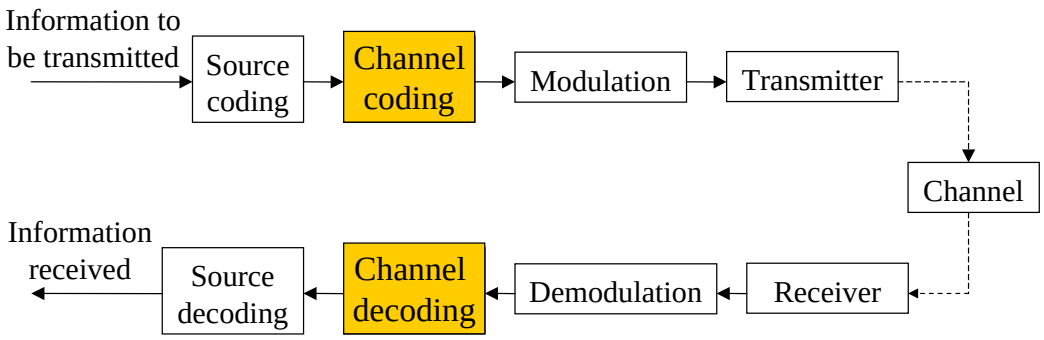
Chapter 4

Channel Coding

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.
- The major categories of FEC codes are
  - Block codes,
  - Cyclic codes,
  - Convolutional codes,
  - and Turbo codes.

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 (received vector = code vector + error vector)
- Tradeoffs between
  - Efficiency
  - Reliability
  - Encoding/Decoding complexity
- In Maximum-likelihood decoding, we compare the received vector with all possible transmitted codes and choose that which is closest in Hamming distance (i.e., which is differs in the fewest bits). This results in a minimum probability of a code word error.

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 $\oplus$ symbol.

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

Block Codes: Linear Block Codes

- Linear Block Code

The codeword block ***C*** of the Linear Block Code is

$$\boldsymbol{C} = m\,\boldsymbol{G}$$

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

$$\boldsymbol{G} = [\mathbf{I}_k \mid \mathbf{P}]_{k \times n}$$

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

- The parity check matrix

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

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.

We have n = Total number of bits = 7, k = Number of information bits = 4, r = Number of parity bits = n - k = 3.

$$\because \quad G = [I \mid P] = \begin{bmatrix} 1\,0\cdots 0\,p_1 \\ 0\,1\cdots 0\,p_2 \\ \cdots\cdots\cdots \\ 0\,0\cdots 1\,p_k \end{bmatrix},$$

where

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

Block Codes: Example (Continued)

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

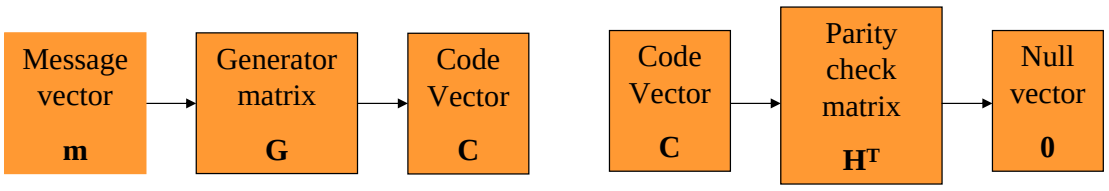
$$p_2 = \mathrm{Re}\left[\frac{x^4}{1+x+x^3}\right] = x+x^2 \rightarrow [011]$$

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

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

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

Block Codes: Linear Block Codes



Operations of the generator matrix and the parity check matrix

The parity check matrix H is used to detect errors in the received code by using the fact that $c * H^T = \mathbf{0}$ (null vector)

Let $x = c \oplus e$ be the received message where c is the correct code and e is the error

Compute $S = x * H^T = (c \oplus e) * H^T = c H^T \oplus e H^T = e H^T$ (s is know as syndrome matrix)

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.

Linear Block Codes

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

$$G = \begin{bmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{bmatrix}$$

Then,

$$H = \begin{bmatrix} 1110100 \\ 1101010 \\ 1011001 \end{bmatrix}$$

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**$^{\mathbf{T}}$=[0, 0,0]

Linear Block Codes

Let c suffer an error such that the received vector

x=**c**$\oplus$ **e**

=[1 0 1 1 0 0 1]$\oplus$[0 0 1 0 0 0 0]

=[1 0 0 1 0 0 1].

Then,

$$s = xH^T$$

$$= \begin{bmatrix} 1001001 \end{bmatrix} \begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \\ 100 \\ 010 \\ 001 \end{bmatrix} = [101]$$

$$= (eH^T)$$

Cyclic Codes

It is a block code which uses a shift register to perform encoding and Decoding (all code words are shifts of each other)
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 c_i is either a 1 or 0.

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

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 codewords c(x) if $m(x)=1+x+x^2$ and $g(x)=1+x+x^3$ 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.

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

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

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:

  M – The original frame (k bits) to be transmitted before adding the Frame Check Sequence (FCS).

  F – The resulting FCS of n bits to be added to M (usually n=8, 16, 32).

  T – The cascading of M and F.

  P – The predefined CRC generating polynomial with pattern of n+1 bits.

  The main idea in CRC algorithm is that the FCS is generated so that the remainder of T/P is zero.

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 is the frame to be transmitted.
- CRC is checked using the following process:
  - Receive the frame
  - Divide it by P
  - Check the remainder. If the remainder is not zero, then there is an error in the frame.

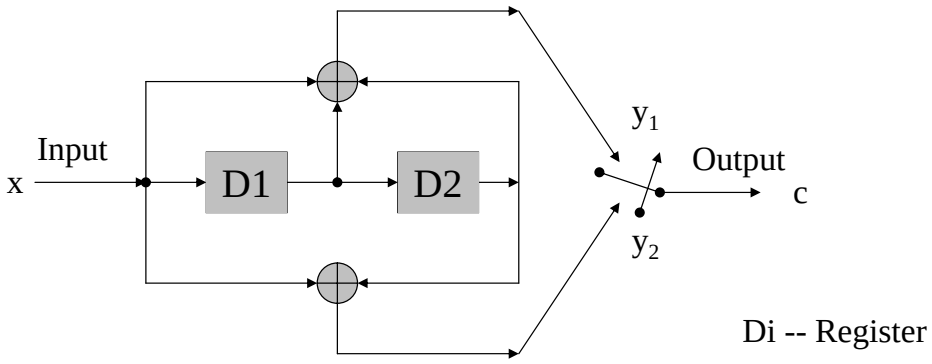
Common CRC Codes

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

- Encoding of information stream rather than information blocks
- Value of certain information symbol also affects the encoding of next *M* information symbols, i.e., memory *M*
- Easy implementation using shift register
  → Assuming *k* inputs and *n* outputs
- Decoding is mostly performed by the Viterbi Algorithm (not covered here)

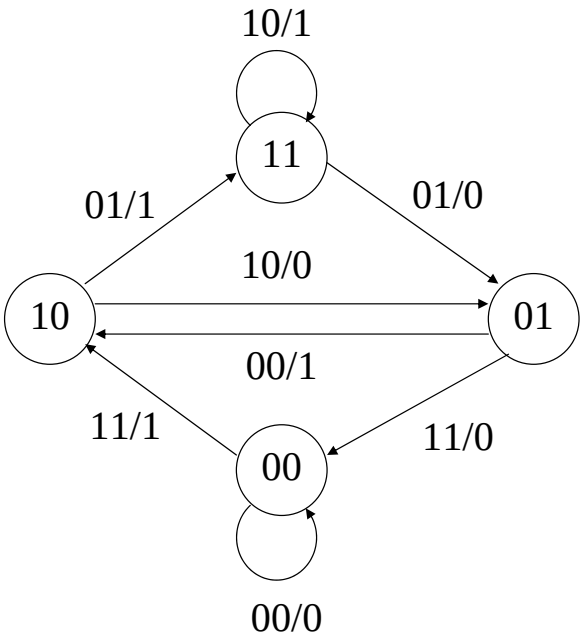
Convolutional Codes: (n=2, k=1, M=2) Encoder



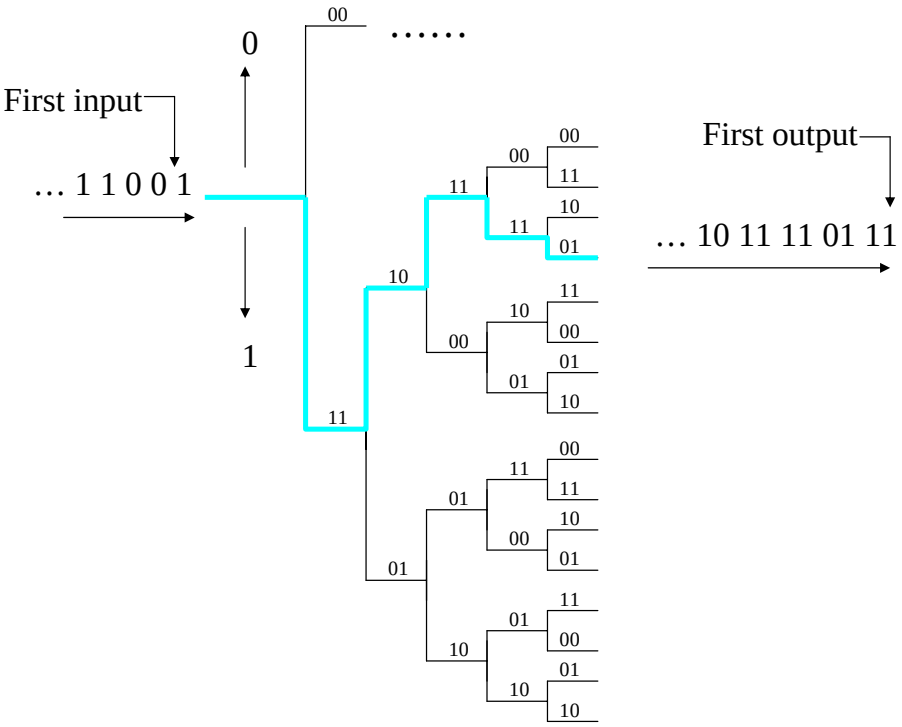
Di -- Register

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

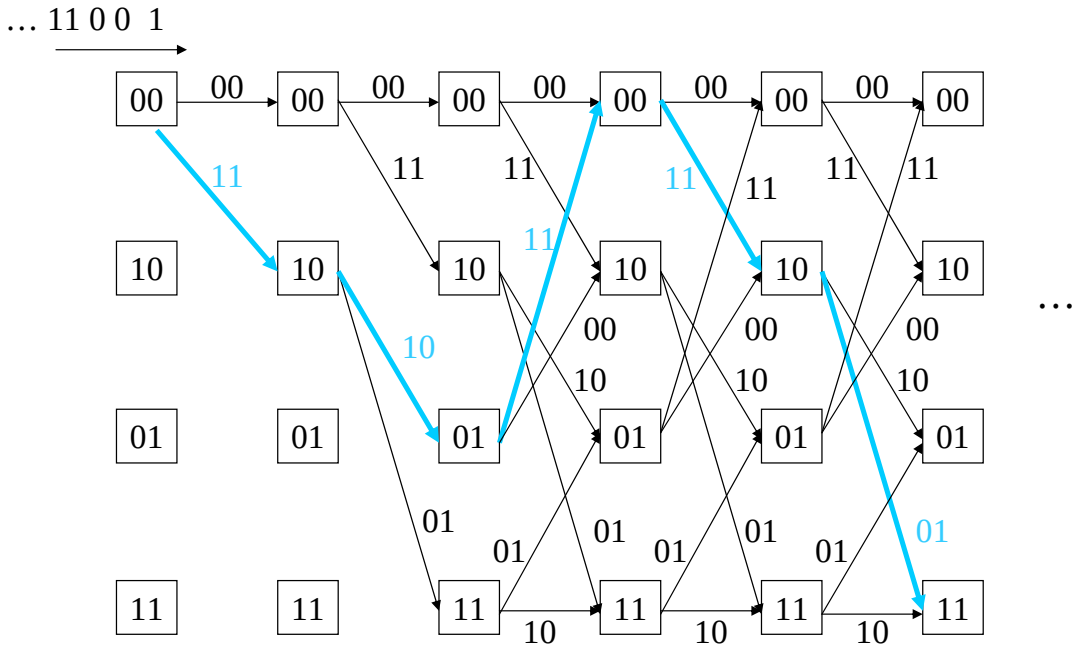
State Diagram



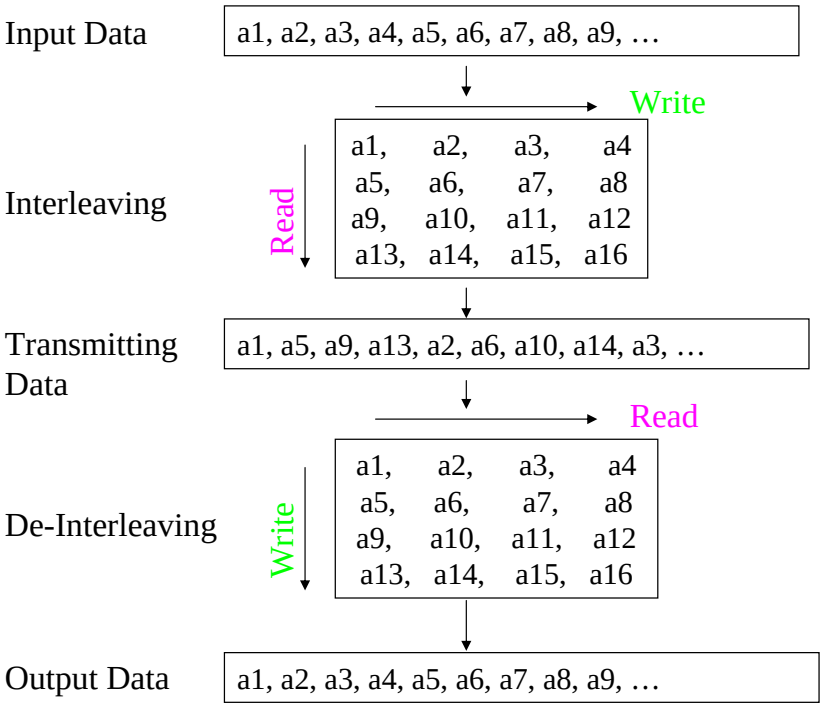
Tree Diagram



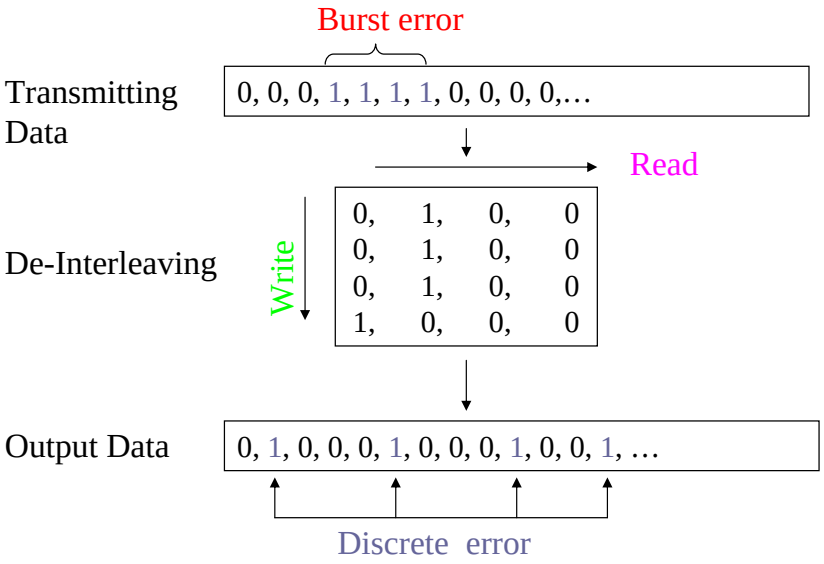
Trellis



Interleaving



Interleaving (Example)

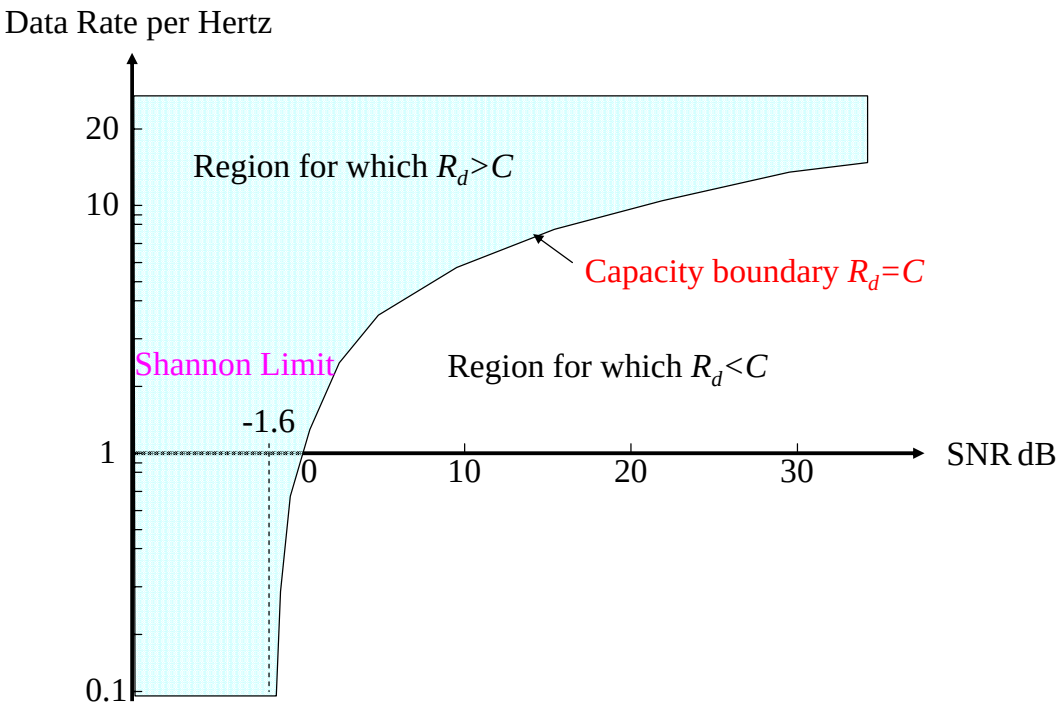


Information Capacity Theorem (Shannon Limit)

- The information capacity (or channel capacity) C of a continuous channel with bandwidth B Hertz satisfies

$$C = B\log_2\left(1+\frac{S}{N}\right) \quad bits/\sec ond$$

Shannon Limit

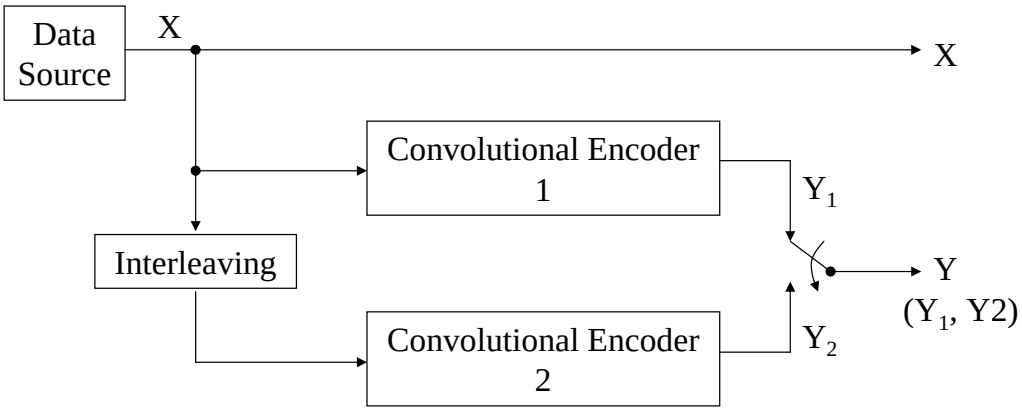


Turbo Codes

- 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, and soft-decision decoder)
- Performance close to the Shannon Limit 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.

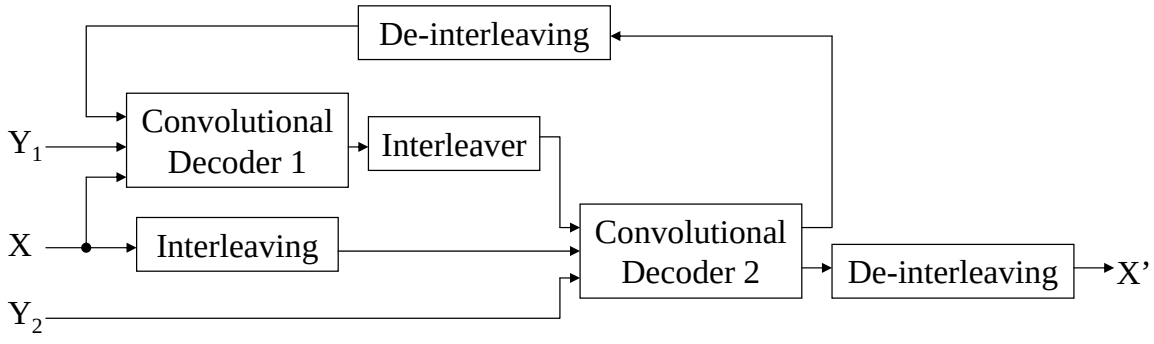
Turbo Codes: Encoder



X: Information

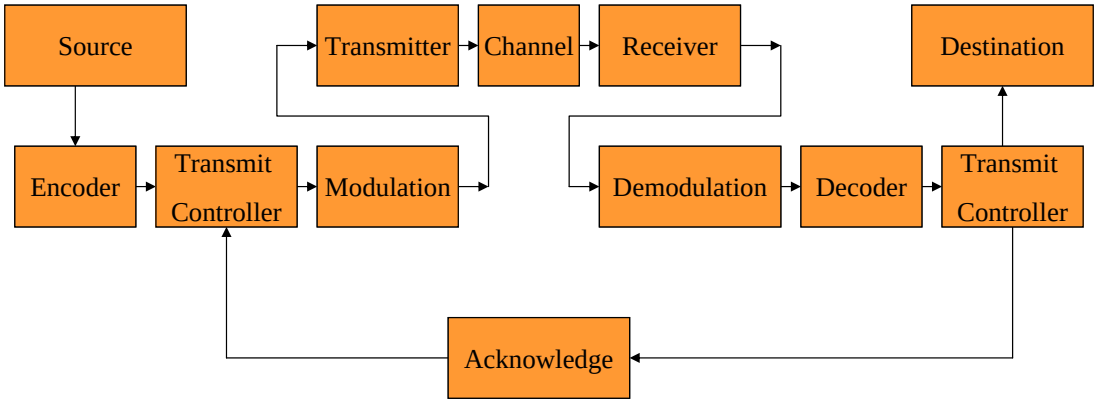
Y_i: Redundancy Information

Turbo Codes: Decoder

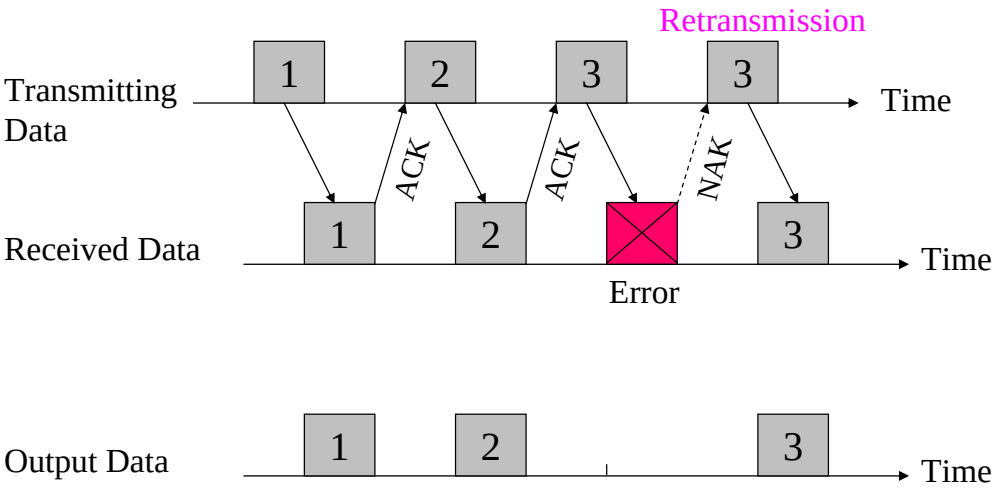


X': Decoded Information

Automatic Repeat Request (ARQ)



Stop-And-Wait ARQ (SAW ARQ)



ACK: Acknowledge

NAK: Negative ACK

Stop-And-Wait ARQ (SAW ARQ)

Throughput:

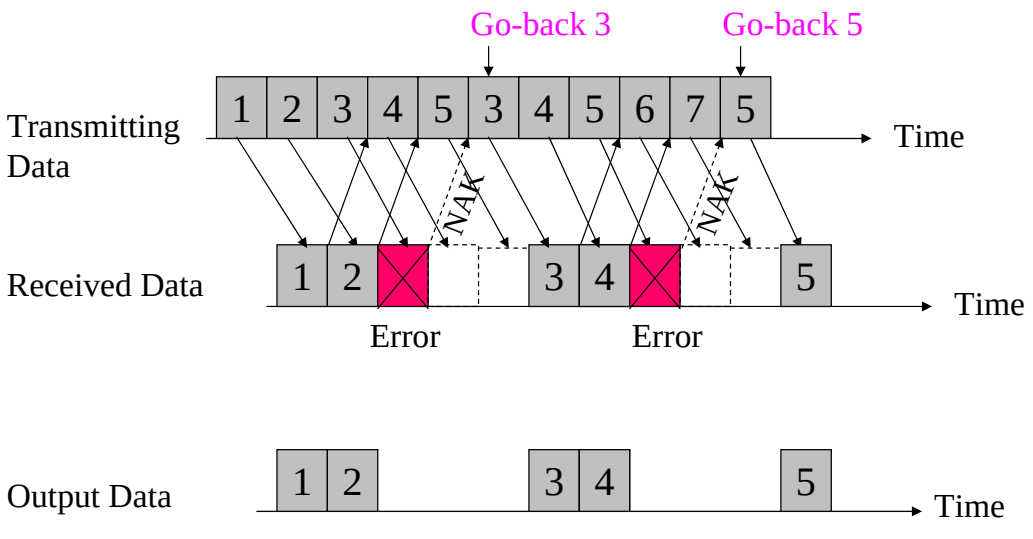
$$S = (1/T) * (k/n) = [(1- P_b)^n / (1 + D * R_b/ n)] * (k/n)$$

where T is the average transmission time in terms of a block duration

$$T = (1 + D * R_b/ n) * P_{ACK} + 2 * (1 + D * R_b/ n) * P_{ACK} * (1- P_{ACK})$$
$$+ 3 * (1 + D * R_b/ n) * P_{ACK} * (1- P_{ACK})^2 + ...$$
$$= (1+ D * R_b/n) * P_{ACK} \sum_{i=1}^{\infty} i * (1-P_{ACK})^{i-1}$$
$$= (1+ D * Rb/n) * P_{ACK}/[1-(1-P_{ACK})]^2$$
$$= (1 + D * R_b/ n) / P_{ACK}$$

where 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 $P_{ACK} = (1- P_b)^n$

Go-Back-N ARQ (GBN ARQ)



Go-Back-N ARQ (GBN ARQ)

Throughput

$$S = (1/T) * (k/n)$$

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

where

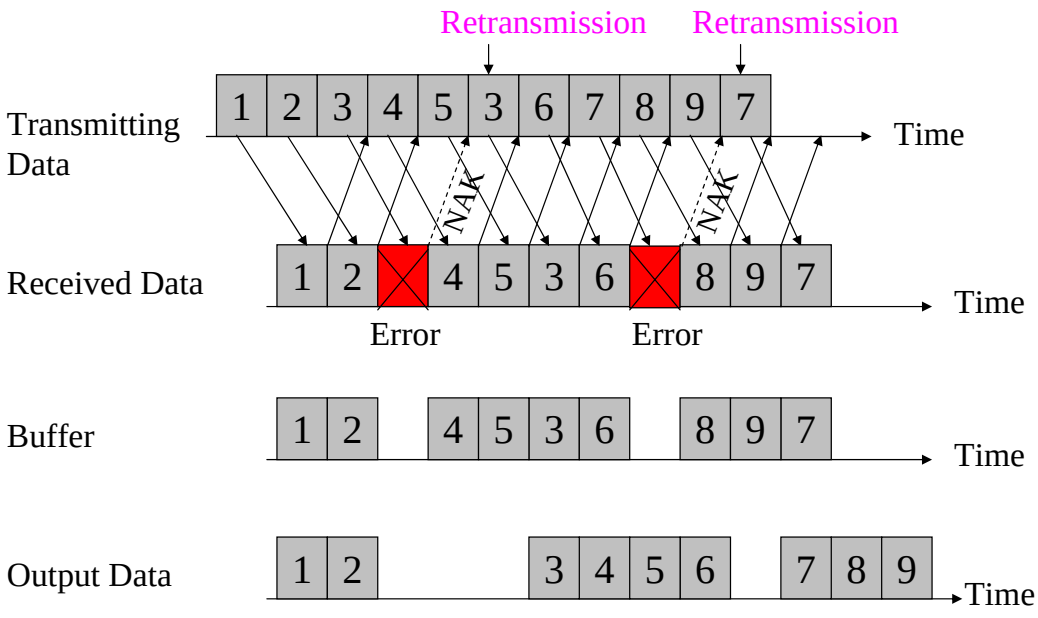
$$T = 1 * P_{ACK} + (N+1) * P_{ACK} * (1- P_{ACK}) +2 * (N+1) * P_{ACK} * (1- P_{ACK})^2 + \ldots.$$

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

$$= P_{ACK}+P_{ACK} *[(1-P_{ACK})/P_{ACK} + N * (1-P_{ACK})/P_{ACK}^2$$

$$= 1 + (N * [1 - (1- P_b)^n])/ (1- P_b)^n$$

Selective-Repeat ARQ (SR ARQ)



Selective-Repeat ARQ (SR ARQ)

Throughput

$$S = (1/T) * (k/n)$$

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

where

$$T = 1 * P_{ACK} + 2 * P_{ACK} * (1- P_{ACK}) + 3 * P_{ACK} * (1- P_{ACK})^2 + \ldots.$$

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

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

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