2018.2.

컴퓨터 네트워킹

VI. 링크 계층: 링크, 접속망, 랜

1. 링크 계층 소개
2. 오류 검출 및 정정 기술
   - (1) 패리티 검사
   - (2) 체크섬 방법
   - (3) 순환중복검사(CRC)
3. 다중 접속 링크와 프로토콜
   - (1) 채널 분할 프로토콜
   - (2) 랜덤 접속 프로토콜
   - (3) 순번 프로토콜

JEJU NATIONAL UNIVERSITY

김예찬 (think.computer@jejunu.ac.kr)

• 링크 계층 소개

Goal :

- 링크 계층에서 사용되는 용어를 설명할 수 있다.
- 링크 계층이 제공하는 서비스를 설명할 수 있다.
- 링크 계층이 구현되는 위치를 설명할 수 있다.
- 링크 계층에서의 캡슐화를 간략하게 설명할 수 있다.

링크 계층에서 사용되는 용어

- 노드

  호스트, 라우터, 스위치, WiFi AP 등
  링크 계층 (layer-2) 프로토콜을 실행하는 모든 장치

- 링크

  통신 경로 상의 인접 노드들을 연결하는 통신 채널
  - 유선
  - 무선 …

- 프레임

  2계층에서의 PDU (데이터그램을 링크 계층 프레임으로 캡슐화한 것.)

트랜스포트/네트워크 계층과의 차이점

- 트랜스포트/네트워크 계층
  종단 간(end-to-end) 연결에 초점을 맞춘 것.

- 링크 계층
  노드 간(node-to-node) 연결에 초점을 맞춘 것.
  인접한 두 노드 간에 데이터그램 전송에 초점을 맞춘 기능을 제공한다.

링크 계층이 제공하는 서비스

- [핵심] 단일 통신 링크 상으로
  데이터그램을 한 노드에서 인접 노드로 이동시키는 것.

- 프레임화 : 링크 계층 프레임으로 캡슐화.
- 흐름 제어(Flow Control)

- MAC(Media Access Control) 프로토콜
- 전이중(Full-duplex), 반이중(Half-duplex).

- 신뢰적 전달 서비스
  - 무선 링크(= 오류율 ↑ 링크) : 매우 중요.
  - 유선 링크(= 오류율 ↓ 링크) : 불필요한 오버헤드 … 신뢰적 서비스 제공 X.
  - 트랜스포트 계층과의 차이점 : '재전송'이 아닌 '오류 정정'으로 신뢰성 확보.

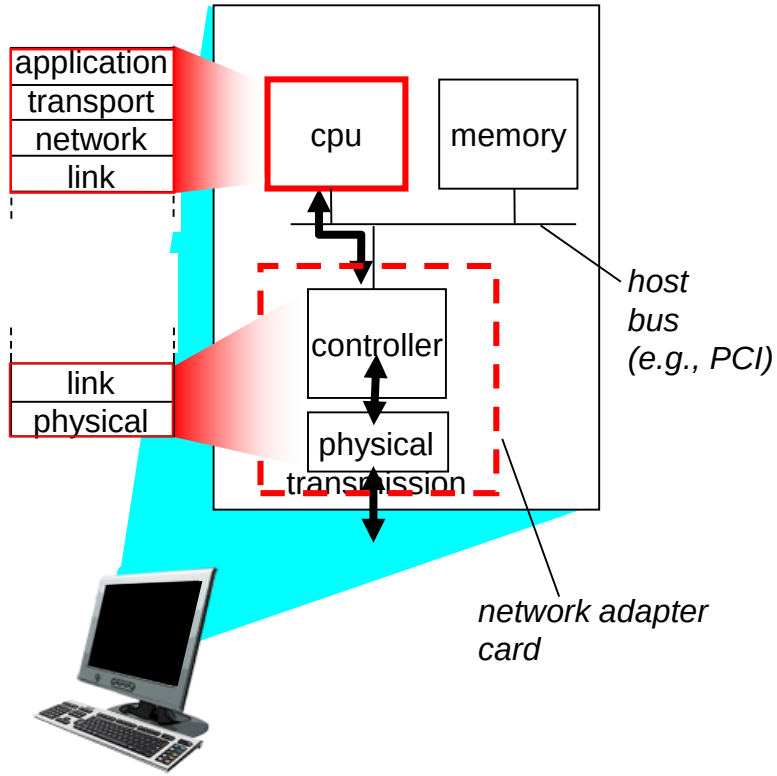
링크 계층이 구현되는 위치

- 모든 호스트에 링크 계층은 있다.

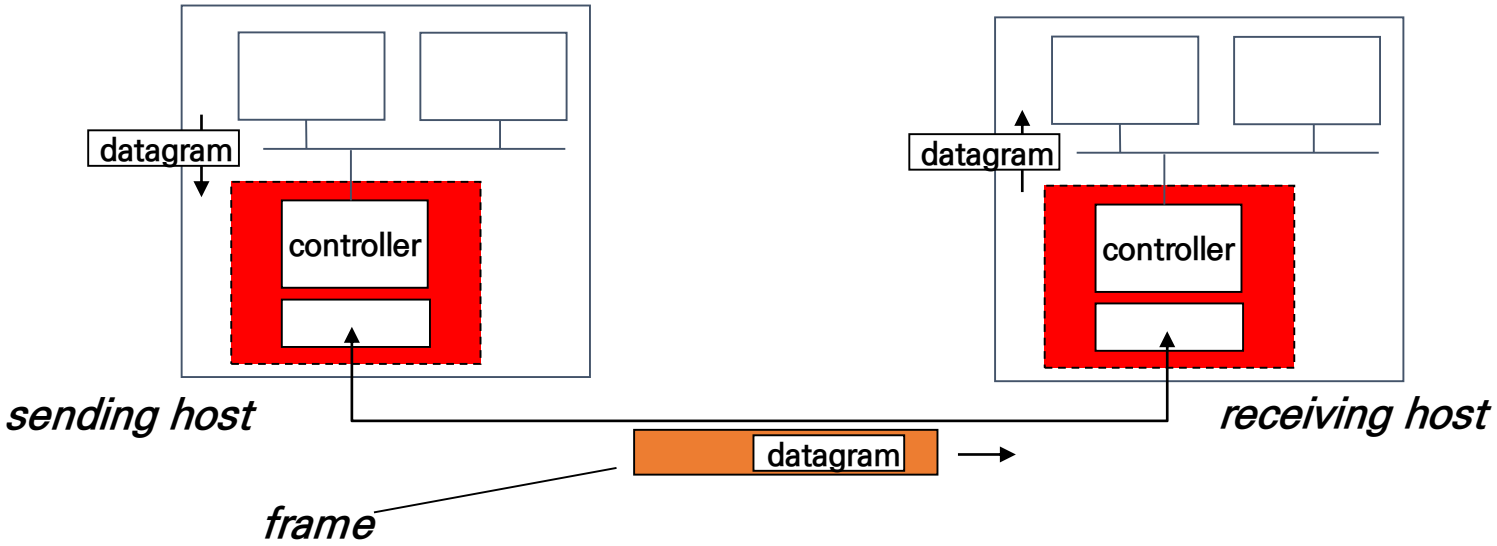
- 링크 계층은 HW적으로 구현된다.

- 링크 계층은 NIC(Network Interface Card) 어댑터/칩(마더보드 내장형) 등에 내장된다.
  - 이더넷 카드
  - 802.11 카드

- 한편 NIC는 호스트의 버스 시스템에 부착됨.
  - 펌웨어(Firmware), 하드웨어(HW)의 결합.
  - 펌웨어 : 캡슐화 / 역캡슐화 등 제공.



링크 계층에서의 캡슐화 개관



- 송신 측:
  - 데이터그램을 프레임에 캡슐화
  - 에러 체크 비트, 흐름 제어 등을 위한 기능 추가
- 수신 측
  - 프레임에서 데이터그램 추출 후 상위 계층으로 전달

• 오류 검출 및 정정 기술

Goal :

- 비트 수준 오류 검출과 정정의 한계를 설명할 수 있다.
- 패리티 검사의 의의와 한계를 설명할 수 있다.
- 체크섬 검사의 의의와 한계를 설명할 수 있다.
- CRC의 의의와 한계를 설명할 수 있다.

링크 계층에서 사용되는 용어

- 핵심
  - 에러 감지 / 에러 정정
    ⇒100% 신뢰 가능한 수준은 아니다.

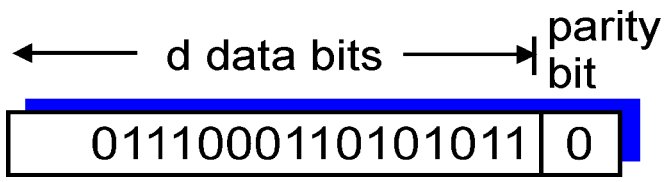
- 어떻게 하면 틀릴 확률을 줄일 수 있을지가 관건.
  - 이는 많은 오버헤드를 수반.

- 패리티(Parity)
- 체크섬(Checksum)
- 순환중복검사(CRC)

패리티 검사(Parity Checking)

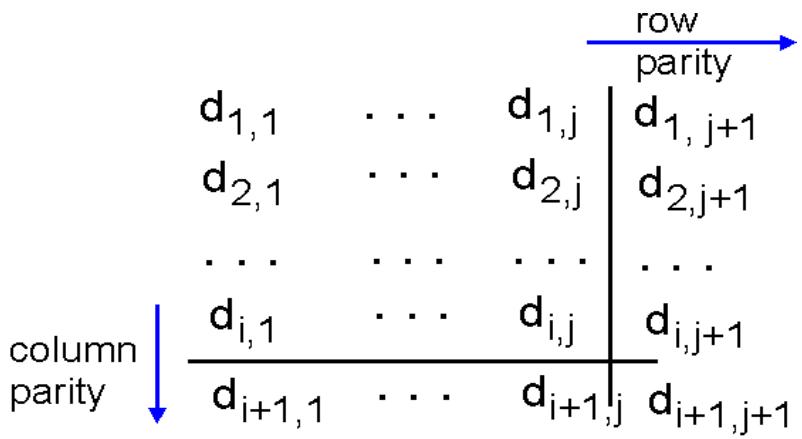
single bit parity:

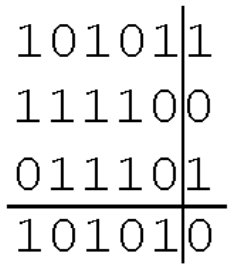
- *d*etect single bit errors



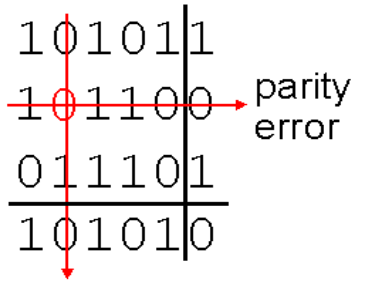
two-dimensional bit parity:

- detect and correct single bit errors
- detect double bit errors (but that case, cannot correct)





no errors



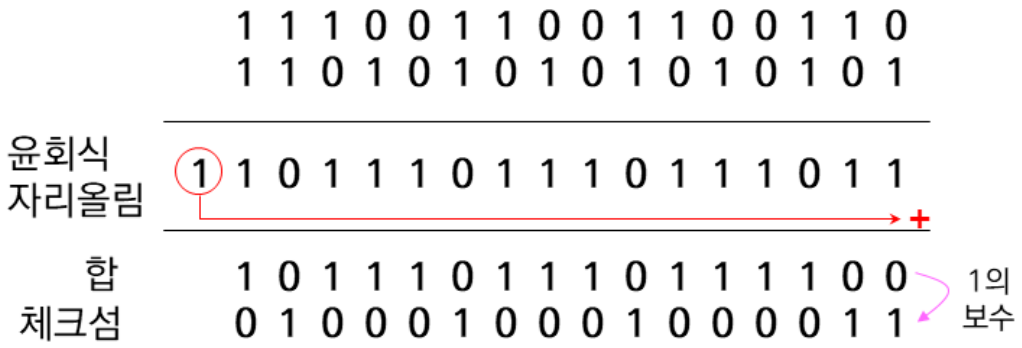
correctable single bit error

한계 :
1bit error에 대한 검출만 가능하다.

체크섬(Checksum) 방법

- 트랜스포트 계층에서만 사용하는 방법
- 패킷에서의 에러를 검출 가능. (여러 flipped bit error에 대해서도)

- 알고리즘 (약식)
  - 16bit 워드 단위로 헤더 필드와 데이터 필드를 더한다.
  - 이때 가산 과정에서 발생하는 오버플로우는 Wrap-around 처리한다.
    (즉, MSB비트로부터의 carry-out 값을 합산한다.)
  - 최종적으로 모든 가산의 결과의 1의 보수를 취한 것을 체크섬 필드에 넣는다.



체크섬(Checksum) 방법

- 장점
  - 오버헤드가 작다. (16bit 사용)
  - 구현이 용이하다. (SW적으로)
  - 간단하고 빠르다.

- 한계
  - 오류 검출(Error Detection) 정확도가 떨어진다.

순환중복검사(Cyclic Redundancy Check, CRC)

- 오늘날 네트워크에서 널리 사용되는 오류 검출 기술. ★
  - Checksum보다 더 강력한 오류 검출 가능한 방법.

- 기본적인 개요 (*v 표시한 것은 개선할 계획.*)
  - 데이터(D)를 전송하고자 한다.
  - 송신자와 수신자는 G로 표기되는 생성자(Generator)로 알려진 비트 패턴(r+1)에 대해서 합의한다.
    - G의 최상위 비트는 반드시 1이어야 한다.
  - D를 G로 나눈 나머지 값을 R(Remainder)이 CRC비트가 된다. v
    - 이 CRC비트는 패킷의 꼬리부(Trailer)에 부착된다.
    - 이 상태로 캡슐화된 패킷이 수신자에게 전달된다.

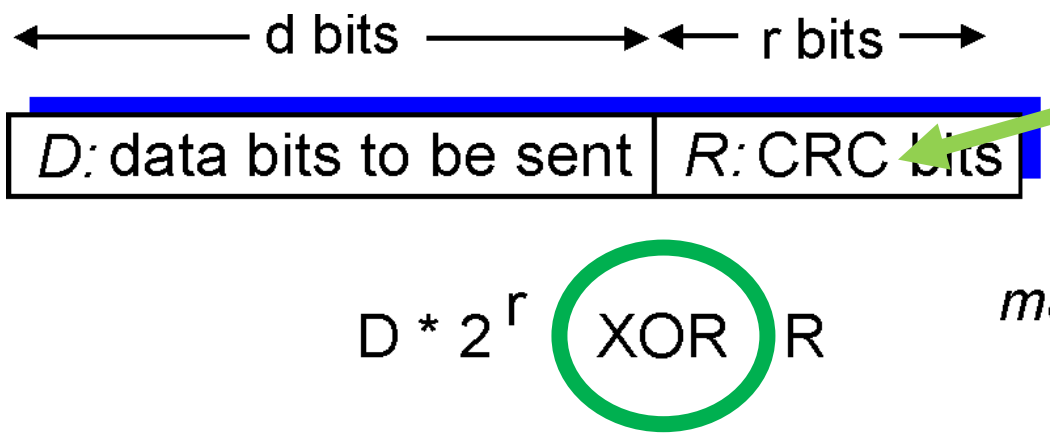
송신자 action

  - 수신자는 패킷을 생성자(합의)로 나누어 나머지가 0(정상)인지 확인한다.

순환중복검사(Cyclic Redundancy Check, CRC)

- 생각해볼 점
  - CPU는 '가산' 회로만 있다. ('감산' 회로는 X.)
    - 나눗셈에서 필요한 '뺄셈'을 XOR로 대체하자.

```
    1011          1011
-   0101      XOR 0101
------        ------
    1110          1110
```



Redundancy (부가적 정보)를 덧붙였다.

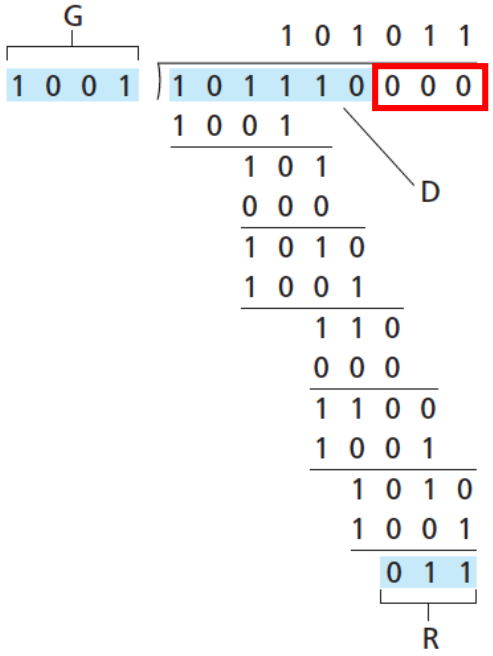
순환중복검사(Cyclic Redundancy Check, CRC)

- 우리가 반드시 할 수 있어야 하는 일
  - $R = \frac{D \cdot 2^r}{G}$의 나머지를 구하자.

- 예제 : 다음 조건일 때 CRC로 사용 가능한 R을 구하시오.

  D = 101110

  G = 1001



G가 r+1비트일 때,
D를 r비트만큼 확장해야 한다.

r+1 비트 미만의
burst error(연집에러)를 모두
감지할 수 있다.

참고 :
Random error(산집에러)

순환중복검사(Cyclic Redundancy Check, CRC)

[제언]

- 오류 검출 확률을 높이기 위해서는 나누는 수 R이 소수(prime number)이자 큰 수여야 한다.

- 국제 표준으로 8bit, 12bit, 16bit, 32bit 생성자 G가 정의되어 있다.

- 참고 : https://ko.wikipedia.org/wiki/%EC%88%9C%ED%99%98_%EC%A4%91%EB%B3%B5_%EA%B2%80%EC%82%AC

순환중복검사(Cyclic Redundancy Check, CRC)

[제언]

- CRC 코드는 다항식 코드(Polynomial code)로도 알려졌는데, 이는 전송되는 비트열에 있는 0과 1 값을 계수로 갖는 다항식처럼 비트열을 생각할 수 있기 때문이다.
  - 또, 비트열에 적용되는 연산을 다항식 연산으로 이해하는 것도 가능하다.

생성자 (G) : 1 0 0 1 1

x^4 x^3 x^2 x^1 x^0

$$G(x) = x^4 + x + 1$$

생성자 ⇔ 생성 다항식

그렇다면 오류 정정은?

- FEC(Forward Error Correction, 순방향 오류 정정)

  송신 측이 전송할 문자나 프레임에 부가적 정보(☞ Redundancy)를
  첨가하여 전송하고,
  수신 측이 에러를 발견 시 이 부가적 정보로 에러 검출 및 에러 정정을 하는 방식

  - 이점
    - 송신자에게 요구되는 재전송 횟수를 줄일 수 있다는 점.
    - 실시간 처리에 있어 유리. (재전송 관련 지연 시간 감소)

  비교 : BEC(역방향 오류 정정) : 송신 측에 재전송(ARQ) 요구하는 방식.

  FEC에 관한 자세한 사항은 본 강의에서 논하지 않는다.

Goal :

- 다중 접속 링크에서 고려해야 할 최대 관건이 충돌이라는 것을 안다.
- 브로드캐스트 채널의 충돌 문제를 해결하기 위한 다중 접속 프로토콜 (MAC Protocol)의 종류를 열거할 수 있다.
- 서로 다른 다중 접속 프로토콜을 비교하여 설명할 수 있다.

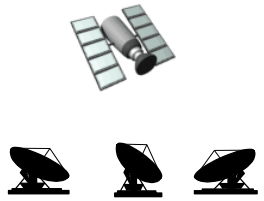
링크의 두 종류

- Point-to-point(점대점) 링크
  - PPP(Point-to-Point Protocol)
  - HDLC(High-level Data-Link Control)
  - Ethernet Switch와 각 호스트 간에 점대점 연결 …

- Broadcast(브로드캐스트) 링크
  - 기존 Ethernet, HFC, 802.11 무선 LAN …

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

<u>2개 이상의 노드가 동시에 프레임을 전송할 경우?</u>

각 수신자에서 전송된 프레임들이 충돌한다.

충돌로 인해 프레임이 손상된다.

대역폭이 낭비된 꼴...!?

다중 접속 프로토콜(Multiple Access Protocol)

- 다수의 노드가 활성화되어 있을 때,
  브로드캐스트 채널이 제대로 보장되도록 보장하기 위해서는?
  - 동시 접속을 분산시키기 위한 알고리즘을 고안해야 한다.
  - 이것이 다중 접속 프로토콜의 책임이다.

- 다중 접속 프로토콜의 유형
  - 채널 분할 프로토콜(Channel Partitioning Protocol)
    - Low-Efficiency 우려.
  - 랜덤 접속 프로토콜(Random Access Protocol)
    - Collision 우려.
  - 순번 프로토콜(Taking-turns Protocol)
    - Collision도 우려되지 않고, 채널 분할에 비해 높은 효율을 기대할 수 있다.

다중 접속 프로토콜(Multiple Access Protocol)

- 이들 프로토콜이 갖춰야 할 공통된 특징
  1. 단 하나의 노드가 전송할 데이터가 있을 때는 그 노드가 R bps의 처리율을 갖는다.
  2. 만일 M개의 노드가 전송하고자 할 때, 각 노드는 R/M bps의 처리율을 갖는다.
     -> 공정성 관련.
  3. fully decentralized:
     - no special node to coordinate transmissions
     - no synchronization of clocks, slots

     -> 누군가의 중앙 제어 시도(원시안적 관리)는 지연 오버헤드를 야기한다.

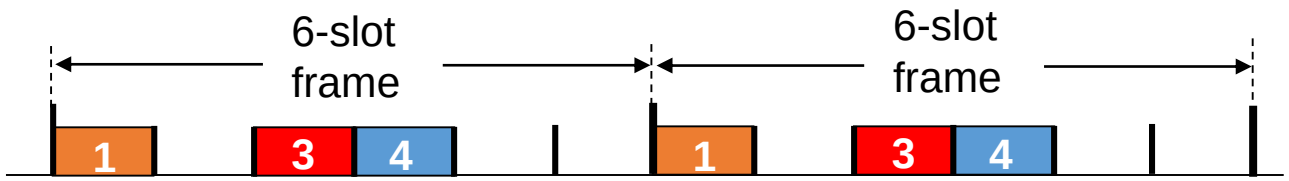
     -> 그냥 개별 노드가 근시안적으로 알아서 정책을 결정하는 것이 낫다.
  4. 구현의 단순함.

     -> 너무 복잡하면, 오히려 독이 된다.

* 다중 접속 프로토콜은 MAC 프로토콜로도 불린다.

(1) 채널 분할 프로토콜

- TDMA(Time Division Multiple Access)
  - 시간을 Time Frame으로 나누고,
  - 각 시간 프레임을 N개의 Time Slot으로 나눈다.



노드가 6개의 TDM에 대한 간단한 예 :
=〉 2,5,6 노드는 idle 상태이다.

- Slot : 한 패킷(2-layer's)이 전송되는 시간

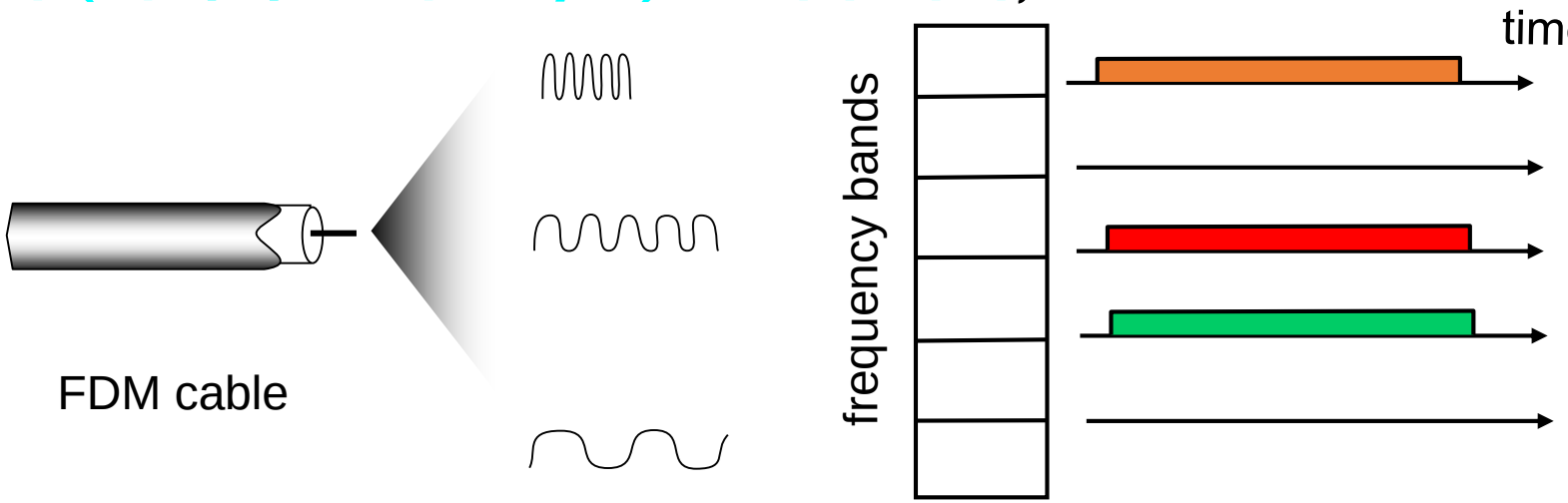
- 장점
  - 충돌이 발생하지 않는다.
  - 공정하다.

- 단점
  - 전송할 패킷이 있는 노드가 단 1개인 경우에도 노드 전송률은 R/N으로 제한됨.
  - 노드가 전송 순서 상 자신의 차례를 항상 기다려야 한다.

(1) 채널 분할 프로토콜

- FDMA(Frequency Division Multiple Access)
  - R bps의 채널의 다른 주파수(대역폭 크기 : R/N)로 나누어서,
  - 각 노드에 할당한다.

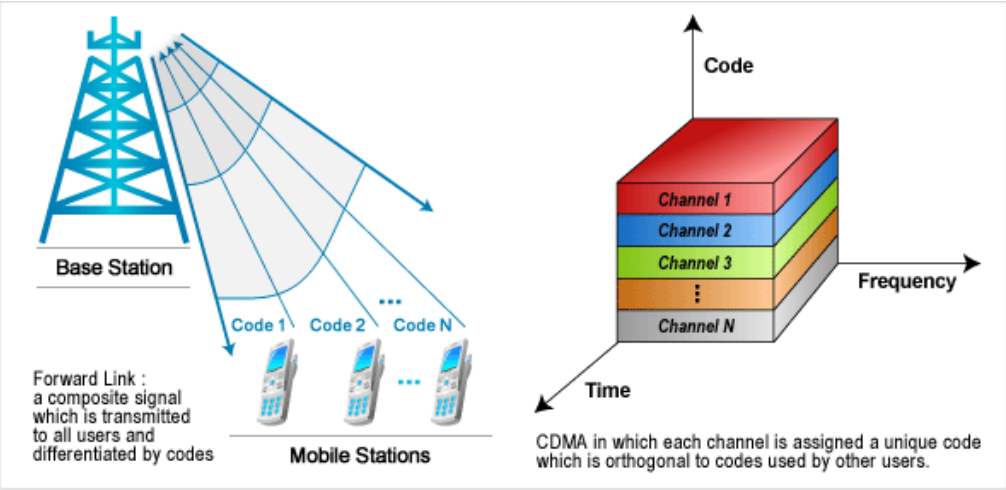
노드가 6개의 FDM에 대한 간단한 예 :
=〉 2,5,6 노드는 idle 상태이다.



- 장점
  - 충돌이 발생하지 않는다.
  - 공정하다.
- 단점
  - 전송할 패킷이 있는 노드가 단 1개인 경우에도 노드 전송률은 R/N으로 제한됨.
  - TDM과 달리, 대역폭이 낭비된다. (∵ Guard-band 기반 주파 간섭 방지)

(1) 채널 분할 프로토콜

- CDMA(Code Division Multiple Access)
  - 앞서 FDMA, TDMA와 비슷한 형태로 명칭을 짓다 보니 이런 용어가 탄생.
    - '코드 분할?' 보다는 '암호화'가 여기선 더 중요한 이슈임.
  - 각 채널 별로 유일한 Code(암호화 방법)를 갖는데,
    이것을 각 호스트가 할당 받아 사용함.
  - FDMA, TDMA보다 개선됨. (간섭에 따른 손상 문제 X, 동시 전송 가능 O)
  - 7장에서 자세하게 다룰 예정.



(2) 랜덤 접속 프로토콜

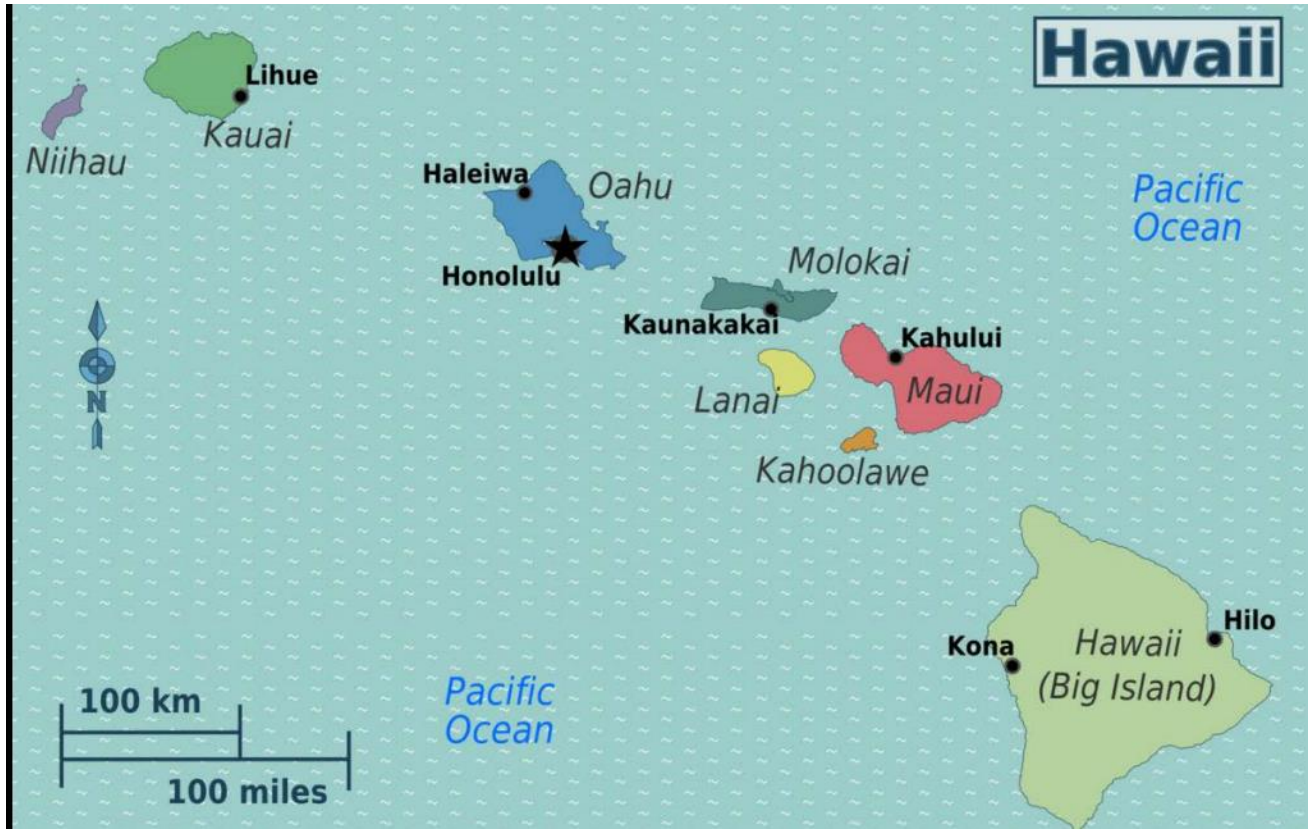
- 개요
  - 노드가 패킷을 전송할 때, 항상 채널의 최대 전송률인 R bps로 전송한다.

  - 하지만, 2개 이상의 노드가 동시에 송신할 경우, "충돌"이 발생한다. 따라서…
    1) 어떻게 충돌(Collision)을 감지(Detect)하여,
    2) 어떻게 이를 복구(Recover)할 것인지를 고려해야 한다.
       * 각 노드 별로 재전송 시간을 임의로(Randomly) 설정하여,
         재전송하면 충돌이 다시 발생할 가능성이 낮을 것이다.

  - 랜덤 접속 프로토콜 종류
    - Slotted ALOHA
    - ALOHA
    - CSMA, CSMA/CD, CSMA/CA

(2) 랜덤 접속 프로토콜

- ALOHA



유선으로 연결하기 어려운 지리적 구조.

무선을 이용한 연결 필요성 대두.

ALOHA 프로토콜은 1971년도에 등장.
-> 그 당시엔 복잡한 프로토콜 구현 곤란.

따라서 매우 단순하고 간단한 알고리즘.

(2) 랜덤 접속 프로토콜

- Slotted ALOHA : 가장 단순한 프로토콜 중 하나.
  - 가정
    - 모든 프레임은 고정 크기이다.
    - 시간은 L/R초의 슬롯들로 나뉜다.
    - 노드는 슬롯의 시작점에서만 프레임 전송이 가능하다.
    - 각 노드는 언제 슬롯이 시작하는지 알 수 있게끔 동기화되어 있다.
    - 각 노드는 준비되는 즉시, 프레임을 전송한다. v
    - 만약 한 슬롯에서 2개 이상의 프레임이 충돌하면,
      모든 노드는 그 슬롯이 끝나기 전에 충돌 발생을 알게 된다.

(2) 랜덤 접속 프로토콜

- Slotted ALOHA
  - 동작 원리
    - 노드는 전송할 새 프레임이 있으면,
      다음 슬롯이 시작할 때까지 기다렸다가
      그 슬롯에 전체 프레임을 전송한다.

Y

"아하! 신뢰적 전송에 실패했구나!"
확률 p로 해당 프레임을 다음 슬롯에서 재전송한다.
(신뢰적 송신에 성공할 때까지)

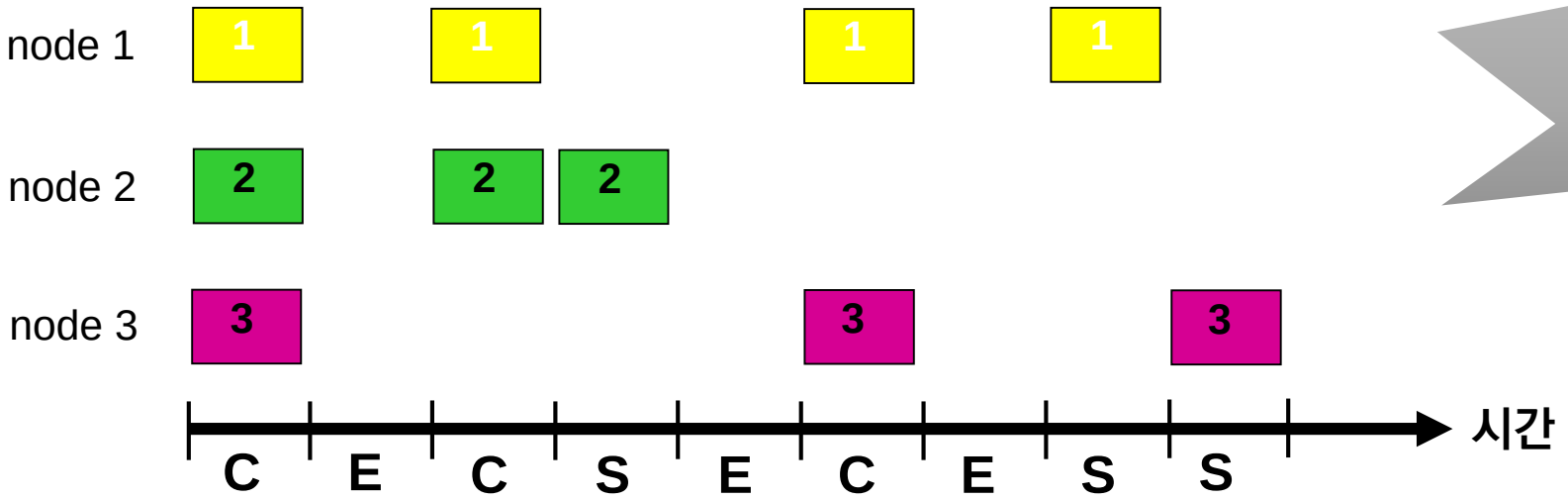
N

"우와! 신뢰적 전송에 성공했어!!! :) "
(전송할 것이 있다면 노드는 새 다음 프레임을 준비한다)

노드는 슬롯이 끝나기 전에
충돌 검출이 가능.

(2) 랜덤 접속 프로토콜

- Slotted ALOHA
  - 예시

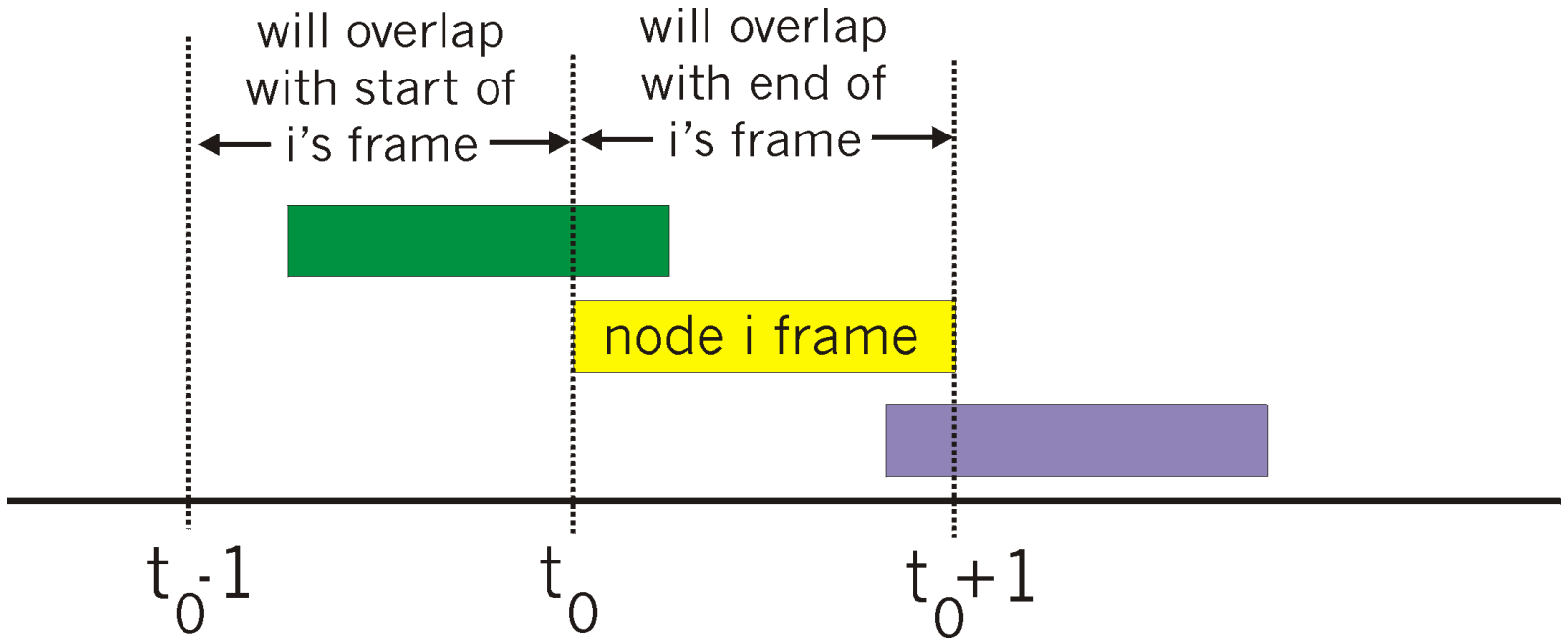


효율성 :
약 37%의
Utilization

- 장점
  - 한 노드가 채널의 최대 대역폭을 사용하여 프레임을 전송한다.
  - 고도의 분산화 :
    - 각 노드가 충돌을 감지하고, 언제 재전송할지를 각자 결정한다.
- 단점
  - 여러 노드가 활성일 경우, 잦은 충돌로 인해 오히려 슬롯이 낭비된다.
  - 확률적 전송 정책 → 유휴(idle) 슬롯.
  - 노드는 슬롯을 동기화시켜야 한다.

(2) 랜덤 접속 프로토콜

- Pure ALOHA = Unslotted ALOHA
  - 더 단순하며, 동기화 개념 X.
  - 성능 효율성이 Slotted에 비해 떨어진다.



효율성 :
약 18%의
Utilization