Principles of Communications

Weiyao Lin
Shanghai Jiao Tong University

Chapter 11: Channel Coding

Textbook: Chapter 13.1 ~13.3

Homework 5

- Ch 12: 12.2, 12.9, 12.15, 12.37
- Ch 13: 13.9, 13.11, 13.13(1)(2)(3),
  13.15(1)(2)(3)(4), 13.17
- Homework questions available on
  - ftp://public.sjtu.edu.cn
  - Username: zhangyihao
    password: public
  - Filename: chapter12_problem.pdf
- Due: **Wednesday (Dec. 26)** In class
- Reminder: Project due **next Friday (Dec. 21)** by Email the TA

Reminder (1)

- My PRP project:
  - Title:多媒体搜索结果可视化与拼贴
    - Currently look for students
    - Plan to look for about 3 students
    - Each student will do his own work
      - Only one student will do this prp topic
      - Other 1-2 students will do other topics on video and image processing
    - Basic target: writing papers, creating useful systems, and hopefully some students can continue to pursue master in my group
      - Publish at least one paper on major conferences
      - Try to publish 2 papers before you finish your undergraduate study

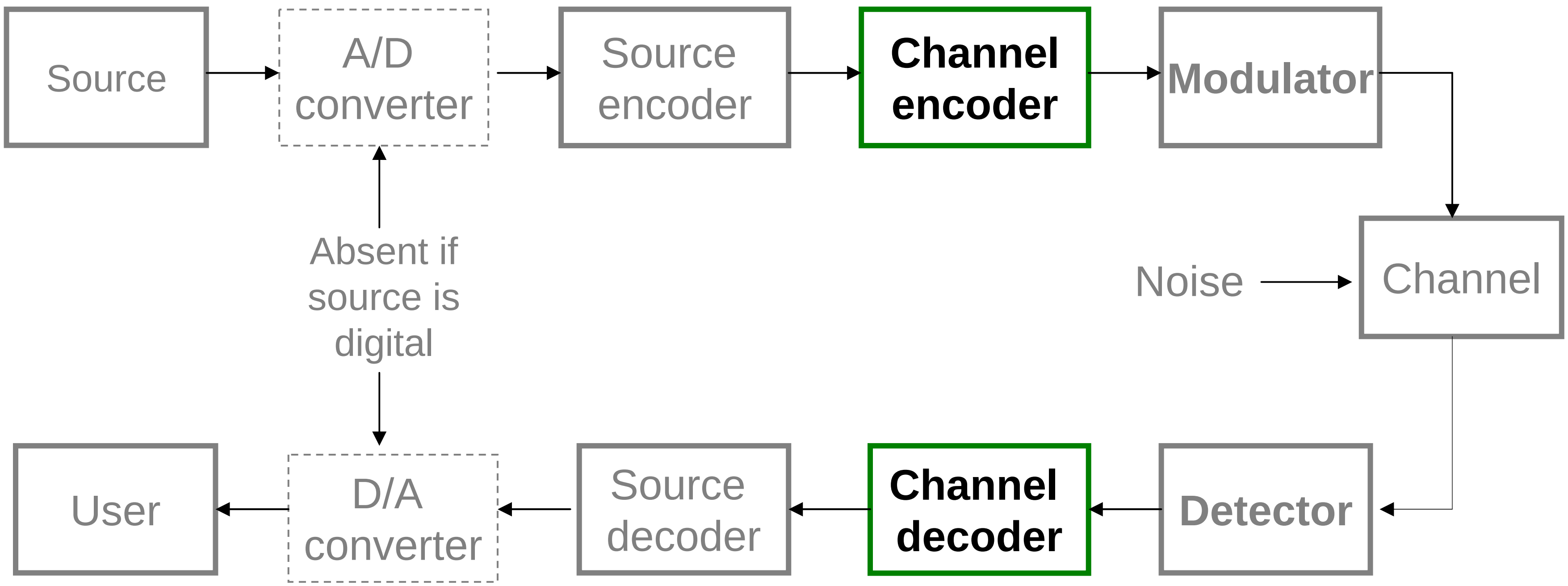
Reminders (2)

- Requirements and preference:
  - Good at English reading and understanding (good to read papers)
  - Self-motivated and quick working
  - Strong ability on programming
    - C (openCV), C++, Matlab
    - Optional: MFC, JAVA…
  - Good at Mathematics

Reminders (6)

- If interested, send your resume to hellomikelin@gmail.com or wylin@sjtu.edu.cn
- 简历里可以列出你们比较得意的经历和能力, 此外,请在简历里包含以下内容:
  - **(1)** 对将来本科毕业的打算, 是准备直接出国,工作, 还是准备继续在交大(特别能在我这里)继续读研究生
  - **(2)** 专业 (信工, 电科, 计算机, 联读班, 试点班, 联合学院,等等)
  - **(3)** 平均**GPA**,
  - **(4)** 上过的主要专业课和计算机编程的课的名字和分数 (如果上我的通信原理课,也把通信原理期中考试的分数列一下).
  - **(5)** 熟悉的编程语言和做过的一些主要的**projects**.
- Also welcome to apply my Summer project (暑期实习)
  - Can be paper writing or doing some projects

Topics to be Covered



- Linear block code
- Convolutional code

Information Theory and Coding

- In 1948, Claude Shannon **showed that** controlled redundancy in digital communications allows transmission at arbitrarily low bit error rate (BER)
- Error control coding (ECC) uses this controlled redundancy to detect and correct errors
- How to use error control coding depends on the system requirements (e.g. data, voice, video) and the nature of the channel (e..g wireless, mobile, high interference)
- The key in error control coding research is to
  - Find a way to add redundancy to the channel so that the receiver can fully utilize that redundancy to detect and correct the errors and to improve the coding gain – the effective lowering of the power required

Example

- We want to transmit data over a telephone link using a modem under the following conditions
  - Link bandwidth = 3kHz
  - The modem can operate up to the speed of 3600 bits/sec at an error probability $P_e = 8 \times 10^{-4}$

- Target: transmit the data at rate of 1200 bits/sec at maximum output SNR = 13 dB with a probability of error 10^{-4}

Solution: Shannon Theorem

- For the above communication link, the channel capacity is

$$C = B\log_2\left(1+\frac{S}{N}\right) = 13{,}000 \text{ bits/sec}$$

  Since B = 3000 and S/N = 20 (13 dB = $10\log_{10}20$)
- Thus, by Shannon's theorem, we can transmit the data with an arbitrarily small error probability
- Note that without coding $P_e = 8 \times 10^{-4}$

  For the given modem, criterion $P_e = 10^{-4}$ is <u>not met.</u>

Solution: A Simple Code Design

- Design a simple code which yields an overall probability of error of 10^{-4}

- Possible solution:

When b_k = "0" or "1", the codeword "000" or "111" is transmitted

- Based on the received codewords, the decoder attempts to extract the transmitted bits using majority-logic decoding scheme

Tx bits b_k	0	0	0	0	1	1	1	1
Codewords	000	000	000	000	111	111	111	111
Rx bits	000	001	010	100	011	101	110	111
$\widehat{b}_k$	0	0	0	0	1	1	1	1

- Clearly, the transmitted bits will be recovered correctly as long as no more than one of the bits in the codeword is affected by noise

- With this simple error control coding, the probability of error is

$$
\begin{aligned}
P_e &= P\left(b_k \neq \hat{b}_k\right) \\
&= P(2 \text{ or more bits in codeword are in error}) \\
&= \binom{3}{2} q_c^2 (1 - q_c) + \binom{3}{3} q_c^3 \\
&= 3q_c^2 - 2q_c^3 \\
&= 0.0192 \times 10^{-4} \leq \text{Required } P_e \text{ of } 10^{-4}
\end{aligned}
$$

Channel Coding

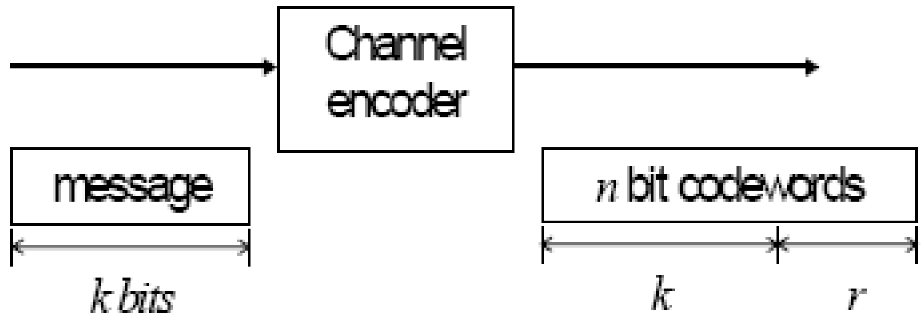
- Coding techniques are classified as either block codes or convolutional codes, depending on the presence or absence of memory
- A block code has no memory, since it collects and therefore isolate k-bit in a buffer prior to processing
  - The encoder adds n-k redundant bits to the buffered bits. The added bits are algebraically related to the buffered bits
  - The encoded block contains n bits, known as codeword
  - Each output codeword of an (n, k) block code depends only on the current buffer
  - The ratio k/n is known as the code rate. The difference 1-k/n is known as redundancy

Channel Coding (cont'd)

- A convolutional coder may process one or more buffers during an encoding cycle
  - If the number of sample points (buffers) > 1, a small FIFO is needed to hold them
  - The encoder acts on the serial bit stream as it enters the transmitter
  - Each bit in the output stream is not only dependent on the current bit, but also on those processed previously. This implies a form of memory
  - Performance of a convolutional coder is less sensitive to SNR variations than that of block codes

11.1 Block Codes

- An (n,k) block code is a collection of $M = 2^k$ codewords of length n.
- Each codeword has a block of k information bits followed by a group of r = n-k check bits that are derived from the k information bits in the block preceding the check bits



- The code is said to be linear if any linear combination of 2 codewords is also a codeword
  - i.e. if c_i and c_j are codewords, then $c_i + c_j$ is also a codeword (where the addition is always module-2)

- Code rate (rate efficiency) = $\dfrac{k}{n}$
- Matrix description

$\mathbf{c} = (c_1, c_2, \ldots, c_n)$ = codeword

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

- Each block code can be generated using a Generator Matrix $\mathbf{G}$ (dim: $k \times n$)
- Given $\mathbf{G}$, then

$$\boxed{\mathbf{c} = \mathbf{m}\mathbf{G}}$$

codeword message

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

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1,n-k} \\ 0 & 1 & & 0 & p_{21} & p_{22} & & p_{2,n-k} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} \end{bmatrix}$$

- I_k = identity matrix of order k
- P = matrix of order $k \times (n-k)$, which is selected so that the code will have certain desirable properties

Systematic Codes

- The form of G implies that the 1st k components of any codeword are precisely the information symbols
- This form of linear encoding is called systematic encoding
- Systematic-form codes allow easy implementation and quick look-up features for decoding
- For linear codes, any code is equivalent to a code in systematic form (given the same performance). Thus we can restrict our study to only systematic codes

Example: Hamming Code

- A family of (n,k) linear block codes that have the following parameters:
  - Codeword length $n = 2^m - 1, \quad m \geq 3$
  - # of message bits $k = 2^m - m - 1$
  - # of parity check bits $n - k = m$
  - Capable of providing single-error correction capability

(7, 4) Hamming Code

- Consider a (7,4) Hamming code with generator matrix

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$$

- Find all codewords

Solution

- Let $\mathbf{m} = [1\ 1\ 1\ 1]$

$$\mathbf{c} = \mathbf{mG} = [1\ 1\ 1\ 1]\left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$$

$$= [1\ 1\ 1\ 1\ 1\ 1\ 1]$$

- This example indicates that in general, the encoder must store G (or at least P) and then performs binary arithmetic operations to generate codewords
- Clearly, the complexity of encoder increases as n increases

List of all Codewords

- n = 7, k = 4 $\rightarrow 2^k = 16$ message blocks

Message				codeword						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	0	1	1	1
0	0	1	1	0	0	1	1	0	1	0
0	1	0	0	0	1	0	0	0	1	1
0	1	0	1	0	1	0	1	1	1	0
0	1	1	0	0	1	1	0	1	0	0
0	1	1	1	0	1	1	1	0	0	1
1	0	0	0	1	0	0	0	1	1	0
1	0	0	1	1	0	0	1	0	1	1
1	0	1	0	1	0	1	0	0	0	1
1	0	1	1	1	0	1	1	1	0	0
1	1	0	0	1	1	0	0	1	0	1
1	1	0	1	1	1	0	1	0	0	0
1	1	1	0	1	1	1	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1

Parity Check Matrix

- For each $\mathbf{G}$, it is possible to find a corresponding parity check matrix $\mathbf{H}$

$$\mathbf{H} = \left[\mathbf{P}^T \;|\mathbf{I}_{n-k}\right]_{(n-k)\times n}$$

- $\mathbf{H}$ is important since it can be used to verify if a codeword C is generated G
- Let C be a codeword generated by $\mathbf{G} = [\mathbf{I}_k|\mathbf{P}]_{k\times n}$

$$\Rightarrow \quad \boxed{\mathbf{c}\mathbf{H}^T = \mathbf{m}\mathbf{G}\mathbf{H}^T = 0}$$

Error Syndrome

- Received codeword $\mathbf{r} = \mathbf{c} + \mathbf{e}$

  where $\mathbf{e}$ = Error vector or Error Pattern

  it is 1 in every position where data word is in error

- Example

$$\mathbf{c} = [1\ \ 0\ \ 1\ \ 0]$$

$$\mathbf{r} = [1\ \ 1\ \ 0\ \ 0]$$

$$\mathbf{e} = [0\ \ 1\ \ 1\ \ 0]$$

Error Syndrome (cont'd)

- $\mathbf{s} \triangleq \mathbf{r}\mathbf{H}^T$ = Error Syndrome
- But

$$\begin{aligned}\mathbf{s} = \mathbf{r}\mathbf{H}^T &= (\mathbf{c}+\mathbf{e})\mathbf{H}^T \\ &= \mathbf{c}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T \\ &= \mathbf{e}\mathbf{H}^T\end{aligned}$$

1. If s=0 $\rightarrow$ r = c and $\mathbf{m}$ is the 1^{st} k bits of r
2. If s $\neq$0, and s is the j^{th} row of $\mathbf{H}^T \rightarrow$ 1 error in jth position of r

- Consider the (7,4) Hamming code example

$$\mathbf{H}^T = \left[\mathbf{P}^T | \mathbf{I}_{n-k}\right]^T = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ \hdashline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- So if $\mathbf{r} = [1\ 1\ 1\ 1\ 1\ 1\ 1]$

$$\Longrightarrow \quad \mathbf{r}\mathbf{H}^T = [0\ 0\ 0]$$

- But if $\mathbf{r} = [1\ 1\ 1\ 1\ 1\ 1\ 0]$

$$\Longrightarrow \quad \mathbf{r}\mathbf{H}^T = [0\ 0\ 1]$$

= Error syndrome **s**

- Note that s is the last row of H^T
- Also note error took place in the last bit

=> Syndrome indicates position of error

Cyclic Codes

- A code $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2^k}\}$ is cyclic if

$$(c_1, c_2, \ldots, c_n) \in C \quad \Rightarrow \quad (c_n, c_1, \ldots, c_{n-1}) \in C$$

- (7,4) Hamming code is cyclic

message	codeword
0001	0001101
1000	1000110
0100	0100011

Important Parameters

- The Hamming Distance between codewords c_i and c_j is:

  $d(c_i, c_j) =$ # of components at which the 2 codewords differ

- The Hamming weight of a codeword c_i is

  $w(c_i) =$ # of non-zero components in the codeword

- The minimum Distance of a code is the minimum Hamming distance between any 2 codewords

$$d_{min} = \min d(c_i, c_j) \quad \text{for all } i \neq j$$

- The minimum Weight of a code is the minimum of the weights of the codewords except the all-zero codeword

$$w_{min} = \min w(c_i) \quad \text{for all } c_i \neq 0$$

- Theorem: In any linear code, $d_{\text{min}} = w_{\text{min}}$

- Example: Find d_{min} for (7,4) Hamming code

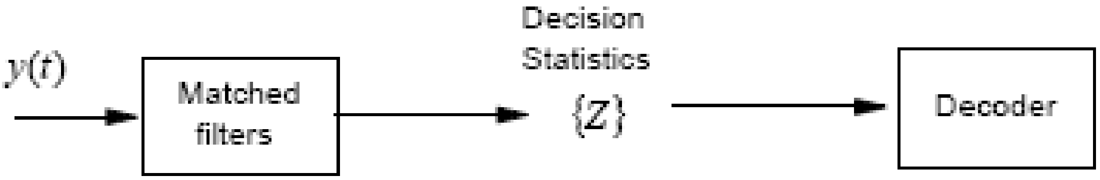
Major Classes of Block Codes

- Repetition Code
- Hamming Code
- Golay Code
- BCH Code
- Reed-Solomon Codes
- Walsh Codes

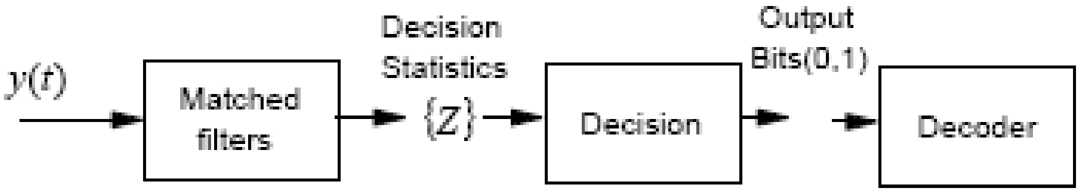
BCH and RS codes are the most frequently used

Soft-Decision and Hard-Decision Decoding

- **Soft-decision decoder** operates directly on the decision statistics



- **Hard-decision decoder** makes “hard” decision (0 or 1) on individual bits

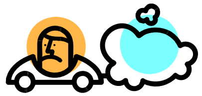


- Here we only focus on hard-decision decoder

Hard-Decision Decoding

- Minimum Hamming Distance Decoding
  - Given the received codeword $\mathbf{r}$, choose c which is closest to r in terms of Hamming distance
  - To do so, one can do an exhaustive search

    – too much if n is large.



- Syndrome Decoding
  - Syndrome testing: $\mathbf{r} = \mathbf{c} + \mathbf{e}$ with $\mathbf{s} = \mathbf{r}\mathbf{H}^T$
  - This implies that the corrupted codeword r and the error pattern have the same syndrome
  - A simplified decoding procedure based on the above observation can be used

Standard Array

- Let the codewords be denoted as $\{c_1, c_2, \ldots, c_M\}$
- A standard array is constructed as

Coset leader

coset

Syndrome s

$$2^{n-k}\left\{\begin{array}{llll} \multicolumn{4}{c}{\overbrace{\hphantom{c_1 \quad e_1\oplus c_2 \quad\quad e_1\oplus c_M}}^{2^k}} \\ c_1 & c_2 & & c_M \\ e_1 & e_1 \oplus c_2 & & e_1 \oplus c_M \\ e_2 & e_2 \oplus c_2 & & e_2 \oplus c_M \\ \vdots & \vdots & & \vdots \\ e_{2^{n-k}-1} & e_{2^{n-k}-1} \oplus c_2 & & e_{2^{n-k}-1} \oplus c_M \end{array}\right. \Rightarrow \begin{array}{l} 0 \\ s = e_1 H^T \\ s = e_2 H^T \\ \vdots \\ s = e_{2^{n-k}-1} H^T \end{array}$$

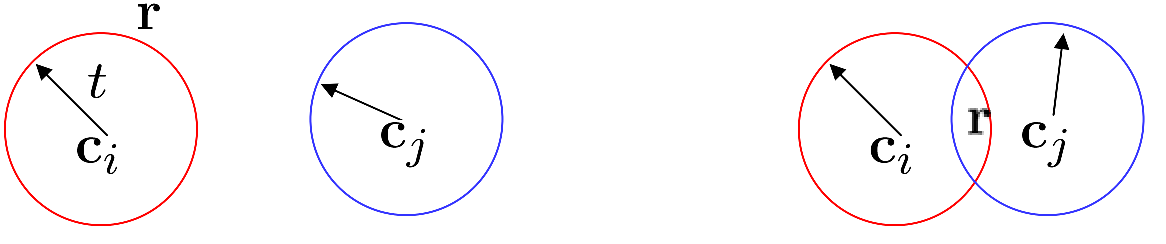
Error patterns

Hard-Decoding Procedure

- Find the syndrome by r using $\mathbf{s} = \mathbf{r}\mathbf{H}^T$
- Find the coset corresponding to s by using the standard array
- Find the cost leader and decode as $\mathbf{c} = \mathbf{r} + \mathbf{e}_j$

Error Correction Capability

- A linear block code with a minimum distance d_{min} can
  - Detect up to $(d_{\min} - 1)$ errors in each codeword
  - Correct up to $t = \lfloor \frac{d_{\min} - 1}{2} \rfloor$ errors in each codeword
  - t is known as the error correction capability of the code



$d(\mathbf{c}_i, \mathbf{c}_j) \geq 2t + 1$ $\qquad$ $d(\mathbf{c}_i, \mathbf{c}_j) < 2t$

Probability of Codeword Error for Hard-Decision Decoding

- Consider a linear block code (n, k) with an error correcting capability *t*. The decoder can correct all combination of errors up to and including t errors.
- Assume that the error probability of each individual coded bit is *p* and that bit errors occur independently since the channel is memoryless
- If we send n-bit block, the probability of receiving a specific pattern of m errors and (n-m) correct bits is

$$p^m(1-p)^{n-m}$$

- Total number of distinct pattern of n bits with m errors and (n-m) correct bits is

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

- So the total probability of receiving a pattern with m errors is

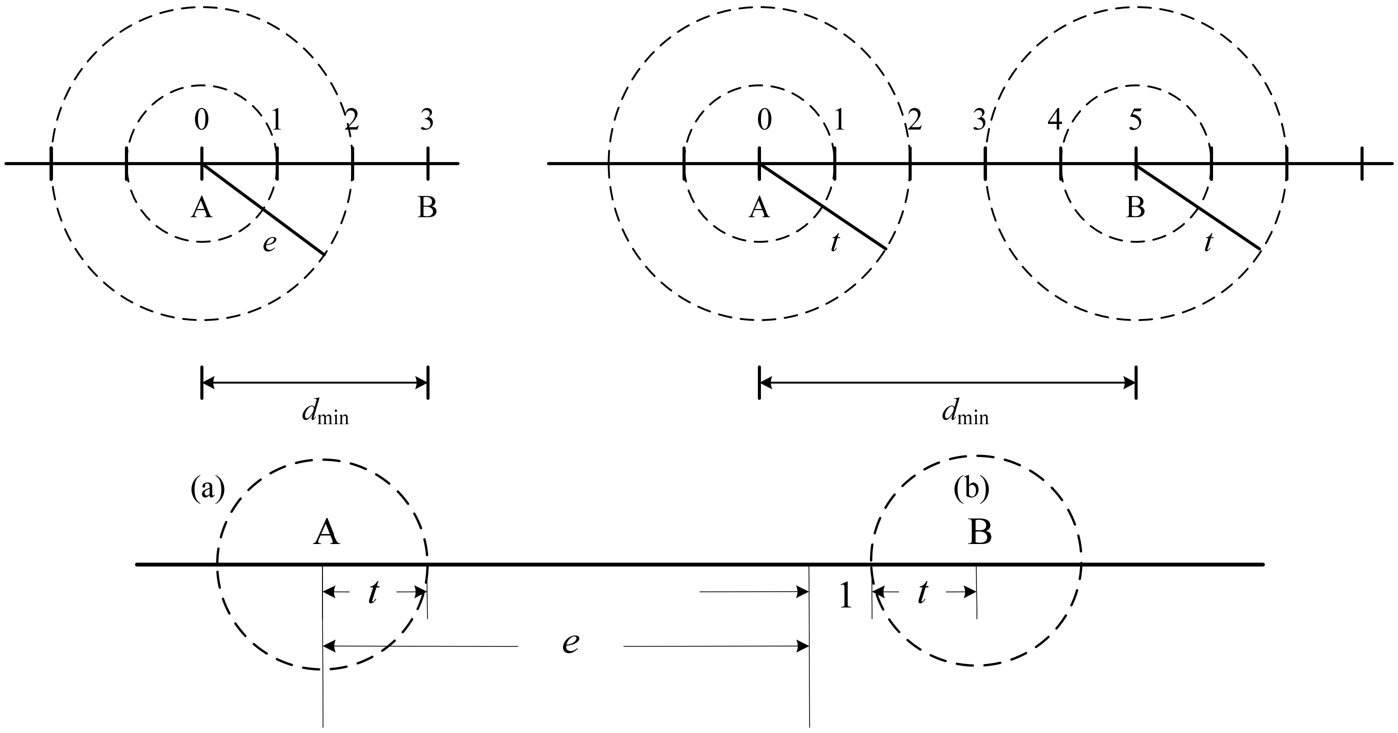
$$P(m,n) = \binom{n}{m} = p^m(1-p)^{n-m}$$

- Therefore, the codeword error probability is upper-bounded by

$$P_M \le \sum_{m=t+1}^{n} \binom{n}{m} p^m (1-p)^{n-m}$$

with equality for perfect codes

Error Detection vs. Error Correction



(c)

- To detect e bit errors, we have $d_{\min} \geq e + 1$
- To correct t bit errors, we have $d_{\min} \geq 2t + 1$

11.2 Convolutional Codes

- A convolutional code has memory
  - It is described by 3 integers: n, k, and K
  - Maps k bits into n bits using previous (K-1) k bits
  - The n-tuple emitted by the encoder is not only a function of the current input k-tuple, but is also a function of the previous K-1 input k-tuples
  - k/n = Code Rate (information bits/coded bit)
  - K is the constraint length and is a measure of the code memory
  - n does not define a block or codeword length

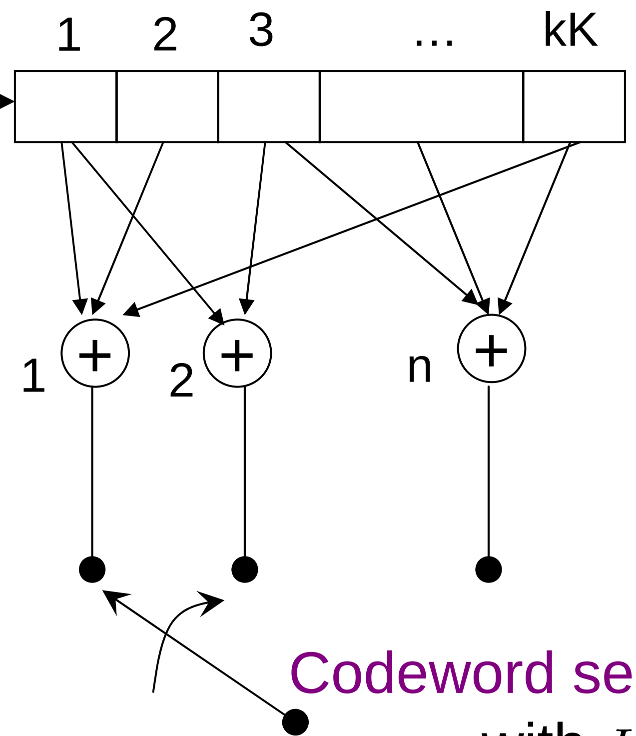
Convolutional Encoding

- A rate k/n convolutional encoder with constraint length K consists of
  - kK-stage shift register and n mod-2 adders
- At each unit of time:
  - k bits are shifted into the 1st k stages of the register
  - All bits in the register are shifted k stages to the right
  - The output of the n adders are sequentially sampled to give the coded bits
  - There are n coded bits for each input group of k information or message bits. Hence R = k/n information bits/coded bit is the code rate (k<n)

Encoder Structure
(rate k/n, constraint length K)

$\mathbf{m} = m_1, m_2, \ldots, m_i, \ldots$

Input sequence
(shift in k at a time)



kK-stage shift register

n modulo-2 adders

Codeword sequence $\mathbf{U} = U_1, U_2, \ldots, U_j, \ldots$

with $U_j = u_{1,j}, u_{2,j}, \ldots, u_{n,j}$

$u_{i,j}$ = i-th binary code symbol of the branch word U_j

Typically binary codes with k=1 are used. Hence, we will mainly consider rate 1/n codes

Conv. Codes Representation

- To describe a convolutional code, we must describe the encoding function that characterizes the relationship between the information sequence **m** and the output coded sequence **U**
- There are 4 popular methods for representation
  - ✓ Connection pictorial and connection polynomials (usually for encoder)
  - ✓ State diagram
  - Tree diagram
  - ✓ Trellis diagram

Usually for decoder

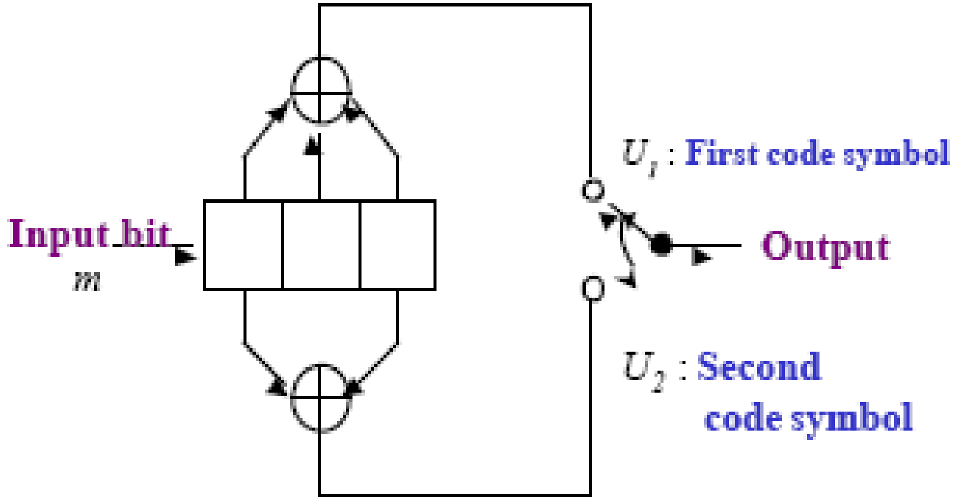
Connection Representation

- Specify n connection vectors, $\mathrm{g}_i, i = 1, \ldots, n$ for each of the n mod-2 adders
- Each vector has kK dimension and describes the connection of the shift register to the mod-2 adders
- A 1 in the i^{th} position of the connection vector implies shift register is connected
- A 0 implies no connection exists

Example: K = 3, Rate 1/2

$g_1 = 1\ 1\ 1$
$g_2 = 1\ 0\ 1$

Or
$g_1(X)=1+X+X^2$
$g_2(X)=1+X^2$

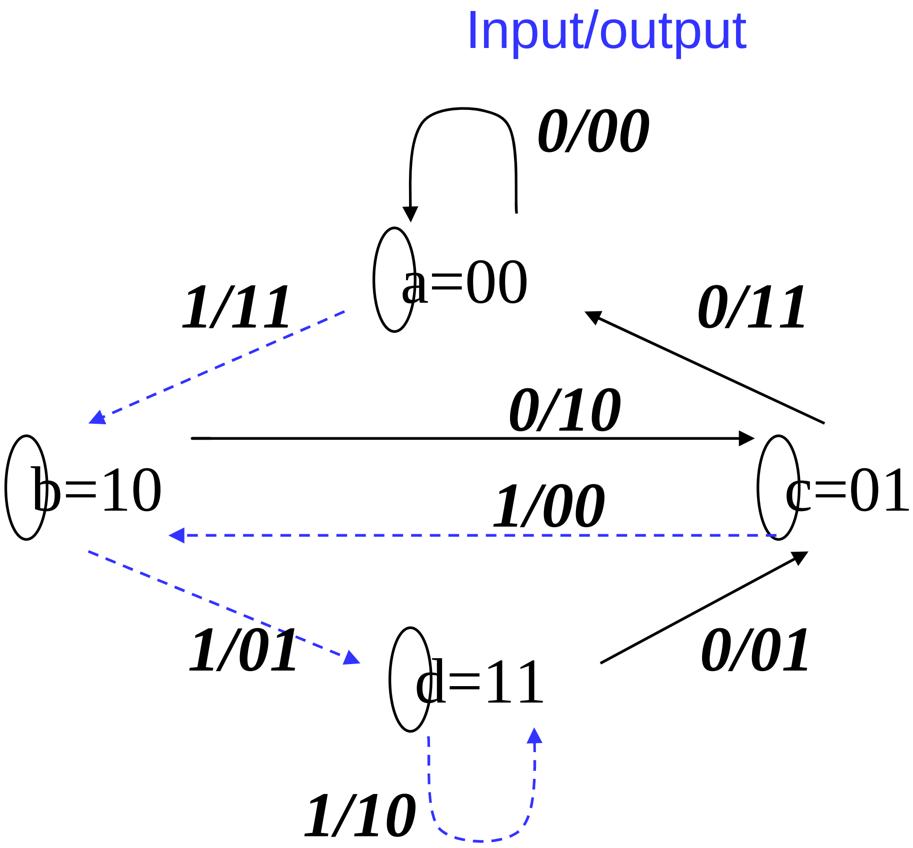


If Initial Register content is 0 0 0 and Input Sequence is 1 0 0. Then Output sequence is 11 10 11.

State Diagram Representation

- The contents of the rightmost K-1 stages (or the previous K-1 bits) are considered the current state => 2^{K-1} states
- Knowledge of the current state and the next input is necessary and sufficient to determine the next output and next state
- For each state, there are only 2 transitions (to the next state) corresponding to the 2 possible input bits
- The transitions are represented by paths on which we write the output word associated with the state transition
  - A solid line path corresponds to an input bit 0
  - A dashed line path corresponds to an input bit 1

Example: K =3, Rate = 1/2



Current State	Input	Next State	Output
00	0	00	***00***
	1	10	***11***
10	0	01	***10***
	1	11	***01***
01	0	00	***11***
	1	10	***00***
11	0	01	***01***
	1	11	***10***

Example

- Assume that **m=11011** is the input followed by K-1 = 2 zeros to flush the register. Also assume that the initial register contents are all zero. Find the output sequence **U**.

Input bit m_i	Register contents	State at time t_i	State at time t_{i+1}	Branch word at time t_i: u_1	u_2
-	0 0 0	0 0	0 0		
1	1 0 0	0 0	1 0	1	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1

State t_i

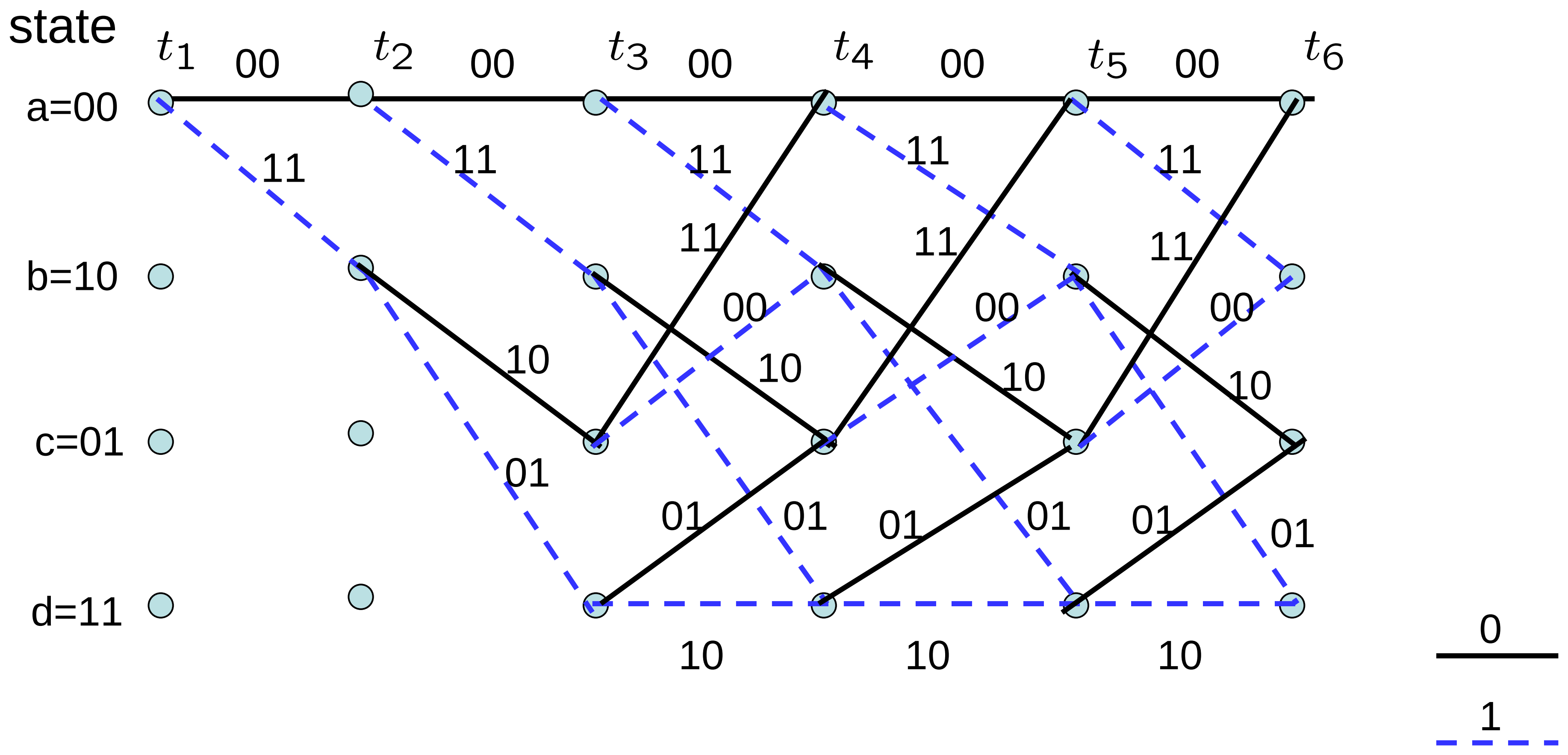
State t_{i+1}

Output sequence: **U** = **11 01 01 00 01 01 11**

Trellis Diagram

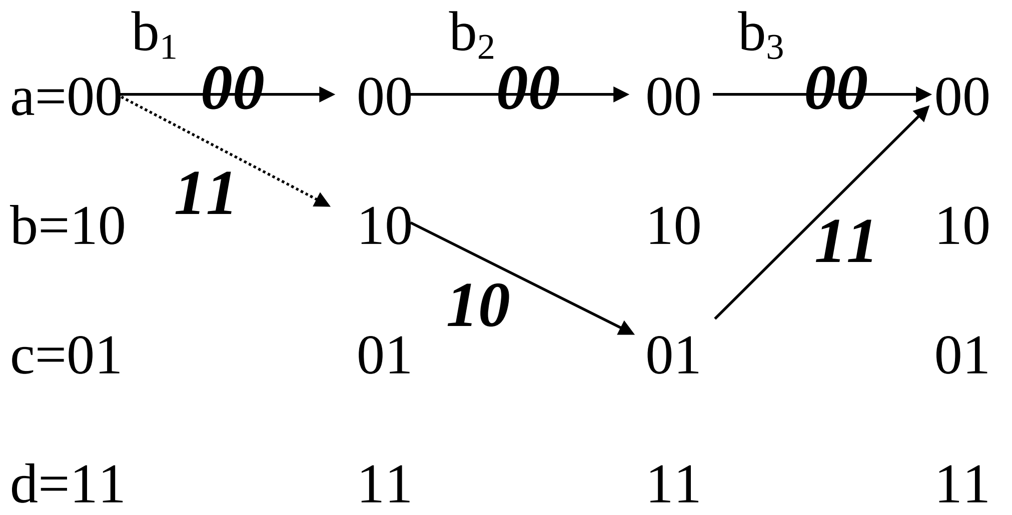
- The trellis diagram is similar to the state diagram, except that it adds the dimension of time
- The code is represented by a trellis where each trellis branch describes an output word

Trellis Diagram



Trellis structure repeats itself after depth K = 3

- Every input sequence $(m_1, m_2, \ldots)$ corresponds to
  - a path in the trellis
  - a state transition sequence $(s_0, s_1, \ldots)$, (assume $s_0=\mathbf{0}$ is fixed),
  - an output sequence $((u_1, u_2), (u_3, u_4), \ldots)$
- Example: Let $s_0=00$, then
  - $b_1b_2b_3=000$ gives output 000000 and states aaaa
  - $b_1b_2b_3=100$ gives output 111011 and states abca



- We have introduced conv. code
  - Constraint length K and rate R = 1/n
  - Polynomials representation
  - State diagram representation
  - Trellis diagram representation

- We will talk about decoding of convolutional code
  - Maximum Likelihood Decoding
  - Viterbi Algorithm
  - Transfer Function

Maximum Likelihood Decoding

- Transmit a coded sequence $\mathbf{U}^{(\mathbf{m})}$ (correspond to message sequence $\mathbf{m}$) using a digital modulation scheme (e.g. BPSK or QPSK)
- Received sequence $\mathbf{Z}$
- Maximum likelihood decoder
  - Find the sequence $\mathbf{U}^{(\mathbf{j})}$ such that

$$P(\mathbf{Z}|\mathbf{U}^{j}) = \max_{\forall \mathbf{U}^{(m)}} P(\mathbf{Z}|\mathbf{U}^{(m)})$$

  - Will minimize the probability of error if $\mathbf{b}$ is equally likely

Maximum Likelihood Metric

- Assume a memoryless channel, i.e. noise components are independent. Then, for a rate 1/n code

$$P(\mathbf{Z}|\mathbf{U}^{(\mathbf{m})}) = \prod_{i=1}^{\infty} P(Z_i|U_i^{(m)}) = \prod_{i=1}^{\infty}\prod_{j=1}^{n} P(z_{ji}|u_{ji}^{(m)})$$

i-th branch of **Z**

- Then the problem to find a path (each path defines a codeword) through the trellis such that

$$\max_{\mathbf{U}^{(m)}} \prod_{i=1}^{\infty}\prod_{j=1}^{n} P(z_{ji}|u_{ji}^{(m)})$$

by taking log

$$\max_{\mathbf{U}^{(m)}} \boxed{\sum_{i=1}^{\infty}\sum_{j=1}^{n} \log P(z_{ji}|u_{ji}^{(m)})}$$

Log-likelihood path metric

i-th branch metric

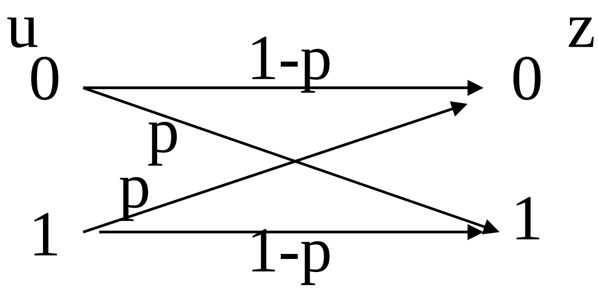
$$= \max_{\mathbf{U}^{(m)}} \sum_{i=1}^{\infty}\sum_{j=1}^{n} LL\left(z_{ji}|u_{ji}^{(m)}\right)$$

Log-likelihood of $z_{ji}|u_{ji}^{(m)}$

Decoding Algorithm: Log-Likelihood

- For AWGN channel (soft-decision)
  - $z_{ji} = u_{ji} + n_{ji}$ and P($z_{ji}|u_{ji}$) is Gaussian with mean u_{ji} and variance σ^2
  - Hence
$$\ln p(z_{ji}|u_{ji}) = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(z_{ji}-u_{ji})^2}{2\sigma^2}$$
  - Note that the objective is to compare which Σ_i ln(p(z|u)) for different **u** is larger, hence, constant and scaling does not affect the results
  - Then, we let the log-likelihood be $LL(z_{ji}|u_{ji}) = -(z_{ji}-u_{ji})^2$
  - and
$$\boxed{\log P(Z|U^{(m)}) = -\sum_{i=1}^{\infty}\sum_{j=1}^{n}\left(z_{ji}-u_{ji}^{(m)}\right)^2}$$
  - Thus, soft decision ML decoder is to choose the path whose corresponding sequence is at the minimum Euclidean distance to the received sequence

- For binary symmetric channel (hard decision)



$$p(z \mid u) = \begin{cases} p & \text{if } z \neq u \\ 1-p & \text{if } z = u \end{cases}$$

$$LL(z_{ji} \mid u_{ji}) = \ln p(z_{ji} \mid u_{ji}) = \begin{cases} \ln p & \text{if } z_{ji} \neq u_{ji_i} \\ \ln(1-p) & \text{if } z_{ji} = u_{ji} \end{cases}$$

$$= \begin{cases} \ln p/(1-p) & \text{if } z_{ji} \neq u_{ji} \\ 0 & \text{if } z_{ji} = u_{ji} \end{cases}$$

$$= \begin{cases} -1 & \text{if } z_{ji} \neq u_{ji} \\ 0 & \text{if } z_{ji} = u_{ji} \end{cases}$$ (as since p<0.5)

- Thus

$$\boxed{\log P(Z|U^{(m)}) = -d_m}$$

Hamming distance between Z and $U^{(m)}$, i.e. they differ in d_m positions

Hard-Decision ML Decoder = Minimum Hamming Distance Decoder

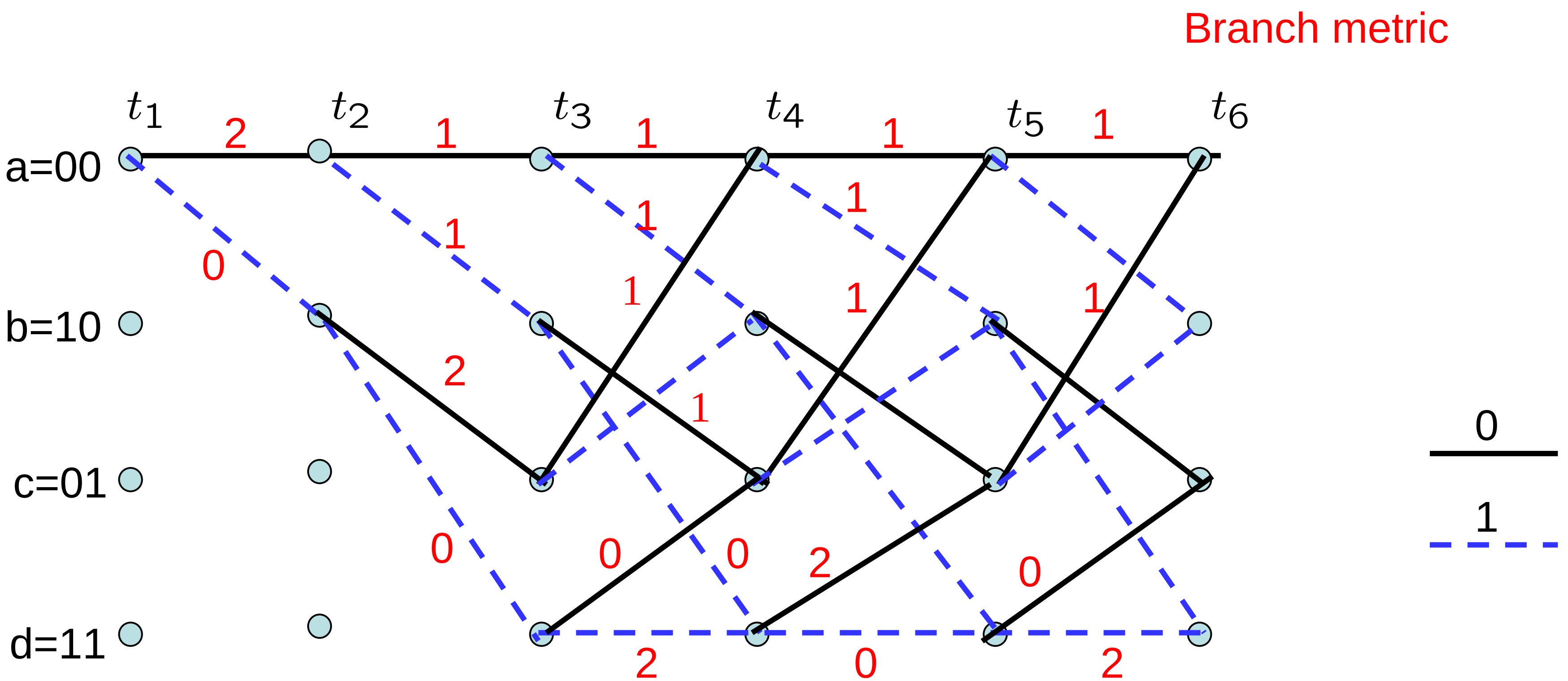
- Maximum Likelihood Decoding Procedure
  - Compute, for each branch i, the branch metric using the output bits $\{u_{1,i}, u_{2,i}, \ldots, u_{n,i}\}$ associated with that branch and the received symbols $\{z_{1,i}, z_{2,i}, \ldots, z_{n,i}\}$
  - Compute, for each valid path through the trellis (a valid codeword sequence $\mathbf{U}^{(m)}$), the sum of the branch metrics along that path
  - The path with the maximum path metric is the decoded path
- To compare all possible valid paths we need to do exhaustive search or brute-force, not practical as the # of paths grow exponentially as the path length increases
- The optimum algorithm for solving this problem is the **Viterbi decoding algorithm** or **Viterbi decoder**

Viterbi Decoding (R=1/2, K=3)

Input data sequence **m**: 1 1 0 1 1 …

Coded sequence **U**: 11 0 1 01 00 01 …

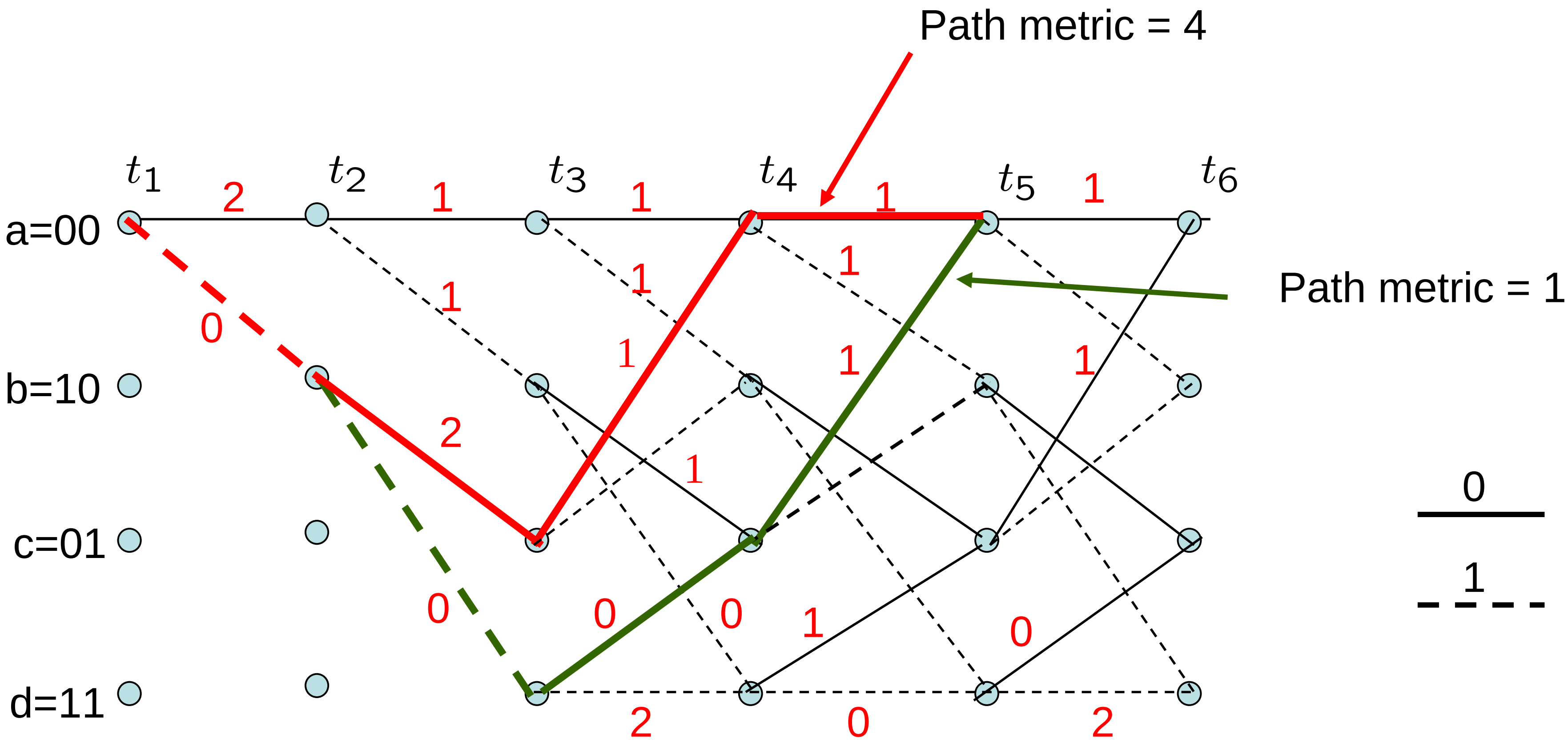
Received sequence **Z**: 11 0 1 01 10 01 …



Viterbi Decoder

- Basic idea:
  - If any 2 paths in the trellis merge to a single state, one of them can always be eliminated in the search
- Let cumulative path metric of a given path at t_i = sum of the branch metrics along that path up to time t_i
- Consider t_5
  - The upper path metric is 4, the lower math metric is 1
  - The upper path metric CANNOT be part of the optimum path since the lower path has a lower metric
  - This is because future output branches depend only on the current state and not the previous state

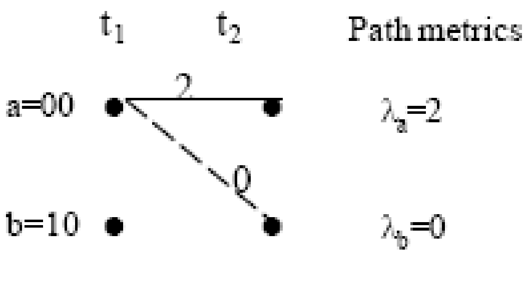
Path Metrics for 2 Merging Paths



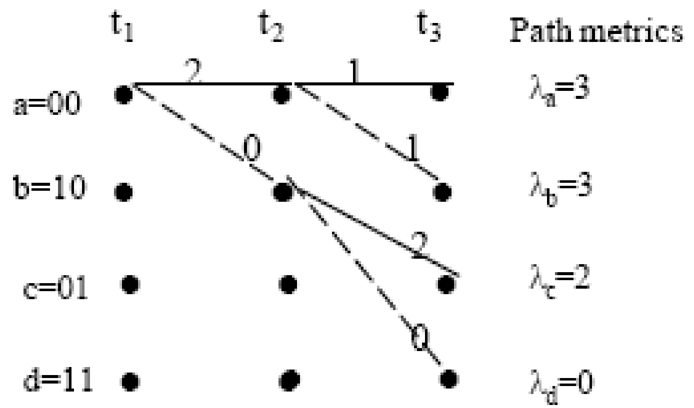
Viterbi Decoding

- At time t_i, there are 2^{K-1} states in the trellis
- Each state can be entered by means of 2 states
- Viterbi Decoding consists of computing the metrics for the 2 paths entering each state and eliminating one of them
- This is done for each of the 2^{K-1} nodes at time t_i
- The decoder then moves to at time t_{i+1} and repeats the process

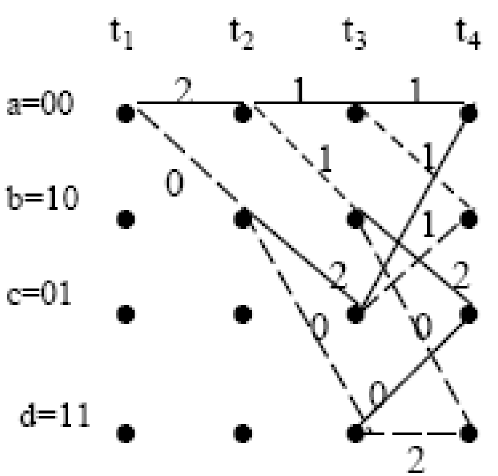
Example

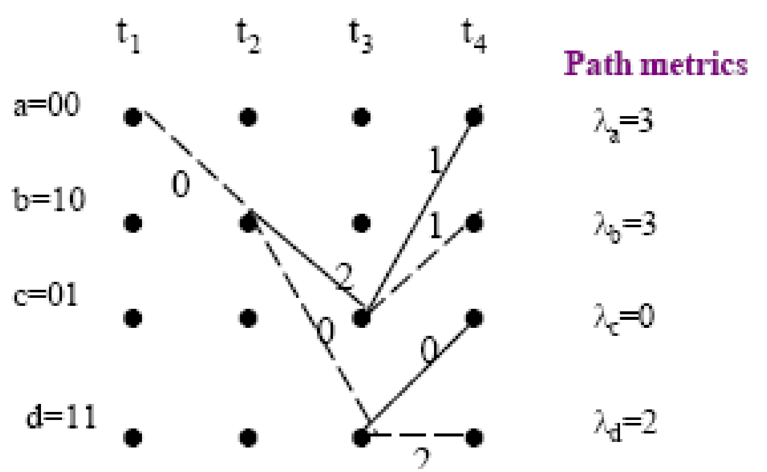

t1
t2
Path metrics
a=00
b=10
2
0
λa=2
λb=0

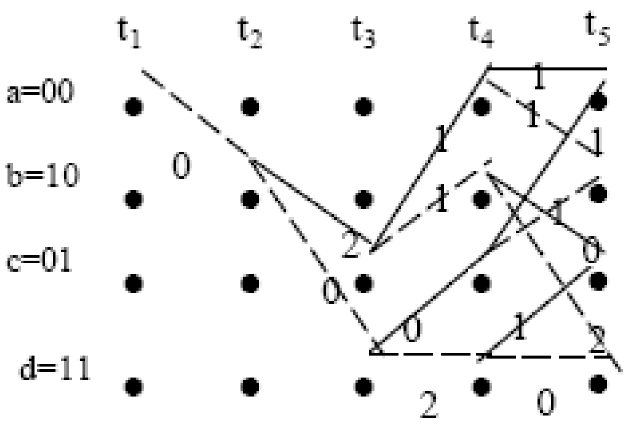
(a)


t1
t2
t3
Path metrics
a=00
b=10
c=01
d=11
2
1
0
1
2
0
λa=3
λb=3
λc=2
λd=0

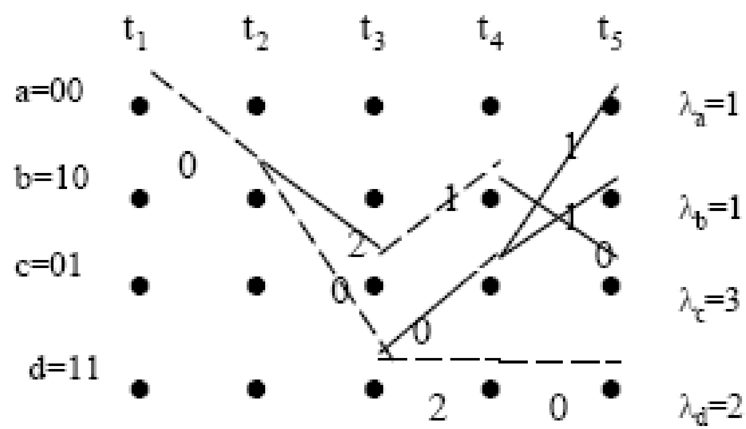
(b)


t1
t2
t3
t4
a=00
b=10
c=01
d=11
2
1
1
0
1
1
1
2
2
0
0
0
2

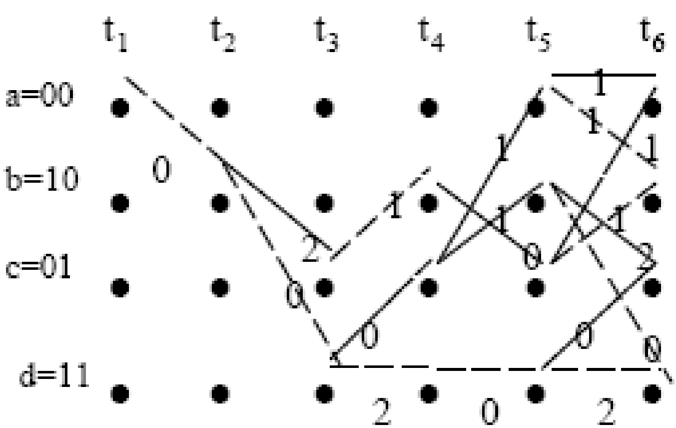

t1
t2
t3
t4
Path metrics
a=00
b=10
c=01
d=11
0
1
1
2
0
0
2
λa=3
λb=3
λc=0
λd=2

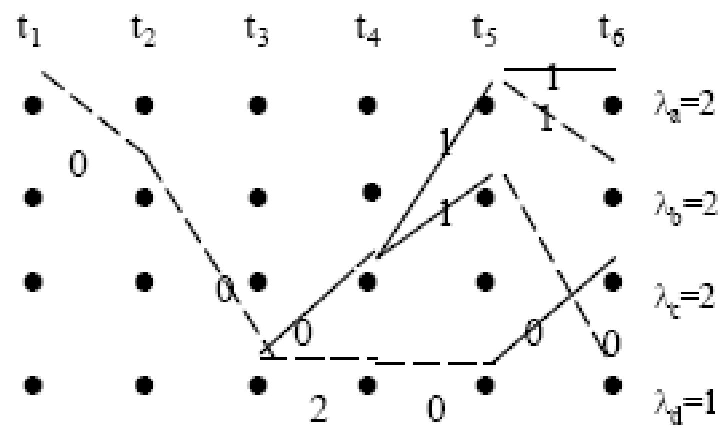

t1 t2 t3 t4 t5
a=00
b=10
c=01
d=11

(e)


t1 t2 t3 t4 t5
a=00
b=10
c=01
d=11
λa=1
λb=1
λc=3
λd=2

(f)

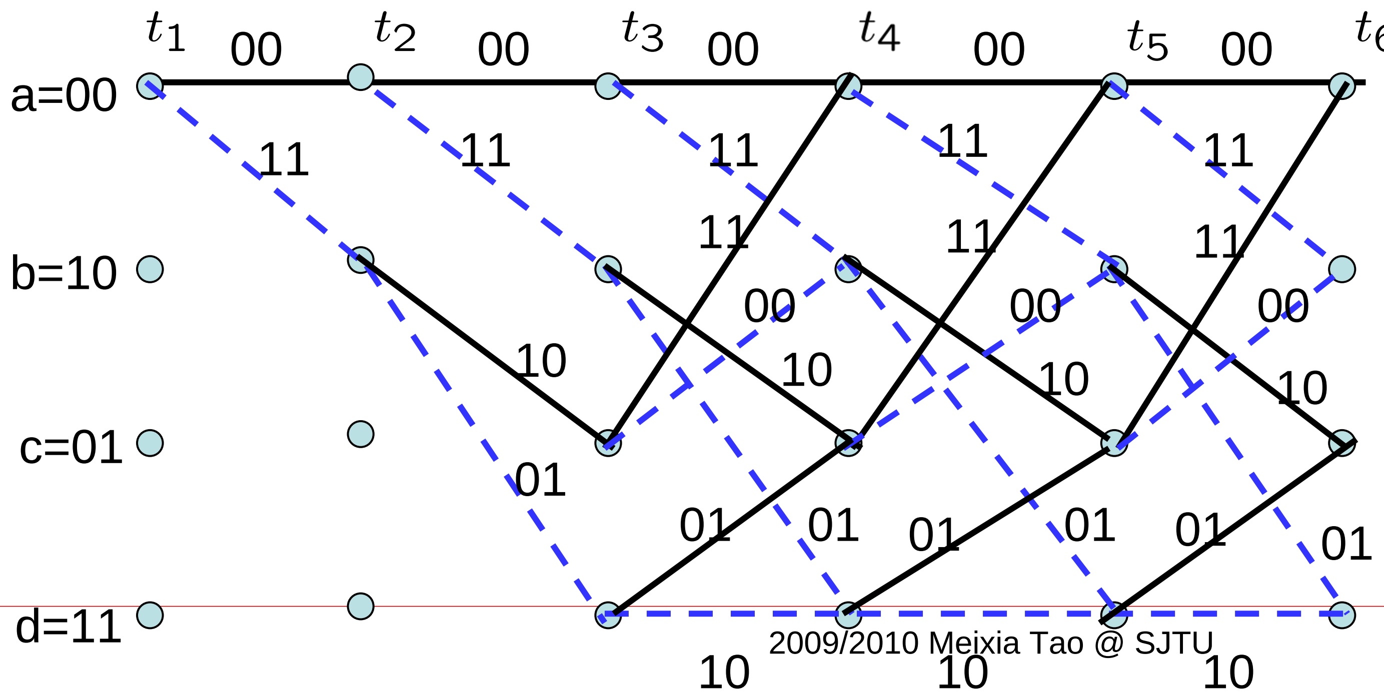

t1 t2 t3 t4 t5 t6
a=00
b=10
c=01
d=11


t1 t2 t3 t4 t5 t6
λa=2
λb=2
λc=2
λd=1

Distance Properties

- d_{free}= Minimum Free distance = Minimum distance of any pair of arbitrarily long paths that diverge and remerge
- A code can correct any t channel errors where (this is an approximation)

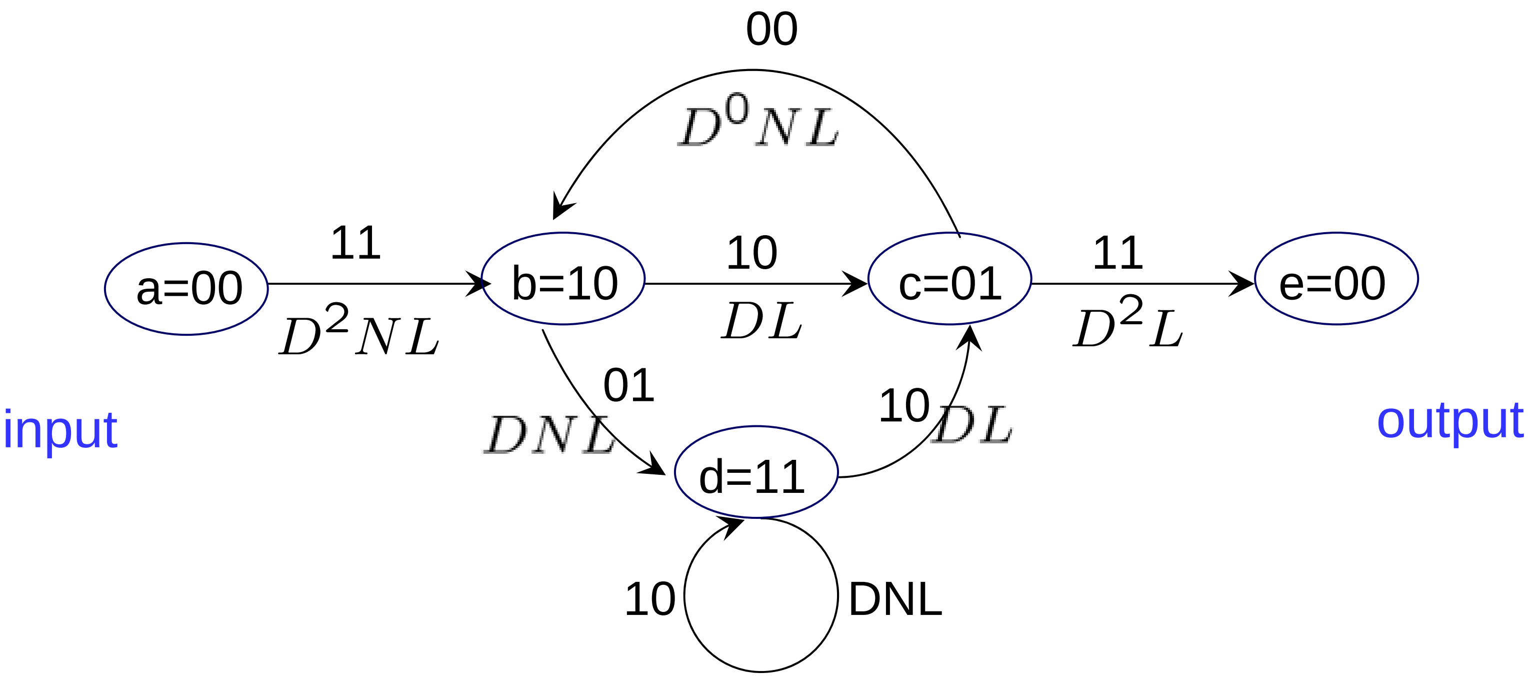
$$t \le \lfloor \frac{d_{\mathrm{free}} - 1}{2} \rfloor$$



Transfer Function

- The distance properties and the error rate performance of a convolutional code can be obtained from its transfer function
- Since a convolutional code is linear, the set of Hamming distances of the code sequences generated up to some stages in the trellis, from the all-zero code sequence, is the same as the set of distances of the code sequences with respect to any other code sequence
- Thus, we assume that the all-zero path is the input to the encoder

State Diagram Labeled according to distance from all-zero path



- D^m denote m non-zero output bits
- N if the input bit is non-zero
- L denote a branch in the path

$$\begin{cases} X_b = D^2NLX_a + LNX_c \\ X_c = DLX_b + DLX_d \\ X_d = DNLX_b + DNLX_d \\ X_e = D^2LX_c \end{cases}$$

- The transfer function T(D,N,L), also called the weight enumerating function of the code is

$$T(D,N,L) = \frac{X_e}{X_a}$$

- By solving the state equations we get

$$T(D,N,L) = \frac{D^5 N L^3}{1 - DNL(1+L)}$$

$$= D^5 N L^3 + D^6 N^2 L^4 (1+L) + D^7 N^3 L^5 (1+L)^2$$

$$+ \ldots + D^{l+5} N^{l+1} L^{l+3} (1+L)^l + \ldots$$

- The transfer function indicates that:
  - There is one path at distance 5 and length 3, which differs in 1 input bit from the correct all-zeros path
  - There are 2 paths at distance 6, one of which is of length 4, the other length 5, and both differ in 2 input bits from all-zero path
  - $d_{\text{free}} = 5$

Known Good Convolutional Codes

- Good convolutional codes can only be found in general by computer search
- There are listed in tables and classified by their constraint length, code rate, and their generator polynomials or vectors (typically using octal notation).
- The error-correction capability of a convolutional code increases as n increases or as the code rate decreases.
- Thus, the channel bandwidth and decoder complexity increases

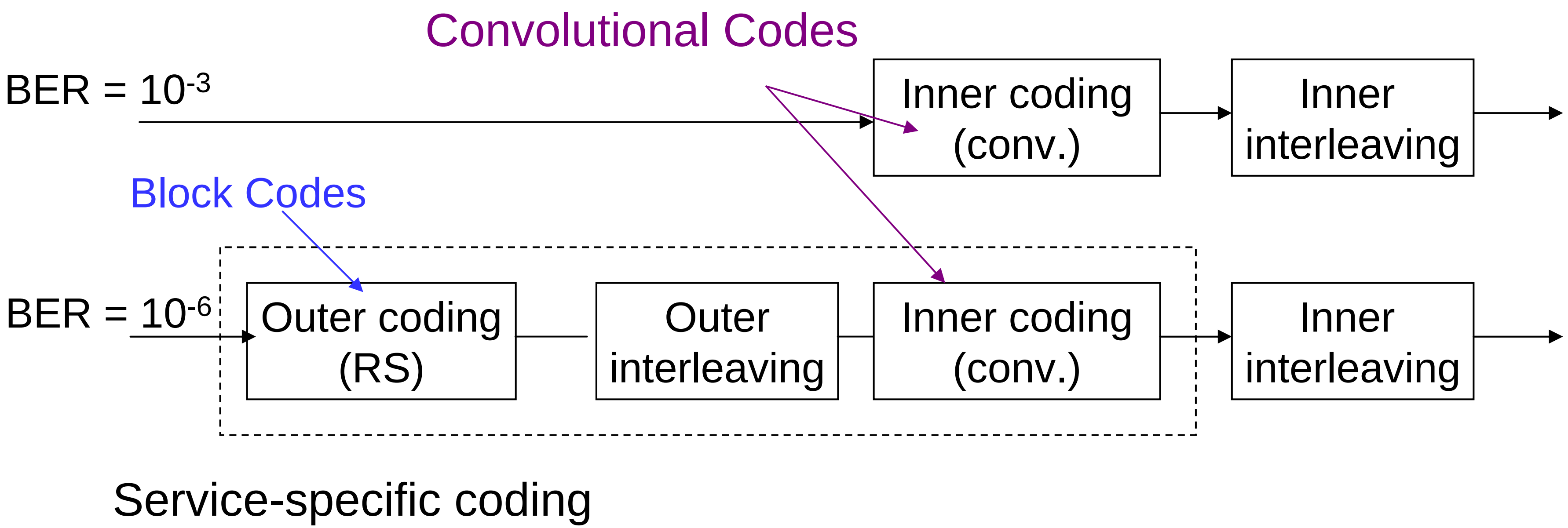
Good Codes with Rate 1/2

Constraint Length	Generator Polynomials	d_{free}
3	(5,7)	5
4	(15,17)	6
5	(23,35)	7
6	(53,75)	8
7	(133,171)	10
8	(247,371)	10
9	(561,753)	12
10	(1167,1545)	12

Good Codes with Rate 1/3

Constraint Length	Generator Polynomials	d_{free}
3	(5,7,7)	8
4	(13,15,17)	10
5	(25,33,37)	12
6	(47,53,75)	13
7	(133,145,175)	15
8	(225,331,367)	16
9	(557,663,711)	18
10	(1117,1365,1633)	20

Basic Channel Coding for Wideband CDMA



Convolutional code is rate 1/3 and rate ½, all with constraint length 9

Channel Coding for Wireless LAN (IEEE802.11a)

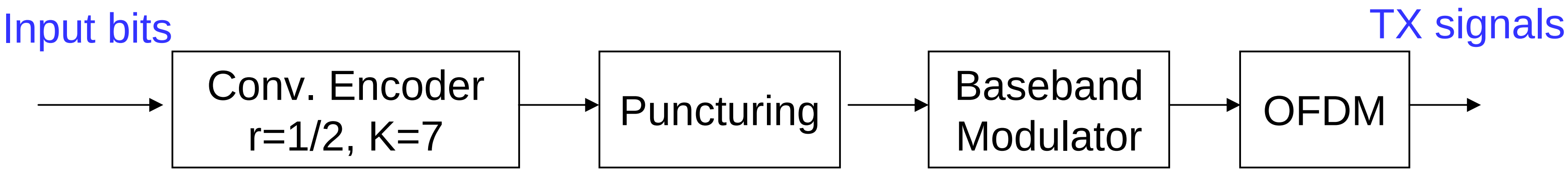


Table 11-3. Encoding details for different OFDM data rates

Speed (Mbps)	Modulation and coding rate (R)	Coded bits per carrier[a]	Coded bits per symbol	Data bits per symbol[b]
6	BPSK, R=1/2	1	48	24
9	BPSK, R=3/4	1	48	36
12	QPSK, R=1/2	2	96	48
18	QPSK, R=3/4	2	96	72
24	16-QAM, R=1/2	4	192	96
36	16-QAM, R=3/4	4	192	144
48	64-QAM, R=2/3	6	288	192
54	64-QAM, R=3/4	6	288	216

Source: 802.11 Wireless Networks: The Definitive Guide / by M. Gast / O'Reilly

Other Advanced Channel Coding

- Turbo codes: Berrou et al 1993
- Low-density parity check codes: Robert Gallager 1960
- Trellis-coded modulation: Ungerboeck 1982
- Space-time coding: Tarokh et al 1998
  - A family of codes that introduce correlation in both time and space (transmit antenna) dimensions
  - It is a combined result of channel coding, modulation and transmit antenna diversity.