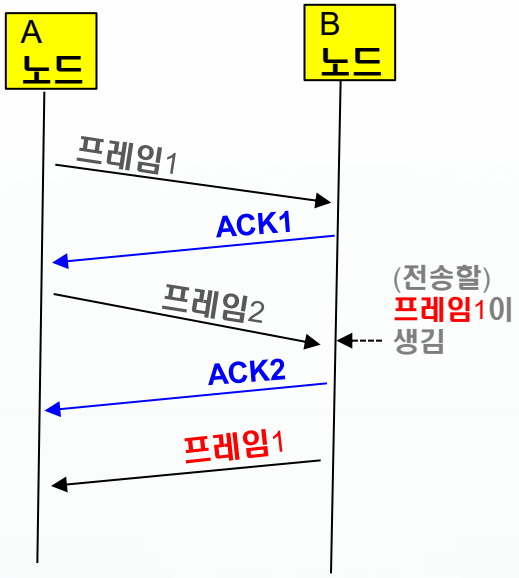
- 사례2) 위성통신의 사례에서 **(즉, 전송지연시간이 매우 김 = ACK프레임까지 수신하는데 488ms 소요)**
(160km~36000km)



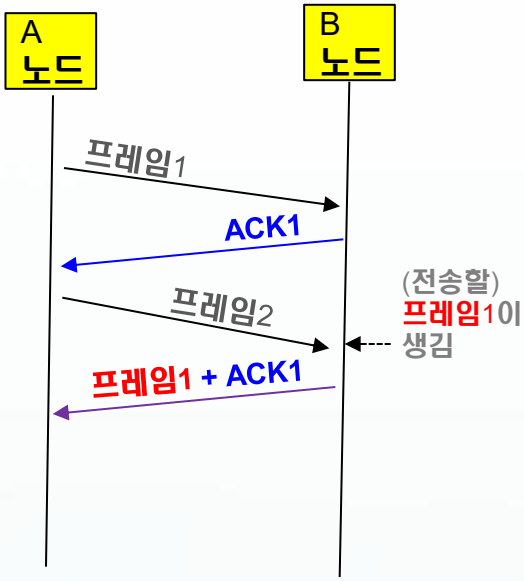


☑ Piggybacking 기법

- 응답용 ACK프레임을 별도로 전송해야 하므로, 전송 트래픽이 많아져, 비효율적임
- 전송효율을 높이기 위해, 데이터프레임(I)에 응답(ACK 또는 NAK)을 같이 포함시켜 보내는 기법 (두번에 걸쳐서 보내야 하는 프레임을 한번으로 줄일 수 있음)
- 성능향상(동일한 대역폭을 효율적으로 활용, 전송지연시간이 감소, 네트워크의 부하가 줄여짐)



Piggybacking **없이 전송**



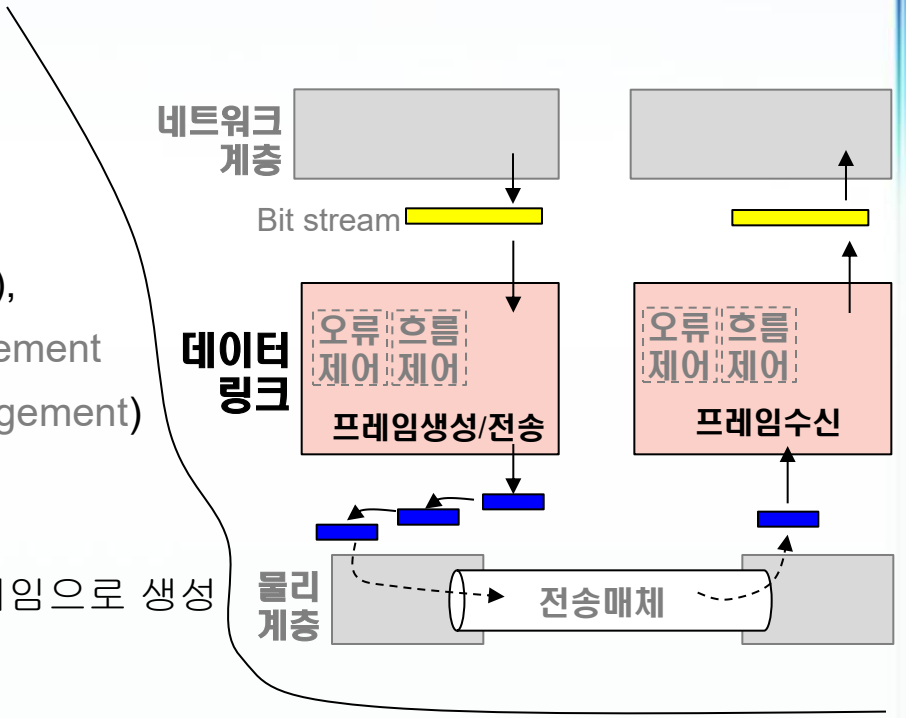
Piggybacking **으로 전송**

데이터링크 계층

■ 프레임생성/관리

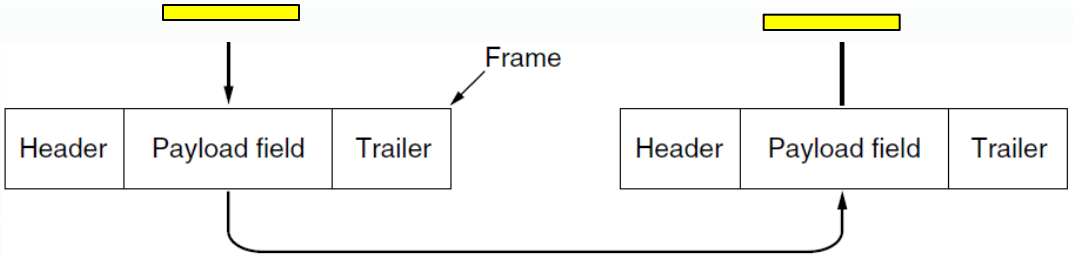
☑ 프레임(Frame) 이란 ?

- 데이터링크 계층에서 전송하는 데이터의 단위
- 프레임의 종류 ..... 데이터프레임(Information),
  응답프레임 (**ACK**nowledgement
  또는 **N**egative **A**c**K**nowledgement)



☑ 프레임생성 방법

- 네트워크 계층에서 내려온 bit stream 을 프레임으로 생성
- 구조: Header + Payload(데이터) + Trailer
- 생성시 4가지 고려사항
  - > Byte count
  - > Character stuffing
  - > Bit stuffing
  - > 물리계층 코딩침해문제

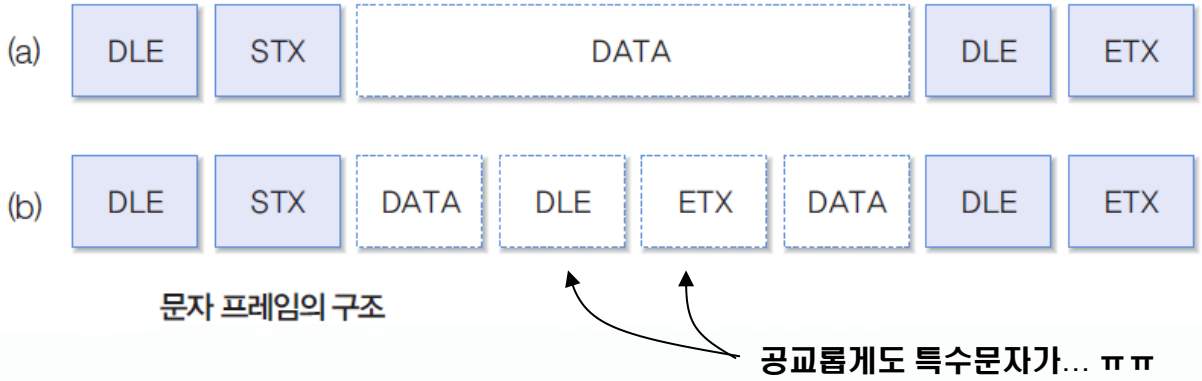


데이터링크 계층

■ 프레임

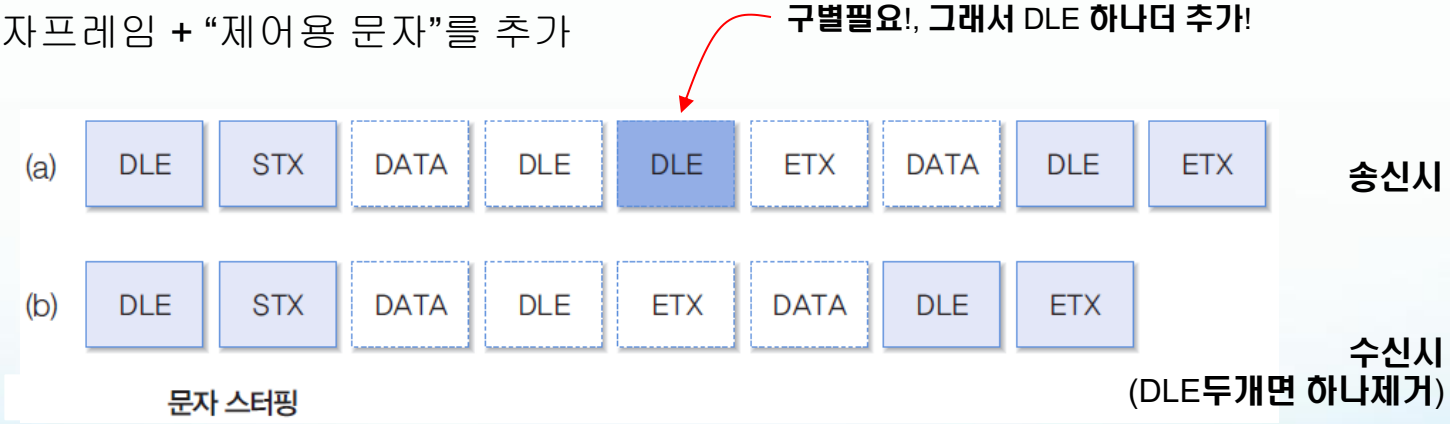
☑ 문자프레임(Character Frame)

- 데이터 내용이 문자로 구성됨(즉, 8비트 단위의 고정크기), ASCII 코드로 정의
- 프레임의 시작(DLE, STX)과 끝(DLE, ETX)를 추가하여 프레임의 경계를 구별함



☑ 문자스터핑(Character stuffing)

- 문자프레임 + "제어용 문자"를 추가

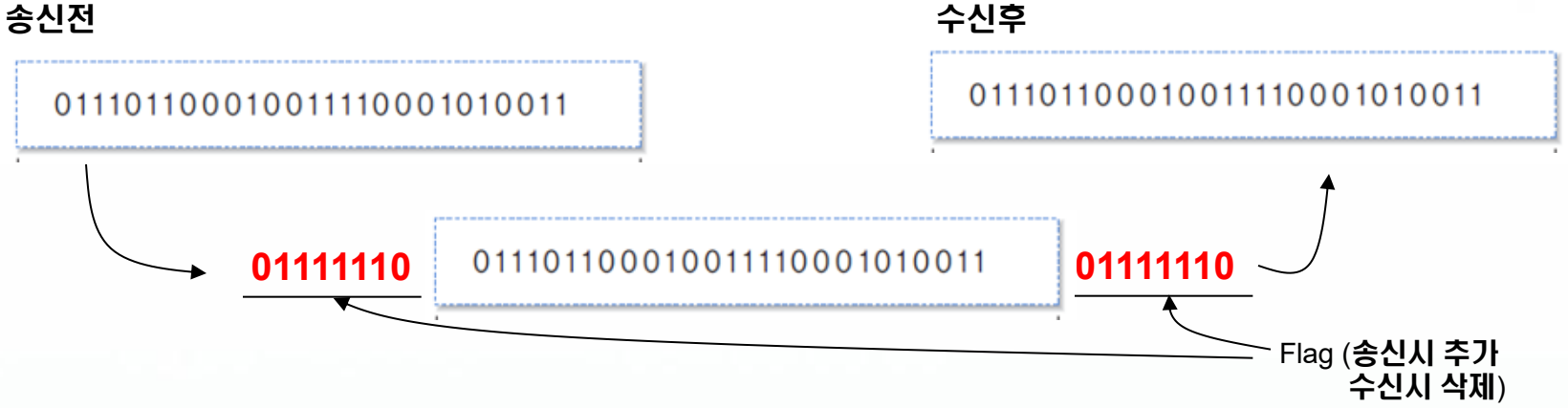


데이터링크 계층

■ 프레임

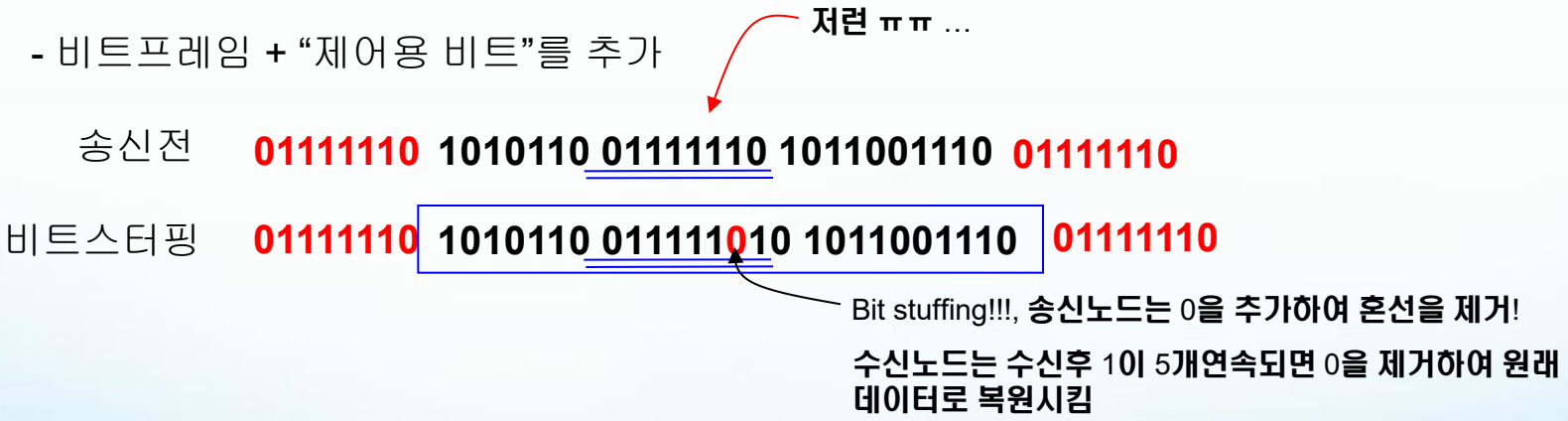
☑ 비트프레임(Bit Frame)

- 프레임 시작과 끝에 특정 비트패턴인 플래그(Flag, **01111110**)를 추가해서 시작과 끝을 구별함



☑ 비트스터핑(Bit stuffing)

- 비트프레임 + “제어용 비트”를 추가



- 접근제어의 목표
  - 공정한 매체의 이용보장
- 링크제어의 목표
  - 신뢰성있는 전송보장

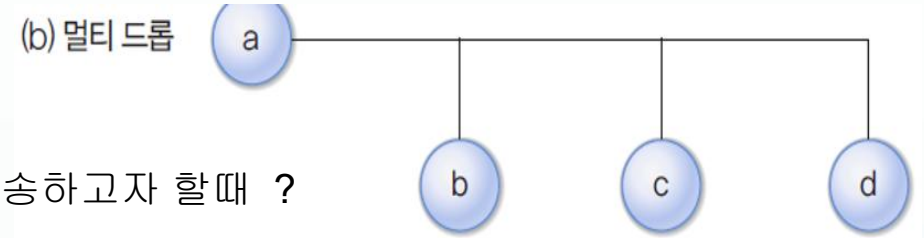
■ 접근제어/링크제어관리

☑ Point-to-Point(점대점, 1:1) 구성

- 매체접근시 노드끼리 경쟁할 필요가 없어, 단순한 송수신 절차만 관리하면 됨.
- 주소가 불필요하며, 프로토콜이 단순함.
- (예시 프로토콜)
  PPP(Point-to-Point Protocol), HDLC

☑ Mullti drop(Multi Point, 1:N) 구성

- 매체를 여러 노드가 공유하므로, (매체사용시 충돌이 예상됨) 즉, 노드끼리 경쟁하여야 함
- **접근제어와 주소가 필요**
- 마스터노드(예, a)가 b노드에서
  프레임을 전송하고자 할 때 ?
- c노드가 마스터노드(a)에게 프레임을 전송하고자 할때 ?
- (예시프로토콜)
  유선LAN(CSMA/CD), 무선LAN(CSMA/CA), 이동통신망(TDMA)



강의 **Q&A**