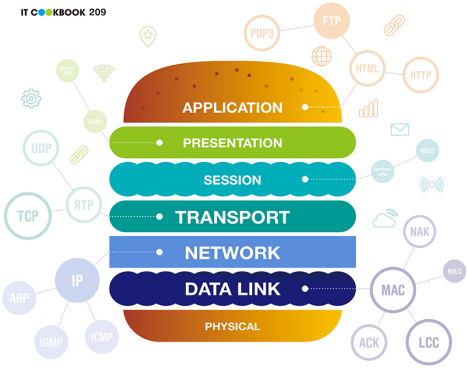


4 데이터 전송의 기초

쉽게 배우는 데이터 통신과 컴퓨터 네트워크(개정판)

한빛아카데미 Hanbit Academy, Inc.

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
  - 요약
  - 연습문제

- **컴퓨터 네트워크 효과**
  - 자원 공유
    - 컴퓨터 하드웨어, 소프트웨어 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
  - 중복 저장으로 신뢰성 향상
    - 중복 저장되므로 데이터 복구가 용이함
    - 신뢰성의 향상 정도만큼 시스템 성능은 저하됨

전송과 교환

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

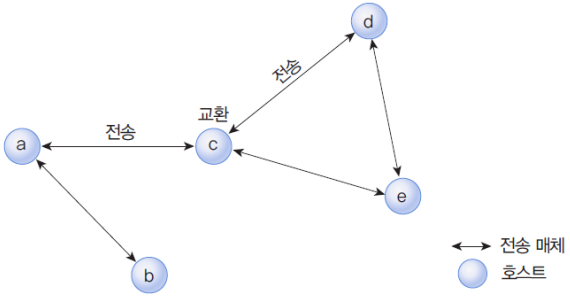


그림 4-1 전송과 교환

- 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
    - LANLocal Area Network(근거리 통신망), MANMetropolitan Area Network(도시규모의 통신망), WANWide Area Network(원거리 통신망) 등
  - 데이터 전송 .교환 기술의 분류 방식
    - 점대점Point-to-Point , 브로드캐스팅Broadcasting 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

- **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

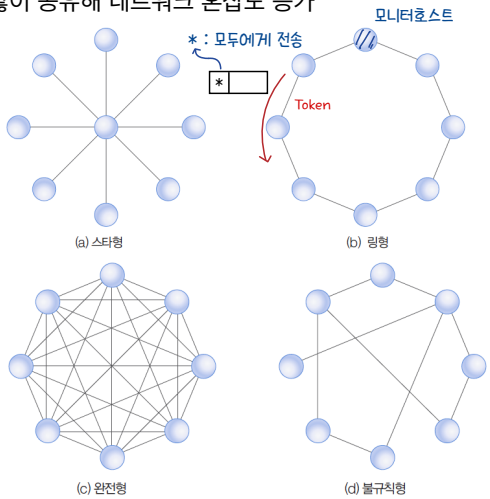
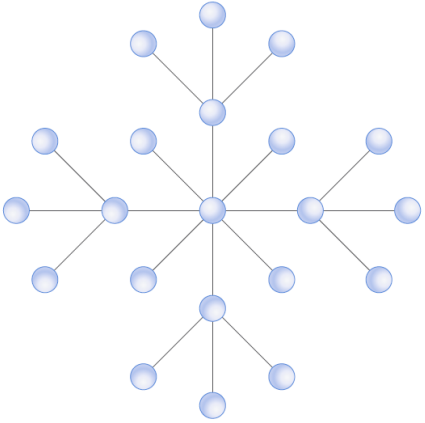


그림 4-2 점대점 방식

01_데이터 전송 방식

- 스타Star형 현재 사용 방식
  - 하나의 중개 호스트 주위로 여러 호스트를 일대일로 연결하는 형태
  - 중앙 호스트의 신뢰성과 성능이 네트워크에 영향 줌
  - 트리Tree형 : 중앙에 있는 스타 구조 주변에 위치한 호스트들을 중심으로 새로운 스타 구조가 확장되는 형태
    - 장단점 : 중개 과정이 간단하나 중앙 호스트에 문제발생시 전체 네트워크의 동작에 영향을 줌

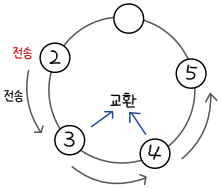


장점: 확장성이 좋다

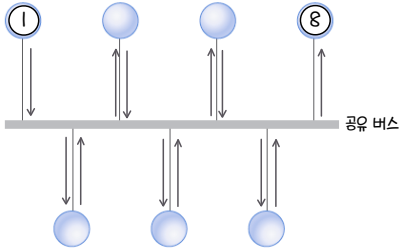
그림 4-3 트리 구조

01_데이터 전송 방식

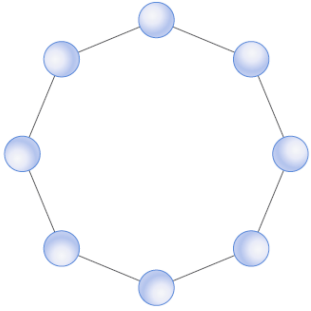
- 링Ring형
  - 호스트의 연결이 순환 고리 구조
  - 모든 호스트가 데이터 전송과 교환 기능을 동시에 수행
  - 토큰Token
    - 호스트 사이의 데이터 송신 시점을 제어하는 기능
    - 데이터의 전송 권한을 의미하는 토큰을 확보
    - 데이터 전송이 완료되면 토큰을 다시 링 네트워크에 돌려줌
  - 단점
    - 한 호스트가 고장나면 전체 네트워크가 동작하지 않을 수 있음
- 완전형
  - 모든 호스트가 다른 모든 호스트와 일대일로 직접 연결하는 방식
  - 단점 : 전송 매체가 증가하면 비용 측면이 비효율적임
- 불규칙형
  - 연결 구조를 특정 패턴으로 분류할 수 없는 방식(일반 네트워크)



- **브로드캐스팅 방식**
  - 특정 호스트가 전송한 데이터가 네트워크에 연결된 모든 호스트에 전달
  - LAN 환경에서 사용(교환 호스트 불필요)
  - 종류 : 버스형, 링형



(a) 버스형



(b) 링형

그림 4-4 브로드캐스팅 방식

01_데이터 전송 방식

충돌 회피 방법
1. CSMA / CD(이더넷)
(공유버스를 썬상(상표브라)) / 충돌감지
: 비어있을 때 보내도, 충돌 감지 해야함
충돌이 생기면 조금 기다렸다가 재전송

2. Token

- 버스Bus형
  - 전송 데이터를 모든 호스트에서 수신할 수 있음
  - 충돌Collision : 둘 이상의 호스트에서 데이터를 동시에 전송할때 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

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

- **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 : 유니캐스팅Unicastin
    - 다수의 수신 호스트 전송 : 멀티캐스팅Multicasting

01_데이터 전송 방식

브로드 캐스팅
: 주소가 특정주소가 아닌 모든 노드를 가리키도록 설정
→ 문제 X

- 멀티포인트 유니캐스팅MultipointUnicasting
  - 일대다 통신을 위해 멀티포인트 유니캐스팅 방식을 사용

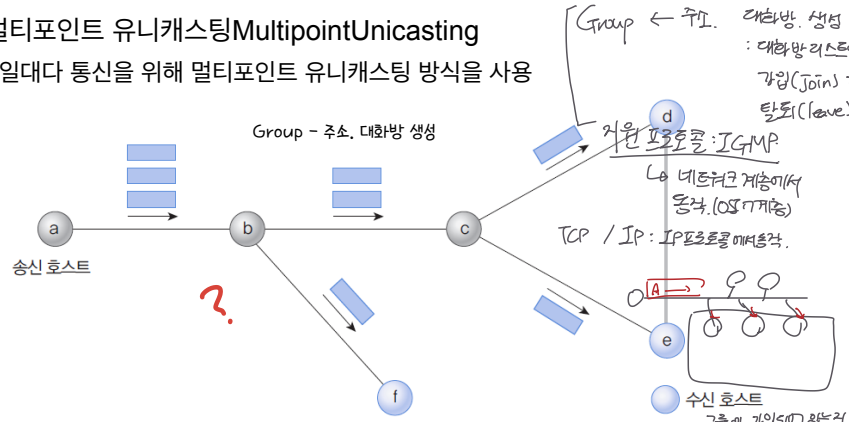


그림 4-5 멀티포인트 유니캐스팅

- 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

- 브로드캐스팅Broadcasting
  - 네트워크에 연결된 모든 호스트에 전송되는 방식
  - 단점 : 호스트 수가 많을수록 네트워크 트래픽이 급격히 증가
    (서브넷Subnet 내에서 이용이 좋음)

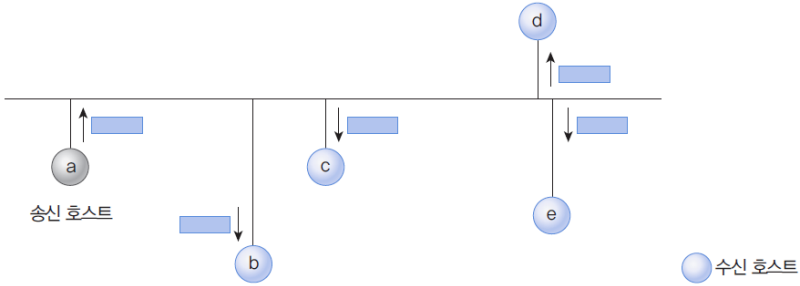


그림 4-6 브로드캐스팅

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

- 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

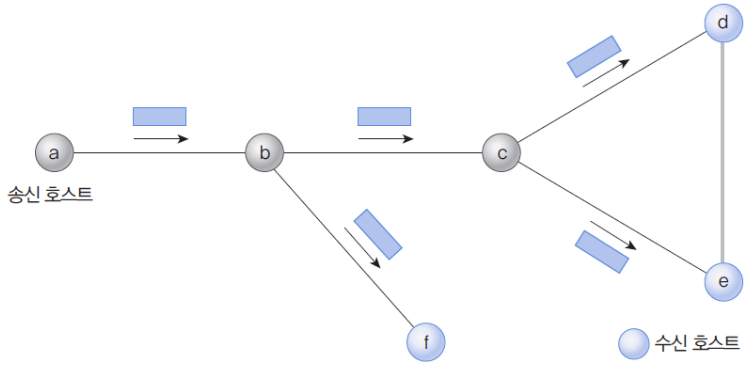


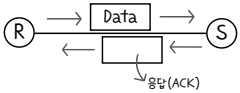
그림 4-7 멀티캐스팅

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



1. 분실 오류 : 응답 x → 재전송 : 송신자가 에러탐지
2. 변형 오류 : 부정응답이 옴 → 재전송: 수신자가 에러탐지

→ 재전송이 빈번하면 혼잡오류 발생 가능성 많음

혼잡에러: 중간에 발생
라우터에 많은 프레임이 몰리면? 혼잡 발생
원인 : 재전송 / 해결 : All Stop(혼잡을 피하는 방법)

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

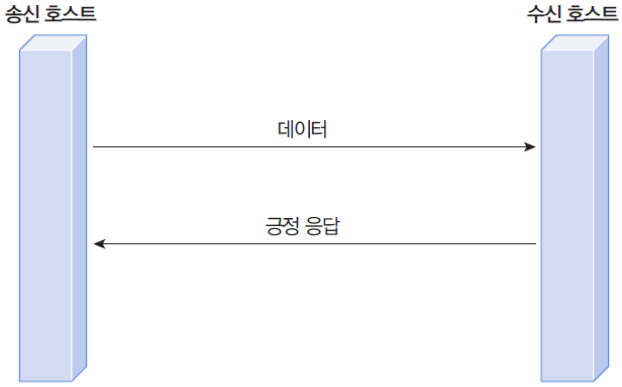


그림 4-8 정상적인 데이터 전송

- **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

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

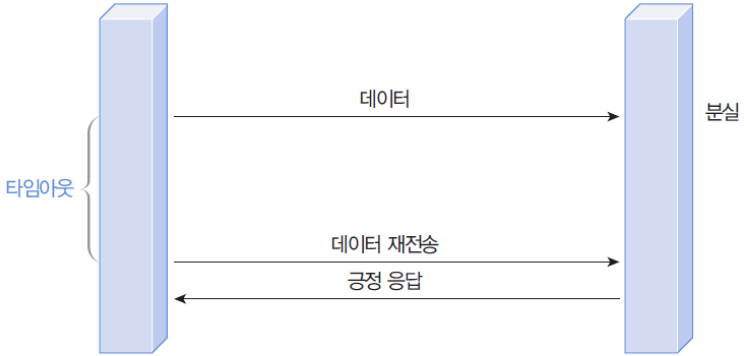
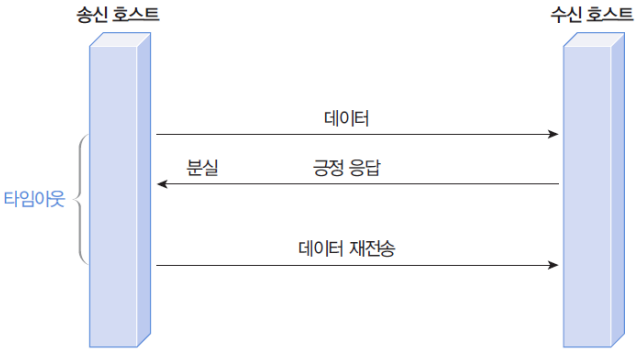


그림 4-10 프레임 분실 오류

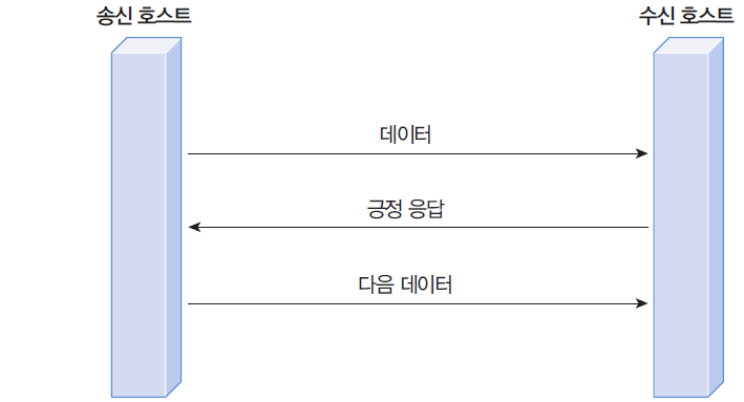
- **순서 번호** Data를 잘게 쪼개서(분할)전송 → 순서 필요(번호 붙이기)
  - 중복 수신 문제를 해결하기 위해 데이터 프레임에게 부여되는 고유 번호
  - 순서 번호의 필요성
    - 긍정 응답 프레임이 사라지는 오류 발생시 송신 호스트의 타임아웃 기능에 따라 재전송 과정이 진행됨 □ 동일한 프레임 중복 수신



(a) 긍정 응답 분실

뒤에서 더 자세히!

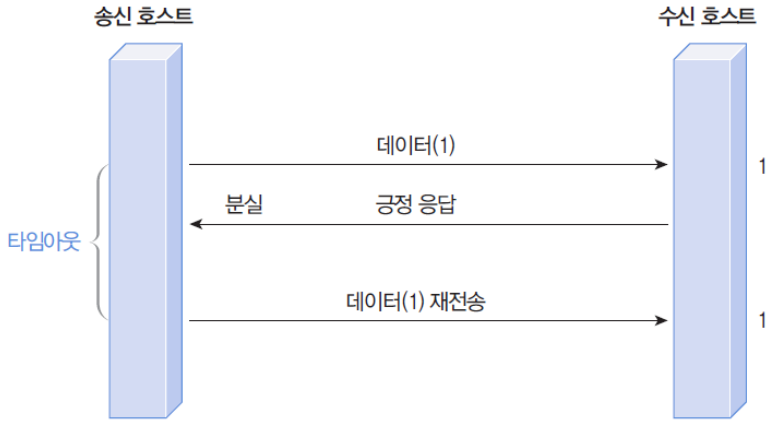
- 수신 호스트가 두 경우((a) 긍정 응답 분실, (b) 긍정 응답 도착)를 구분할 수 있도록 데이터 프레임별로 고유의 순서 번호를 부여하는 방식이 필요함



(b) 긍정 응답 도착

그림 4–11 **순서 번호가 없는 경우**

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



(a) 긍정 응답 분실

- 서로 다른 데이터 프레임이 도착

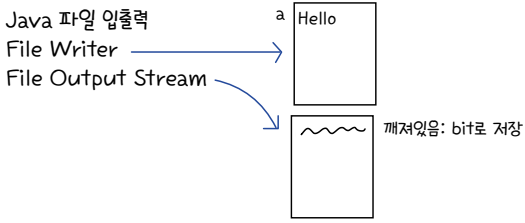
(b) 긍정 응답 도착

그림 4-12 순서 번호가 있는 경우

- **흐름 제어Flow Control** 송신과 수신 사이에 전송속도를 정해서 보내주어야 함
  - 수신 호스트가 감당할 수 있을 정도의 전송 속도를 유지하면서 데이터 프레임을 전송
  - 너무 빨리 전송하는 경우 (동기화)
    - 수신 호스트가 내부 버퍼에 보관하지 못할 수 있음
    - 이는 프레임 분실과 동일한 효과를 야기함 → 재전송 발생
  - 기본 원리
    - 수신 호스트가 송신 호스트의 전송 시점을 제어
    - 대표적인 예: 슬라이딩 윈도우 프로토콜

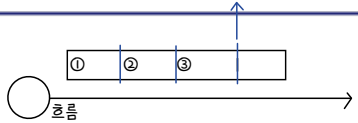
- 데이터 링크 계층 : 전송 데이터를 프레임Frame이라는 단위로 나누어 처리
- 전송프레임 : 체크섬Checksum, 송수신 호스트의 주소, 제어 코드 등 정보포함
- 내부정보를 표현하는 방식에 따라 문자 프레임과 비트 프레임으로 구분

- **문자 프레임Character Frame** 비트프레임 주로사용
  - 내용이 문자로 구성
  - 8비트 단위(또는 ASCII 문자 코드)의 고정 크기로 동작



03_프레임

시작, 끝을 가리키는 프레임 필요
→ 특수문자 사용: 스터핑 기법
스터핑 기법: 시작과 끝을 잘못 인식하는 것을 방지하는 기법



- 프레임의 구조
  - 프레임의 시작과 끝에 특수 문자 사용
    - 시작 : DLE / STX
    - 끝 : DLE / ETX
  - 전송 데이터에 특수 문자가 포함되면 혼선이 발생

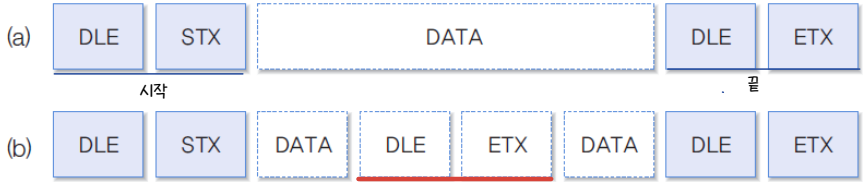


그림 4-13 문자 프레임의 구조

- 문자 스터핑Character Stuffing
  - 문자 프레임의 전송 과정에서 제어 문자를 추가하는 기능

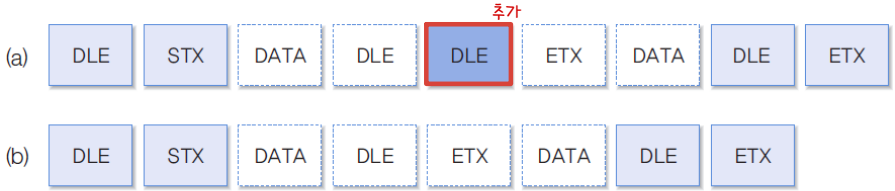


그림 4-14 문자 스터핑

- 전송 데이터가 DLE 문자를 포함하면 뒤에 DLE 문자 하나를 강제 추가
- 데이터에 DLE 문자가 두 번 연속 있으면 하나의 DLE 문자 삭제 → 받는쪽에서

- 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

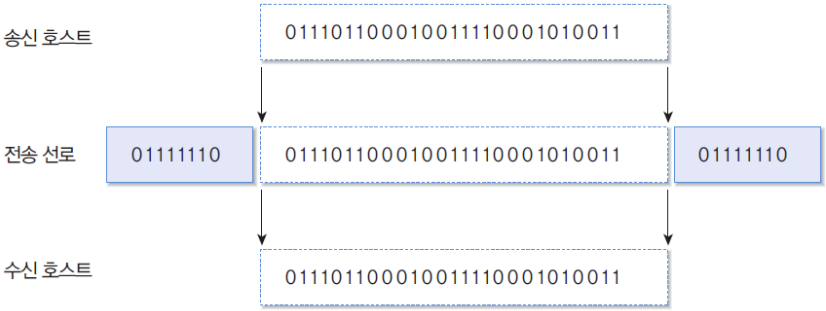


그림 4-15 비트 프레임의 구조

- 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

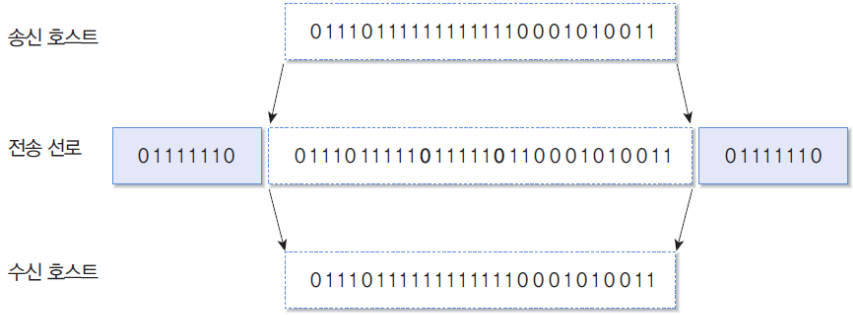


그림 4-16 **비트 스터핑**

- 오류를 극복하는 방법
  - 전송 프레임에 오류 검출 코드를 넣어 수신 호스트가 전송 과정의 오류를 검출
  - 순방향 오류 복구FEC, Forward Error Correction : 오류 복구 코드를 사용해 수신 호스트가

오류 복구 기능을 수행하는 방식

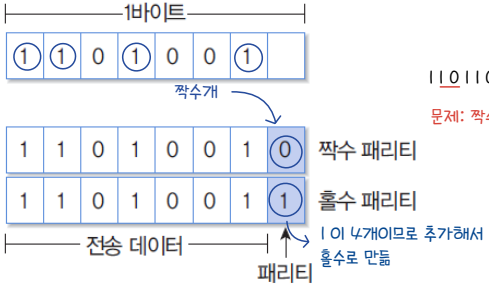
- **오류 검출**
  - 역방향 오류 복구BEC, Backward Error Correction (ARQAutomatic Repeat reQuest 방식)

: 재전송 방식을 이용해 오류 복구(네트워크에서 일반적으로 사용)

ARQ

- 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

오류 검출 기법
① 패리티 기법(짝수, 홀수)
② 블록 검사
③ 다항 코드



1 1 0 1 1 0 **0** → 1 0 1 1 1 0 **0**

문제: 짝수개가 바뀌면 변형이 일어나도 검출X (발견X)

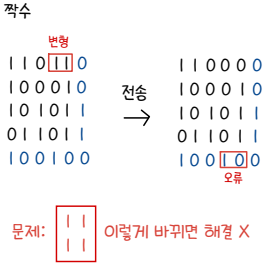
그림 4-17 패리티 비트

- 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

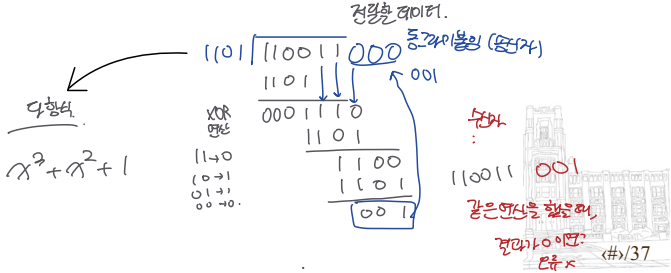
블록 검사

그림 4-18 **블록 검사**



- **다항 코드Polynomial Code (CRCCyclic Redundancy Code)**
  - 생성 다항식
    - 계수가 0과 1인 다항식 형태를 기반
    - 다항코드 100101 = 1×x5 + 0×x4 + 0×x3 + 1×x2 + 0×x1 + 1×x0
      = 생성 다항식 x5 + x2 + 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

다항 코드 BCC 코드 덧붙이기



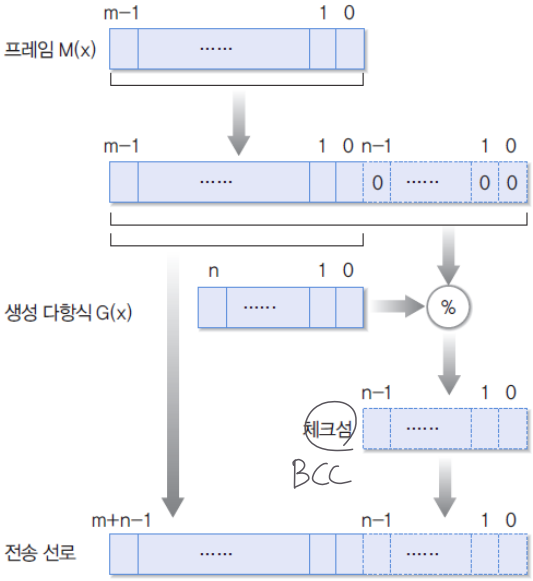

m−1
1 0
프레임 M(x)
.....
m−1
1 0 n−1
1 0
0
.....
0 0
n
1 0
생성 다항식 G(x)
.....
%
n−1
1 0
체크섬
.....
BCC
m+n−1
n−1
1 0
전송 선로
.....
.....

그림 4-19 생성 다항식

04_다항 코드

- 체크섬 계산인 나누기 과정에서 발생하는 다항 연산은 모듈로-2 방식
- 덧셈과 뺄셈은 배타적 논리합 연산과 동일한 결과를 얻음

```
  1 0 0 1 1 1 1 0 1 1 0 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        +1 1 0 0 0 1 1 1 0 1 1 0
-----------------------          -----------------------
  0 1 0 1 1 0 0 1 1 0 1 1          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) = x5 + x2 + 1
  - 전송 데이터 : 101101001
  - 체크섬
    - 계산을 통해 얻은 나머지 00010
  - 송신데이터 : 101101001**00010**

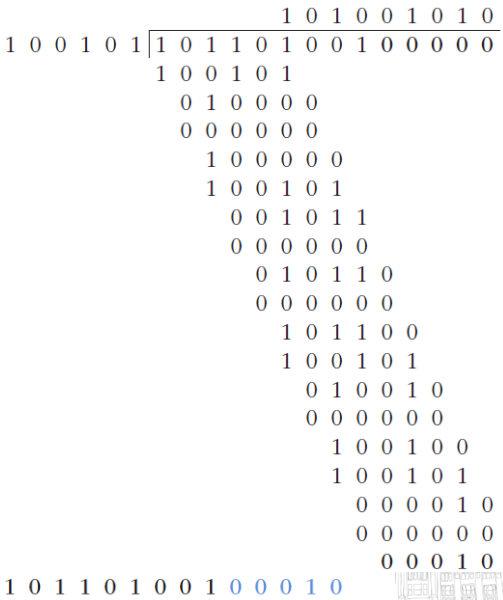


그림 4–20 **체크섬 계산의 예**

- 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 : $x^{16} + x^{12} + x^5 + 1$

Thank You