Overview

- Last Lecture
  - Data Integrity 1
- This Lecture
  - Data Integrity 2
  - Source: Sections 10.2, 10.5
- Next Lecture
  - Data Security 1
  - Source: Sections 31.1, 31.2

Last Lecture: Parity Checking

Simple Parity Checking (Last Lecture)

- Simple parity checking: Add an extra bit (parity bit) to the data, such that the total number of 1 bits is even (even parity) or odd (odd parity)

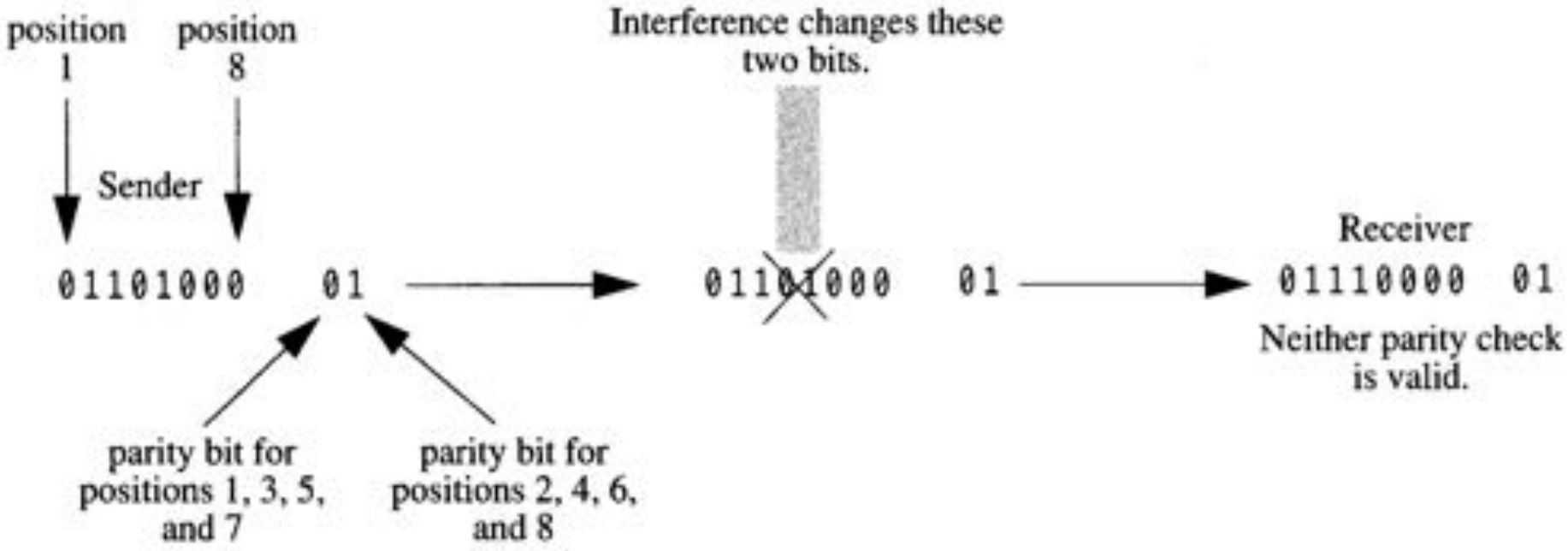
Even parity	Odd Parity
1101 0100 0	1101 0100 1

- Limitations of Simple Parity Checking
  - Cannot detect an even number of bit errors
  - Burst errors (Many bits are damaged)
  - Does this make it useless?

Parity Checking (Double Bit Error)

- Double Bit Error Detection: Use two parity bits (One for even numbered bits, One for odd numbered bits). If one bit errors or some double bit errors occur, they are detected

Figure 4.2 Detecting Consecutive Double-Bit Errors Using Parity Checking

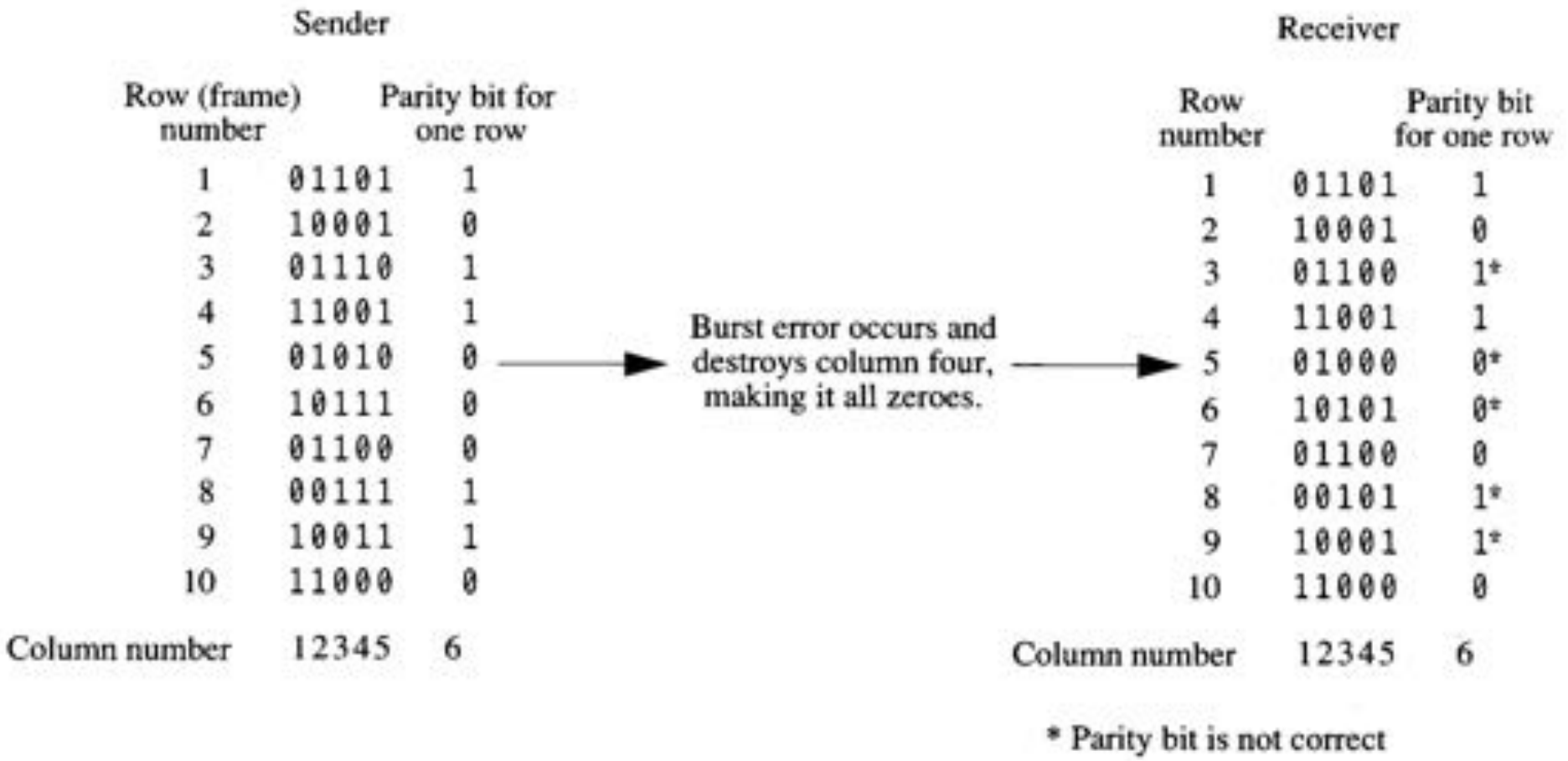


Parity Checking (Multiple Bit Errors)

- What happens if two even numbered (or odd numbered) bits are in error?
- Multiple bit errors: A two-dimensional bit array
  - Rows - byte or several bytes
  - Parity bit for each row
  - Transmit the bits by column
- Can detect any single burst error whose duration is less than the time to send a column.
- Parity Checking forms a basis for an error correction scheme

Parity Checking (Multiple Bit Errors, cont.)

Figure 4.3 Detecting Burst Errors Using Parity Bits



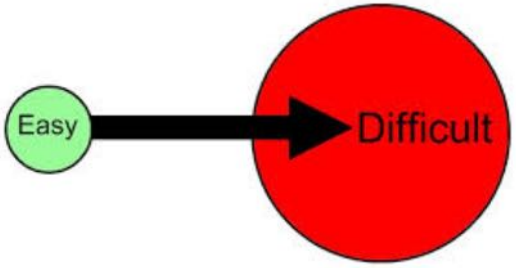
Error Correction

- Can be handled two ways:
  - Ask the sender to retransmit the data
  - Correct the errors using an error correction code
- Hamming codes
  - Even parity is used in the examples
  - Assume we are sending bytes
  - Assume only one error
  - Can be extended to multiple bit errors

Hamming code for data correction is one of the most difficult parts in 244 lectures

But,

You are so clever to make it easy

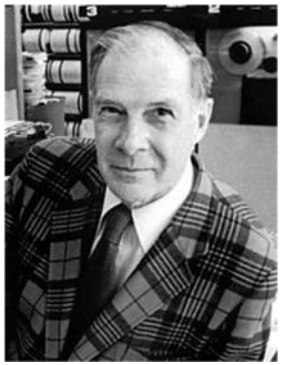


Binary Card game: http://gwydir.demon.co.uk/jo/numbers/binary/cards.htm

Example

- We want to send 0110 0111
- Suppose we receive 0101 0110 0111. Is there an error? If so, what is the correct bit pattern?
- Using the Hamming code, we can not only detect single bit error, but also correct them!

Hamming code

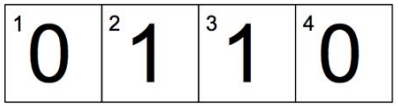


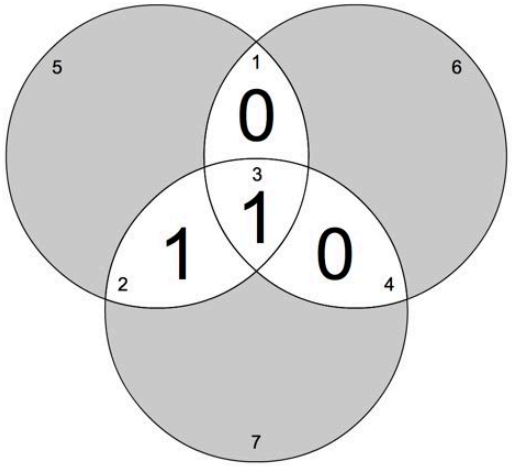
- Invented by Richard Hamming in 1950
- can detect and correct on bit error

- What we will learn for Hamming code in this lecture are:
- construct Hamming code
  (e.g. data: 01100111, hamming code: 010111010111)
- detect and correct one-bit error (e.g. data received: 010101010111)

Hamming Code

- Basic idea:
  - Use multiple extra parity bits
  - Combine bits in a smart way that each extra bit checks whether there is an error in a subset of data bits



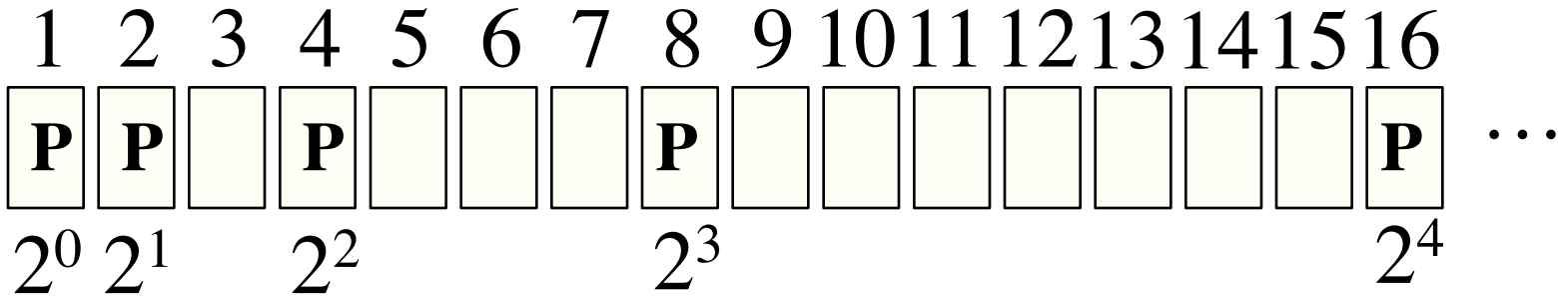


Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 1: determine the positions of parity bits

  The bits whose position number is a power of 2 (1, 2, 4, 8, ..) are reserved for parity bits.



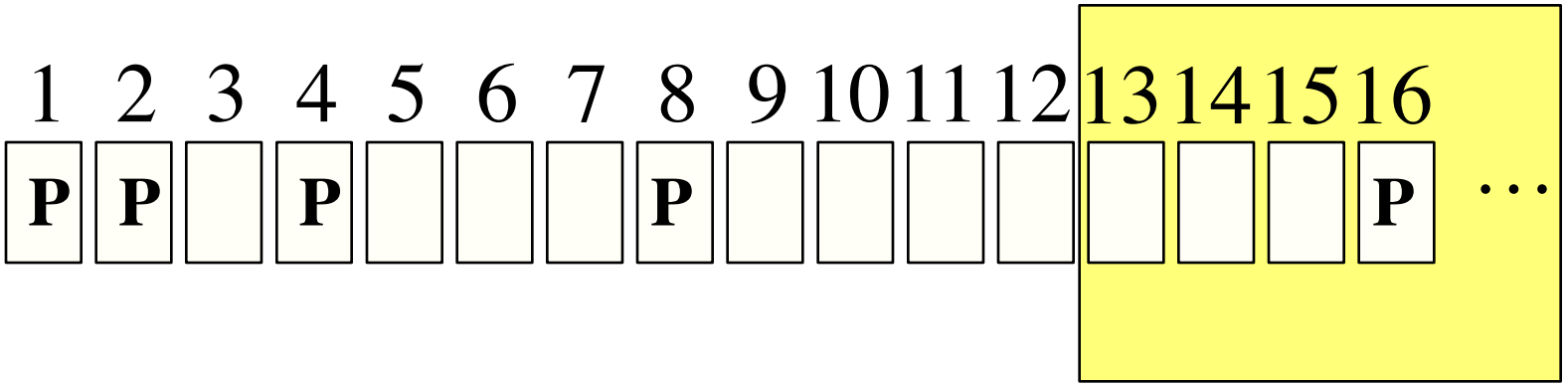
Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 2: insert data bits

Insert data bits into unoccupied positions from left to right

Data to send: **0 1 1 0 0 1 1 1**

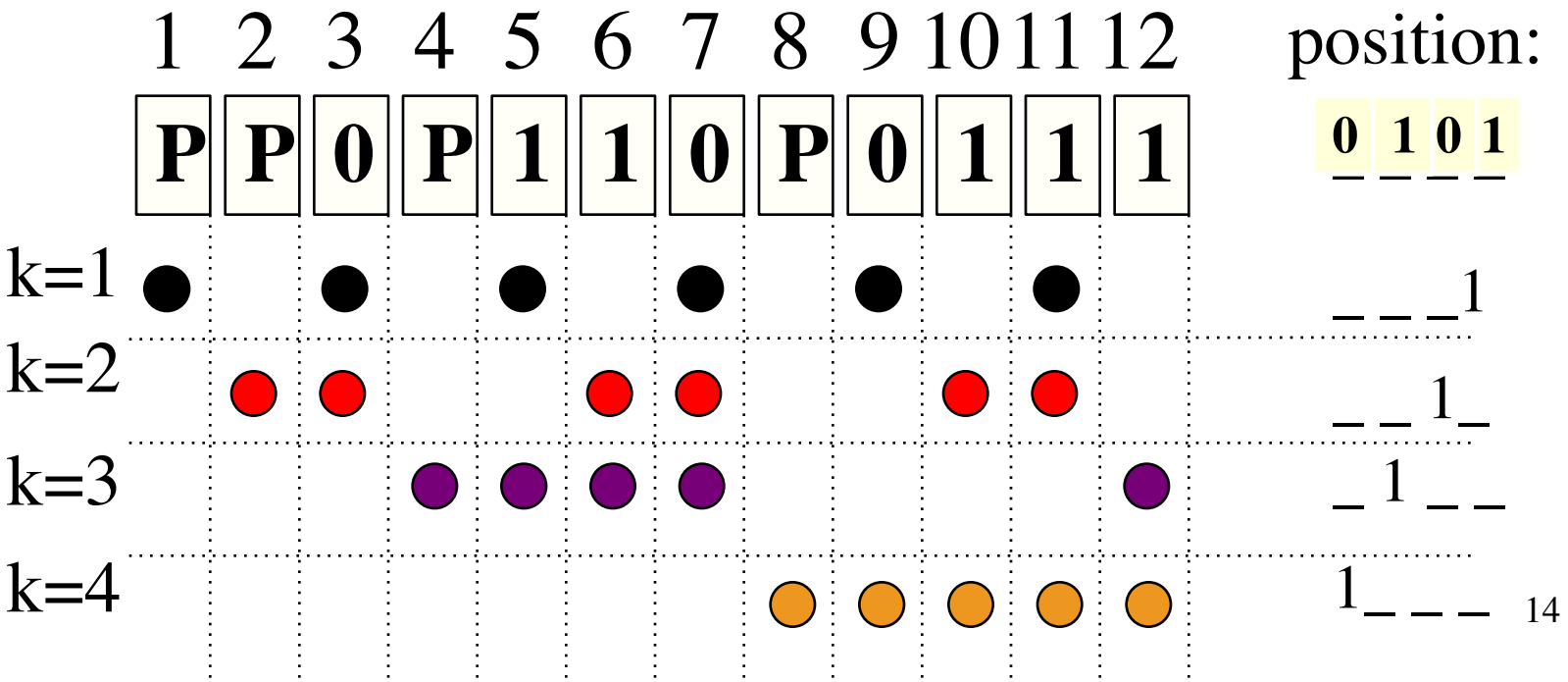


Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 3: Group the bits into different subsets with each subset containing one parity bit

Start from the 2^{k-1} bit, choose 2^{k-1} bits, skip 2^{k-1} bits, choose 2^{k-1} bits …

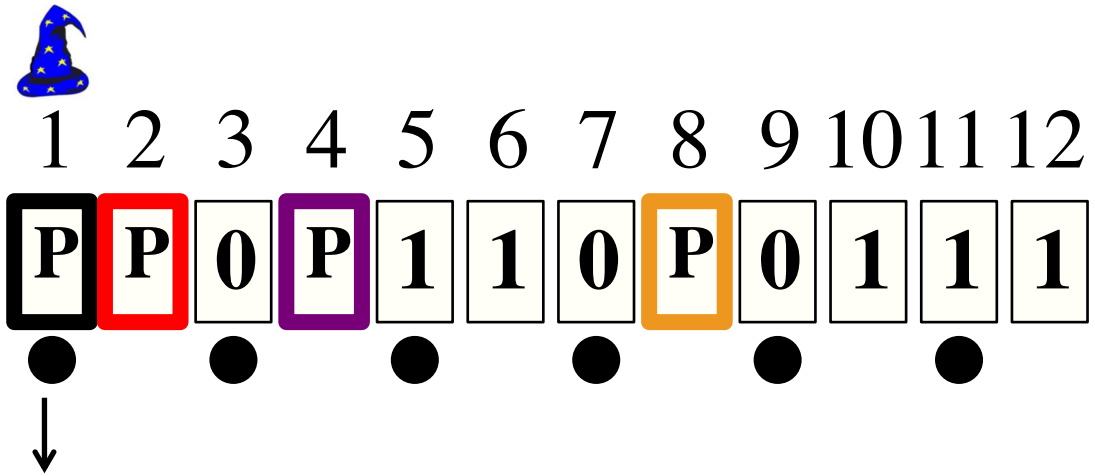


Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



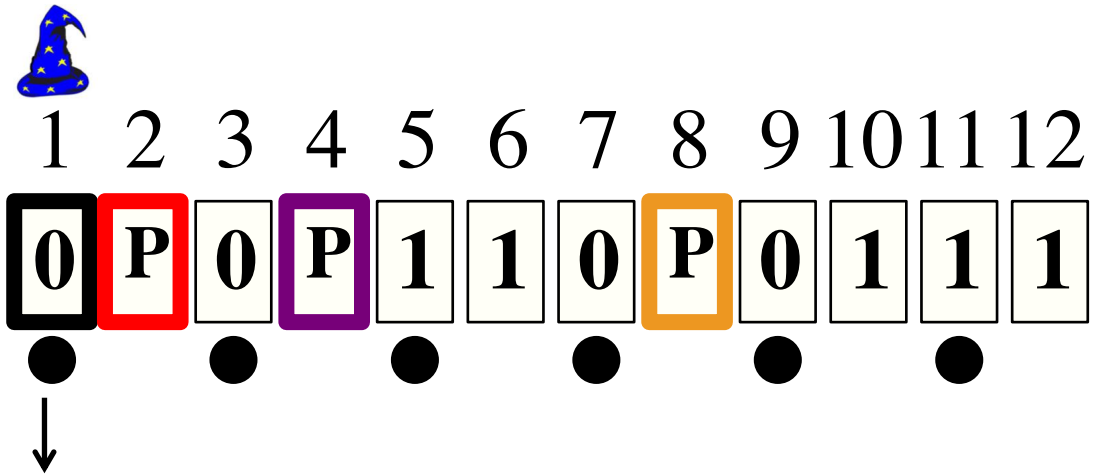
Even parity check for positions 1, 3, 5, 7, 9 _ _ _ 1

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



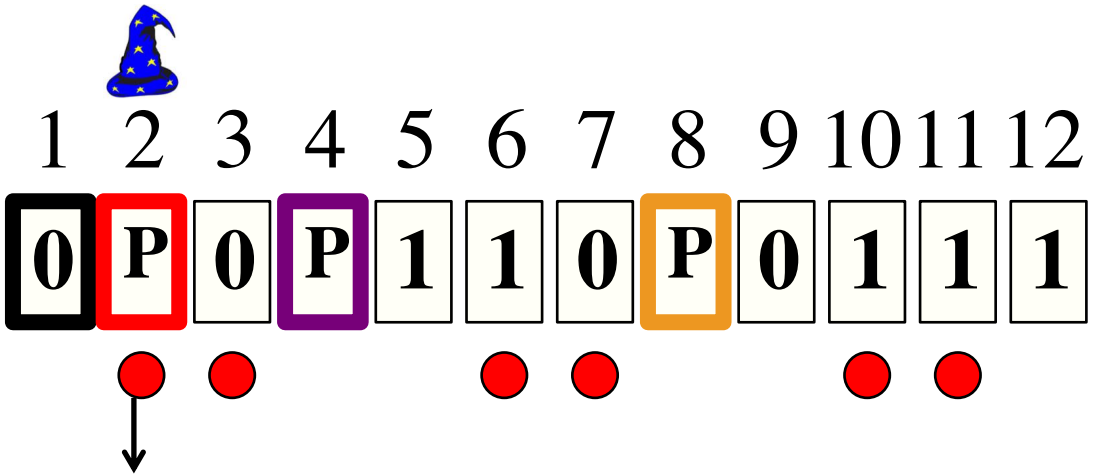
Even parity check for positions 1, 3, 5, 7, 9 _ _ _ _1

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



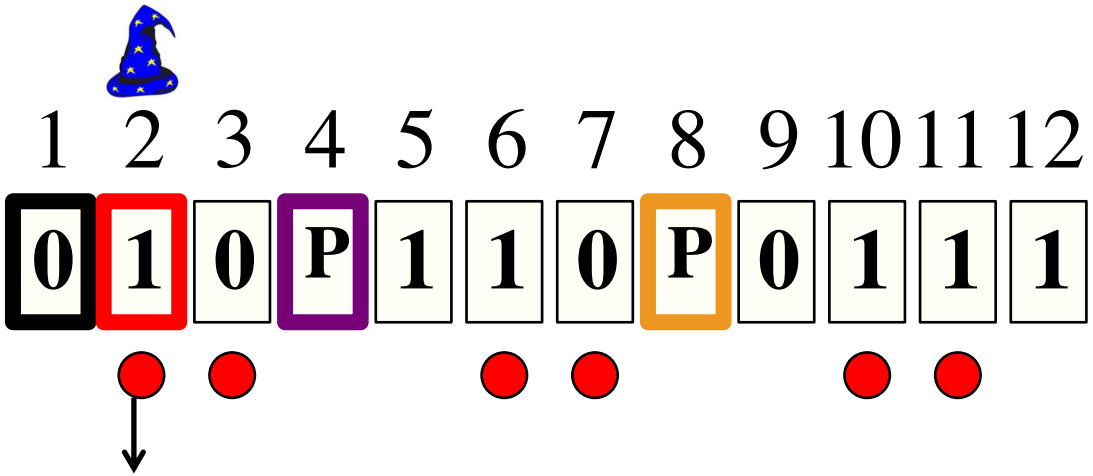
Even parity check for positions 2, 3, 6, 7, 10, 11 _ _ 1_

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



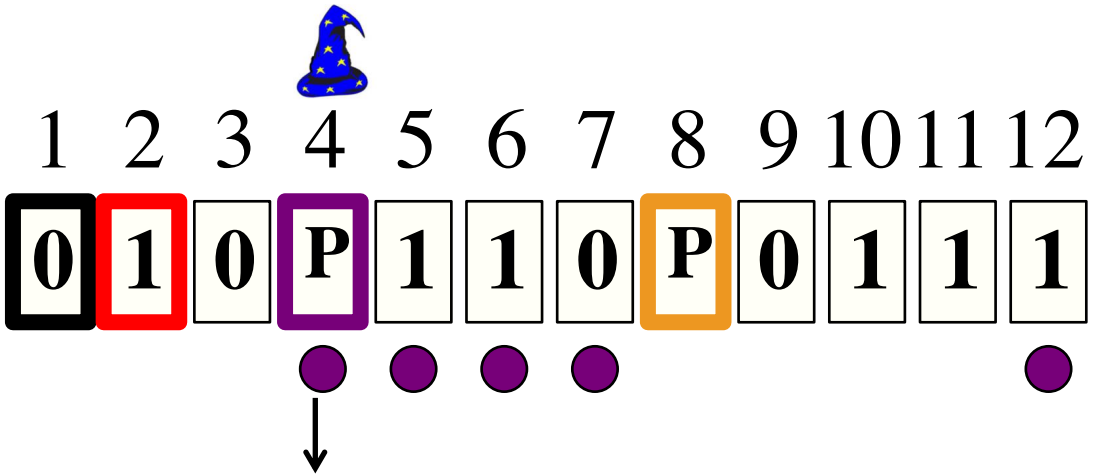
Even parity check for positions 2, 3, 6, 7, 10, 11 _ _ 1_

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



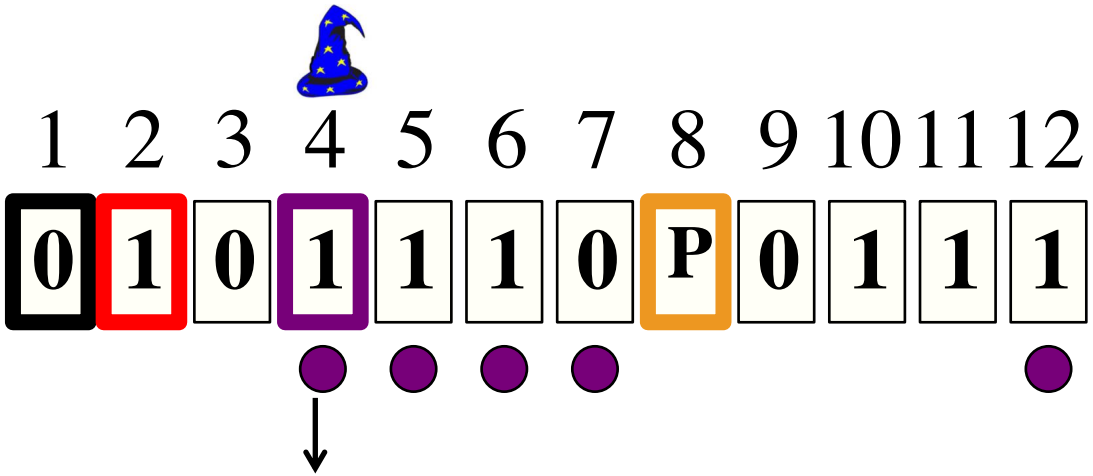
Even parity check for positions 4, 5, 6, 7, 12 _ 1 _ _

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check

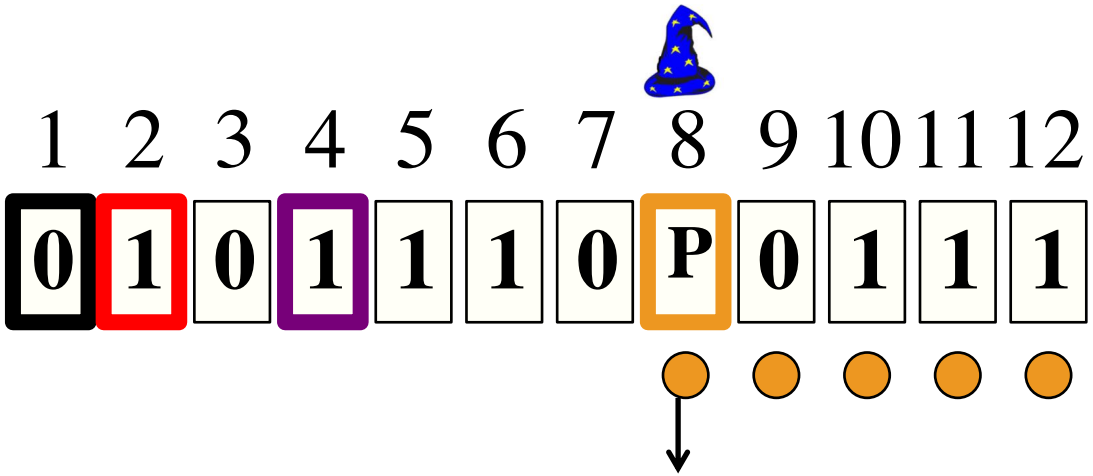


Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



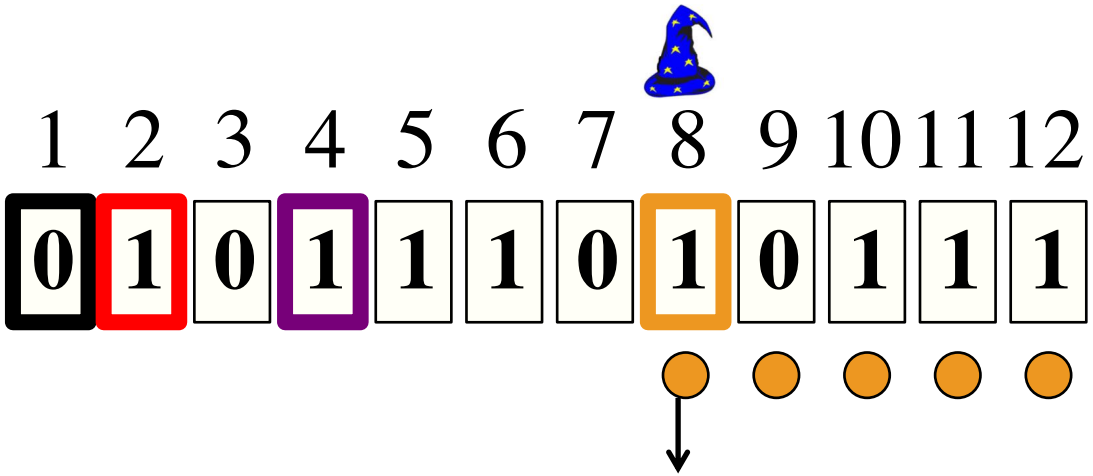
Even parity check for positions 8, 9, 10, 11, 12 1_ _ _ _

Hamming Code Construction

- A four-step approach to construct Hamming code

  Step 4: calculate the values for the parity bits

Within each group, apply single parity check



Even parity check for positions 8, 9, 10, 11, 12 1_ _ _

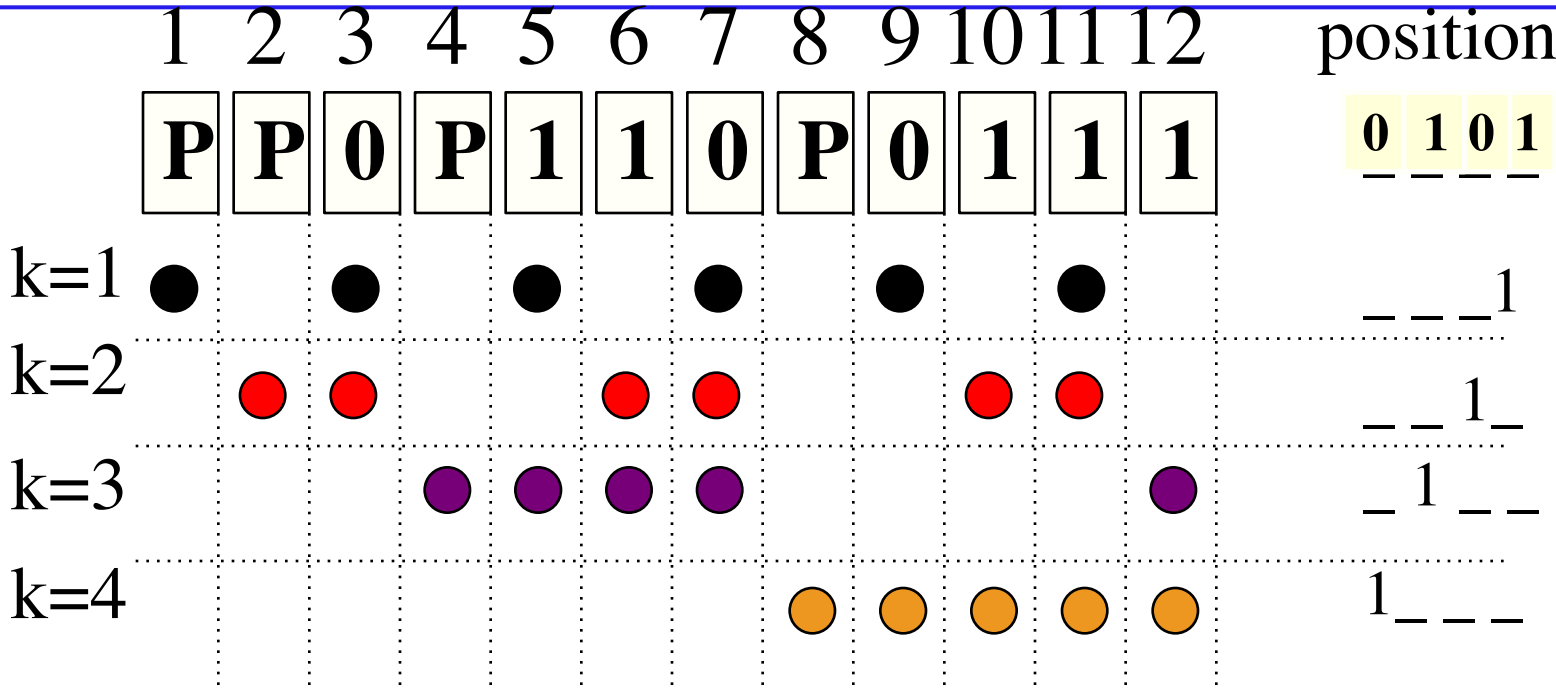
Hamming Code Construction

Data to send: **0 1 1 0 0 1 1 1**

Hamming code: **0 1 0 1 1 1 0 1 0 1 1 1**

We got the Hamming code!

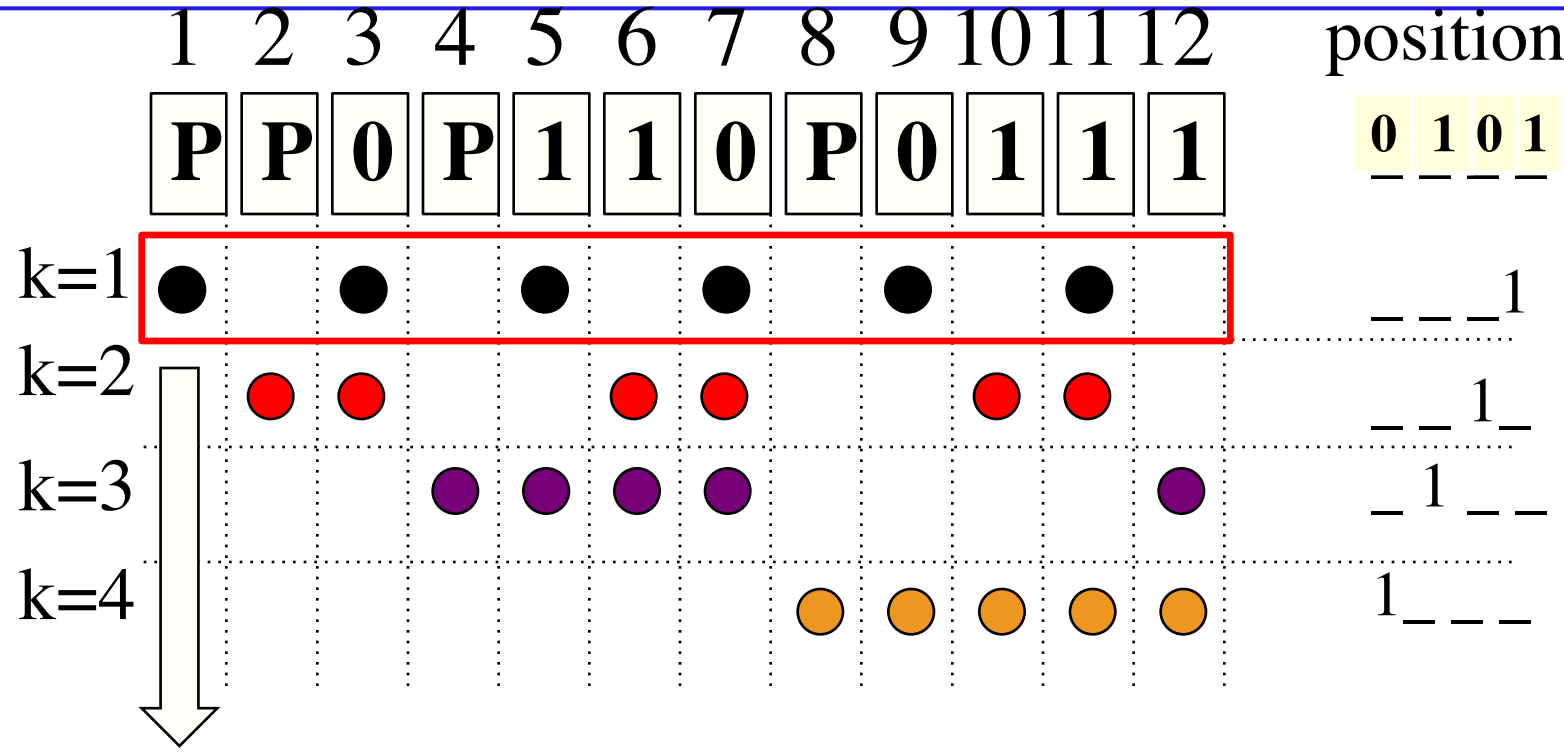
Hamming Code Construction



Why should Hamming code be constructed in such a way?

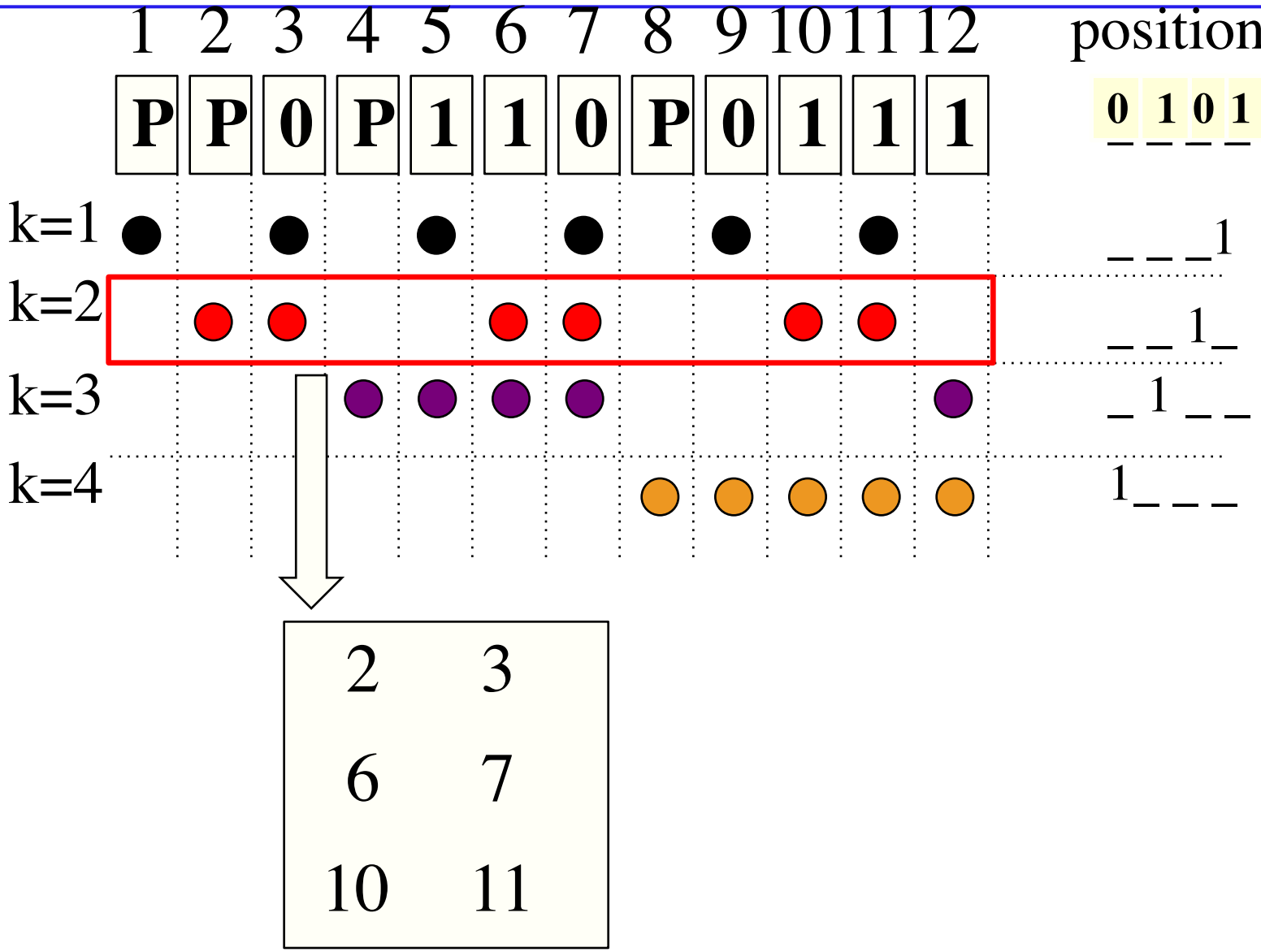
magic trick (guess your birthday month)

Hamming Code Construction

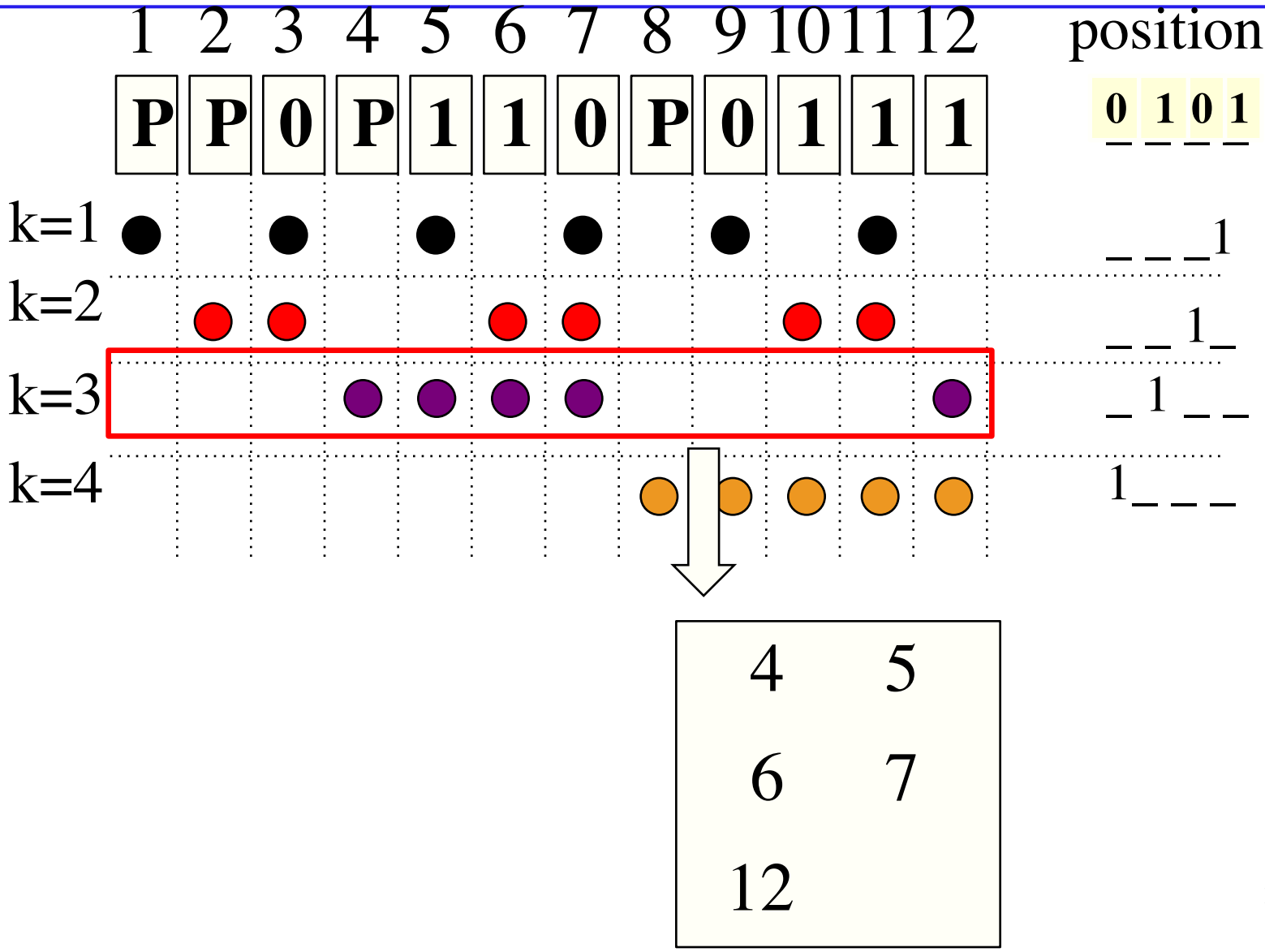


1	3
5	7
9	11

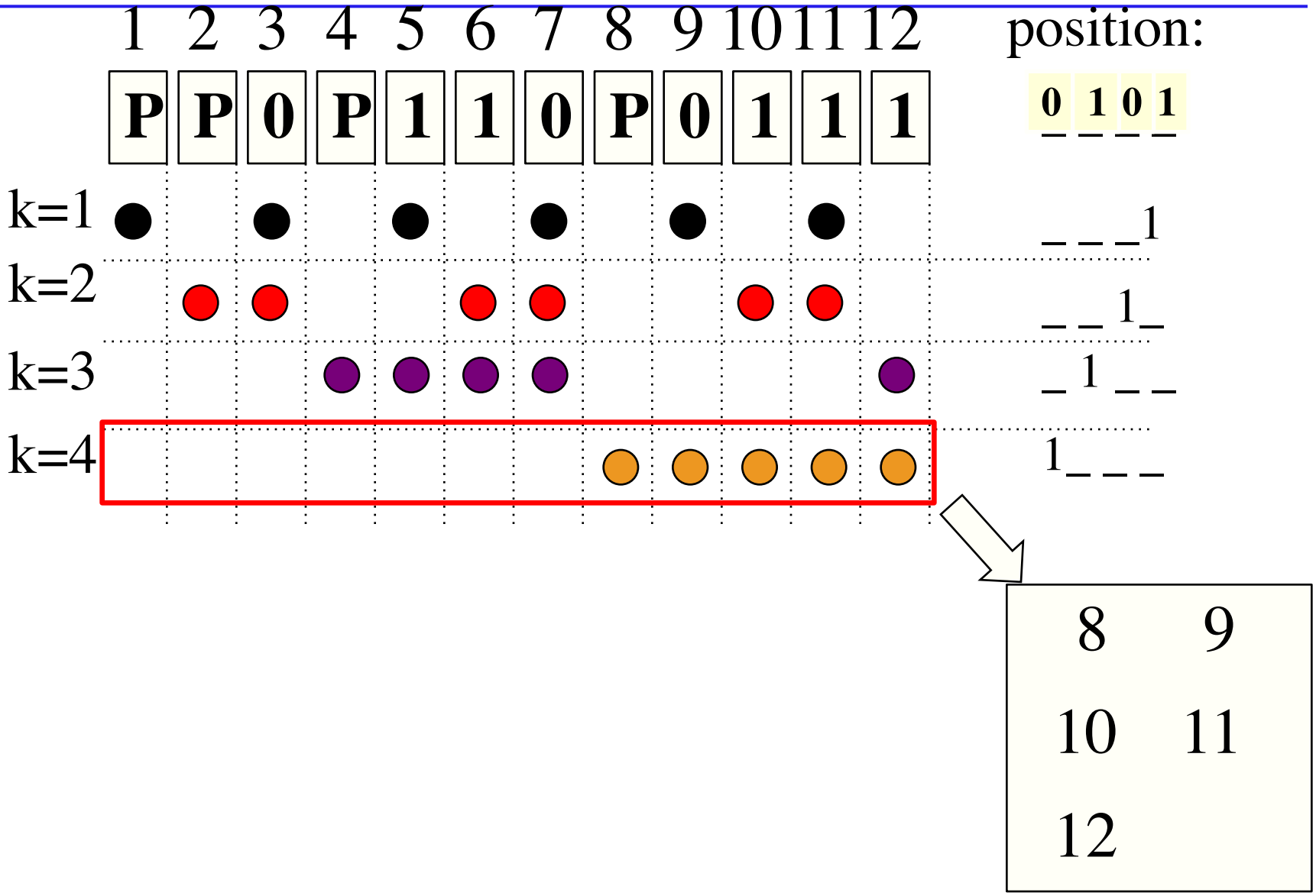
Hamming Code Construction



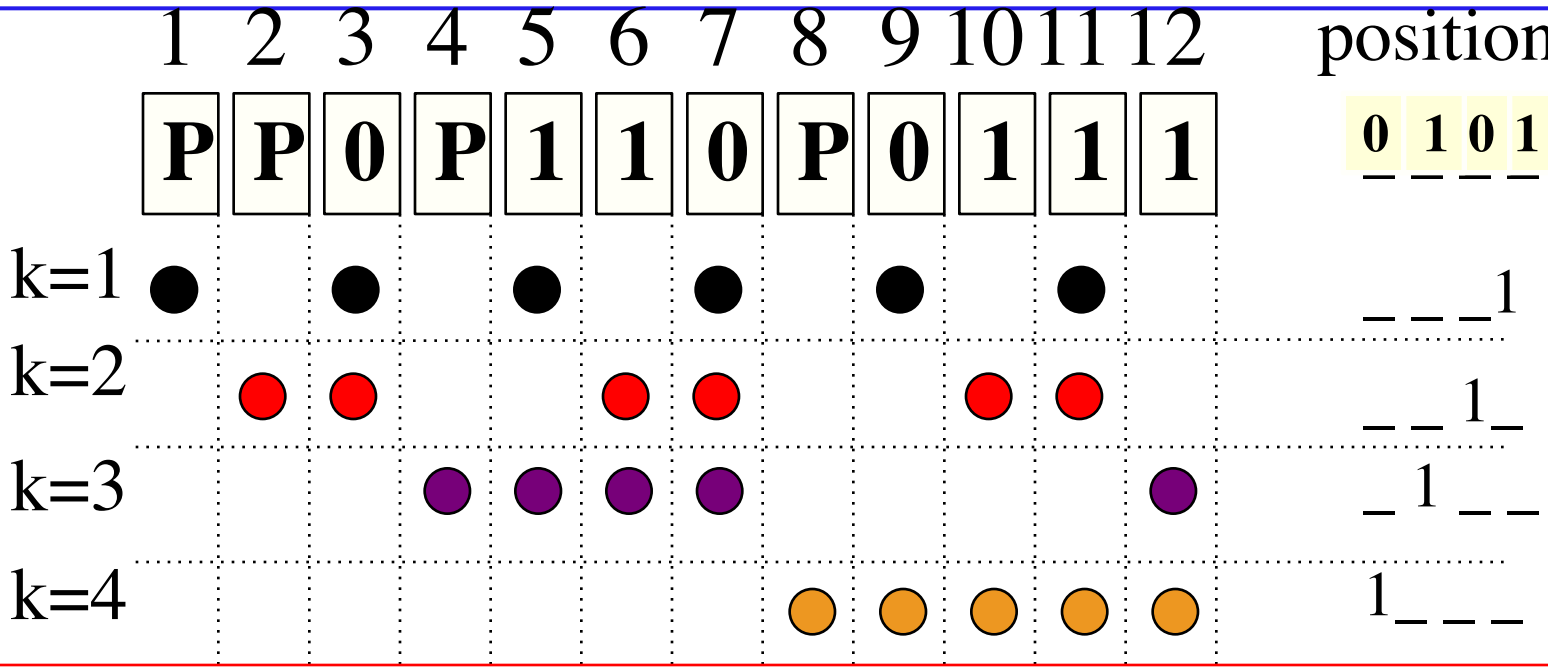
Hamming Code Construction



Hamming Code Construction



Hamming Code Construction



guess your birthday month

1	3
5	7
9	11

2	3
6	7
10	11

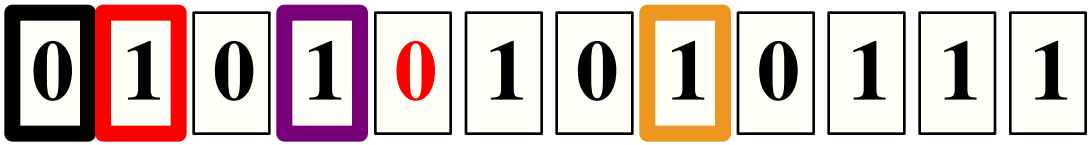
4	5
6	7
12	

8	9
10	11
12	

Hamming Code Error Correction

The receiver received the hamming code

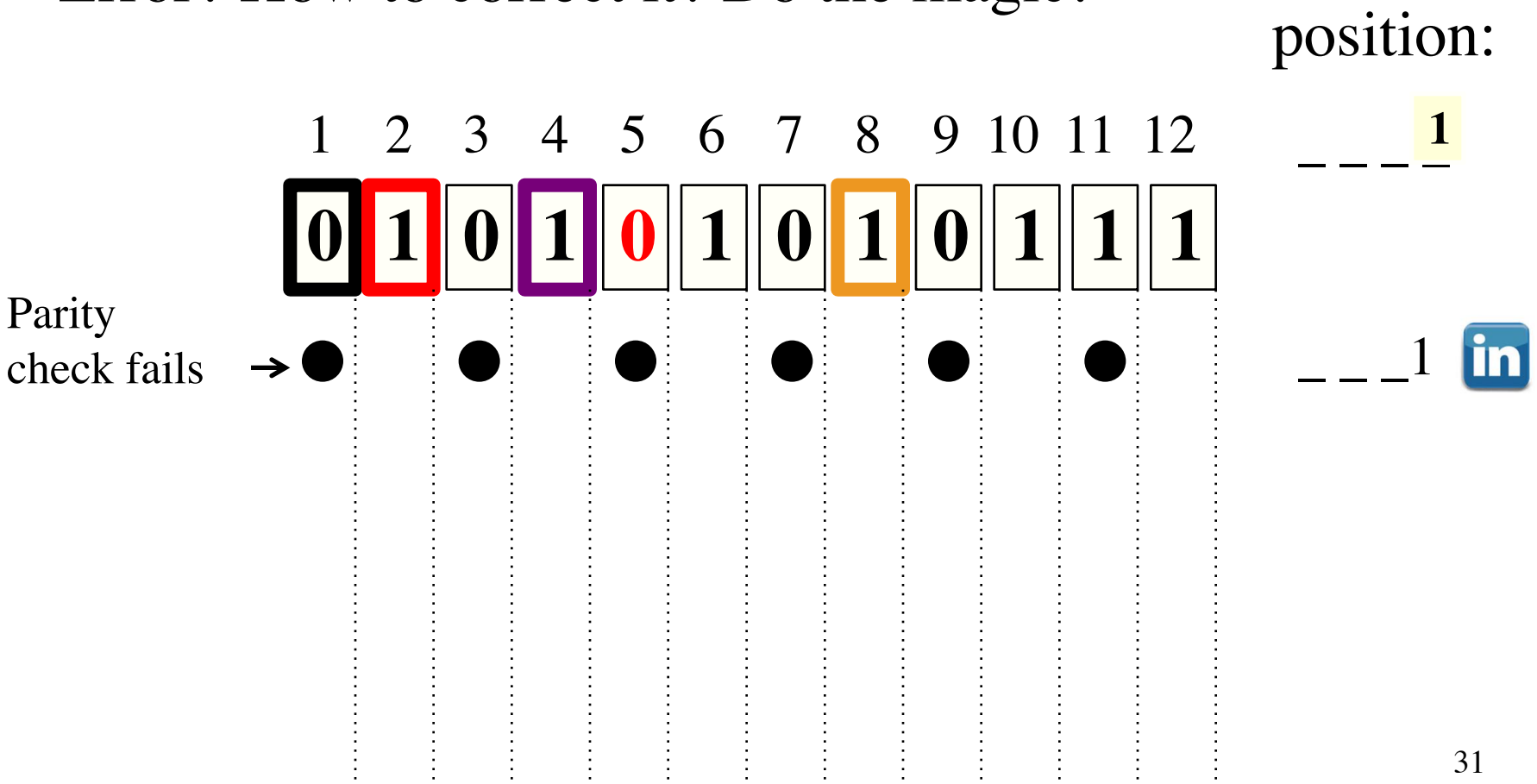
Error? How to correct it? Do the magic!



Hamming Code Error Correction

The receiver received the hamming code

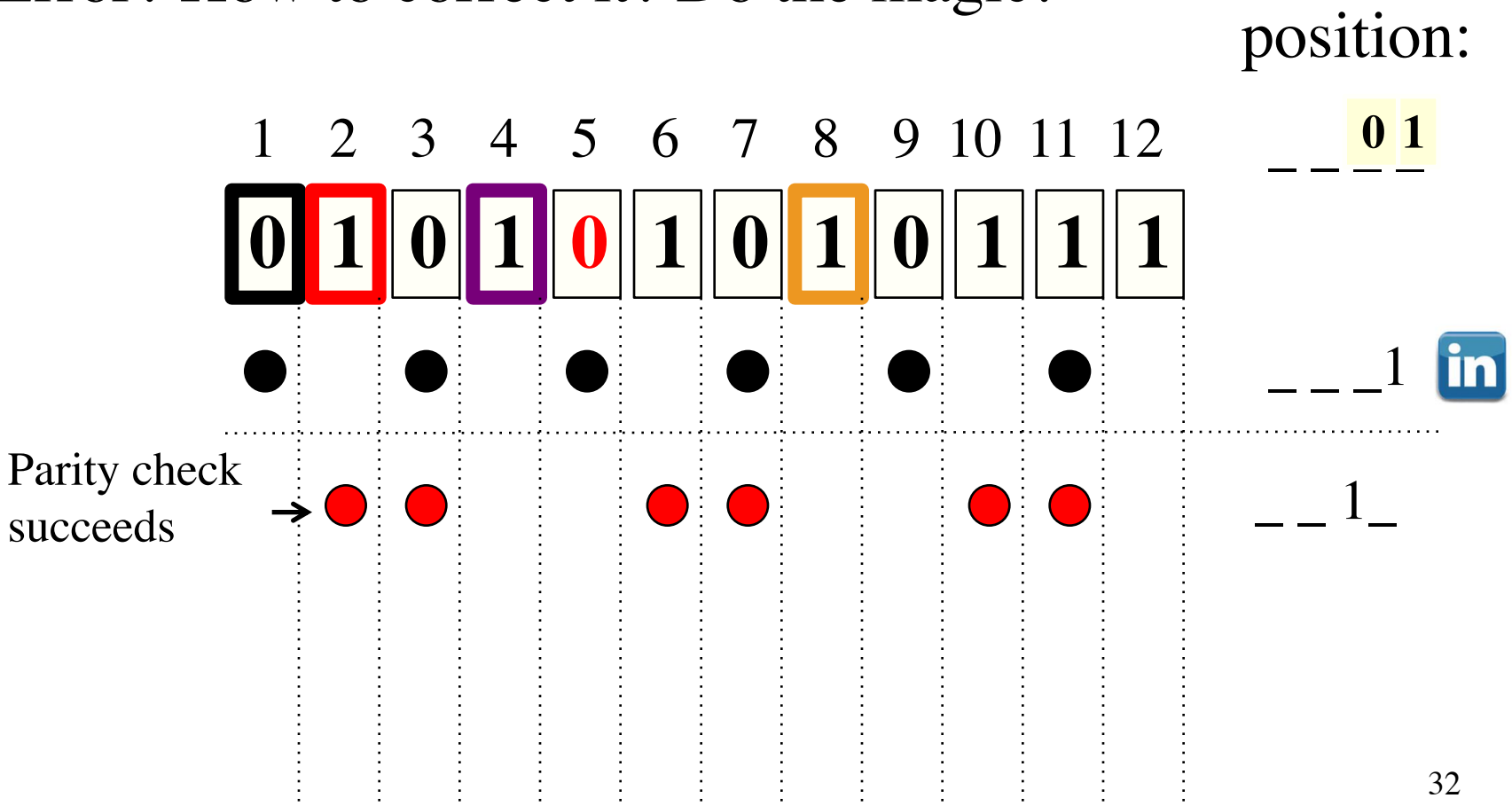
Error? How to correct it? Do the magic!



Hamming Code Error Correction

The receiver received the hamming code

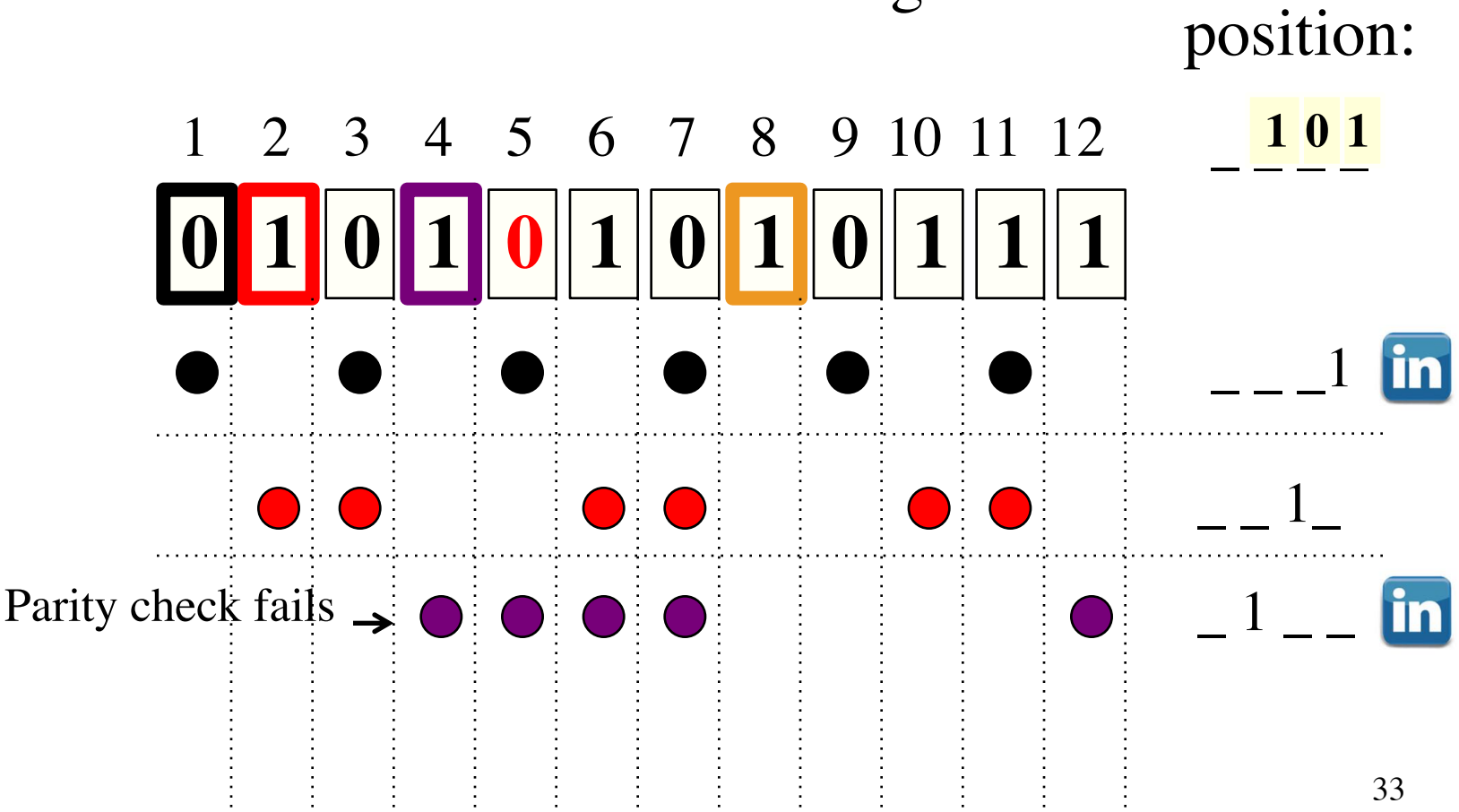
Error? How to correct it? Do the magic!



Hamming Code Error Correction

The receiver received the hamming code

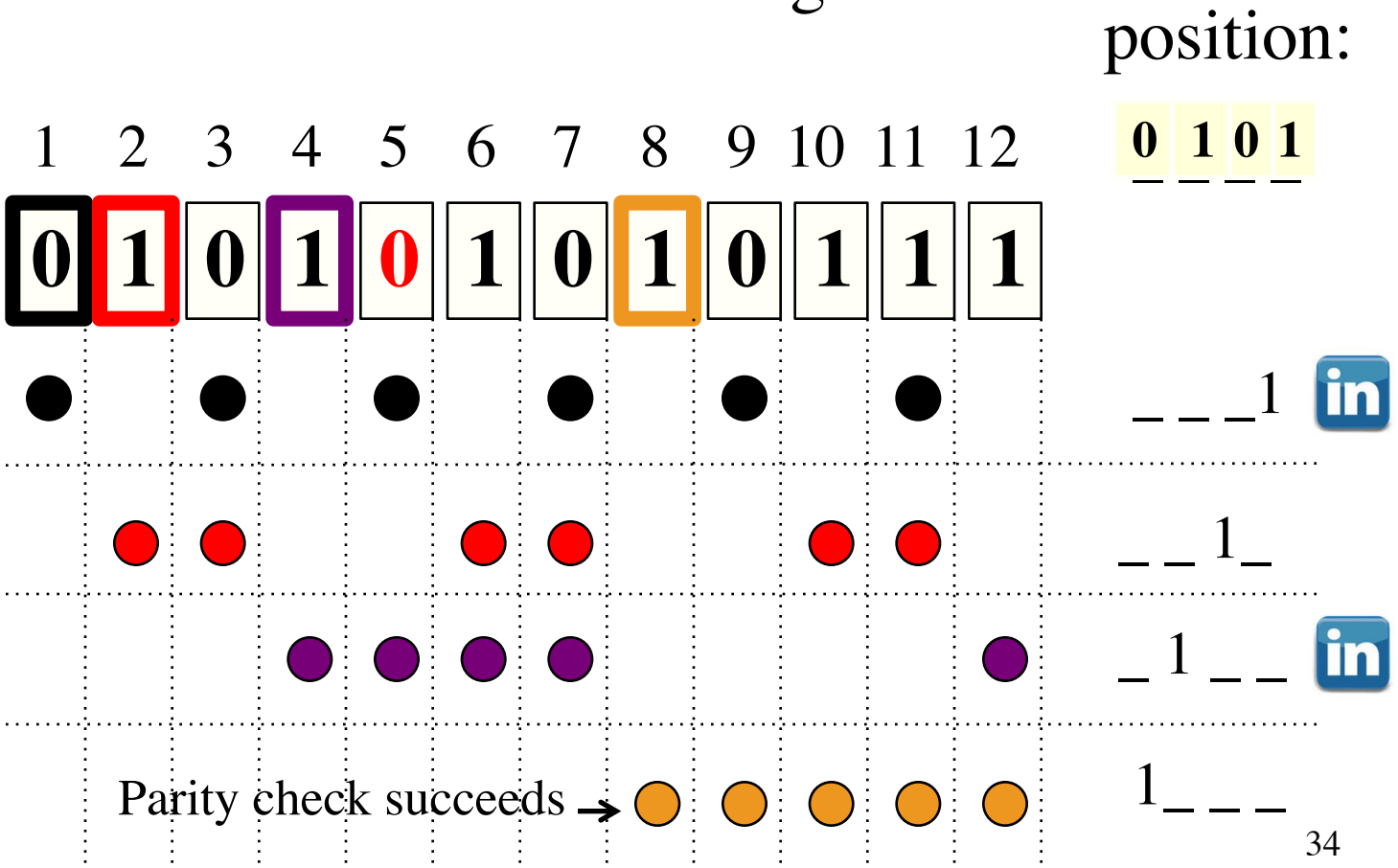
Error? How to correct it? Do the magic!



Hamming Code Error Correction

The receiver received the hamming code

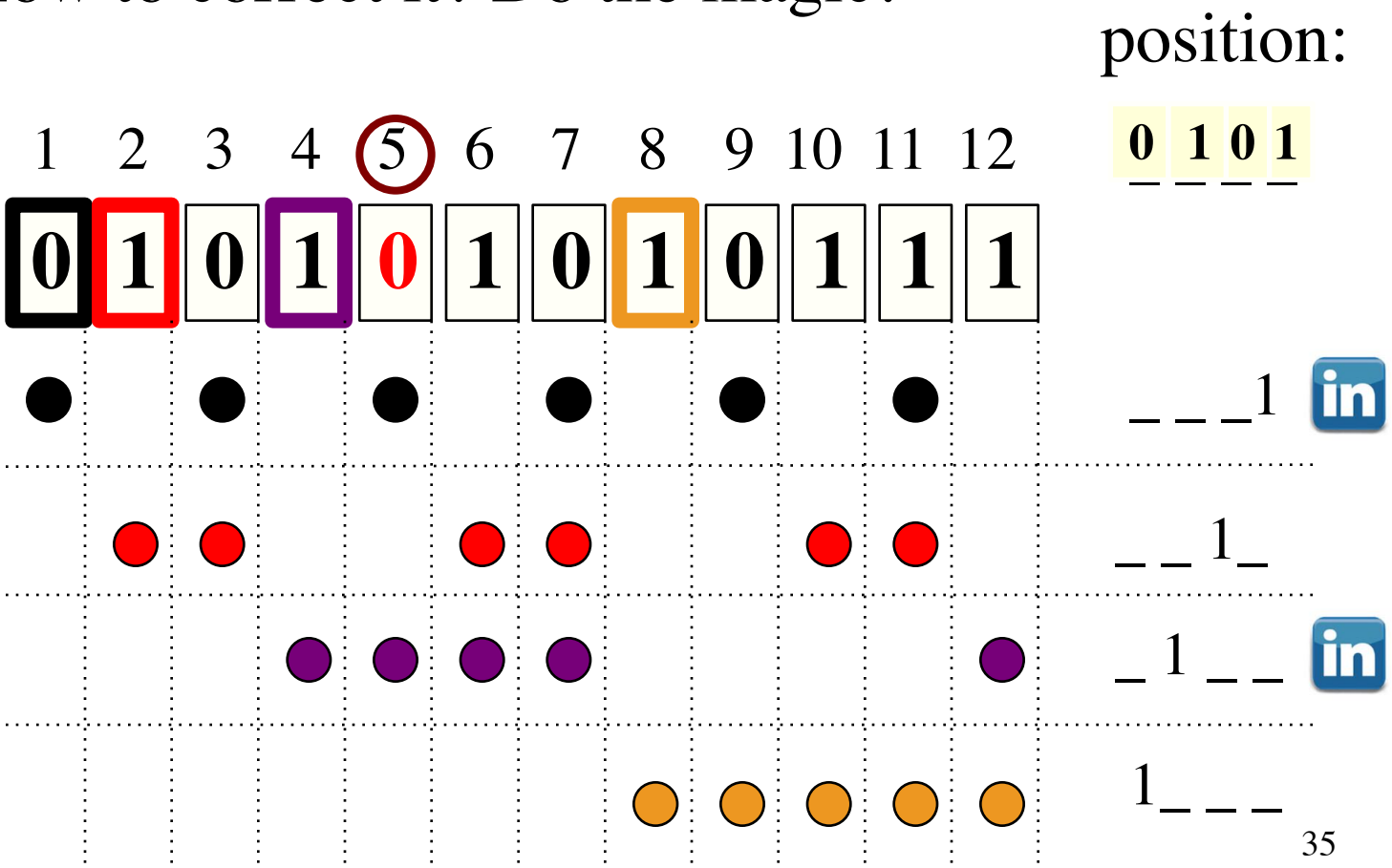
Error? How to correct it? Do the magic!



Hamming Code Error Correction

The receiver received the hamming code

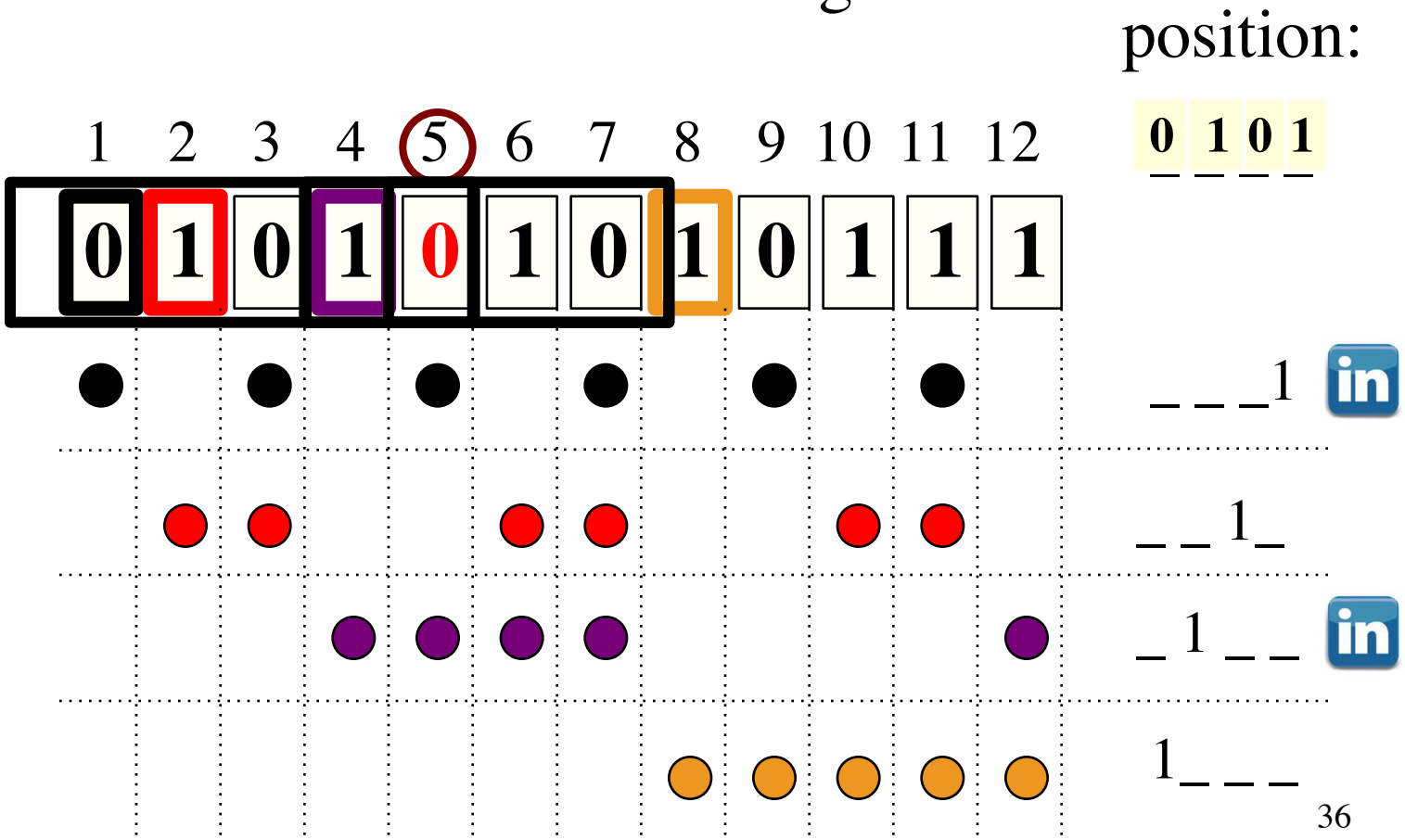
Error? How to correct it? Do the magic!



Hamming Code Error Correction

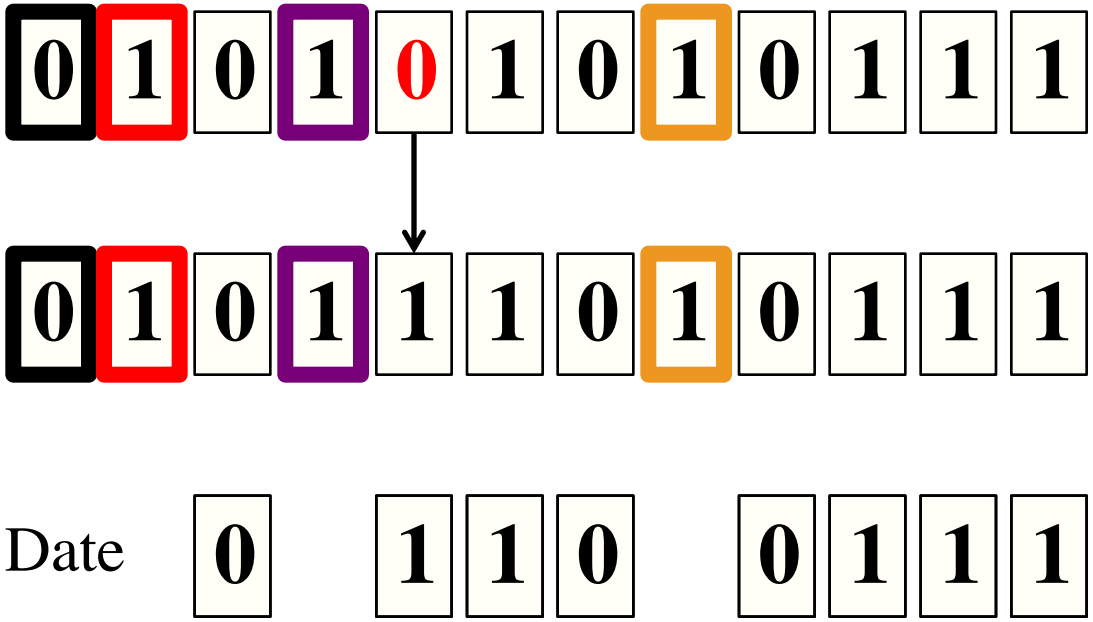
The receiver received the hamming code

Error? How to correct it? Do the magic!



Hamming Code Error Correction

One error can be automatically corrected without the need of retransmission!



If you have n parity bits, it can identify 2^n-1 bit positions

Hamming Code (summary)

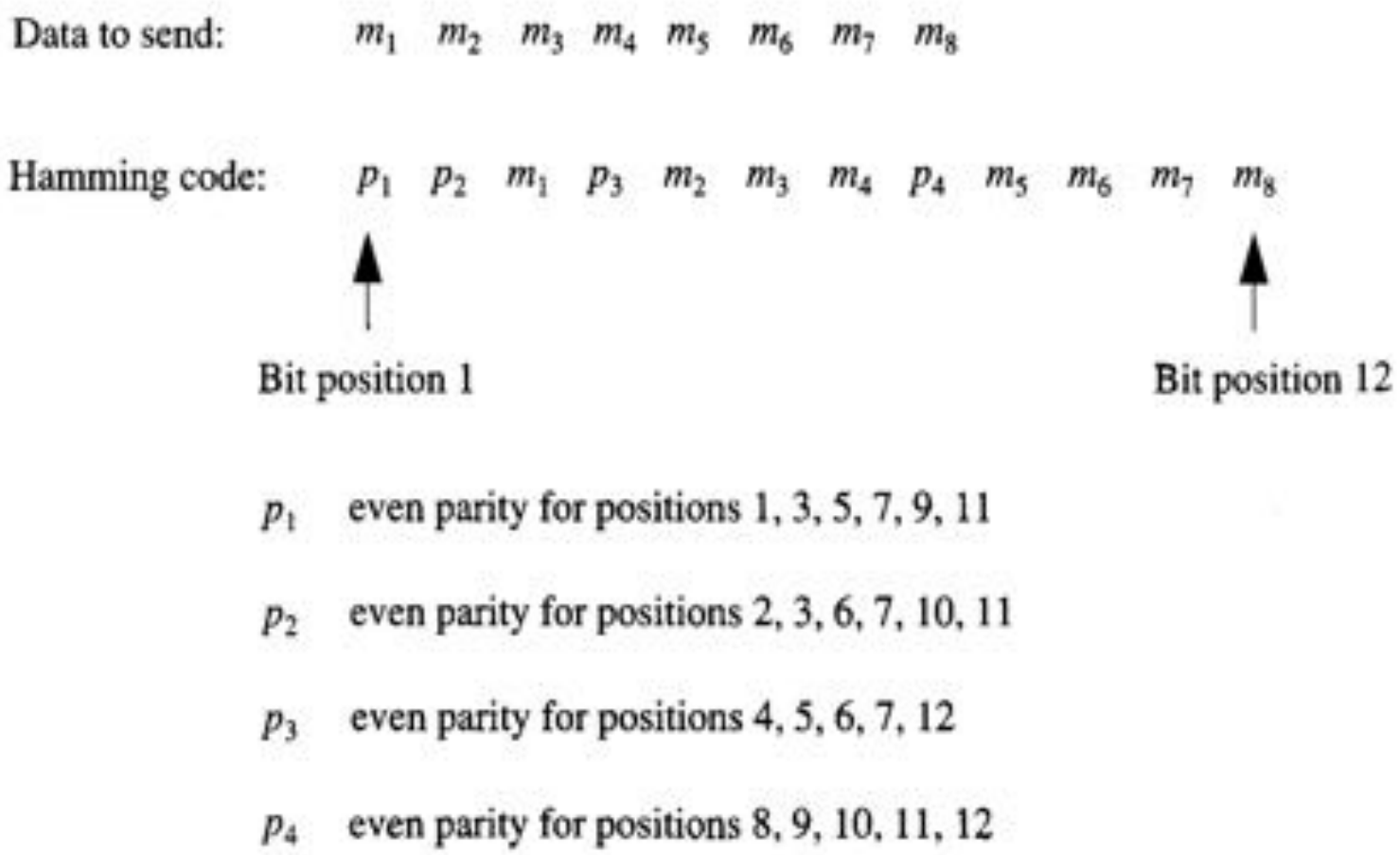


Figure 4.9 Hamming Code for Single-Bit Error Correction

Hamming Code (summary)

Figure 4.10 Bit Stream Before Transmission

Data:	0	1	1	0	0	1	1	1				
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8				
Hamming code:	0	1	0	1	1	1	0	1	0	1	1	1
	p_1	p_2	m_1	p_3	m_2	m_3	m_4	p_4	m_5	m_6	m_7	m_8

Hamming Codes (summary)

- Consider a 4 bit number b_4, b_3, b_2, b_1

$b_i = 0$ if parity check for p_i succeeds, otherwise $b_i = 1$

Table 4.2 Bit Position Errors and Associated Parity Errors

Erroneous Bit Position	Invalid Parity Checks	b_4, b_3, b_2, and b_1
No error	None	0000
1	p_1	0001
2	p_2	0010
3	p_1 and p_2	0011
4	p_3	0100
5	p_1 and p_3	0101
6	p_2 and p_3	0110
7	p_1, p_2, and p_3	0111
8	p_4	1000
9	p_1 and p_4	1001
10	p_2 and p_4	1010
11	p_1, p_2, and p_4	1011
12	p_3 and p_4	1100

Hamming Code (summary)

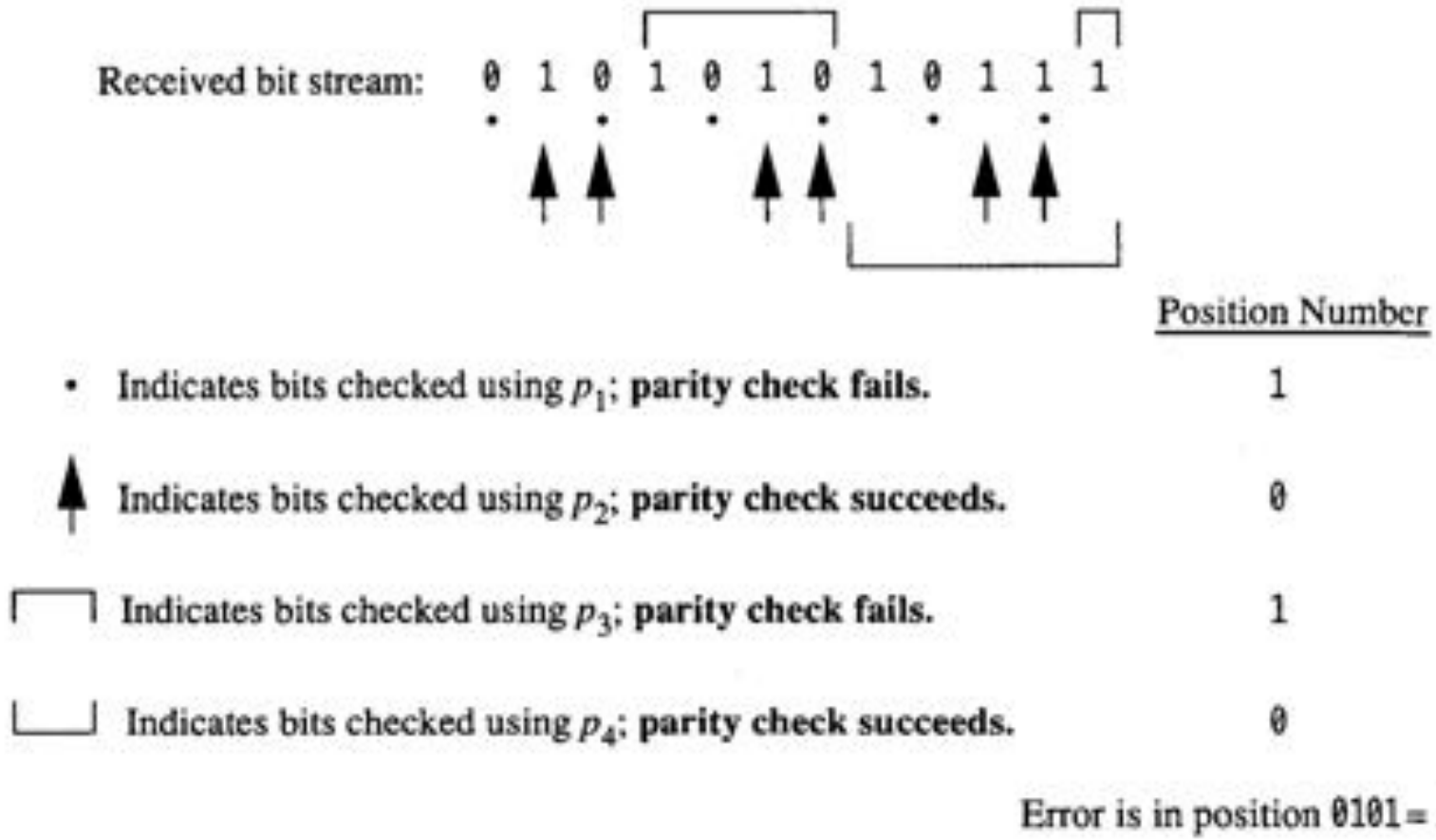


Figure 4.11 Parity Checks of Frame After Transmission

Multiple Bit Error Correction

- Use the idea similar to multiple bit error detection.
- Create the Hamming codes for each byte.
- Arrange the Hamming codes into an array.
- Send the array a column at a time.
- As long as no burst is longer than a column, correction can be done.

Error Correction vs. Error Detection

- Error detection
  - Error detection is the ability to determine that data has an error
  - Simpler, Cheaper, Requires retransmission
- Error correction
  - Error correction is the ability to correct the erroneous data.
  - No retransmission, Complex mechanism, Sometimes required

Summary (this lecture)

- Hamming code
  - Code construction
  - Error correction
- What should you learn from this lecture:
  - construct the Hamming code
    (e.g. data: 01100111, hamming code: 010111010111)
  - detect and correct one-bit error
    (e.g. data received: 010101010111)

Summary (Data integrity)

- Error detection schemes
  - Parity checking (single bit, double bit, burst)
  - CRC (burst)
- Error correction schemes
  - Hamming codes (single bit)
- Comparison of error detection and correction

Summary of L1-L6

- Written test (7%) in Lab
- check the date and sample test at: http://www.cs.otago.ac.nz/cosc244/assessment.php
- What you should learn from L1-L6
  - L1: Basic components of data communication
  - L2: Digital Transmission (NRZ/Manchester/ Differential Manchester) and Analogy Transmission (ASK/FSK/PSK)
  - L3: Transmission: modes/directions/Multiplexing
  - L4: Compression Methods: Huffman/Run length encoding/LZW
  - L5: Error Detection Methods: Parity check/CRC
  - L6: Error Correction Method: Hamming code