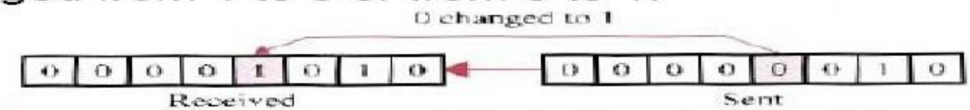
Error Detection and Correction

Error Control

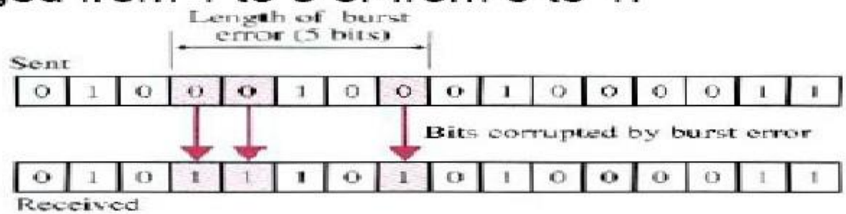
- Is a set of procedures to provide reliable delivery of data between two physically connected devices.
- The reasons we need flow control:
  - Data can be corrupted during transmission.
  - For reliable communication, errors must be detected and corrupted.
- Error control can include:
  - Error detection:
    - Allows the receiver to detect the presence of errors
  - Error correction:
    - Allows the receiver to correct the errors.

Errors:

- Can be caused by signal attenuation or noise.
- Types of errors:
  - Single-bit error: only one bit of a given data unit is changed from 1 to 0 or from 0 to 1.

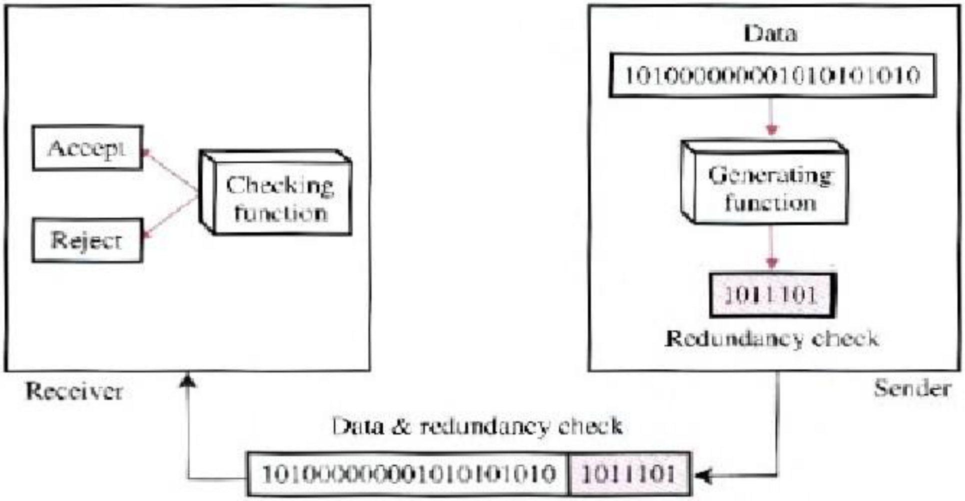


  - Burst-error: two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.



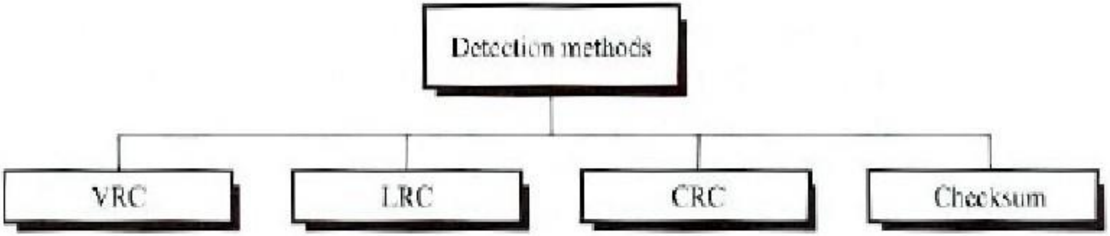
Error Detection

- Error detection uses the concept of redundancy.
  - Adds extra bits for detecting errors at destination.
    - For efficiency, extra bits $k << n$ (information bits).
    - Generating function is used to generate these extra bits.



Error Detection

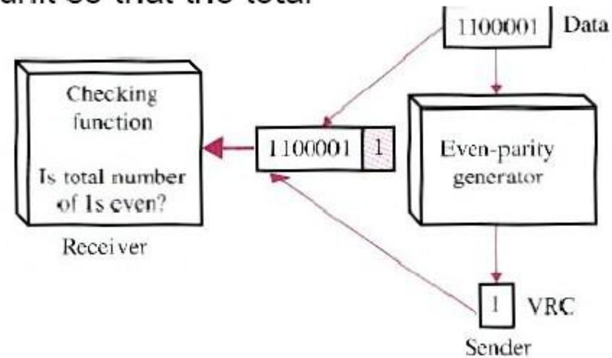
- Four types of redundancy checks:



- VRC: vertical redundancy check or parity check
- LRC: longitudinal redundancy check
- CRC: cyclic redundancy check
- VRC, LRC, CRC used by data link layers.
- Checksum used by higher-layer protocols.

Can detect all single-bit errors.Can detect burst errors only if the total number of errors in each data unit is odd.

- Vertical redundancy check (VRC):
  - Adds a parity bit to every data unit so that the total number of 1s becomes
    - even – even parity checking
    - odd – odd-parity checking



- Example:
  - The word "cute" is coded in ASCII as:

    1100011 1110101 1110100 1100101
    c u t e
  - Using even-parity checking, the sender will send:

    11000110 11101011 11101000 11001010
  - If the word is not corrupted during transmission:
    - The receiver counts the 1s in each character and comes up with (4, 6, 4, 4) – all even numbers.
  - If the word is corrupted during transmission, say:

    11010110 11101011 11101000 11000010
    - The receiver counts the 1s in each character and comes up with (5, 6, 4, 3).

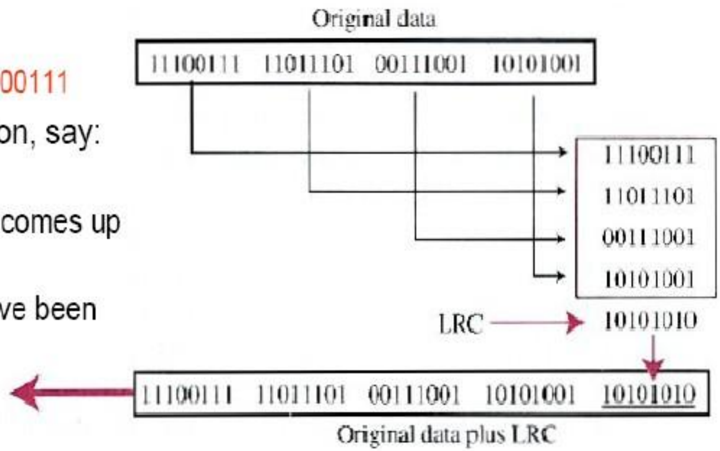
•Increases the likelihood of detecting burst errors.
•n bits LRC can detect a burst error of n bits.
•Errors may be undetected if:
  •They occur in exactly the same positions.
  •Have even number of errors in that position.

- Longitudinal redundancy check (LRC):
  - 2-dimensional parity checking.
    - Divides a block of bits into rows of *n* bits.
    - Calculates the even/odd parity for each column.
      - This results in an extra row of parity bits.
- Example:
  - The word "cute" is coded in ASCII as:

    1100011 1110101 1110100 1100101
    c u t e
  - Using LRC, the sender will send:

    1100011 1110101 1110100 1100101 0000111
  - If the word is corrupted during transmission, say:

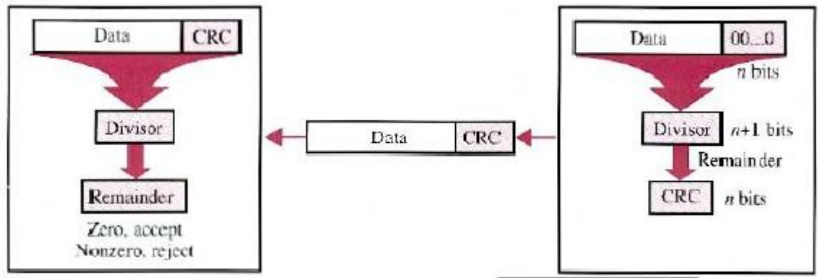
    1101101 0100101 1110100 1100001
    - The receiver calculates the LRC and and comes up with 1011001 ≠ 0000111.
    - 5 bits of LRC have changed – 5 errors have been detected.

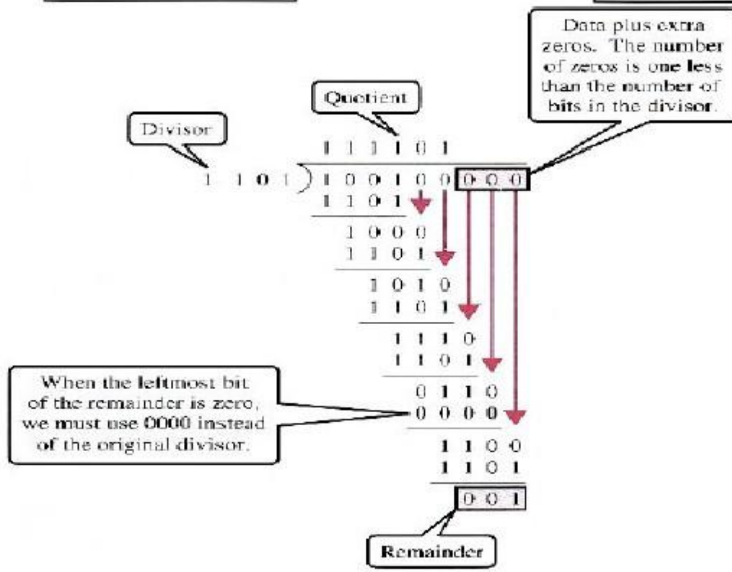


- Cyclic redundancy check (CRC):
  - Based on binary division
    - Cf. VRC and LRC based on addition
  - Divides the data unit by predetermined divisor or generator using modulo-2 division
  - CRC = remainder.
- CRC generator:
  - For n-bits data unit and m-bits divisor:
    - Forms dividend: n-bits + (m-1) bits of zeros.
    - Divides dividend by divisor.
    - Subtracting each bit of divisor without disturbing the next higher bit.

      1 – 1 or 0 – 0 = 0

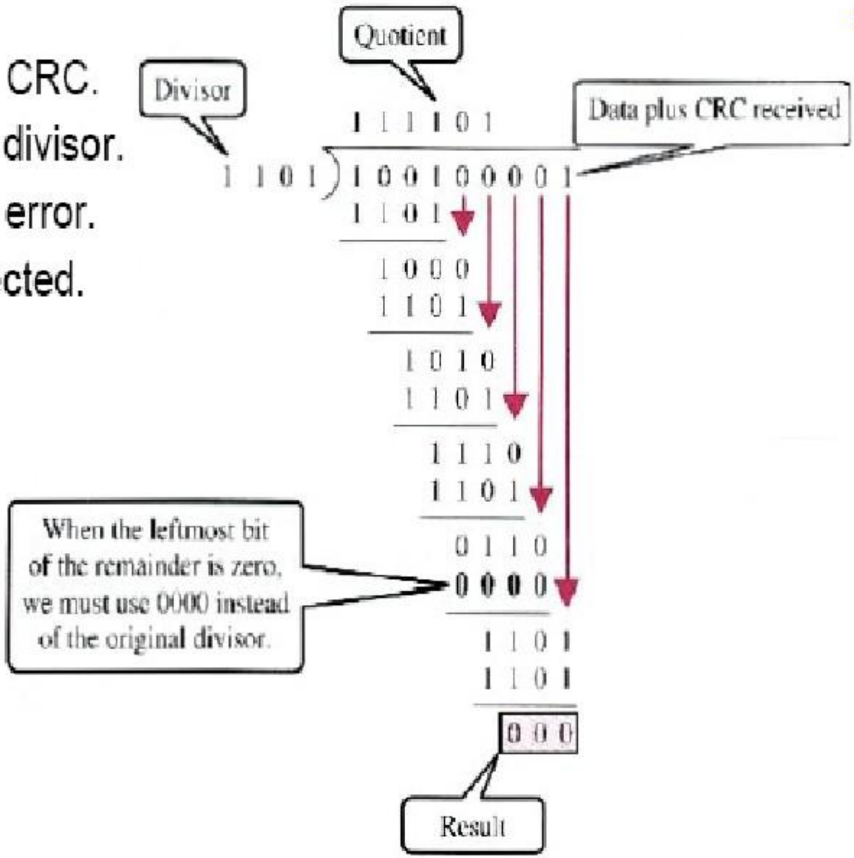
      1 – 0 or 0 – 1 = 1
  - Sends data + CRC.



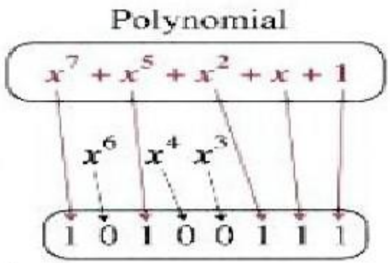


CRC checker:

- Receives data + CRC.
- Divides them by divisor.
- If result = 0 – no error.
- Else – error detected.

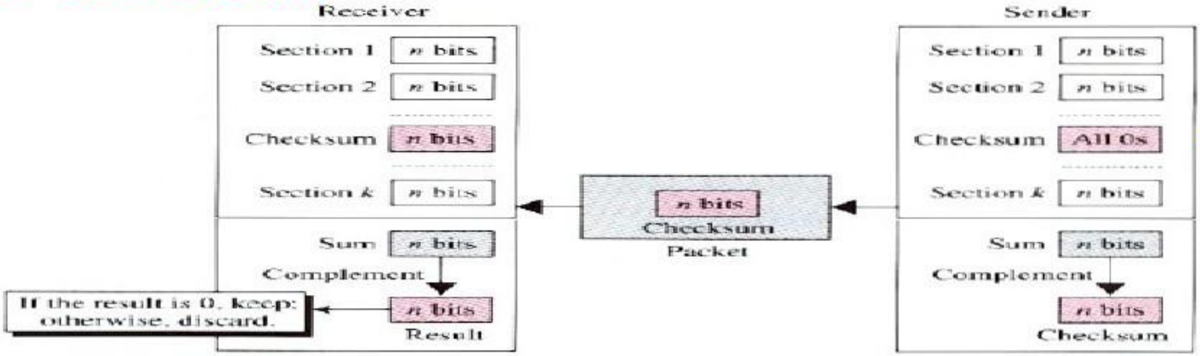


- CRC generator or divisor:
  - Often represented as an algebraic polynomial for two reasons:
    - It is short.
    - It can be used to prove the concept mathematically.
  - How to select the polynomial is beyond this course.



- CRC performance:
  - If the divisor is selected properly, then CRC:
    - Can detect all burst errors that affect an odd number of bits (similar to VRC and LFC).
    - Can detect all burst errors of length $\leq$ degree of polynomial.
    - Can detect with a very high probability all burst errors of length > degree of polynomial.
      - e.g. CRC-12 ($x^{12} + x^{11} + x^3 + x + 1$) can detect 99.97 percent of the time burst errors with a length of 12 or more.

- Checksum:



- Checksum generator (sender):
  - Divide data unit into segments of n bits (usually $n = 16$).
  - Add together all segments using one's complement to get the sum (see Appendix C).
  - Complement the sum to become the checksum.
  - Send the checksum with the data.
- Checksum checker (receiver):
  - Divide data unit into segments of n bits.
  - Add together all segments using one's complement to get the sum.
  - Complement the sum.
  - If the result is zero, accept the data, otherwise, reject the data.

Idea: view message as a sequence of 16-bit integers. Add these integers together using 16-bit ones complement arithmetic, and then take the ones complement of the result. That 16-bit number is the checksum.

- Example:
  - The following block of 16 bits is to be sent
    10101001 00111001
  - The sender calculates an 8-bit checksum

```
  10101001
  00111001
  --------
  11100010      (sum)
  00011101      (checksum)
```

    Send 10101001 00111001 00011101
  - If the data is not corrupted during transmission,

```
  10101001      11100010
  00111001      00011101
  --------      --------
  11100010      11111111      (sum)
                00000000      (complement) - OK
```

  - If the data is corrupted during transmission, say we receive
    10101111 11111001 00011101
    Adding them:

```
                    10101111
                    11111001
                    --------
                1   10101000
                    00011101
                    --------
Result          1   11000101
Carry                      1
                    --------
Sum                 11000110
Complement          00111001      error detected !!!
```

- Checksum performance:
  - Can detect all errors involving an odd number of bits (similar to VRC, LRC, CRC).
  - Can detect most errors involving an even number of bits.

Error Correction

- Error correction can be handled in two ways:
  - Receiver can ask the sender to retransmit the corrupted data unit.
  - Receiver can use an error-correcting code, which automatically correct certain errors.
    - Most error correction is limited to 1 – 3 bits errors.
    - Much more redundancy bits are needed to correct larger bit errors – often becomes inefficient to use.
  - The choice depends on the distance between devices and applications.