Transmission Errors

Error Detection and Correction

Transmission Errors

- Transmission errors are caused by:
  - thermal noise {Shannon}
  - impulse noise (e..g, arcing relays)
  - signal distortion during transmission (attenuation)
  - crosstalk
  - voice amplitude signal compression (companding)
  - quantization noise (PCM)
  - jitter (variations in signal timings)
  - receiver and transmitter out of synch.

Error Detection and Correction

- *error detection* :: adding enough "extra" bits to deduce that there is an error but not enough bits to correct the error.

- If only error detection is employed in a network transmission ➔ retransmission is necessary to recover the frame (data link layer) or the packet (network layer).

- *At the data link layer, this is referred to as* ARQ (Automatic Repeat reQuest).

Error Detection and Correction

- *error correction* :: requires enough additional (redundant) bits to deduce what the correct bits must have been.

Examples

Hamming Codes

FEC = Forward Error Correction *found in MPEG-4.*

Hamming Codes

codeword **::** a legal dataword consisting of *m* data bits and *r* redundant bits.

Error detection involves determining if the received message matches one of the legal codewords.

Hamming distance **::** the number of bit positions in which two bit patterns differ.

Starting with a complete list of legal codewords, we need to find the two codewords whose Hamming distance is the smallest. This determines the Hamming distance of the code.

Error Correcting Codes

| Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Note
Check bits occupy
power of 2 slots

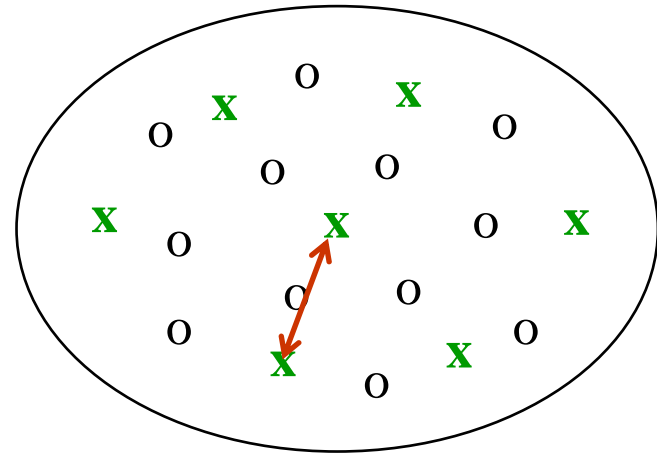Figure 3-7. Use of a Hamming code to correct burst errors.

(a) A code with poor distance properties
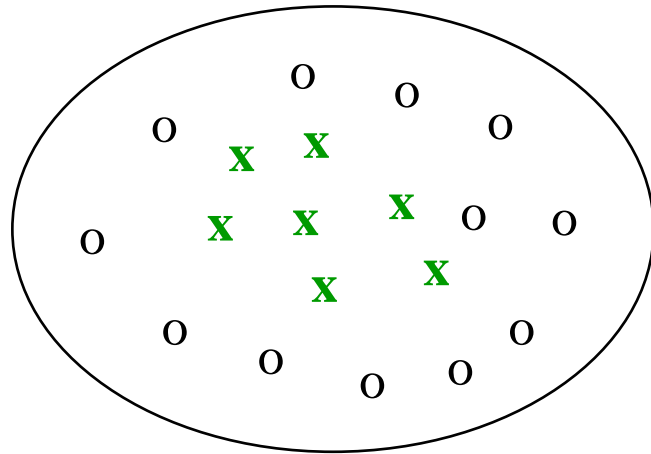
(b) A code with good distance properties

x = codewords   o = **non-codewords**

Figure 3.51

Computer Networks: Transmission Errors

7

Hamming Codes

- To **detect** *d* single bit errors, you need a *d+1* code distance.

- To **correct** *d* single bit errors, you need a *2d+1* code distance.

➔ In general, the price for redundant bits is <u>too expensive</u> to do error correction for network messages.

➔ Network protocols use error detection and ARQ.

Error Detection

*Remember – errors in network transmissions are **bursty**.*

➔ *The percentage of damage due to errors is lower.*

➔ *It is harder to detect and correct network errors.*

- Linear codes
  - Single parity check code :: take k information bits and appends a single **check bit** to form a codeword.
  - Two-dimensional parity checks
- IP Checksum
- Polynomial Codes

    Example: **CRC** (Cyclic Redundancy Checking)

General Error-Detection System

All inputs to channel
satisfy pattern/condition

Channel
output

User
information → **Encoder** → **Channel** → **Pattern Checking** → Deliver user
information
or
set error alarm

Leon-Garcia & Widjaja: *Communication Networks*   Figure 3.49
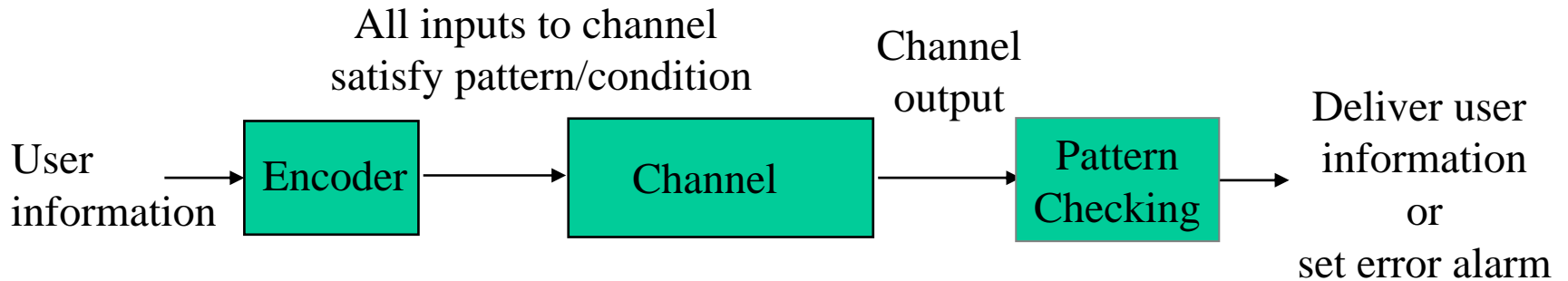
Error-Detection System using Check Bits

Information bits

Received information bits

Calculate check bits

Channel

Recalculate check bits

Compare

Check bits

Received check bits

Information accepted if check bits match

Leon-Garcia & Widjaja: *Communication Networks*

Figure 3.50

Computer Networks: Transmission Errors

11

Two-Dimensional Parity

Parity bits

Data		Parity bits
0101001		1
1101001		0
1011110		1
0001110		1
0110100		1
1011111		0

Parity byte | 1111011 | 0 |
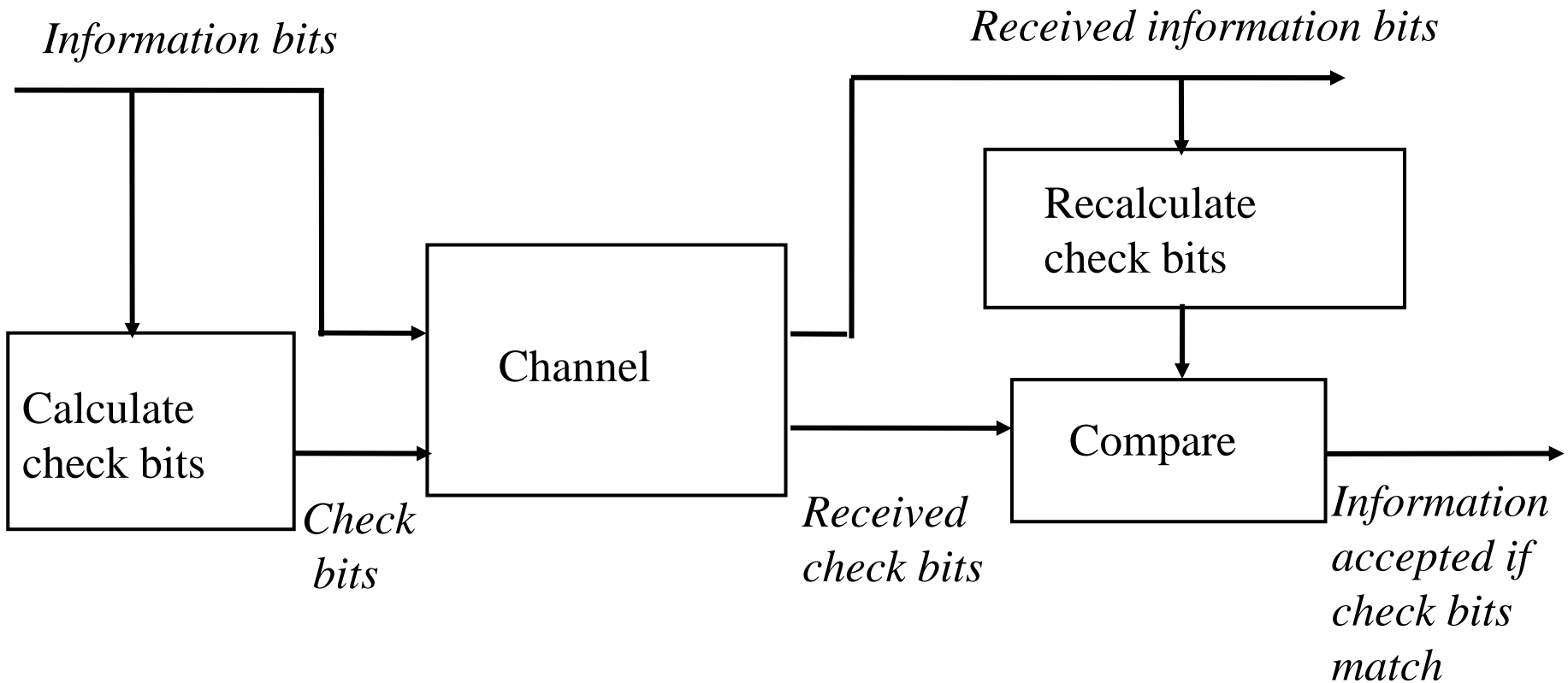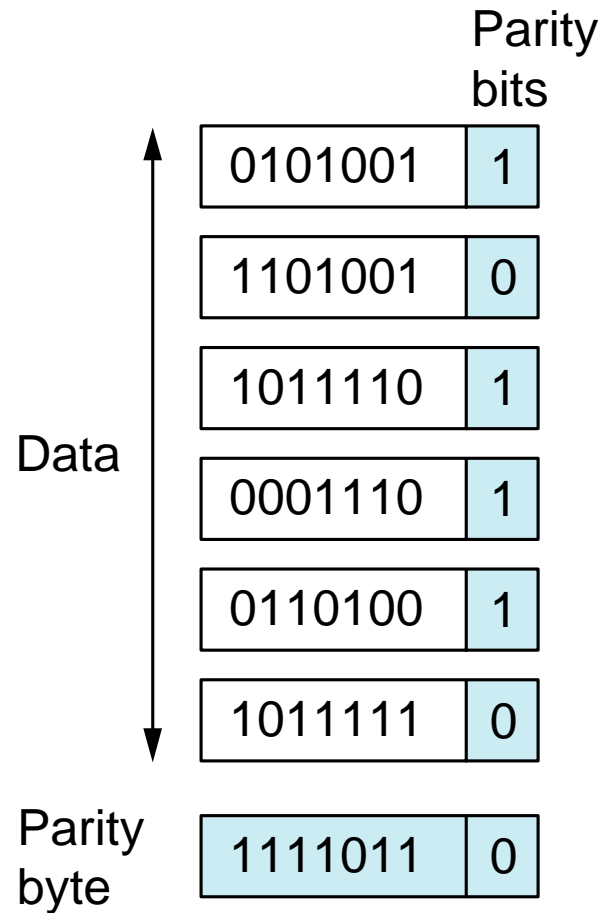
Figure 2.16

Two-Dimensional Parity

```
1  0  0  1  0 | 0

0  1  0  0  0 | 1

1  0  0  1  0 | 0

1  1  0  1  1 | 0
................
1  0  0  1  1 | 1
```

Last column consists of
check bits for each row

Bottom row consists of
check bit for each column

Leon-Garcia & Widjaja: *Communication Networks*

Figure 3.52

Computer Networks: Transmission Errors

13

```
1 0 0 1 0 0
0 (0) 0 0 0 1   ←          One
1 0 0 1 0 0                error
1 1 0 1 1 0
───────────
1 0 0 1 1 1
        ↑
```
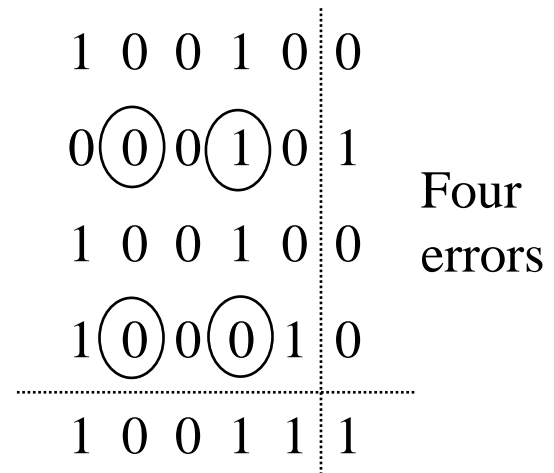
```
1 0 0 1 0 0
0 (0) 0 0 0 1   ←          Two
1 0 0 1 0 0                errors
1 (0) 0 1 1 0   ←
───────────
1 0 0 1 1 1
```

```
1 0 0 1 0 0
0 (0) 0 (1) 0 1            Three
1 0 0 1 0 0                errors
1 (0) 0 1 1 0   ←
───────────
1 0 0 1 1 1
      ↑
```

```
1 0 0 1 0 0
0 (0) 0 (1) 0 1            Four
1 0 0 1 0 0                errors
1 (0) 0 (0) 1 0
───────────
1 0 0 1 1 1
```
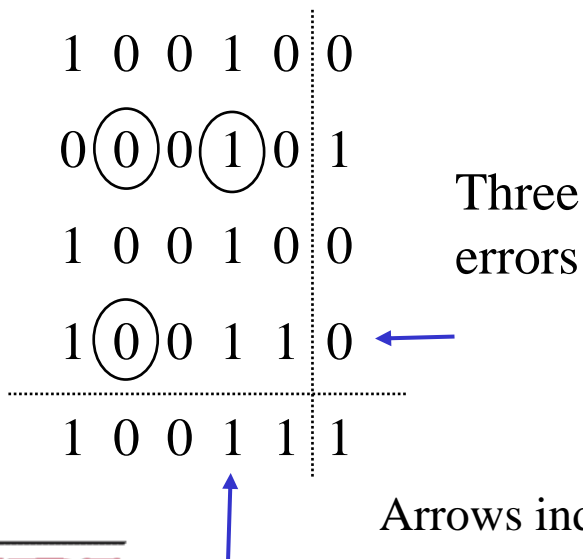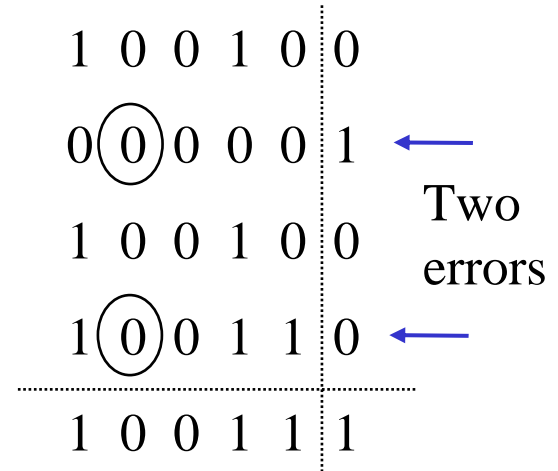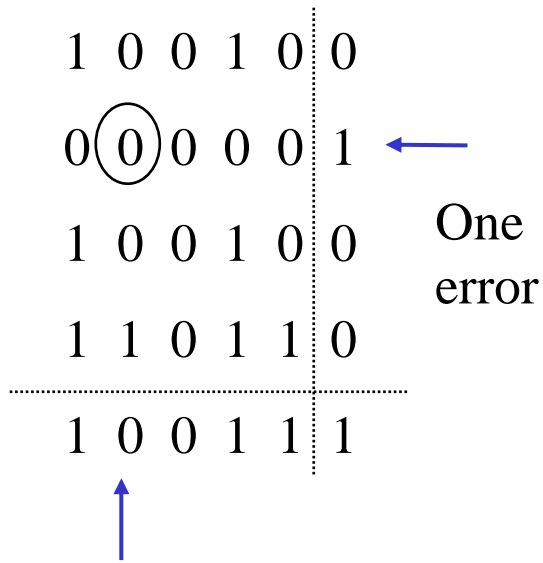
Arrows indicate failed check bits

Computer Networks: Transmission Errors

```c
unsigned short cksum(unsigned short *addr, int count)
  {
     /*Compute Internet Checksum for "count" bytes
      *  beginning at location "addr".
      */
  register long sum = 0;
  while ( count > 1 ) {
      /* This is the inner loop*/
          sum += *addr++;
          count -=2;
      }

      /*  Add left-over byte, if any  */
  if ( count > 0 )
      sum += *addr;

      /* Fold 32-bit sum to 16 bits */
  while (sum >>16)
      sum = (sum & 0xffff) + (sum >> 16) ;

  return ~sum;
}
```

Leon-Garcia & Widjaja: *Communication Networks*

Figure 3.54

Computer Networks: Transmission Errors

Polynomial Codes [LG&W pp. 161-167]

- Used extensively.

- Implemented using `shift-register circuits` for speed advantages.

- Also called CRC (cyclic redundancy checking) because these codes generate check bits.

- **Polynomial codes ::** bit strings are treated as representations of polynomials with ONLY binary coefficients (0's and 1's).

Polynomial Codes

- The *k bits* of a message are regarded as the coefficient list for an information polynomial of degree *k-1*.

$$I :: i(x) = i_{k-1} x^{k-1} + i_{k-2} x^{k-2} + \ldots + i_1 x + i_0$$

Example:

1 0 1 1 0 0 0

$$i(x) = x^6 + x^4 + x^3$$

Polynomial Notation

- Encoding process takes *i(x)* produces a codeword polynomial *b(x)* that contains information bits and additional check bits that satisfy a pattern.

- Let the codeword have *n* bits with *k* information bits and *n-k* check bits.

- We need a *generator polynomial* of degree *n-k* of the form

$$G = g(x) = x^{n-k} + g_{n-k-1} x^{n-k-1} + \dots + g_1 x + 1$$

Note – the first and last coefficient are always 1.

CRC Codeword

k information bits	n-k check bits

n bit codeword
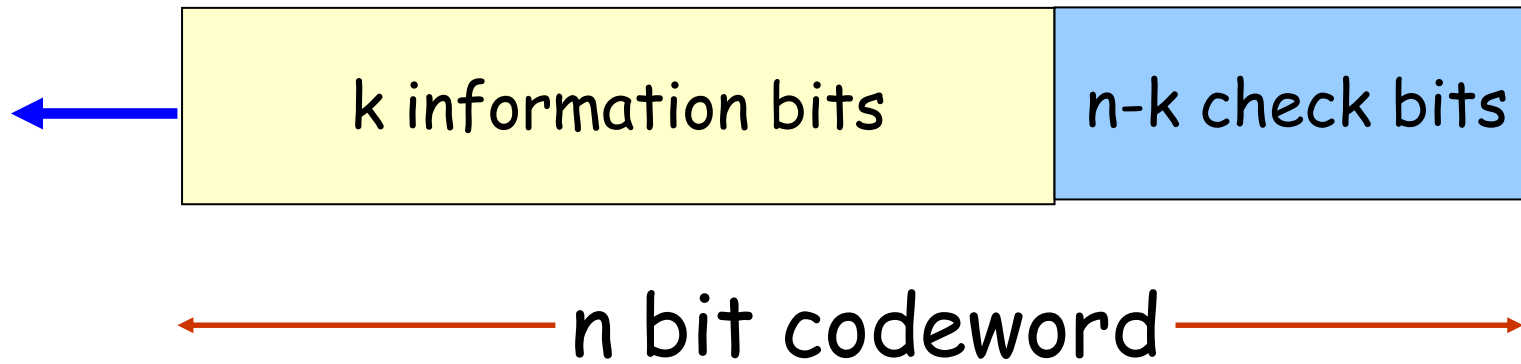
Polynomial Arithmetic

Addition: $(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + (1+1)x^6 + x^5 + 1$
$$= x^7 + x^5 + 1$$

Multiplication: $(x+1)(x^2 + x + 1) = x^3 + x^2 + x + x^2 + x + 1 = x^3 + 1$

Division:

$$
\begin{array}{r}
x^3 + x^2 + x \quad = q(x) \text{ quotient} \\
x^3 + x + 1 \overline{)\ x^6 + x^5 \quad\quad\quad} \text{ dividend} \\
x^6 + \quad x^4 + x^3 \\
\overline{\quad\quad\quad\quad\quad} \\
x^5 + x^4 + x^3 \\
x^5 + \quad x^3 + x^2 \\
\overline{\quad\quad\quad\quad\quad} \\
x^4 + \quad\quad x^2 \\
x^4 + \quad\quad x^2 + x \\
\overline{\quad\quad\quad\quad\quad} \\
x \quad = r(x) \text{ remainder}
\end{array}
$$

divisor

$$
\begin{array}{r}
3 \\
35\ \overline{)\ 122} \\
105 \\
\overline{\quad\quad} \\
17
\end{array}
$$

Leon-Garcia & Widjaja: *Communication Networks*

Figure 3.55

Computer Networks: Transmission Errors

CRC Algorithm
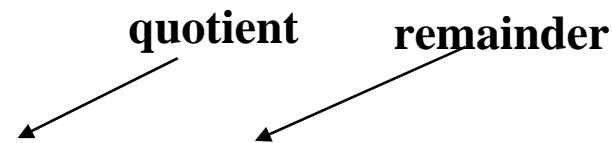
CRC Steps:

1) Multiply $i(x)$ by x^{n-k} (puts zeros in $(n-k)$ low order positions)

2) Divide $x^{n-k} i(x)$ by $g(x)$

quotient   remainder

$$x^{n-k}i(x) = g(x) \, q(x) + r(x)$$

3) Add remainder $r(x)$ to $x^{n-k} i(x)$

   (puts check bits in the $n-k$ low order positions):

$$b(x) = x^{n-k}i(x) + r(x) \quad \longleftarrow \quad \text{transmitted codeword}$$

Leon-Garcia & Widjaja: *Communication Networks*   **Figure 3.56**

Computer Networks: Transmission Errors

Information: (1,1,0,0) $\longrightarrow$ $i(x) = x^3 + x^2$

Generator polynomial: $g(x) = x^3 + x + 1$

Encoding: $x^3 i(x) = x^6 + x^5$

$$
\begin{array}{r}
x^3 + x^2 + x \\
\hline
x^3 + x + 1 \ ) \ x^6 + x^5 \\
x^6 + \quad x^4 + x^3 \\
\hline
x^5 + x^4 + x^3 \\
x^5 + \quad x^3 + x^2 \\
\hline
x^4 + \quad x^2 \\
x^4 + \quad x^2 + x \\
\hline
x
\end{array}
$$

$$
\begin{array}{r}
1110 \\
\hline
1011 \ ) \ 1100000 \\
1011 \\
\hline
1110 \\
1011 \\
\hline
1010 \\
1011 \\
\hline
010
\end{array}
$$

Transmitted codeword:

$b(x) = x^6 + x^5 + x$

$\underline{b} = (1,1,0,0,0,1,0)$

Leon-Garcia & Widjaja: *Communication Networks*   Figure 3.57
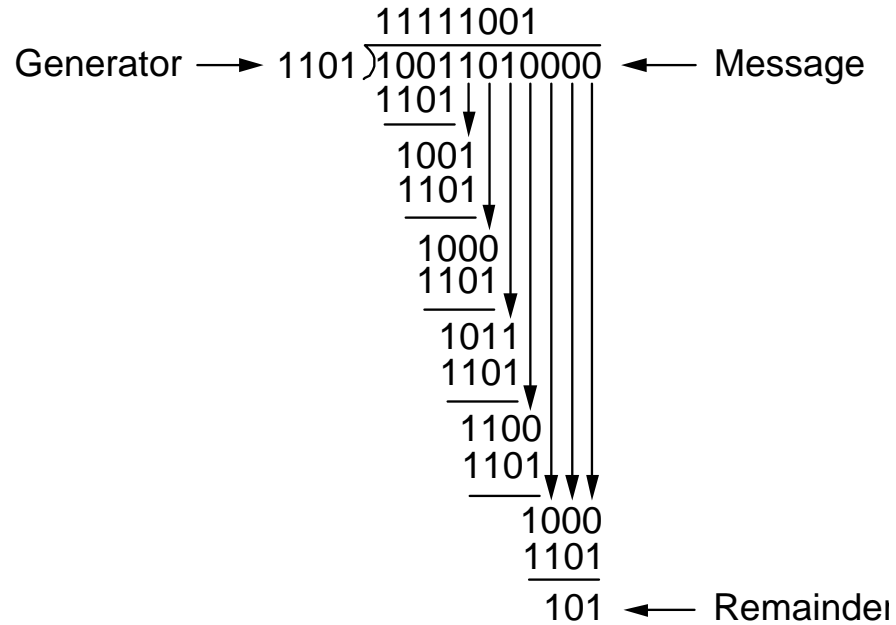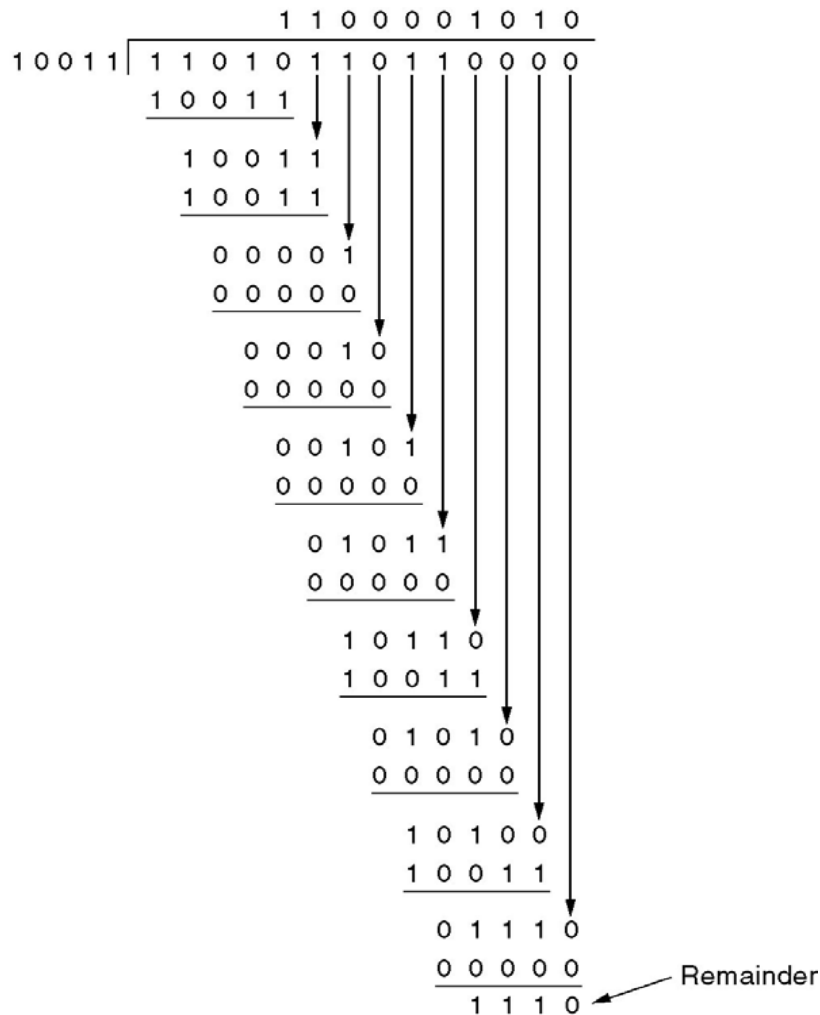
Computer Networks: Transmission Errors

22

CRC Example

Figure 2.17

Frame   : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

```
                            1 1 0 0 0 0 1 0 1 0
              10011 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
                      1 0 0 1 1
                      ─────────
                        1 0 0 1 1
                        1 0 0 1 1
                        ─────────
                          0 0 0 0 1
                          0 0 0 0 0
                          ─────────
                            0 0 0 1 0
                            0 0 0 0 0
                            ─────────
                              0 0 1 0 1
                              0 0 0 0 0
                              ─────────
                                0 1 0 1 1
                                0 0 0 0 0
                                ─────────
                                  1 0 1 1 0
                                  1 0 0 1 1
                                  ─────────
                                    0 1 0 1 0
                                    0 0 0 0 0
                                    ─────────
                                      1 0 1 0 0
                                      1 0 0 1 1
                                      ─────────
                                        0 1 1 1 0
                                        0 0 0 0 0
                                        ───────── ◄── Remainder
                                          1 1 1 0
```

Cyclic Redundancy Checking

Figure 3-8. Calculation of the polynomial code checksum.

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Generating Polynomial Properties for Detecting Errors

1. **Single bit errors:** $e(x) = x^i$ $\qquad 0 \le i \le n\text{-}1$

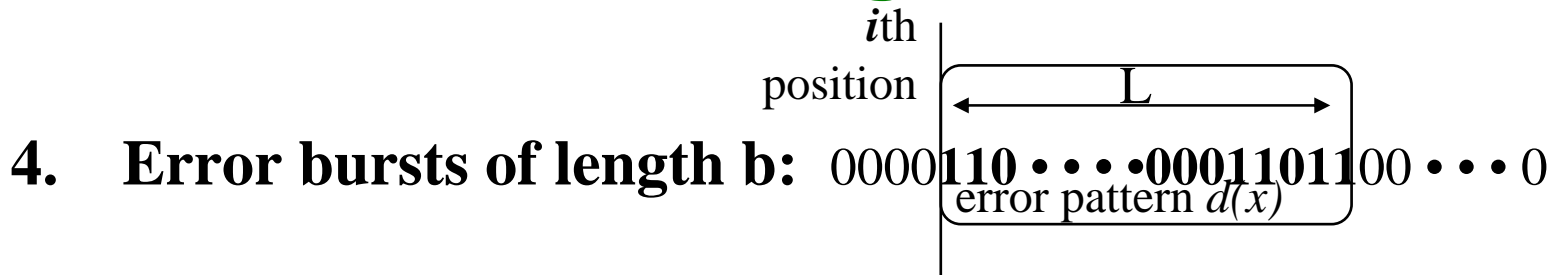   If $g(x)$ **has more than one term**, it cannot divide $e(x)$

2. **Double bit errors:** $e(x) = x^i + x^j \quad 0 \le i < j \le n\text{-}1$
   $$= x^i(1 + x^{j\text{-}i})$$

   If $g(x)$ **is primitive**, it will not divide $(1 + x^{j\text{-}i})$ for $j\text{-}i \le 2^{n\text{-}k}\text{-}1$

3. **Odd number of bit errors:** $e(1) = 1$ If number of errors is odd.

   If $g(x)$ **has $(x+1)$ as a factor**, then $g(1) = 0$ and all codewords have an even number of 1s.

Leon-Garcia & Widjaja: *Communication Networks*   Figure 3.60

Computer Networks: Transmission Errors

Generating Polynomial Properties for Detecting Errors

*i*th position

$$\overset{\text{L}}{\longleftrightarrow}$$

4.   Error bursts of length b: 0000**110 • • • •0001101**00 • • • 0

error pattern *d(x)*

$$e(x) = x^i \; d(x) \quad \text{where } \deg(d(x)) = L\text{-}1$$

$$g(x) \text{ has degree } n\text{-}k;$$

$$g(x) \text{ cannot divide } d(x) \text{ if } \deg(g(x)) > \deg(d(x))$$

- **$L = (n\text{-}k)$** or less: all errors will be detected
- **$L = (n\text{-}k+1)$**: $\deg(d(x)) = \deg(g(x))$

  i.e. d(x) = g(x) is the only undetectable error pattern,

  fraction of bursts which are undetectable $= 1/2^{L-2}$

- **$L > (n\text{-}k+1)$**: fraction of bursts which are undetectable $= 1/2^{n-k}$

Leon-Garcia & Widjaja: *Communication Networks*

Computer Networks: Transmission Errors

Figure 3.61

Basic ARQ with CRC

Packet sequence

Transmitter

Station A

Information frames

Receiver

Station B

Error-free packet sequence

Control frames

CRC

Information packet

Header

Information Frame

CRC

Header

Control frame

Leon-Garcia & Widjaja: *Communication Networks*

Figure 5.8