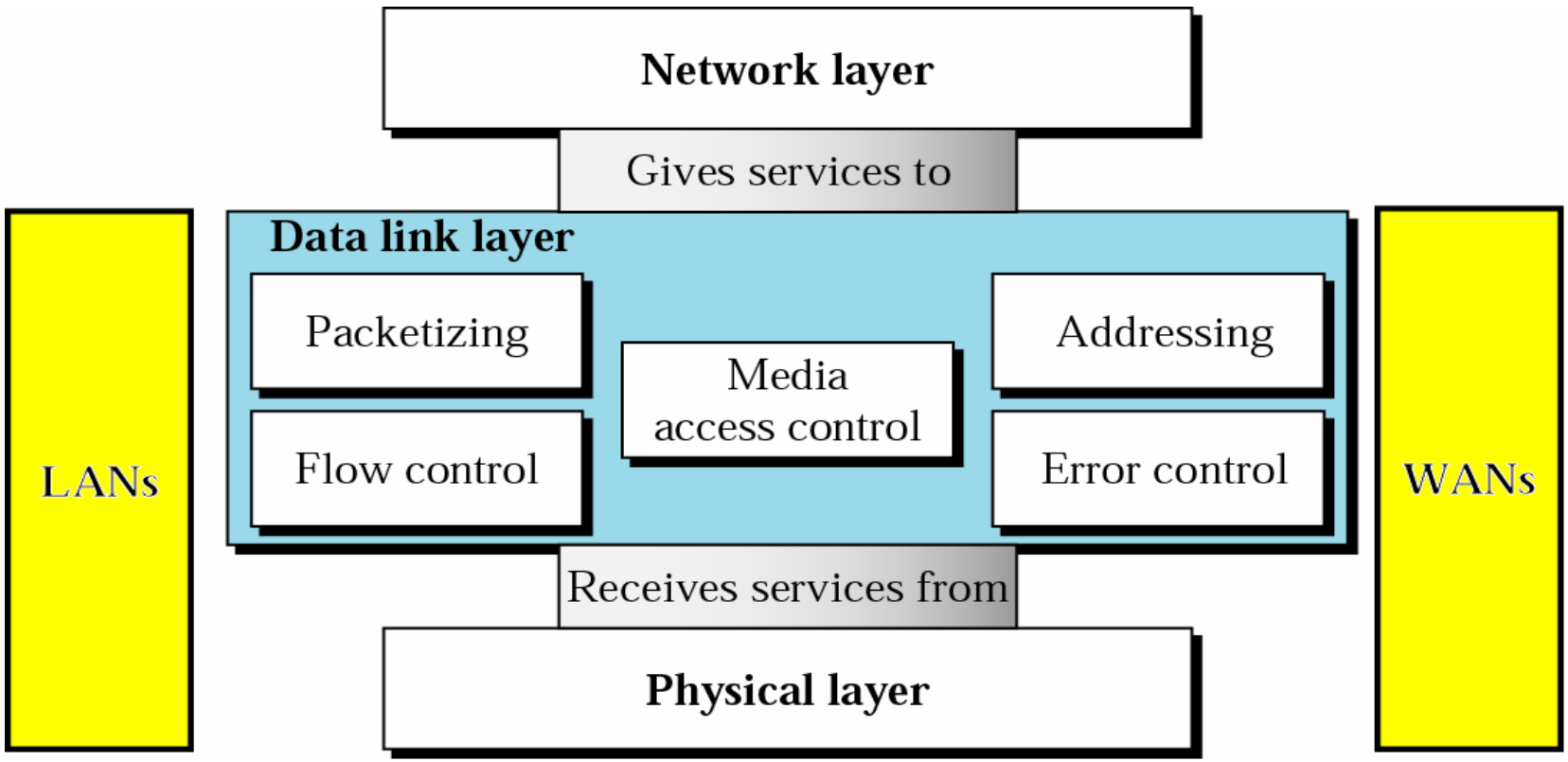
Data Link Layer

Paramate Horkaew

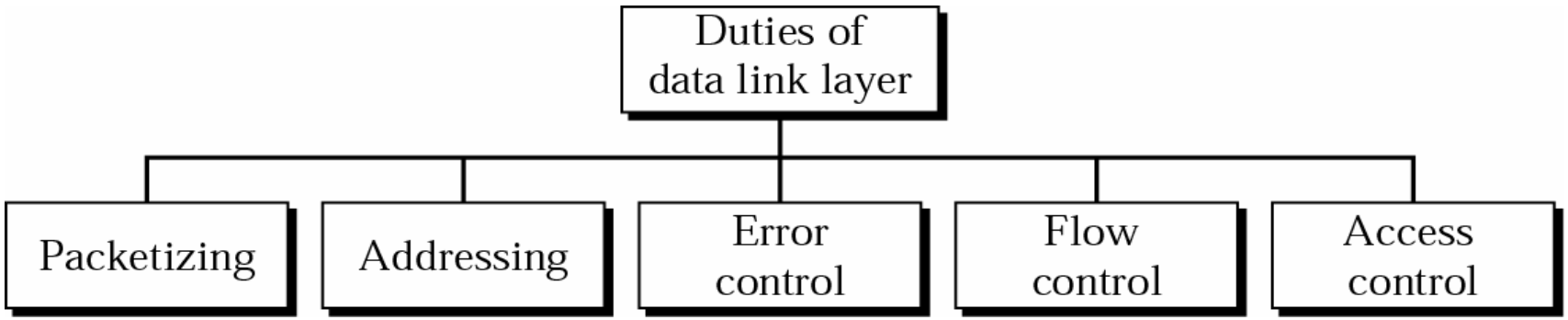
School of Computer Engineering
Institute of Engineering, Suranaree University of Technology

Position of Data Link Layer

Data Link Layer ทำหน้าที่เชื่อมต่อระหว่าง Network และ Physical Layer
โดยเฉพาะอย่างยิ่ง รับผิดชอบการส่งข้อมูล hop-to-hop อย่างถูกต้อง–เหมาะสม

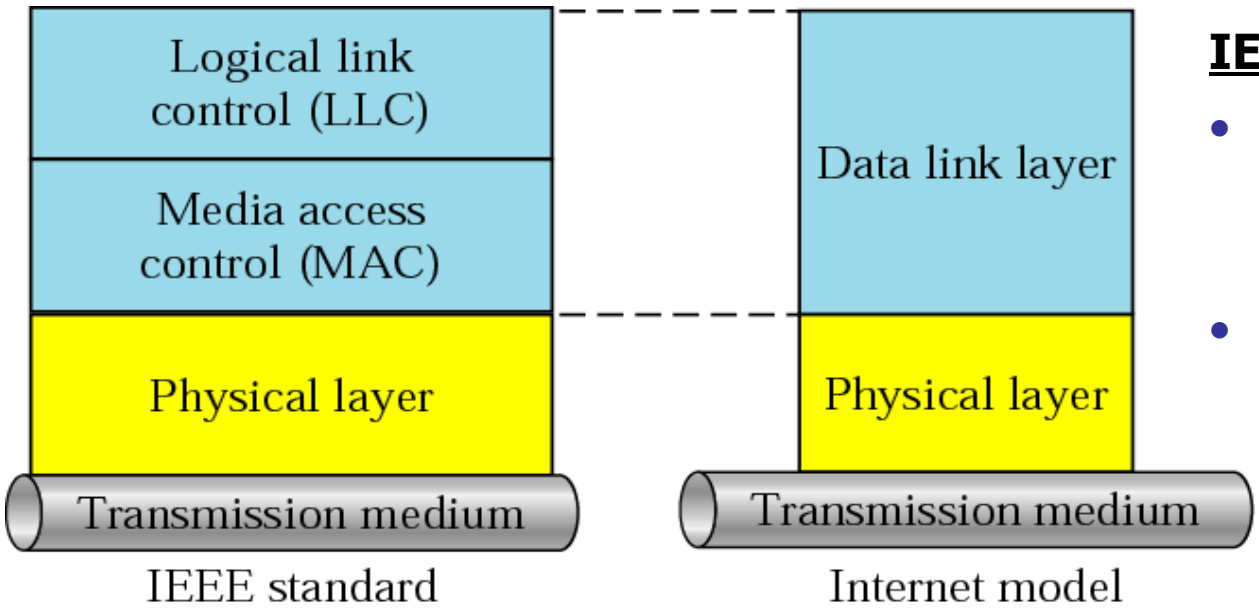


Functions of Data Link



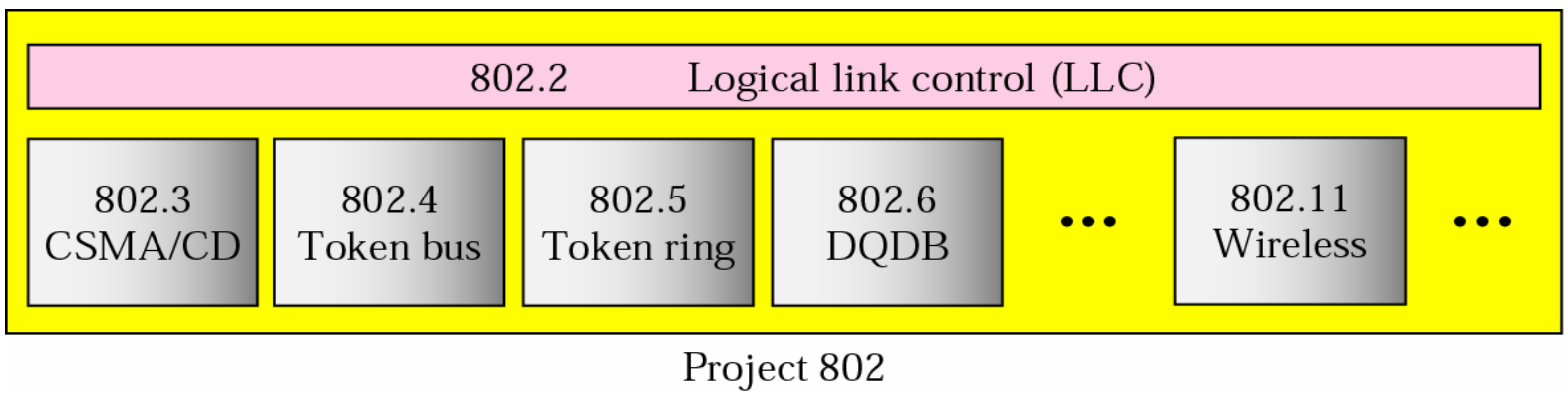
- **Packetizing** นิยามการ Encapsulate ข้อมูลในรูปแบบของ Packet หรือ Cell ตามข้อกำหนดของ Protocol
- **Addressing** นิยามกลไกการกำหนดตำแหน่งของอุปกรณ์ใน Local Network
- **Error Control** ควบคุม (ตรวจจับ และแก้ไข) ข้อผิดพลาดของข้อมูล
- **Flow Control** กำหนดปริมาณข้อมูลในการส่งให้สอดคล้องกับสมรรถนะในการประมวลผลของอุปกรณ์ด้านรับ เพื่อป้องกันการไหลท่วมท้นของข้อมูล
- **Medium Access Control (MAC)** ควบคุม หรือจัดสรร การเข้าถึง (เข้าใช้) ตัวกลางของอุปกรณ์สื่อสารข้อมูล

Local Area Network Model



IEEE แบ่ง DLL ออกเป็น

- **LLC** ได้แก่การเชื่อมโยงระหว่างอุปกรณ์ภายในเครือข่าย
- **MAC** ได้แก่การเข้าถึงตัวกลางการสื่อสาร



Error Detection and Correction

Paramate Horkaew

School of Computer Engineering
Institute of Engineering, Suranaree University of Technology

Lecture Outline

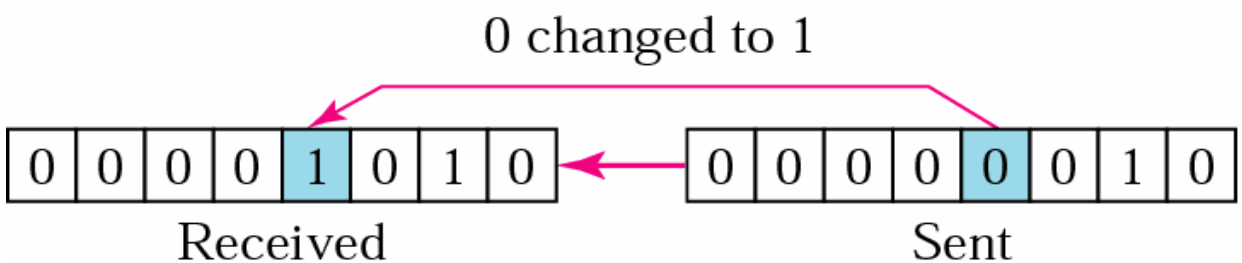
- **Type of Errors**
  - Single Bit Error
  - Burst Error
- **Detection**
  - Redundancy
  - Parity Check
  - Cyclic Redundancy Check (CRC)
  - Checksum
- **Error Correction**
  - Retransmission
  - Forward Error Correction
  - Burst Error Correction

Types of Errors (I)

การรบกวนที่เกิดขึ้นระหว่างการรับส่งข้อมูล อาจก่อให้เกิดการเปลี่ยนรูปร่างของสัญญาณ ซึ่งทำให้การตีความที่ด้านรับผิดพลาด ซึ่งในกรณี การสื่อสารข้อมูลดิจิตอล รหัส 0 จะเปลี่ยนเป็น 1 และรหัส 1 จะเปลี่ยนเป็น 0

- **Single-Bit Error**

  หมายถึง การผิดพลาดที่เกิดขึ้นเพียง 1 บิต ต่อหน่วยข้อมูล 1 หน่วย (เช่น Byte, Character, Data Unit, หรือ Packet)

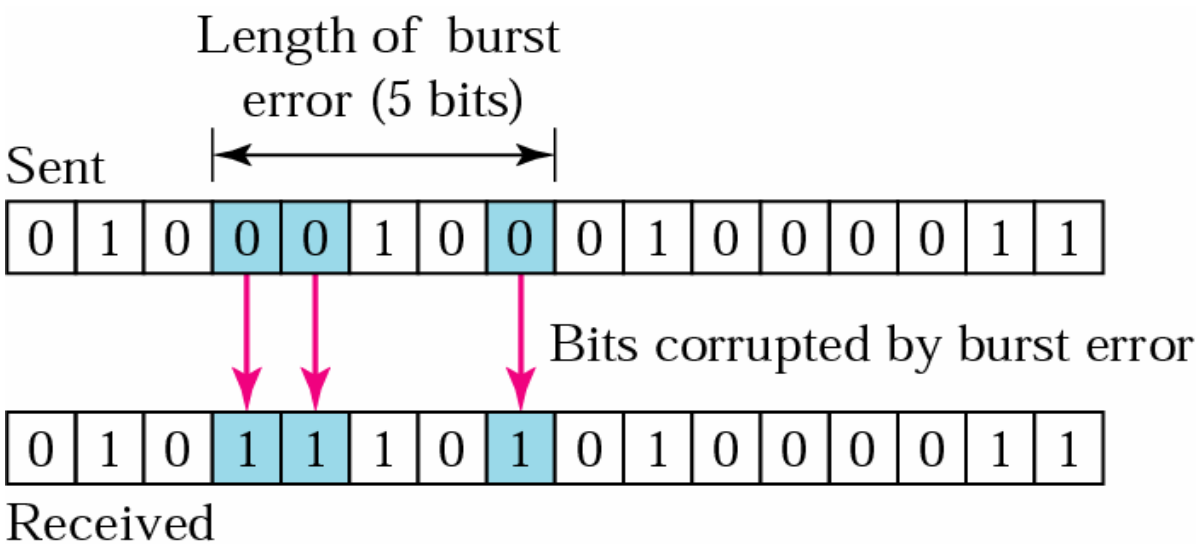


การสื่อสารข้อมูลทั่วไป มีโอกาสเกิด Error ชนิดนี้น้อยมาก เนื่องจากช่วงเวลา 1 บิตมีค่าสั้นกว่าสัญญาณรบกวนปกติ (แต่มักเกิดขึ้นกับการส่งข้อมูลแบบขนาน)

Types of Errors (II)

- **Burst Error**

หมายถึง การผิดพลาดที่เกิดขึ้นตั้งแต่ 2 บิตขึ้นไป ต่อหน่วยข้อมูล 1 หน่วย (เช่น Byte, Character, Data Unit, หรือ Packet) จากตัวอย่าง สังเกตว่า Error ชนิดนี้ **ไม่จำเป็นต้องเกิดกับบิตที่ติดกัน** ความยาวของ Error วัดได้จาก บิตแรก จนถึงบิตสุดท้าย ที่เกิดข้อผิดพลาด โดยที่ข้อมูลบางบิตในช่วงนั้นอาจถูกต้องก็ได้ จำนวนบิตที่อาจเกิด Error ขึ้นกับช่วงเวลาของ Noise



Detection

จุดประสงค์หลักของการศึกษา Error คือเพื่อต้องการแก้ไข ทั้งนี้จำเป็นต้องมีการตรวจจับ Error ก่อน

- **Basic Redundancy Concept**

  วิธีตรวจจับ Error ที่ง่ายที่สุด คือการส่งข้อมูล **ซ้ำซ้อน** 2 ชุด แล้วอุปกรณ์ทางด้านรับจะทำการเปรียบเทียบหาความแตกต่าง ซึ่งถ้ามี แสดงว่าได้มี Error เกิดขึ้น

- **ข้อดี**

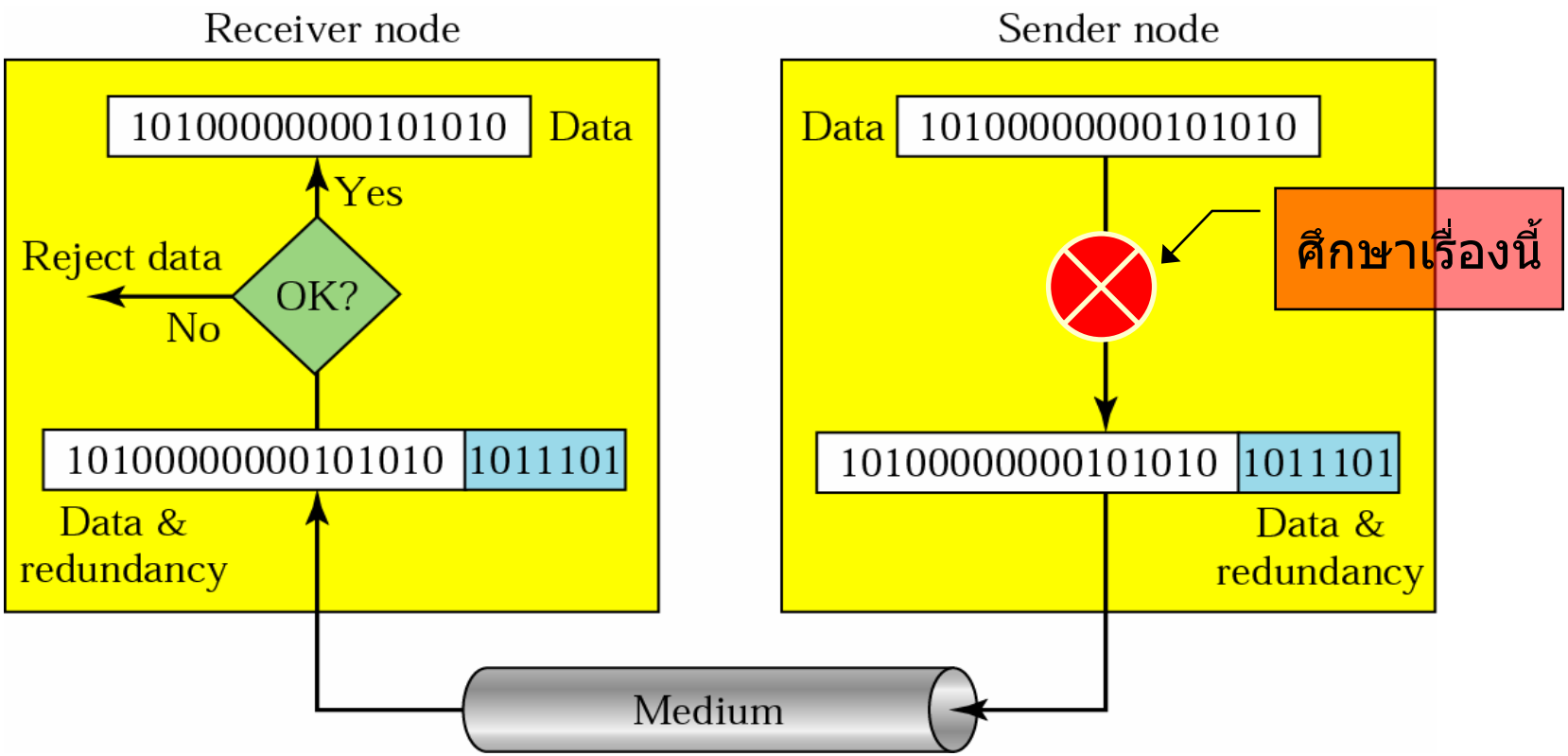
  เป็นวิธีที่สะดวก โอกาสตรวจไม่พบต่ำ เนื่องจากโอกาสที่ข้อมูลทั้ง 2 ชุดที่ซ้ำกัน ผิดตำแหน่งเดียวกัน เกือบเป็นไปไม่ได้

- **ข้อเสีย**

  ช้ามาก และ สิ้นเปลืองทรัพยากรช่องสัญญาณ เช่นต้องเพิ่ม BW อีก N เท่า

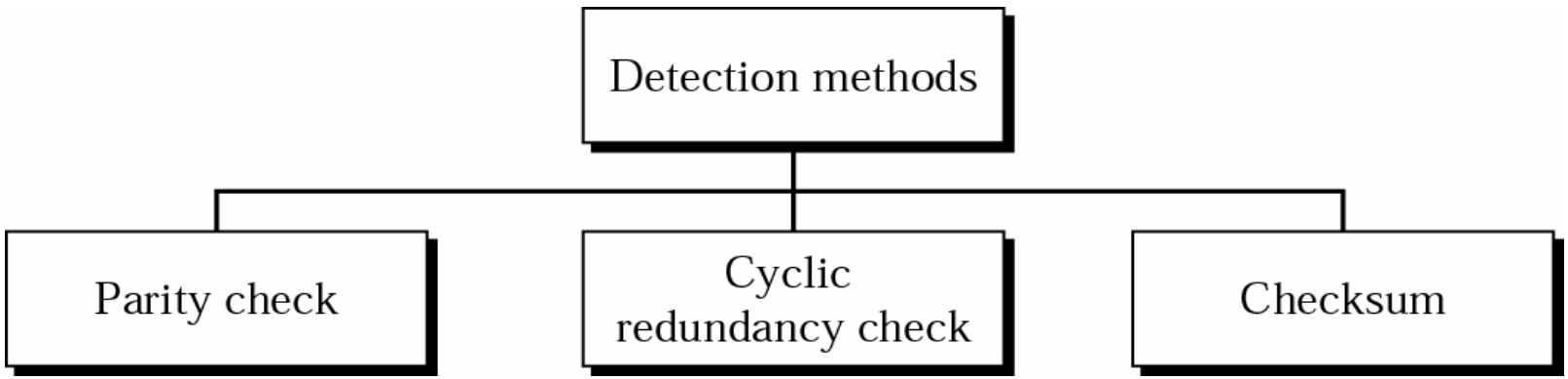
Redundancy

ด้วยหลักการเดียวกัน เราอาจเพิ่มจำนวนบิตข้อมูล (ที่มีขนาดเล็กกว่าหน่วยข้อมูล) เพื่อใช้ในการตรวจจับ Error โดยเฉพาะ ดังรูป



Detection Methods

ขั้นตอนวิธีการตรวจสอบข้อผิดพลาด ที่นิยมใช้ในการสื่อสารข้อมูลมี 3 วิธี ได้แก่ Parity Check, CRC Check และ Checksum

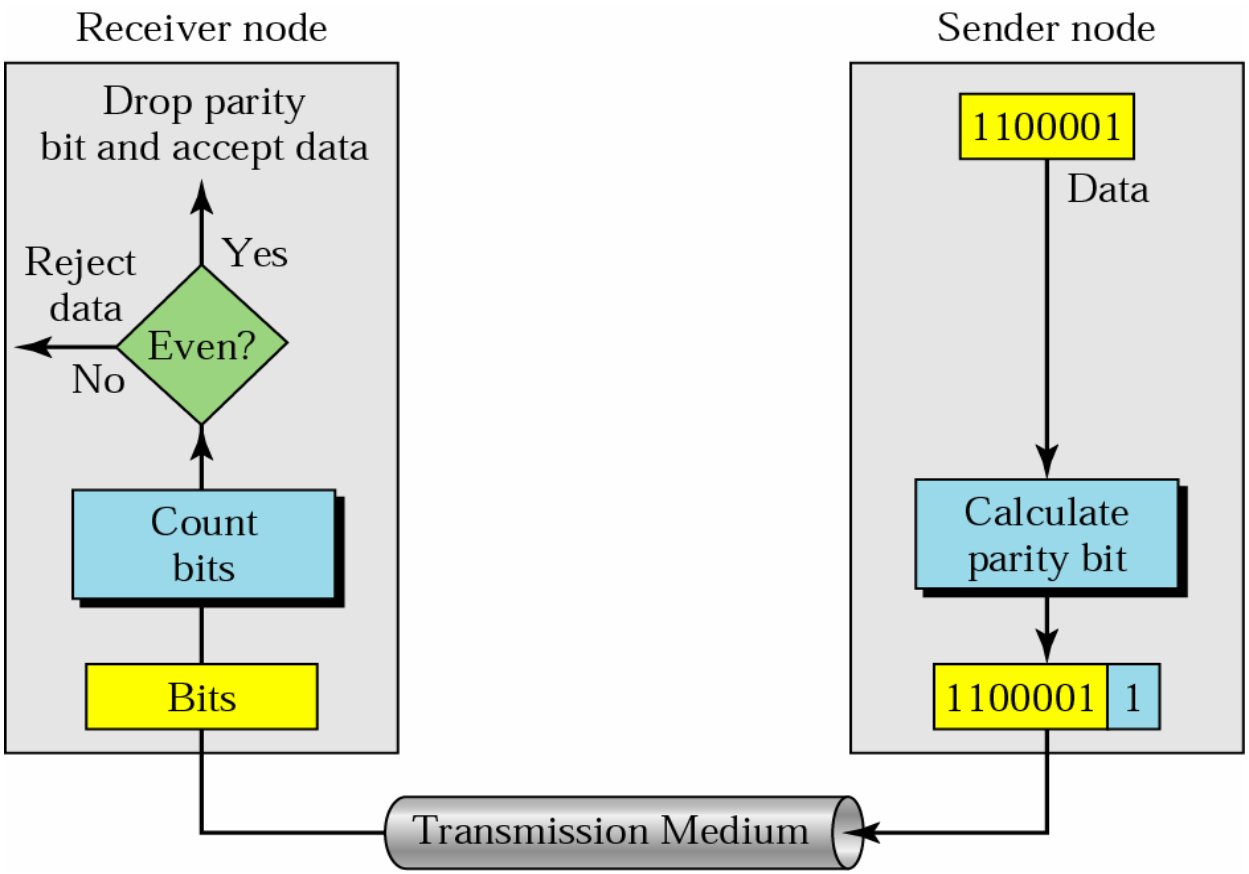


นอกจากนี้ยังมีวิธีการเข้ารหัสขั้นสูง ที่มีประสิทธิภาพมากกว่า ได้แก่
Hadamard, Quadratic Residue, Golay, Reed Codes เป็นต้น

** Golay Code ได้มีการใช้จริงในโครงการอวกาศ Voyager I, II (1979-81)

Parity Check (I)

เป็นขั้นตอนวิธีแพร่หลายที่สุด และใช้ทรัพยากรน้อยที่สุด มี 2 ประเภท ได้แก่ อย่างง่าย และ 2D



Parity Check (II)

Even Parity เพิ่มบิตท้ายหน่วยข้อมูล เพื่อให้จำนวน 1 ทั้งหมดรวม Parity เป็นคู่

Odd Parity เพิ่มบิตท้ายหน่วยข้อมูล เพื่อให้จำนวน 1 ทั้งหมดรวม Parity เป็นคี่

1110 111 1100 100 → 1110 111 0 1100 100 1

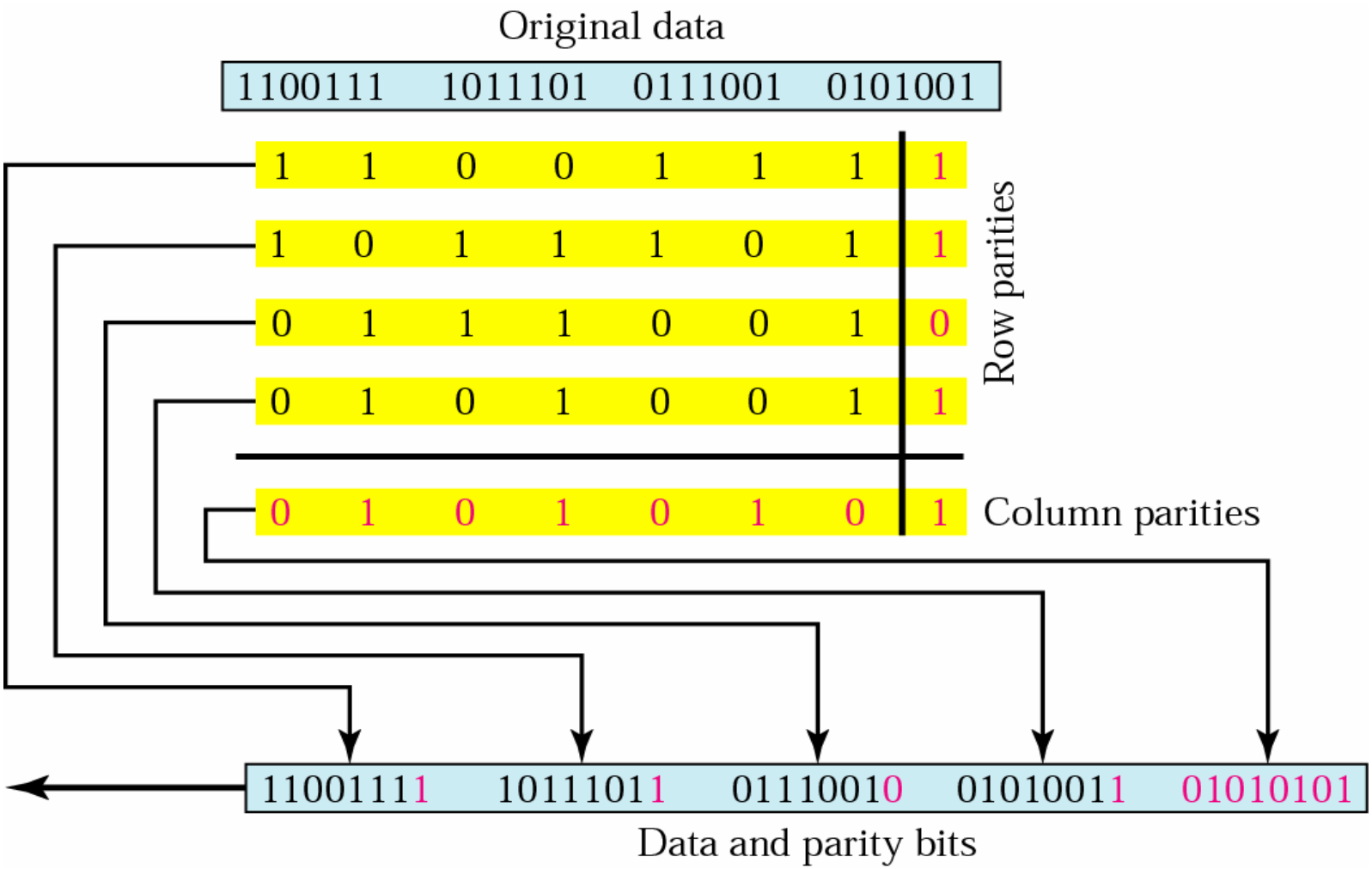
Send W and D ASCII Codes — Parity 0 for W and 1 for D

1110 101 0 1111 100 1 → 1110 101 0 1111 100 1

Errors 1 Bit at W and 2 Bits at D — Error found in W but not D

Parity Check สามารถตรวจจับ Single-Bit Error เท่านั้น ไม่เหมาะกับกรณี Burst Error นอกจากเราจะทราบล่วงหน้าว่าจะเกิด Burst Error **เป็นจำนวนคี่**

2-Dimensional Parity Check



2-D Parity Check Performance

2D Parity Check สามารถตรวจจับ Burst Error ได้ **ยกเว้น** กรณีที่หน่วยข้อมูลจำนวนคู่ (เช่น 2 Bytes) มีบิตผิดพลาด ณ ตำแหน่งตรงกัน เป็นจำนวนคู่

ตัวอย่าง

สมมติว่าส่งข้อมูล ASCII 4 ตัวอักษร พร้อม Column Parities ดังต่อไปนี้

1010100 **1** 0011100 **1** 1101110 **1** 1110011 **1** 10101010

ภายในตัวกลางเกิดสัญญาณรบกวนขนาด 8 Bits ทำให้มีข้อมูลบางส่วนผิดพลาด (Burst)

1010**0011** **1000**100 1 1101110 1 1110011 1 10101010

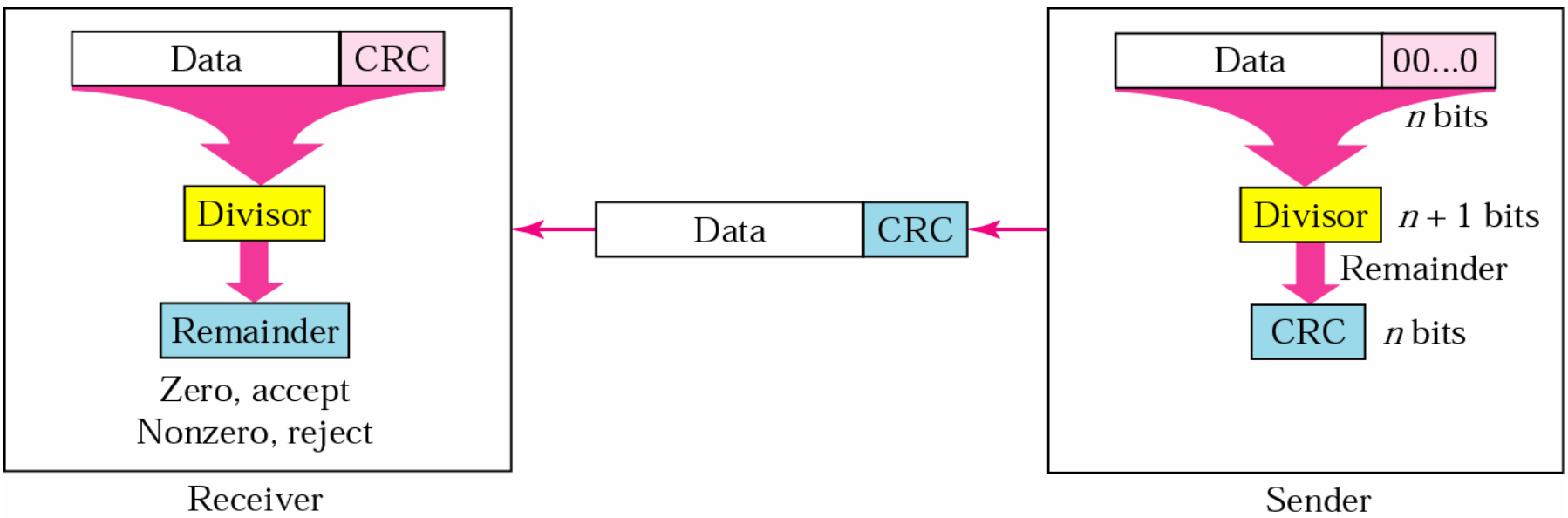
เมื่อด้านอุปกรณ์ด้านรับตรวจสอบ Parity พบว่าเกิดข้อผิดพลาดขึ้น (ต้องส่งใหม่)

10100011 10001001 11011101 11100111 **1**0**101**0**1**0

Cyclic Redundancy Check

CRC Check ใช้หลัก การหารเลขฐาน 2 โดยการเพิ่ม **กลุ่มของบิต** ต่อท้ายหน่วยข้อมูล เพื่อให้ ผลลัพธ์สามารถหารด้วย จำนวนที่กำหนดไว้ล่วงหน้า (เรียกว่า ตัวหาร – Divisor หรือ กุญแจรหัส) ลงตัว

ทางด้านรับตรวจสอบ โดยการ หารข้อมูลที่รับได้ด้วย กุญแจรหัส ซึ่งถ้าผลลัพธ์เป็นการหารลงตัวแสดงว่าข้อมูลถูกต้อง มิฉะนั้น แสดงว่าเกิด Error ขึ้น



CRC Basics

คุณสมบัติของ CRC มี 2 ประการ

1. ความยาวของ CRC Bits น้อยกว่าของตัวหาร (Divisor) อยู่ 1
2. เมื่อ CRC นำมาต่อท้ายข้อมูลเดิมแล้ว ทำให้หารด้วย Divisor ลงตัว

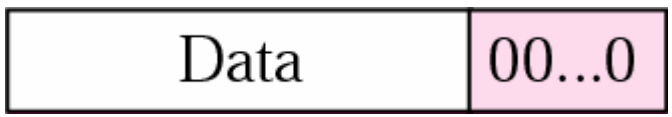
หลักการคำนวณ Binary Division (คล้ายกับการหารยาวปกติ)

1. ตัวตั้งที่สามารถหารด้วย Divisor ได้ต้องมีจำนวนบิต (ความยาว) เท่ากับตัวหาร และผลหารที่ได้มีค่าเท่ากับ 1
2. การลบตัวตั้งด้วยตัวหารในแต่ละขั้น ใช้การลบเลขฐานสองแบบไม่มีตัวทด
3. ถ้าตัวตั้งมีความยาวน้อยกว่า Divisor ผลหารมีค่าเท่ากับ 0 และแทนค่าตัวหารในขั้นตอนนั้น ด้วยเลข 0 ที่มีความยาวเท่ากับ Divisor
4. เศษเหลือในขั้นตอนสุดท้าย เมื่อใช้เลขตัวตั้งครบทุกบิต

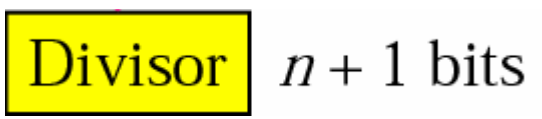
CRC Generator (I)

การสร้าง CRC Bits สำหรับข้อมูลที่กำหนดให้มีขั้นตอนดังนี้

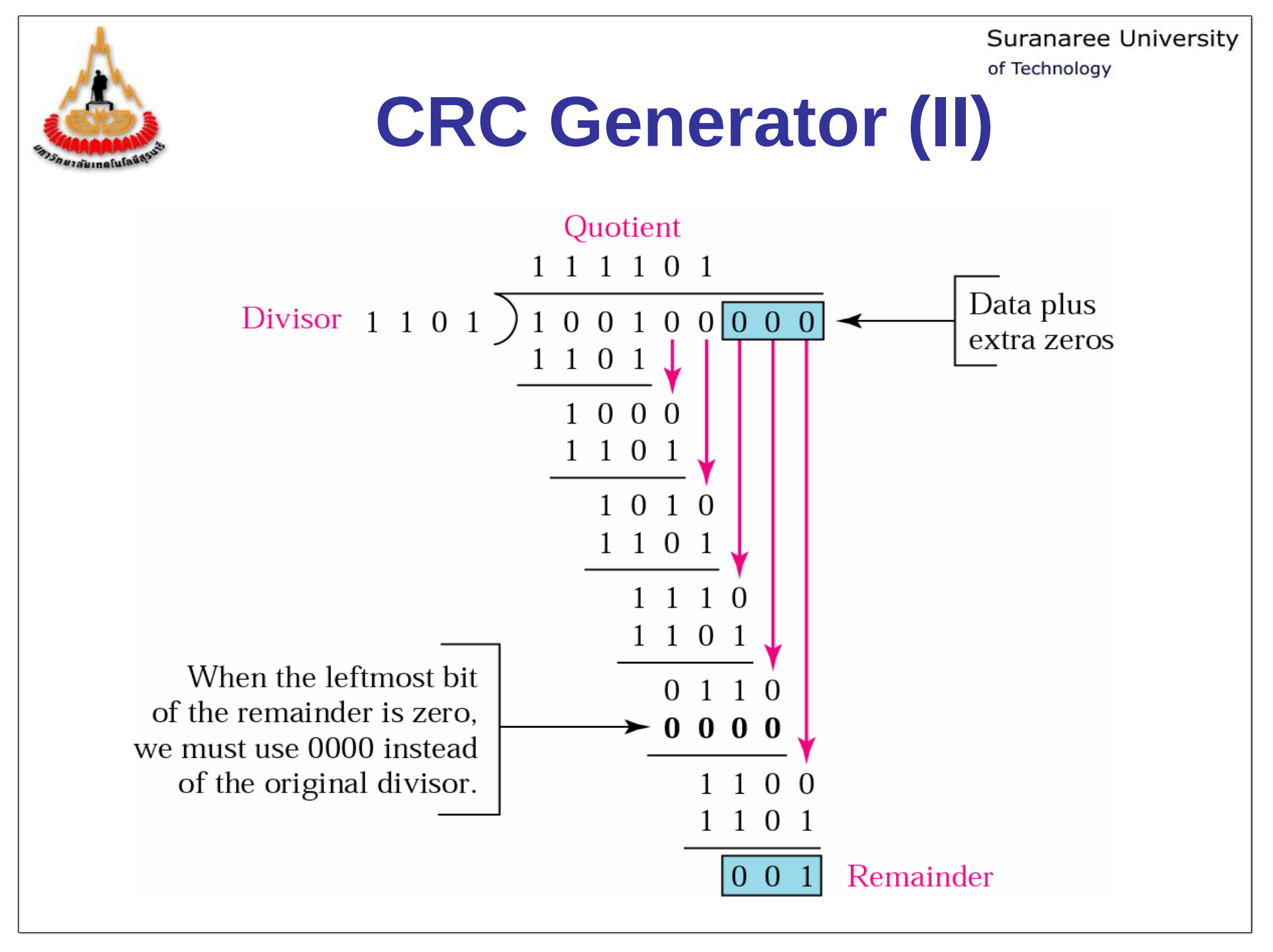
1. กำหนดให้ความยาวของ CRC เท่ากับ n Bits
2. เลือกตัว Divisor ที่มีความยาว n + 1 Bits และมีบิตซ้ายมือเท่ากับ 1
3. นำเลข 0 จำนวน n Bits มาต่อท้ายข้อมูล



4. หารผลลัพธ์ที่ได้ในข้อ 3 ด้วย Divisor ด้วยวิธี Binary Division



5. เศษที่เหลือคือ CRC – ทำให้มีความยาว n Bits โดยเติม 0 ไปด้านซ้ายมือ
6. แทนที่เลข 0 ในข้อ 3 ด้วย CRC Bits ที่ได้ในข้อ 5

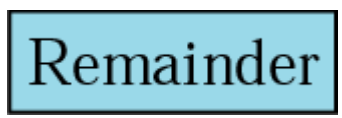

Suranaree University
of Technology
CRC Generator (II)
Quotient
1 1 1 1 0 1
Divisor 1 1 0 1
1 0 0 1 0 0 0 0 0
Data plus
extra zeros
1 1 0 1
1 0 0 0
1 1 0 1
1 0 1 0
1 1 0 1
1 1 1 0
1 1 0 1
0 1 1 0
When the leftmost bit
of the remainder is zero,
we must use 0000 instead
of the original divisor.
0 0 0 0
1 1 0 0
1 1 0 1
0 0 1
Remainder

CRC Checker (I)

การตรวจสอบ CRC Bits สำหรับข้อมูลที่กำหนดให้มีขั้นตอนดังนี้

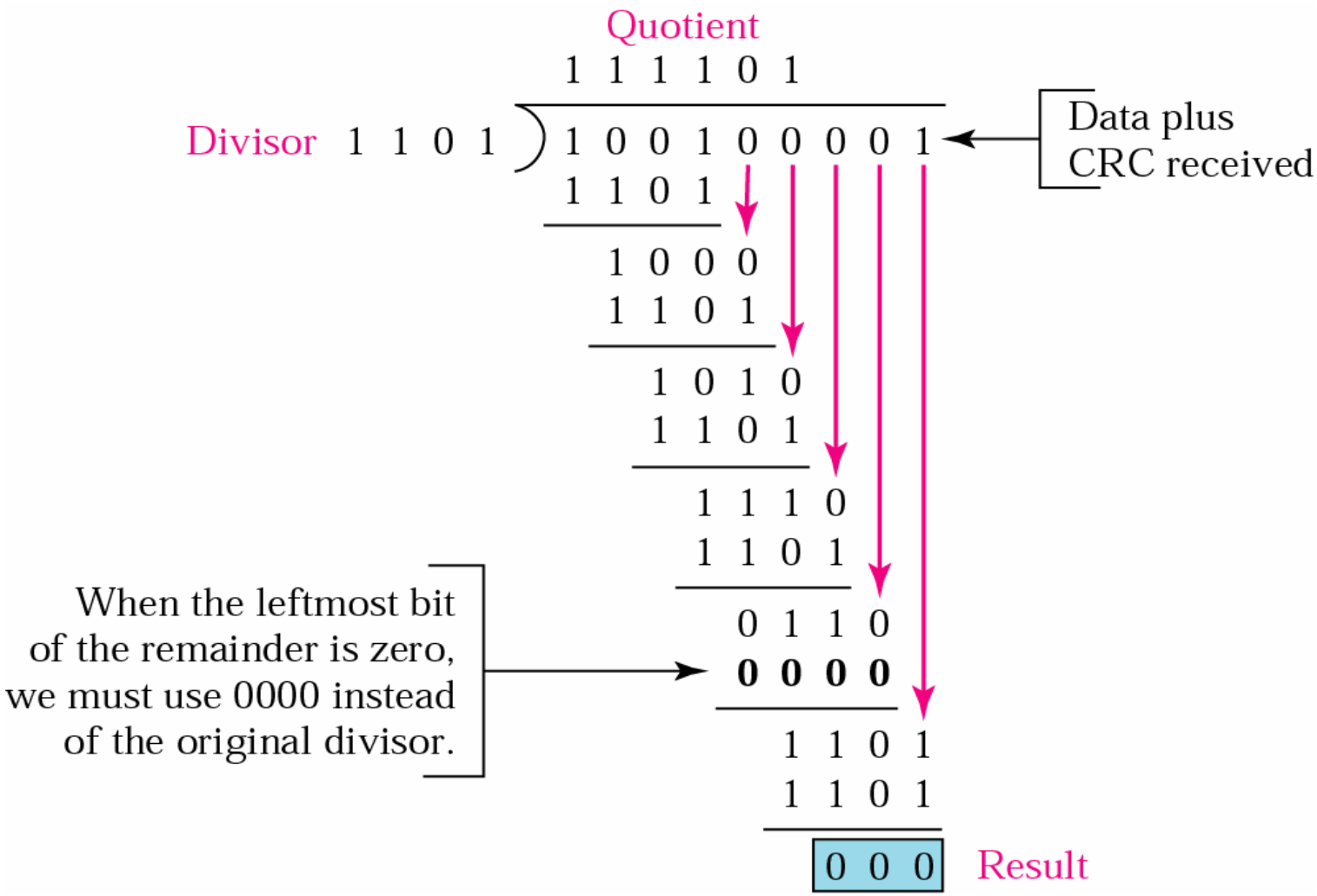
1. รับข้อมูลที่มี CRC ต่อท้าย n Bits
2. เลือกตัว Divisor ที่มีความยาว n + 1 Bits **ตัวเดียวกับทางด้านรับ**
3. หารข้อมูลที่ได้ในข้อ 1 ด้วย Divisor ด้วยวิธี Binary Division

4. พิจารณาเศษที่เหลือจากการหาร



5. ถ้าเศษที่เหลือมีค่าเท่ากับ 0 แสดงว่าข้อมูลที่รับเข้ามาในข้อ 1 ถูกต้อง ให้ตัดส่วนที่เป็น CRC ทิ้งเพื่อนำข้อมูลไปใช้งาน มิฉะนั้น แสดงว่าเกิด Error ขึ้นให้ส่งสัญญาณไปยังอุปกรณ์ส่ง เพื่อให้ทำการส่งข้อมูลใหม่

CRC Checker (II)



Divisor Polynomial

Divisor สำหรับ **CRC** มักจะแสดงในรูปของฟังก์ชันพหุนาม (Polynomial) เพราะ

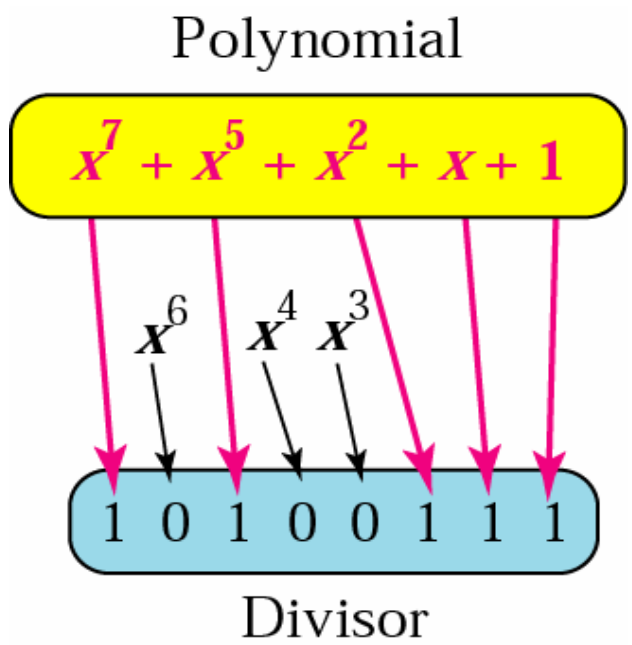
1. เป็นการแสดงในรูปแบบที่กระชับ สื่อความหมายได้ดี
2. เอื้อประโยชน์ต่อการพิสูจน์การทำ CRC ทางคณิตศาสตร์

พหุนามดีกรีเท่ากับ 7

$$x^7 + x^5 + x^2 + x + 1$$

คุณสมบัติของ Divisor มีดังนี้

1. ต้องหารด้วย x **ไม่ลงตัว**
2. ต้องหารด้วย x + 1 **ลงตัว**
3. สามารถ Detect Burst Error ความยาว ≤ degree ได้



Standard of Polynomials

ตารางด้านล่าง แสดงรูปแบบของ Divisor Polynomial ที่นิยมใช้กันในการสื่อสารด้วย Protocol ต่างๆ

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
ITU-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
ITU-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Checksum

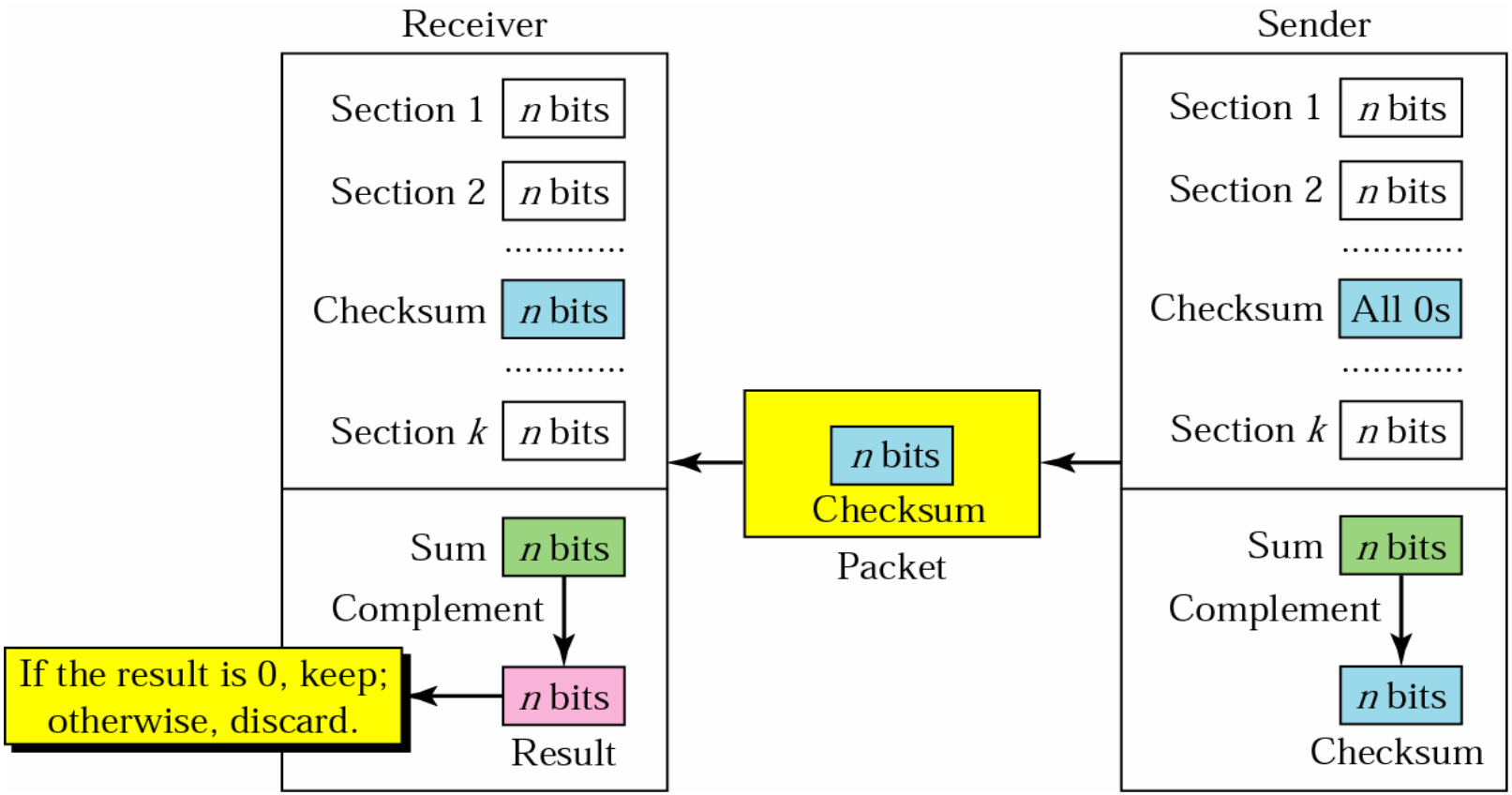
Checksum มีหลักการคล้ายกับ Parity และ CRC ตรงที่มีการใช้ข้อมูลซ้ำซ้อน (Redundancy)

หลักการทำงานของ **Checksum** สามารถอธิบายได้ดังต่อไปนี้

1. ข้อมูลต้นฉบับจะถูกแบ่งออกเป็น Segment ซึ่งมีความยาว n bits (เช่น n = 16)
2. ข้อมูลในแต่ละ segment จะนำมาบวกกันด้วยวิธี One's Complement ผลลัพธ์ที่ได้จะมีขนาด n bits.
3. ผลลัพธ์ที่ได้ในข้อ 2 จะถูกทำ Complement ซ้ำ แล้วนำไปต่อท้ายข้อมูลต้นฉบับ กลายเป็น Redundant Bits.
4. ข้อมูลต้นฉบับ พร้อมกับ Redundant Bits จะถูกส่งออกไปยังเครื่องรับปลายทาง ผ่านระบบเครือข่าย

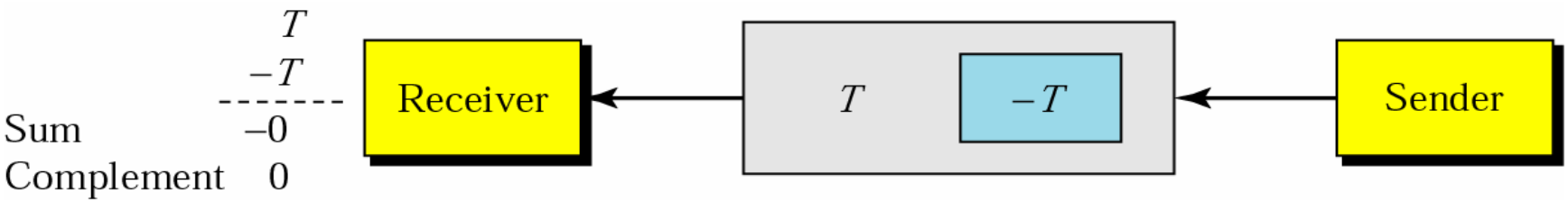
Block Diagram

ในการสื่อสารทั่วไป มักจะทำ Checksum กับข้อมูลหลาย segments พร้อมๆ กัน ดังรูป แสดงการทำ Checksum กับข้อมูล k segments แต่ละ segment ยาว n



Generating Checksum

การตรวจสอบด้วยวิธี Checksum อาจแสดงได้ด้วยแผนผังดังนี้



สมมติให้ส่งข้อมูลจำนวน 16 bits โดยแบ่งออกเป็น 2 segments และแต่ละ segment มีความยาว 8 bits ดังนี้ 10101001 00111001

The numbers are added using one's complement

10101001

00111001

Sum 11100010

Checksum 00011101

The pattern sent is 10101001 00111001 **00011101**

Detecting Checksum

สมมติให้รับข้อมูลเป็น Pattern ที่ส่งมาจากขั้นตอนที่แล้ว โดยไม่มีข้อผิดพลาด (No Error Case) ดังนี้

10101001 00111001 00011101

เมื่อบวกข้อมูลทั้ง 3 segments เข้าด้วยกัน จะได้ผลลัพธ์เป็น 1 ทั้งหมด ซึ่งเมื่อทำ Complement แล้วจะได้ผลลัพธ์เป็น 0 หมายถึงไม่มีข้อผิดพลาดใดๆ

	10101001	
	00111001	+
	00011101	
Sum	11111111	
Complement	00000000	means that the pattern is OK.

Checksum: Error Case

สมมติกรณีที่เกิด Burst Error ความยาว 5 บิตซึ่งทำให้ข้อมูลผิดไป 4 บิต ดังนี้

10101**111 11**111001 00011101

เมื่อนำข้อมูลทั้ง 3 segments มาบวกกันจะได้

	10101111	
	11111001	
	00011101	
Partial Sum	**1** 11000101	
Carry	**1**	
Sum	11000110	
Complement	**00111001**	**the pattern is corrupted**

Check Sum ไม่สามารถตรวจพบ Error ได้ถ้า

ความผิดพลาดใน segment หนึ่ง สมดุลกับความผิดพลาดในอีก segment หนึ่ง ซึ่งเกิดในตำแหน่งเดียวกัน

"จำนวน 1 และ 0 คงที่"

Lecture Outline

- **Type of Errors**
  - Single Bit Error
  - Burst Error
- **Detection**
  - Redundancy
  - Parity Check
  - Cyclic Redundancy Check (CRC)
  - Checksum
- **Error Correction**
  - Retransmission
  - Forward Error Correction
  - Burst Error Correction