Basic concepts

- ★ Networks must be able to transfer data from one device to another with complete accuracy.
- ★ Data can be corrupted during transmission.
- ★ For reliable communication, errors must be detected and corrected.
- ★ **Error detection and correction** are implemented either at the **data link layer** or the **transport layer** of the OSI model.

Types of Errors

Single-bit error

Single bit errors are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.

Example:

- ★ If data is sent at 1Mbps then each bit lasts only 1/1,000,000 sec. or 1 μs.
- ★ For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare.

Burst error

Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
Burst error
0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1
Received

Two errors
0 1 0 0 0 0 1 0
Sent
0 0 0 0 1 0 1 0
Received

The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Burst errors does not necessarily **mean that the errors occur in consecutive bits**, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

- **Burst error is most likely to happen in serial transmission** since the duration of noise is normally longer than the duration of a bit.
- The number of bits affected depends on the data rate and duration of noise.

Error detection

Error detection means to decide whether the received data is correct or not without having a copy of the original message.

Error detection **uses the concept of redundancy**, **which means** adding extra bits for detecting errors at the destination.

Redundancy

Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

Four types of redundancy checks are used in data communications

Vertical Redundancy Check
VRC

Example

Suppose the sender wants to send the word *world*. In ASCII the five characters are coded as

1110111	**1101111**	**1110010**	**1101100**	**1100100**
w	o	r	l	d

The following shows the actual bits sent

1110111<u>0</u> 1101111<u>0</u> 1110010<u>0</u> 1101100<u>0</u> 1100100<u>1</u>

Performance

- It can detect burst errors only if the total number of errors is odd.

Longitudinal Redundancy Check
LRC

Performance

- LCR increases the likelihood of detecting burst errors.
- If two bits in one data units are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

Two-dimensional parity (LRC + VRC)

Cyclic Redundancy Check
CRC

Cyclic Redundancy Check

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a *frame check sequence (FCS)*, so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.

- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.

Binary Division

Polynomial

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

Polynomial and Divisor

Standard Polynomials

CRC-12

$$x^{12} + x^{11} + x^{3} + x + 1$$

CRC-16

$$x^{16} + x^{15} + x^{2} + 1$$

CRC-ITU

$$x^{16} + x^{12} + x^{5} + 1$$

CRC-32

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Dataword
$x^3 + 1$
Divisor
$x^3 + x + 1$
$x^3 + x$
$x^6 + x^3$
$x^6 + x^4 + x^3$
x^4
$x^4 + x^2 + x$
$x^2 + x$
Remainder
Dividend: augmented dataword
Codeword
$x^6 + x^3$
$x^2 + x$
Dataword
Remainder

Checksum

Section K
Section 1
n bits
n bits
n bits
Section 1
n bits
Section 2
n bits
Section K
n bits
Sum
n bits
Complement
n bits
Checksum
Sender
Section k
Section 1
n bits
n bits
n bits
n bits
Checksum
Section 1
n bits
Section 2
n bits
Section K
n bits
Checksum
n bits
Sum
All 1s, accept
Otherwise, reject
Receiver

How can we represent the number 21 in one's complement arithmetic using only four bits?

How can we represent the number −6 in one's complement arithmetic using only four bits?

Sender site
7
11
12
0
6
0
Sum
36
Wrapped sum
6
Checksum
9
7, 11, 12, 0, 6, 9
Packet
Receiver site
7
11
12
0
6
9
Sum
45
Wrapped sum
15
Checksum
0
1 0 0 1 0 0 36
1 0
0 1 1 0 6
1 0 0 1 9
Details of wrapping and complementing
1 0 1 1 0 1 45
1 0
1 1 1 1 15
0 0 0 0 0
Details of wrapping and complementing

At the sender

- The unit is divided into *k* sections, each of *n* bits.
- All sections are added together using one's complement to get the sum.
- The sum is complemented and becomes the checksum.
- The checksum is sent with the data

At the receiver

- The unit is divided into *k* sections, each of *n* bits.
- All sections are added together using one's complement to get the sum.
- The sum is complemented.
- If the result is zero, the data are accepted: otherwise, they are rejected.

Performance

- The checksum detects all errors involving an odd number of bits.
- It detects most errors involving an even number of bits.
- If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sums of those columns will not change and the receiver will not detect a problem.

Error Correction

It can be handled in two ways:

1) receiver can have the sender retransmit the entire data unit.
2) The receiver can use an error-correcting code, which automatically corrects certain errors.

Single-bit error correction

To correct an error, the receiver reverses the value of the altered bit. To do so, it must know which bit is in error.

Number of redundancy bits needed

- Let data bits = m
- Redundancy bits = r

$\therefore$ Total message sent = $m+r$

The value of r must satisfy the following relation:

$$2^r \geq m+r+1$$

Error Correction

Hamming Code

Hamming Code

Hamming Code

Example of Hamming Code

Single-bit error

Error Detection