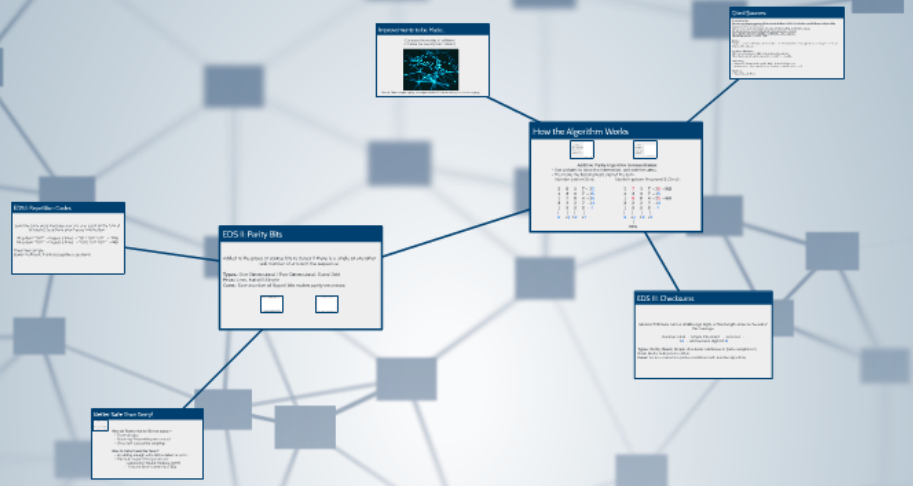
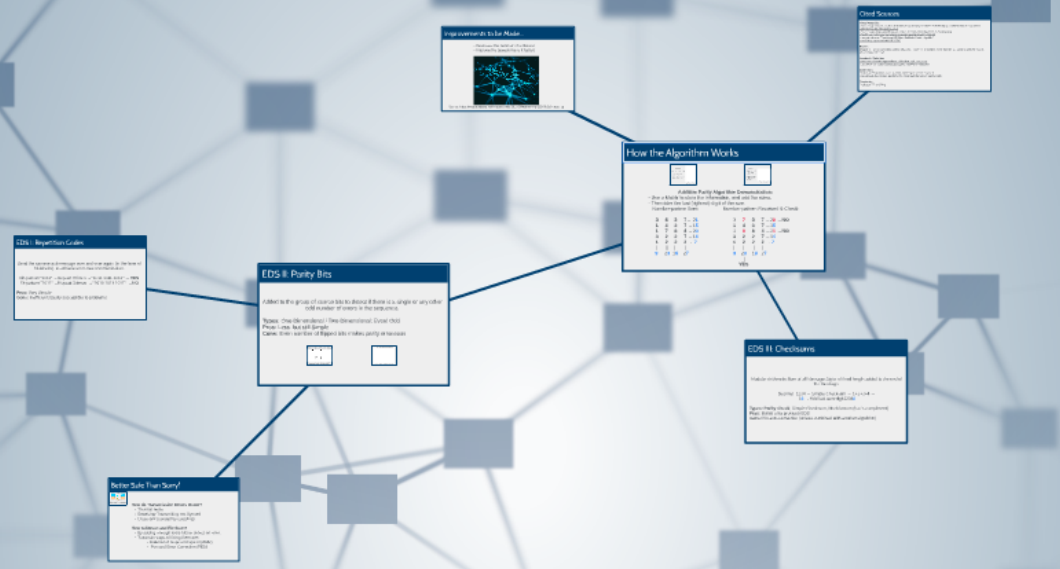


Reliable Messaging

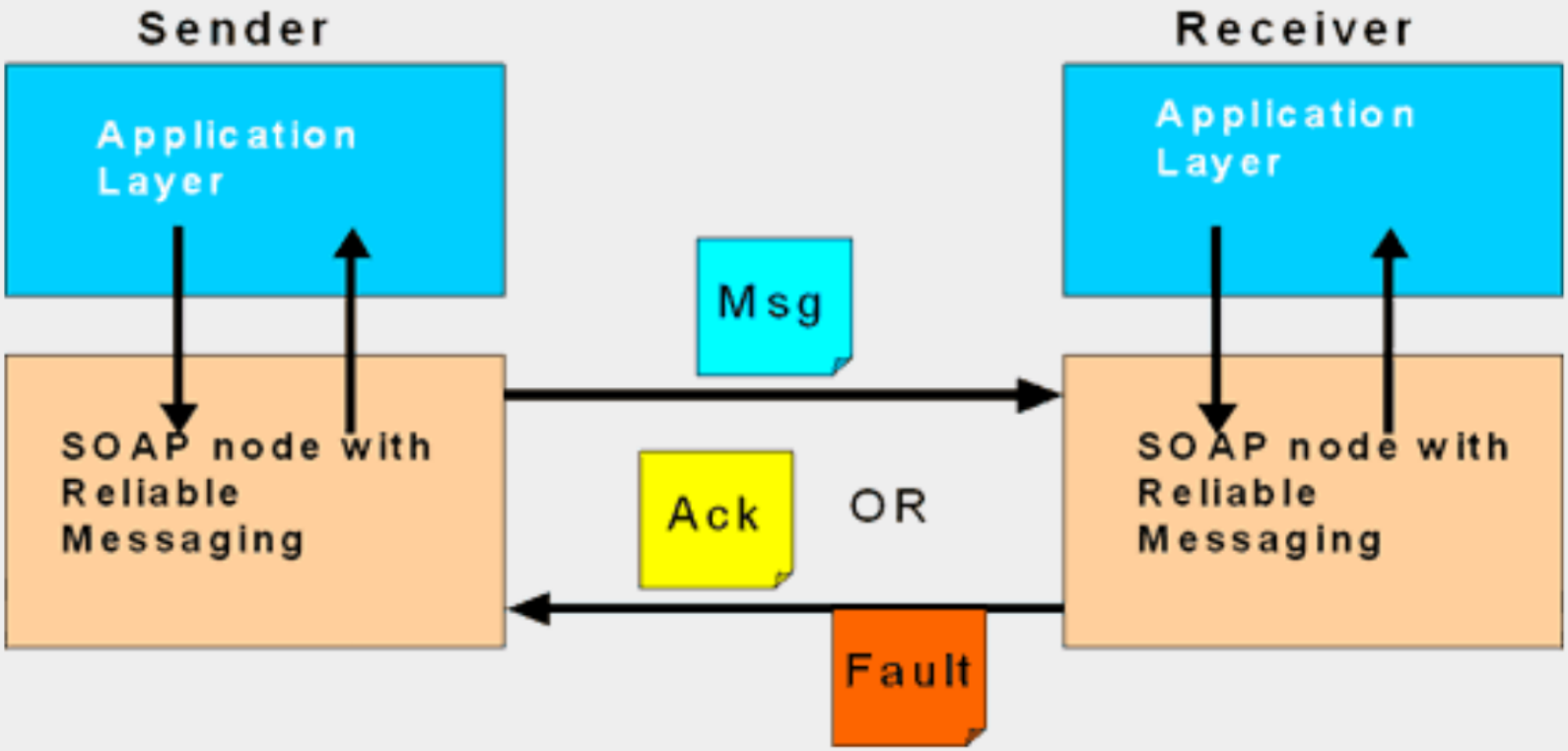
By Tisa (te557) & Ruofan (rz829)



Reliable Messaging

By Tisa (te557) & Ruofan (rz829)

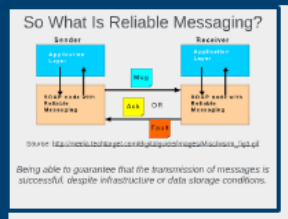
So What Is Reliable Messaging?



Source: http://media.techtarget.com/digitalguide/images/Misc/wsrm_fig3.gif

Being able to guarantee that the transmission of messages is successful, despite infrastructure or data storage conditions.

Better Safe Than Sorry!



How do Transmission Errors Occur?

- Thermal noise
- Receiving-Transmitting not Synced
- Cross-talk (caused by coupling)

How to Detect and Fix them?

- By adding enough *extra* bits to detect an error.
- Two main ways of fixing them are:
    - Automated Repeat Request (ARR)
    - Forward Error Correction (FED)

EDS I: Repetition Codes

Send the same exact message over and over again (in the form of bit-blocks), to achieve error-free communication.

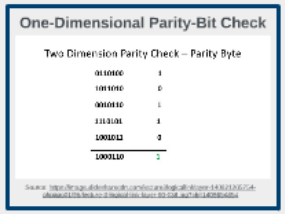
Bit-pattern "1011" → Repeat 3 times → "1011 1011 1011" → **YES**
Bit-pattern "1011" → Repeat 3 times → "1010 1011 1011" → **NO**

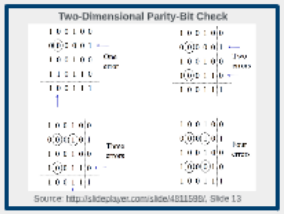
Pros: Very Simple
Cons: Inefficient; Easily susceptible to problems

EDS II: Parity Bits

Added to the group of source bits to detect if there is a single or any other odd number of errors in the sequence.

Types: One-Dimensional / Two-Dimensional; Even/ Odd
Pros: Less, but still Simple
Cons: Even number of flipped bits makes parity erroneous





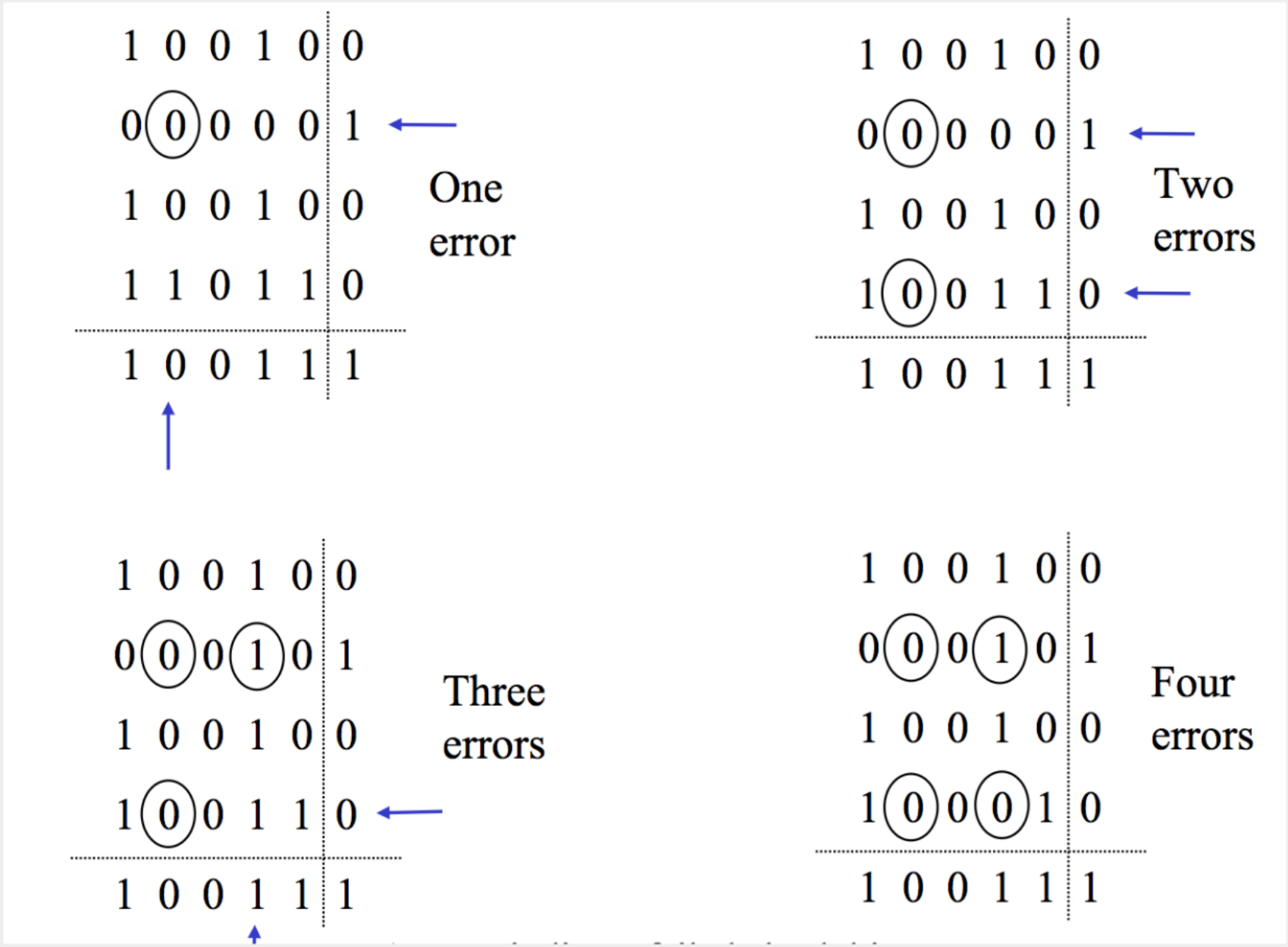
One-Dimensional Parity-Bit Check

Two Dimension Parity Check – Parity Byte

0110100	1
1011010	0
0010110	1
1110101	1
1001011	0
1000110	1

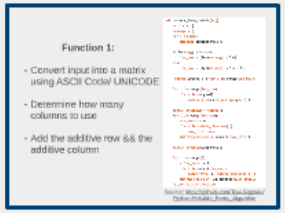
Source: https://image.slidesharecdn.com/lecture3logicallinklayer-140821205754-phpapp01/95/lecture-3-logical-link-layer-60-638.jpg?cb=1408654854

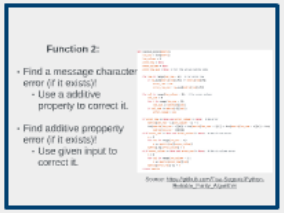
Two-Dimensional Parity-Bit Check



Source: http://slideplayer.com/slide/4811598/, Slide 13

How the Algorithm Works





Additive Parity Algorithm Demonstration:

- Use a Matrix to store the information, and add the sums.
- Then take the last (rightest) digit of the sum.

Number-pattern Sent:

3	8	3	7 → 21
1	4	3	7 → 15
1	7	8	4 → 20
3	2	2	7 → 14
1	2	2	2 → 7
|	|	|	|
9	23	18	27

Number-pattern Received & Check:

3	7	3	7 → 20 → NO
1	4	3	7 → 15
1	8	8	4 → 21 → NO
3	2	2	7 → 14
1	2	2	2 → 7
|	|	|	|
9	23	18	27
	|		
	YES		

Function 1:

- Convert input into a matrix using ASCII Code/ UNICODE
- Determine how many columns to use
- Add the additive row && the additive column

```
def numbers_into_matrix(msg):
    matrix = []
    message = []
    for i in msg:
        message.append(ord(i))

    if len(msg) % 4 == 0:
        len_row = (len(message) // 4)
    else:
        len_row = (len(message) // 4 + 1)

    [matrix.append([]) for i in range(len_row)]

    for i in range(len_row):
        for a in range(4):
            matrix[i].append(message.pop(0))

    matrix_checksum = matrix[:]
    for i in range(len_row):
        row_total = 0
        for e in matrix_checksum[i]:
            row_total += e
        matrix_checksum[i].append(row_total)

    matrix_checksum.append([])

    for i in range(5):
        colum_total = 0
        for a in range(len_row):
            colum_total += matrix_checksum[a][i]
        matrix_checksum[-1].append(colum_total)
    return matrix_checksum
```

Source: https://github.com/Tisa-Segovic/Python-Reliable_Parity_Algorithm

Function 2:

- Find a message character error (if it exists)!
  - Use a additive property to correct it.
- Find additive propperty error (if it exists)!
  - Use given input to correct it.

```python
def correct_error(matrix):
    len_row = len(matrix)
    len_column = 5
    error_row = None
    error_column = None
    error_row_sum = None  # For the actual matrix code

    for row in range(len_row - 1):  # For error row
        if np.sum(matrix[row][:4]) != matrix[row][4]:
            error_row = row
            error_row_sum = np.sum(matrix[row][:4])

    for col in range(len_column - 1):  # For error column
        col_sum = 0
        for i in range(len_row - 1):
            col_sum += matrix[i][col]
        if col_sum != matrix[-1][col]:
            error_column = col

    if error_row == None and error_column == None:  # No error
        matrix[len_row - 1][len_column - 1] = (
        int(matrix[len_row - 1][0]) + int(matrix[len_row - 1][1]) + int(matrix[len_row - 1][2]) + int(
            matrix[len_row - 1][3]))
    elif error_row == None and error_column != None:  # Matrix row error
        m = 0
        for row in range(len_row - 1):
            m += matrix[row][error_column]
        matrix[-1][error_column] = m
    elif error_column == None and error_row != None:  # Matrix column error
        n = 0
        for col in range(len_column - 1):
            n += matrix[error_row][col]
        matrix[error_row][-1] = n
    return matrix
```

Source: https://github.com/Tisa-Segovic/Python-Reliable_Parity_Algorithm

EDS III: Checksums

Modular Arithmetic Sum of all *Message Digits* of fixed length added to the end of the message.

Decimal 1234 → Simple Checksum → 1+2+3+4 →
10 → Add last sum digit12340

Types: Parity Check, Simple Checksum, Modularsum (two's compliment)
Pros: Builds onto previous EDS
Cons: No auto-correction (unless combined with another algorithm)

Improvements to be Made...

- Decrease the number of collisions
- Improve the speed (make it faster)

Source: https://images.idgesg.net/images/article/2017/09/networking-100735059-large.jpg

Cited Sources

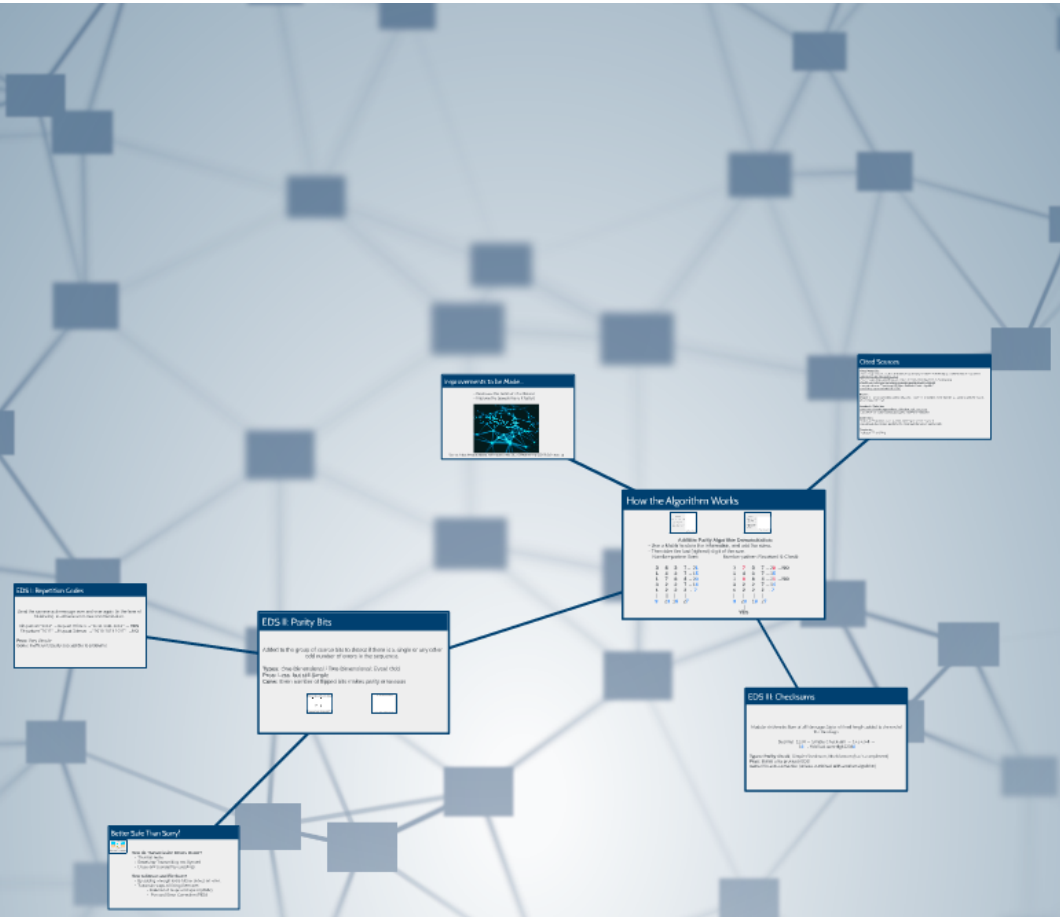
Visual Materials:
https://image.slidesharecdn.com/lecture3logicallinklayer-140821205754-phpapp01/95/lecture-3-logical-link-layer-60-638.jpg?cb=1408654854
https://images.idgesg.net/images/article/2017/09/networking-100735059-large.jpg
http://media.techtarget.com/digitalguide/images/Misc/wsrm_fig3.gif
https://github.com/Tisa-Segovic/Python-Reliable_Parity_Algorithm
http://slideplayer.com/slide/4811598/

Books:
Chapter 5 - Error-Correcting Codes: Mistakes That Fix Themselves, Nine Algorithms that Changed the Future, *MacCormick*, 2012 ed.

Academic Websites:
https://en.wikipedia.org/wiki/Error_detection_and_correction
https://www.computerscience.gcse.guru/theory/error-detection

Code from:
Professor ZZ skeleton code used for Unit Project 1, Unit Project 2
Pseudocode from Nine Algorithms that Changed the World, MacCormick

Thanks to:
Professor ZZ, and Bing



Reliable Messaging

By Tisa (te557) & Ruofan (rz829)