Networks must be able to transfer data from one device to another with acceptable accuracy

Data can be corrupted during transmission

Some application can tolerate a small level of error such as random errors in audio or video transmission
But transmission of text requires very high level of accuracy

Thus, some applications require that errors be detected and corrected

In order to cope with data transmission errors
Error detection and correction bits

Error detection and correction issues

some issues related, directly or indirectly, to error detection and correction

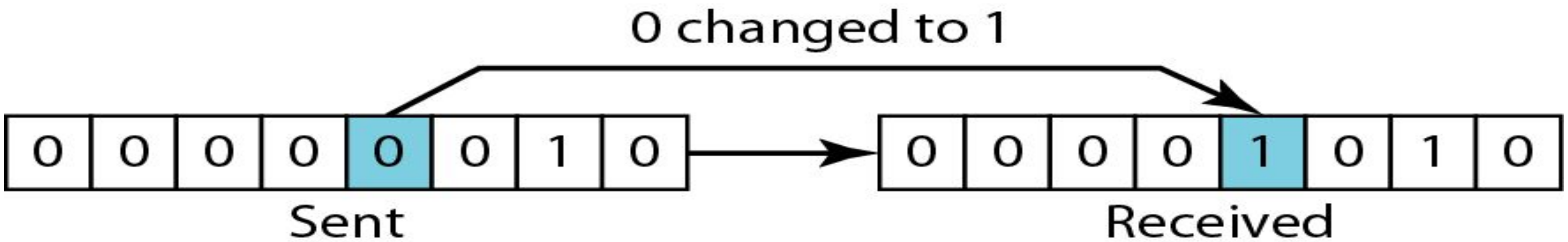
- *Types of Errors*
- Redundancy
- *Coding*
- *Detection versus Correction*
- *Error Correction Methods*
- *Modular Arithmetic*

NOTE: Block of data transmitted from one protocol entity to another is known as protocol data unit (PDU)

Types of Errors

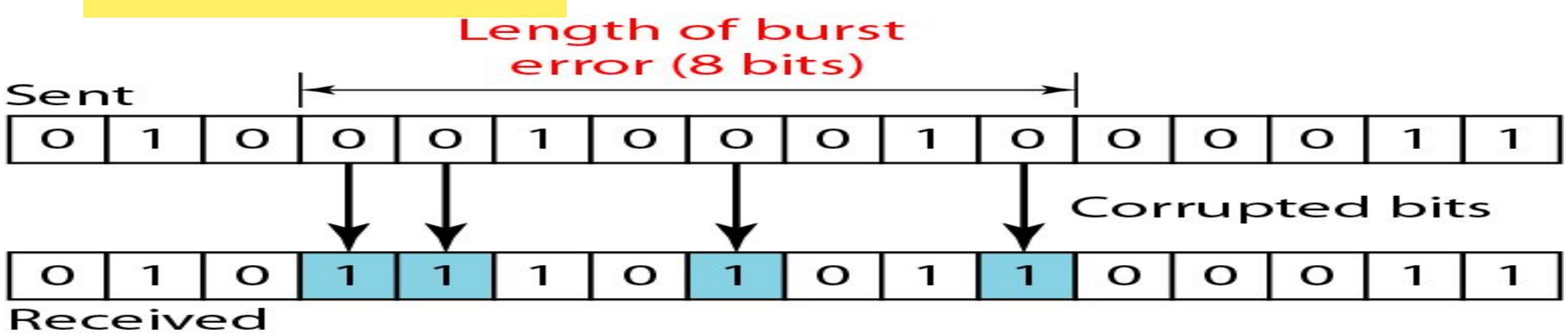
Single bit error is only 1 bit in the data unit has changed

– least likely type of error in serial data transmission



In a burst error, 2 or more bits in the data unit get changed

The length of the burst is measured from the first corrupted bit to the last corrupted bit. OR distance between the first and last errors in data block



Redundancy: Central concept in detecting /correcting errors

- ❖ Need to send extra (redundant) bits with the data
- ❖ Extra bits are added by the sender and removed by the receiver

- Presence of redundant allows receiver to detect or correct corrupted bits

Coding: Various coding schemes to achieve redundancy

- The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits
- The receiver checks the relationships between the two sets of bits to detect or correct the errors
- **In any coding scheme, t**he ratio of redundant bits to the data bits and the robustness of the process are important factors

Detection Versus Correction

Error detection concerns only to see if any error has occurred

- *Simply Yes or No*
- *Even not interested in the number of errors (corrupted bits)*
    - *Single bit error is same as the burst error*

Error correction: need to know the exact number of corrupted bits and their location in the message (more important)

- The number of the errors and the size of the message are important factors
- In an 8 bit data
    - To correct a single bit error, need to consider eight possible error locations
    - To correct two errors, need to consider 28 possibilities or combinations
- Imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits

Error Correction Methods

- **Forward error correction**
  - ❖ The receiver tries to guess the message by using redundant bits if the number of errors is small
- **Correction by retransmission**
  - ❖ A technique in which the receiver detects the occurrence of an error and asks the sender to resend the message
  - ❖ Resending is repeated until a message arrives error free (as per believe of the receiver)

Modular Arithmetic

- uses only a limited range of integers
- *modulo-N arithmetic*
  - ❖ define an upper limit, called a modulus *N*
  - ❖ then use only the integers 0 to *N - I, inclusive*
  - ❖ no carry when adding two digits in a column
  - ❖ no borrow when subtracting one digit from another in a column
- **In Modulo-2 arithmetic** *(XORing of two single bits or words*

$0 \oplus 0 = 0 \qquad 1 \oplus 1 = 0$

a. Two bits are the same, the result is 0.

$0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1$

b. Two bits are different, the result is 1.

$$
\begin{array}{cccccc}
 & 1 & 0 & 1 & 1 & 0 \\
\oplus & 1 & 1 & 1 & 0 & 0 \\
\hline
 & 0 & 1 & 0 & 1 & 0
\end{array}
$$

c. Result of XORing two patterns

Error Control Requirements

The most common techniques for error control are based on some or all of the following ingredients:

- **Error detection**
  - Receiver detects errors and discards PDUs in error
- **Positive acknowledgement**
  - Destination returns acknowledgment of successfully received, error-free PDUs
- **Retransmission** after timeout
  - Source retransmits unacknowledged PDUs after a predetermined amount of time
- **Negative acknowledgement** and **retransmission**
  - Destination returns negative acknowledgment to PDUs in which an error is detected
  - The source retransmits such PDUs

Error Detection Codes

- Data are transmitted as one or more contiguous sequences of bits, called frames
- Data transmission can contain errors (single bit or burst)
- Error detection codes detect the presence of an error
- **How to detect errors?**
  - ❖ If only data is transmitted, errors cannot be detected
  - ❖ Send more information with data that satisfies a special relationship
    - Add redundancy
- **Error-detecting codes are commonly used in link, network, and transport layers**

Error Detection Process

Transmitter

- For a given frame of bits , the transmitter adds additional bits that constitute an error-detecting code
- an error-detecting code (check bits) is calculated from transmitted (data) bits
- Check bits are appended to data bits

Receiver

- Separates incoming frame into data bits and check bits
- Calculates check bits from received data bits
- Compares calculated check bits against received check bits
- A detected error occurs if and only if there is a mismatch

Error Detection Process

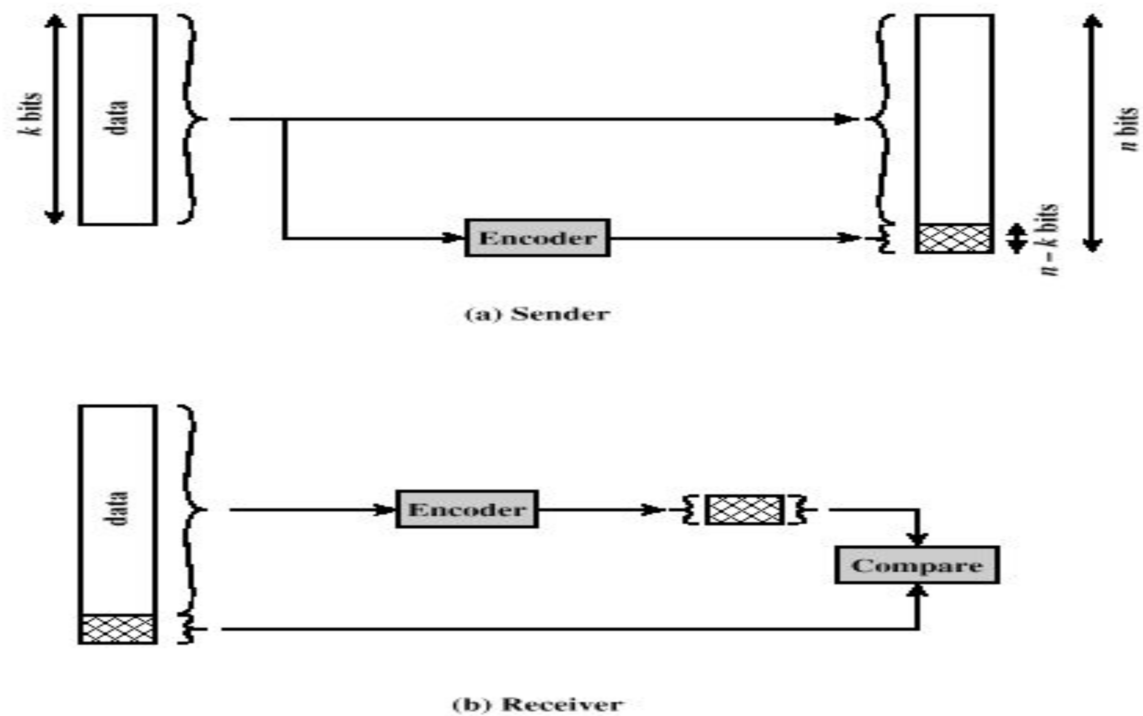


Figure 8.1 Error Detection Process

Taken from “Wireless Communications & Networks” by William Stallings

Error Detection Codes
Parity Check

- Parity bit (a single bit) appended at the end of data block
- **Even parity**
  - Added bit ensures an even number of 1s
- **Odd parity**
  - Added bit ensures an odd number of 1s
- Example, 7-bit character [1110001]
  - Even parity [1110001**0**]
  - Odd parity [1110001**1**]

Parity Check

- If the transmitter is transmitting **1110001** and using odd parity
  - ❖ It will append a **1** and transmit **11100011**
  - ❖ The receiver examines the received character and if the total number of 1s is odd
    - ❖ No error
  - ❖ If 1 bit or any odd number of bits is inverted during transmission, For example,
  - ❖ Then the receiver will detect an error
- **Performance:**
  - ❖ Detects all odd-number errors in a data block (1,3,5,…bits in error)
  - ❖ Detects NO errors that flip an even number of bits (2, 4, 6, … bits in error)

Error Detection Codes: Cyclic Redundancy Check (CRC)

- One of the most common and powerful error-detecting codes
- **Transmitter**
  - For a given ***k*-**bit block, transmitter generates an (***n*-*k***)-bit **frame check sequence (FCS)**
  - Resulting frame consisting of ***n*** bits is exactly divisible by **predetermined number (a pattern)**
- **Receiver**
  - Divides incoming frame by **predetermined number**
  - If no remainder, assumes no error
- Procedure can be represented by
  - Modulo 2 Arithmetic
  - Polynomials

Modulo 2 Arithmetic

- Modulo 2 arithmetic is performed digit by digit on binary numbers
- Each digit is considered independently from its neighbors
- Binary addition with no carries: Exclusive-OR (XOR)
- Binary subtraction with no borrows: as the XOR operation

$$\begin{array}{r} 1111 \\ +\ \underline{1010} \\ 0101 \end{array} \qquad \begin{array}{r} 1111 \\ -\ \underline{0101} \\ 1010 \end{array} \qquad \begin{array}{r} 11001 \\ \underline{\times\ 11} \\ 11001 \\ \underline{11001} \\ 101011 \end{array}$$

CRC using Modulo 2 Arithmetic

Parameters:

- ***T* = *n*-bit frame to be transmitted**
- ***D*** = ***k*-bit block of data; the first *k* bits of *T***
- *F* = *(n – k)*-bit FCS; the last *(n – k)* bits of *T*
- *P* = pattern of ***n–k*+1** bits; this is the **predetermined divisor**
- *Q* = Quotient
- *R* = Remainder

CRC using Modulo 2 Arithmetic: Example

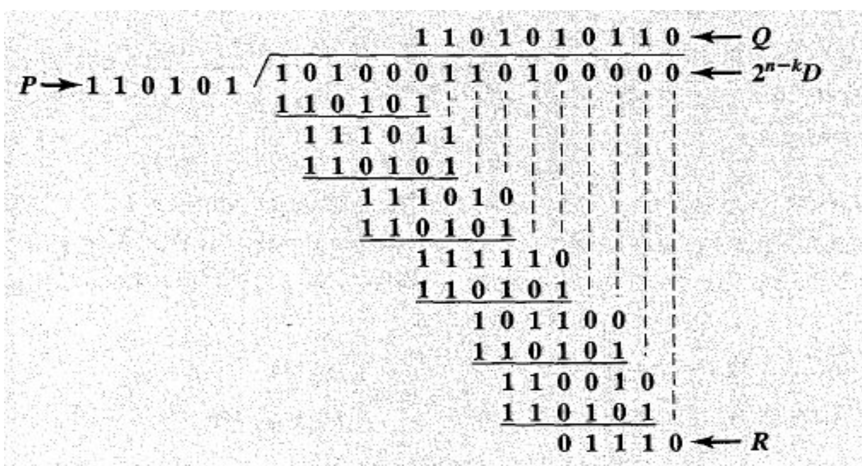
Given: ***Pattern*** *(deviser)P = 110101 (6 bits) --------- n-k+1*

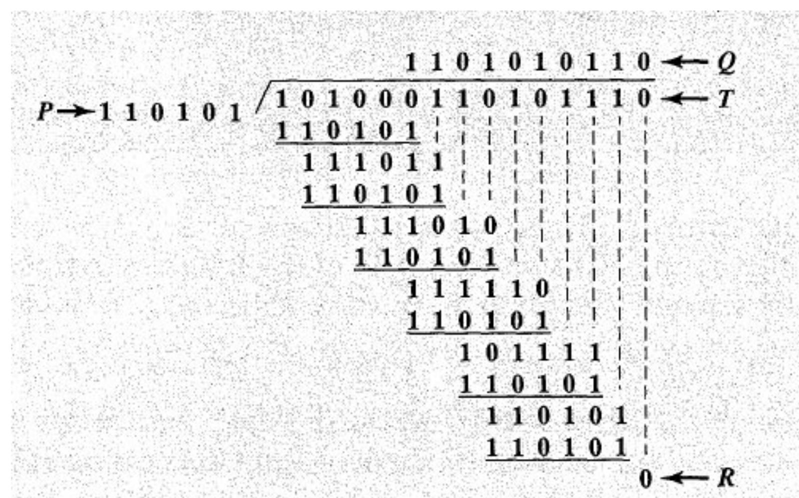
FCS *= to be calculated (n-k = 5 bits)*

Message *D (k-bit block) = 1010001101 (10 bits)*

  – *Thus, n (total bits) = 15 (as n-k+1 = 6), k = 10 and n-k = 5 bits*

- *The message is multiplied by 2^5, producing 101000110100000*
- *The product is divided by P*
- *The remainder is added to 2^5D to give T = 101000110101110*
- If no errors, then receiver receives T as it is. The received frame is divided by P
- **If there is no remainder, it is assumed that there have been no errors**





CRC using Modulo 2 Arithmetic: Example

Given: ***Pattern*** *(diviser)P* = 11001 *(5 bits) --------- n-k+1*

FCS *= to be calculated (n-k = 4 bits)*

Message *D (k-bit block)* = 110011 *(6 bits)*

– *Thus, n (total bits) = 10 (as n-k+1 = 5), k = 6 and n-k = 4 bits*

□ *Transmitted block T* = 1100111001

```
            100001
11001 ) 1100110000
        11001
            10000
            11001
             1001 = R(x)
```

Send the block 110011 1001

At Receiver

```
11001 ) 1100111001
        11001
            11001
            11001
            00000
```

No remainder

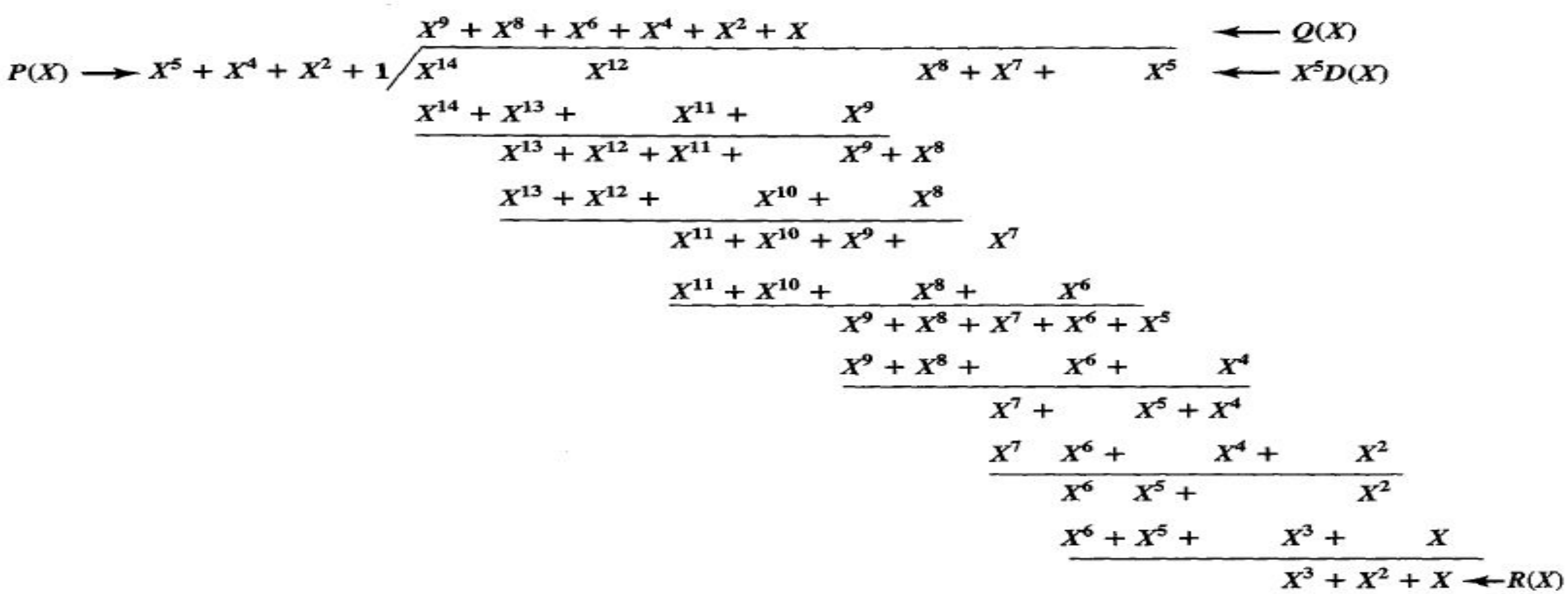
□ Accept

CRC using Polynomials

- All values expressed as polynomials
  - ❖ Dummy variable *X* with binary coefficients
  - ❖ The coefficients correspond to the bits in the binary number
- For D = 1010001101 -----$D(X) = X^9 + X^7 + X^3 + X^2 + 1$
- For P = 110101 -----$P(X) = X^5 + X^4 + X^2 + 1$

CRC using Polynomials: Example

- **Taking previous example**
- For D = 1010001101 -----$D(X) = X^9 + X^7 + X^3 + X^2 + 1$
- For P = 110101 -----$P(X) = X^5 + X^4 + X^2 + 1$
- End up with R = 01110-----$R(X) = X^3 + X^2 + X$



Error Correction Codes

- **Error detection** is found a useful technique in data link control protocols and in transport protocols (TCP)
- Error detection requires **retransmission** (using Automatic Repeat reQuest)
- **Detection** inadequate for **wireless applications**
  - **wireless** links are notoriously **noisy** and error prone when compared to optical fibers
  - **Bit error rate** on wireless link can be high, results in a large number of retransmissions
  - **Long propagation delay** compared to transmission time
- Without **error-correcting codes**, it would be hard to get anything through

We need error control mechanisms to detect and correct errors that occur in the transmission of PDUs

Forward error correction codes (FEC)

- Designed to detect and correct errors
- **Widely used form of error correction code**
  - Block error correction codes
- Follow the same general layout as in error detection codes
  - Take as input ***k-bit*** block, add ***r = n-k*** bits to produce **n** bit-block

Forward Error Correction Process

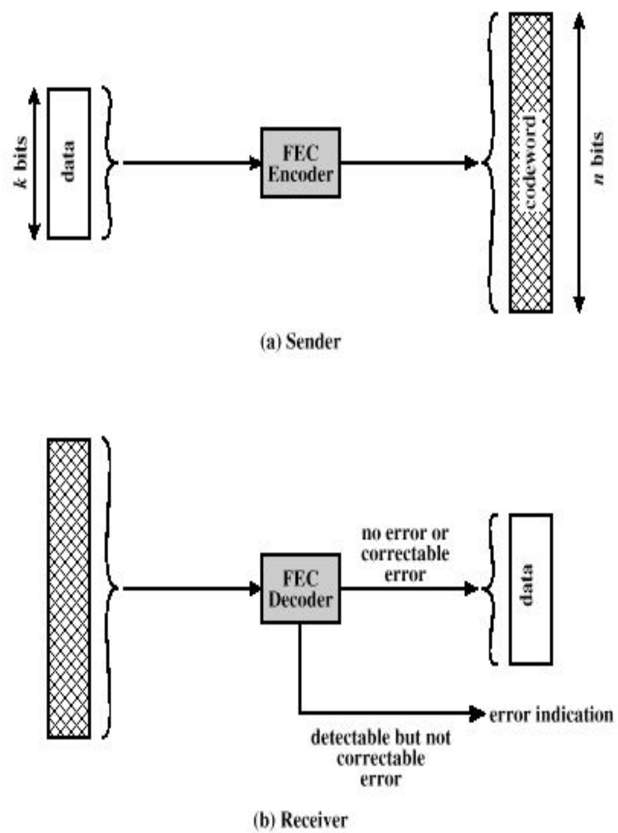


Figure 8.5 Forward Error Correction Process

FEC Decoder Outcomes

When a block is passed through FEC, possible outcomes

- **No errors present**
  - Input to the FEC decoder matches original codeword
  - Decoder produces the original data block as output
- **Decoder detects and corrects** bit errors for certain error patterns
- **Decoder detects but cannot correct** bit errors for certain error patterns
  - Decoder simply reports uncorrectable error
- **Decoder detects no bit errors** (for rare error patterns), though errors are present