Media Access Control(MAC)

- 고려할점(사람의 의사소통)
    - 모두에게 말할 기회를 줌
    - 질문이 있으면 손을 듬
    - 권한이 주어질 때까지 말하지 않음
    - 누군가 말할때 방해하지 않음
    - 대화를 독점하지 않음
    - 다른 누군가가 얘기할때 잠들지 않음

Random Access

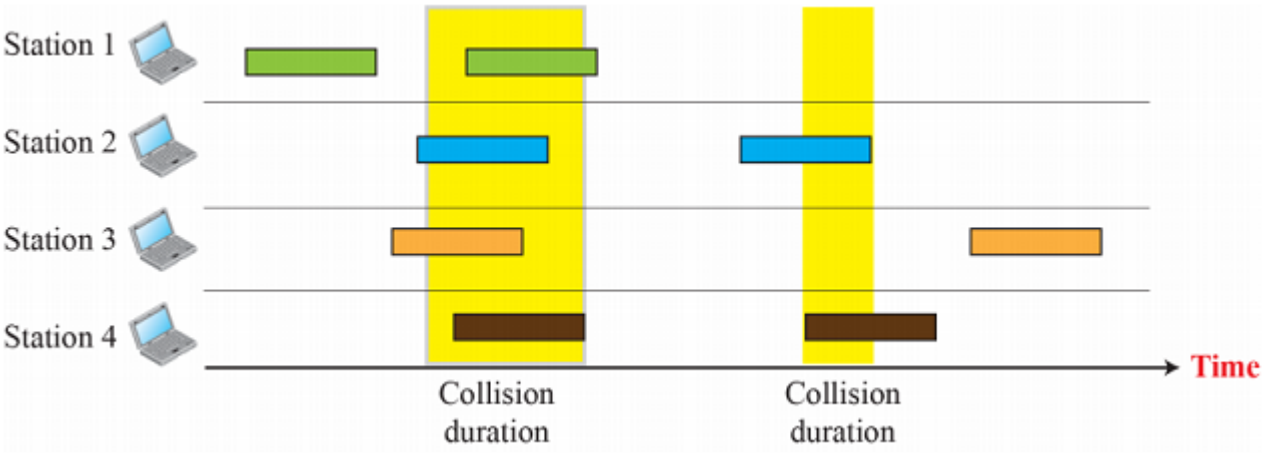
- 한 스테이션이 다른 스테이션보다 우위에 있거나하지 않음
    - 한 스테이션이 다른 스테이션을 컨트롤 못함
- 어떤 스테이션이 언제 보낼지 정해진 시간이 없음
- **Contention**(논쟁)
    - 어느 스테이션이 다음에 보낼지에 대한 규칙이 없음
    - 스테이션들이 서로 경쟁해서 Medium에 접근권한을 얻음
- 각 인스턴스에서, 보낼 데이터가 있는 스테이션이 프로콜이 정의한 프로시져를 사용해 보낼지 안보낼지를 정함 -> 이 결정은 Medium의 상태에따라 달라짐(idle or busy)
- 충돌 : 두개 이상의 스테이션이 데이터 전송을 시도 할 경우, 접근 충돌이 발생해서 프레임이 제거되거나 수정됨

ALOHA

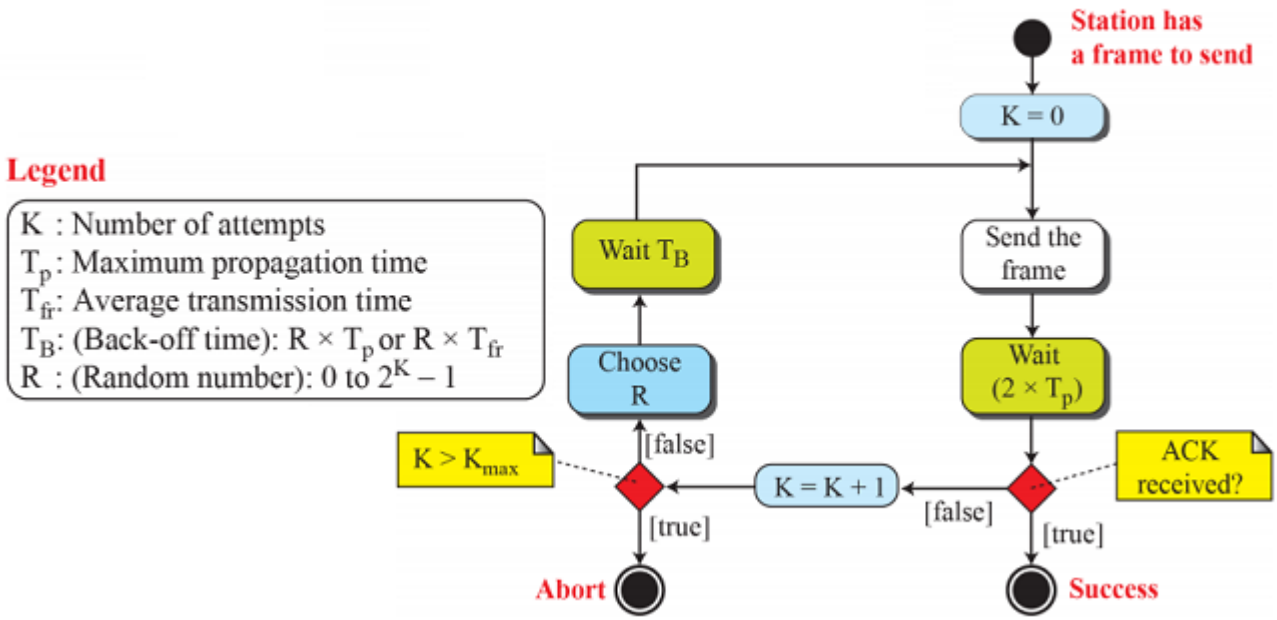
- 가장 오래된 Random Access 방법 중 하나
- 라디오 LAN을 위해 설계됐지만, 아무 공유된 Medium에서나 사용 가능
- 잠재적인 **충돌 가능성이 존재함**
- 스테이션들 사이에 Medium이 공유됨, 한 스테이션이 데이터를 보낼 때 다른 스테이션도 보내기를 시도할 수 있음. 두 데이터가 충돌하고 혼동이 있을 수 있음

Pure ALOHA

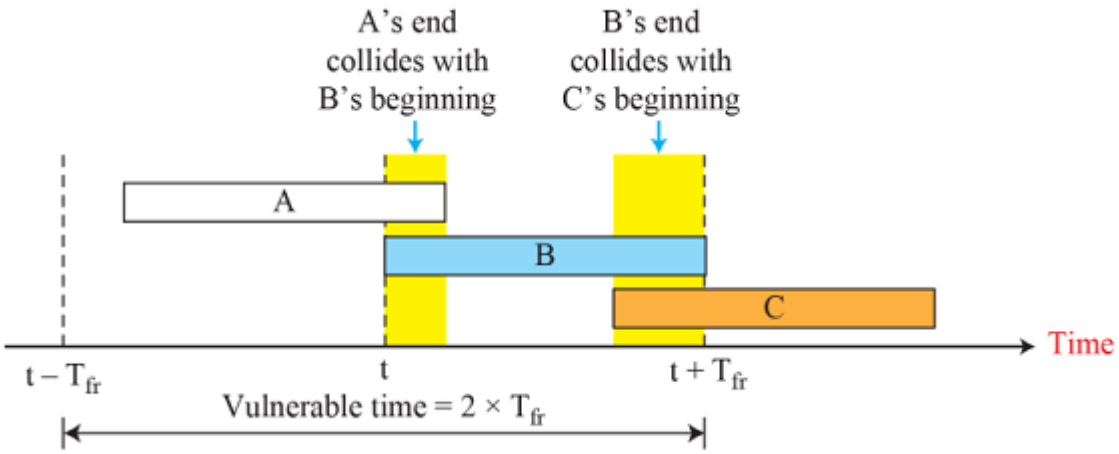
- 보낼 프레임이 생길 때 마다 각각의 스테이션이 프레임을 전송함
- 하나의 채널만 있기 때문에, 충돌 가능성이 있음
- Main Idea
    - Sender가 보낼 프레임이 있으면 보내고, Receiver로 부터 ACK를 받기를 기대
    - ACK를 받으면, 통신 끝
    - ACK를 못받으면, 타임아웃 발생, 프레임이 망가졌다 생각하고 재송신



- 문제점 : 2개 이상의 스테이션이 충돌할 가능성, 해당 스테이션들이 계속해서 재송신을 동시에 시도할 가능성이 있음 -> 재충돌
    - 해결법 : Random Backoff
        - 타임아웃이 발생하면, 각 스테이션이 랜덤길이만큼의 시간동안 기다렸다가 재송신
        - 충돌이 줄어듬
        - 기다리는 랜덤 시간 = Random Backoff Time T_B
        - 재송신되는 프레임들 때문에 채널이 혼잡해지는 것을 방지하기 위해, Maximum number of attempts(**K_max**)만큼 재송신을 시도한 뒤에는 포기하고 나중에
- T_p : Sender -> Receiver의 최고 전송 딜레이
    - Timeout Period = 2*T_p
    - Maximum possible round-trip propagation delay
    - 가장 멀리 벌어진 스테이션 간의 데이터 수신을 위해 걸리는 시간 * 2
- R : Random Value(K에 따라 달라짐)
    - 0 < R < 2^(K)-1
    - K = 실패한 송신의 숫자
    - K_max = Maximum 시도 숫자(보통 15)
- T_fr : 프레임을 보내는 데에 걸리는 평균 시간, 보내고자하는 frame의 크기/bps
- Backoff time : T_B = **R** * T_p or T_B = **R** * T_fr



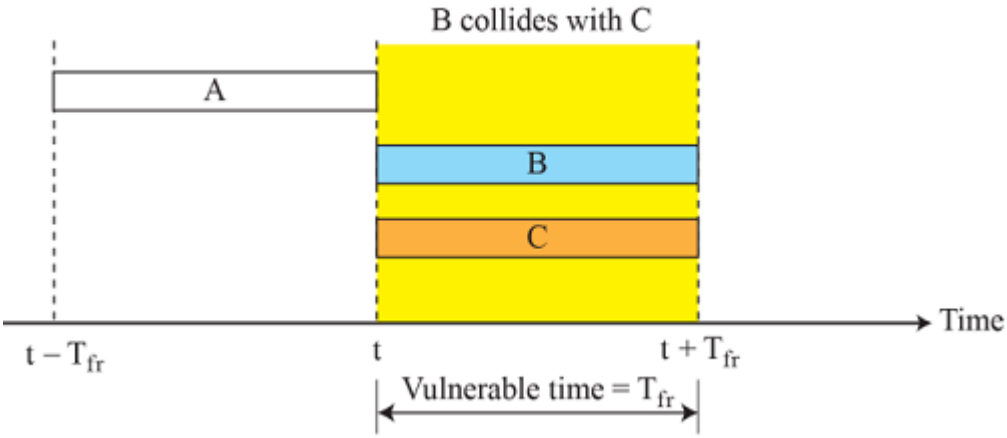
- Vulnerable Time : 충돌 가능성이 있는 **시간의 길이**



- Throughput : S = G * e^(-2G)
  - S = 성공적으로 전송한 프레임의 평균 숫자
  - G = 하나의 프레임 전송시간동안 시스템이 만든 프레임의 평균 숫자
  - e = 2.71
  - S_max = 0.184 (**G = 1/2**)

Slotted ALOHA

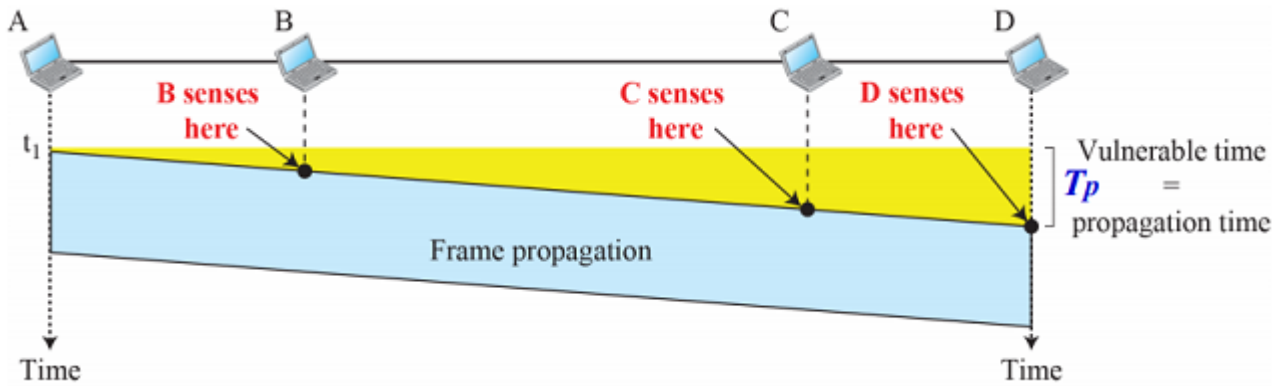
- Pure ALOHA Vulnerable Time : 2 * T_fr
- 시간을 Time Slot 단위로 쪼갬(T_fr seconds / slot)
  - 노드들에게 해당 Time Slot이 시작할때만 송신을 시도할 수 있도록 강제함
  - 노드가 때를 놓치면, 다음 Time Slot을 기다려야 함
  - 충돌 가능성은 여전히 존재
- Vulnerable Time : T_fr



- Throughput : S = G * e^(-G)
  - Maximum : 0.368 (**G = 1**)

CSMA

- Carrier Sense Multiple Access
- 충돌의 가능성을 최소화하고, 성능을 향상시키기 위함
- 스테이션이 송신하기전에 Medium을 확인하면 충돌을 최소화 할 수 있음
- Sense Before Transmit / Listen Before Talk
- 전송 딜레이 때문에, 충돌의 가능성은 여전히 존재
- Vulnerable Time : Propagation Time T_p
  - 시그널 하나가 Medium의 끝에서 끝으로 가는데에 걸리는 시간

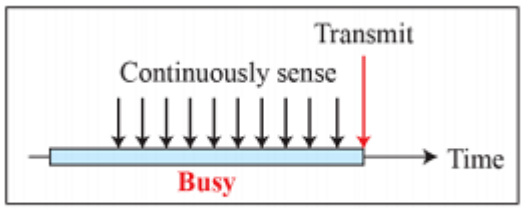


Persistence Methods

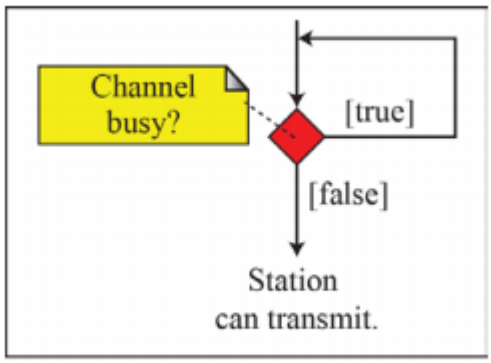
- 채널이 busy/idle일 때 NODE가 해야 할 일

1-Persistent

- 간단한 방법
- 보낼 프레임이 있으면 계속해서 Line을 Sensing하고 있음
- 라인이 Idle이면, 바로 송신함(가능성 = 1)
- 충돌 가능성이 가장높음(모든 노드들이 동시에 반응할 가능성이 높음)



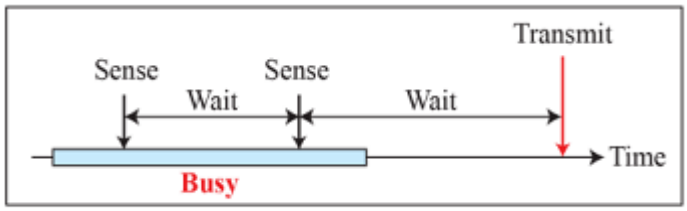
a. 1-persistent



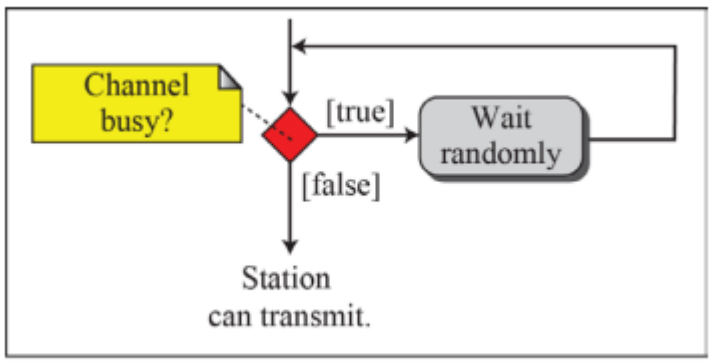
a. 1-persistent

non-Persistent

- 보낼 프레임이 있으면 계속해서 Line을 Sense
- 라인이 Idle이면, **랜덤시간만큼 기다린 후** 다시 라인을 Sense
- 충돌가능성을 줄임
- 충돌이 일어나면 다시 랜덤시간만큼 기다린 후 다시 보내기 시도



b. Nonpersistent

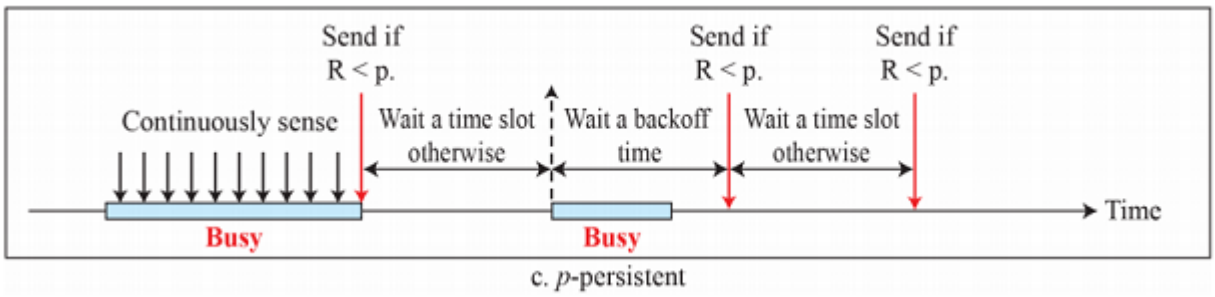


b. Nonpersistent

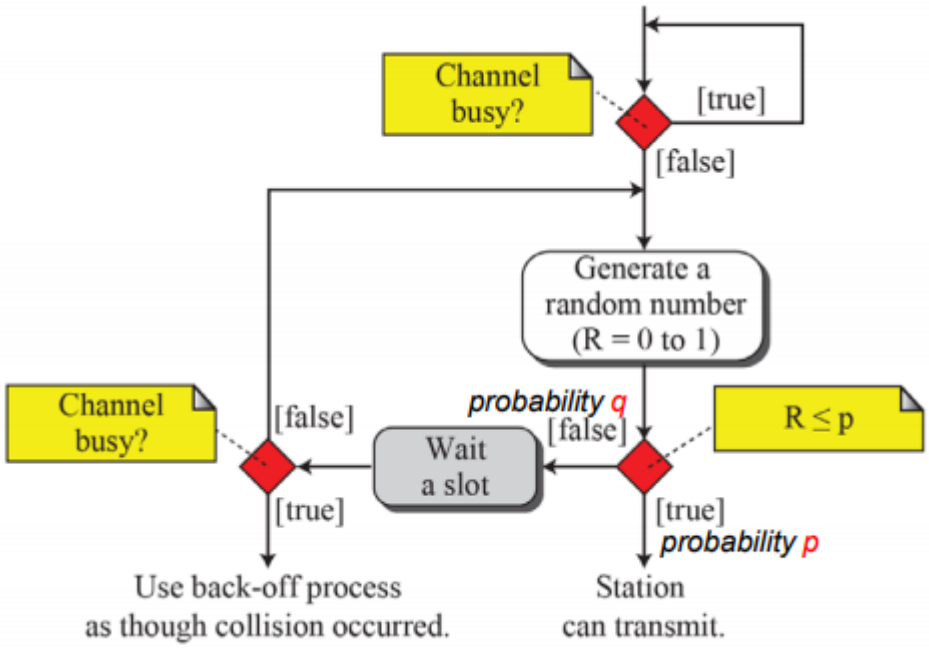
p-Persistent

- 채널에 Time Slot이 있고, Time Slot Duration >= MAX Propagation Time 일 시 사용
- 라인이 Idle이면, 다음 스텝을 따라감
  - p만큼의 확률로 노드가 프레임을 보냄

- q = 1-p의 확률로 다음 타임슬롯의 시작지점까지 기다리고 다시 라인을 확인함
  - idle -> 첫 스텝으로
  - busy -> 충돌이 발생했다고 생각하고 행동, Back-off 프로시져 실행

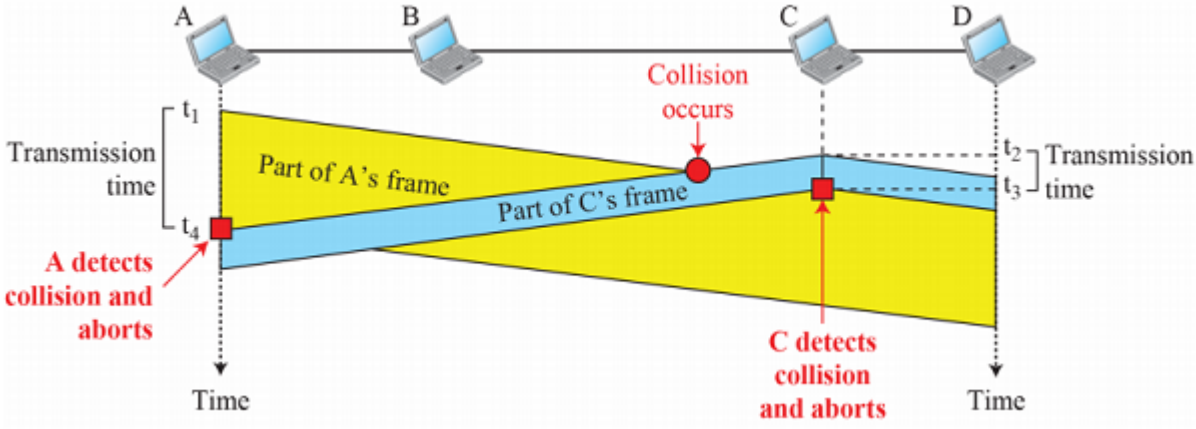


c. *p*-persistent

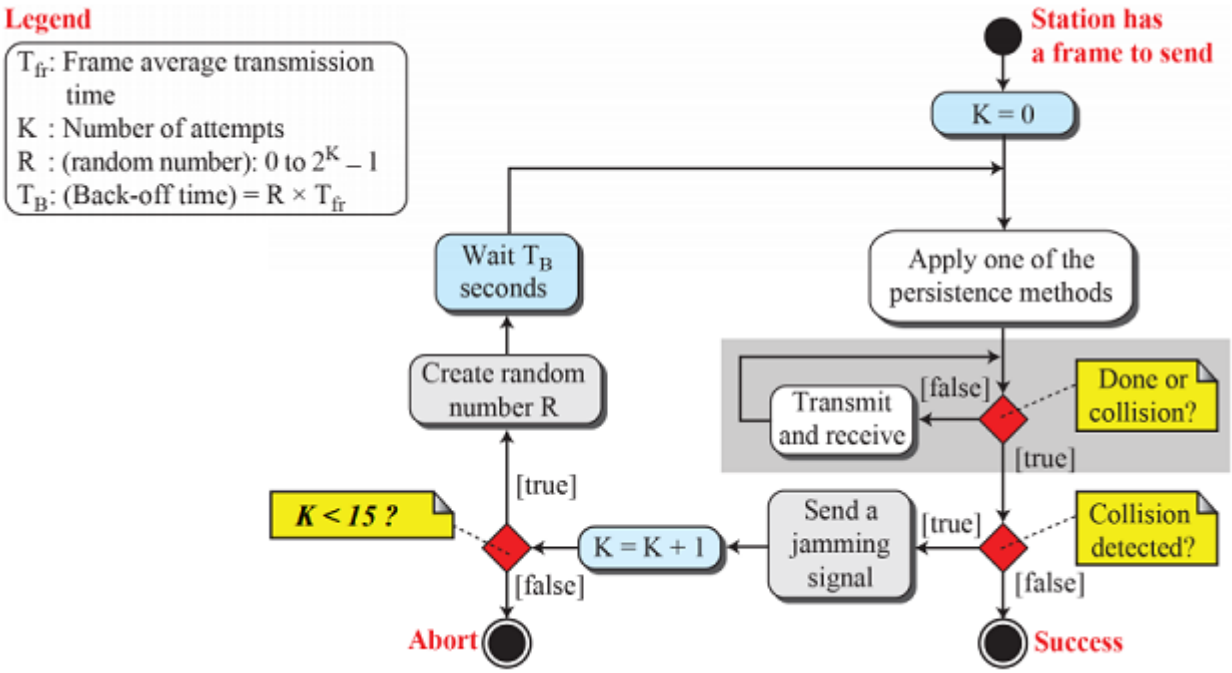


CSMA/CD

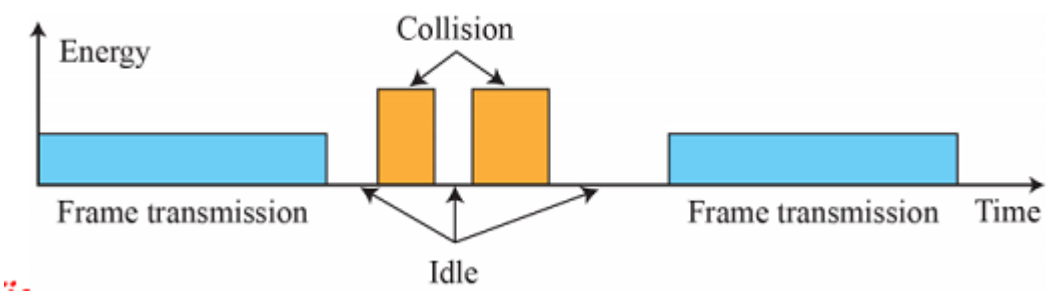
- Carrier Sense Multiple Access with Collision Detection
- CSMA는 충돌에 따른 프로시져를 정의하지 않음
- 충돌이 발생하면 충돌을 handle하기 위한 알고리즘 추가
- 스테이션이 프레임을 보낸 후 Medium을 Sense해 전송이 성공적인지 확인
  - 성공적이면 끝
  - 충돌이 발생하면 프레임 재송신



- 프레임 사이즈에 제한이 있음
  - 프레임의 마지막 비트를 보내기 전에, Sending Node에서 충돌을 감지하고, 충돌 발생 시 취소함
  - 프레임 하나 전체를 보낸 뒤에는 해당 프레임의 복사본을 더이상 갖고있지 않고, 더이상 Medium을 체크하지 않기 때문
  - T_fr = 2 * T_p -> Bandwidth * T_fr = **MIN frame size**



- Sender가 실질적으로 충돌을 탐지하는 법
  - Energy Level을 사용: 채널에 3개의 값이 있음(Zero, Normal, Abnormal)
    - zero : 채널이 IDLE 상태
    - Normal : 노드 하나가 채널을 먹고 있고, 프레임을 보내는중임
    - Abnormal : 에너지가 노말 레벨의 두배일때, 충돌상태



    - 보낼 데이터가 있으면, 케이블의 Energy level이 0인지를 확인
    - 0이면 데이터를 보내고, 계속해서 Energy Level 관찰
    - 충돌이 발생하면(Abnormal), 전송을 멈추고 즉시 32비트 JAMMING SIGNAL을 보냄
        - 다른 노드들도 충돌에 대해서 알게 하기 위함
- Throughput : pure/slotted ALOHA보다 높음
    - 1-Persistent : G = 1, MAX S = 50%
    - Non-Persistent : 3 < G < 8, MAX S = 90%

CSMA/CA

- Carrier Sense Mulitple Access with Collision Aviodance
- for **Wireless Networks**
- 세가지 전략으로 충돌 방지

Interframe Space (IFS)

- Idle 상태여도 전송을 미룸
    - Idle 채널을 찾으면, 바로 보내지 않음
    - IFS 만큼의 시간동안 기다림
        - 멀리 있는 노드의 통신이 안 끝났을 가능성을 생각
    - 기다린 후, 여전히 Idle이면, 프레임을 전송하지만, Contention Window만큼의 시간을 또 기다려야 함
    - 노드나 프레임 타입의 우선순위를 정할때 IFS 활용 가능
        - 짧은 IFS = 높은 우선순위

Contention Window

- Slot 단위로 쪼개진 시간
- 송신 준비가 완료된 노드는 랜덤넘버*슬롯 만큼의 시간을 대기시간으로 부여 받음
- Binary Exponential Back-off strategy에 대응되게 Slot의 수가 변함

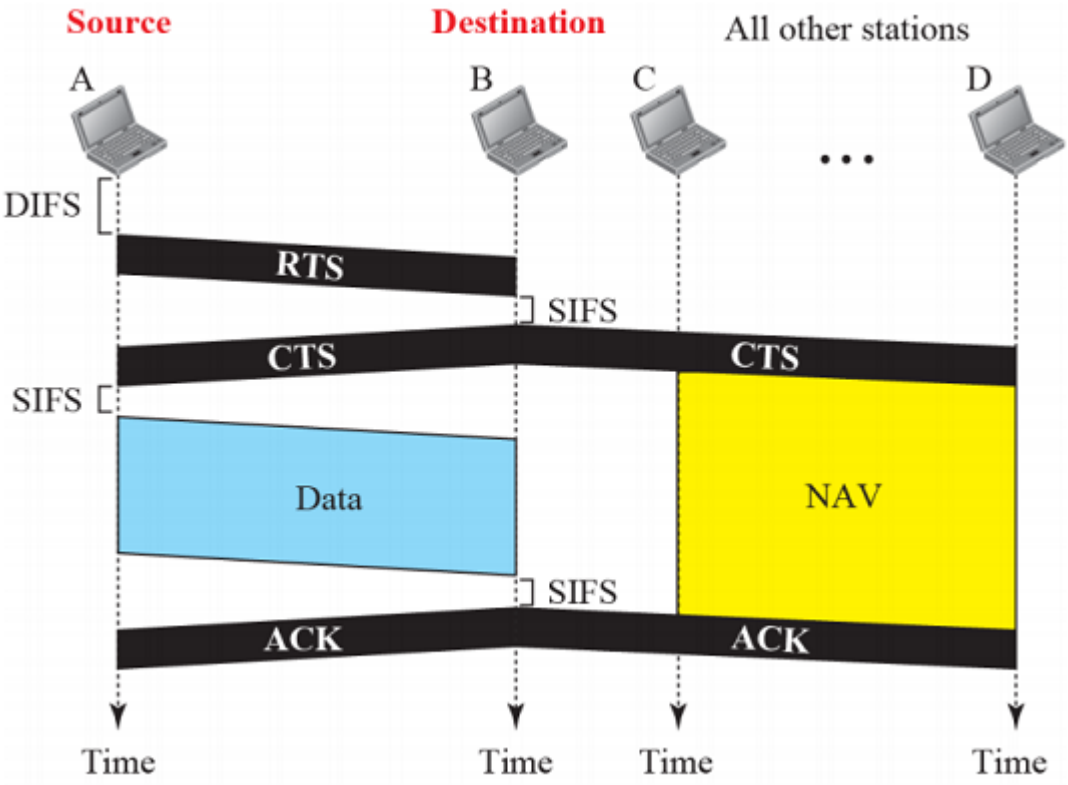
Acknowledgements

- 여전히 충돌을 배제할 수 없음
- 송신 과정에서 손상이 될 수도 있음
- Positive ACK / 타임아웃 타이머를 추가적으로 사용

Frame Exchange Time Line

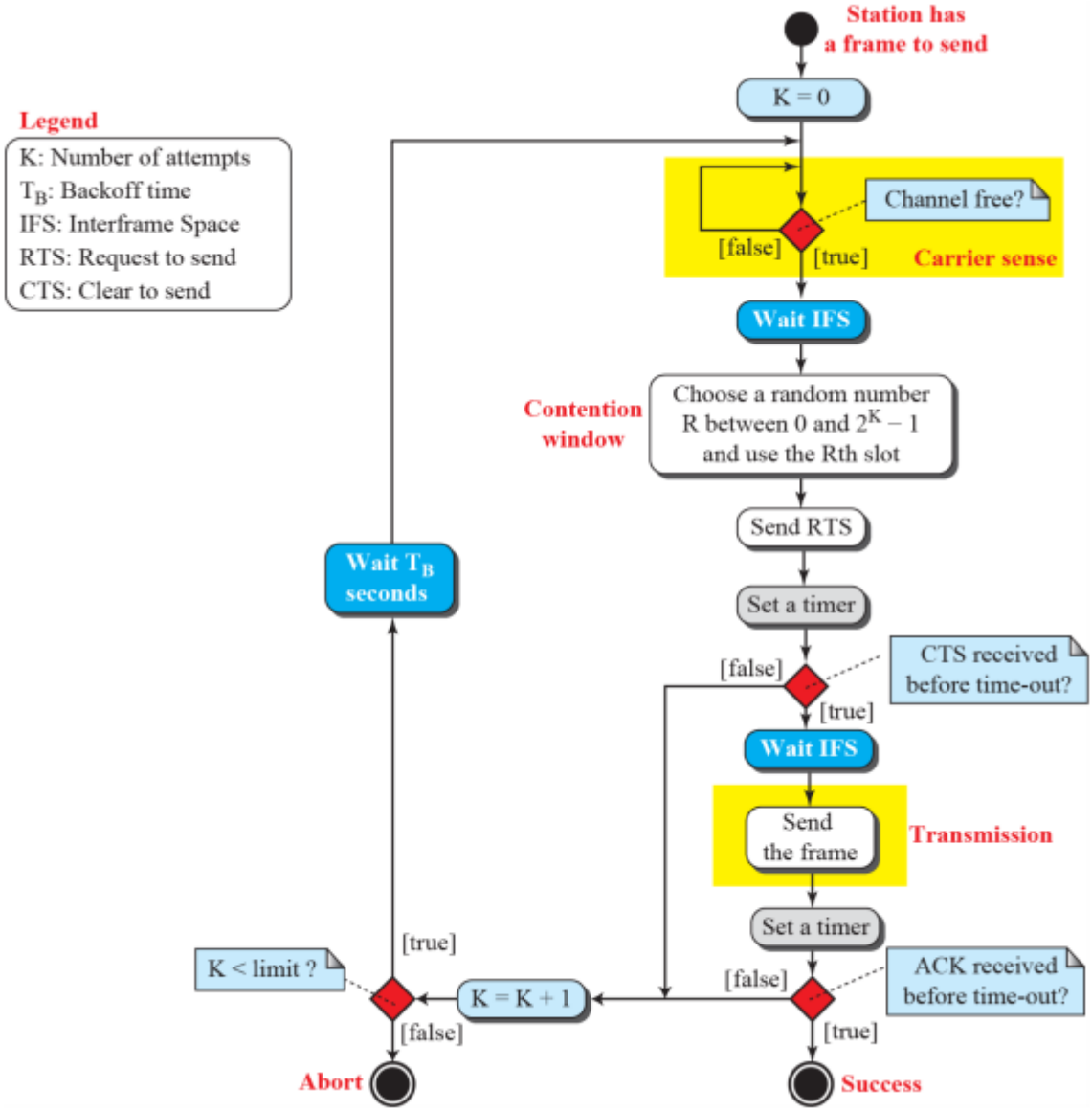
- 시간 기준 데이타 / 컨트롤 프레임의 교환
    - 프레임을 보내기전에, Sender A가 carrier frequency에서 링크의 에너지레벨을 채크함
        - persistence strategy with back-off를 채널이 idle 될때까지 반복
        - 링크가 idle이 되면, Sender A가 DIFS(Distributed Coordination Function Interframe Space) 만큼 기다림
        - RTS(Request To Send - Control Frame임)를 Receiver B에게 보냄
    - B가 RTS를 받으면, SIFS(Short Interframe Space)만큼 기다리고, Clear To Send(CTS)라는 컨트롤 프레임을 보냄 - 데이터를 받을 준비가 다 됐다는 뜻
    - A가 SIFS만큼 기다린 후, 프레임을 보냄

- B가 SIFS만큼 기다린 후, ACK를 보내 프레임이 안전하게 도착했음을 알림



Network Allocation Vector(NAV)

- 다른 노드들은 한 노드가 통신중일 때 어떻게 데이터 송신을 연기하는가?
- 한 Node가 RTS(Request To Send)를 보낼 때, 채널을 얼마만큼의 시간동안 채널을 먹고있어야하는지에 대한 정보도 들어있음
- 이 정보를 받은 노드들은 NAV라는 타이머를 만들어서 채널이 idle인지 체크할때까지 얼마나 기다려야 하는지를 기록하고 있는다
- 한 스테이션이 RTS 프레임을 보낼때마다, 다른 스테이션들은 NAV를 시작한다
- 링크를 체크하기전에 NAV가 끝났는지부터 확인한다

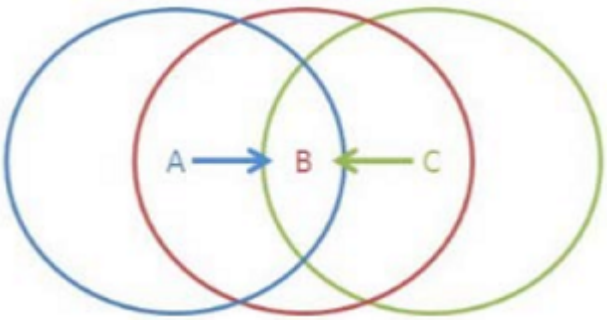


Collision During Handshaking

- RTS / CTS 컨트롤 프레임을 주고받는 시간(Handshaking Period)에서 충돌이 일어날경우
- 충돌 탐지를 위한 메커니즘은 없음
- RTS 보냈는데 CTS를 못받았으면, 충돌이 있었다고 가정
- Back-Off Strategy를 적용하고, Sender가 재시도함

Hidden Station Problem

- Node A와 C의 통신거리가 너무 짧아서 서로 인지하지 못해서 동시에 B에게 RTS를 보냄



- A와 C가 거의 동시에 B에게 RTS를 보내고, B는 A에게 먼저 반응
- B가 CTS를 A에게 보내고 A와 B의 통신 시작
- C는 이 CTS를 받은 뒤, 해당 시간만큼 통신 안함

Controlled Access

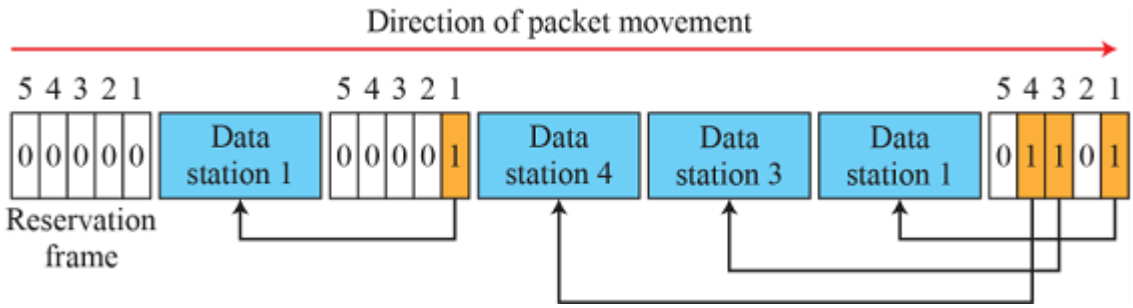
- 스테이션들이 어떤 스테이션이 보낼 권리가 있는지 서로 확인함
- 다른 스테이션들에게 확인 받기 전까지 못 보냄

Reservation

- 데이터를 보내기 전에 Reservation을 해야 함
- 시간이 Interval로 쪼개짐
- 각각의 Interval에, Reservation Frame이 Data Frame보다 먼저 보내짐

N reservation minislots in the reservation frame for N Nodes

- Node 하나당 해당하는 Minislot이 있음
- Node가 프레임을 보내야 할 때, 자기 자신의 minislot에 reservation을 만듬
- Reservation을 만든 Node들은 Reservation Frame을 보낸 뒤 데이터 프레임을 보냄



Polling

- 하나의 장치가 Primary Station으로 지정되고, 다른 장치들이 Secondary Station으로 지정 됐을 때
- 모든 데이터 교환이 Primary Device를 통해 이뤄져야 함(궁극적 목적지가 Secondary라도)
- Primary Device가 링크를 컨트롤함, Secondary Devices들은 Primary의 명령에 따름
    - 어떤 장치가 채널을 사용할지 정하는건 Primary Time임

Select

- Primary Device가 보낼 데이터가 있을때 사용됨

Poll

- Primary Device가 Secondary Device로 부터 데이터 송신을 요청받았을 때
- Primary Device가 각 순서대로 Secondary Device에게 보낼 데이터가 있는지 물어봄
    - Secondary는 이에 NAK 또는 데이터를 보냄
    - 데이터를 받은 경우 Primary가 잘 받았다는 의미로 ACK를 보냄

Token Passing

- 스테이션들이 논리적인 원을 형성함
    - 각각의 스테이션이 전임자/후임자를 가진다는 뜻
- 현재 스테이션 : 현재 채널에 접근 하고 있는 스테이션
- 전임자 : 현재 스테이션의 앞에 위치하는 녀석
- 후임자 : 현재 스테이션의 뒤에 위치하는 녀석
- Token이라는 Special Packet이 Ring을 돈다
    - 이 토큰이 해당 노드에게 채널 사용권이 있음을 뜻함
    - 첫 스테이션에 도달할 때까지 반복, 첫 스테이션은 다시 계속 반복
- Logical Ring : 물리적으로 연결되어 있을 필요 없음 -> 여러 Topology 사용 가능

Physical Ring Topology

- 토큰을 가지고있을 때 후임자의 주소를 몰라도 됨
- 문제 : 인접한 두 노드간의 링크에 문제가 생길 경우, 시스템 다 터짐

Dual Ring Topology

- 반대 방향으로 도는 두번째(예비) 링을 사용
- 두번째 링은 그냥 비상용임
- FDDI(Fiber Distributed Data Interface), CDDI(Copper Distributed Data Interface)

Bus Ring Topology

- 토큰 버스라고 불림
- 노드들이 버스라고 불리는 케이블 하나에 다 연결됨
- Token Bus LAN (IEEE)에 사용

Star Ring Topology

- Star Topology 사용
- 노드들이 Hub에 연결됨
- Hub내의 wiring이 링을 이룸, 노드들은 이 링에 two wire connection 형태로 연결
- 링크 하나가 고장나더라도 다른 링크에는 지장이 없고 통신 가능
- 노드를 추가하거나 제거하는 작업이 수월함
- Token Ring LAN (IBM)에 사용

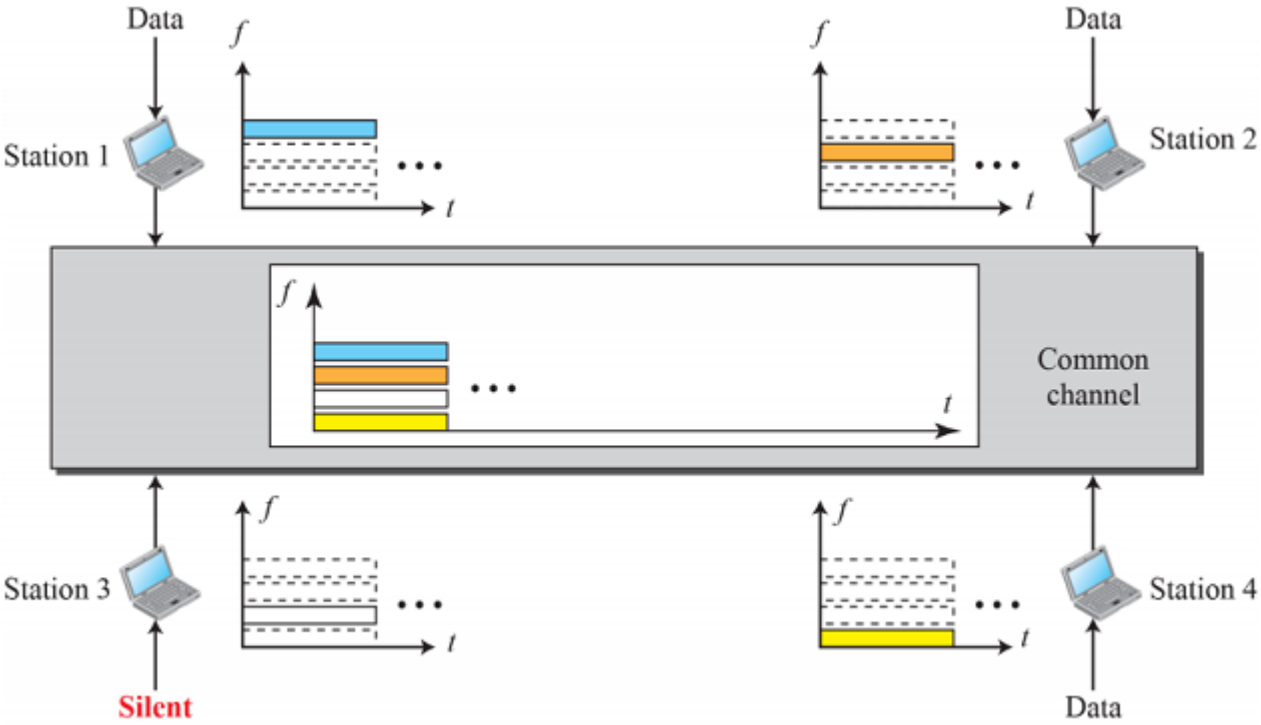
Channelization

- 다중접근방법, 링크의 가용한 Bandwidth가 Time, Freqeuncy, 또는 Code를 통해 여러 스태이션에서 공유 되는 것

FDMA

- Frequency Division Mulitple Access
- 가용한 Bandwidth가 Frequency Band들로 나눠짐
- 각각의 스테이션은 데이터를 보낼 수 있는 Band를 할당받음
    - 각각의 Band는 특정 스테이션을 위해 Reserved, 항상 그 스테이션에 속함
    - 각각의 스테이션은 bandpass filter를 사용해 Transmitter Frequency를 제한함

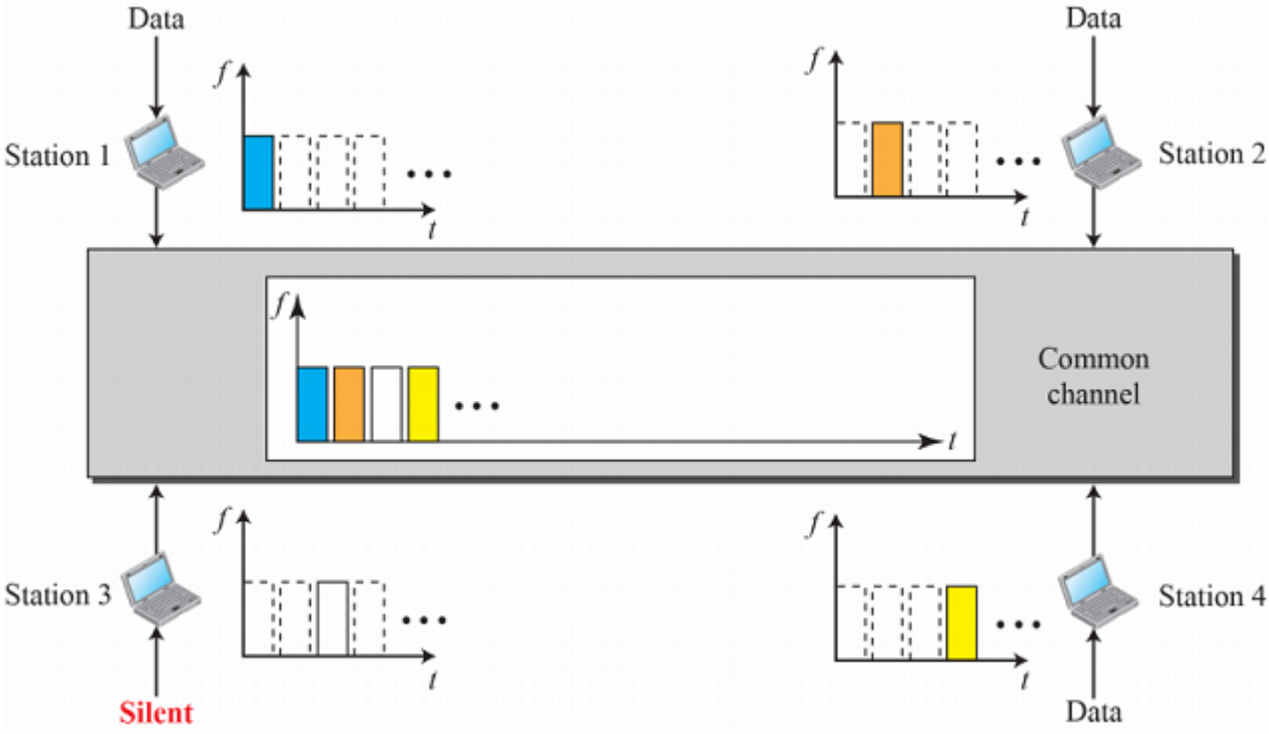
- 스테이션 혼선을 방지하기 위해, 할당된 밴드들은 각자 작은 Guard Band들을 사용해 고립시킴



- FDMA가 전체 통신기간동안 미리 결정된 Frequency Band를 지정함
- 데이터 스트리밍에 최적격

TDMA

- Time-Division Mulitple Access
- 각 스테이션들이 채널의 Bandwidth를 공유함
- 각 스테이션이 데이터를 전송할 수 있는 Time Slot을 할당받음
- 각 스테이션이 할당된 Time Slot에서 데이터 전송



CDMA

- Code Division Multiple Access
- 수십년전에 상상으로만 존재, 현재는 기술의 발전으로 구현이 가능해짐
- 하나의 채널이 Bandwidth 전체를 차지
- 모든 스테이션이 데이터를 동시에 보낼 수 있음(No Timesharing)

Analogy

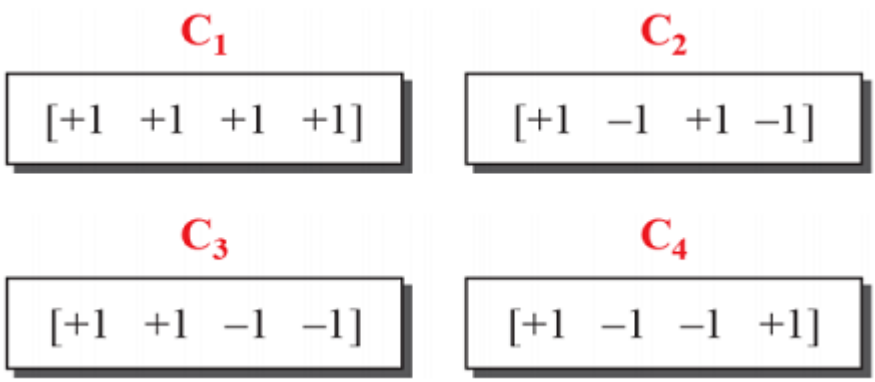
- 다른 코드와의 통신
- 모든 스테이션이 같이 대화하더라도, 자신들만의 언어로만 대화한다면 대화에 지장이 없다
- 이론상 한번에 여러개의 통신이 가능함

Idea

- 가정
  - 4개의 노드 (1,2,3,4)가 하나의 채널에 연결
  - d1, d2, d3, d4 데이터를 의미
  - c1, c2, c3, c4 코드를 의미
  - 코드들은 두개의 소유물을 가짐
    - 첫번째 : 서로 다른 코드를 서로 곱하면 0이 됨
    - 두번째 : 같은 코드를 셀프 곱하면 4가 됨(스테이션의 수)
- 각 스테이션은 데이터에 자신의 코드를 곱해서 채널에 보냄
  - 스테이션들이 보낸 값들의 합이 채널의 데이터로 돌아다님
- 스테이션 1의 말을 듣기 위해서 채널의 데이터에 c1을 곱함 -> 4d1이 나옴
  - 나누기 4하면 원하던 데이터가 나옴

Chips

- Coding Theory에 바탕을 둠
- 각 스테이션은 chips라고 불리는 코드를 지정받음
    - Chips : Sequence of numbers(Orthogonal Sequence)

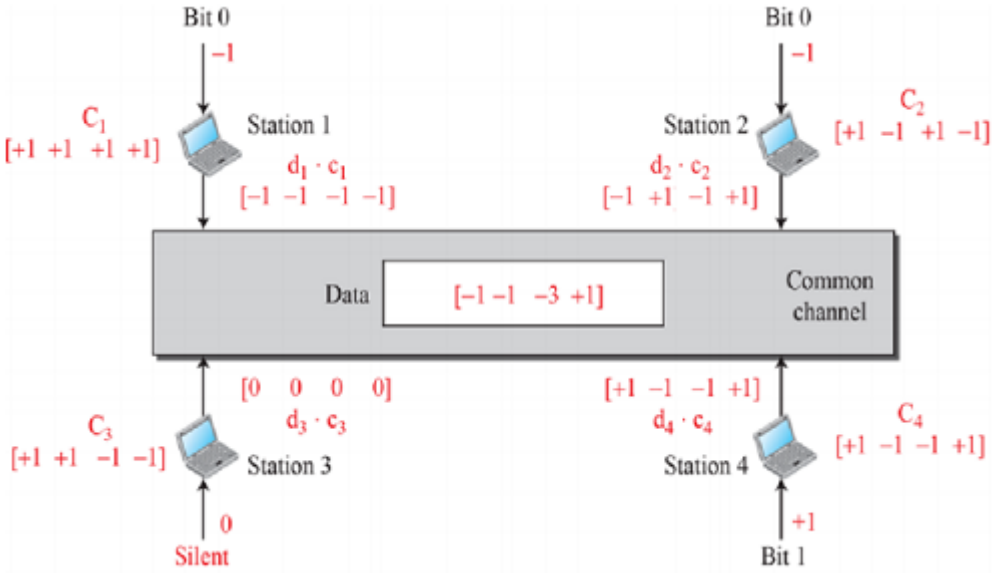


- Orthogonal Sequence
    - 각각의 시퀀스는 N개의 요소로 이루어짐(N = 스테이션의 수)
    - 시퀀스에 숫자를 곱하는건 벡터연산
    - 같은 시퀀스 두개를 서로 곱하면 결과는 N이 됨
    - 서로 다른 시퀀스를 곱하게 되면, 결과는 0이 됨
    - 두개의 시퀀스를 더하는건 벡터연산, 결과또한 시퀀스

Data Representation

- 스테이션이 0 비트를 보내야 하면 -1로 코딩
- 스테이션이 1 비트를 보내야 하면 +1로 코딩
- 스테이션이 idle 상태이면, 시그널을 안보냄 (0)

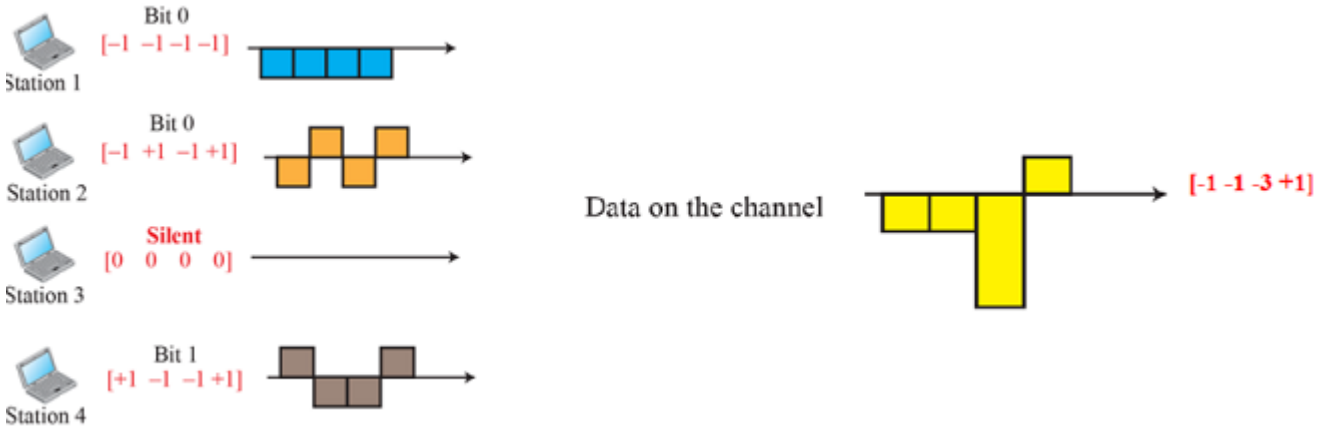
Encoding and Decoding



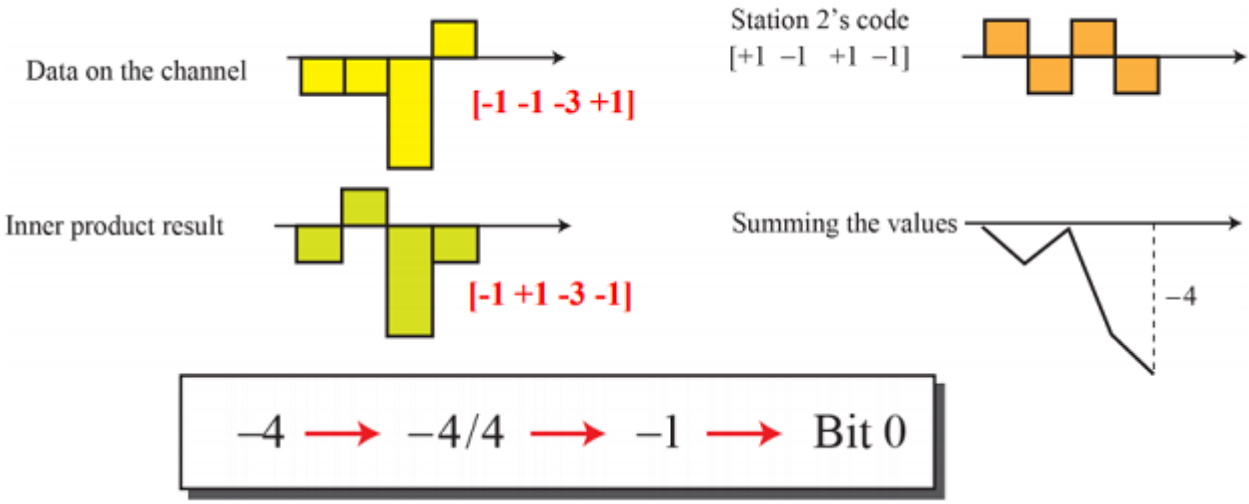
- 각 노드들은 자신이 보내고자 하는 비트를 곱해서 채널에 보냄
- 채널은 이를 합친 시퀀스를 저장하고 있음
- 값을 받고자 하는 스테이션은 이 값을 받아와서 자신의 칩에 곱한다
    - 결과값이 bit값을 의미

Signal Level

- Sender 노드들 : 각각의 디지털 시그널이 NRZ-L방식으로 생성됨

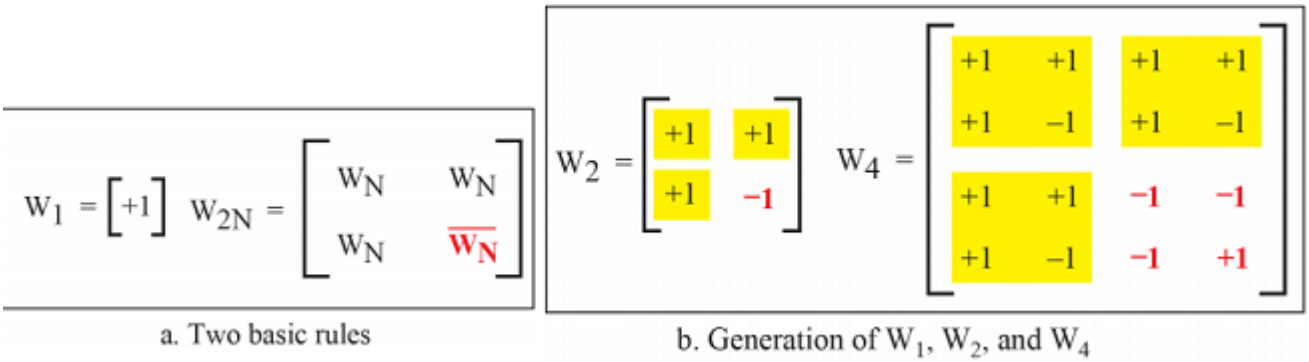


- Receivere 노드들



Sequence Generation

- Chip Sequence를 만들기 위해서 Walsh Table을 사용해야 함
  - 같은 수의 행/열로 이루어진 2차원 테이블임
  - 각각의 행이 Chip 하나임
  - 값은 1 또는 -1로 이루어짐

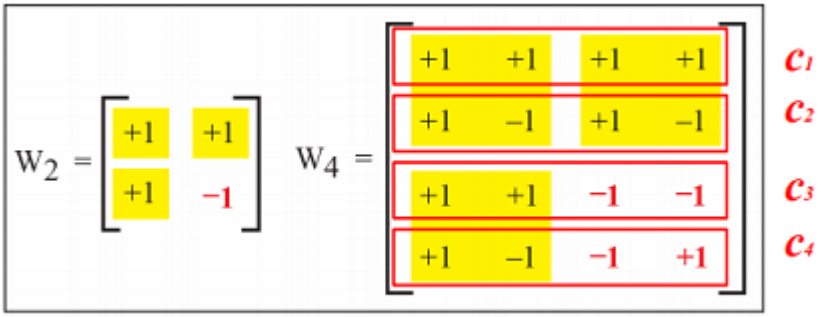


a. Two basic rules

b. Generation of W_1, W_2, and W_4

- W_n : N 시퀀스를 위한 테이블

- W_2n : 2N 시퀀스를 위한 테이블
- 필요한만큼 곱해서 테이블 생성 가능



b. Generation of W_1, W_2, and W_4