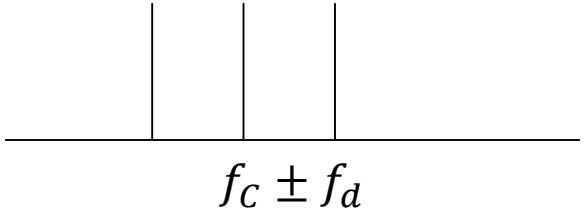
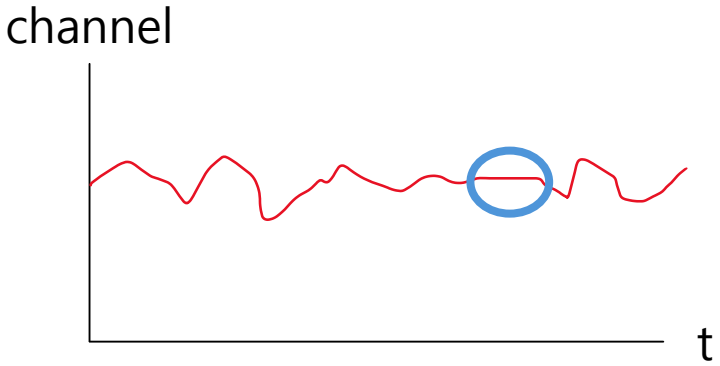
# 2

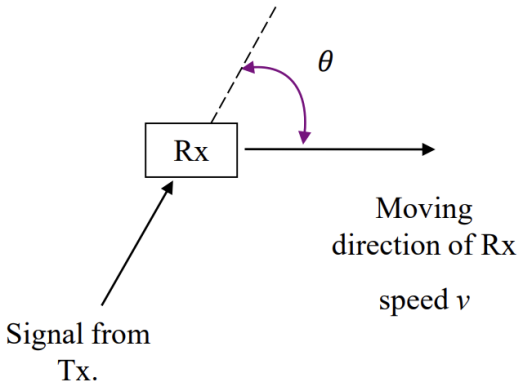
Mid: 85/100

- Doppler effect
  - Between Tx and Rx
  - Carrier frequency $f_C$
  - 특수한 경우가 아니라면 수신자 주파수도 $f_C$
  - 특수한 경우, $f_r = f_C \pm f_d$(doppler frequency)
  - 가까워질 때
    - 수신 주파수 증가, 파워 증가
  - 멀어질 때
    - 수신 주파수 감소, 파워 감소
  - 도플러 효과는 주파수가 바뀌는 현상
    - 바뀌는 주파수 양 = doppler frequency
    - 바뀌는 주파수 표준편차 = doppler spread

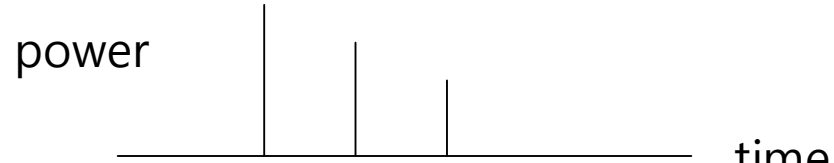
  - Coherent time
    - 시간 t와 채널에 대한 그래프에서 채널이 거의 바뀌지 않는 구간
    - 정적인 경우 coherent time very long = $f_d$ is smaller
    - 반대의 경우 coherent time very small = $f_d$ is larger
    - 1/Doppler spread와 비례관계





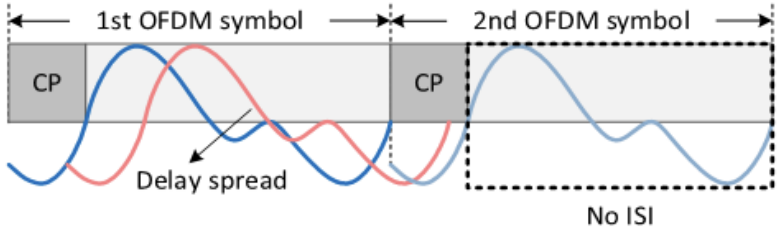

$$f_d = \frac{v}{\lambda}\cos\theta$$




- 시간 축에서의 현상 delay spread
  - Tx to Rx
  - 신호를 보낼 때, 신호는 omni-direction
    - 전 방향 spread
  - 수신자 입장에서 다양한 path가 존재
  - 여러 path는 도착시간과 파워의 차이
  - 심볼의 길이가 길다면 ISI 감소
  - 심볼이 심하게 겹치면 burst error

- Multipath delay profile
- Delay spread – standard deviation of delay

- Like echo, delayed symbol occur
- Arrive time, power difference
- Symbol
  - Waveform
  - contains some information
  - in limited time duration




The average delay

$$\tau_m = \int_0^\infty tp(t)dt$$

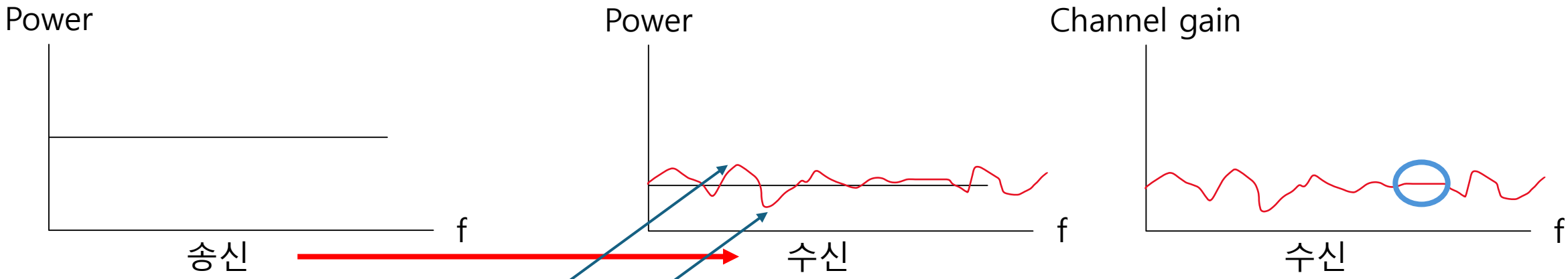
$p(t)$ : pdf of the delay t

평균

The delay spread is defined as

$$\tau_d = \sqrt{\int_0^\infty (t-\tau_m)^2 p(t)dt}$$

표준편차




- Coherent bandwidth
  - 송수신 관계에서 파워는 감소한 상태로 전달(path loss)
  - 파워는 주파수에 따라 일정하지 않고 이득과 손해 존재
  - High channel gain = 파워가 big
  - Small channel gain = 파워가 small

- 주파수에 따라 파워 gain이 달라진다.
- =주파수마다 path loss가 약간씩 다르다.
- Channel gain이 매우 빠르게 변할 수도, 거의 정적인 형태를 취할 수도 있다.

- Channel gain이 flat한 주파수 구간
  - coherence(coherent) bandwidth
  - 상관대역폭

- Frequency domain = coherence bandwidth
- Time domain = coherence time

- Coherent bandwidth ∝ 1/delay spread
- Coherent time ∝ 1/doppler spread
- Bandwidth와 time은 독립(상관관계X)

- Antenna diversity
  - 안테나 2개를 사용해서 신호파워가 좋은 것만 선택적으로 취합, 평균 파워 높이기
- Frequency diversity
  - Bandwidth구간 이상의 서로 질이 다른 주파수의 신호를 취합, channel gain 높이기

- 40page important!!!
- Frequency nonselective
  - Coherent bandwidth > transmit bandwidth
  - Receiver complexity low
  - 모든 송신 신호가 한 채널
  - No nonlinear transformation
- Frequency selective
  - Coherent bandwidth < transmit bandwidth
  - Receiver complexity high
  - 송신 신호가 여러 채널
  - Nonlinear transformation

- Tx to Rx

- Redundancy(여분)
- Original information + Redundancy information

- Like encoding
  - Redundancy 양에 따라 감지 가능한 오류 비트 개수 한정 (number of detecting bit)

  - Error correction
  - Error correcting code
  - Error detection
  - Error detecting code

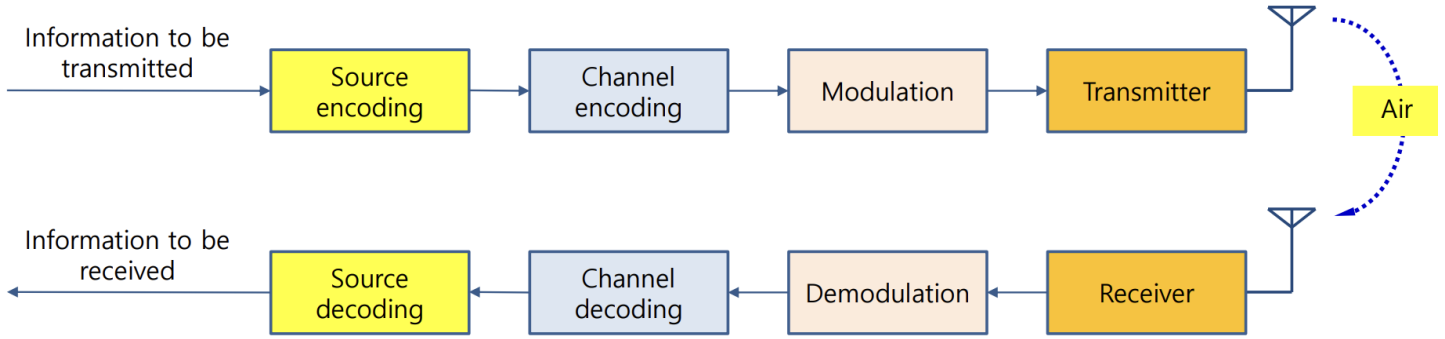
  - Error correction is more difficult
  - Transmission efficiency, decrease redundancy, increase error correction/detection

- Transmit info → Rx report error → Tx retransmit
- Acknowledge (Ack)
- Nacknowledge (Nack)
- ARQ = automatic repeat request

4

- 셀들 간의 overlap에 의해

- Channel coding
- Original + Redundancy
  - More source
  - More bits
  - More bandwidth
  - But can correct or detect errors




- FEC(앞으로 발생할 오류를 대비해서 기존 데이터 + redundancy를 미리 저장)
  - Forward Error correction

- Linear block code

- 1011(original) + 101(redundancy)
- 1bit error will be automatically corrected

- 정보의 한 블록은 다음과 같은 형태
- K bit information + R bit redundancy
- Total N bits = K + R
- Code rate = K/N(전체 신호에 실제 정보가 있는 비율)
- Code rate의 최댓값은 1
  - No redundancy case
  - 오리지널 정보만 전송
  - No error detection/correction
- Code rate의 최솟값은 0근사
  - high redundancy case
  - 더 많은 오류(비트 수) 감지 및 교정

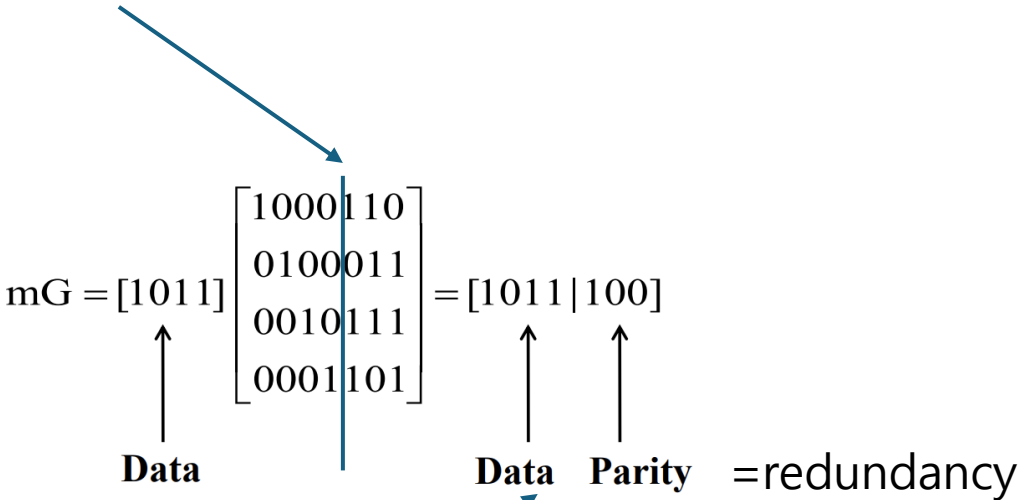
- Channel is good → increase code rate
- Channel is bad → decrease code rate
- Code rate high → Efficiency high, reliability low
- Code rate low → Efficiency low, reliability high

- Modulo 2 addition → XOR
- 비트의 합을 2로 나눈 나머지

- 4bit vector(original), 7bit vector(total) 예
- 2^4 = 16
- 2^7 = 128
  - 128개의 벡터 중 16개 코드만 1대1 매칭 가능
  - Partition 16개, 서로 겹치는 부분 없이 각 partition에는 2^3=8개의 코드 존재
  - 0000 original bit code word에 대해
  - 0000000, 0000001, 0000010, 0000011…

- 0000000에 대해 1bit error의 경우 7개의 오류 존재 (1000000, 0100000…), 이 7개로 하나의 partition 구성, 0000000을 부분의 중심으로 설정, 디코딩 과정에서 0000000 추출 가능
- 2bit 에러는 감지 불가능
- Systematic code = 원래 정보가 그대로 유지
- Non systematic code = 원래 정보 유지 X

Encoding → decoding




$$\begin{bmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{bmatrix}$$

계산 팁... 1011, 두번째 제외하고
Modulo 2 operation, 1011 100

Identity matrix의 modulo 2를 했을 때의 결과가 원래 데이터가 나오도록 연산

1. 4*4 identity matrix
2. Randomly given signal = generate polynomial이 parity 비트를 생성 및 구성하는 핵심 기술
3. c = m*G, (code V = message V * Generator matrix), Data = systematic code

Data + parity 벡터와 H transpose 벡터의 곱은 항상
0 (1x7 * 7x3)=[0, 0, 0](syndrome vector)

만약 receive vector가 0벡터가 아니라면 error occur

- [Example] Find linear block code encoder **G** if code generator polynomial $g(x)=1+x+x^3$ for a (7, 4) code;
  - Total number of bits n = 7,
  - Number of information bits k = 4,
  - Number of parity bits r = n - k = 3

$$p_1 = rem\left[\frac{x^3}{1+x+x^3}\right] = 1+x \rightarrow [110]$$

$$p_2 = rem\left[\frac{x^4}{1+x+x^3}\right] = x+x^2 \rightarrow [011]$$

$$p_3 = rem\left[\frac{x^5}{1+x+x^3}\right] = 1+x+x^2 \rightarrow [111]$$

$$p_4 = rem\left[\frac{x^6}{1+x+x^3}\right] = 1+x^2 \rightarrow [101]$$

$$G = \left[\begin{array}{c|c} 1000 & 110 \\ 0100 & 011 \\ 0010 & 111 \\ 0001 & 101 \end{array}\right] = [I \mid P]$$

I is the identity matrix
P is the parity matrix

- 인코딩 과정에서의 생성된 매트릭스 G
  - 4*4 identity matrix
  - 4*3 parity matrix
  - G = I + P
- 계수를 읽는 순서는 상관 X

- **Parity Check Matrix: H**

Define matrix $\mathbf{H^T}$ as $\mathbf{H}^T = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}$

110

011

111

101

1 0 0
0 1 0
0 0 1

- 송신 벡터 c
- 에러 벡터 e
- 수신 벡터 x

- x = c + e
  - $x * H^T$ = 0, fine
  - $x * H^T$ != 0, error

**Syndrome:** $\mathbf{S = xH^T = (c \oplus e)H^T = cH^T \oplus eH^T = eH^T}$

**Let c suffer an error such that the received vector**

**x=c ⊕ e**

**=[ 1 0 1 1 0 0 1 ] ⊕ [ 0 0 1 0 0 0 0 ]**

**=[ 1 0 0 1 0 0 1 ]**

**Then,**

**Syndrome $s=xH^T$** $= [1001 \mid 001]$

$$\begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \\ -- \\ 100 \\ 010 \\ 001 \end{bmatrix} = [101] = (eH^T)$$

This indicates error position, giving the corrected vector as [1011001]

$(c \times (I|P) + e) \times \left(\frac{P}{I}\right)$ 꼴...

- 수신 벡터 값이 1인 위치 1, 4, 7번째 벡터를 더하면 계산 편리
- 수신 벡터 [1 0 1], 0이 아닌 에러 벡터, 3번째 비트 에러 판별

- Cyclic codes

- Message vector = [1, 0, 1, 1]
- M(x) = $x^3 + x^1 + x^0$

- Parity vector = [1, 0, 1]
- P(x) = $x^2 + x^0$

- Code vector = $[C_1, C_2, \ldots, C_n]$
- C(x) = $C_1 x^{n-1} + C_2 x^{n-2} + \cdots + C_n x^0$

- 예를 들어... [1, 0, 1, 1]
- M(x) = $x^3 + x^1 + x^0$ 벡터를 옆으로 shift 1하면 M(x) = $x^4 + x^2 + x^1$

- $m(x)x^{n-k}$ → n-k = 3, shift 3
- [1, 0, 1, 1, 0, 0, 0]

- The codeword can be expressed by the data polynomial m(x) and the check polynomial $c_p(x)$ as

  **$c(x) = m(x)\, x^{n-k} + c_p(x)$**

  where $c_p(x)$ = remainder from dividing $m(x)\, x^{n-k}$ by generator g(x).

  $$\Rightarrow c_p(x) = \text{rem}\left[\frac{m(x)x^{n-k}}{g(x)}\right]$$

- Syndrome s(x)=0 if there is no error.

  where, $s(x) = \text{rem}\left[\frac{c(x)+e(x)}{g(x)}\right]$, e(x) is error polynomial.

- If s(x) ≠ 0, then there is an error.

Code vector = (메시지 벡터 shift) + parity vector

- Find the codeword c(x) for (7,4) cyclic code if $m(x) = 1+x+x^2$ and $g(x) = 1+x+x^3$
  - n: codeword length, n=7
  - k: No. of information bits, k=4
  - n-k: No. of parity bits, n-k=3

- $c_p(x) = \text{rem}\left[\frac{m(x)x^{n-k}}{g(x)}\right] = \text{rem}\left[\frac{x^5+x^4+x^3}{x^3+x+1}\right] = x.$
- Thus, $c(x) = m(x)x^{n-k} + c_p(x) = x + x^3 + x^4 + x^5$
  - Here, $s(x) = \text{rem}\left[\frac{c(x)+e(x)}{g(x)}\right] = 0$ , if e(x)=0

Message V = [0, 1, 1, 1]

$m(x)x^{n-k}$ % g(x) = 나머지 = parity

- (message + parity) % g(x) = 0
- (message + parity + error) % g(x) != 0
- (message + parity + error) % g(x) = syndrome

- Message to be encoded: 11010011101100

```
11010011101100 000 <--- input right padded by 3 bits
1011               <--- divisor
01100011101100 000 <--- result
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------
00000000000000 100 <---remainder (3 bits)
```

```
11010011101100 100 <--- input with check value
1011               <--- divisor
01100011101100 100 <--- result
 1011              <--- divisor ...
00111011101100 100

......

00000000001110 100
          1011
00000000000101 100
           101 1
-----------------
             0 <--- remainder
```

  - 3-bit CRC: R = 100
  - Message & CRC: J = 11010011101100**100**

CRC(Cyclic Redundancy Check) or check sum

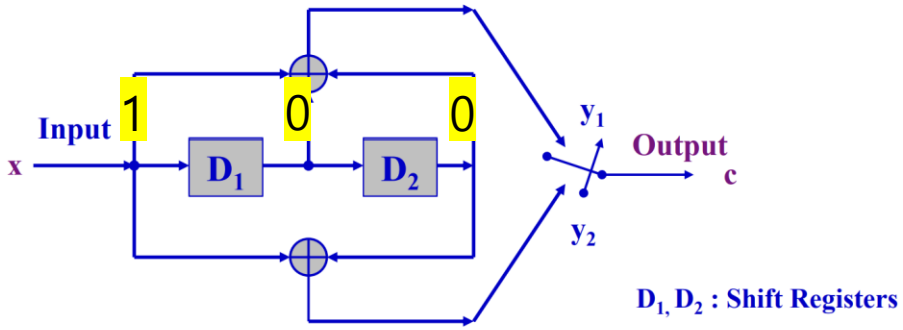
Cyclic code를 계수만을 이용
공동의 divisor 이용(generator polynomial)

Message + CRC(parity) = 전송신호
(전송신호%divisor) → 0, no error
(전송신호%divisor) → !0, error

- Convolutional codes (길쌈 부호)
  - Block code, CRC와 같이 어떤 특정 block(4bit)을 사용하지 않고 information stream 자체를 사용
  - Viterbi algorithm 사용
  - 복잡하지 않으며 성능이 좋은 알고리즘

  - Code rate = ½
  - One input, Two output
  - 인풋과 D1, D2의 값을 밀어내며 업데이트
  - $x$, $y_1$, $y_2$에 대해 exclusive-or 연산

▪ A convolution code with code rate r=1/2, M=2, K=3 (Encoder)




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Input x: | 1 | 1 | 1 | 0 | 0 | 0 | … |
| Output $y_1$,$y_2$: | 11 | 01 | 10 | 01 | 11 | 00 | … |

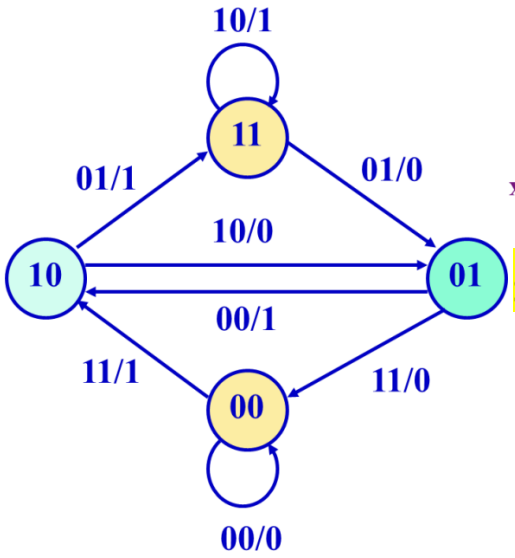
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Input x: | 1 | 0 | 1 | 0 | 0 | 0 | … |
| Output $y_1$,$y_2$: | 11 | 10 | 00 | 10 | 11 | 00 | … |

First state, $D_1D_2 = 00$

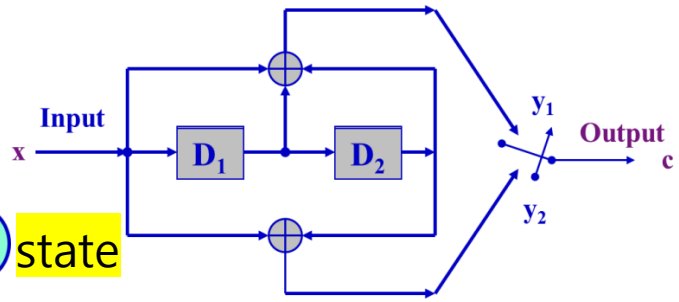
Input : 1 → $y_1 = 1$, $y_2 = 1$, $D_1D_2 = 10$
Input : 1 → $y_1 = 0$, $y_2 = 1$, $D_1D_2 = 11$
Input : 1 → $y_1 = 1$, $y_2 = 0$, $D_1D_2 = 11$
Input : 0 → $y_1 = 0$, $y_2 = 1$, $D_1D_2 = 01$
Input : 0 → $y_1 = 1$, $y_2 = 1$, $D_1D_2 = 00$
Input : 0 → $y_1 = 0$, $y_2 = 0$, $D_1D_2 = 00$

# State diagram, Trellis diagram

State diagram




Out/In

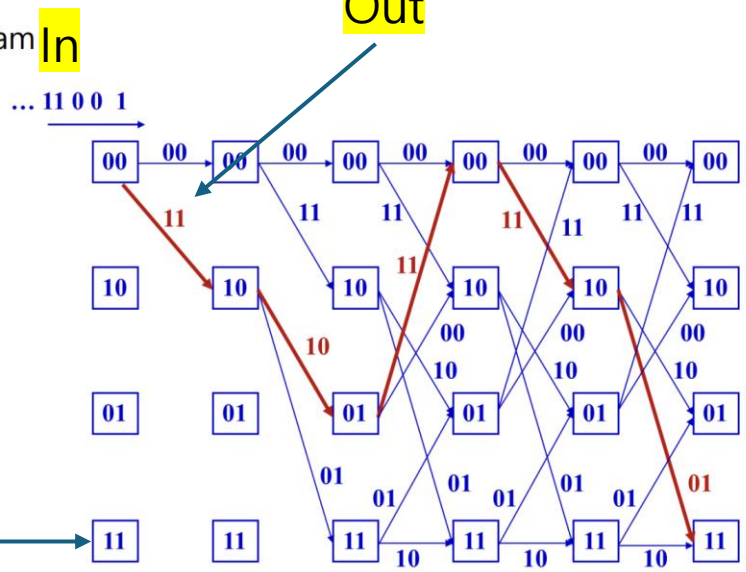


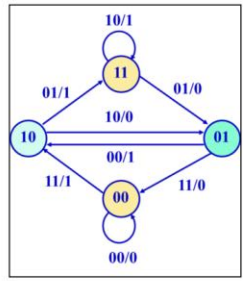

state

- Trellis Diagram

In

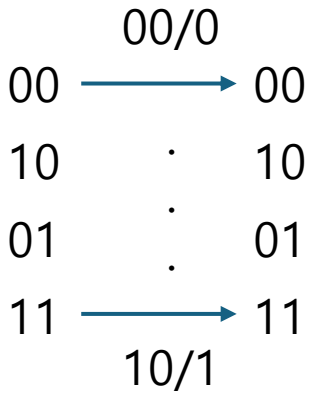
Out



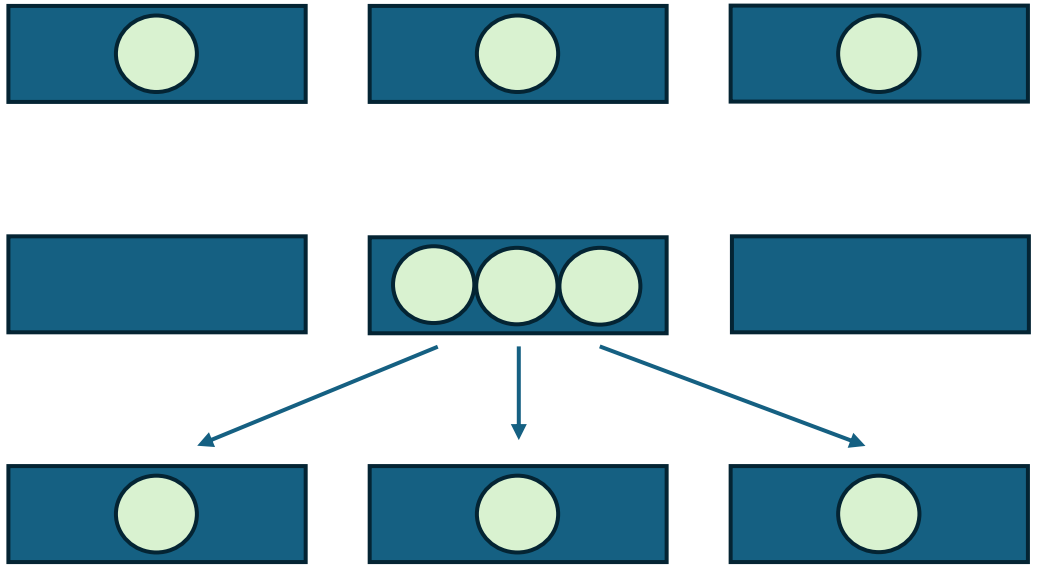

...

state

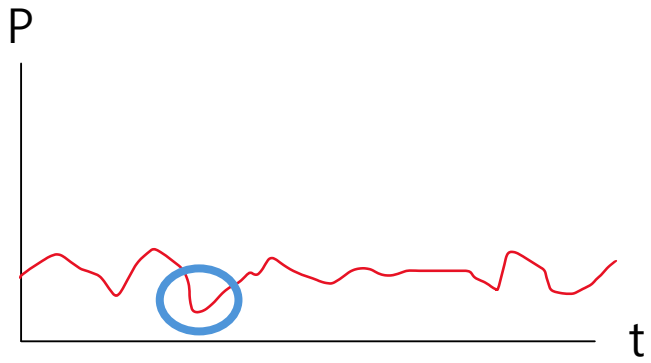


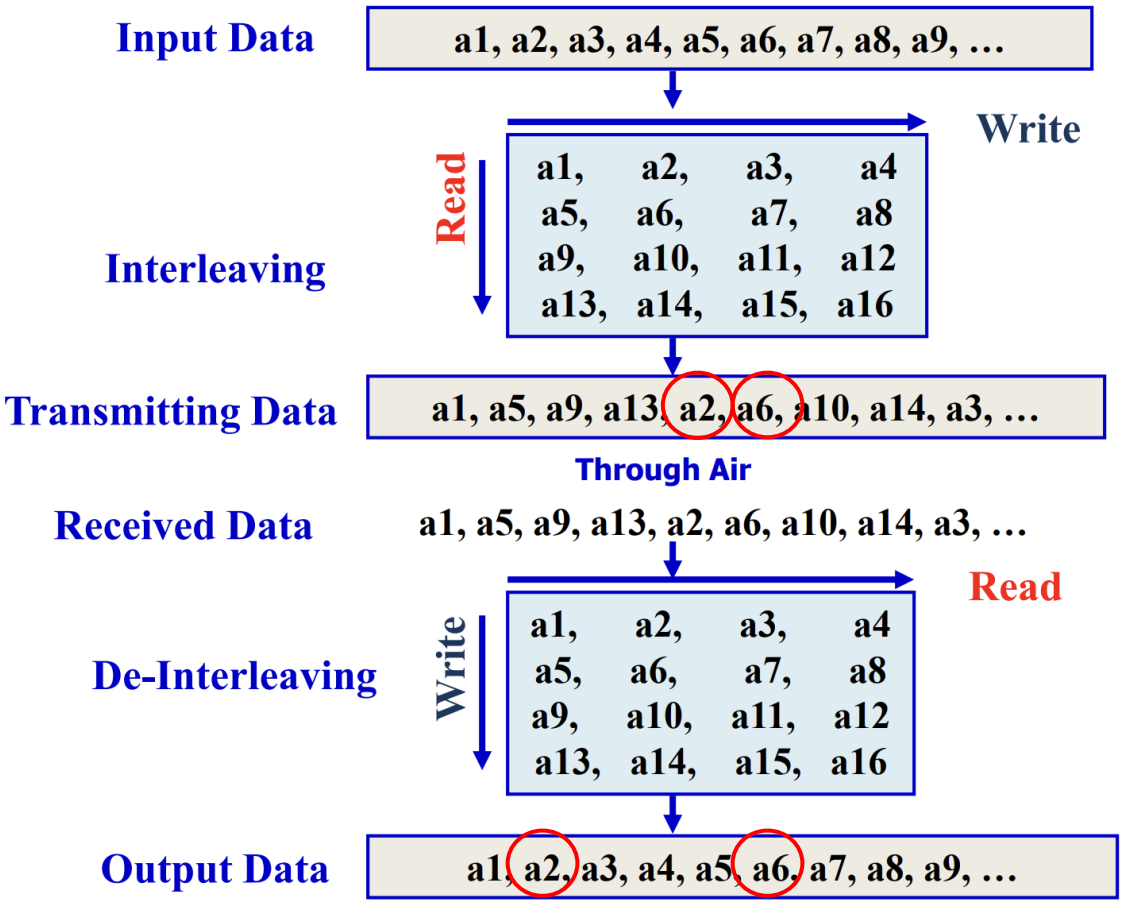

하나의 state에 대해 화살표 2개
Upper → for input 0
Lower → for input 1

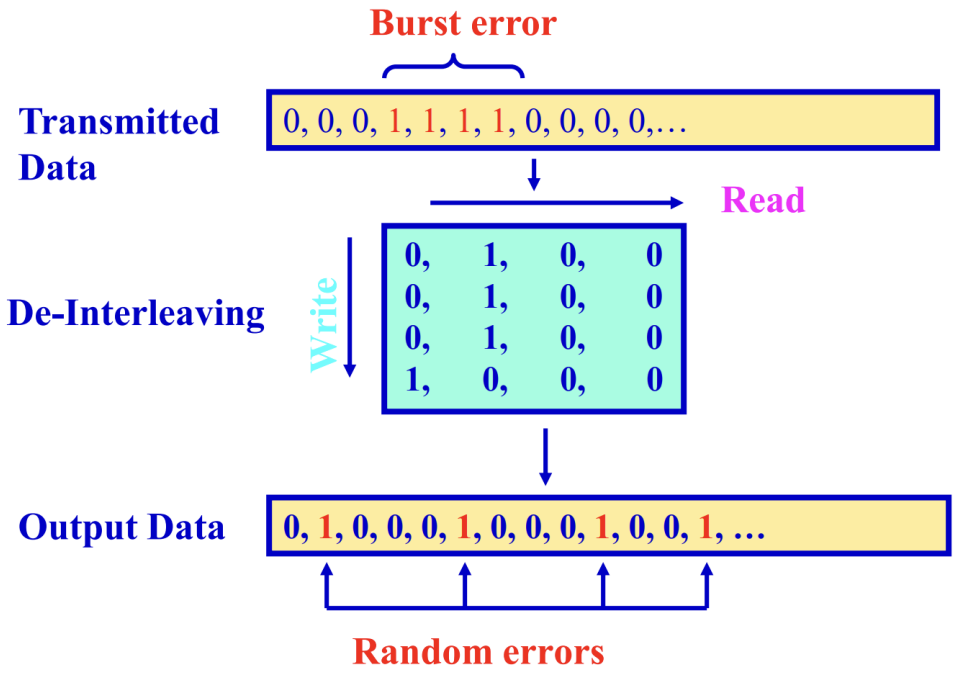
- Error correcting capability
- 오류 정정 능력
- 4bit + 3bit(redundancy)는 최대 1개의 에러 감지 가능
- 하나의 block 당 하나의 오류 해결
  - 다음과 같은 경우는 retransmit 필요
  - Burst error
  - 일정 시간 내에 생성된 많은 에러
  - 수신 파워가 작아질 때(like fading case)
  - 실제 시스템에서는 burst error 빈도 높음
  - Burst error + individual error



- Interleaver
  - 에러의 위치를 퍼뜨려 에러의 수는 변함 없되, 오류 정정 능력의 효과 극대화
  - Redundancy 발생하지 않음
  - Latency 발생
  - 일정 코드가 쌓일 때까지 기다려야 함
  - Delay-sensitive한 서비스의 경우 메모리 (일정 코드를 쌓는 양)을 줄일 필요 있음




Input Data
a1, a2, a3, a4, a5, a6, a7, a8, a9, …
Write
Read
Interleaving
a1, a2, a3, a4
a5, a6, a7, a8
a9, a10, a11, a12
a13, a14, a15, a16
Transmitting Data
a1, a5, a9, a13, a2, a6, a10, a14, a3, …
Through Air
Received Data
a1, a5, a9, a13, a2, a6, a10, a14, a3, …
Read
De-Interleaving
Write
a1, a2, a3, a4
a5, a6, a7, a8
a9, a10, a11, a12
a13, a14, a15, a16
Output Data
a1, a2, a3, a4, a5, a6, a7, a8, a9, …


Burst error
Transmitted Data
0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,…
Read
De-Interleaving
Write
0, 1, 0, 0
0, 1, 0, 0
0, 1, 0, 0
1, 0, 0, 0
Output Data
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, …
Random errors

- Turbo codes

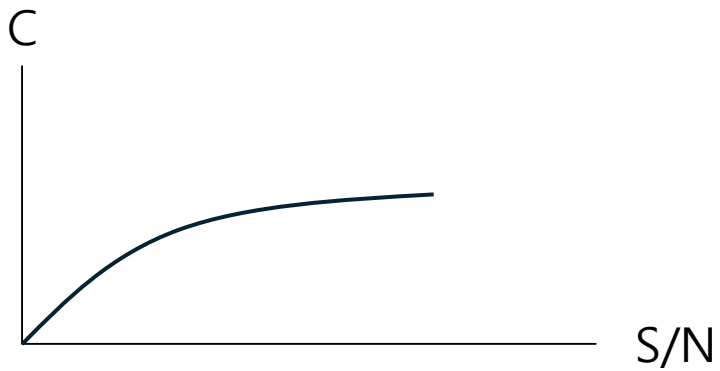
- 3G
  - HSDPA – high speed downlink packet access
  - HSUPA – high speed uplink packet access
  - HSPA (HSPA evolution = HSPA+)

- 4G
  - LTE (long term evolution)
  - LTE-A (advanced)
  - LTE-A pro

- 5G
  - LDPC codes
  - Can be done with parallel processing

- Shannon
  - Channel capacity
  - $C = W \cdot \log_2\left(1 + \frac{S}{N}\right)$
    - C = maximum BPS
    - W = bandwidth(f)

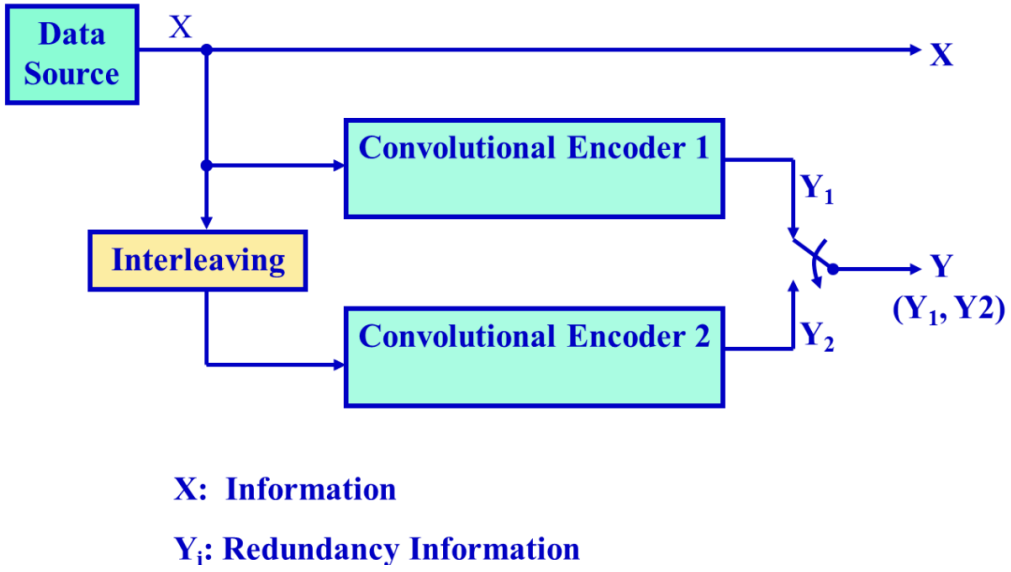
- 이상적인 이론일 뿐 실제와는 큰 격차 존재
- By using turbo code, we can calculate real case

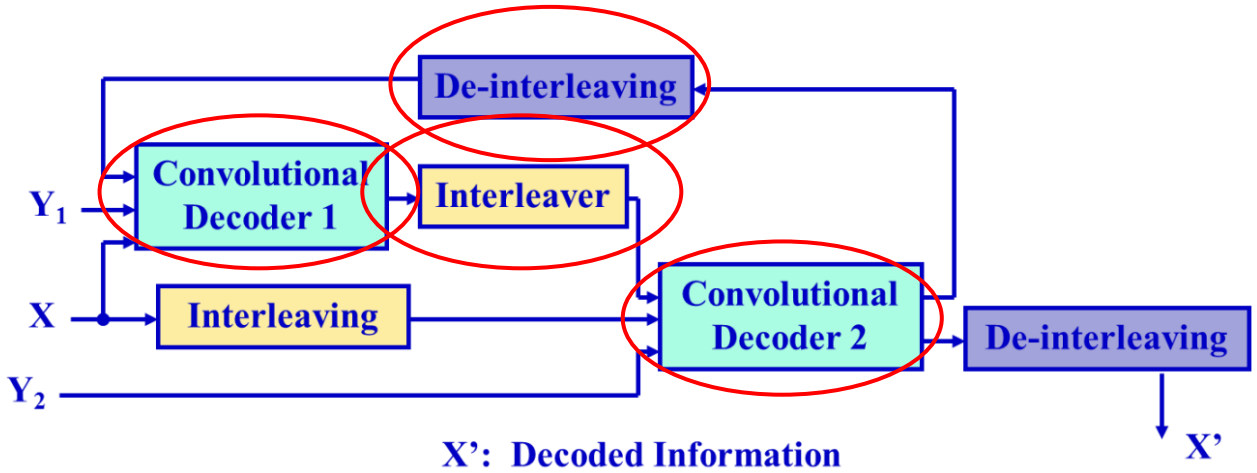


- Turbo codes
  - R = 1/3
  - Iterative decoding
  - 반복되는 디코딩 사용
  - 4 element iteratively work
  - Serial processing

- Serial processing = 직렬 = CPU
- Parallel processing = 병렬 = GPU

- Turbo codes는 고성능 CPU 사용
- 현재 5G는 GPU 사용

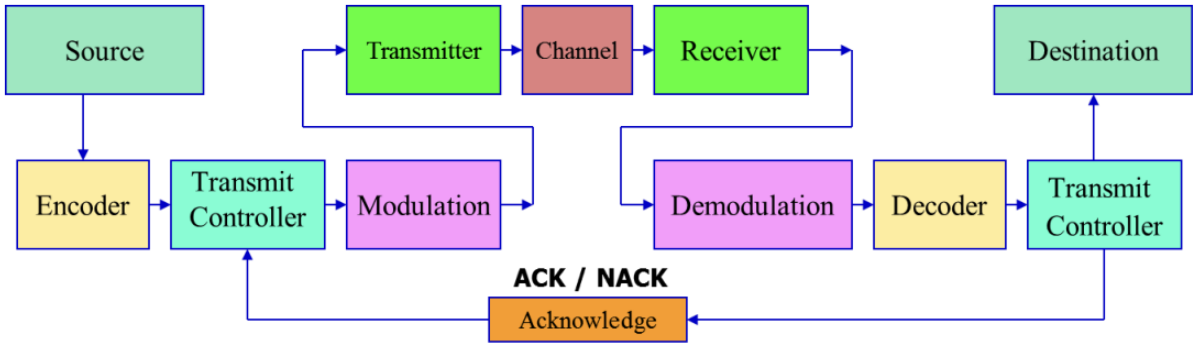


- ARQ
  - Automatic repeat request

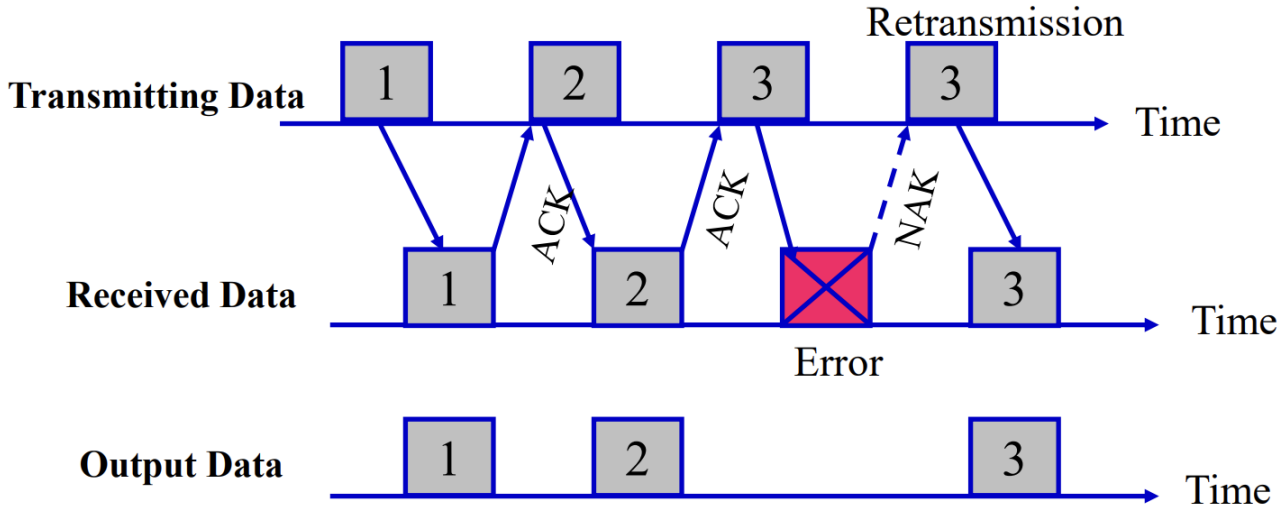
- ACK(acknowledgement) = No error
- NACK(negative ack) = error

- Stop-And-Wait ARQ (SAW ARQ)
- Go-Back-N ARQ (GBN ARQ)
- Selective-Repeat ARQ (SR ARQ)



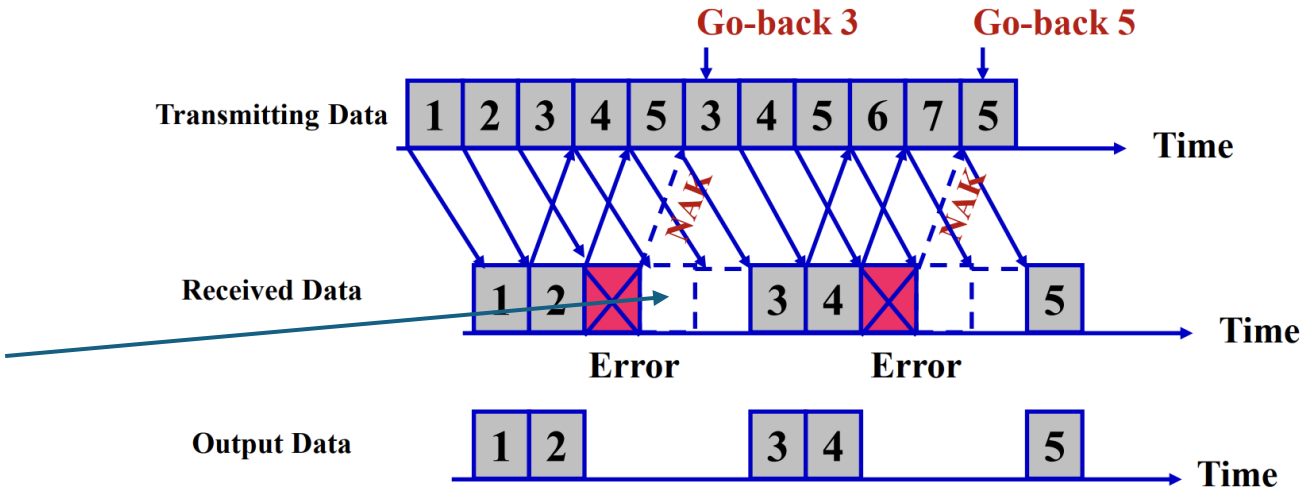
SAW ARQ

- 구현하기 쉬움
- Sending → Wait = takes long time
- 효율이 매우 떨어지는 구조



GBN ARQ

- 오류가 발생한 해당 패킷으로 이동 후 순차적으로 다시 전송
- 항상 패킷을 전송하는 상태
- 일정 수량의 패킷을 이용(n)
- 또한 올바르게 전송된 신호도 무시됨
- 잘못된 패킷 이상 전송
- 불필요한 재전송 발생

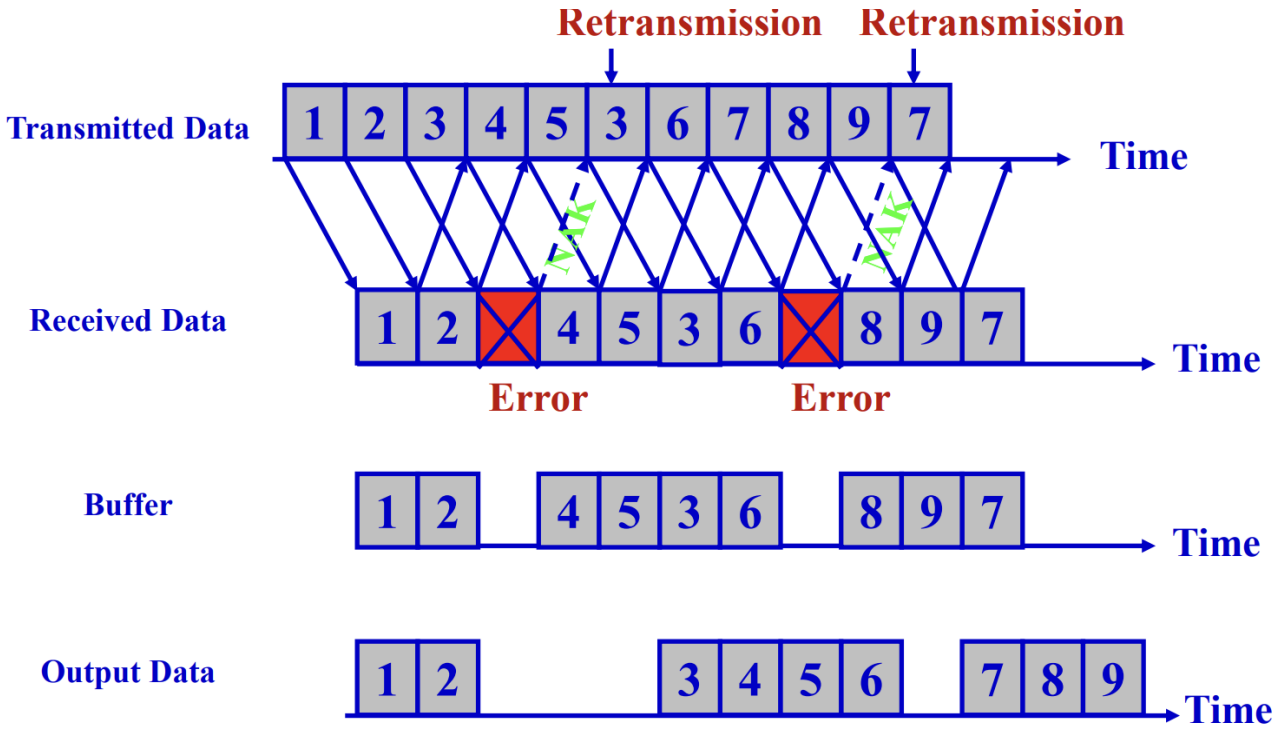



- SAW에서 Transmit을 채우자…

SR ARQ

- 가장 복잡하면서 효과적인 구조
- GBN ARQ에서 error 패킷만 재전송
- 버퍼에 데이터가 도착한 순서대로 임시 저장 후, 실제로는 정렬되어 저장
- 불필요한 전송 방지

- 이제 Receive까지 채우자…



6

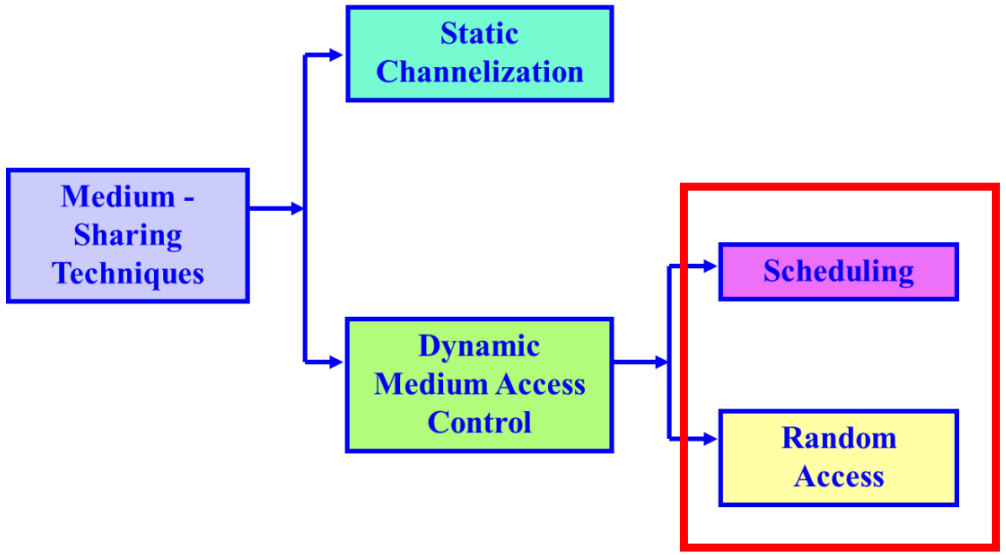
- BS to MS(terminal)
- 64(bit) sequencies, from terminal there's a lot of Preambles(통신 전 동기화 신호 요청)

- Choose one sequence
- Random access

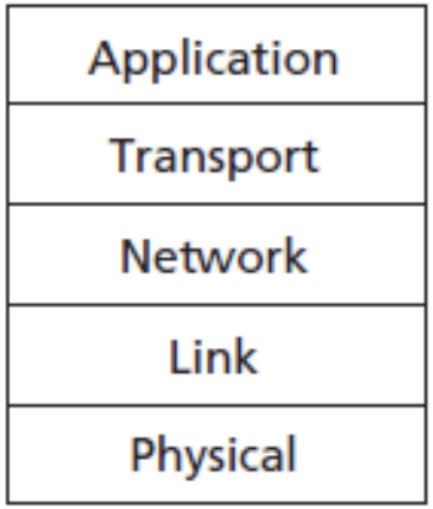
- Many terminals send many preambles each, like competition, only one will be accepted

- Collision = 두 transmitter가 동시에 보내는 정보에 의한 충돌

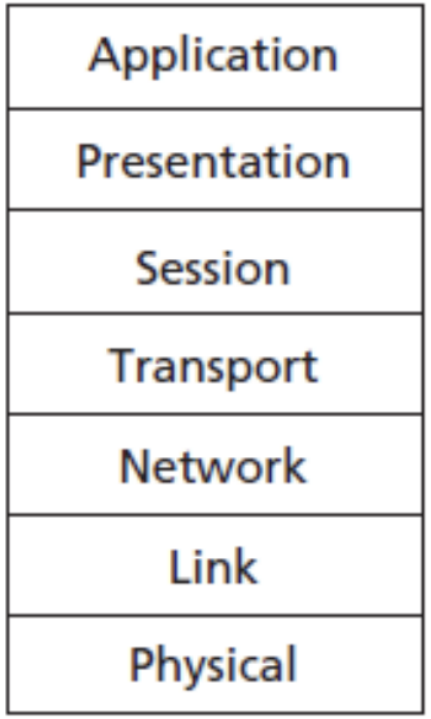
- Static allocation = 정적 할당
  - 채널 지정이 한번 지정되면 변화X
- Dynamic allocation = 동적 할당
  - 채널이 필요에 따라 변화




실제 5G에서 많은 경우 함께 사용




a. Five-layer Internet protocol stack




b. Seven-layer ISO OSI reference model

- Network
- 노드의 연결 확정

- Link
- 노드와 노드 간 연결

- L2 in 4G
  - PDCP (Packet Data Convergence Protocol)
  - RLC (Radio Link Control)
  - MAC (Medium Access Control)

- Random Access (Procedure)
  - IDLE state → Active state
  - 비 통신 상태에서 통신 상태로의 변경
  - Random Access preamble 를 사용
  - To find BS(디바이스에서 기지국 찾기)
  - 네트워크에 접속 전, 보내는 신호

- Page message
  - To find MS(네트워크에서 디바이스 찾기)

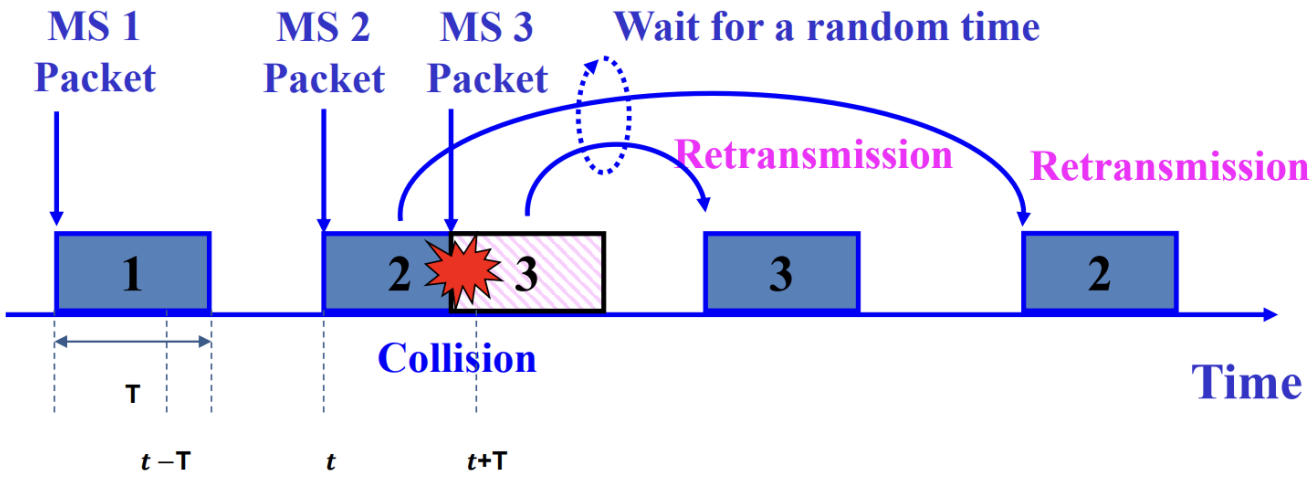
- Multiple access를 이용
- 무선접속 프로토콜

- Pure ALOHA
- Slotted ALOHA
- CSMA
- CD

- Pure ALOHA

- 전송 중 발생한 패킷간 충돌
- 올바르게 전송되지 않은 에러를 포함한 신호
- 임의의 재전송 → 순서를 따르지 않아도 됨
- 낮은 퍼포먼스
- 충돌에 의해 실제 패킷 개수보다 더 많이 전송
- 2T(vulnerable period)동안 패킷이 없으면 $e^{-2gT}$를 따름
- Throughput=전체 시간에서 실제 유용한 패킷 전송 시간의 비율
- 100분 중 18분만이 최대 실제 전송 시간

- Collision Mechanism in Pure ALOHA



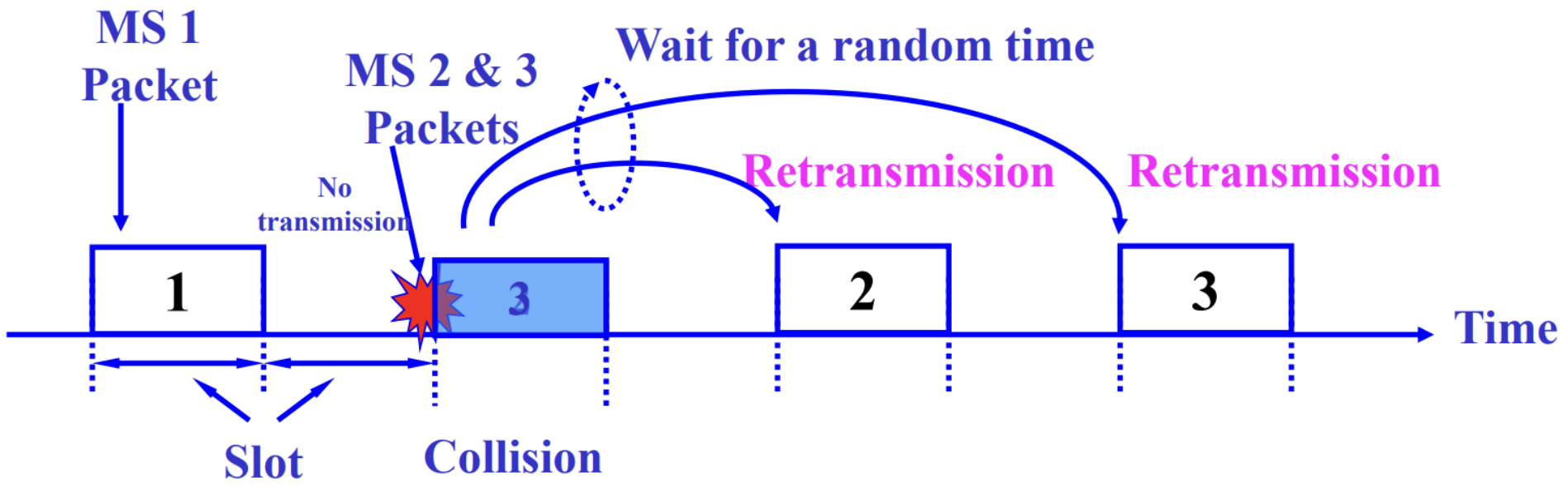
- The probability of successful transmission $P_s$
  - The probability no other packets scheduled between the instants $t - T$ and $t + T$ (interval of length $\boldsymbol{2T}$: **vulnerable period**)
  - $P_s = P(\text{no collision}) = P(\text{no transmission in } 2T) = e^{-2gT}$
- The rate of successful transmission = $gP_s$ (successful packets/sec)
- **Throughput $\boldsymbol{S_{th}}$** is defined as "the fraction of time during which the useful information is carried on the channel."
  - $S_{th} = T \cdot gP_s = gTe^{-2gT}$
- Defining normalized offered load to the channel: $G = gT$
  - $S_{th} = Ge^{-2G}$
  - $\frac{dS_{th}}{dG} = -2Ge^{-2G} + e^{-2G} = 0 \quad \Rightarrow S_{th\,max}$ occurs at $G = \frac{1}{2}$
  - **Maximum throughput of pure ALOHA: $S_{th\,max} = \frac{1}{2e} \approx 0.184$**

- Slotted ALOHA

- Pure ALOHA의 성능 증가를 위해 개발
- 충돌의 확률을 줄이기 위해 미리 정해진 slot(timeline)에만 전송
- 부분 충돌은 절대 발생하지 않음
- 완전한 충돌만 발생
- $e^{-gT}$를 따름
- 2배 더 증가된 성능(덜 일어나는 충돌)

하나의 라우터에 의해 MS로 전송되는 패킷

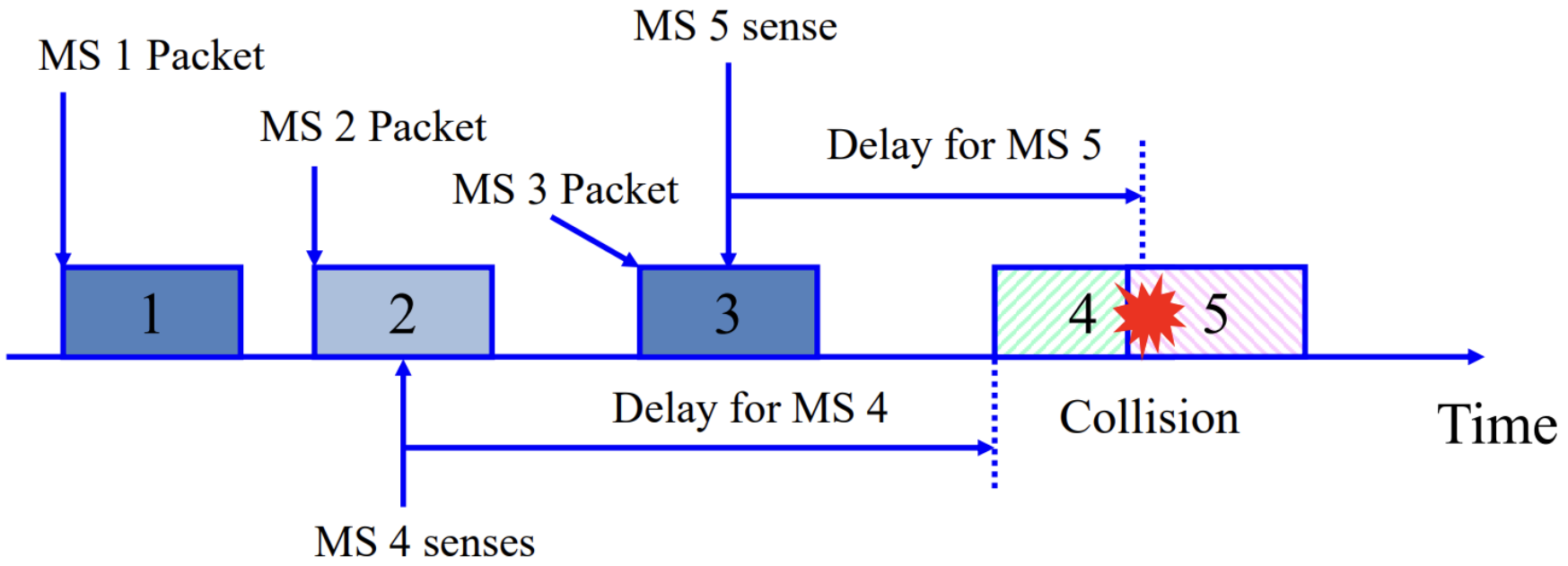
이산 → Poisson process
연속 → Exponential pdf

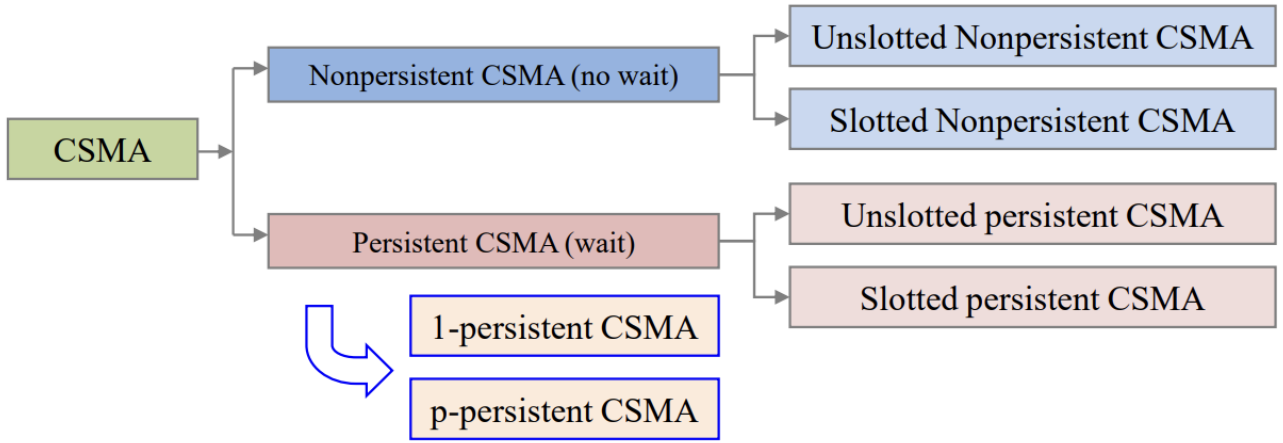


- The probability of successful transmission $P_s$
  - The probability no other packets scheduled between the instants $t$ and $t+T$ (interval of length $\boldsymbol{T}$: **vulnerable period**)
  - $P_s = P(\text{no collision}) = P(\text{no transmission in } T) = e^{-gT}$
- The rate of successful transmission = $gP_s$ (successful packets/sec)
- **Throughput $S_{th}$**
  - $S_{th} = T \cdot gP_s = gTe^{-gT}$
- Defining normalized offered load to the channel: $G = gT$
  - $S_{th} = Ge^{-G}$
  - $\frac{dS_{th}}{dG} = -Ge^{-G} + e^{-G} = 0 \quad \Rightarrow S_{th\,max}$ occurs at $G = 1$
  - **Maximum throughput of slotted ALOHA: $S_{th\,max} = \frac{1}{e} \approx 0.368$**

- CSMA
- Carrier Sense Multiple Access

- 모바일 4번이 2번을 감지하고 기다린 후, 모바일 5번도 3번을 감지하고 기다림
- Random한 시간을 기다리기 때문에 충돌하는 경우가 생김
- Persistent하지 않고 Non-Persistent한 상태 (기다린 후 돌아올 때 대기줄이 유지 되는가?)
- Persistent → 1-persistent(대기줄에서 기다림), p-persistent(일정 확률을 이용하여 전송)






- CD (collision detection)
- CSMA는 데이터를 충돌과 관계없이 진행하기에 CD가 필요

- CA (collision avoidance)
- 전송이 조용할 때까지 CA를 이용하는 방법도 존재

- Non-Persistent
- 대기줄을 기다리지 않음
- 인지 → 복귀 → 대기 → 다시 확인

- Persistent
- 대기줄을 기다림
- 인지 → 대기 → 전송

- 1-persistent
  - 먼저 접근한 사람이 기다리고 끝나는 순간 바로 전송

- P-persistent
  - 사람이 많다면, 전송 확률 p를 낮추기
  - 사람이 적다면, 전송 확률 p를 높이기
  - 10명의 유저, p값 0.3 → 3명의 충돌
  - 10명의 유저, p값 0.1 → 충돌 방지
  - (Number of user * p <= 1)이 이상적