Chapter 4
Channel Coding and Error Control

Dept. of Mobile Systems Engineering
Dankook University

Suhan Choi

Outline

- 4.1 Introduction
- 4.2 Linear Block Codes
- 4.3 Cyclic Codes
- 4.4 CRC (Cyclic Redundancy Check)
- 4.5 Convolutional Codes
- 4.6 Interleaver
- 4.7 Turbo Codes
- 4.8 ARQ (Automatic Repeat reQuest) Techniques

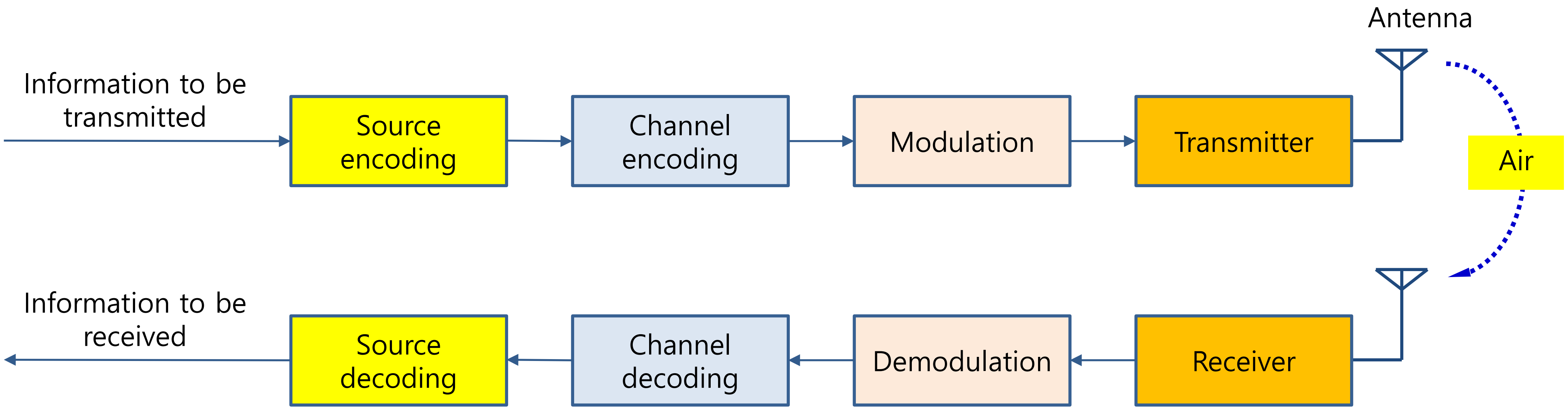
4.1 Introduction

- Why do we need channel coding and error correction for radio communication?
  - Severe transmission conditions in mobile radio communications due to multipath fading and very low SNR (Signal to Noise Ratio)
- Channel Coding
  - Adds **redundant** (or redundancy) information to the original information at the Tx side, following some logical relation with the original information.
  - At the Rx side, the original information can possibly be extracted from received data based on the logical relationship between original info. and redundancy information.
  - **Redundancy**
    - Causes channel coding to consume more bandwidth during the transmission
    - However, it offers benefits of recovering from higher bit error rates.

    ⇒ **can correct or detect errors**

4.1 Introduction

- Channel coding in a wireless comm. systems.



- Categories of FEC codes
  - Block codes / Cyclic codes / Reed-Solomon codes
  - Convolutional codes / Turbo codes, etc.

4.2 Linear Block Codes

- Information is divided into **blocks of length k**
- r parity bits or check bits are added to each block
  (total length n = k + r)
- **Code rate R = k/n**
- Decoder looks for codeword closest to received vector (code vector + error vector)
- Tradeoffs between
  - Efficiency
  - Reliability
  - Encoding/Decoding complexity

4.2 Linear Block Codes

- Modulo 2 Addition

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ \underline{0} & \underline{1} & \underline{0} & \underline{1} \\ 0 & 1 & 1 & 0 \end{array}$$

- (n, k) block code
  - k information bits are encoded into n bits.
  - **2^k valid codewords** from a subset of the 2^n possible bit patterns
  - The uncoded k information bits $\Rightarrow$ $\mathbf{m}$ vector:

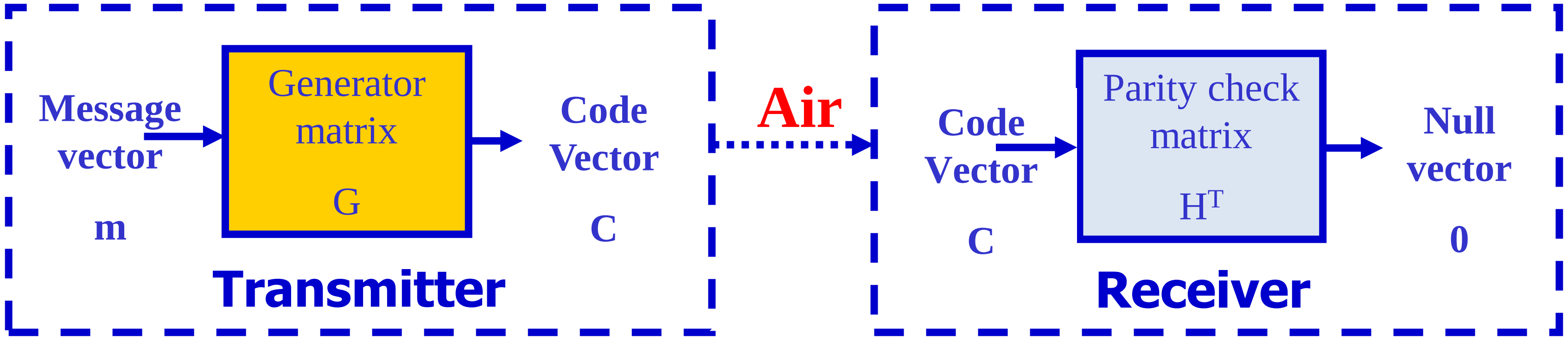
$$\mathbf{m} = (m_1, m_2, \dots, m_k)$$

  - Corresponding n-bit codeword

$$c = (c_1, c_2, \dots, c_k, c_{k+1}, \dots c_{n-1}, c_n)$$

4.2 Linear Block Codes

- Operations of the generator matrix and the parity check matrix



  - To add parity to the information bits at the Tx side using **the generation matrix G.** ⇒ **$c = mG$**
  - To use **the parity check matrix** to take care of possible errors during transmission

4.2 Linear Block Codes

- Each parity bit consists of a weighted modulo 2 sum of the data bits represented by $\oplus$ symbol.
- codeword: $\mathbf{c=mG}$
  - **G**: generator matrix with dimension (k x n)
  - **G** = [$\mathbf{I}_k$ | **P**]

$$\begin{cases} c_1 = m_1 \\ c_2 = m_2 \\ \ldots \\ c_k = m_k \\ c_{k+1} = m_1 p_{11} \oplus m_2 p_{21} \oplus \ldots \oplus m_k p_{k1} \\ \ldots \\ c_n = m_1 p_{1(n-k)} \oplus m_2 p_{2(n-k)} \oplus \ldots \oplus m_k p_{k(n-k)} \end{cases}$$

Parity Matrix P

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1(n-k)} \\ 0 & 1 & 0 & \cdots & 0 & p_{21} & p_{22} & \cdots & p_{2n-k)} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{k1} & p_{k2} & \cdots & p_{k(n-k)} \end{bmatrix}$$

- Parity Matrix P (with dimension k x (n-k))

$$P = \begin{bmatrix} p_{11}\ p_{12} \cdots p_{1(n-k)} \\ p_{21}\ p_{22} \cdots p_{2(n-k)} \\ \cdots\cdots\cdots\cdots\cdots \\ p_{k1}\ p_{k1} \cdots p_{k(n-k)} \end{bmatrix} = \begin{bmatrix} P^1 \\ P^2 \\ \cdots \\ P^k \end{bmatrix}$$

where

$$P^i = \mathrm{rem}\left[\frac{x^{n-k+i-1}}{g(x)}\right], \quad \text{for} \quad i = 1, 2, \cdots, k$$

- g(x): generator polynomial

4.2 Linear Block Codes

- [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

$$
\left.
\begin{aligned}
p_1 &= rem\left[\frac{x^3}{1+x+x^3}\right] = 1+x \rightarrow [110] \\
p_2 &= rem\left[\frac{x^4}{1+x+x^3}\right] = x+x^2 \rightarrow [011] \\
p_3 &= rem\left[\frac{x^5}{1+x+x^3}\right] = 1+x+x^2 \rightarrow [111] \\
p_4 &= rem\left[\frac{x^6}{1+x+x^3}\right] = 1+x^2 \rightarrow [101]
\end{aligned}
\right\}
\quad
G = \begin{bmatrix} 1000 | 110 \\ 0100 | 011 \\ 0010 | 111 \\ 0001 | 101 \end{bmatrix} = [I \,|\, P]
$$

I is the identity matrix
P is the parity matrix

4.2 Linear Block Codes

- The Generator Polynomial can be used to determine the Generator Matrix **G** that allows determination of parity bits for a given data bits of m by multiplying as follows:

$$mG = \underset{\uparrow \atop \textbf{Data}}{[1011]} \begin{bmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{bmatrix} = [\underset{\uparrow \atop \textbf{Data}}{1011} | \underset{\uparrow \atop \textbf{Parity}}{100}]$$

  - Can be done for other combination of data bits, giving the code word **c**
  - Other combinations of **m** can be used to determine all other possible code words

4.2 Linear Block Codes

- **Parity Check Matrix: H**

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

- Received code vector x = c ⊕ e,

where e is an error vector, the matrix H^T has the property

$$\mathbf{c}\mathbf{H}^T = [\mathbf{m} | \ \mathbf{c}_p] \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}$$
$$= \mathbf{m}\mathbf{P} \oplus \mathbf{c}_p = \mathbf{c}_p \oplus \mathbf{c}_p = \mathbf{0}$$

- **The transpose of matrix $\mathbf{H}^{\mathbf{T}}$ is** $\mathbf{H} = [\mathbf{P}^T \ \mathbf{I}_{n-k}]$

4.2 Linear Block Codes

- **Syndrome:** $\mathbf{S} = \mathbf{x}\mathbf{H}^{\mathbf{T}} = (\mathbf{c} \oplus \mathbf{e})\mathbf{H}^{\mathbf{T}} = \mathbf{c}\mathbf{H}^{\mathbf{T}} \oplus \mathbf{e}\mathbf{H}^{\mathbf{T}} = \mathbf{e}\mathbf{H}^{\mathbf{T}}$
  - If there are **no errors (e = 0), then s = 0.**
  - **There are errors (e $\neq \mathbf{0}$), if s $\neq$ 0.**
  - **s** has (n-k) dimensions.
- Example: For the (7,4) linear block code, given by **G** as

$$G = \begin{bmatrix} 1000\,|\,111 \\ 0100\,|\,110 \\ 0010\,|\,101 \\ 0001\,|\,011 \end{bmatrix} \qquad H = \begin{bmatrix} 1110\,|\,100 \\ 1101\,|\,010 \\ 1011\,|\,001 \end{bmatrix}$$

  - **m** = [1 0 1 1] and **c** = **mG** = [1 0 1 1| 0 0 1]
  - If there is no error, the received vector **x**=**c**, and **s**=**cH**^{T}=[0, 0, 0]

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]

4.3 Cyclic Codes

- Linear block codes with a cyclic structure

  ⇒ leads to more practical implementation

- An advantage of cyclic codes
  - Relatively easy to encode and decode

- Uses a shift register to perform encoding and decoding

- The code word with n bits is expressed as:

$$c(x)=c_1x^{n-1}+c_2x^{n-2}\ldots\ldots+c_n$$

where each coefficient c_i ($i=1,2,..,n$) is either a 1 or 0

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

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

where $c_p(x)$ = remainder from dividing $m(x)\, x^{n\text{-}k}$ 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 $s(x) \neq 0$, then there is an error.

4.3 Cyclic Codes (Example)

- 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

4.4 Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Check (CRC)
  - An error-checking code
  - Widely used in communication systems
  - The transmitter appends an extra n-bit sequence to every frame called **Frame Check Sequence (FCS)**.
  - The FCS holds redundant information about the frame that helps the receivers detect errors in the frame.
- CRC is based on polynomial manipulation using modulo arithmetic.
  - **Message polynomial:** input bit as coefficients of the polynomial
    - Ex: binary 10111 $\Rightarrow 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$
  - **Generator polynomial:** polynomial with constant coefficients

4.4 Cyclic Redundancy Check (CRC)

- Generator polynomial divides into the message polynomial, giving quotient and remainder, the coefficients of the remainder form the bits of final CRC
- Define:
  - Q : The original frame (k bits) to be transmitted
  - P : The predefined CRC generating polynomial
  - F : The resulting frame check sequence (FCS) of n-k bits to be added to Q (usually n=8, 16, 32)
  - J : The cascading of Q and F
- **The main idea in CRC algorithm:**

  ⇒ FCS is generated so that J should be exactly divisible by P

4.4 Cyclic Redundancy Check (CRC)

- The CRC creation process is defined as follows.
  - Get the block of raw message
  - Left shift the raw message by **n-k** bits and then divide it by P
  - Get the remainder R as FCS (i.e., F)
  - Append the R to the raw message.

    The result J is the frame to be transmitted. $J=Q\,x^{n-k}+F$
  - J should be exactly divisible by P
- Dividing $Q.x^{n-k}$ by P gives $Qx^{n-k}/P=Q'+R/P$
  - Where Q' is quotient, R is the remainder.
  - $J=Qx^{n-k}+R$

    ⇒ This value of J should yield a zero remainder for J/P

4.4 Cyclic Redundancy Check (Example)

- 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**

4.4 Cyclic Redundancy Check (CRC)

- Most Commonly Used CRC Polynomials

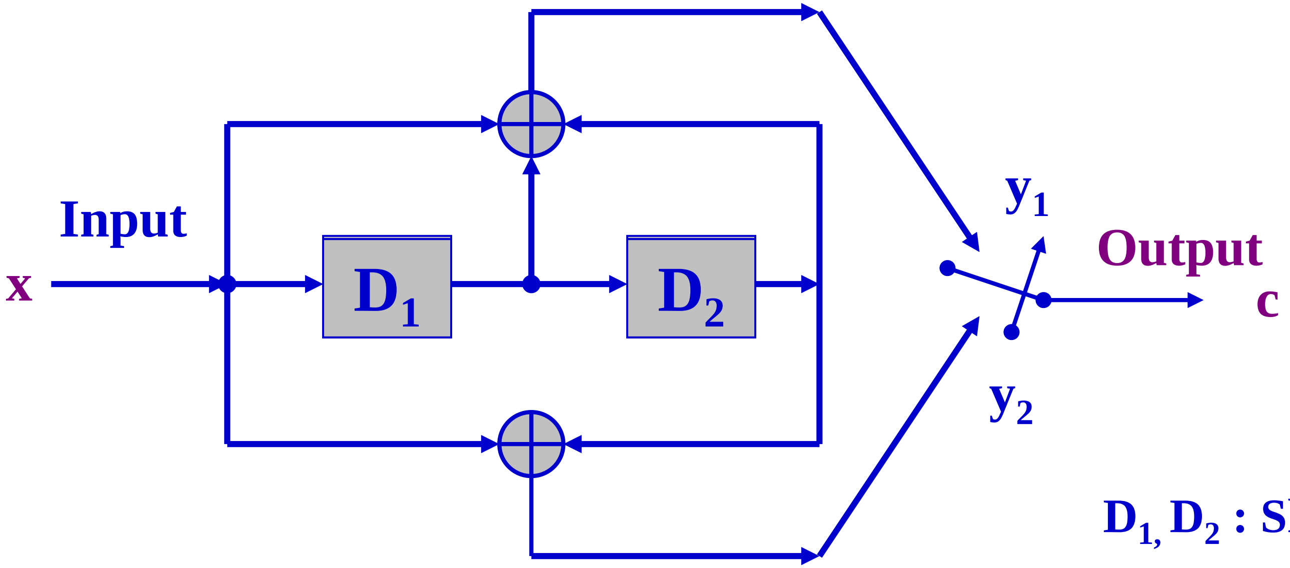
CRC Types	Generator polynomial g(x)	Parity Bits
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$	12
CRC-16	$x^{16}+x^{15}+x^2+1$	16
CRC-CCITT	$x^{16}+x^{12}+x^5+1$	16
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	32

4.5 Convolutional Codes

- Most widely used channel code in practical communication systems.
  - GSM, CDMA (IS-95), WCDMA, HSPA, LTE, etc.
  - Primarily used for real-time error correction
- Encoding of information stream rather than information blocks
  - The encoded bits depend not only on the current input bits but also on past input bits.
- Decoding is mostly performed by the **Viterbi Algorithm**
- The **constraint length K** for a convolution code is defined as **K=M+1**
  - M : the maximum number of stages in any shift register
- The **code rate r** is defined as **r = k/n**
  - k : the number of parallel information bits
  - n : the number of parallel output encoded bits at one time interval

4.5 Convolutional Codes

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

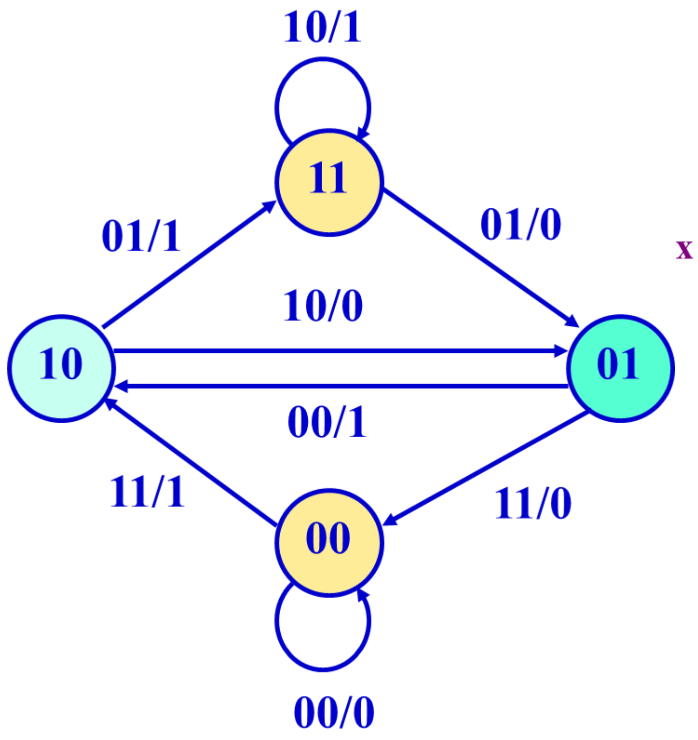


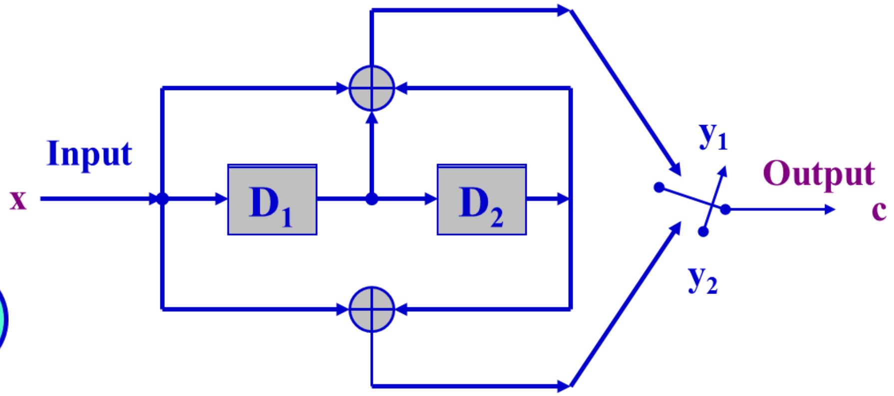
D1, D2 : Shift Registers

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

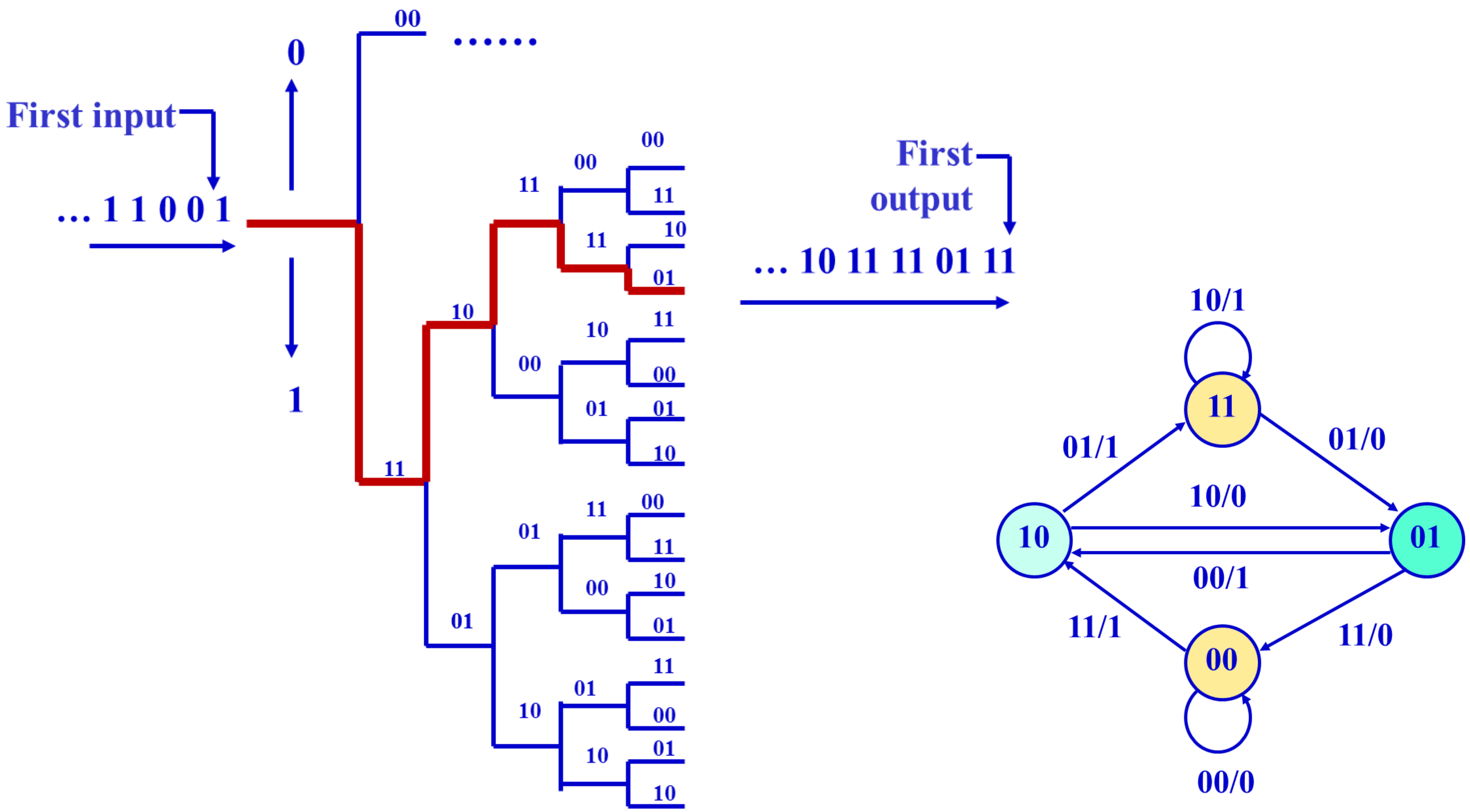
Input x:	1	0	1	0	0	0	…
Output y_1,y_2:	11	10	00	10	11	00	…

- State Diagram
  - Based on the input sequence, the state changes.
  - State transitions depend on the input bits and current state.



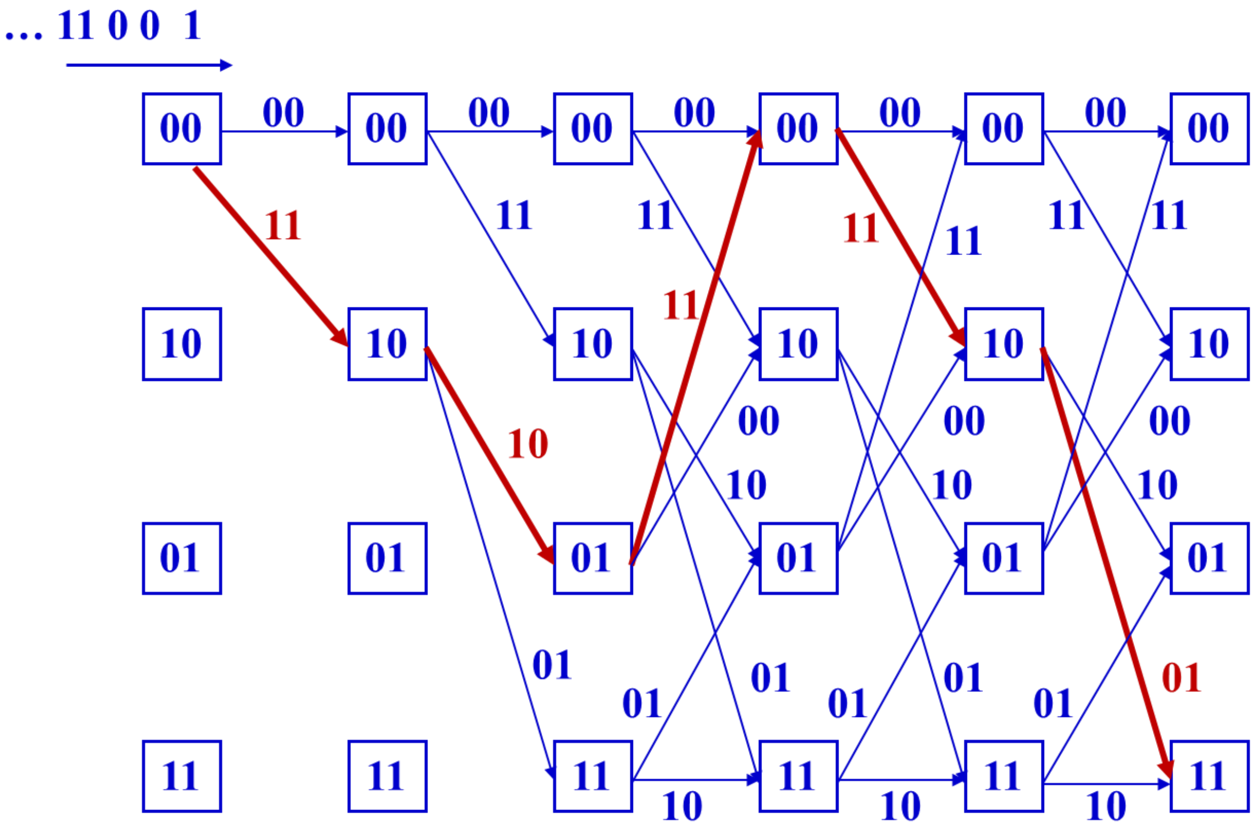


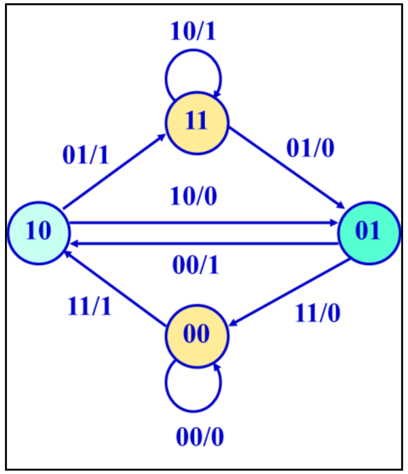
- Tree Diagram



4.5 Convolutional Codes

- Trellis Diagram





…

4.6 Interleaver

- Interleaving is heavily used in the wireless communications.
  - **Basic Objective**

    ⇒ to protect the transmitted data from **burst errors**

    ⇒ To disperse burst errors into multiple individual (or random) errors which can be handled by the error-correcting code.
- Many different interleavers
  - block / random / circular / semirandom / odd-even / optimal (near-optimal)
- Block interleaver
  - Most commonly used in wireless comm. systems.
  - Basic idea: to write data row-wise from top to bottom and left to right and read out column-wise from left to right and top to bottom.

4.6 Interleaver

Input Data: a1, a2, a3, a4, a5, a6, a7, a8, a9, …

Interleaving (Write →, Read ↓)

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

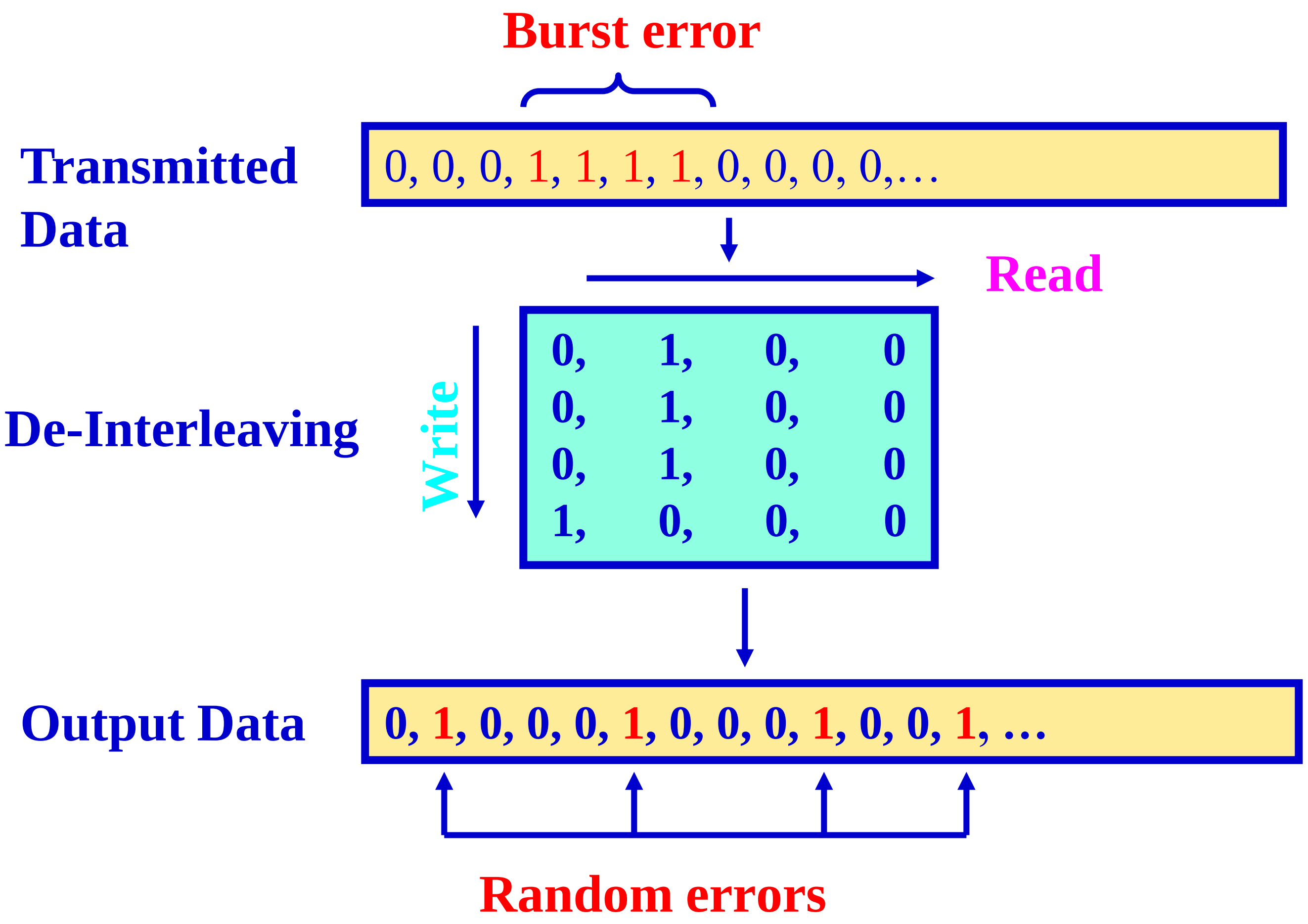
Received Data: a1, a5, a9, a13, a2, a6, a10, a14, a3, …

De-Interleaving (Read →, 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, …

4.6 Interleaver



4.6 Interleaver

- Interleaving dose not introduce any redundancy

  ⇒ Does not have error-correcting capability

  ⇒ Interleaving is always used in conjunction with an error-correcting code.

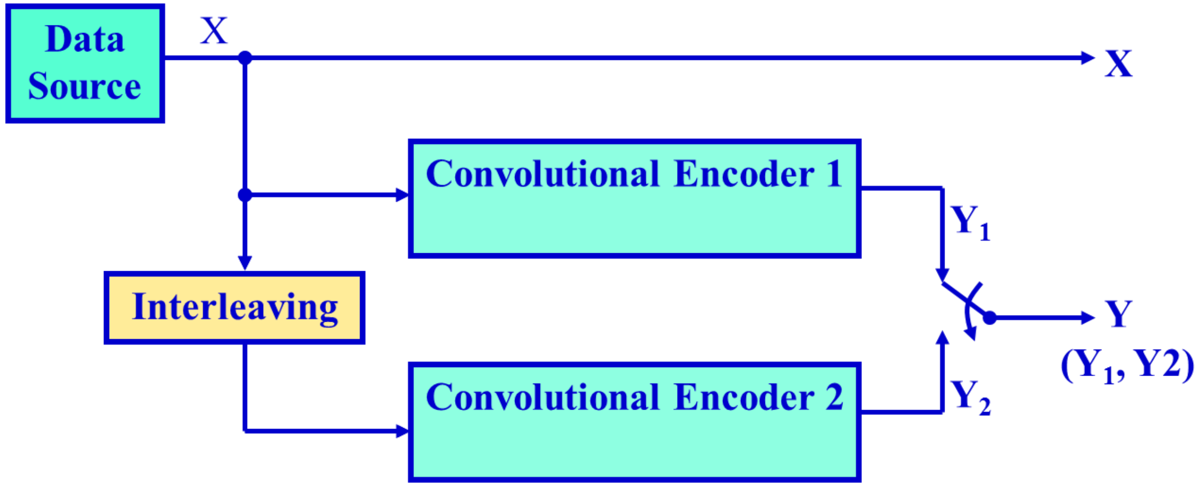
  ⇒ Does not add an extra bandwidth requirement

- Disadvantage of interleaving
  - Additional delay since the sequence needs to be processed block by block
  - Therefore, small memory size interleaving is preferred in delay-sensitive applications.

4.7 Turbo Codes

- Turbo codes are the most recently developed codes and are extremely powerful.
- A brief historic of turbo codes:
  - The turbo code concept was first introduced by C. Berrou in 1993.
  - Today, Turbo Codes are considered as the most efficient coding schemes for FEC.
- Scheme with known components (simple convolutional or block codes, interleaver, soft-decision decoder, etc.)
- Performance close to the Shannon Limit
- Turbo codes have been proposed for:
  - Low-power applications such as deep-space and satellite communications
  - Interference limited applications such as 3G/4G cellular, personal communication services, ad hoc and sensor networks

- Turbo Encoder
  - Built using two identical RSC (Recursive Systematic Convolutional) codes with parallel concatenation.
  - The interleaver randomizes the information sequence of the second encoder to make the inputs of the two encoders uncorrelated.

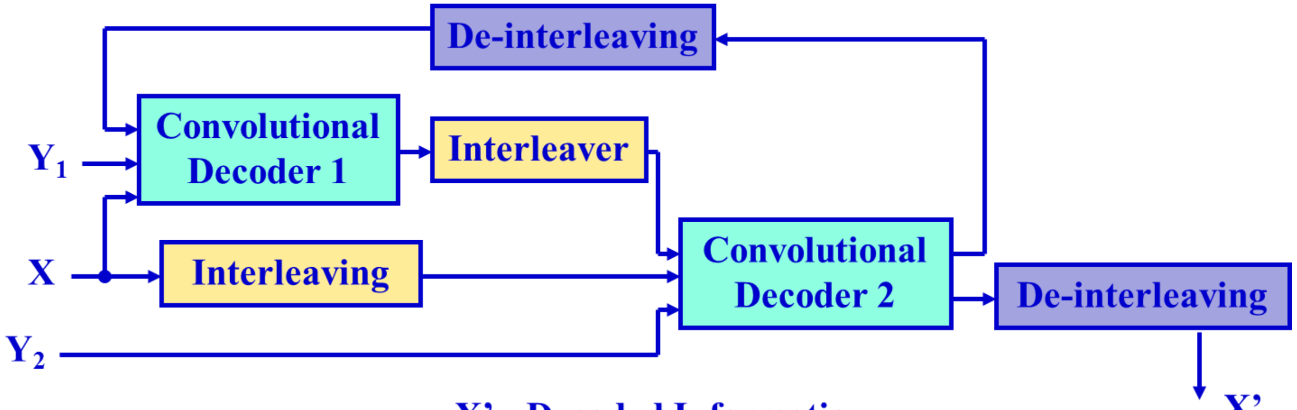


X: Information

Y_i: Redundancy Information

4.7 Turbo Codes

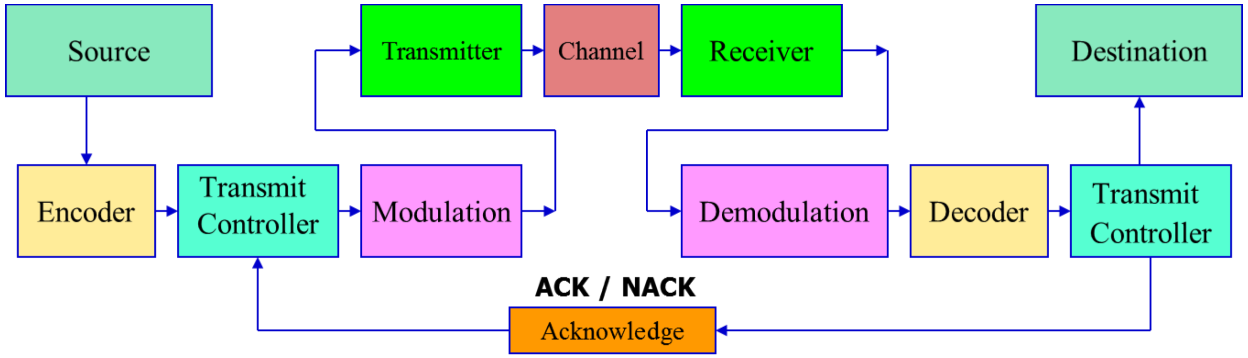
- Turbo Decoder
  - Since there are two encoded sequences, the turbo decoder consists of two RSC decoders corresponding to the two RSC encoded sequence respectively.
  - The decoding begins by decoding one of them to get the first estimate of the information sequence.
  - Based on the estimate from the 1st RSC decoder, the 2nd RSC decoder gets more precise estimate of the info. sequence.
  - To improve the correctness of the estimate, the estimate from the 2nd RSC decoder feeds back to the 1st RSC decoder continuously. ⇒ **“Turbo”**



X’: Decoded Information

4.8 ARQ Techniques

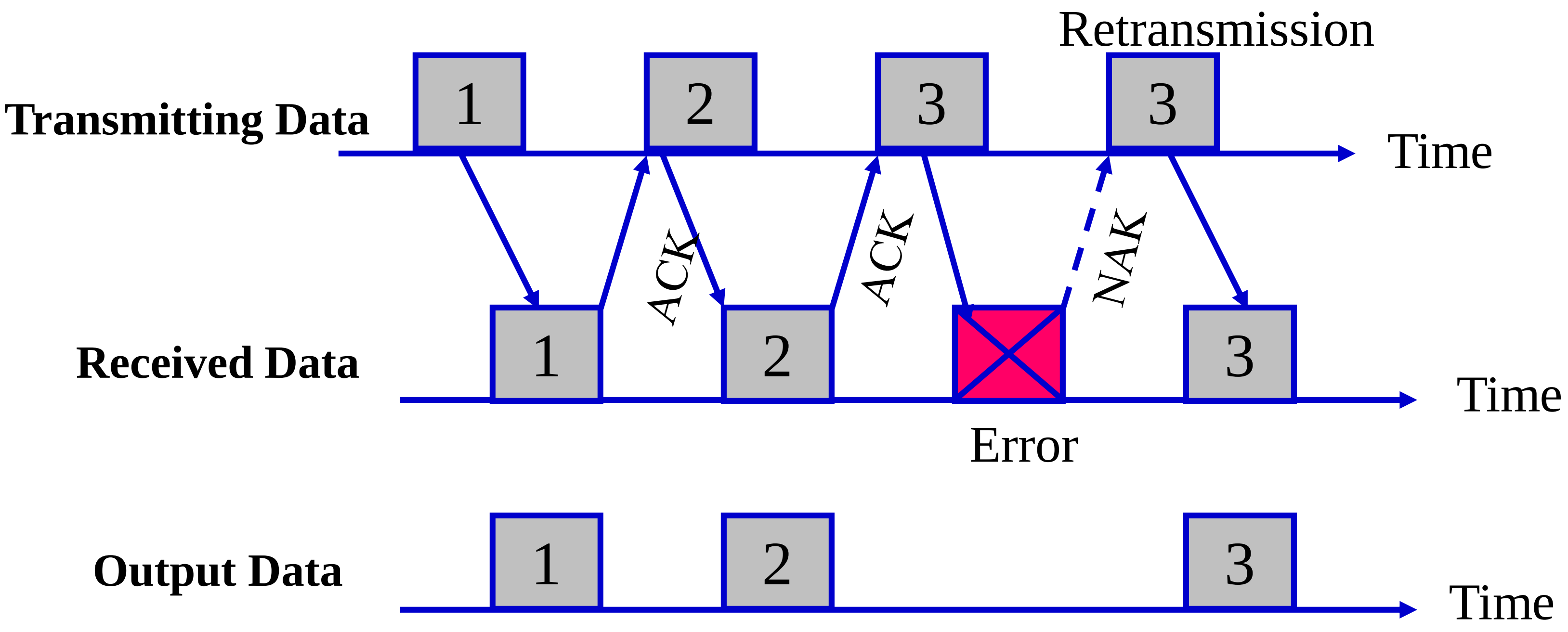
- The concept of ARQ (Automatic Repeat reQuest)
  - When the receiver detects errors in a packet (that cannot be corrected), it simply drops the packet and the sender needs to transmit it again.
  - No error ⇒ Rx sends **ACK** (acknowledgement)
  - Errors ⇒ Rx sends **NACK (or NAK)** (negative acknowledgement)



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

Stop-And-Wait ARQ

- The simplest ARQ scheme
  - The sender sends one data packet each time.
  - The receiver receives that data packet and checks if the data packet has been received correctly.
  - If the packet is not corrupted, the Rx sends an ACK packet; otherwise, NACK.



Stop-And-Wait ARQ

- The throughput for the SAW ARQ scheme.

$$S_{SAW} = \frac{1}{T_{SAW}}\left(\frac{k}{n}\right)$$

where n is the number of bits in a block,
k is the number of information bits in a block,
D is the round-trip propagation delay time,
T_{SAW} is the average transmission time in terms of a block duration.

- $T_{SAW} = \left(1+\frac{DR_b}{n}\right)P_{ACK} + 2\left(1+\frac{DR_b}{n}\right)P_{ACK}(1-P_{ACK}) + 3\left(1+\frac{DR_b}{n}\right)P_{ACK}(1-P_{ACK})^2 + \ldots$

$$= \left(1+\frac{DR_b}{n}\right)P_{ACK}\sum_{i=1}^{\infty} i(1-P_{ACK})^{i-1} = \left(1+\frac{DR_b}{n}\right)P_{ACK}\frac{1}{[1-(1-P_{ACK})]^2}$$

$$= \frac{1+\frac{DR_b}{n}}{P_{ACK}}$$

R_b: bit rate

Stop-And-Wait ARQ

- P_{ACK} : the probability to return ACK in the Rx side

$$P_{ACK} \approx (1 - P_b)^n$$

- Therefore, the throughput for the SAW ARQ scheme.

$$S_{SAW} = \frac{1}{T_{SAW}}\left(\frac{k}{n}\right) = \frac{(1-P_b)^n}{1+\frac{DR_b}{n}}\left(\frac{k}{n}\right)$$

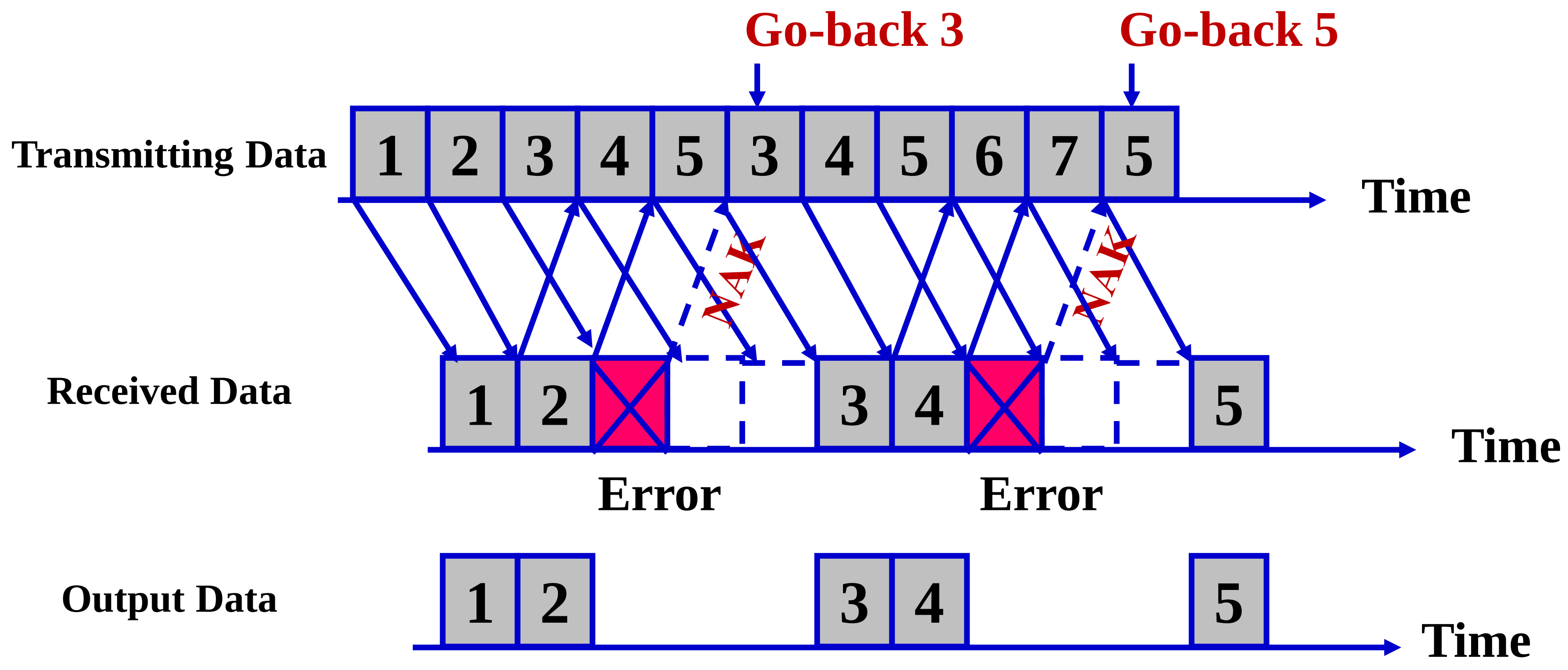
$$\sum_{i=1}^{\infty} ix^{i-1} = 1 + 2x + 3x^2 + \ \dots = \frac{1}{(1-x)^2}$$

$$\int \sum_{i=1}^{\infty} ix^{i-1}\, dx = x + x^2 + x^3 + \ \dots = \frac{x}{1-x}$$

$$\frac{d}{dx}\int \sum_{i=1}^{\infty} ix^{i-1}\, dx = \sum_{i=1}^{\infty} ix^{i-1} = \frac{1(1-x) - x(-1)}{(1-x)^2} = \frac{1}{(1-x)^2}$$

Go-Back-N ARQ

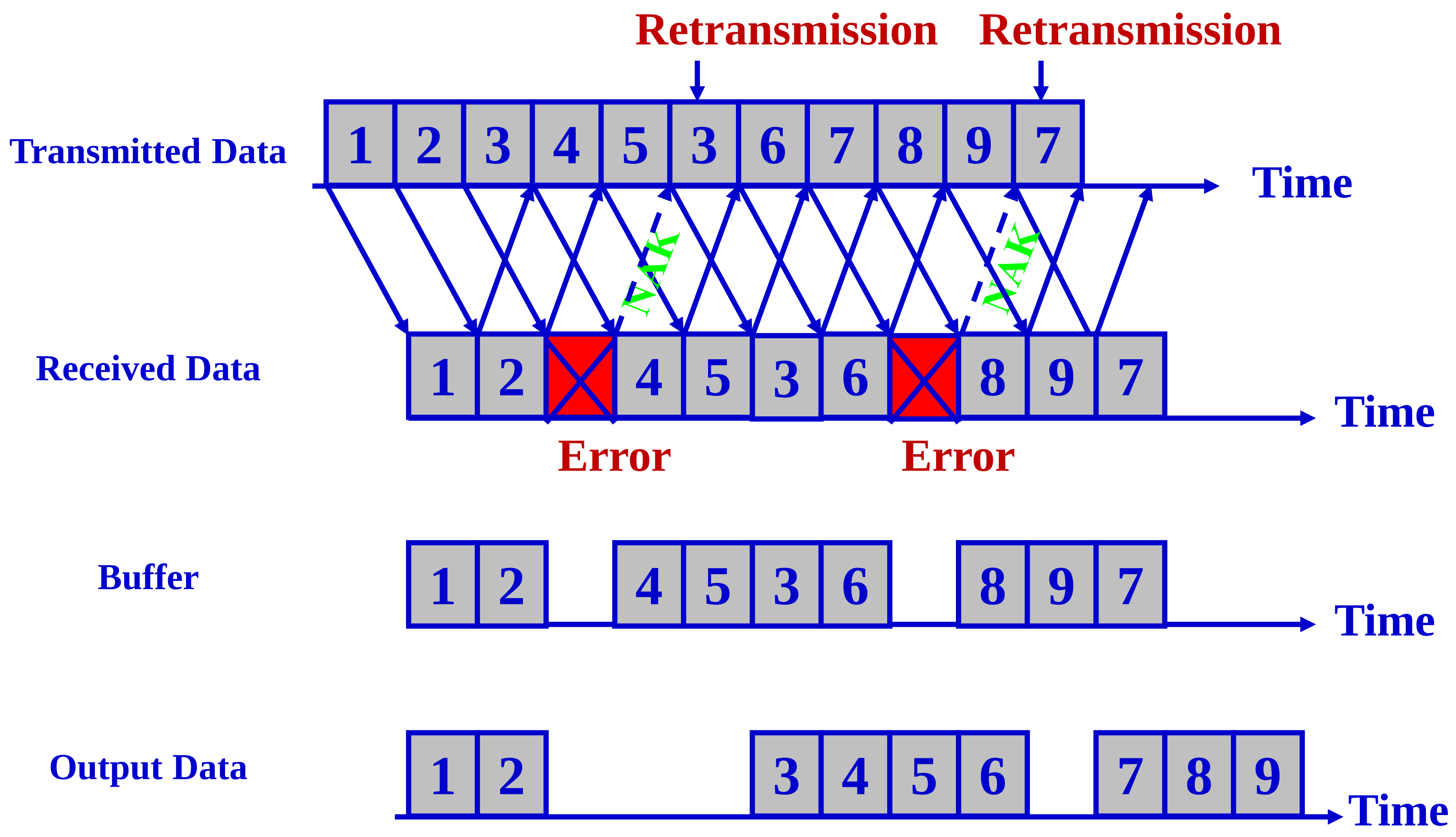
- SAW ARQ exhibits poor utilization of the channel since the Tx. does not send the next packet until it receives an ACK from the Rx.
- In the GBN ARQ scheme, the Tx is allowed to send N packets without waiting for acknowledgement of the prior packets.
- When Tx receives NACK from the Rx, the Tx has to retransmit all the packets that have been sent after that corrupted packet.



Selective-Repeat (SR) ARQ

- In the GBN ARQ scheme, a single packet error can cause the sender to retransmit several packets, most of which may be unnecessary.
- The SR ARQ provides improvement.
  - When the sender does not receive any ACK from a receiver, (due to packet loss or corruption), <u>it retransmits only that packet.</u>
    <u>⇒ avoids unnecessary retransmission.</u>
- Implementation of the SR ARQ protocol is more complex than that of the other two protocols.
- But, it provides the best efficiency.
- In practice, all of the three ARQ schemes should be implemented using a set of timers.
  - This is because both the data packet and the ACK/NACK packets may be lost during transmission.
  - If the sender cannot receive a response from the receiver in a certain prespecified time, it must retransmit those unacknowledged packets.

Selective-Repeat (SR) ARQ



Thank You !