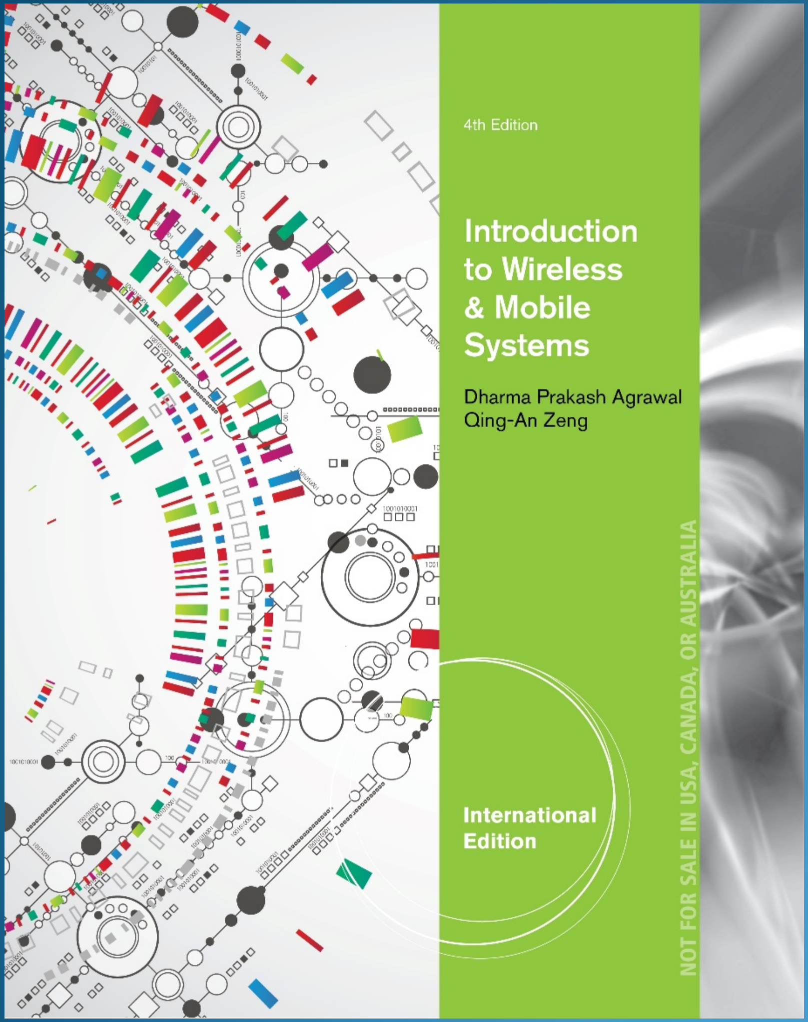


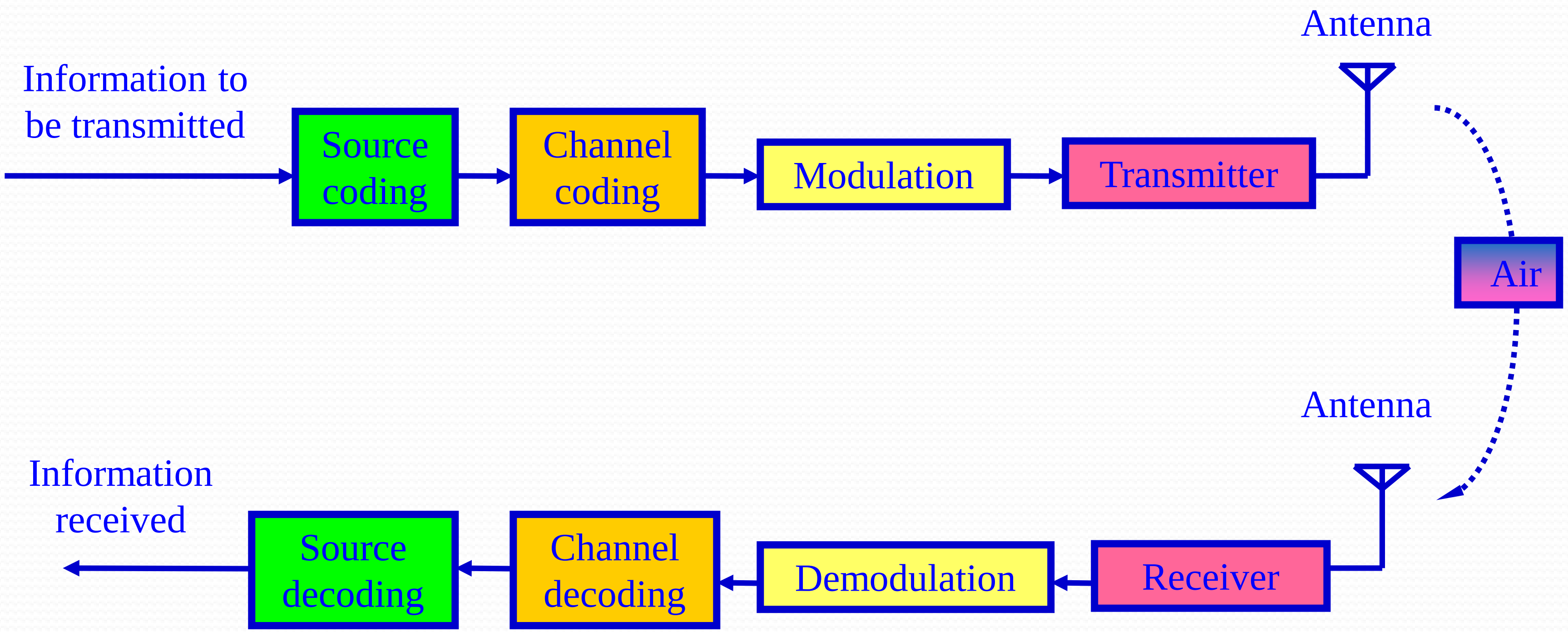
Chapter 4

Channel Coding and Error Control

Outline

- Introduction
- Block Codes
- Cyclic Codes
- CRC (Cyclic Redundancy Check)
- Convolutional Codes
- Interleaving
- Information Capacity Theorem
- Turbo Codes
- ARQ (Automatic Repeat Request)
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective-repeat ARQ

Introduction



Forward Error Correction (FEC)

- The key idea of FEC is to transmit enough redundant data to allow receiver to recover from errors all by itself. No sender retransmission required
- A simple redundancy is to attach a parity bit

  1010110
- The major categories of FEC codes are
  - Block codes
  - Cyclic codes
  - Reed-Solomon codes (Not covered here)
  - Convolutional codes, and
  - Turbo codes, etc.

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
- Modulo 2 Addition

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ \underline{0} & \underline{1} & \underline{0} & \underline{\ \ 1} \\ 0 & 1 & 1 & 10 \end{array}$$

Linear Block Codes: Example

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

Coefficients of $x^0 x^1 x^2$

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

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

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

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

$$G = \begin{bmatrix} 1000\,|\,110 \\ 0100\,|\,011 \\ 0010\,|\,111 \\ 0001\,|\,101 \end{bmatrix} = [I \mid P]$$

I is the identity matrix

P is the parity matrix

Linear Block Codes: Example

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:

$$m.G = \underbrace{[1011]}_{\text{Data}} \begin{bmatrix} 1000110 \\ 0100011 \\ 0010111 \\ 0001101 \end{bmatrix} = [\underbrace{1011}_{\text{Data}} | \underbrace{100}_{\text{Parity}}]$$

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

Linear Block Codes

k- data and $r = n$-k redundant bits

$$\begin{cases} c_1 = m_1 \\ c_2 = m_2 \\ \ldots \\ c_k = m_k \end{cases}$$

$$\begin{cases} c_{k+1} = m_1 p_{1(k+1)} \oplus m_2 p_{2(k+1)} \oplus \ldots \oplus m_k p_{k(k+1)} \\ \ldots \\ c_n = m_1 p_{1n} \oplus m_2 p_{2n} \oplus \ldots \oplus m_k p_{kn} \end{cases}$$

Linear Block Codes

- The uncoded k data bits be represented by the **m** vector:

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

  The corresponding codeword be represented by the n-bit **c** vector:

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

- Each parity bit consists of weighted modulo 2 sum of the data bits represented by symbol for Exclusive OR or modulo 2 addition

Linear Block Codes

- Linear Block Code

The block length C of the Linear Block Code is

$$\mathbf{C} = \mathbf{mG}$$

where m is the information codeword block length, **G** is the generator matrix.

$$\mathbf{G} = [I_k \mid P]_{k \times n}$$

where P_i = Remainder of $[x^{n-k+i-1}/g(x)]$ for $i=1, 2, .., k$, and **I** is unit or identity matrix.

- At the receiving end, parity check matrix can be given as:

$\mathbf{H} = [\mathbf{P}^{\mathrm{T}} \mid \mathbf{I}_{n-k}]$, where $\mathbf{P}^{\mathrm{T}}$ is the transpose of the matrix **p**.

Linear Block Codes

Example: Find linear block code encoder **G** if code generator polynomial $g(x)$ with k data bits, and r parity bits = n - k

$$\because \quad G = [I \mid P] = \begin{bmatrix} 1\,0 \cdots\cdots 0 P^1 \\ 0\,1 \cdots\cdots 0 P^2 \\ \cdots\cdots\cdots\cdots\cdots \\ 0\,0 \cdots\cdots 1 P^k \end{bmatrix}$$

where

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

Linear Block Codes

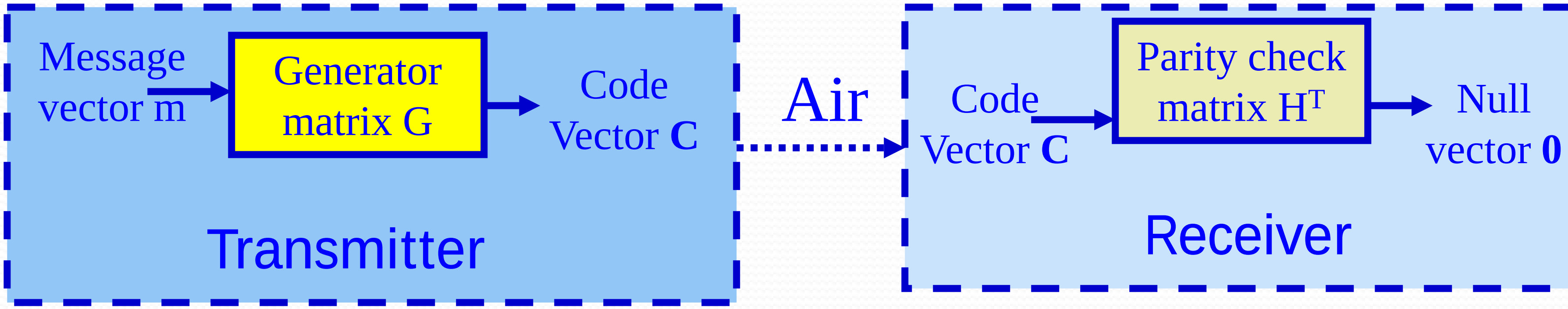
Example: The parity matrix **P** (*k* by *n*-*k* matrix) is given by:

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

where

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

Block Codes: Linear Block Codes



Operations of the generator matrix and the parity check matrix

- Consider a (7, 4) linear block code, given by **G** as

$$G = \begin{bmatrix} 1000 \mid 111 \\ 0100 \mid 110 \\ 0010 \mid 101 \\ 0001 \mid 011 \end{bmatrix} \qquad H^T = \begin{bmatrix} P \\ I_{n-k} \end{bmatrix} = \begin{bmatrix} 1110 \mid 100 \\ 1101 \mid 010 \\ 1011 \mid 001 \end{bmatrix}^T$$

For convenience, the code vector is expressed as

$$\mathbf{c} = \left[\mathbf{m} \middle| \mathbf{c}_p \right]$$

Where $\mathbf{c}_p = \mathbf{m}P$ is an *(n-k)*-bit parity check vector

Block Codes: Linear Block Codes

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

Received code vector $x = c \oplus e$, here **e** is an error vector, the matrix $\mathbf{H}^T$ has the property

$$cH^T = \begin{bmatrix} m \mid c_p \end{bmatrix} \begin{bmatrix} P \\ I_{n-k} \end{bmatrix}$$

$$= mP \oplus c_p = c_p \oplus c_p = 0$$

The transpose of matrix $\mathbf{H}^T$ is

$$H^T = \begin{bmatrix} P^T I_{n-k} \end{bmatrix}$$

Where $\mathbf{I}_{n\text{-}k}$ is a *n-k* by *n-k* unit matrix and $\mathbf{P}^T$ is the transpose of parity matrix **P**.

H is called parity check matrix.
Compute syndrome as $s = xH^T = (c \oplus e) \times H^T = cH^T \oplus eH^T = eH^T$

Linear Block Codes

If **S** is **0** then message is correct else there are errors in it, from common known error patterns the correct message can be decoded.

- For the (7, 4) linear block code, given by **G** as

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

- For **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]

Linear Block Codes

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,

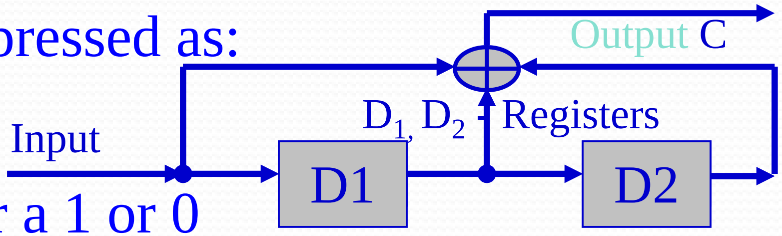
Syndrome **s** = **xH**^{T}

$$= [1001 | 001] \begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \\ -- \\ 100 \\ 010 \\ 001 \end{bmatrix} = [101] = (eH^T)$$

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

Cyclic Codes

It is a block code which uses a shift register to perform encoding and decoding the code word with n bits is expressed as:



$$c(x) = c_1 x^{n-1} + c_2 x^{n-2} \ldots\ldots + c_n$$

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

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

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

if the received signal is $c(x) + e(x)$ where $e(x)$ is the error

To check if received signal is error free, the remainder from dividing $c(x) + e(x)$ by $g(x)$ is obtained(syndrome). If this is 0 then the received signal is considered error free else error pattern is detected from known error syndromes

Cyclic Code: Example

Example: Find the code words $c(x)$ if $m(x) = x^2 + x + 1$ and $g(x) = x^3\ x^3 + x + 1$ for (7, 4) cyclic code

We have n = total number of bits = 7, k = number of information bits = 4, r = number of parity bits = $n - k = 3$

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

$$= rem\left[\frac{x^5 + x^4 + x^3}{x^3 + x + 1}\right] = x$$

Then,

$$c(x) = m(x)x^{n-k} + c_p(x) = x + x^3 + x^4 + x^5$$

Cyclic Redundancy Check (CRC)

- Cyclic Redundancy Code (CRC) is an error-checking code
- 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. Blocks of input bit as coefficient-sets for polynomials is called message polynomial. Polynomial with constant coefficients is called the generator polynomial

Cyclic Redundancy Check (CRC)

- Generator polynomial is divided 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

  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*

  P – The predefined CRC generating polynomial

The main idea in CRC algorithm is that the FCS is generated so that *J* should be exactly divisible by P

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* bits and then divide it by *p*
  - Get the remainder *R* as FCS
  - 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 $Q.x^{n-k}/P=Q+R/P$
  - Where *R* is the reminder
  - $J=Q.x^{n-k}+R$. This value of *J* should yield a zero reminder for *J/P*

Common CRC Codes

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

Common CRC Codes

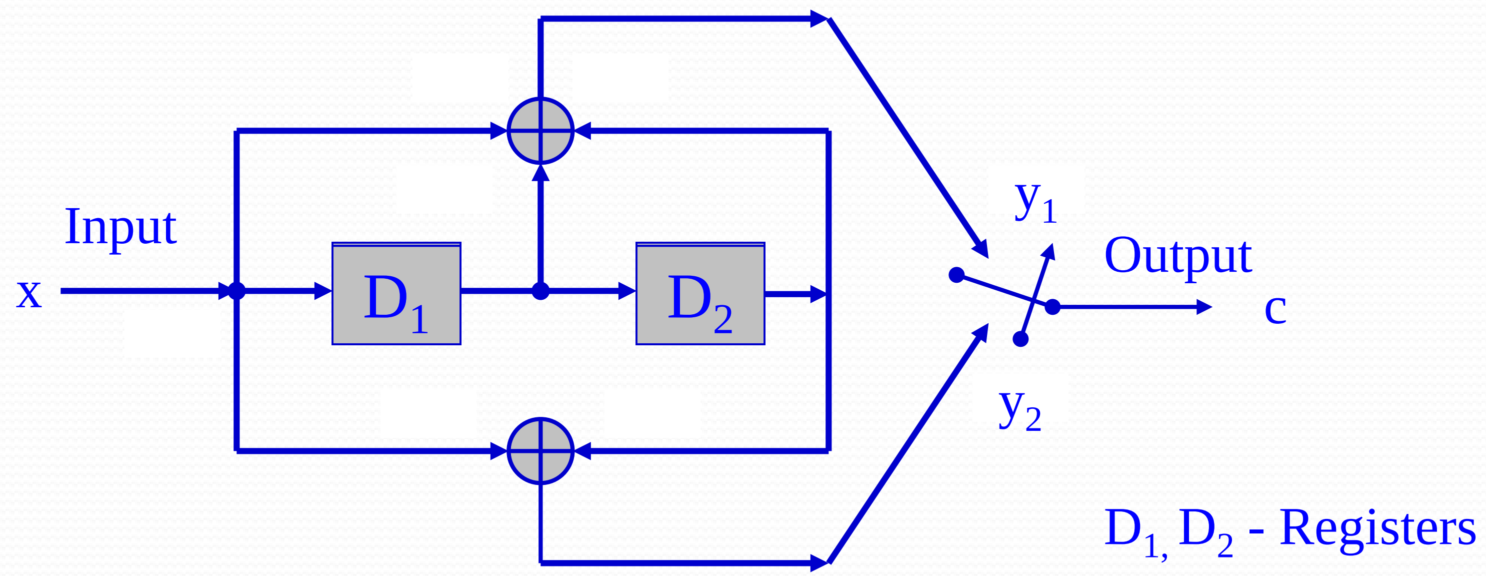
Code	Generator polynomial $g(x)$	Parity check bits
CRC-12	$1+x+x^2+x^3+x^{11}+x^{12}$	12
CRC-16	$1+x^2+x^{15}+x^{16}$	16
CRC-CCITT	$1+x^5+x^{15}+x^{16}$	16

Convolutional Codes

- Most widely used channel code
- Encoding of information stream rather than information blocks
- Decoding is mostly performed by the Viterbi Algorithm (not covered here)

- The constraint length *K* for a convolution code is defined as $K=M+1$ where *M* is the maximum number of stages in any shift register
- The code rate *r* is defined as $r = k/n$ where *k* is the number of parallel information bits and *n* is the the number of parallel output encoded bits at one time interval
- A convolution code encoder with $n=2$ and $k=1$ or code rate $r=1/2$ is shown next

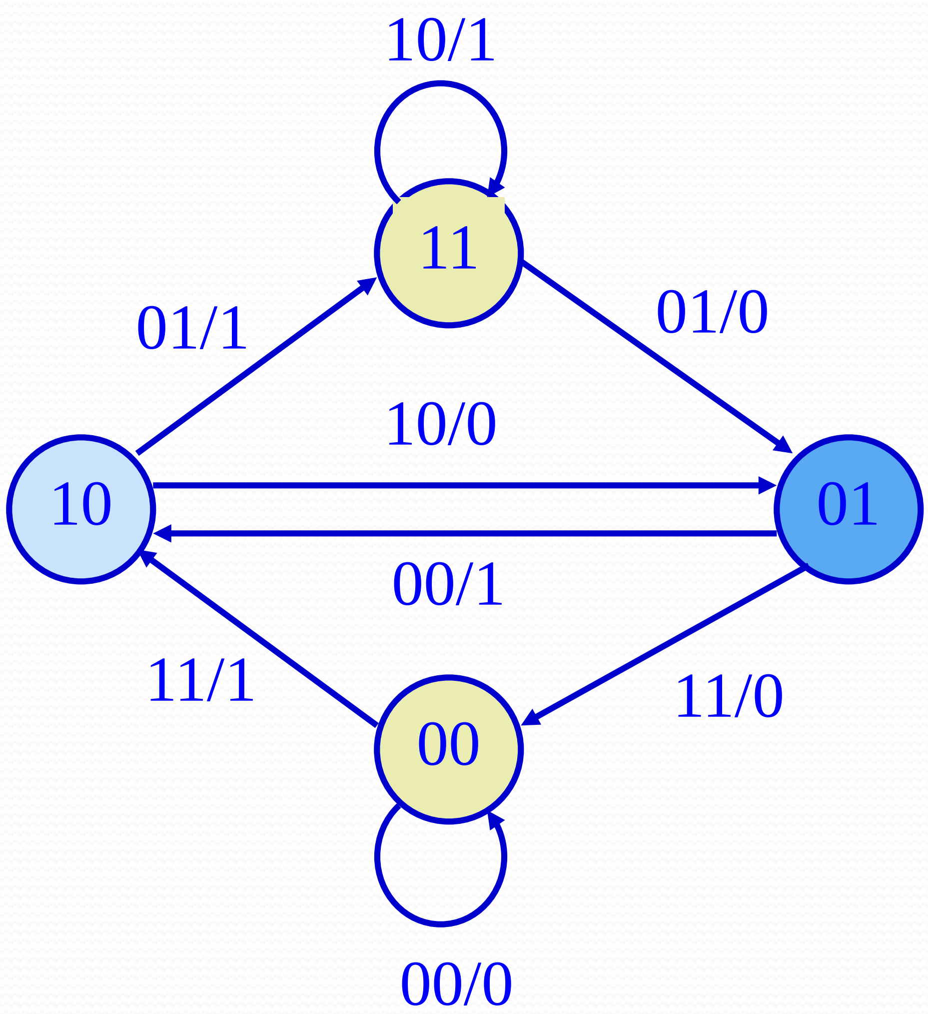
Convolutional Codes: ($n=2$, $k=1$, $M=2$) Encoder



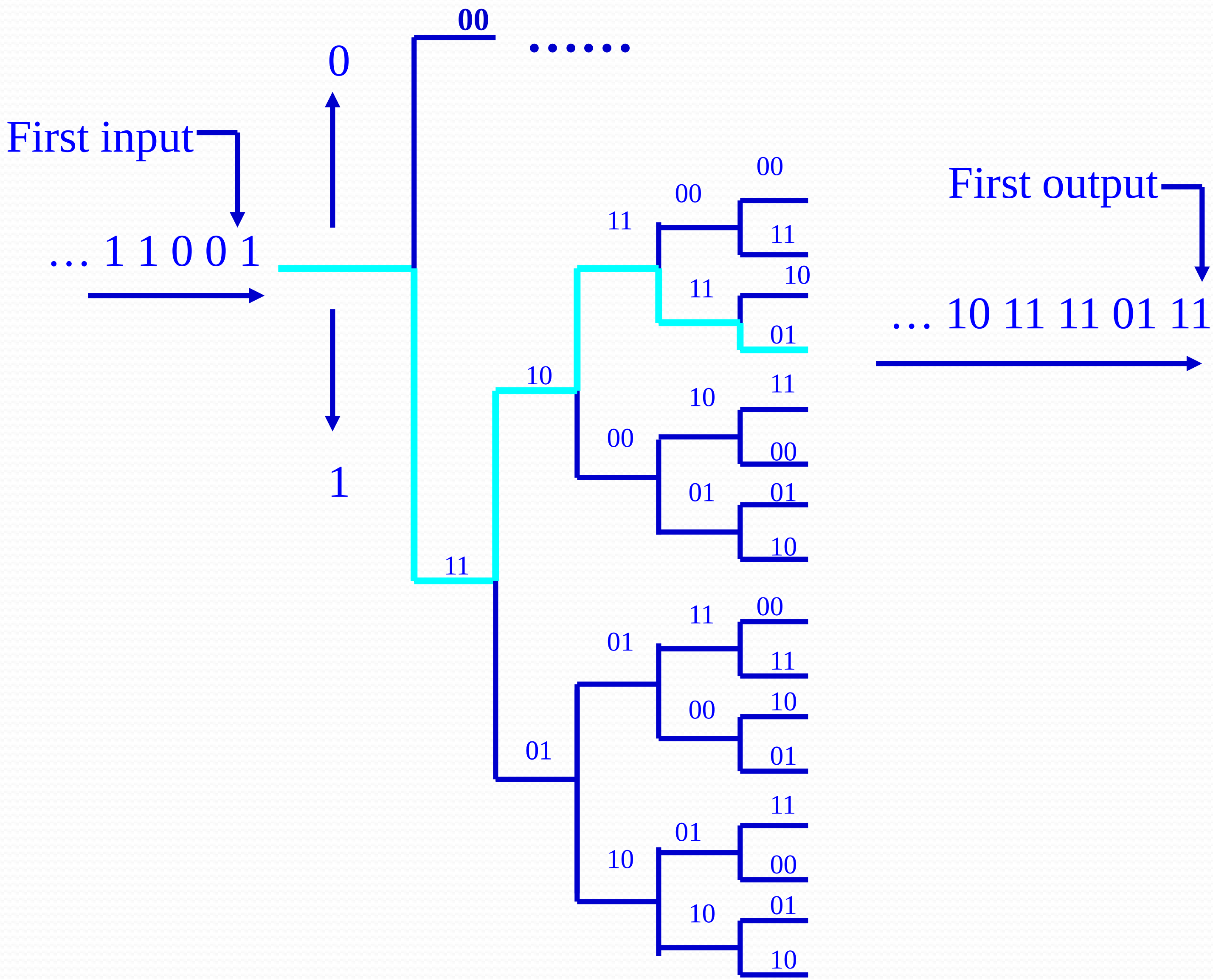
Input x:	1	1	1	0	0	0	…
Output y_1y_2:	11	01	10	01	11	00	…

Input x:	1	0	1	0	0	0	…
Output y_1y_2:	11	10	00	10	11	00	…

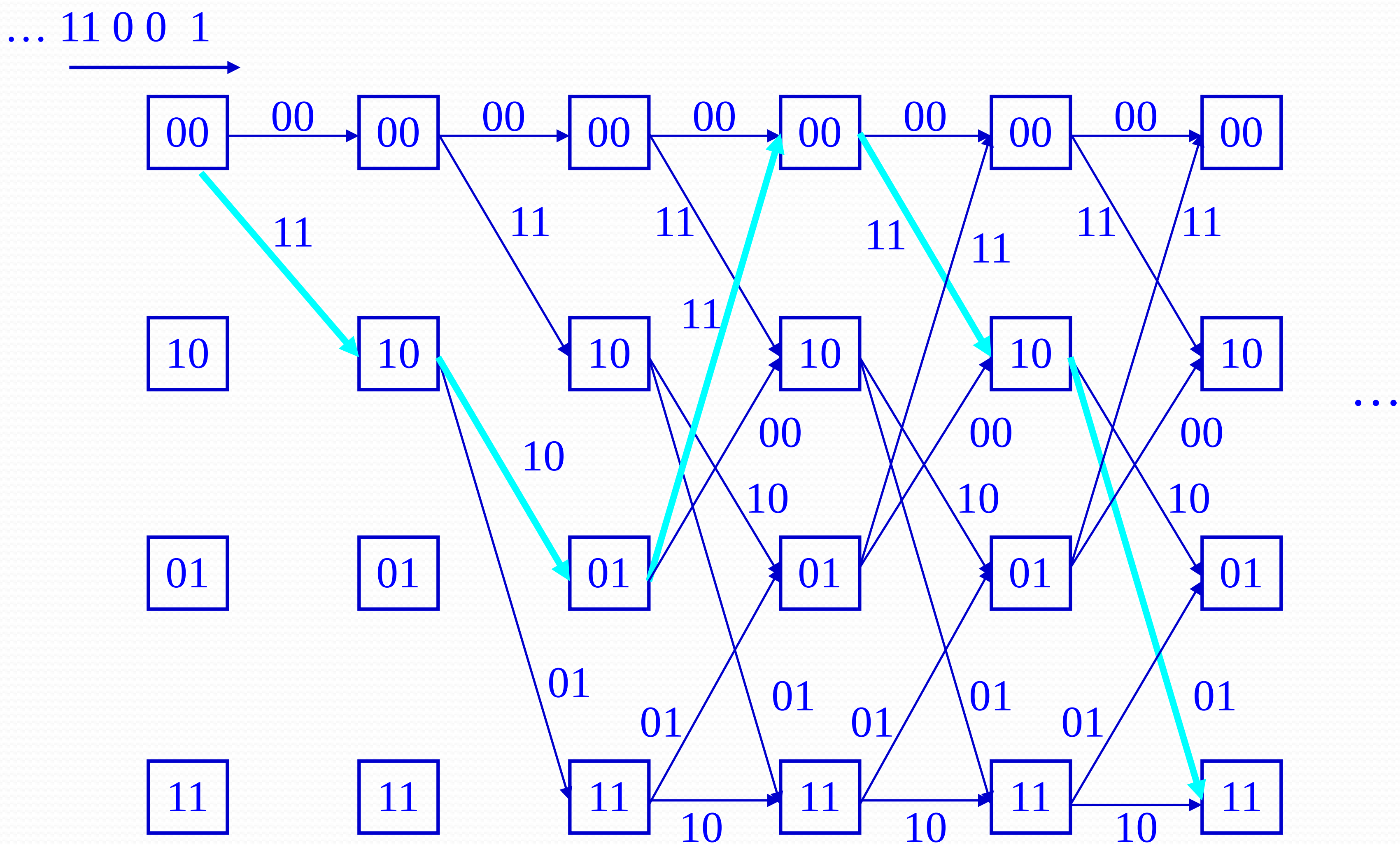
State Diagram



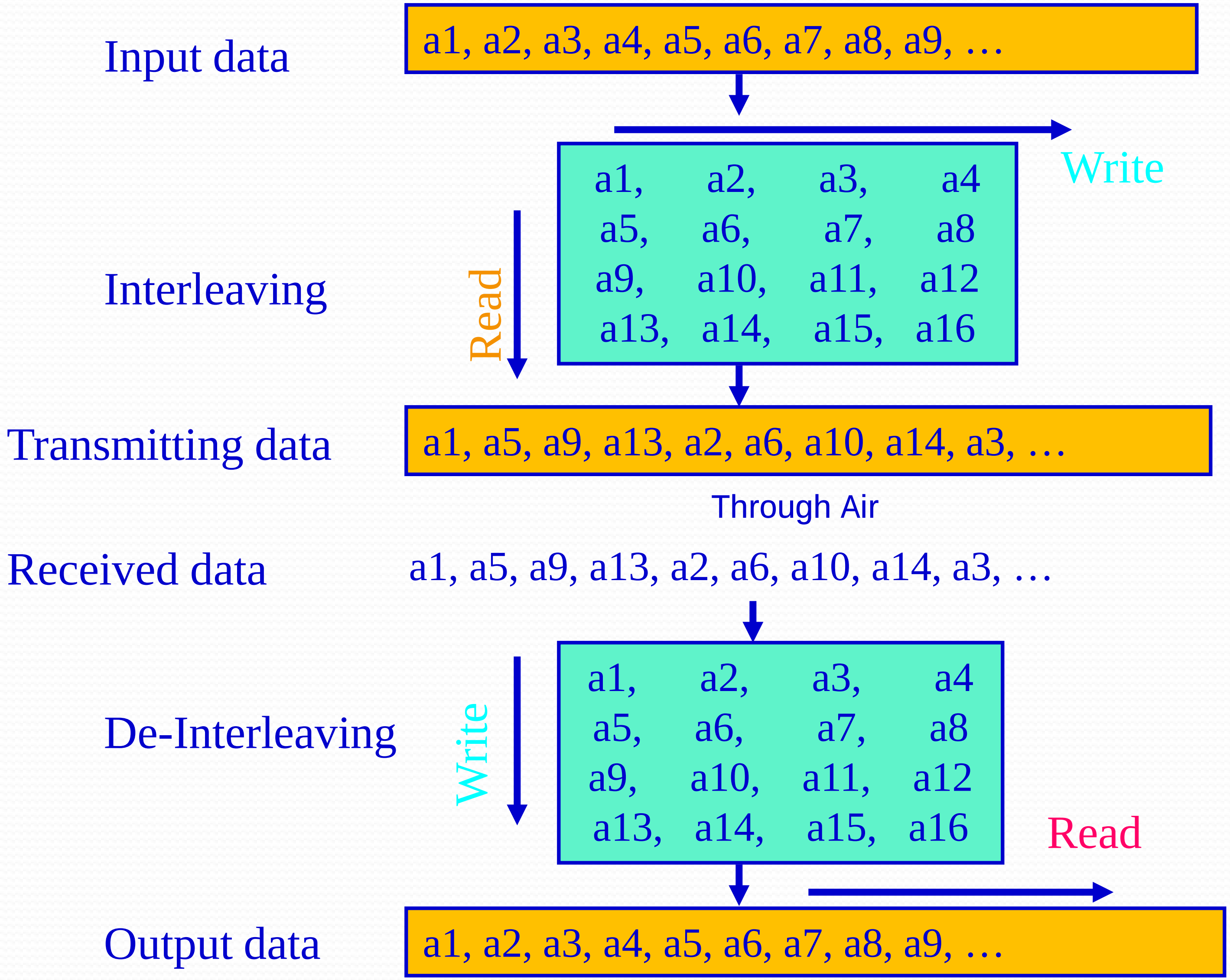
Tree Diagram



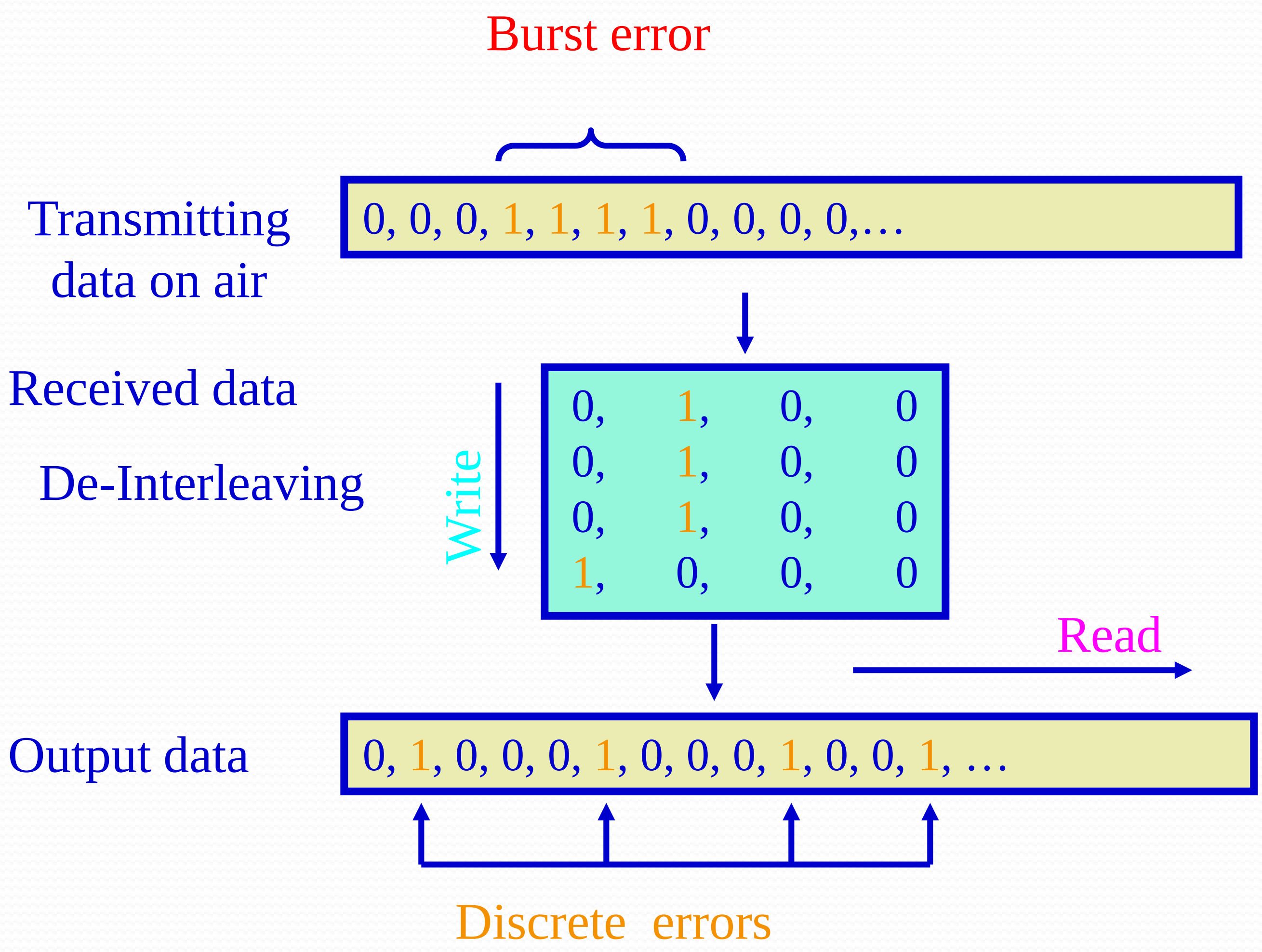
Trellis



Interleaving



Interleaving (Example)



Information Capacity Theorem (Shannon Limit)

- The information capacity (or channel capacity) C of a continuous channel with bandwidth B Hertz can be perturbed by additive Gaussian white noise of power spectral density $N_0/2$, provided bandwidth B satisfies

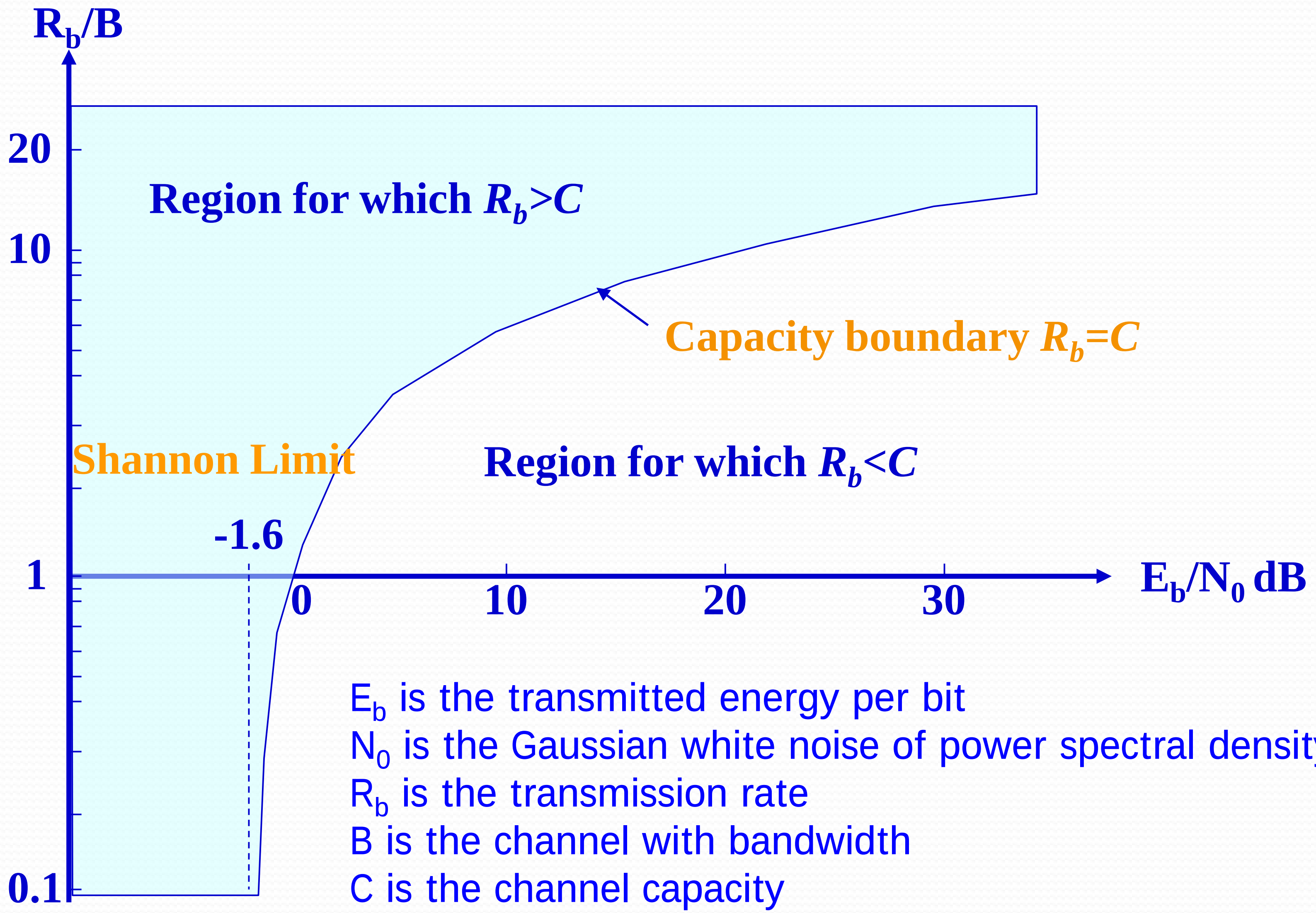
$$C = B\log_2\left(1+\frac{P}{N_0 B}\right) \quad bits/second$$

where P is the average transmitted power $P = E_b R_b$ (for an ideal system, $R_b = C$)

E_b is the transmitted energy per bit

R_b is transmission rate

Shannon Limit



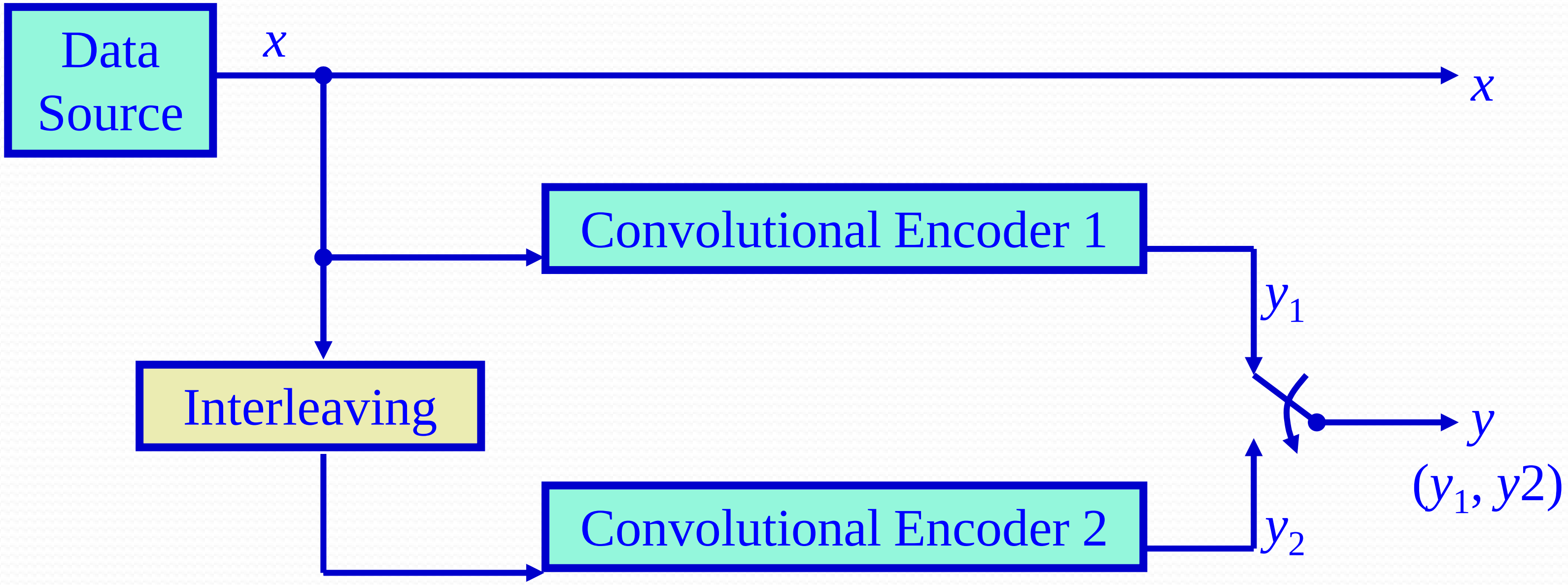
E_b is the transmitted energy per bit
N_0 is the Gaussian white noise of power spectral density
R_b is the transmission rate
B is the channel with bandwidth
C is the channel capacity

Turbo Codes

- 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 (E_b/N_0 = -1.6 db if $R_b \rightarrow 0$) at modest complexity!
- Turbo codes have been proposed for low-power applications such as deep-space and satellite communications, as well as for interference limited applications such as third generation cellular, personal communication services, ad hoc and sensor networks

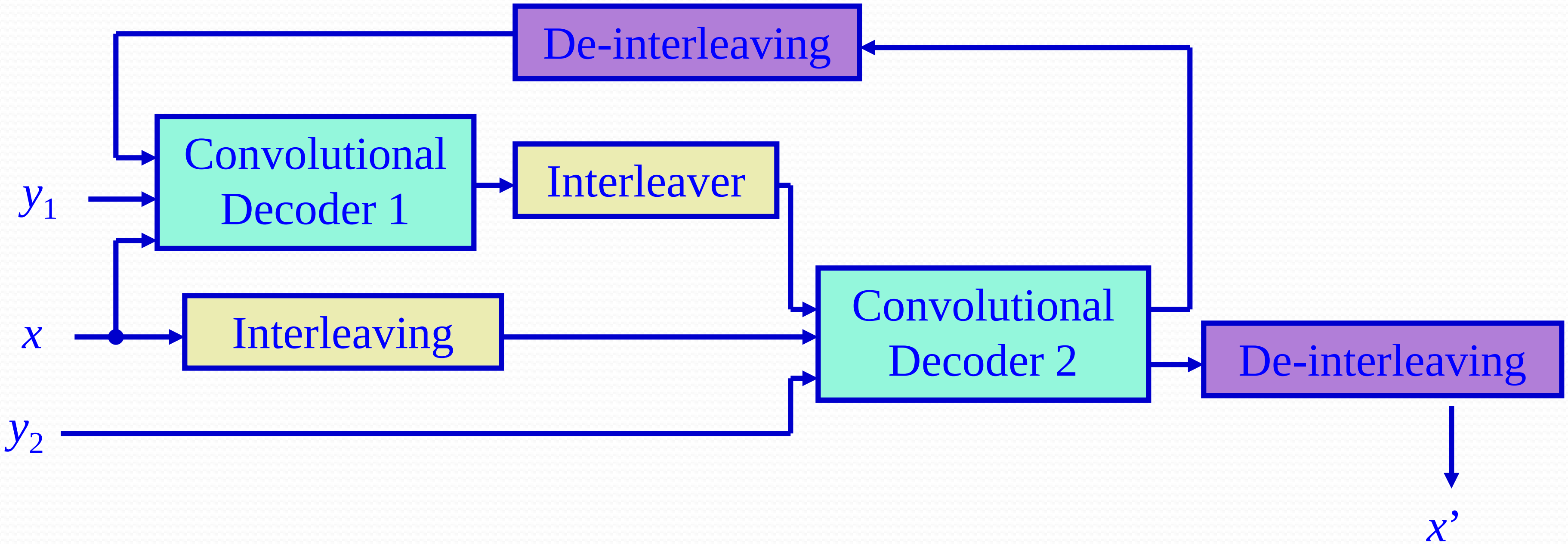
Turbo Codes: Encoder



x: Information

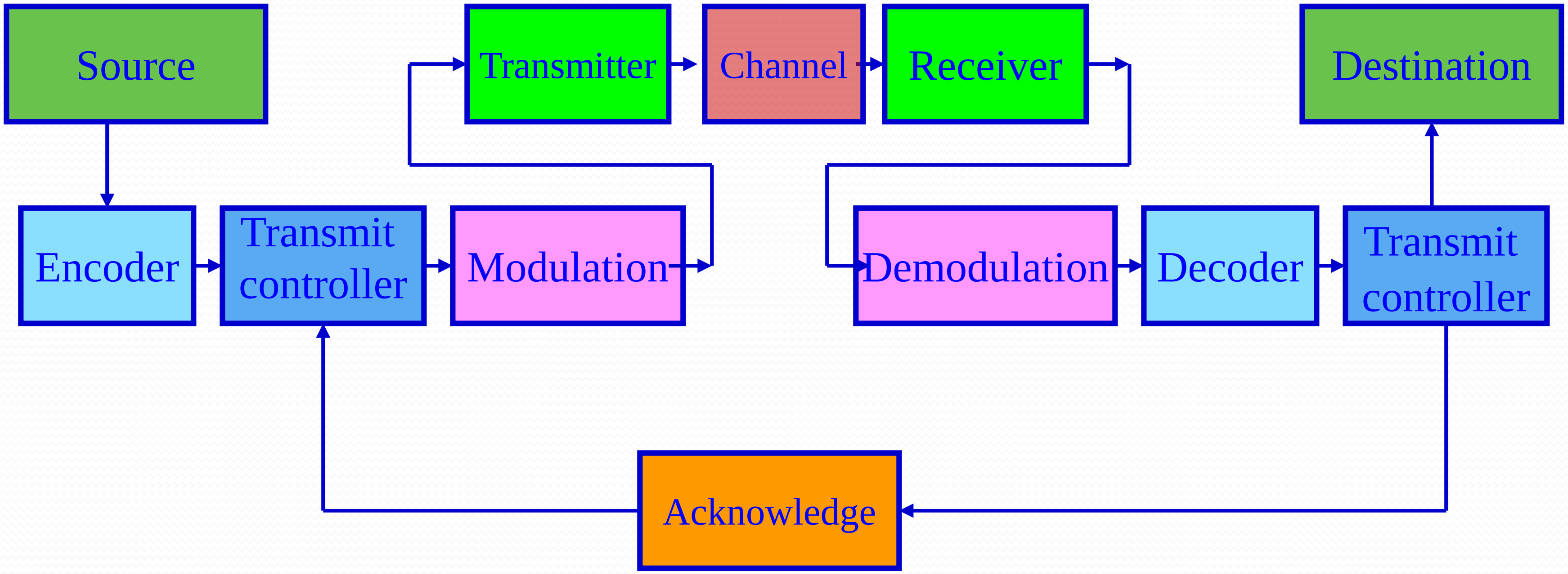
y_i: Redundancy Information

Turbo Codes: Decoder

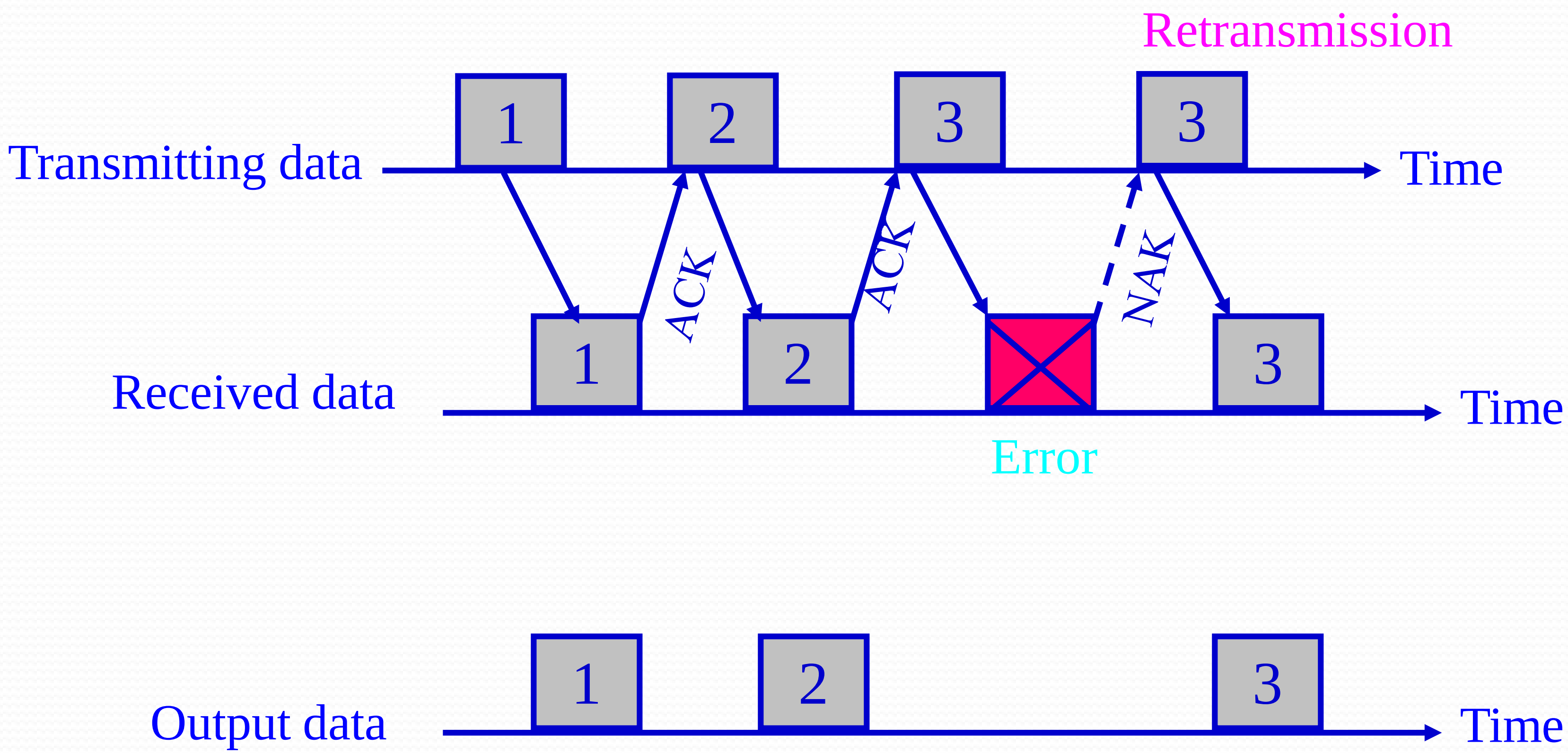


x': Decoded Information

Automatic Repeat Request (ARQ)



Stop-And-Wait ARQ (SAW ARQ)



ACK: Acknowledge

NAK: Negative ACK

Stop-And-Wait ARQ (SAW ARQ)

Given n = number of bits in a block, k = number of information bits in a block, D = round trip delay, R_b= bit rate, P_b = BER of the channel, and $P_{ACK} \approx (1- P_b)^n$

Throughput:

$$S_{SAW} = (1/T_{SAW}) \,.\, (k/n) = [(1- P_b)^n \,/\, (1 + D * R_b/ n)] * (k/n)$$

where T_{SAW} is the average transmission time in terms of a block duration

$$T_{SAW} = (1 + D \,.R_b/ n).\, P_{ACK} + 2\,(1 + D\, R_b/ n)\,\, P_{ACK}\,(1- P_{ACK})$$

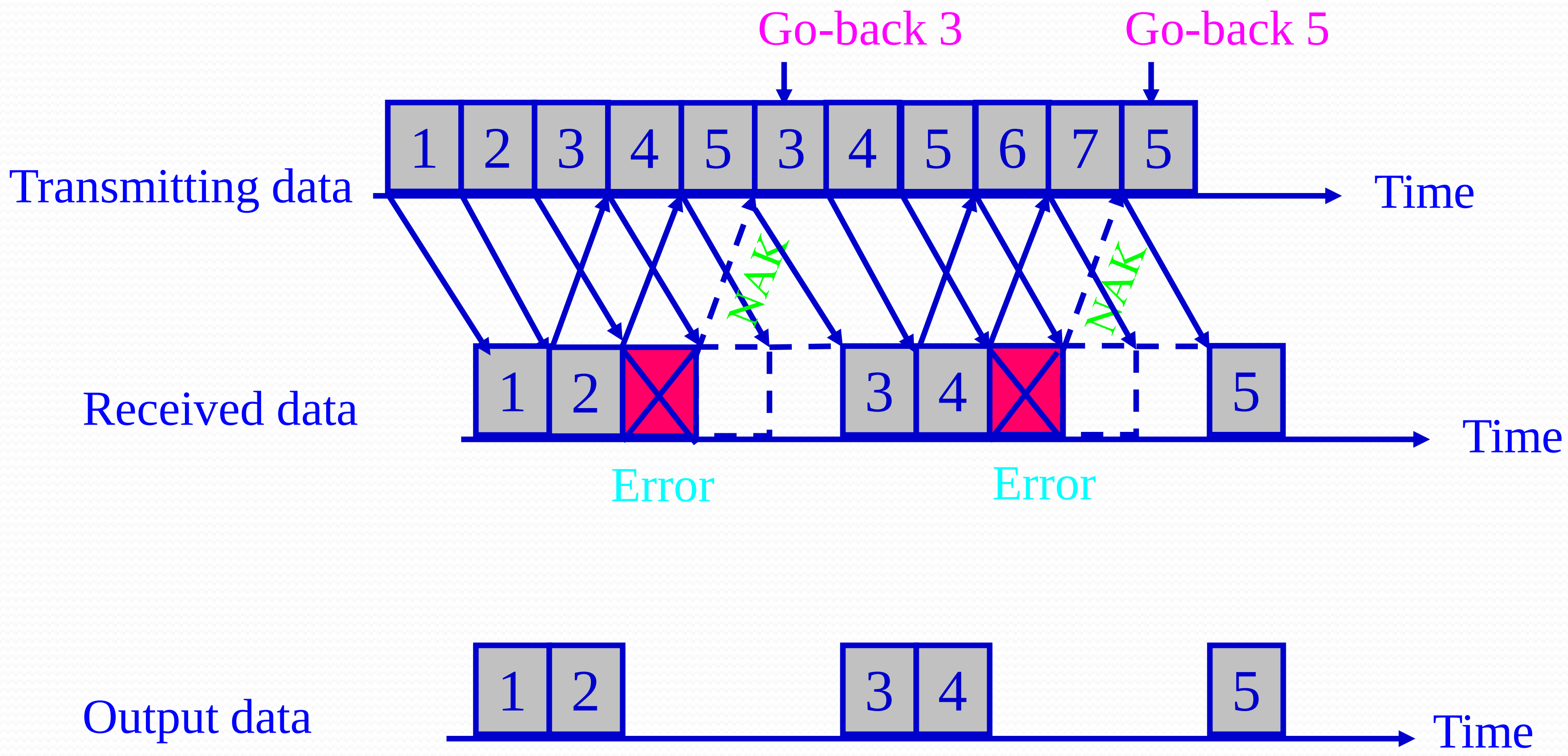
$$+\, 3\,\,(1 + D\,\, R_b/ n)\, P_{ACK}\,(1- P_{ACK})^2 + \ldots..$$

$$= (1+ D\,\, R_b/n)\,\, P_{ACK} \sum_{i=1}^{\infty} i\,\,(1-P_{ACK})^{i-1}$$

$$= (1+ D\,\, R_b/n)\,\,\, P_{ACK}/[1-(1-P_{ACK})]^2$$

$$= (1 + D\,\, R_b/ n) \,/\, P_{ACK}$$

Go-Back-N ARQ (GBN ARQ)



Go-Back-N ARQ (GBN ARQ)

Throughput

$$S_{GBN} = (1/T_{GBN})\,(k/n)$$

$$= [(1-P_b)^n \;/\; ((1-P_b)^n + N\;(1-(1-P_b)^n\,)\,)]\,(k/n)$$

where

$$T_{GBN} = 1 \,.\, P_{ACK} + (N+1) \,.\, P_{ACK} \,.\, (1-P_{ACK}) + 2 \,.\, (N+1) \,.\, \mathrm{P}_{ACK} \,.$$

$$(1-P_{ACK})^2 + \ldots.$$

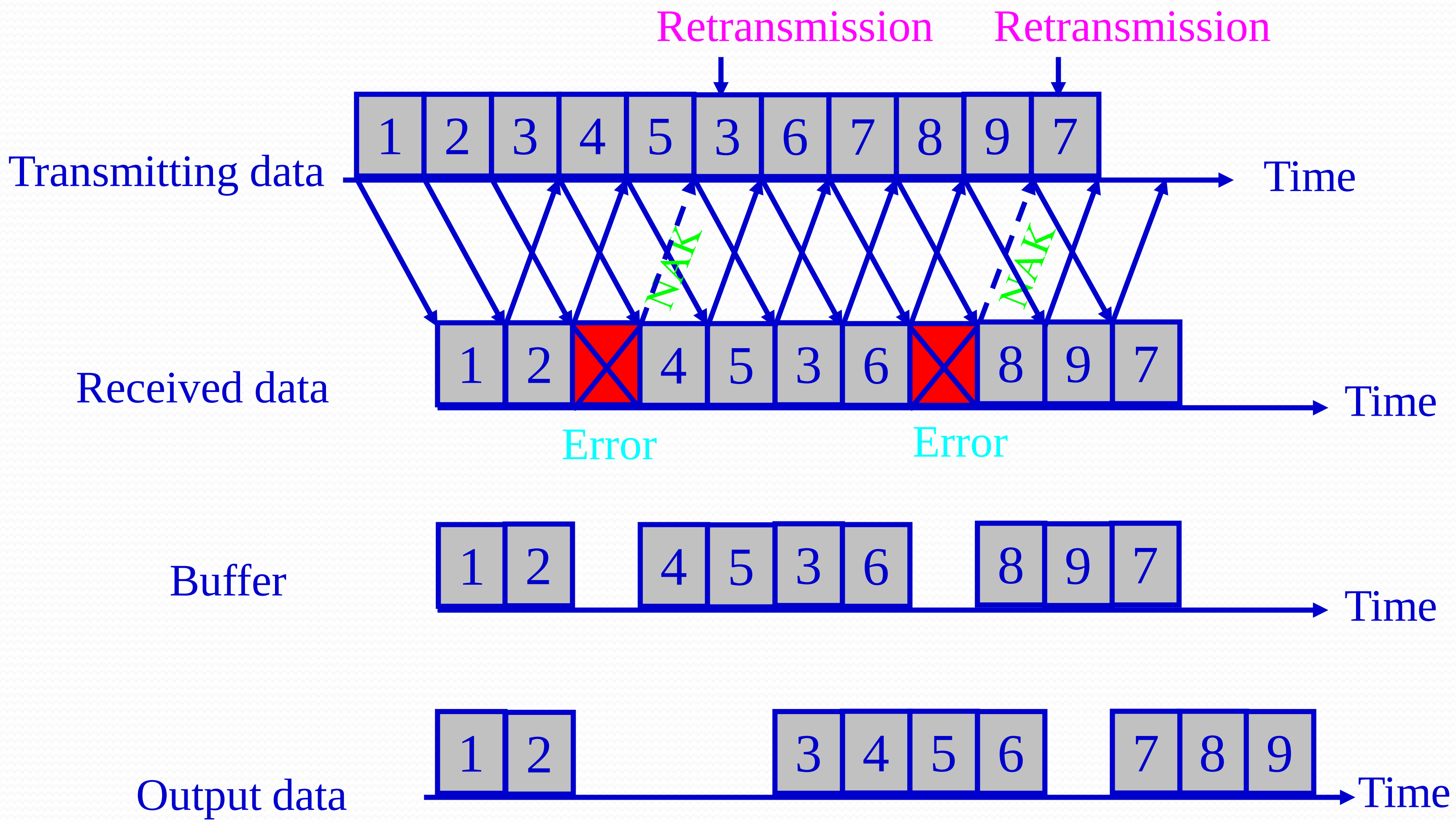
$$= P_{ACK} + P_{ACK}\;[(1-P_{ACK}) + (1-P_{ACK})^2 + (1-P_{ACK})^3 + \ldots] +$$

$$P_{ACK}[N\;(1-P_{ACK}) + 2\;N\;(1-P_{ACK})^2 + 3\;N\;(1-P_{ACK})^3 + \ldots]$$

$$= P_{ACK} + P_{ACK}\,[(1-P_{ACK})/\{1-(1-P_{ACK})\} + N\;(1-P_{ACK})/\{1-(1-P_{ACK})\}^2]$$

$$= 1 + N(1-P_{ACK})/P_{ACK} \approx 1 + (N\;[1-(1-P_b)^n\,])/\,(1-P_b)^n$$

Selective-Repeat ARQ (SR ARQ)



Selective-Repeat ARQ (SR ARQ)

Throughput

$$S_{SR} = (1/T_{SR}) * (k/n)$$

$$= (1- P_b)^n * (k/n)$$

where

$$T_{SR} = 1 . P_{ACK} + 2\ P_{ACK}\ (1- P_{ACK}) + 3\ P_{ACK}\ (1- P_{ACK})^2$$

$$+ \ldots.$$

$$= P_{ACK} \sum_{i=1}^{\infty} i\ (1-P_{ACK})^{i-1}$$

$$= P_{ACK}/[1-(1-P_{ACK})]^2$$

$$= 1/(1- P_b)^n \qquad \text{where } P_{ACK} \approx (1-P_b)^n$$