4 데이터 전송의 기초

가천대학교 - 2019학년도 1학기 -

Contents

학습목표

- 전송과 교환 시스템의 구조와 원리를 이해한다.
- 프레임 전송 과정에서 발생되는 오류의 유형을 살펴본다.
- 문자 프레임과 비트 프레임의 구조를 이해한다.
- 오류 검출 코드의 종류와 원리를 이해한다.
- 생성 다항식을 이용한 오류 검출 방식을 알아본다.

내용

- 데이터 전송 방식
- 오류 제어
- 프레임
- 다항 코드

01_데이터 전송 방식

❖ 컴퓨터 네트워크

- 독립적으로 실행되는 호스트들을 연결해 하나의 통신망 구성

❖ 컴퓨터 네트워크 효과

- 자원 공유
  - 컴퓨터 하드웨어, 소프트웨어, 정보 등 모든 종류의 물리적, 논리적 자원을 공유
  - 자원 활용의 극대화
- 병렬 처리에 의한 성능 향상
  - 하나의 공유 시스템 버스에 다수의 메인 프로세서를 장착
  - I/O 장치의 처리 속도를 향상시키기 위해 I/O 전용 프로세서를 설치
  - 네트워크가 시스템 버스 역할을 수행하는 방식의 분산 병렬 처리 시스템(네트워크 속도가 관건)
- 중복 저장으로 신뢰성 향상
  - 중복 저장되므로 데이터 복구가 용이함
  - 신뢰성의 향상 정도만큼 시스템 성능은 저하됨

01_데이터 전송 방식

❖ 전송과 교환

- 교환Switching : 라우터에서 데이터를 어느 방향으로 전달할지를 선택하는 기능
- 전송Transmission : 일대일(1:1)로 직접 연결된 두 시스템간의 신뢰성 있는 데이터 전송을 보장

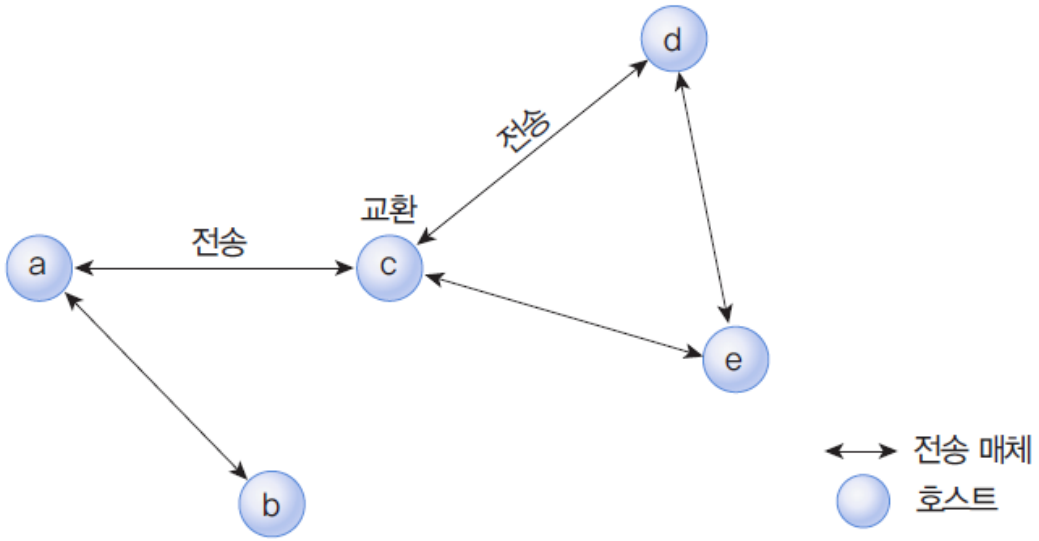


그림 4-1 전송과 교환

- 호스트 a에서 호스트 d로 데이터를 전달
  - ❶ 호스트 a와 호스트 c 간의 직접 연결에 의한 전송
  - ❷ 호스트 c에서의 올바른 경로 선택을 의미하는 교환
  - ❸ 호스트 c와 호스트 d 간의 직접 연결에 의한 전송

01_데이터 전송 방식

- 전송 방식의 종류
  - 지리적 분포에 따른 분류 방식
    - LAN(Local Area Network)(근거리 통신망)
    - MAN(Metropolitan Area Network)(도시규모의 통신망)
    - WAN(Wide Area Network)(원거리 통신망) 등
  - 데이터 전송 .교환 기술의 분류 방식
    - 점대점(Point-to-Point) , 브로드캐스팅(Broadcasting) 방식
  - 점대점 방식
    - 호스트가 중개 호스트와 일대일로 연결
    - 원거리에 있는 시스템 사이의 통신 방식, WAN 환경에서 주로 사용
  - 브로드캐스팅 방식
    - 네트워크에 연결된 모든 호스트에 데이터가 전송
    - 별도의 교환 기능이 불필요
    - LAN처럼 지리적으로 가까운 호스트 사이의 통신에서 주로 사용

01_데이터 전송 방식

❖ 점대점 방식

- 교환 호스트가 송수신 호스트의 중간에 위치, (스타형, 링형, 완전형, 불규칙형)
- 연결 개수가 많아지면 성능은 유리하나 비용이 많이 소요, 연결 개수가 적어지면 전송 매체를 많이 공유해 네트워크 혼잡도 증가
- 네트워크 트래픽이 많은 구간에서는 전송 매체의 수를 늘리고, 그렇지 않는 구간에서는 줄이는 것이 좋음

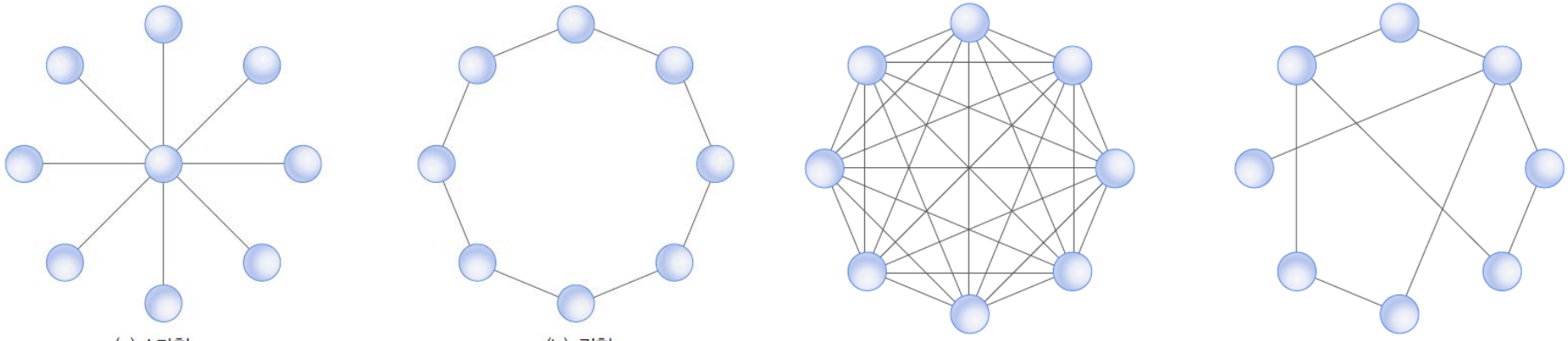
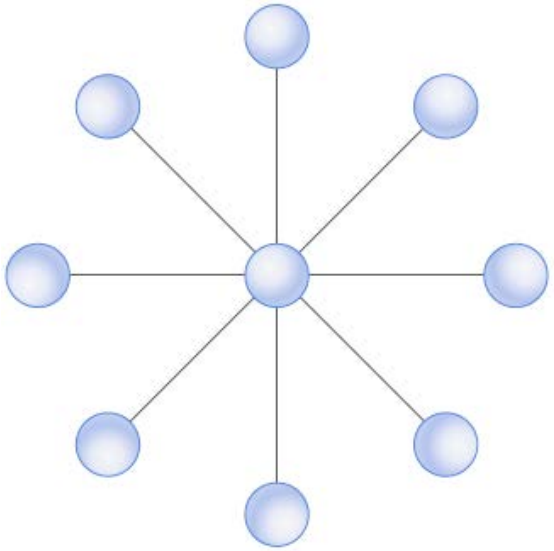


그림 4-2 점대점 방식

01_데이터 전송 방식

- 스타Star형
  - 하나의 중개 호스트 주위로 여러 호스트를 일대일로 연결하는 형태
  - 주변 호스트가 데이터를 송수신하려면 반드시 중앙의 중개 호스트를 거쳐야 함
  - 중앙 호스트의 신뢰성과 성능이 네트워크에 영향 줌
  - 데이터를 적절한 목적지로 전송하는 중개 기능도 중앙 호스트가 독점적으로 담당
  - 중개 과정이 간단하나 중앙 호스트에 문제발생시 전체 네트워크의 동작에 영향을 줌



01_데이터 전송 방식

- 트리Tree형
  - 중앙에 있는 스타 구조 주변에 위치한 호스트들을 중심으로 새로운 스타 구조가 확장되는 형태
  - 주변 호스트가 중개 호스트로 확장되는 과정을 반복해 네트워크를 무한대로 확장
  - 중개 과정이 간단하나 중앙 호스트에 문제발생시 전체 네트워크의 동작에 영향을 줌

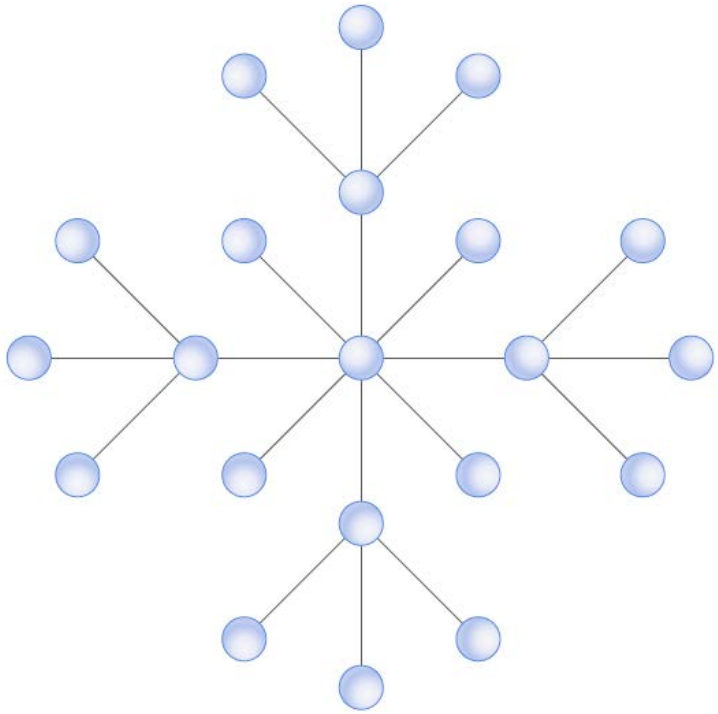
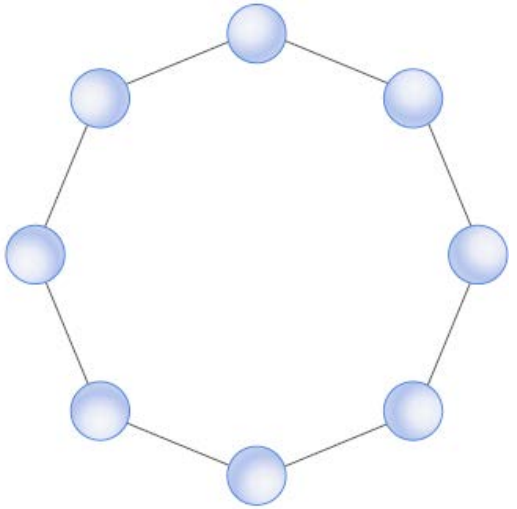


그림 4-3 트리 구조

01_데이터 전송 방식

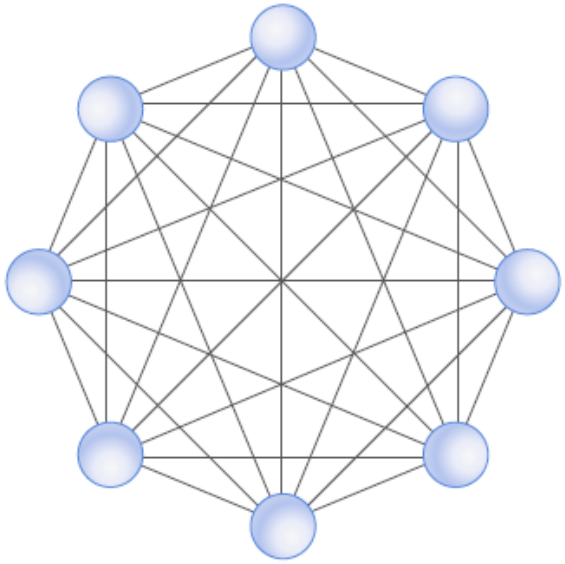
- 링Ring형
  - 호스트의 연결이 순환 고리 구조
  - 송신호스트와 수신호스트의 거리가 멀수록 데이터 전달과정에 개입하는 중개 호스트의 개수도 자연히 증가
  - 모든 호스트가 데이터 전송과 교환 기능을 동시에 수행
  - 데이터를 한 방향으로만 전달하도록 설계됨 : 전달 방향을 고정하면 복잡한 처리 과정이 단순해져 중개되는 호스트의 개수를 줄이는 것보다 유리
  - 토큰Token
    - 호스트 사이의 데이터 송신 시점을 제어하는 기능
    - 데이터의 전송 권한을 의미하는 토큰을 확보
    - 데이터 전송이 완료되면 토큰을 다시 링 네트워크에 돌려줌
  - 단점
    - 한 호스트가 고장나면 전체 네트워크가 동작하지 않을 수 있음



(b) 링형

01_데이터 전송 방식

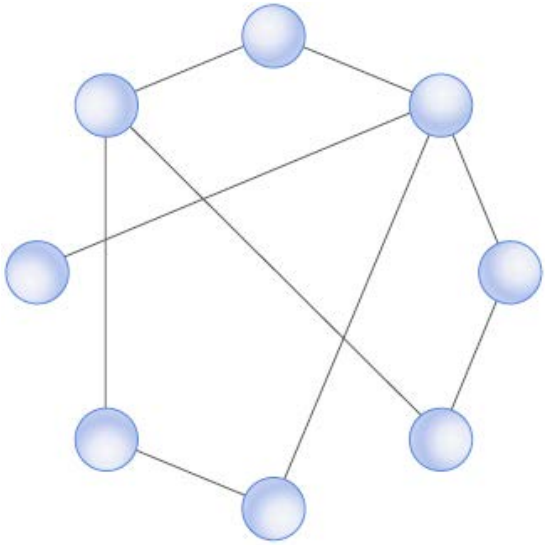
- 완전형
  - 모든 호스트가 다른 모든 호스트와 일대일로 직접 연결하는 방식
  - 단점 : 전송 매체가 증가하면 비용 측면이 비효율적임
  - 전체 호스트의 개수가 N이라고 가정하면 연결에 필요한 전송 매체의 수는 {Nx(N-1)}/2



(c) 완전형

01_데이터 전송 방식

- 불규칙형
  - 연결 구조를 특정 패턴으로 분류할 수 없는 방식(일반 네트워크)
  - 예) 실제 네트워크에서는 특정한 두 호스트사이에 통신 트래픽이 많으면 이들을 직접 연결하고, 적으면 다른 호스트의 중개 과정을 거쳐서 데이터를 주고받도록 설계



(d) 불규칙형

01_데이터 전송 방식

점대점 방식

- 전송 매체를 사용해 각 호스트를 일대일로 직접 연결한다는 전제
- 어떤 호스트를 어떤 구조를 연결하는지?
- WAN 같은 원거리 통신망에서는 특정 호스트 사이에 직접 연결한 전송 매체가 있는지, 아니면 어떤 우회 경로로 몇 단계를 거쳐서 연결할 수 있는지에 대한 판단이 중요

브로드캐스팅 방식

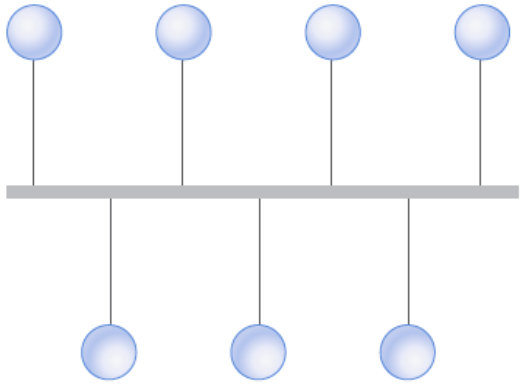
- 특정 호스트가 전송한 데이터가 네트워크에 연결된 모든 호스트에 전달
- LAN에서는 연결의 존재 여부보다 데이터를 목적지 호스트까지 얼마나 효율적으로 전달할 수 있는지가 중요
- 네트워크의 모든 호스트가 하나의 전송 매체로 연결하므로 중개 기능을 하는 교환 호스트 불필요
- 목적지 호스트는 해당 데이터를 수신하고 보관하지만, 다른 호스트들은 수신 데이터를 버림
- 버스형과 링형

01_데이터 전송 방식

- 버스Bus형
  - 공유 매체인 버스에 모든 호스트 연결
  - 전송 데이터를 모든 호스트에서 수신할 수 있음
  - 충돌Collision : 둘 이상의 호스트에서 데이터를 동시에 전송할 때 충돌 발생
  - 충돌 해결 방법

    1. 호스트의 전송 권한을 제한함
    - 사전에 전송 권한을 확보하는 방법 : 시간대를 다르게 지정하는 방법,
      토큰으로 전송 권한을 순환적으로 이용하는 방법

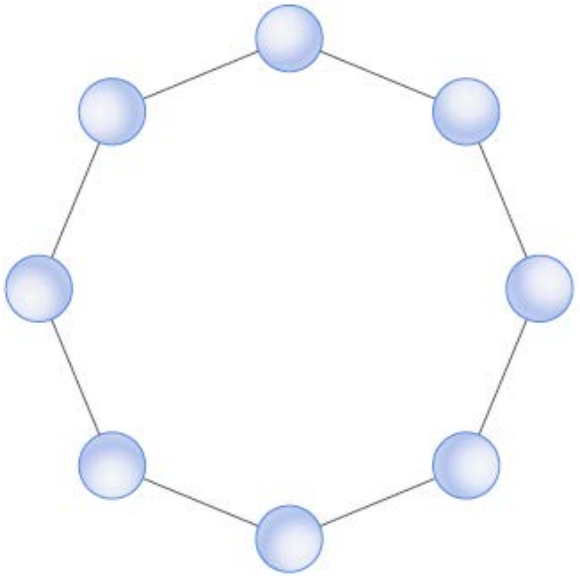
    2. 충돌 허용
    - 둘 이상의 호스트가 데이터를 동시에 전송할 수 있도록 허용, 충돌 발생시에 해결 과정 필요
    - 예 : 이더넷Ethernet



(a) 버스형

01_데이터 전송 방식

- 링Ring형
  - 호스트를 순환 구조로 연결
  - 송신 호스트가 전송한 데이터는 링을 한 바퀴 순환한 후 송신 호스트에 되돌아옴
  - 링에 연결된 모든 호스트에 데이터가 전달됨
  - 중간의 호스트 중에서 수신 호스트로 지정된 호스트만 데이터를 내부에 저장
  - 데이터를 전송하기 위해서는 토큰 확보가 필수



(b) 링형

01_데이터 전송 방식

❖ 멀티포인트 통신

- 유니캐스팅Unicasting 방식
  - 두 호스트 사이의 데이터 전송 (텔넷, FTP, 웹 검색)
- 멀티포인트Multipoint
  - 일대다(1:n), 다대다(n:n) 형식 (화상 회의, 원격 교육, 인터넷 채팅)

- 하나의 송신 호스트를 기준으로
  - 수신 호스트 하나와 연결 : 유니포인트Unipoint
  - 다수의 수신 호스트와 연결 : 멀티포인트Multipoint

- 송신 호스트가 한 번의 전송으로
  - 수신 호스트 하나에만 데이터 전송 : 유니캐스팅Unicasting
  - 다수의 수신 호스트 전송 : 멀티캐스팅Multicasting

01_데이터 전송 방식

- 멀티포인트 유니캐스팅MultipointUnicasting
  - 유니캐스팅 방식을 이용한 일대다 통신을 위해 멀티포인트 유니캐스팅 방식을 사용

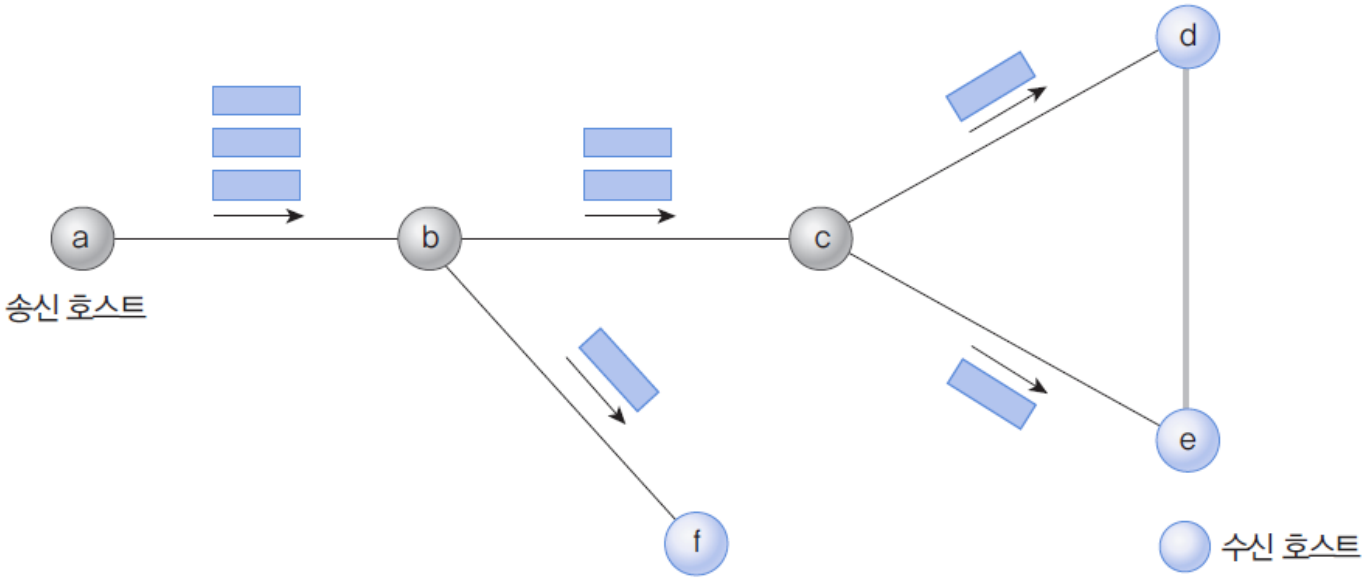


그림 4-5 멀티포인트 유니캐스팅

  - 단점 : 수신 호스트 수가 많아지면 성능 에서 문제 발생
  - 장점 : 송수신 호스트 사이의 흐름 제어와 수신 호스트의 응답 기능 및 재전송 기능 등을 구현하기 좋음

01_데이터 전송 방식

- 브로드캐스팅Broadcasting
  - 네트워크에 연결된 모든 호스트에 전송되는 방식
  - 수신된 데이터의 처리는 수신 호스트가 담당(수신 데이터의 목적지 주소로 판단)
  - 단점 : 호스트 수가 많을수록 네트워크 트래픽이 급격히 증가
    (서브넷Subnet 내에서 이용이 좋음)

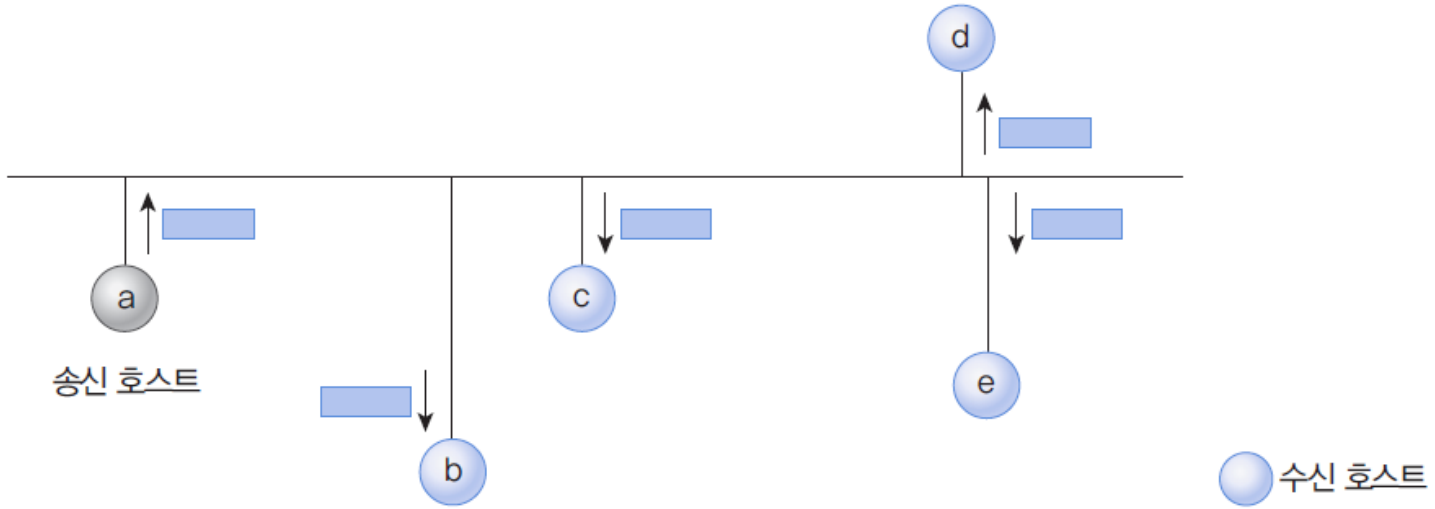
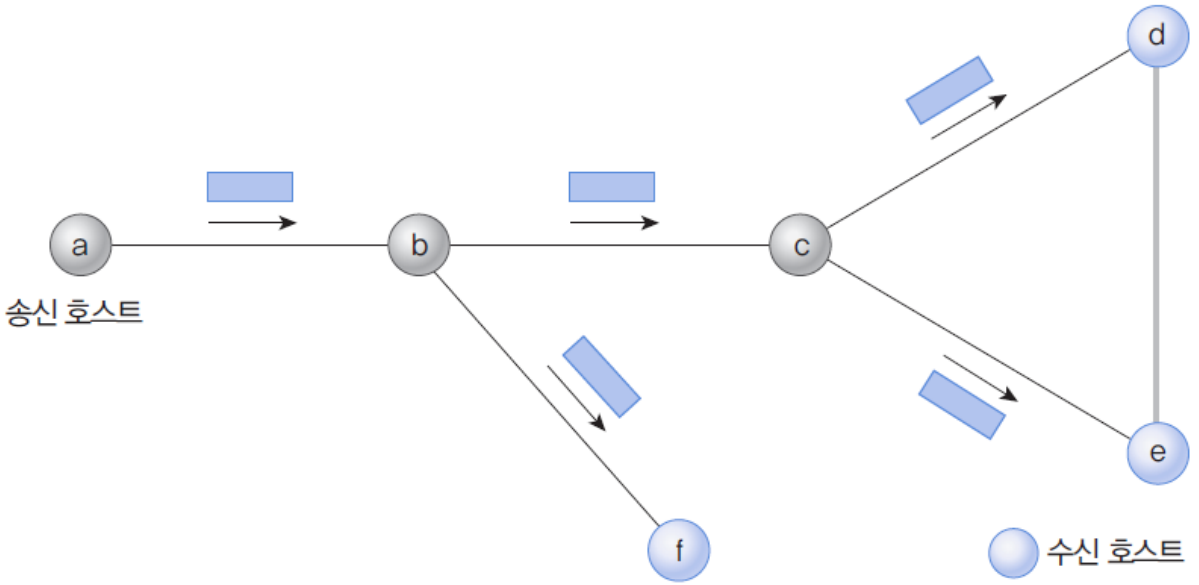


그림 4-6 브로드캐스팅

  - 특정한 브로드캐스팅 주소로 전송
  - 네트워크 장비에서는 전달된 패킷을 복사하여 네트워크 전체로 전송

01_데이터 전송 방식

- 멀티캐스팅Multicasting
  - 일대다 전송 기능을 구현 : 통신 환경을 연결 설정 요구 한 번으로 지원 가능
  - 송신호스트의 전송 요구 한 번으로 모든 수신 호스트에 데이터를 전달
  - 예) 비디오.오디오 서비스, 화상 회의 서비스, 인터넷 뉴스, 인터넷 주식



  - 멀티캐스트 그룹을 생성하고 관리하는 기능 필요
  - 임의의 호스트가 특정 멀티캐스트 그룹에 가입하고 탈퇴하는 기능
  - 멀티캐스트 데이터를 중개하는 라우터에서 멀티캐스트 그룹 주소 인식, 다수의 수신 호스트에 중개하는 등 멀티캐스트 트래픽 처리 기능 구현 필요

02_오류 제어

전송 오류 기능

- 오류 발생 여부 인지
  - 오류의 종류 : 프레임 변형, 프레임 분실
- 오류 복구
  - 데이터 프레임은 원래의 데이터 외에 오류 검출을 위한 정보를 함께 제공
  - 수신 호스트에서 오류를 감지하는 기능만 하는 정보와 오류가 발생한 프레임을 복구하는 기능을 하는 정보
  - 일반적으로 송신 호스트가 원래의 데이터를 재전송하는 기법을 사용

02_오류 제어

❖ 전송 오류의 유형

- 오류 복구 기능
  - 수신 호스트의 응답 프레임 : 송신 호스트에 응답 프레임을 전송해 원래의 데이터 프레임을 재전송하도록 요구
    - 긍정 응답 프레임
    - 부정 응답 프레임 : 송신 호스트의 재전송 기능 작동
  - 송신 호스트의 타이머 기능
    - 타임아웃Timeout : 데이터 프레임을 전송한 후에 일정 시간 이내에 수신 호스트로부터 긍정 응답 프레임 회신이 없으면 데이터 프레임을 재전송함
  - 순서 번호 기능
    - 수신 호스트가 중복 프레임을 구분할 수 있도록 지원
    - 데이터 프레임 내에 프레임 구분을 위한 일련 번호 부여

02_오류 제어

❖ 정상적인 전송

- 송신 호스트가 전송한 데이터 프레임이 수신 호스트에 오류 없이 도착
- 수신 호스트는 송신 호스트에게 긍정 응답 프레임을 회신

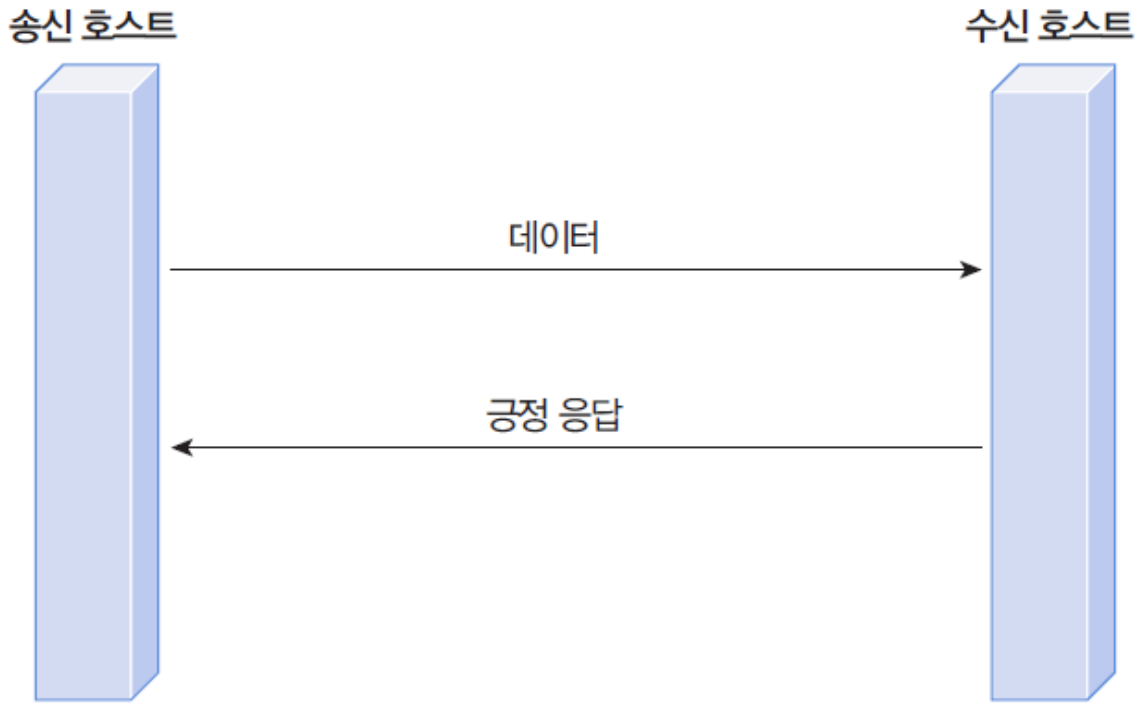


그림 4-8 정상적인 데이터 전송

02_오류 제어

❖ 프레임 변형

- 프레임 변형 오류를 인지한 수신 호스트는 송신 호스트에 부정 응답 프레임을 전송, 원래의 데이터 프레임을 재전송함
- 부정 응답 프레임을 사용하지 않는 프로토콜에서는 송신 호스트의 타임아웃 기능에 따라 복구 과정을 시작

그림 4-9 프레임 변형 오류

02_오류 제어

❖ 프레임 분실

- 송신 호스트는 데이터 프레임을 전송한 후에 특정 시간까지 수신 호스트의 긍정 응답 프레임이 도착하지 않으면 타임아웃 기능에 따라 원래의 프레임을 스스로 재전송

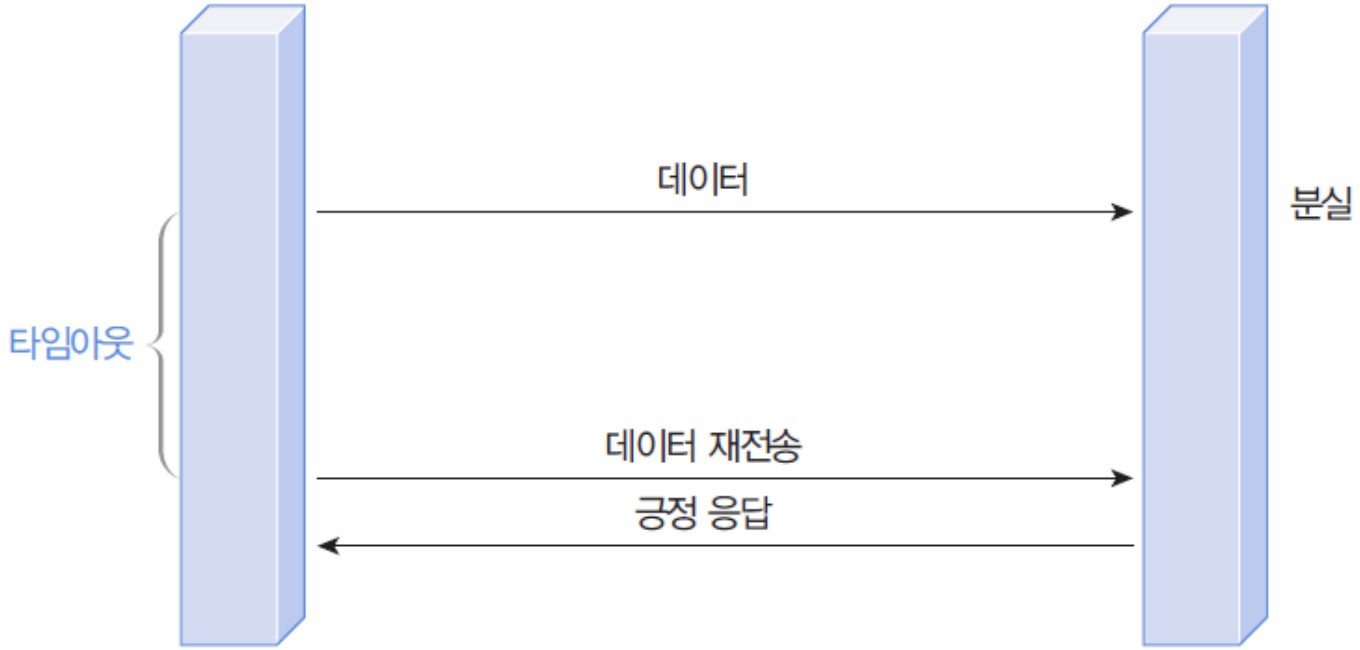
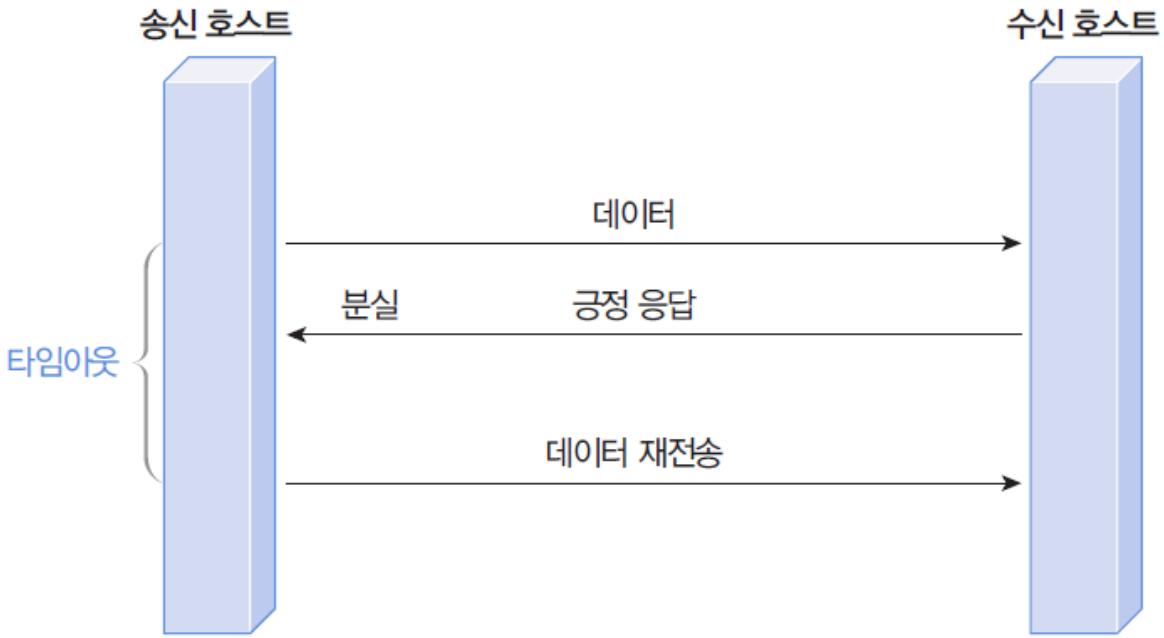


그림 4-10 프레임 분실 오류

02_오류 제어

❖ 순서 번호Sequence Number

- 중복 수신 문제를 해결하기 위해 데이터 프레임에게 부여되는 고유 번호
- 순서 번호의 필요성
  - 긍정 응답 프레임이 사라지는 오류 발생시 송신 호스트의 타임아웃 기능에 따라 재전송 과정이 진행됨 → 동일한 프레임 중복 수신



(a) 긍정 응답 분실

02_오류 제어

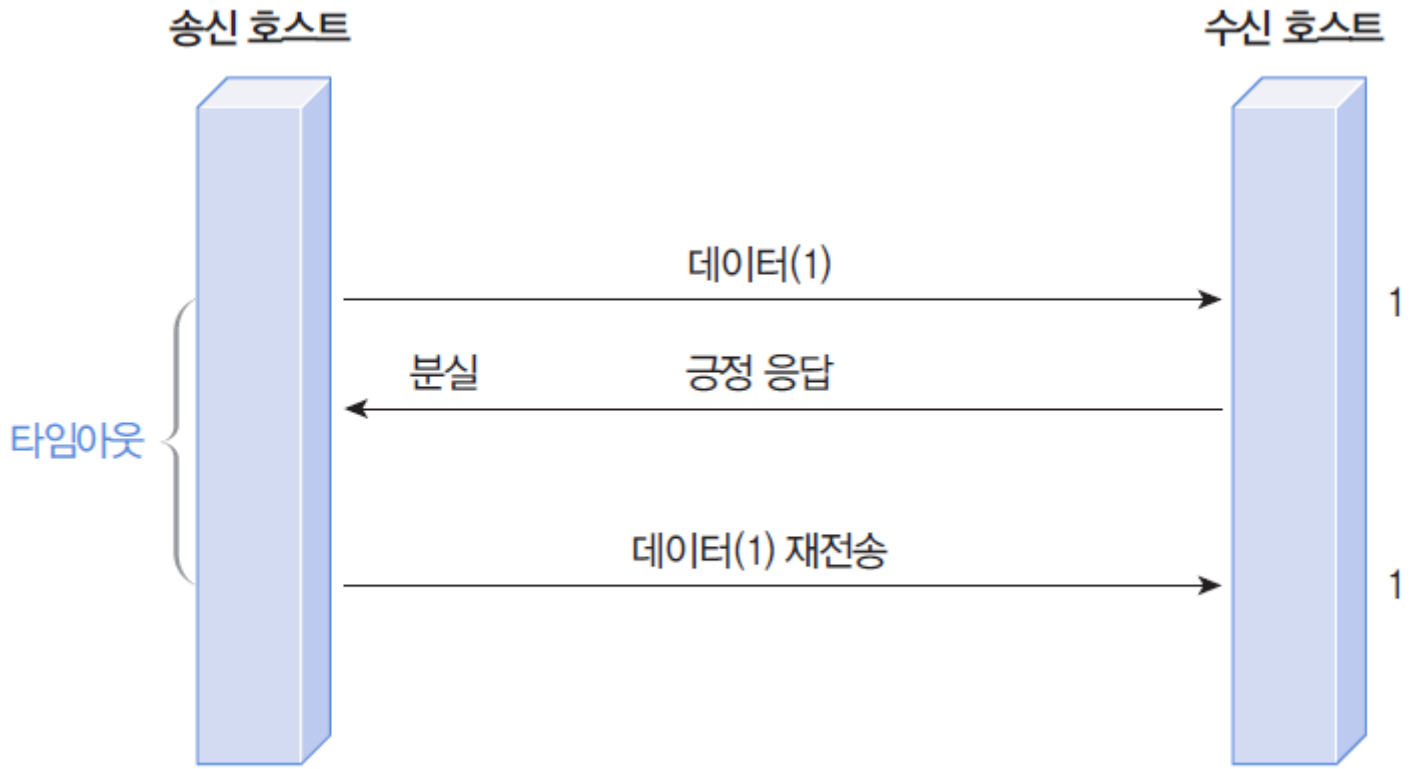
- 수신 호스트가 두 경우((a) 긍정 응답 분실, (b) 긍정 응답 도착)를 구분할 수 있도록 데이터 프레임별로 고유의 순서 번호를 부여하는 방식이 필요함

(b) 긍정 응답 도착

그림 4-11 순서 번호가 없는 경우

02_오류 제어

- 순서 번호에 의한 프레임 구분
  - 순서 번호에 근거하여 동일한 데이터 프레임이 중복 도착여부를 확인 가능



(a) 긍정 응답 분실

02_오류 제어

- 서로 다른 데이터 프레임이 도착

(b) 긍정 응답 도착

그림 4-12 순서 번호가 있는 경우

02_오류 제어

흐름 제어Flow Control

- 수신 호스트가 감당할 수 있을 정도의 전송 속도를 유지하면서 데이터 프레임을 전송

- 너무 빨리 전송하는 경우
  - 수신 호스트가 내부 버퍼에 보관하지 못할 수 있음
  - 이는 프레임 분실과 동일한 효과를 야기함

- 기본 원리
  - 수신 호스트가 송신 호스트의 전송 시점을 제어
  - 대표적인 예: 슬라이딩 윈도우 프로토콜 (6장에서 다룸)
    - 위도우 크기(Window Size) : 임의의 시점에서 송신 호스트가 수신 호스트로부터 긍정 응답 프레임을 받지 않고도 전송할 수 있는 정보 프레임의 최대 개수

03_프레임

- 데이터 링크 계층 : 전송 데이터를 프레임Frame이라는 단위로 나누어 처리
- 전송프레임 : 체크섬Checksum, 송수신 호스트의 주소, 제어 코드 등 정보포함
- 프레임을 수신한 호스트는 제일 먼저 체크섬을 확인 해 전송 오류(프레임 변형)이 발생했는지 확인
  - 오류가 발생하면 부정 응답 프레임 회신 -> 송신 호스트가 데이터 프레임 재전송
- 프레임 내용에 포함되는 정보는 프로토콜에 따라 다름
- 내부정보를 표현하는 방식에 따라 문자 프레임과 비트 프레임으로 구분

03_프레임

문자 프레임Character Frame

- 내용이 문자로 구성. 문자 데이터를 전송할 때 사용
- 8비트 단위(또는 ASCII 문자 코드)의 고정 크기로 동작
- 프레임의 구조
  - 프레임의 시작과 끝에 특수 문자 사용
    - 시작 : DLE / STX
    - 끝 : DLE / ETX
  - 전송 데이터에 특수 문자가 포함되면 혼선이 발생

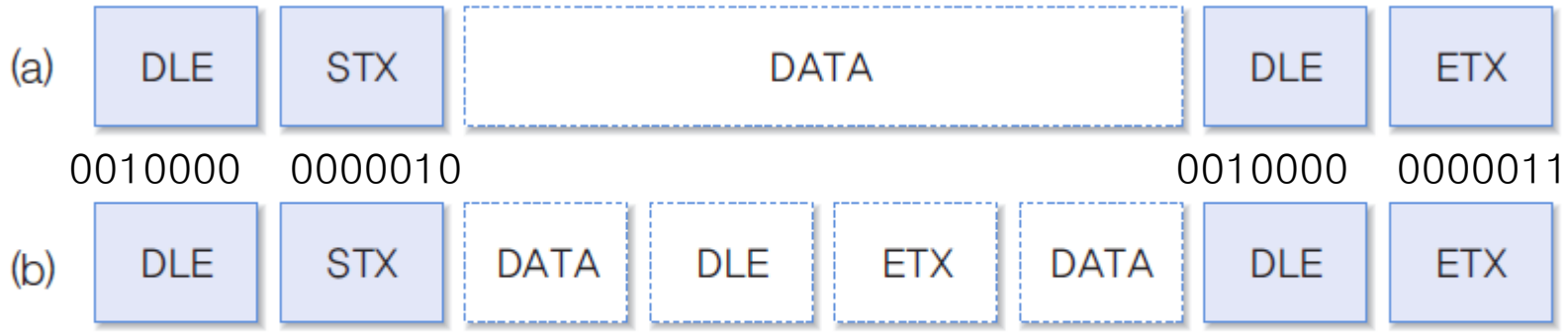


그림 4-13 문자 프레임의 구조

[참고] ASCII코드

제어 문자 | 공백 문자 | 구두점 | 숫자 | 알파벳

10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

03_프레임

- 문자 스터핑Character Stuffing
  - 문자 프레임 내부의 전송 데이터에 DLE 문자가 포함되면서 발생하는 혼란 예방
  - 문자 프레임의 전송 과정에서 제어 문자를 추가하는 기능

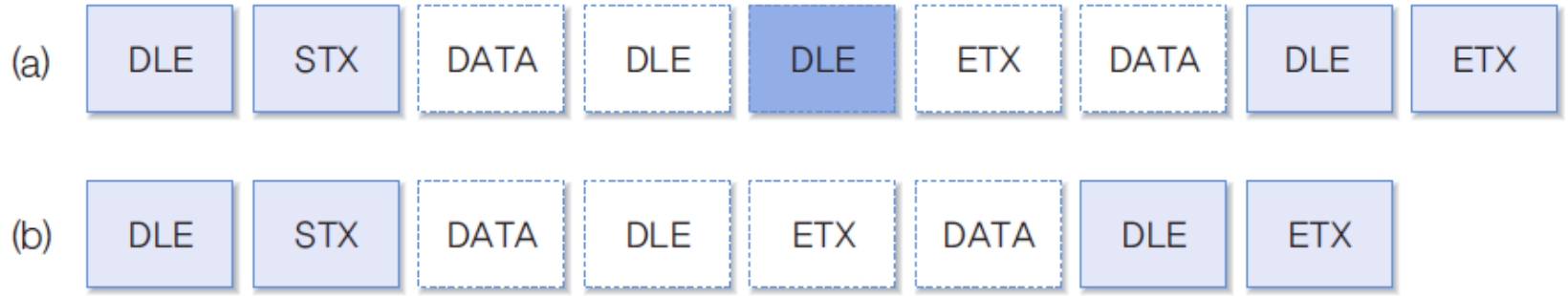


그림 4-14 문자 스터핑

  - 전송 데이터가 DLE 문자를 포함하면 뒤에 DLE 문자 하나를 강제 추가
  - 데이터에 DLE 문자가 두 번 연속 있으면 하나의 DLE 문자를 삭제하고 상위 계층으로 데이터 전달

03_프레임

비트 프레임Bit Frame

- 프레임의 시작과 끝 위치에 플래그라는 특수하게 정의된 비트 패턴(01111110)을 사용해 프레임 단위를 구분
- 프레임의 구조
  - 데이터 전송 전에 프레임의 좌우에 플래그를 추가, 수신 호스트는 이 플래그를 제거해 전송 데이터와 필요한 제어 정보를 상위 계층 전달

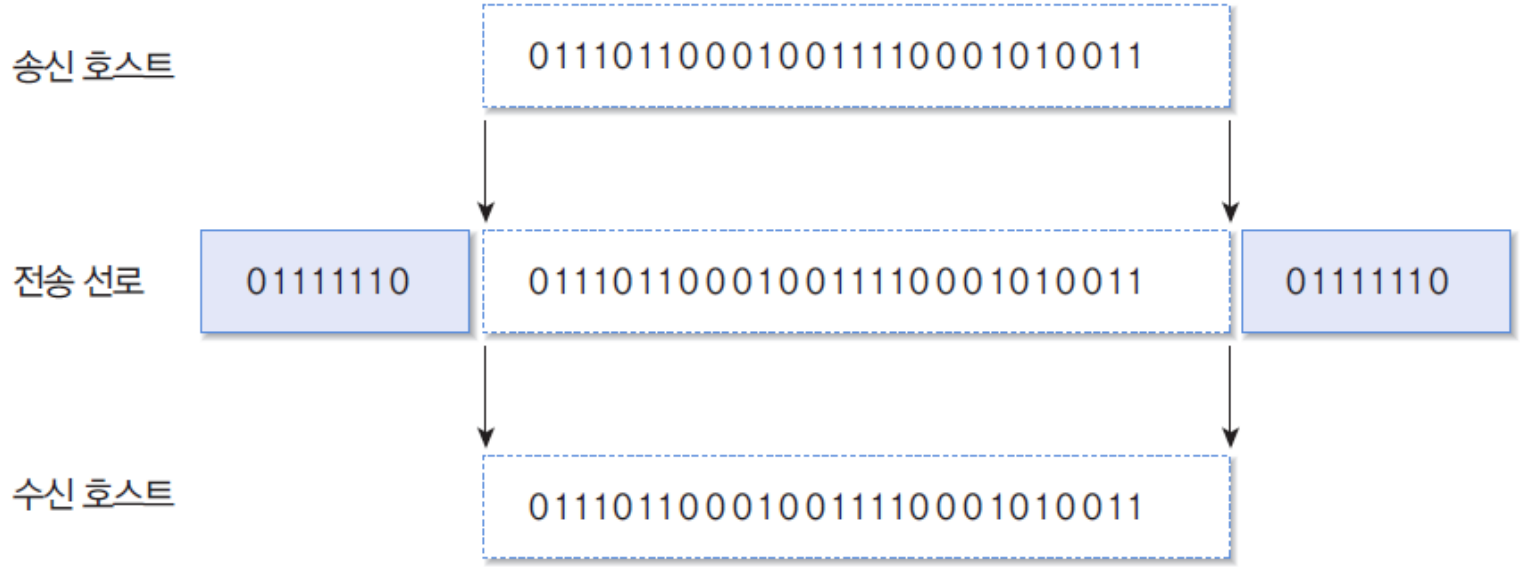


그림 4-15 비트 프레임의 구조

03_프레임

- 비트 스터핑Bit Stuffing
  - 전송 데이터에 플래그 패턴이 포함되면 혼선이 발생
  - 송신 호스트 : 전송 데이터가 1이 연속해서 5번 발생하면 강제로 0을 추가
  - 수신 호스트 : 송신 과정에 추가된 0을 제거하여 원래의 데이터를 상위 계층 전달

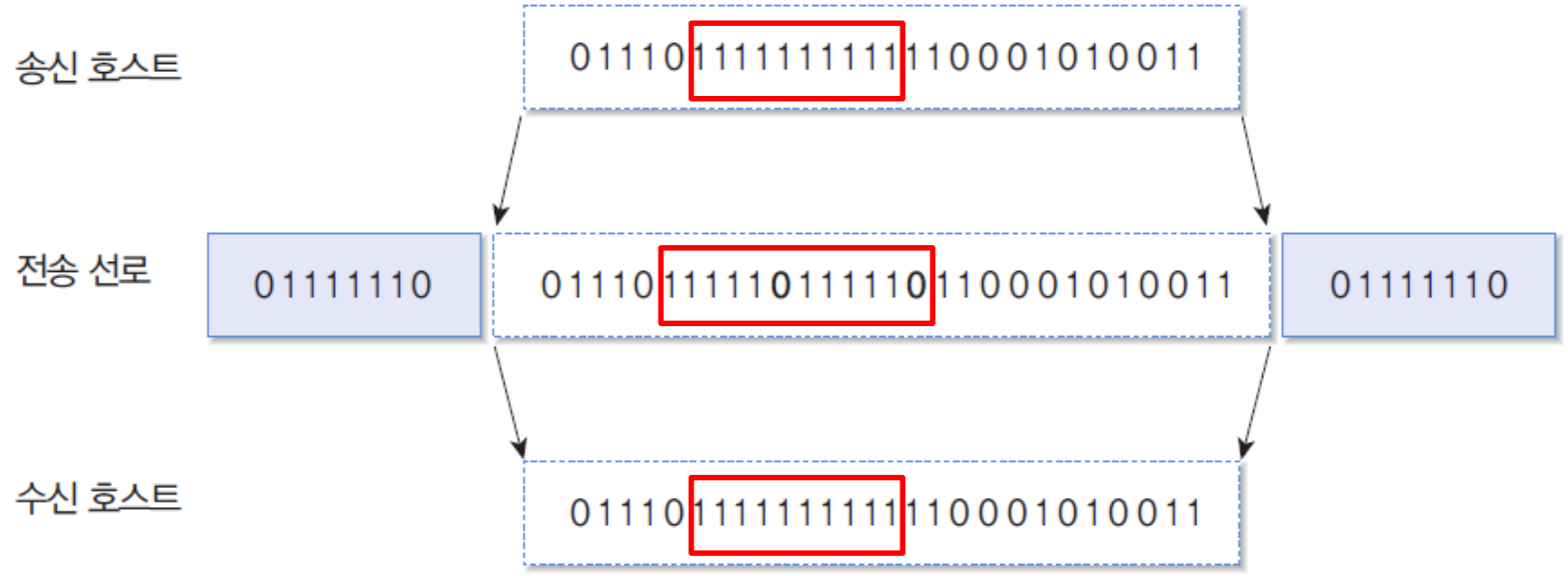


그림 4-16 비트 스터핑

04_다항 코드

❖ 오류를 극복하는 방법

- 오류 검출
  - 역방향 오류 복구BEC, Backward Error Correction (ARQAutomatic Repeat reQuest 방식)
  - 재전송 방식을 이용해 오류 복구(네트워크에서 일반적으로 사용)
  - 전송 프레임에 오류 검출 코드를 넣어 수신 호스트가 전송 과정의 오류를 검출
  - 오류 검출 코드 : 패리티 비트, 블록 검사, 다항 코드 등

- 오류 검출 & 복구
  - 순방향 오류 복구FEC, Forward Error Correction
  - 오류 복구 코드를 사용해 수신 호스트가 오류 복구 기능을 수행하는 방식
  - 예) 해밍 코드Hamming Code : 1비트 오류를 검출하고 복구하는 기능

04_다항 코드

❖ 오류 검출

- 패리티Parity 비트
  - 1 바이트(8비트) = 7 비트 ASCII 코드 + 1 패리티 비트
  - 짝수 패리티 : 데이터 끝에 패리티 비트를 추가해 전체 1의 개수를 짝수로 만듦
  - 홀수 패리티 : 1의 개수를 홀수로 만드는 것
  - 송신 호스트와 수신 호스트는 동일한 패리티 방식을 사용해야 함

그림 4-17 패리티 비트

04_다항 코드

- 블록 검사
  - 짝수개의 비트가 깨지는 오류를 검출
  - 수평, 수직 방향 모두에 패리티 비트를 지정

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

블록 검사

그림 4-18 **블록 검사**

04_다항 코드

다항 코드Polynomial Code (CRCCyclic Redundancy Code)

- 현재의 통신 프로토콜에서 가장 많이 사용하는 오류 검출 기법
- 특정 위치에서 집중적으로 발생하는 버스트 에러Burst Error를 검출하는 확률이 높음

- 생성 다항식
  - 계수가 0과 1인 다항식 형태를 기반
  - 다항코드 100101 = $1\times x^5 + 0\times x^4 + 0\times x^3 + 1\times x^2 + 0\times x^1 + 1\times x^0$
    = 생성 다항식 $x^5 + x^2 + 1$
- [그림 4-19]
  - 전송 데이터 : m비트의 M(x)
  - n+1비트의 생성 다항식 G(x)를 사용해 오류 검출 코드를 생성, 오류 제어
  - 전송 데이터 M(x)를 생성 다항식 G(x)로 나누어 체크섬Checksum 정보를 얻음
  - 연산에서 얻은 나머지 값을 체크섬이라 함

04_다항 코드

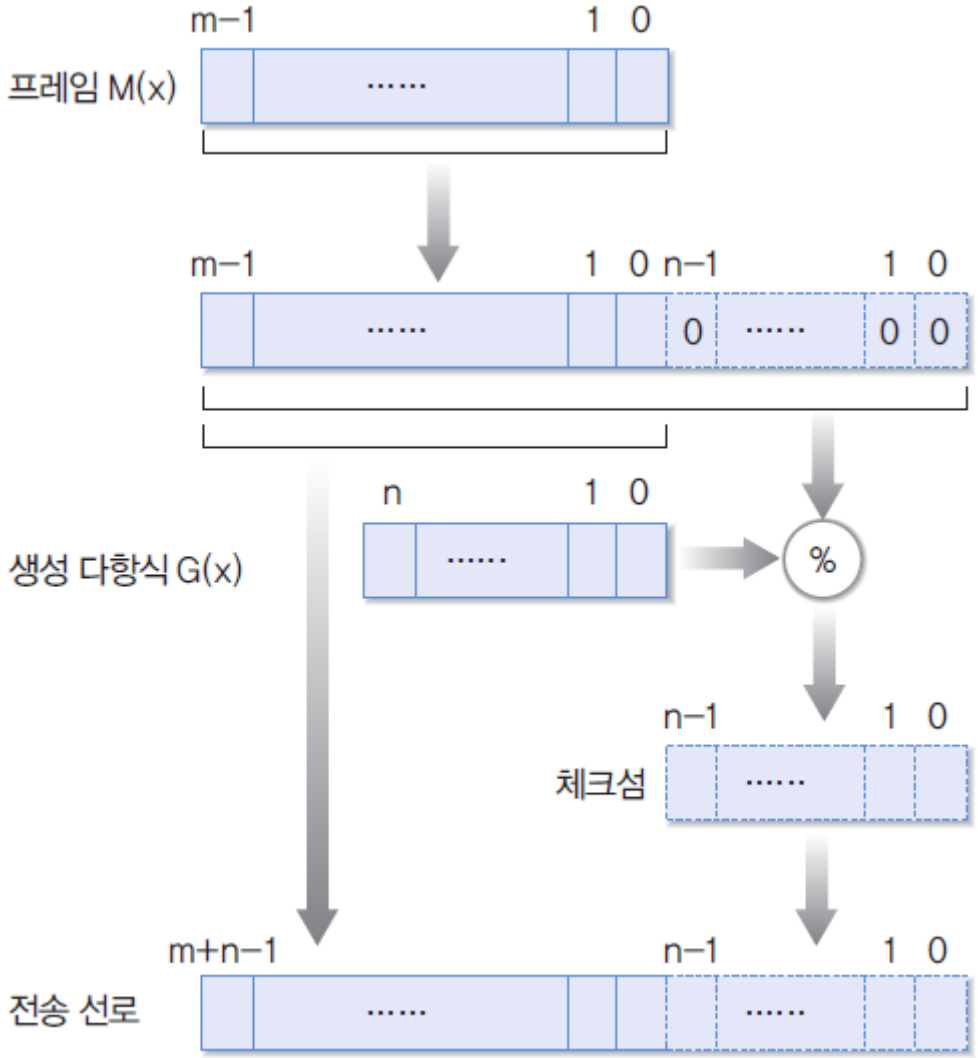


그림 4-19 생성 다항식

04_다항 코드

- 체크섬 계산인 나누기 과정에서 발생하는 다항 연산은 모듈로-2 방식(합을 2로 나눈 나머지를 결과로 하는 연산)

- 덧셈과 뺄셈은 배타적 논리합 연산과 동일한 결과를 얻음

```
  1 0 0 1 1 1 1 0 1 1 0 1
−1 1 0 0 0 1 1 1 0 1 1 0
-----------------------
  0 1 0 1 1 0 0 1 1 0 1 1
```

```
  1 0 0 1 1 1 1 0 1 1 0 1
+1 1 0 0 0 1 1 1 0 1 1 0
-----------------------
  0 1 0 1 1 0 0 1 1 0 1 1
```

- 수신 호스트는 전송 오류가 발생했는지를 판단하기 위해 수신한 m+n비트의 데이터를 생성 다항식 G(x)로 나누는 연산을 수행
  - 나머지가 0 : 전송 오류가 없음
  - 나머지가 0이 아님 : 오류가 발생했다고 판단

04_다항 코드

- 체크섬의 예
  - 생성 다항식 $G(x) = x^5 + x^2 + 1$
  - 전송 데이터 : 101101001
  - 체크섬
    - 계산을 통해 얻은 나머지 00010
  - 송신데이터 : 101101001**00010**

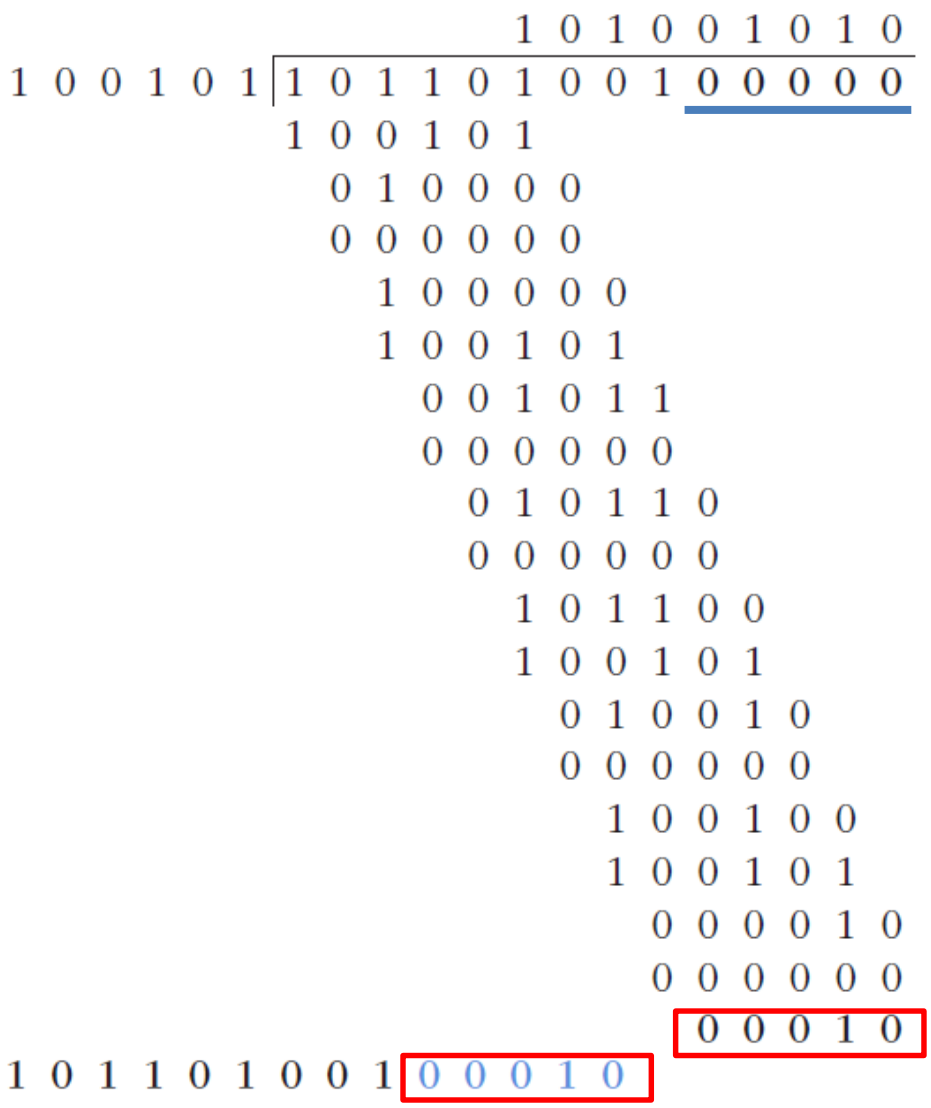
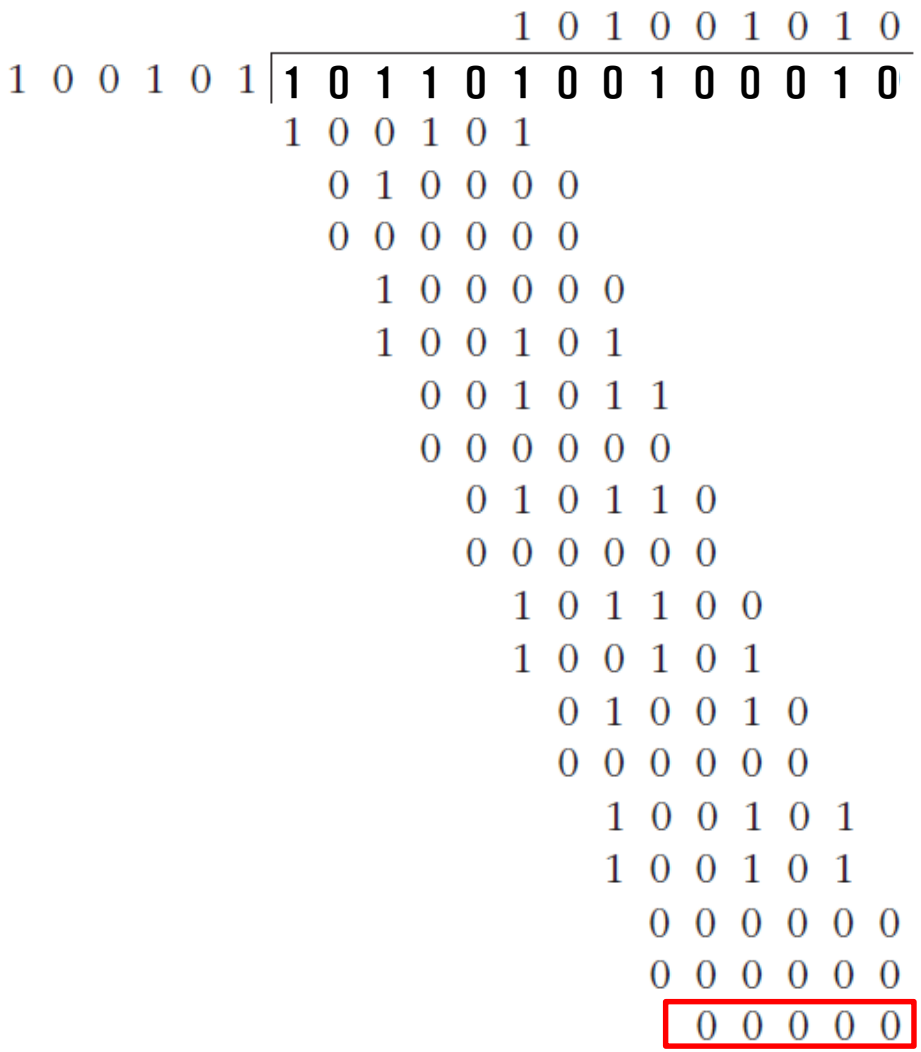


그림 4-20 체크섬 계산의 예

04_다항 코드

- 수신 호스트의 계산
  - 생성 다항식 $G(x) = x^5 + x^2 + 1$
  - 수신 데이터 : 101101001000 10
  - 계산을 통해 얻은 나머지 00000
    => 전송 오류가 없음



04_다항 코드

- 국제 표준으로 이용되는 생성 다항식

  - CRC-12 : $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
  - CRC-16 : $x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT : $x^{16} + x^{12} + x^5 + 1$

04_다항 코드

- [문제1]
  - 생성 다항식 $G(x) = x^3 + x + 1$
  - 전송 데이터 : 1101 0011 1011 00
  - 체크섬 ?
  - 송신데이터 ?

- [문제2]
  - 생성 다항식 $G(x) = x^5 + x^2 + 1$
  - 전송 데이터 : 1101 0011 1011 00
  - 체크섬 ?
  - 송신데이터 ?

[참고] 오류 검출&복구 : 해밍코드

❖ 해밍코드Hamming code

$$H = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix}$$
$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{matrix} \\ r \\ \\ \end{matrix} \begin{matrix} H3 \\ H2 \\ H1 \end{matrix}$$

2^r-1

- 송신 데이터 : 1 0 0 1

H1	H2	D1	H3	D2	D3	D4	
		1		0	0	1	
H1 : H1		1		0		1	0
H2 :	H2	1			0	1	0
H3 :			H3	0	0	1	1

0 0 1 1 0 0 1

- 수신 데이터 : 0 0 1 1 0 1 1

H1	H2	D1	H3	D2	D3	D4	
0	0	1	1	0	1	1	
H1 : 0		1		0		1	O
H2 :	0	1			1	1	X
H3 :			1	0	1	1	X

[참고] 오류 검출&복구 : 해밍코드

해밍코드Hamming code

- [문제1]
  - 송신해야 할 데이터 : 1101
  - 짝수 패리티로 변환하여 송신될 해밍코드는?

- [문제2]
  - 4비트로 구성된 정보비트를 짝수 패리티로 변환하여 송신하고, 수신한 해밍코드 Digit가 1011000이다.
  - ① 오류 발생 여부를 조사
  - ② 만일 오류가 발생하였다면 정정하고(수정한 해밍코드(7비트)를 표현하세요.)
  - ③ 정정하였다면 정정한 후 처음 송신한 정보비트 4비트를 구하세요.

Thank You