การตรวจจับข้อผิดพลาด การควบคุมการไหลของข้อมูล และการควบคุมข้อผิดพลาด

(Error Detection, Flow Control and Error Control)

Error Detection Techniques

- Parity Check
- Checksum
- CRC (Cyclic Redundancy Checksum)

Parity Check

- Simple Parity Check
  - ในการส่ง-รับ จะมีการตกลงวิธีการตรวจสอบข้อมูล โดยการเพิ่มบิตจำนวน 1 บิต เรียกว่า “Parity bit” อาจเป็น 0 หรือ 1 ขึ้นอยู่กับวิธีที่ตกลงกัน
  - Even Parity เพิ่มบิตตรวจสอบ(0หรือ1)รวมกับบิตข้อมูล แล้วนับจำนวนบิต “1” ทั้งหมดให้ได้จำนวนคู่
  - Odd Parity เพิ่มบิตตรวจสอบ(0หรือ1)รวมกับบิตข้อมูล แล้วนับจำนวนบิต “1” ทั้งหมดให้ได้จำนวนคี่
  - ตัวอย่าง วิธี Even Parity

ส่ง : 100010000 รับ : 10001000 0 คำนวณแล้ว ได้ คู่ = ถูกต้อง

- วิธีนี้ถ้ามีข้อมูลผิดพลาดเพียงบิตเดียว ก็ตรวจสอบได้ ถ้าผิดหลายๆบิตจะตรวจได้กรณีที่เกิดการผิดพลาดไปเป็นเลขคี่เท่านั้น

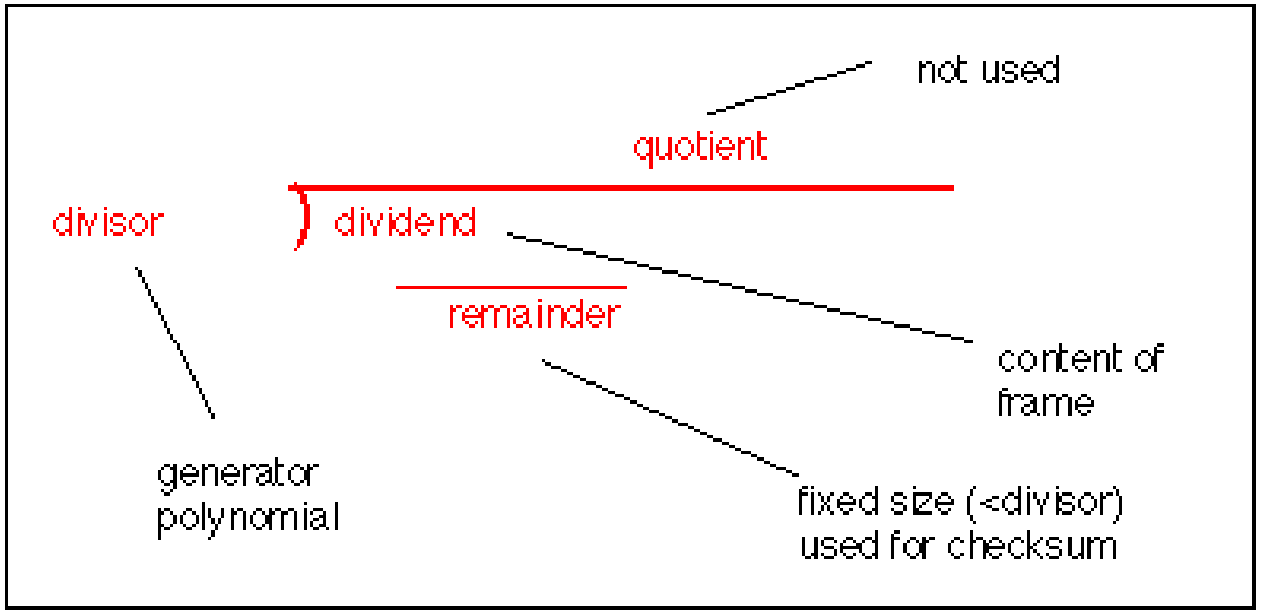
Parity Check

- Simple Parity Check
- ตัวอย่าง ส่งคำวา “world” ใช้วิธี even parity

	w	o	r	l	d
	1110111	1101111	1110010	1101100	1100100
คำนวณหา พาริตี้บิต	11101110	11011110	11100100	11011000	11001001
ฝั่งรับ คำนวณได้	6	6	4	4	4
	✓	✓	✓	✓	✓

CRC (Cyclic Redundancy Checksum)

- มีประสิทธิภาพสูงว่าแบบพาริตี้บิต
- แบบพาริตี้บิตใช้การบวก แต่ CRC ใช้การหาร
- แบบพาริตี้บิตจะแทรกบิตตรวจสอบลงในข้อมูล แต่แบบ CRC จะต้องนำบิตตรวจสอบไปต่อท้ายข้อมูล



การหาบิตตรวจสอบของเทคนิค CRC

1. ถ้าตัวหารมีจำนวนบิตเท่ากับ n+1 บิตแล้วจะต้องเติมบิต 0 จำนวน n ตัวที่ส่วนท้ายของข้อมูล
2. ใช้บิตข้อมูลลบด้วยตัวหาร(ใช้วิธี XOR) เมื่อลบแล้วผลที่ได้จากการลบ ถ้า
   2.1 บิตซ้ายสุดของเศษเป็น 1 ให้นำตัวหารมาเป็นตัวลบอีกครั้ง
   2.2 บิตซ้ายสุดของเศษเป็น 0 ให้นำ 0000 มาเป็นตัวลบ
3. ทำในข้อ 2 จนกระทั่งไม่สามารถลบกันได้อีกแล้ว (จำนวนบิตของเศษน้อยกว่าจำนวนบิตของตัวหาร) จะถือได้ว่าเศษที่ได้จากการหารนั้นคือ <u>บิตตรวจสอบ</u>
4. นำบิตตรวจสอบที่ได้ไปแทนที่บิต 0 จำนวน n ตัวที่ส่วนท้ายของข้อมูล

การหาบิตตรวจสอบของเทคนิค CRC

```
1101 | 100100000
       1101
        1000
        1101
         1010
         1101
          1110
          1101
           0110
           0000
            1100
            1101
             001
```

Divisor หามาจากสูตรโพลิโนเมียล

Remainder

ดังนั้นเวลาส่งข้อมูลไปจะได้

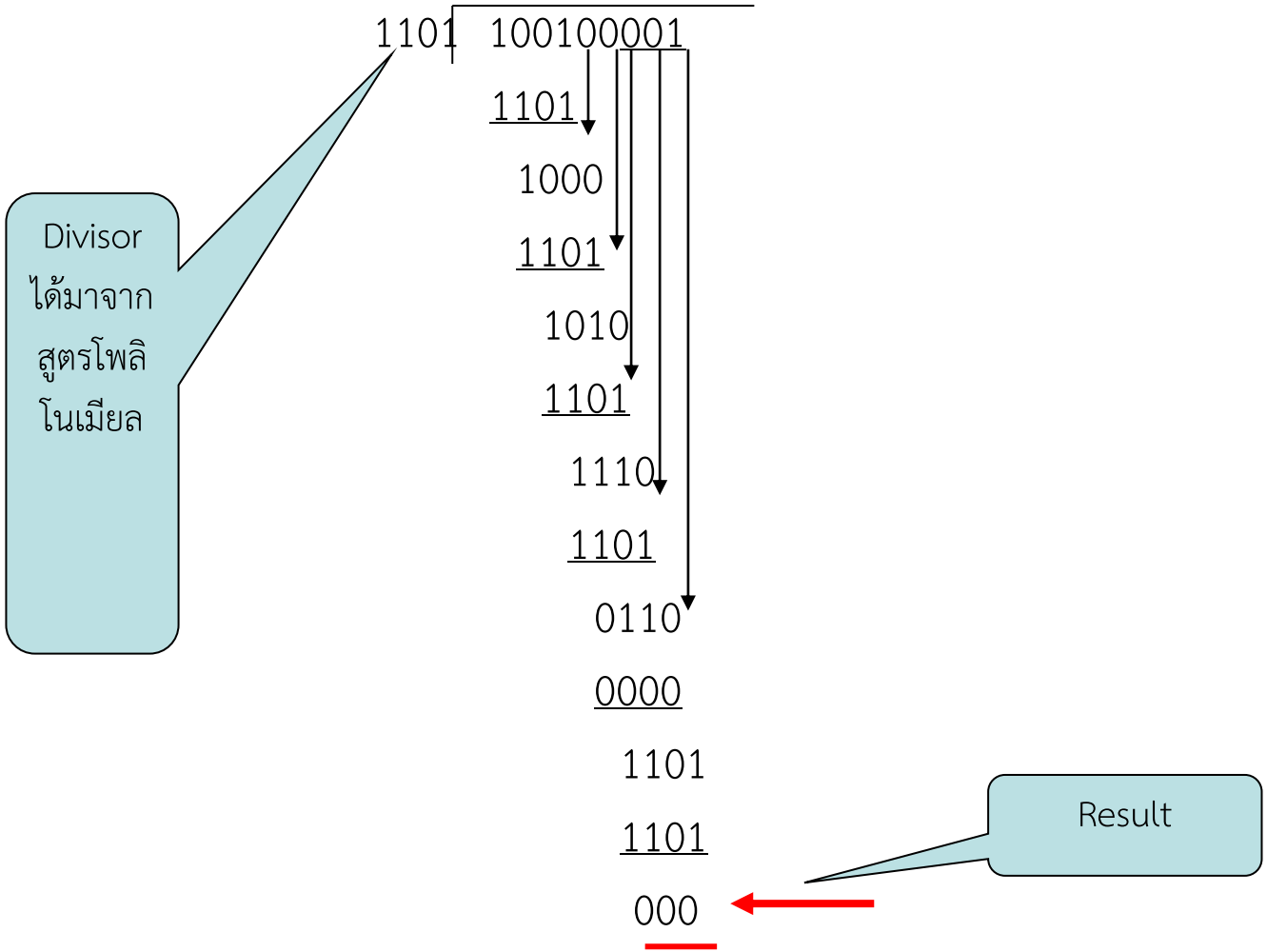
100100 + 001

= 100100001

CRC การตรวจสอบความผิดพลาดของข้อมูล

- เมื่อผู้รับได้รับบิตข้อมูลรวมทั้งบิตตรวจสอบที่อยู่ส่วนท้ายของบิตข้อมูลแล้ว จะใช้วิธีการหารในโมดูโล2 เช่นกัน
  - ถ้าเศษของการหารเท่ากับ 0 หรือหารลงตัว แสดงว่าข้อมูลชุดนั้นเป็นข้อมูลที่ถูกต้อง
  - แต่ถ้าหารไม่ลงตัวคือมีเศษ แสดงว่าข้อมูลที่ได้รับมานั้นเป็นข้อมูลผิดพลาด

การตรวจสอบของเทคนิค CRC ฝั่งรับ



CRC (Cyclic Redundancy Checksum)

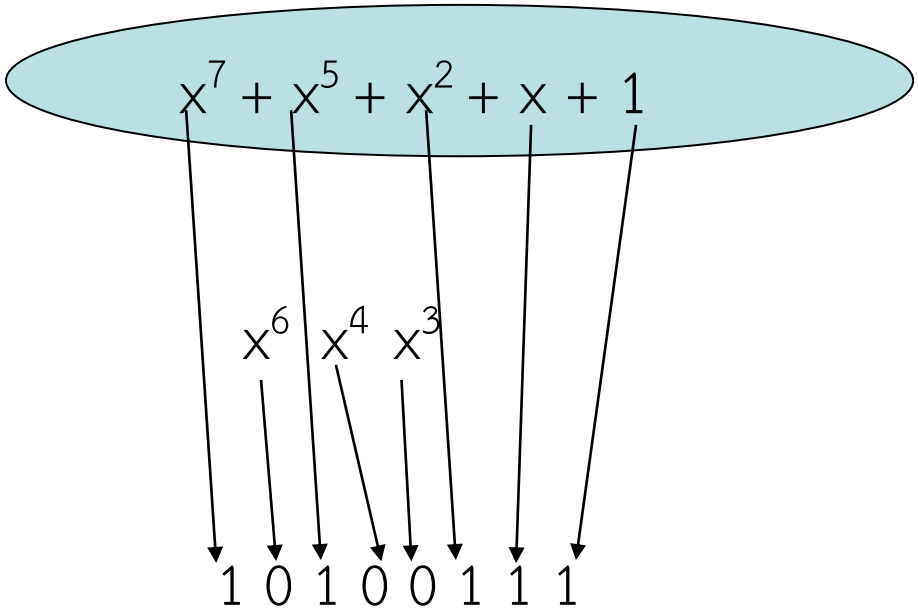
- โพลิโนเมียล (Polynomial)

  ปกติแล้วในการแทนบิตข้อมูลของตัวหารจะไม่ใช้รูปของเลขฐานสอง เนื่องจากค่อนข้างยาวและจำได้ยาก แต่จะเขียนให้อยู่ในรูปของโพลิโนเมียล เช่น ถ้าตัวหารมีค่าเป็น 10100111 จะสามารถเขียนให้อยู่ในรูปของโพลิโนเมียลได้

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

CRC (Cyclic Redundancy Checksum)

- ความสัมพันธ์กันระหว่างโพลิโนเมียลกับเลขฐานสอง



CRC (Cyclic Redundancy Checksum)

- สำหรับตัวหารที่เป็นมาตรฐานทั่วไป ที่มีการนำไปใช้งานในโปรโตคอลต่างๆ มีดังนี้

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

- วิธีการแบบ checksum จะคล้ายคลึงกับวิธีแบบ CRC และพาริตี้ ซึ่งวิธีการแบบนี้ยังคงอยู่บนพื้นฐานของการเพิ่มบิตตรวจสอบเข้าไปพร้อมกับบิตข้อมูล

0000 0101 1111 0010	0000 0101 1111 0010	0000 0101 1111 0010	00000101 11110010
0110	00010110	0111	11110111
Single–Precision	Double–Precision	Residue	Honeywell

Checksum

- ก่อนที่จะหาบิตตรวจสอบได้นั้น จะต้องทำการแบ่งข้อมูลออกเป็นเซ็กเมนต์ก่อน โดยแต่ละเซ็กเมนต์จะมีข้อมูล n บิต (ปกติจะแบ่งเป็นเซ็กเมนต์ละ 16 บิต)
- จากนั้นนำบิตข้อมูลของทุกเซ็กเมนต์มาบวกกันด้วยวิธีการแบบ 1’s complement ซึ่งจะได้ผลลัพธ์ออกมา n บิต
- เมื่อได้ผลลัพธ์จากการบวกแล้วจะนำมาทำคอมพลีเมนต์หรือทำการกลับบิตให้เป็นตรงกันข้าม (0 เป็น 1 , 1 เป็น 0)
- ผลที่ได้จากการทำคอมพลีเมนต์คือบิตตรวจสอบ ที่จะต้องนำไปต่อไว้ส่วนท้ายของบิตข้อมูล ก่อนที่จะส่งออกไป

Checksum

- สรุปการหาบิตตรวจสอบสำหรับวิธีการตรวจสอบแบบ checksum

  1. ข้อมูลจะถูกแบ่งออกเป็นเซ็กเมนต์ย่อยๆ เซ็กเมนต์ละ n บิต
  2. นำข้อมูลของทุกเซ็กเมนต์มาบวกกันด้วยวิธีการแบบ 1’s complement
  3. นำผลรวมของทุกเซ็กเมนต์มาทำคอมพลีเมนต์
  4. ส่งบิตตรวจสอบไปพร้อมกับข้อมูล

- การตรวจสอบความผิดพลาดของข้อมูล

  1. รวบรวมข้อมูลแต่ละเซ็กเมนต์ ๆ ละ n บิต
  2. นำข้อมูลของทุกเซ็กเมนต์มาบวกกันด้วยวิธีการแบบ 1’s complement
  3. นำผลรวมของทุกเซ็กเมนต์มาทำคอมพลีเมนต์
  4. ถ้าผลลัพธ์ที่ได้เท่ากับ 0 แสดงว่าข้อมูลชุดนั้นถูกต้อง

     แต่ถ้าผลลัพธ์ที่ได้ไม่เท่ากับ 0 แสดงว่าข้อมูลชุดนั้นมีความผิดพลาด

Checksum

- **ตัวอย่าง ฝั่งส่ง**

  สมมติว่าต้องการที่จะส่งบิตข้อมูลจำนวน 16 บิตออกไป และใช้วิธีการตรวจสอบแบบ checksum โดยมีบิตตรวจสอบ 8 บิต ซึ่งบิตข้อมูลที่ต้องการส่งมีดังนี้

10101001 00111001

วิธีทำ

นำข้อมูลของทุกเซ็กเมนต์มาบวกกันด้วยวิธีแบบ 1's complement ได้ดังนี้

	10101001
	00111001
sum	11100010
Checksum	00011101

10101001 00111001 00011101

ข้อมูลและบิตตรวจสอบที่จะส่งออกไป

Checksum

- **ตัวอย่าง ฝั่งรับ**

  จากตัวอย่างการส่งที่ผ่านมาเมื่อฝั่งรับได้รับข้อมูลดังข้างล่างนี้ จงตรวจสอบข้อมูลดังกล่าวว่าถูกต้องหรือไม่

  10101001 00111001 00011101

  วิธีทำ นำข้อมูลของทุกเซ็กเมนต์มาบวกกันด้วยวิธีแบบ 1's complement ได้ดังนี้

  10101001

  00111001

  00011101

  sum 11111111

  complement 00000000 ✓

Flow control

- หน้าที่หลักของ Data link layer
- การควบคุมการไหลของข้อมูล หรือ Flow control คือ เป็นขั้นตอนกระบวนการที่ควบคุมจำนวนของข้อมูลที่ถูกส่งออกไปให้อยู่ในปริมาณที่เหมาะสม ก่อนที่จะได้รับการตอบรับยืนยันจากผู้รับข้อมูล นั่นหมายความว่าข้อมูลที่ถูกส่งไปนั้นจะต้องถูกจำกัดเอาไว้จำนวนหนึ่ง ถ้ายังไม่ได้รับการตอบรับจากผู้รับ ว่าได้รับข้อมูลชุดนั้นๆแล้ว ผู้ส่งจะต้องไม่ส่งข้อมูลชุดถัดไป
  - สาเหตุที่ต้องมีการควบคุมอัตราการไหลก็เนื่องจากว่า ผู้รับอาจจะมีความเร็วในการรับข้อมูลไม่เท่ากับผู้ส่ง หรือมีหน่วยความจำที่ใช้ในการเก็บข้อมูลอย่างจำกัด หรือมีความเร็วในการประมวลผลต่ำ

  ดังนั้นการควบคุมอัตราการไหลของข้อมูลจึงมีความจำเป็นอย่างมาก ถ้าผู้รับยังไม่พร้อมที่จะรับข้อมูลในขณะนั้น ผู้ส่งจะต้องทำการหยุดส่งชั่วคราวก่อน หรือถ้าผู้รับมีความเร็วในการประมวลผลต่ำ ผู้ส่งจะต้องส่งข้อมูลด้วยอัตราเร็วที่เหมาะสม เพื่อให้เกิดความสมดุลกันในการรับส่งข้อมูล

Error control

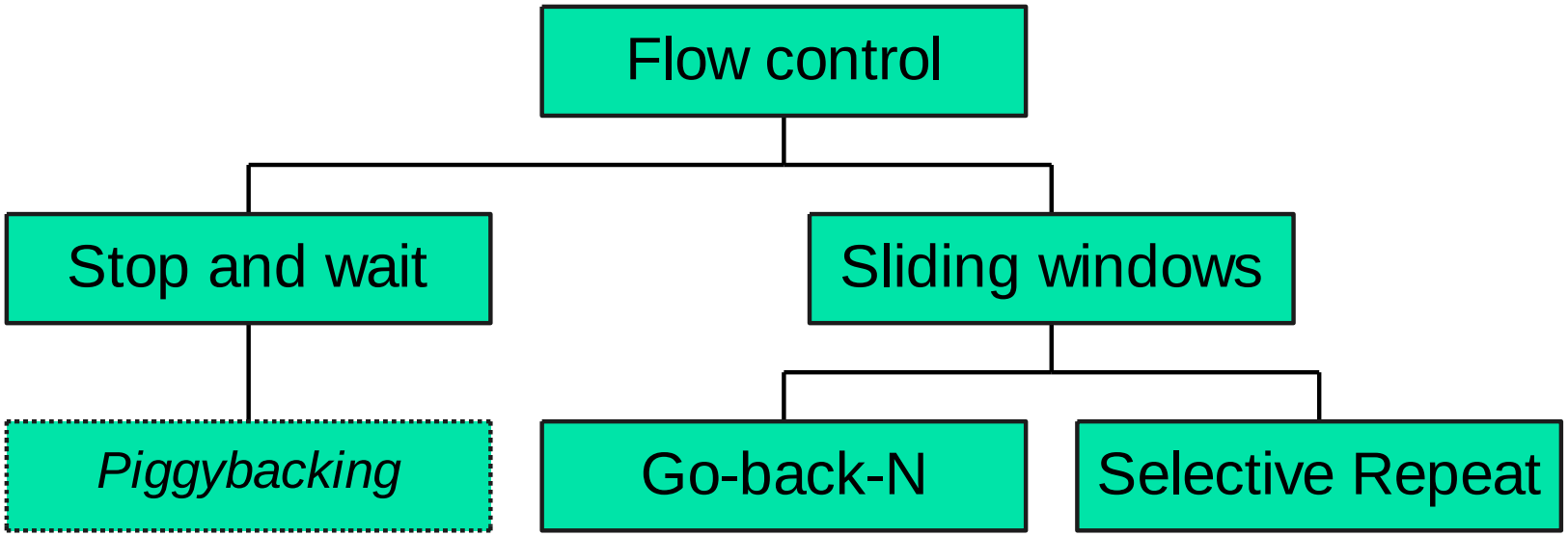
- หน้าที่หลักของ Data link layer
- การควบคุมข้อผิดพลาดของข้อมูล หรือ error control คือ การที่ผู้ส่งต้องส่งข้อมูลไปใหม่อีกครั้งหนึ่ง ถ้าผู้รับไม่สามารถรับข้อมูลหรือได้รับข้อมูลที่ไม่ถูกต้อง
  - สาเหตุที่ต้องมีการควบคุมก็เนื่องจากว่า ข้อมูลจะต้องเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง จึงมีความเป็นไปได้ที่ข้อมูลชุดนั้นจะเกิดสูญหายหรือเสียหายในระหว่างการเดินทางได้

ดังนั้นผู้รับจะต้องมีกระบวนการในการตรวจสอบความผิดพลาดของข้อมูล นอกจากนั้นจะต้องมีการส่งข้อความไปบอกกับผู้ส่งข้อมูลด้วยว่าข้อมูลชุดไหนที่ได้รับมามีความผิดพลาด ซึ่งฝ่ายส่งจะต้องทำการส่งข้อมูลชุดนั้นๆไปใหม่อีกครั้งหนึ่ง

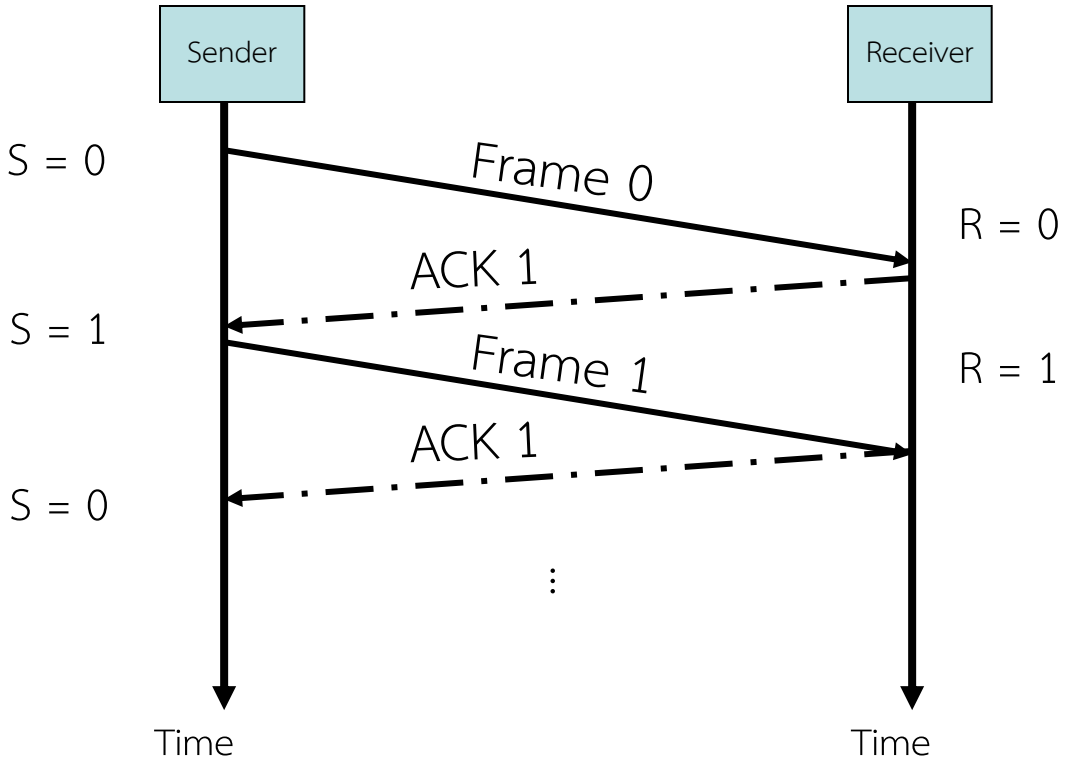
Flow control

- กลไกในการควบคุมอัตราการไหลและควบคุมความผิดพลาดของข้อมูล
  - Stop-and-wait
  - Go-Back-N
  - Selective-Repeat

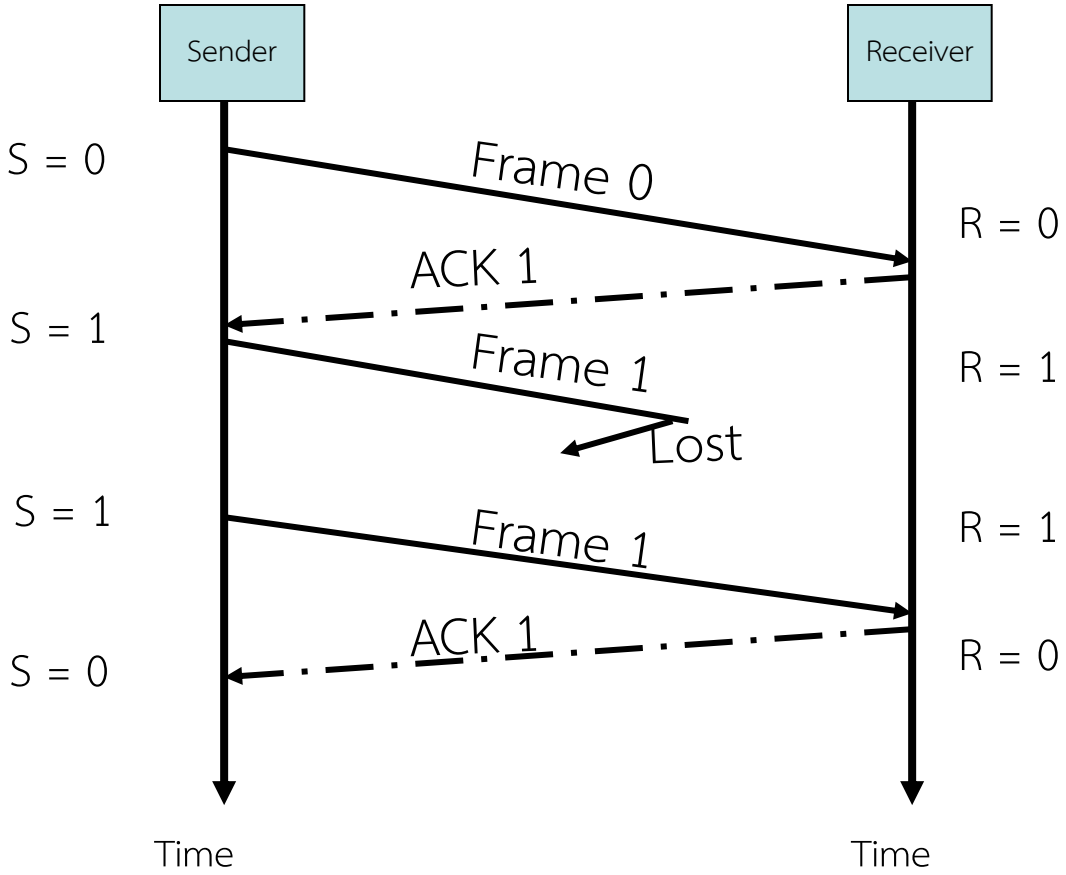
Flow control



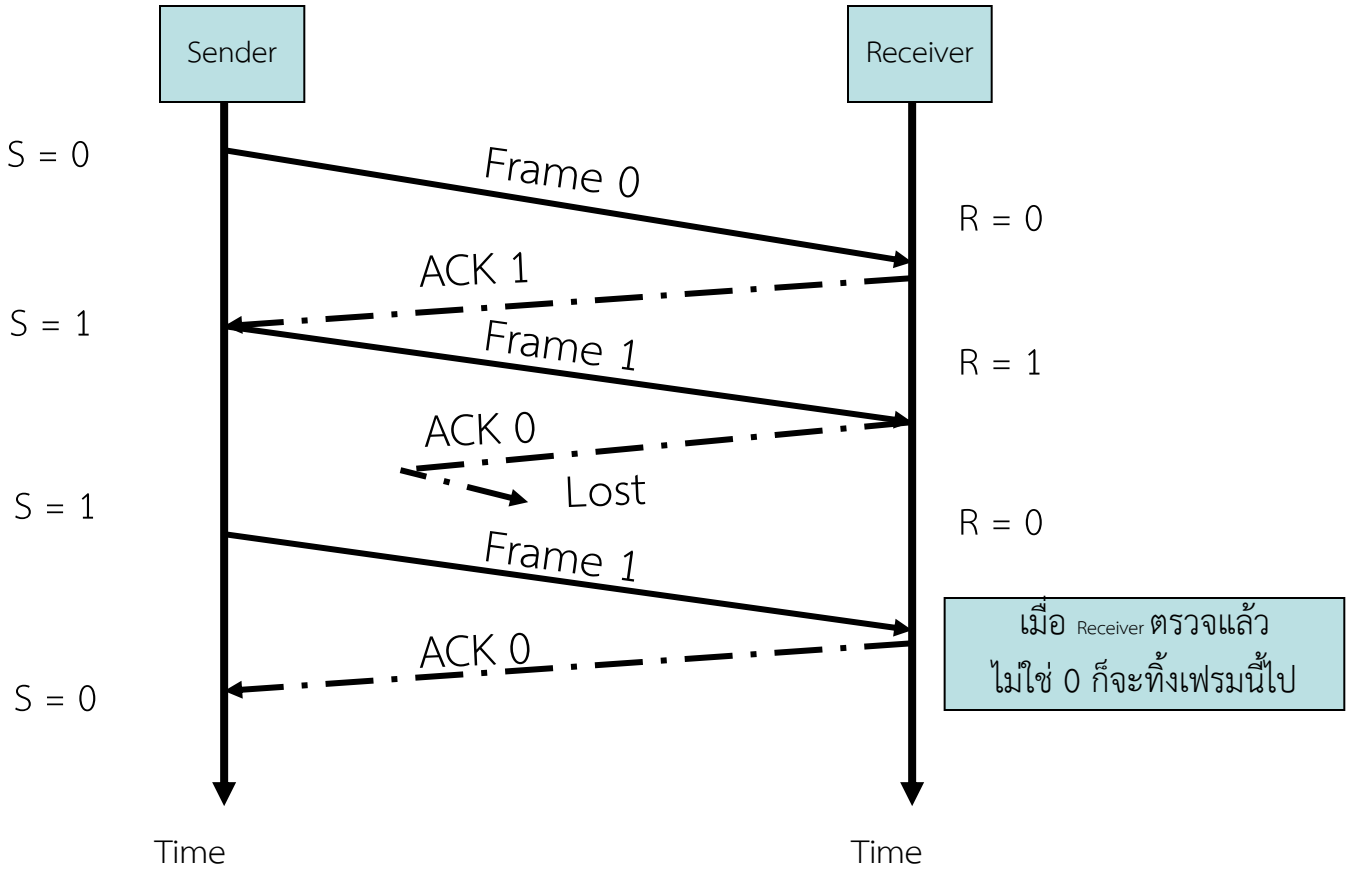
Stop-and-wait 1.กรณีปกติ



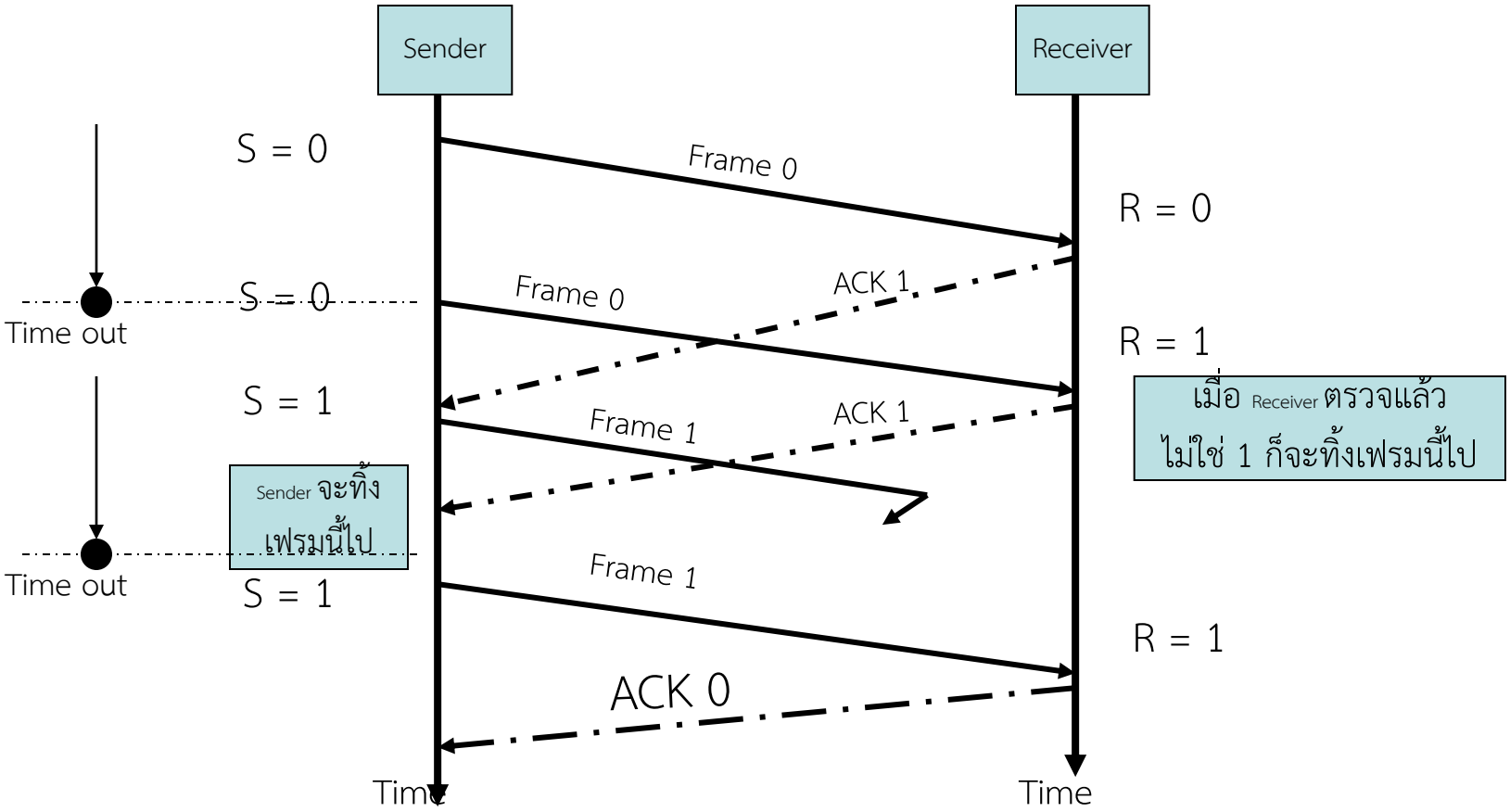
Stop-and-wait 2.กรณีเฟรมข้อมูลสูญหายหรือเสียหาย



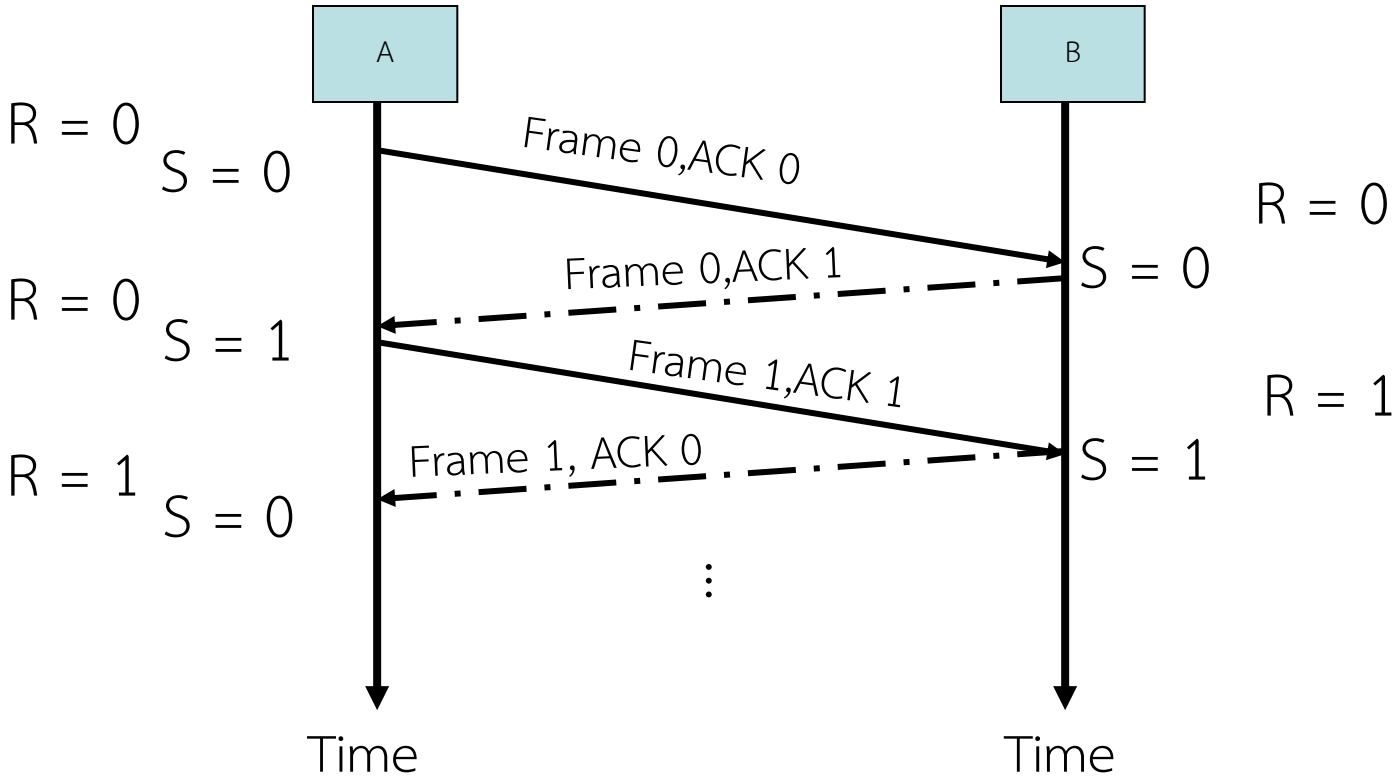
Stop-and-wait 3.กรณีเฟรม ACKสูญหายหรือเสียหา



Stop-and-wait 4.กรณีเฟรม ACK ล่าช้า



Stop-and-wait การส่งข้อมูลแบบสองทาง Piggybacking



Sliding Window Go-back-*N*

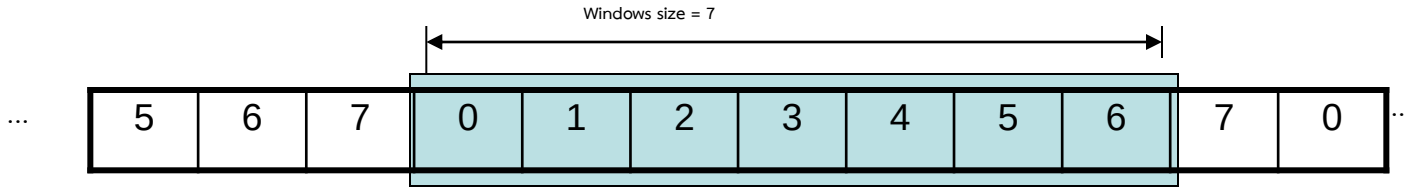
•กลไก Go-back-N จะทำการส่งเฟรมข้อมูลจำนวน w เฟรม ก่อนที่จะได้รับ ack ดังนั้นผู้ส่งจะต้องทำการสำเนาเฟรมข้อมูลทั้ง w เฟรมเอาไว้ด้วย และเฮดเดอร์แต่ละเฟรมจะมีหมายเลขลำดับด้วย โดยที่จะเริ่มจาก 0 เป็นต้นไป และเริ่มจากซ้ายไปขวา

•**การเลื่อนหน้าต่างของผู้ส่งข้อมูล** : ผู้ส่งข้อมูลจะต้องทำการเก็บเฟรมข้อมูลเอาไว้ในบัฟเฟอร์ก่อน จนกระทั่งได้รับเฟรม ack กลับมา จึงจะทำการลบเฟรมข้อมูลนั้นทิ้งได้ ดังนั้นหลักการของการ **เลื่อนหน้าต่าง** (sliding window) จะเสมือนกับการมีหน้าต่างมาครอบเฟรมข้อมูลไว้ แล้วใช้การเลื่อนหน้าต่างไปมา เฟรมข้อมูลที่อยู่ด้านซ้ายของหน้าต่างจะหมายถึงเฟรมที่ได้รับ ack แล้ว สามารถที่จะลบทิ้งเฟรมนั้นออกจากบัฟเฟอร์ได้ ส่วนเฟรมที่อยู่ด้านขวาของหน้าต่าง เป็นเฟรมที่ยังไม่ได้มีการส่งออกไป จะต้องรอจนกระทั่งหน้าต่างเลื่อนมาถึงจึงจะสามารถส่งเฟรมข้อมูลได้

Sliding Window Go-back-*N*

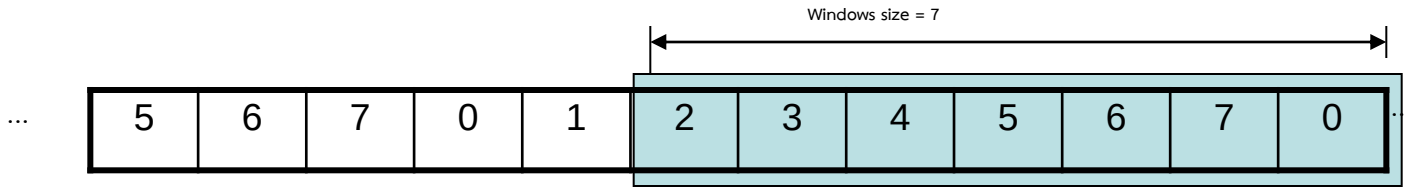
การเลื่อนหน้าต่างของผู้ส่งข้อมูล :

ก่อนเลื่อนหน้าต่าง



จากตัวอย่าง เฟรม 0 ถึง 6 ได้ถูกส่งออกไปและกำลังรอเฟรม ack ตอบกลับมาจากผู้รับ

หลังเลื่อนหน้าต่าง 2 เฟรม



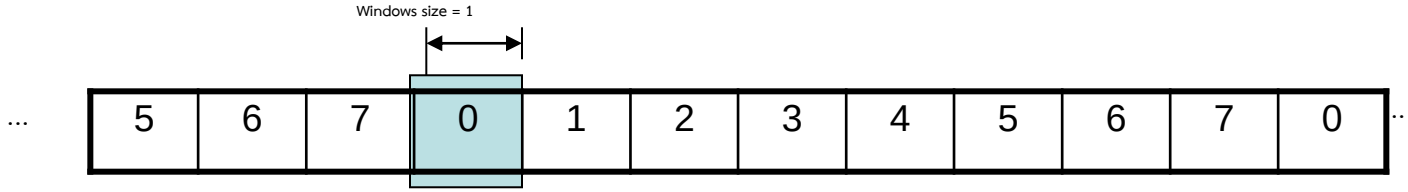
และเมื่อผู้ส่งได้รับเฟรม ack ตอบกลับมาแล้วว่าผู้รับได้รับเฟรม 0 และเฟรม 1 เรียบร้อยแล้ว ผู้ส่งจะทำการเลื่อนหน้าต่างไปทางด้านขวา 2 เฟรม

Sliding Window Go-back-*N*

การเลื่อนหน้าต่างของผู้รับข้อมูล :

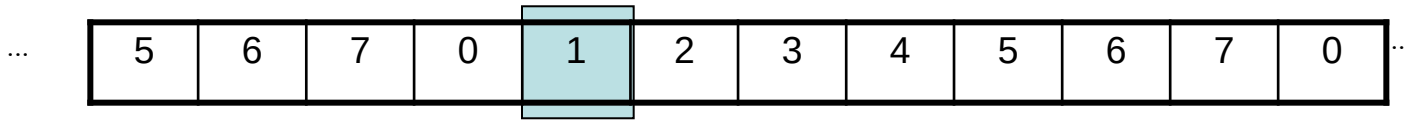
ขนาดหน้าต่างของผู้รับจะมีค่าเท่ากับ 1 เสมอ หมายความว่าหน้าต่างของผู้รับจะสามารถครอบเฟรมข้อมูลได้เพียงเฟรมเดียว ดังนั้นเฟรมที่ถูกหน้าต่างครอบไว้คือเฟรมที่ต้องการรับและยังไม่ได้รับ

ก่อนเลื่อนหน้าต่าง



จากตัวอย่าง หมายความว่าผู้รับต้องการจะรับเฟรม 0

หลังเลื่อนหน้าต่าง



เมื่อได้รับเฟรม 0 เรียบร้อยแล้ว จึงจะทำการเลื่อนหน้าต่างไปทางด้านขวา เพื่อบ่งบอกว่าเฟรมถัดไปที่ต้องการรับคือเฟรม 1

Sender A

Receiver B