

Lecture 6

Reliability and Channel Coding



Topics in this Lecture


- **error** effect
- **error detection** and correction
- basic error detection coding
 - **Simple parity checking**
 - **Internet checksum**
 - **CRC code**
- ARQ and FEC
- **stop-and-wait protocol(ARQ)**
- Chapter 8




Internet Banking

- Postgiro
- Bankgior
- OCR

Mottagare

-- Välj ur listan --  [Redigera mottagare](#)


Ny mottagare - PG/BG-nummer

 **Spara mottagare**

Ange PG/BG-nummer med bindestreck

Detaljer

Belopp

Betaldatum 

Lämnas blank för betalning snarast möjligt.

☒ **OCR-nummer**

☐ **Meddelande (om inte OCR-nummer finns)**

(max 70 tecken)



OCR-Number (OCR-nummer) ?

- A reference number links your payment to the invoice
- Typed OCR in-correctly ? Error detection?
- E24.se reported (how it is possible?)
 - Företagaren Thomas Hultberg fyllde i fel OCR-nummer när han skulle betala in skatten. Nu tvingas han betala in de **229 000 kronorna** en gång till.



The Three Main Sources of Transmission Errors

- All data communications systems are susceptible to errors
- There are three main categories of transmission errors:

1. Interference



- electromagnetic radiation emitted from devices such as electric motors and background cosmic radiation

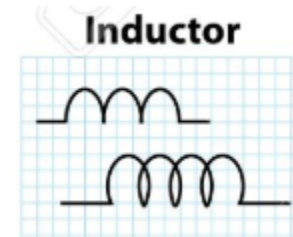
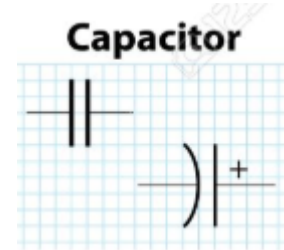
Interference can cause noise that can disturb radio transmissions and signals traveling across wires.



The Three Main Sources of Transmission Errors

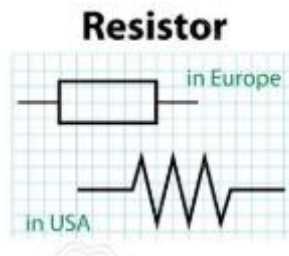
2. Distortion

- All physical systems distort signals
- Wires have properties of **capacitance** and **inductance**
 - that block signals at some frequencies while admitting signals at other frequencies

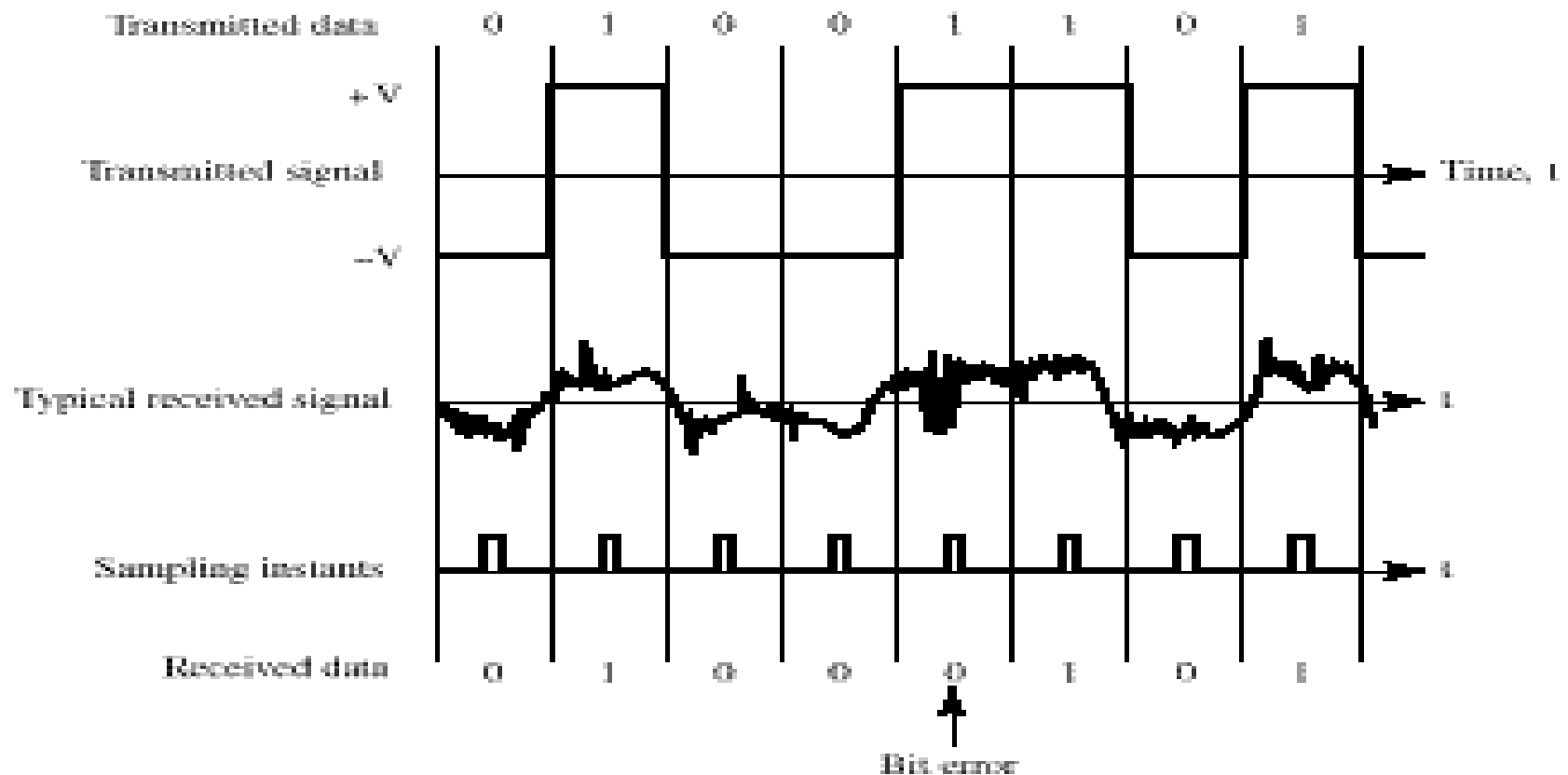


3. Attenuation

- Wires have property of **resistance**
- As a signal passes across a medium, the signal becomes weaker



Errors and Error Effect



Random Errors and Burst Errors

- Random errors: independent errors
- Burst errors: For a burst error, the **burst size**, or **length**, is defined as the number of bits from the start of the corruption to the end of the corruption

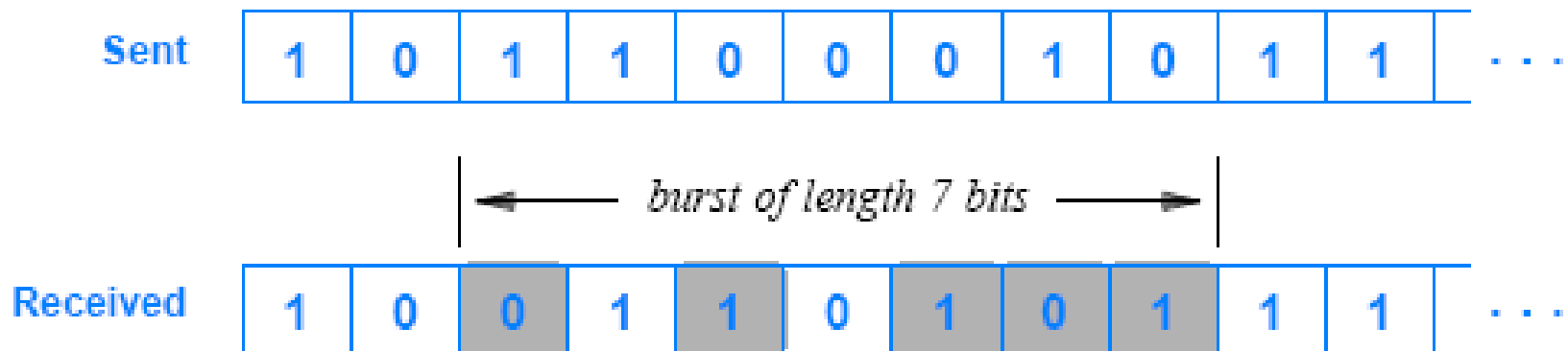


Figure 8.2 Illustration of a burst error with changed bits marked in gray.

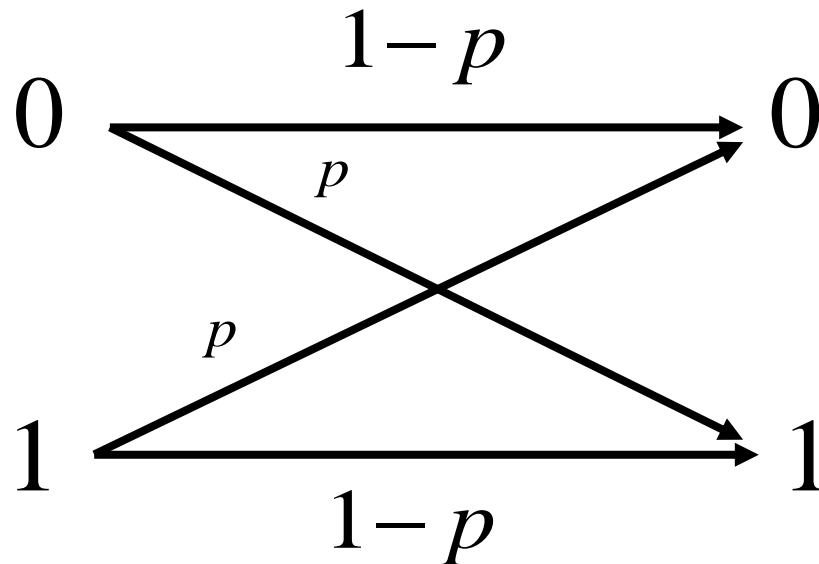
Errors and Error Effect

- Errors
 - $0 \rightarrow 1$ or $1 \rightarrow 0$
 - Bits can be lost
- Error effect
 - downloaded programs from Internet ?
 - CD music ?
 - Internet banking services ?
- Errors must be detected/corrected !!!



Channel model -- BSC

- Binary symmetric channel
 - p : the bit error probability



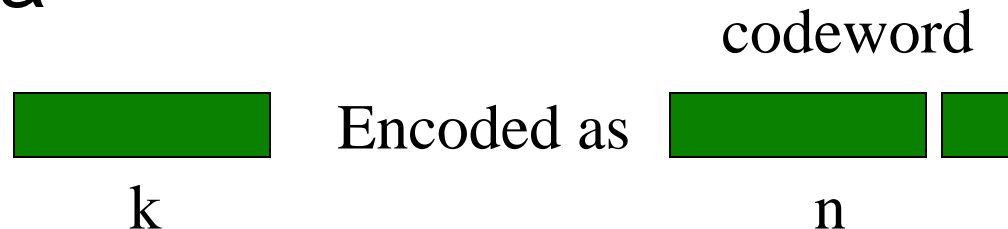
Why Channel Coding ?

- Bit error rate p
 - $p = 1/1000000 = 10^{-5}$ for optical disks
 - $p = 10^{-11}$ for a fiber link
- Some calculations
 - $p = 10^{-6}$
 - download a file of length 10^7 bits → 10 bit errors
 - Data rate at 10 Mbps → 10 bit errors in every 1 sec!!
 - $p = 10^{-11}$, and data rate 10 Gigabits/sec
→ 1 bit error each 10 seconds !



Channel Coding – Principle

- add additional information, or **redundancy** to data



- added by sender, checked by receiver
- k data digits encoded to a codeword of n digits
- Code rate $r = k / n$



Application Example– Swedish personal ID

- **640823-3234 ?**
- **yy mm dd – nnnP**
 - **yy mm dd**– year month day
 - **nnn** – serial number
 - odd– for male, even for female
 - **P** is the parity check digit
 - **Used for error detection !**
- **OCR number uses the same technique**



Personal ID Encoding Method

position	<u>1</u>	2	<u>3</u>	4	<u>5</u>	6	<u>7</u>	8	<u>9</u>	10
	6	4	0	8	2	3	3	2	3	?
2×odd	12	4	0	8	4	3	6	2	6	
add										
2-digits	3	4	0	8	4	3	6	2	6	
sum =	3+	4+	0+	8+	4+	3+	6+	2+	6=	36
take the last digit of the sum:										6
parity check digit =	10	-	6	=	4					
→	640823-3234									



Personal ID Error Detection

640823-3234 → 460823-3234 ?

position	<u>1</u>	2	<u>3</u>	4	<u>5</u>	6	<u>7</u>	8	<u>9</u>	10
	4	6	0	8	2	3	3	2	3	?

2×odd	8	6	0	8	4	3	6	2	6	
--------------	---	---	---	---	---	---	---	---	---	--

add

2-digits	8	6	0	8	4	3	6	2	6	
-----------------	---	---	---	---	---	---	---	---	---	--

sum = $8 + 6 + 0 + 8 + 4 + 3 + 6 + 2 + 6 = 43$

take the **last digit** of the sum: **3**

parity check digit = $10 - 3 = 7$

→ It is not equal to 4 → Error in the number !



Parity Check Applications

- Räkna ut sista siffran i personnumret- kalkylator, räkna ut, beräkna (kalkyleramera.se)
- The same coding methods have been used for
 - OCR reference number
 - Bankgironummer
 - Organisationsnummer



Two Strategies for Handling Channel Errors

- two broad categories:
 1. Forward Error Correction (**FEC**) mechanisms
 2. Automatic Repeat reQuest (**ARQ**) mechanisms
- FEC
 - Used in storage systems, space communications
 - The error correcting code used for correcting errors
- ARQ
 - Used in data communications
 - Error detecting code used for detecting errors
 - Protocol is used for error correction



Forward Error Correction (**FEC**)

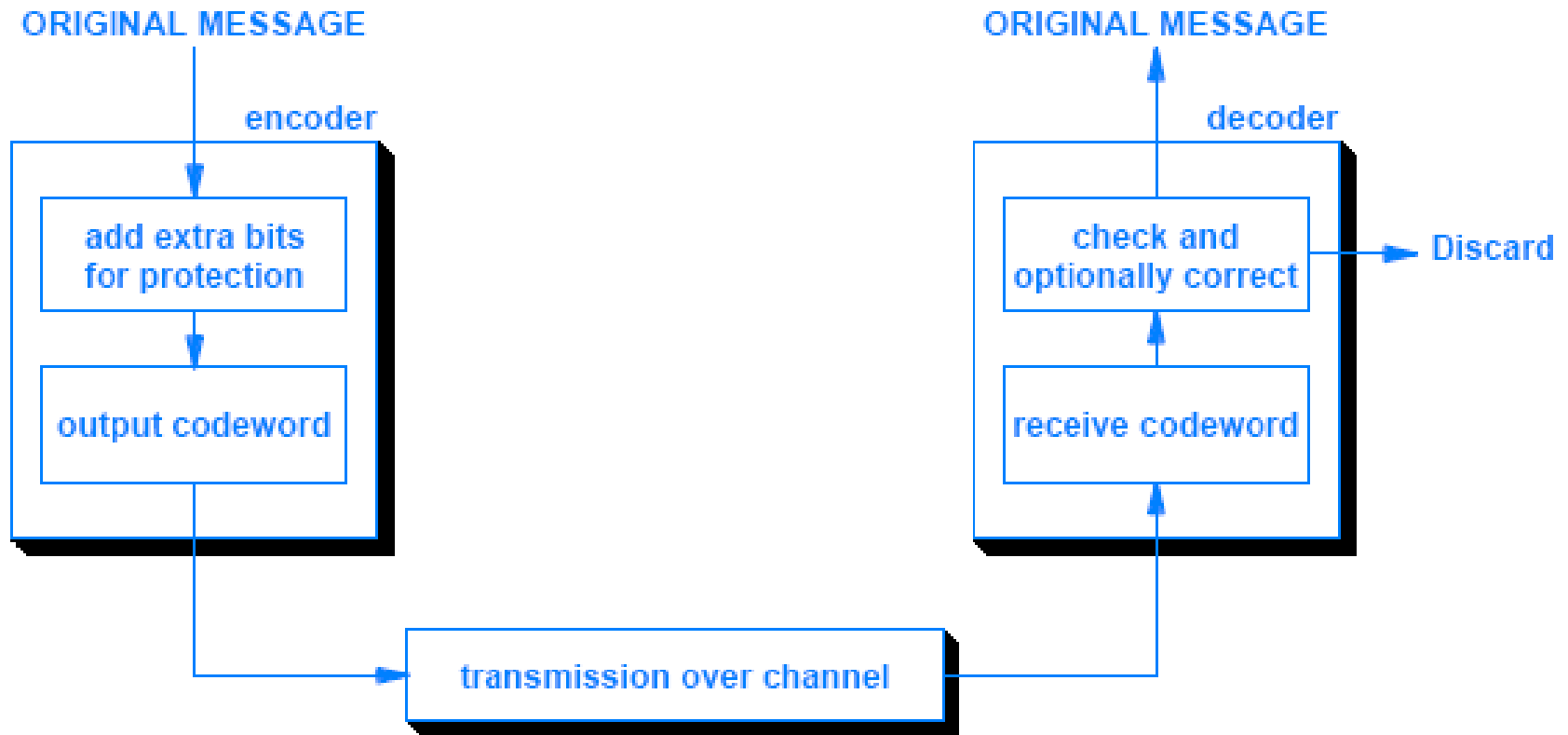


Figure 8.3 The conceptual organization of a forward error correction mechanism.



Example 1

Single Parity Checking (SPC)

- Given k info bits, attach 1 parity check bit
- Parity check bit can be even parity or odd parity
 - Total number of 1's is even or odd



An Example Block Error Code: Single Parity Checking

Original Data	Even Parity	Odd Parity
0 0 0 0 0 0 0 0	0	1
0 1 0 1 1 0 1 1	1	0
0 1 0 1 0 1 0 1	0	1
1 1 1 1 1 1 1 1	0	1
1 0 0 0 0 0 0 0	1	0
0 1 0 0 1 0 0 1	1	0

Figure 8.4 Data bytes and the corresponding value of a single parity bit when using even parity or odd parity.

SPC capability

- SPC can only detect errors
- SPC can only detect errors where an **odd number** of bits are changed
 - If one of the nine bits (including the parity bit) is changed during transmission, the receiver will declare that the incoming byte is **invalid**
 - However, if **two, four, six, or eight** bits change value, the receiver will incorrectly classify the incoming byte as **valid**



Example 2: The 16-Bit Checksum Used in the Internet

- A particular coding scheme plays a key role in the Internet
 - Known as the **Internet checksum**, the code consists of a **16-bit 1's complement** checksum
- The Internet checksum does not impose a fixed size on data to be checked
 - the algorithm allows a message to be arbitrarily long
 - and computes a checksum over the entire message
- The Internet checksum treats data in a message as a series of 16-bit integers.



The 16-Bit Checksum Used on the Internet

- Known as the **Internet checksum**, the code consists of a **16-bit 1's complement** checksum
- The Internet checksum treats data in a message as a series of 16-bit integers.



The 16-Bit Checksum Implementation

- Software implementation
 1. 1's complement sum of 16-bit integers
 2. 1's complement of the sum
- Reference
 - [RFC 1071 - Computing the Internet checksum](#)



Example 3 Cyclic Redundancy Codes (CRCs)

- A form of channel coding known as a Cyclic Redundancy Code (CRC) is used in high-speed data networks
- Excellent error detection
- Fast hardware implementation



CRC encoding

- CRC Encoding

Given **k** information bits, add $n - k$ parity check bits, obtain a codeword of n bits



Cyclic Redundancy Codes (CRC)

- Mathematicians
 - explain a CRC computation as the remainder from a division of two **polynomials** with binary **coefficients**
 - one representing the message and another representing a fixed **divisor**
- Theoretical Computer Scientists
 - explain a CRC computation as the **remainder** from a division of two binary numbers
 - one representing the message and the other representing a fixed divisor



Binary number and polynomial

- A binary number : sequence of 0/1
- k bits of message: $m_{k-1} \dots m_1 m_0$
- Binary polynomial: coefficients are 0 or 1
- Map binary number and binary polynomial

$$m_{k-1} \dots m_1 m_0 \Leftrightarrow m_{k-1} x^{k-1} + \dots + m_1 x + m_0$$

where $m_i = 0$ or 1 , $i = 0, 1, \dots, k-1$

Example: $110001 \Leftrightarrow x^5 + x^4 + 1$



CRC Encoding

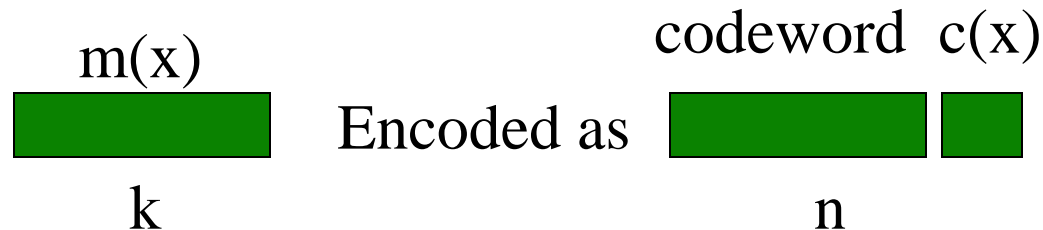
- k bits information is mapped to $m(x)$:

$$m_{k-1} \dots m_1 m_0 \Leftrightarrow m_{k-1} x^{k-1} + \dots + m_1 x + m_0 = m(x)$$

- **generator polynomial $g(x)$** is defined as

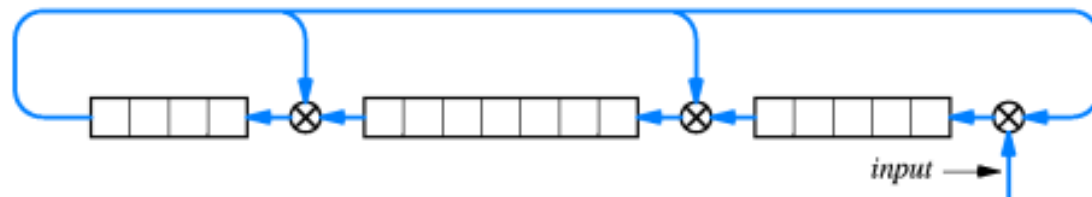
$$g(x) = x^{n-k} + g_{n-k-1} x^{n-k-1} \dots + g_1 x + 1 \quad \text{with } g_i = 0/1$$

- **Codeword** $c(x) = m(x)x^{n-k} + p(x) \equiv 0 \pmod{g(x)}$
- **$p(x) \equiv m(x)x^{n-k} \pmod{g(x)}$**



CRC hardware implementation

- Encoding/decoding implemented as hardware
 - Binary operation
 - Exclusive-OR (xor)
 - Shift-register
 - After all info bits are shifted into the unit, the registers contain the 16-bit CRC parity check bits
- Example: $g(x) = x^{16} + x^{12} + x^5 + 1$



CRC Decoding

- **Decoding(error detection)**

Received codeword $r(x)$

if $(r(x) \equiv 0 \bmod g(x))$, then most likely no errors in $r(x)$

otherwise, there must be errors!!!

- **Reference**

The CRC Pitstop

<http://ross.net/crc/index.html>



Cyclic Redundancy Codes (CRC)

Given message of 1010, and $g(x) = x^3 + x + 1$

$k = 4$, $n - k = 3$, $n = 7$

Using program crc.zip to check the encoding/decoding !

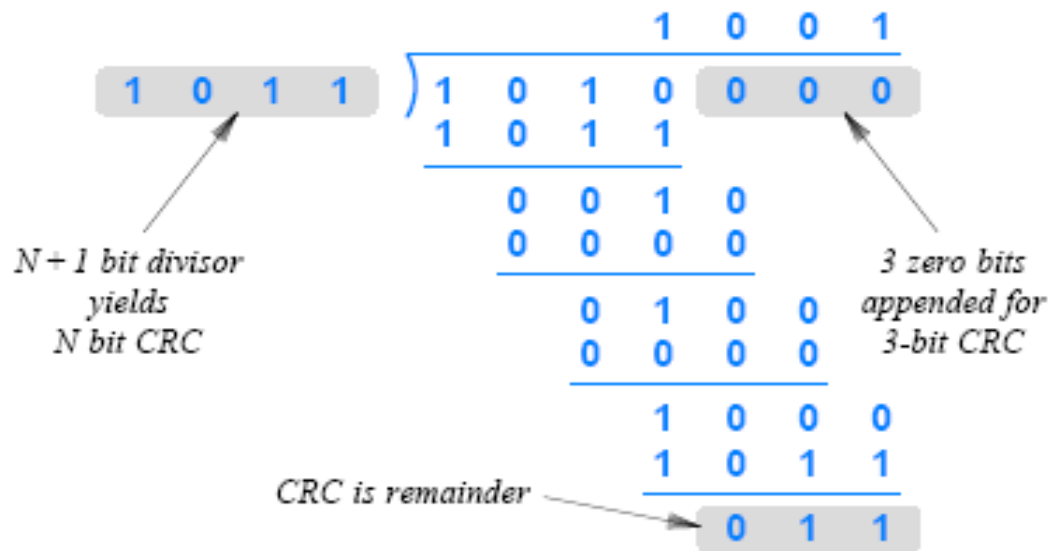


Figure 8.12 Illustration of a CRC computation viewed as the remainder of a binary division with no carries.

CRC Error Detecting Capability

- Received codeword
 - $r(x) = c(x) + e(x)$ $e(x)$ -- error polynomial
 - $r(x) \equiv e(x) \pmod{g(x)}$

1. Detect all single errors

$e(x) = x^i$ which is not divisible by $g(x)$

2. Detect all double errors

$e(x) = x^i + x^j = x^i (1 + x^{j-i})$ and

$g(x)$: $(1 + x^t)$ is not divisible by $g(x)$ for small t



CRC Error Detecting Capability(2)

3. Detect all odd no. of errors if $g(x) = (1+x)f(x)$
4. Detect any burst errors of length $\leq n-k$
5. Detect $(1 - 2^{-n+k+1}) * 100\%$ burst errors of length $n - k + 1$
6. Detect $(1 - 2^{-n+k}) * 100\%$ burst errors of length $\geq n - k + 2$



Standard CRC Codes

- **Standard CRC Codes**

CRC-12 $g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$

CRC-16 $g(x) = x^{16} + x^{15} + x^2 + 1$

CRC-CCITT-16 $g(x) = x^{16} + x^{12} + x^5 + 1$

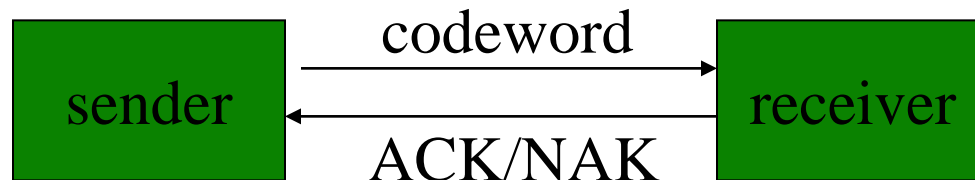
- **Example of Error Detection Capability**

CRC-16 can detect 100% of 1, 2, or odd no. of errors, burst errors of length up to 16; 99.997% of burst errors of length 17 and 99.998% of burst errors of length > 17 .



Automatic Repeat reQuest (ARQ) Mechanisms

- Based on error-detection codes (like CRC)
- Use protocol to correct errors
- Positive **acknowledgement** (ACK) message
- Negative acknowledgement (NAK) message
- Used in data communications and computer networks



ARQ Protocols

- **How to Realize Reliable Transmission**

- (1) error detection coding

- (2) frame numbering to avoid duplicates

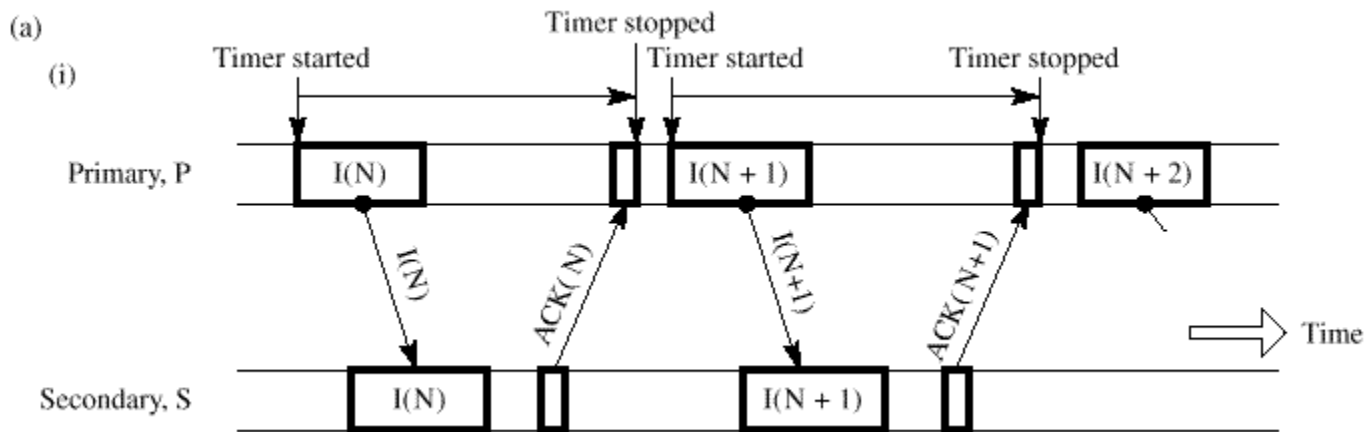
- (3) timer to detect frame lost

- (4) flow control to avoid overflow



Simplex Stop-and-Wait Protocol for Noisy Channel

- Simplex: Data flow in one direction
Noisy channel: need error control coding
Finite buffer: need feedback control(ACK)
- Principle



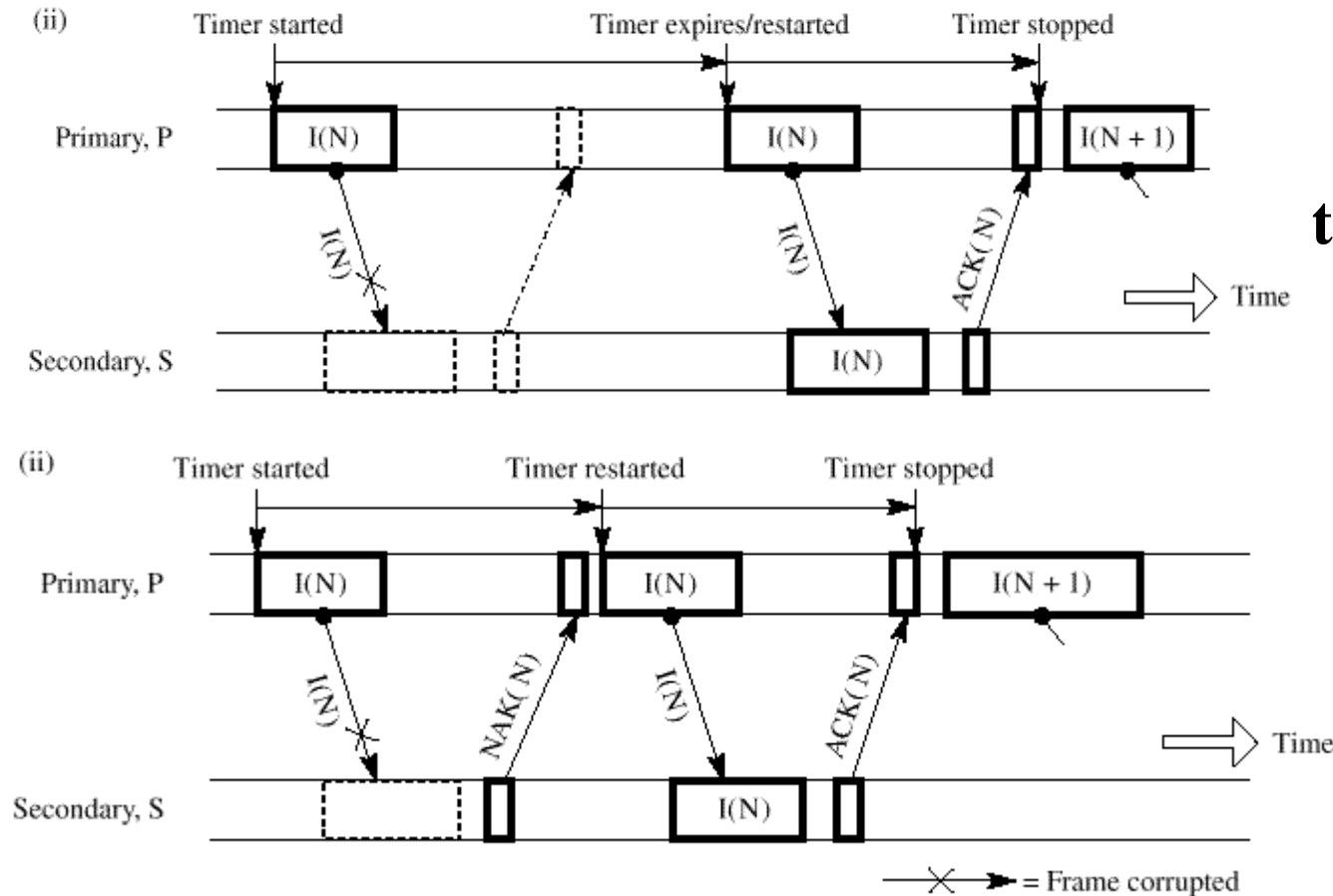
No errors



Simplex Stop-and-Wait Protocol for Noisy Channel

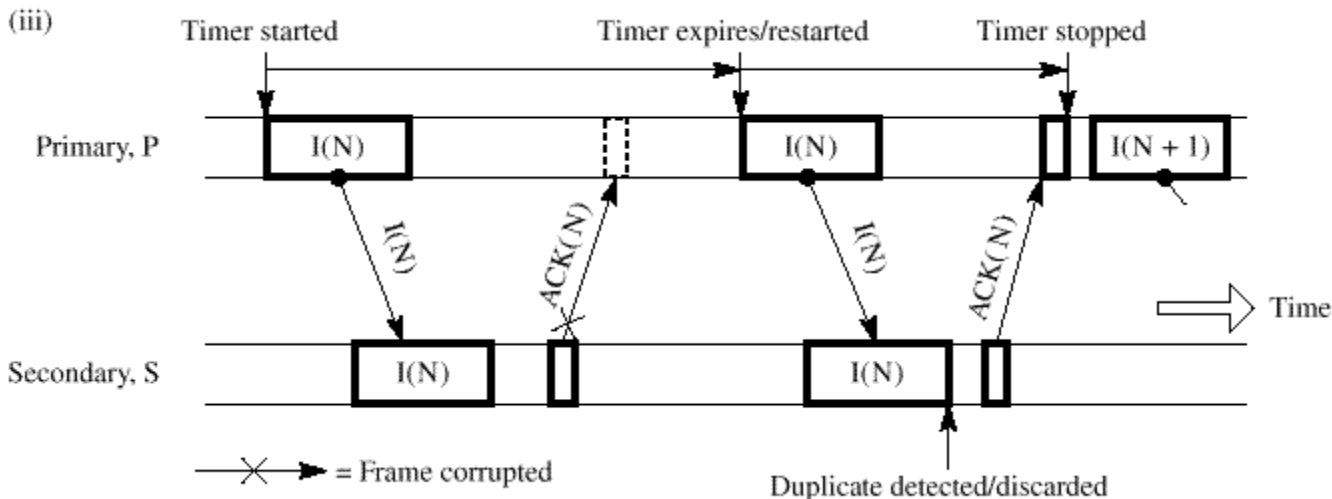
Use timer to retransmit

error frame?
Send NAK



Simplex Stop-and-Wait Protocol for Noisy Channel(3)

- ACK frame lost or in errors
 - Timer used to re-transmit the frame
 - Duplicate detected and discarded



Simplex Stop-and-Wait Protocol for Noisy Channel(4)

- Summary on stop-and-wait protocol
 1. Error detection code used to detect errors
 2. ACK frame to tell the correct receipt
 3. NAK frame to tell the erroneous receipt (optional)
 4. Timer used to start retransmission
 5. Sequence number used to detect duplicates

