Data Link Layer

Responsibilities of DLL

- Provide following services to upper layer protocols:
- Logical Link Control Sublayer
  - Framing
  - Error Control
  - Flow Control
  - Reliable Transmission

- In multipoint network, MAC sublayer will be present after LLC at DLL.
  - Medium Access Control

Network Performance Matrix

- Network performance can be measured in two different ways:
  - Latency
  - Bandwidth (Throughput)

- **<u>Defn</u>**: Network ***<u>throughput</u>*** (or ***<u>effective throughput</u>***) is the measured number of bits that can be transmitted over a particular medium in a given amount of time. Usually, described in *bits/sec* (or *bps*).
- The throughput is the maximum number of bits/sec an application can expect to receive.

Bandwidth >= Throughput

- For applications, we can describe throughput as the "bandwidth requirements of an application."

Network Performance Matrix

- **<u>Defn</u>:** **Latency** (or **end-to-end delay**) is the amount of time is takes for a single bit to propagate from one end of a network to another. It depends upon the following factors:

1. Propagation delay (
2. Transmission time
3. Queueing delay
4. Processing delays

Network Performance Matrix

1. **Propagation delay**
   - We calculate this using the speed-of-light propagation delay:
     - in a vacuum, $3.0 * 10^8$ meters/sec
     - in a cable, $2.3 * 10^8$ meters/sec
     - in fiber, $2.0 * 10^8$ meters/sec
   - This value is a function of the distances and the speed-of-light delay.
2. **Transmission Delay**
   - This is the amount of time it takes to transmit the data onto the transmission media. This value is a function of the bandwidth and the packet size.
3. **Queueing Delay**

   This is the time the data spends in being waiting for its turn (queueing) to be transmitted.
4. **Processing Delay**
   - This is the time the data spends in being processed to be transmitted.

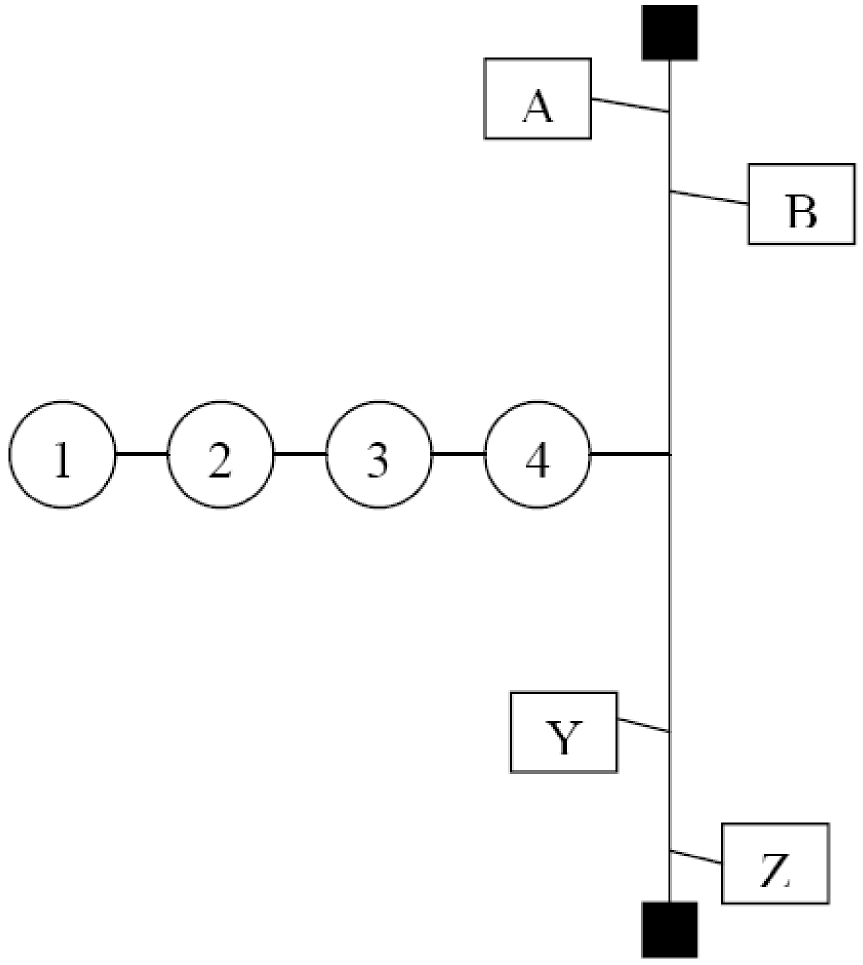
Latency

$$Latency\ (L) = T_{prop} + T_{trans} + T_{queu} + T_{proc}$$

Example

- Given the internet pictured below with a propagation speed of 200 m/microsec on the packet-switched WAN and the LAN and:
  - nodes 1-4 equidistantly spaced 2 km apart on a WAN with 50 Mbps links between nodes.
  - Assume processing time for these nodes is **0.**
  - nodes A, B, Y, Z and 4 are on a 10BASE5 coaxial LAN with nodes A and Z at the absolute
  - edge of the LAN segment and node 4 connected at the middle of the LAN.
  - Assume a packet = frame = 1000 bytes on this internet, and no processing delay at any nodes.
- **How long will it take to send a packet from node 1 to node Z in the situation that when the packet arrives at node 4 there are two packets in front of it waiting to go out on the LAN to Node A.**

Example



Solution

```
delay 1 to z = delay 1 to 4 + delay 4 to z


delay 1 to 4 =  3 * (ttrans + tprop + tproc + tqueue)        assume tproc = tqueue = 0


                   1000 * 8 bits                   2000 m
            =  3 * (---------------------- +  ------------------)
                   50 * 10 ** 6 bits/sec    200 m /microsec

                   8 *  10**3 bits
            =  3 * (-------------------  + 10 microseconds)
                   50 * 10**6 bits/sec


            =  3 * (.16 * 10 ** -3 seconds + 10 microseconds)

            =  3 * (160 microseconds + 10 microseconds) =  510  microseconds
```

Solution

```
delay 4 to z =  queuing at 4 + ttransLAN + tpropLAN

                queuing at 4 =  2 *ttransLAN

delay 4 to z =  3 * ttransLAN + tpropLAN


                      8 * 10 ** 3 bits               250 m
             =  3 * (--------------------) +  ---------------
                     10 * 10 ** 6 bits/sec     200 m /microsec

             =  3 * (.8 * 10 ** -3 seconds) +   1.25 microseconds

             =  3 * 800 microseconds + 1.25 microseconds  =  2401.25 microseconds

delay 1 to z  =  510 microseconds + 2401.25 microseconds =  2.91125 milliseconds
```

Framing Protocols

- In computer networks, we are operating in a packet switching network which means block of data (not individual bits) are exchanged between the nodes.
- Network adapter enables the nodes to exchange frames (block of data).
- The problem at the DLL is that how to mark the start of the frame and end of the frame in the bit stream received from the physical later.

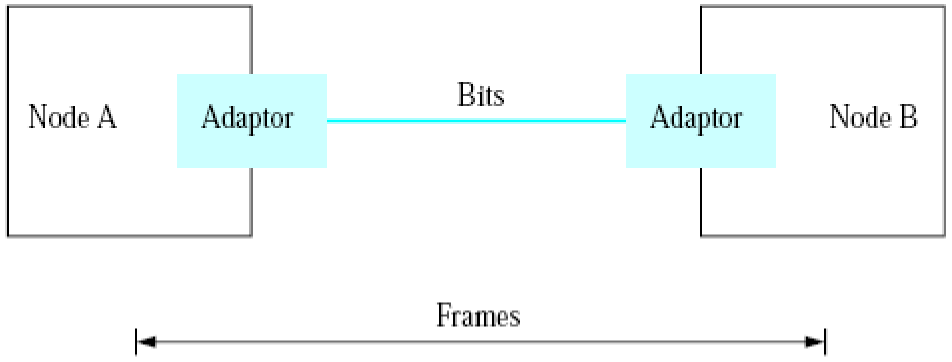
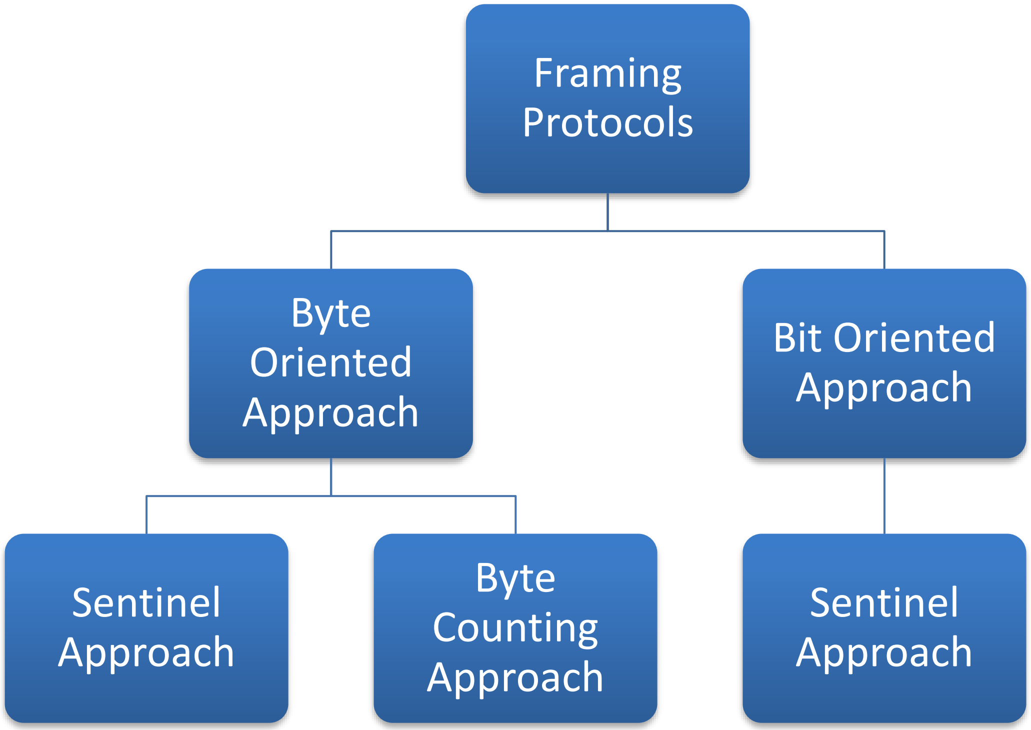


Figure 5: Frames between hosts

Framing Protocols

- Several approached are used to handle this problem:

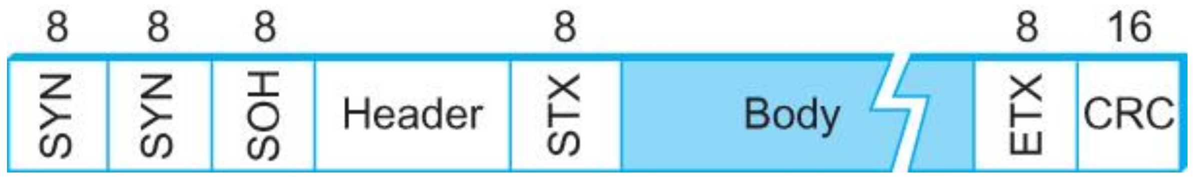


Framing Protocols (Contd.)

- Byte-oriented Protocols
  - To view each frame as a collection of bytes (characters) rather than bits.
  - Sentinel Approach
    - BISYNC (Binary Synchronous Communication) Protocol
    - Developed by IBM (late 1960)
  - Byte Counting Approach
    - DDCMP (Digital Data Communication Protocol)
    - Used in DECNet

Framing Protocols (Contd.)

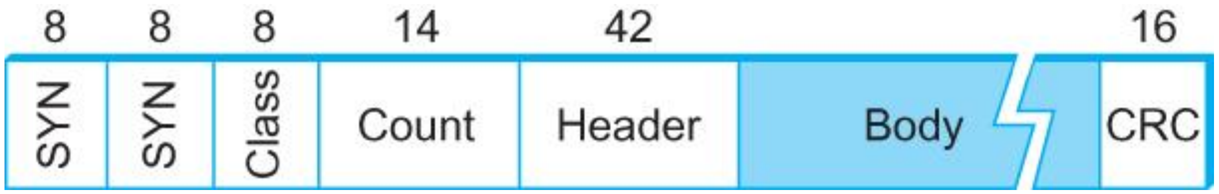
- BISYNC – Sentinel Approach (Byte Oriented)
  - Frames transmitted beginning with leftmost field
  - Beginning of a frame is denoted by sending a special SYN (synchronize) character
  - Data portion of the frame is contained between special sentinel character STX (start of text) and ETX (end of text)
  - SOH : Start of Header
  - DLE : Data Link Escape
  - CRC: Cyclic Redundancy Check



BISYNC Frame Format

Framing Protocols (Contd.)

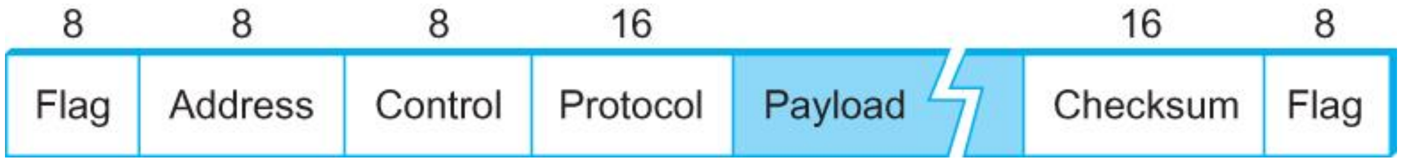
- DDCMP: Byte-counting approach (Byte Oriented)
  - *count* : how many bytes are contained in the frame body
  - If *count* is corrupted
    - Framing error



DDCMP Frame Format

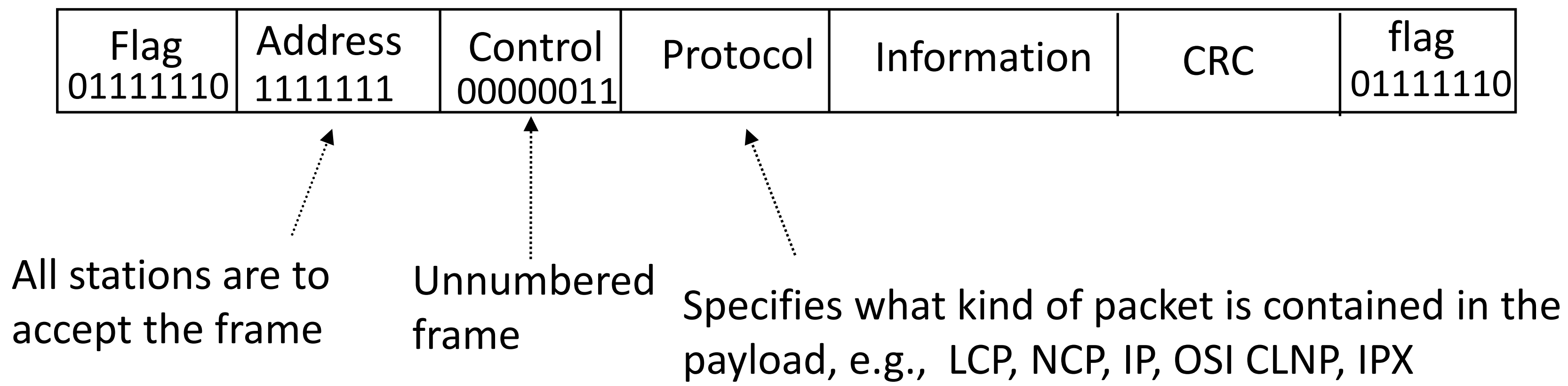
Framing Protocols (Contd.)

- Point to Point Protocol (PPP): Sentinel Approach (Byte Oriented)
- which is commonly run over Internet links uses sentinel approach
  - Special start of text character denoted as Flag
    - 0 1 1 1 1 1 1 0
  - Address, control: default numbers
  - Protocol for demux: IP / IPX
  - Payload : negotiated (1500 bytes)
  - Checksum: for error detection



PPP Frame Format

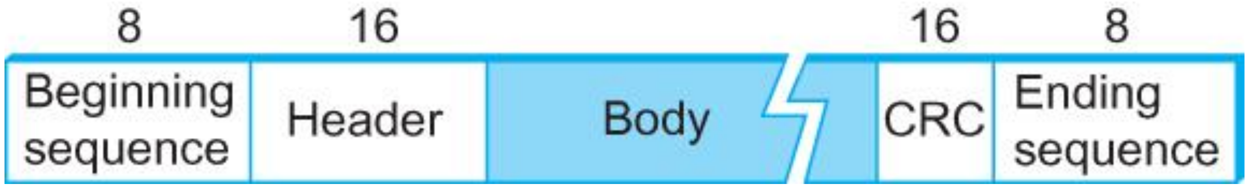
PPP header fields details



Specifies what kind of packet is contained in the payload, e.g., LCP, NCP, IP, OSI CLNP, IPX

Framing Protocols (Contd.)

- HDLC: High Level Data Link Control: Sentinel Approach (Bit Oriented)
  - Beginning and Ending Sequences
    - 0 1 1 1 1 1 1 0



HDLC Frame Format

Framing Protocols (Contd.)

- Problem with the framing protocols: If the special character appears in between the frame in Byte Oriented Approach.
- Solution: Byte Stuffing.

- Problem with the framing protocols: If the special bit sequence appears in between the frame in Bit Oriented Approach.
- Solution: Bit Stuffing.

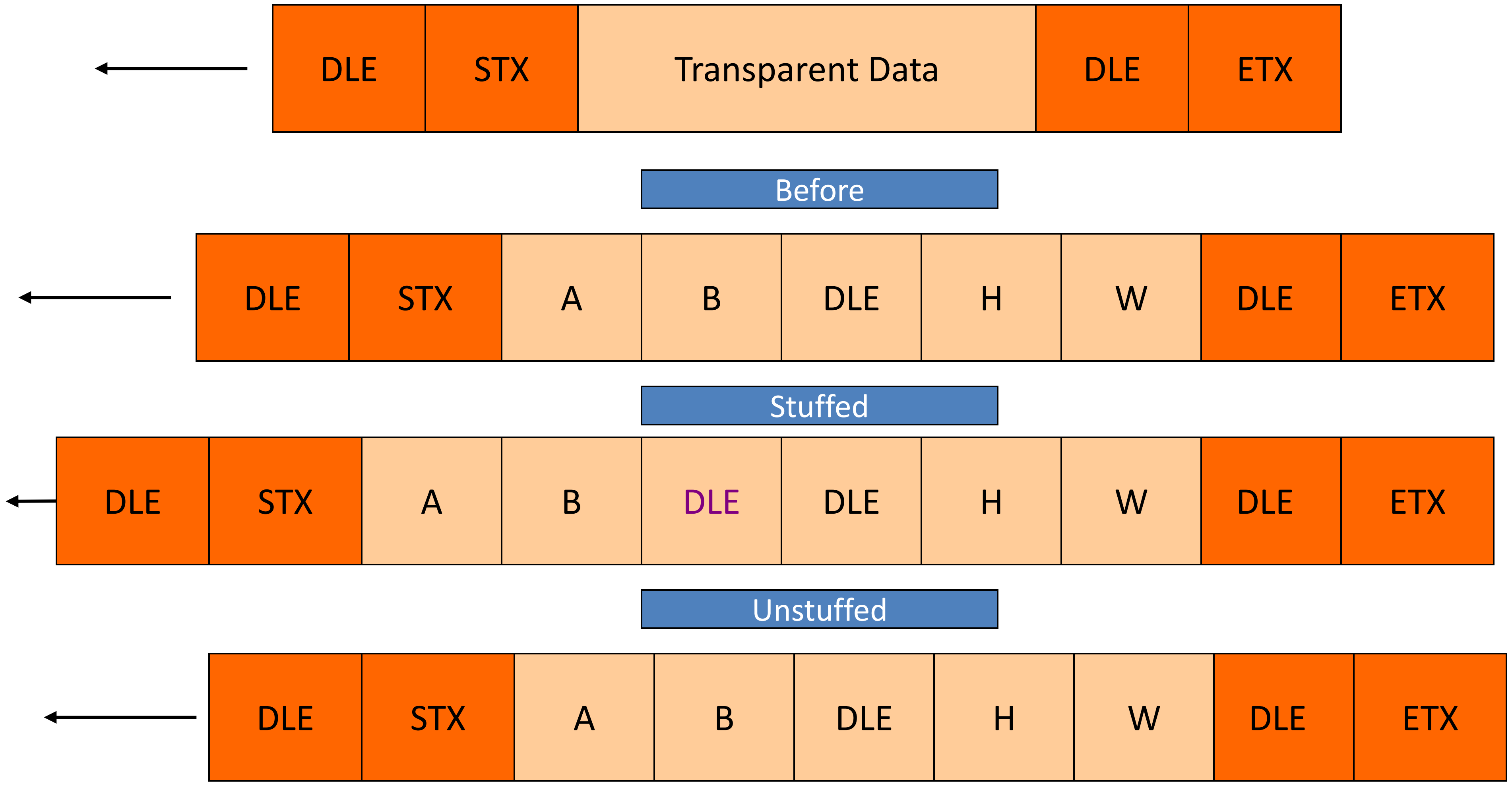
Byte Stuffing

- *Also referred to as <u>character stuffing.</u>*
- ASCII characters are used as framing delimiters (e.g. DLE STX and DLE ETX)
- The problem occurs when these character patterns occur within the "transparent" data.

Solution: sender stuffs an **extra DLE** into the data stream just before each occurrence of an "accidental" DLE in the data stream.

The data link layer on the receiving end unstuffs the DLE before giving the data to the network layer.

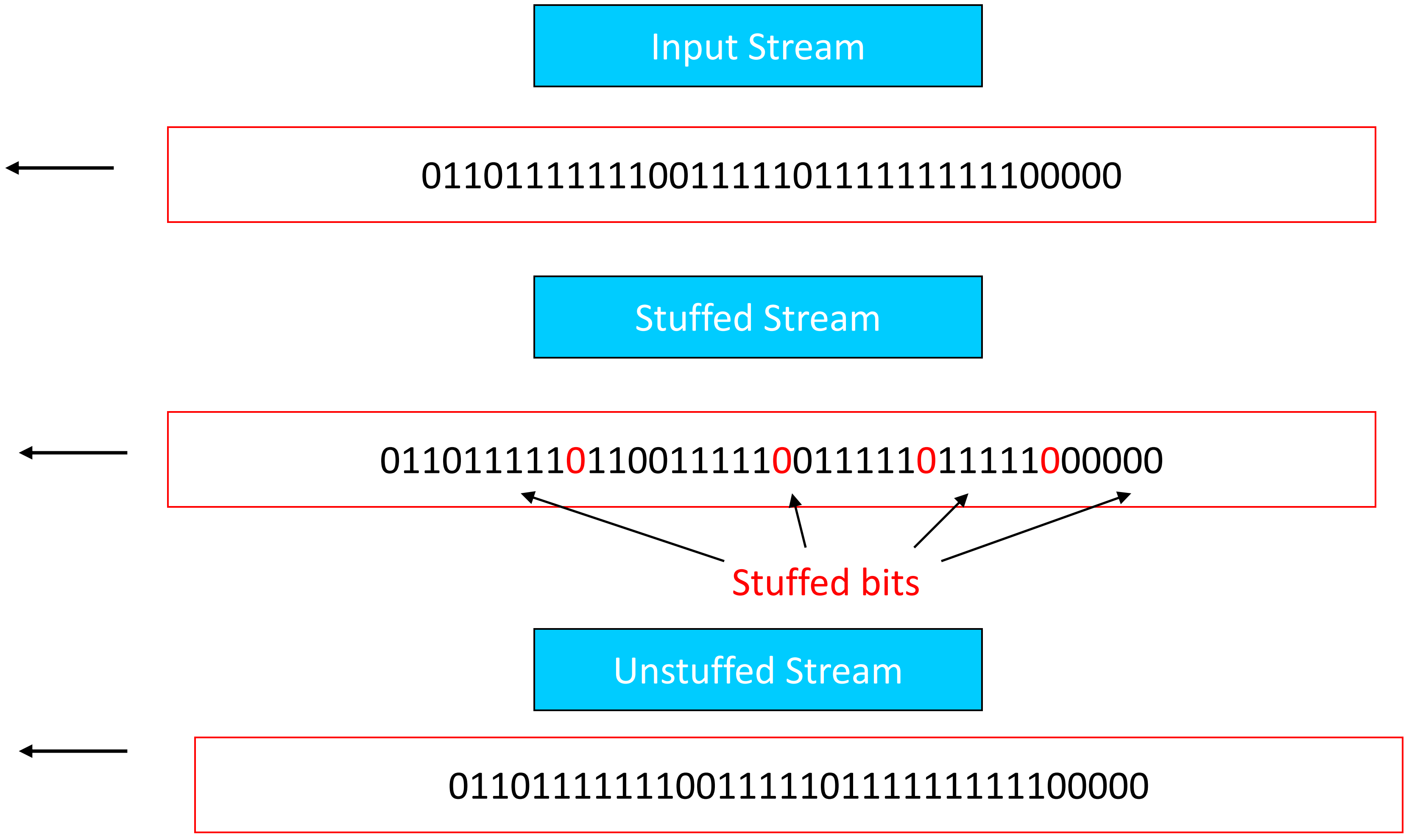
Byte Stuffing (Example)



Bit Stuffing

- Each frame begins and ends with a special bit pattern called a flag byte [01111110]. {Note this is **7E in hex**}
- Whenever sender data link layer encounters *five consecutive ones* in the data stream, it automatically stuffs a 0 bit into the outgoing stream.
- When the receiver sees *five consecutive incoming ones followed by a 0 bit*, it automatically destuffs the 0 bit before sending the data to the network layer.

Bit Stuffing (Example)



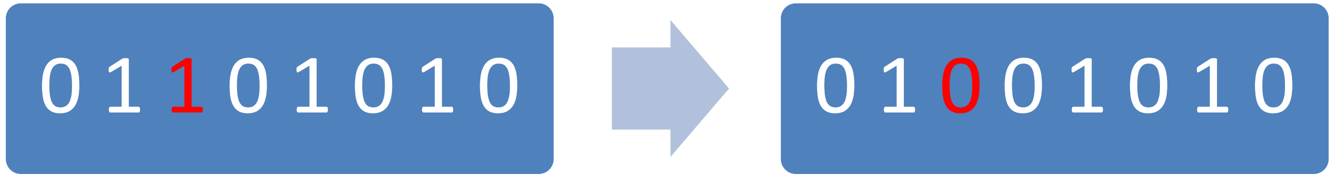
Error Control

- It is possible that data may get corrupted during the transmission because of noise, interference etc. This is known as error in the communication.

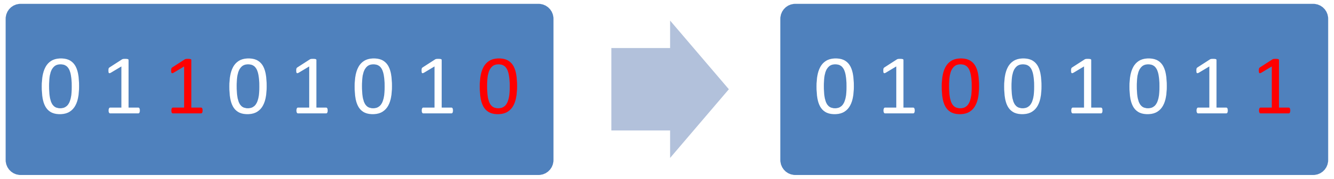
- It is the responsibility of DLL and other higher layer to ensure error free communication.

Error Control (Contd.)

- Error in the communication can be classified as:
  - Single Bit Error

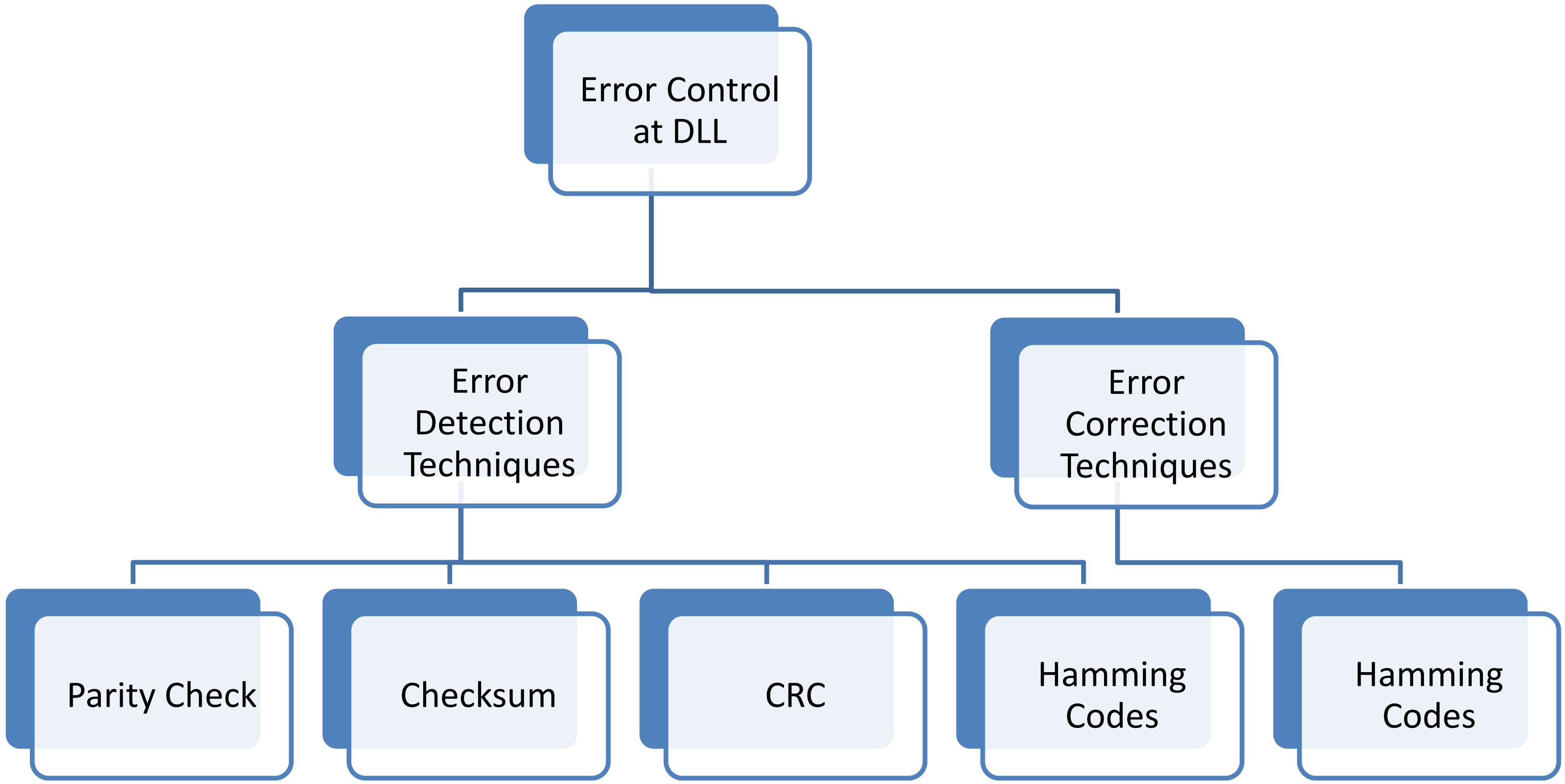


  - Burst Error



Burst Error of 6 bits.

Error Control (Contd.)



Error Control (Contd.)

- Basic Idea of Error Detection
  - To add redundant information to a frame that can be used to determine if errors have been introduced.
  - In general, we can provide strong error detection technique
    - k redundant bits, n bits message, k << n
    - In Ethernet, a frame carrying up to 12,000 bits of data requires only 32-bit CRC.
- Parameter of evaluation
  - Total number of redundant bits requird.
  - Power of error detection / correction.

Error Control (Contd.)

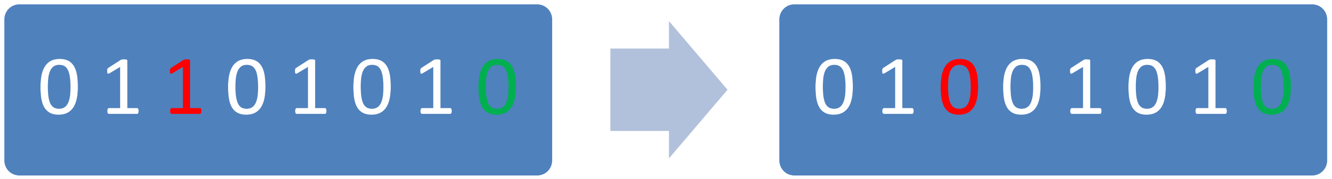
- Extra bits are redundant
  - They add no new information to the message
  - Derived from the original message using some algorithm
  - Both the sender and receiver know the algorithm

Sender
m
r
Receiver
m
r

Receiver computes *r* using *m bits.* If they match, no error else data received has error.

Parity Check (One dimensional parity)

- Algo:
  - Divide the message into words (7 bit word, 15 bit word, 31 bit word etc.)
  - Append a single parity bit at the end to the sequence of message bits.
    - Odd Parity: The parity bit is chosen to make the total number of 1’s in the message odd.
    - Even Parity: The parity bit is chosen to make the total number of 1’s in the message even.
    - Example of even parity: Green bit is the parity bit

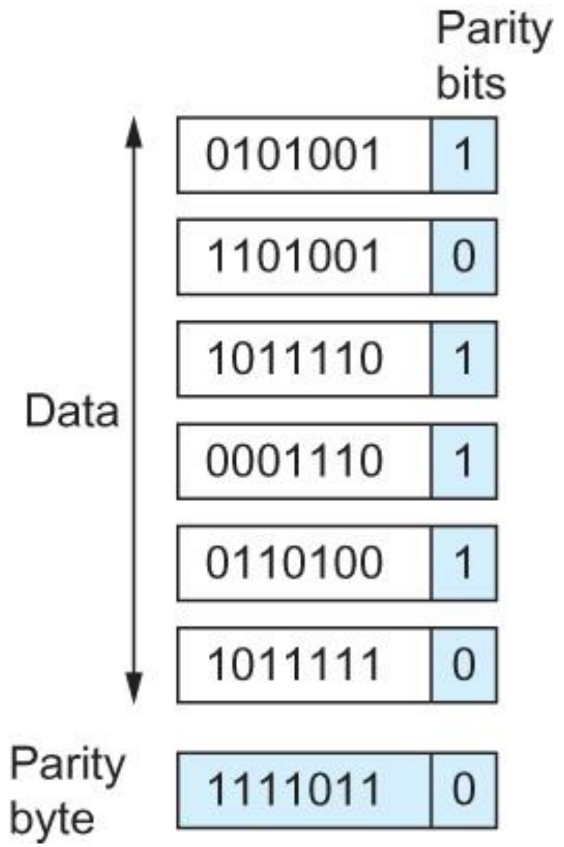


    - Number of redundant bit required is very high.
    - Power of error detection: Can detect if odd number of bits are involved in the error.

Two-dimensional parity check

- Two-dimensional parity is exactly what the name suggests
  - Two-dimensional parity does a similar calculation for each bit position across each of the bytes contained in the frame
  - This results in an extra parity byte for the entire frame, in addition to a parity bit for each byte.

Two-dimensional parity check



Two Dimensional Parity

Number of redundant bits required is very high.

Two-dimensional parity catches all 1-, 2-, and 3-bit errors and most 4-bit errors.

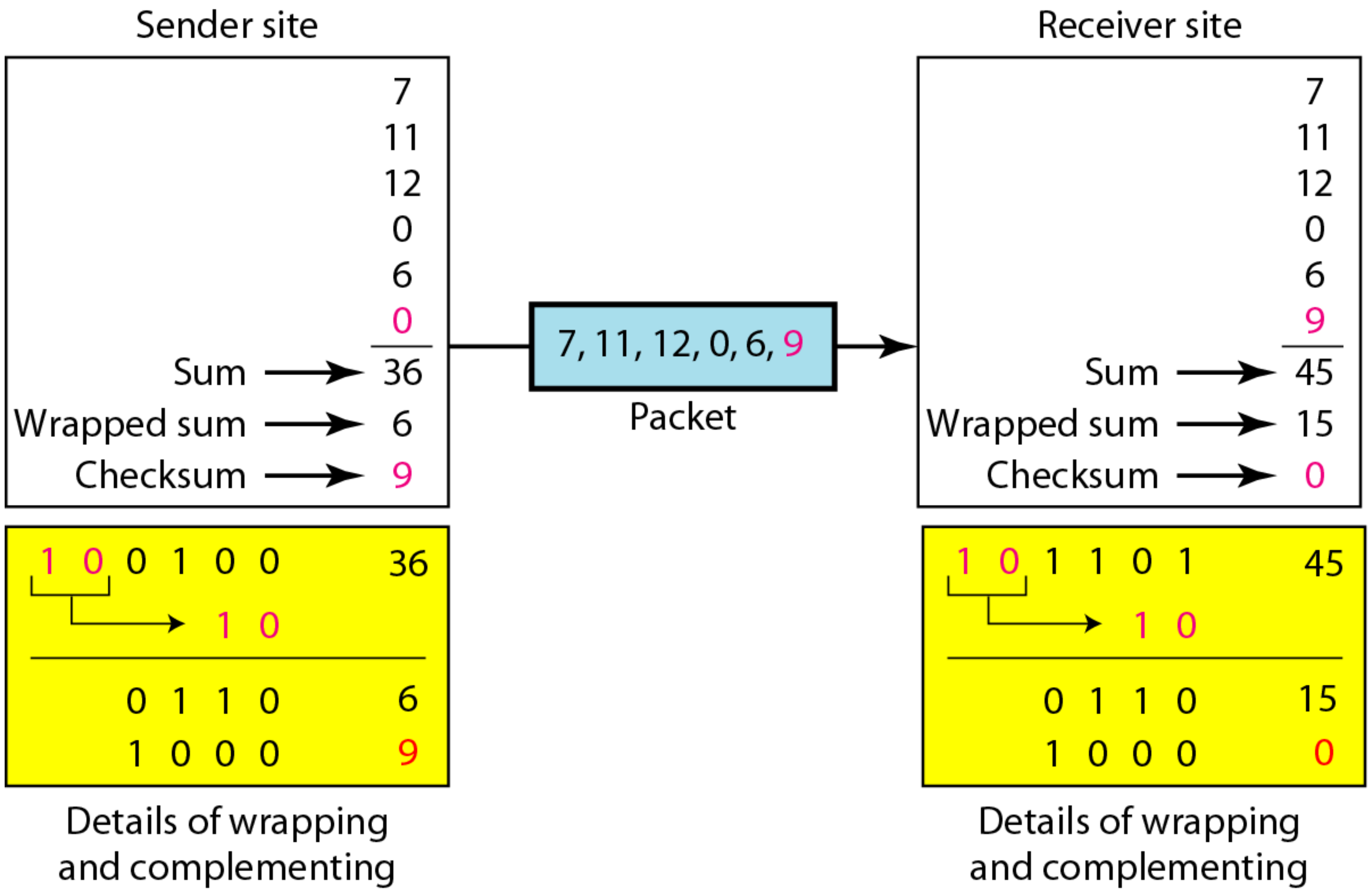
Checksum

- Transmitter’s Algo:
  - 1’s complement arithmetic is used for implementation.
  - Divide the message into words. (8 / 16 / 32 bit words).
  - Calculate the sum of all the units using 1’s complement arithmetic.
  - Take ones complement of the result. This is called the checksum.
  - Transmit the checksum along with the message.
- Receiver’s Algo:
  - Divide the message into words including the checksum. (8 / 16 / 32 bit words).
  - Calculate the sum of all the units using 1’s complement arithmetic.
  - Take ones complement of the result.
  - If the result is all zero then accept else discard.

1’s Complement Arithmetic

- In ones complement arithmetic, a negative integer −x is represented as the complement of x;
  - Each bit of x is inverted.
- When adding numbers in ones complement arithmetic, a carryout from the most significant bit needs to be added to the result.

Example of decimal representation



Checksum Example

- Consider, for example, the addition of −5 and −3 in ones complement arithmetic on 4-bit integers
  - +5 is 0101, so −5 is 1010; +3 is 0011, so −3 is 1100
- If we add 1010 and 1100 ignoring the carry, we get 0110.
- In ones complement arithmetic, the fact that this operation caused a carry from the most significant bit causes us to increment the result, giving 0111, which is the ones complement representation of −8 (obtained by inverting the bits in 1000), as we would expect.
- At receiver's end If we add 1010, 1100 and 1000 we get 1111 as sum.
- Complement of 1111 is 0000. Which means there is no error in the transmission.

Checksum Example

```
   0001
   f203
   f4f5
   f6f7
 +(0000)
 ------
  2ddf0

   ddf0
 +    2
 ------
   ddf2

   220d
```

- Sender's Side

```
   0001
   f203
   f4f5
   f6f7
 + 220d
 ------
  2fffd
    ↓
   fffd
 +    2
 ------
   ffff
    ↓
   0000
```

Receiver's Side

Checksum Properties

- Number of redundant bits are constant and will depends on the word size chosen.

- It can detect all the errors where odd number of bits are involved in the error, most of the 2-bits and most of the 4-bits errors.

Cycle Redundancy Check (CRC)

- Reduce the number of extra bits and maximize protection.
- CRC uses a special class of polynomial arithmetic known as “Polynomial Arithmetic Modulo 2”.
- For the purpose of implementation following properties of Polynomial Arithmetic Modulo 2 are important:
  - Any polynomial B(x) can be divided by a divisor polynomial C(x) if B(x) is of higher degree than C(x).
  - Any polynomial B(x) can be divided once by a divisor polynomial C(x) if B(x) is of the same degree as C(x).
  - The remainder obtained when B(x) is divided by C(x) is obtained by subtracting C(x) from B(x).
  - To subtract C(x) from B(x), we simply perform the exclusive−OR (XOR) operation on each pair of matching coefficients.

Cycle Redundancy Check (CRC)

- $Given\ a\ bit\ string$ 110001 $we\ can\ associate\ a\ polynomial\ on\ a\ single\ variable\ x\ for\ it.$

$$1.x^5 + 1.x^4 + 0.x^3 + 0.x^2 + 0.x^1 + 1.x^0 \ = \ x^5 + x^4 + 1\ and\ the\ degree\ is\ 5.$$

$A\ k - bit\ frame\ has\ a\ maximum\ degree\ of\ k - 1.$

- $Let\ M(x)\ be\ a\ message\ polynomial\ and\ C(x)\ be\ a\ generator\ polynomial.$

Cycle Redundancy Check (CRC)

- Sender’s Algorithm:
- Let r be the degree of code polynomial can call it c(x) (Both sender and receiver know it)
- Append r zero bits at the end of message bits and call it S(x).
- Divide S(x) by C(x) using modulo 2 division to get the remainder R(x).
- Subtract R(x) from S(x) and call it T(x).
- Transmit T(x).

Cycle Redundancy Check (CRC)

- Receiver’s Algorithm:
- Let receiver receive a message and call it M(x).
- Divide M(x) by C(x) to get remainder R(x).
- If R(x) = 0 then accept else discard.

Example (Sender Side)

- At Sender:
- Let M = 1 1 0 0 →$M(x) = x^3 + x^2 \rightarrow S(x) = x^6 + x^5 \Rightarrow 1\ 1\ 0\ 0\ 0\ 0\ 0$
- Let C(x) = $x^3 + x + 1$ →C=1 0 1 1; Degree of C(x) = 3

$$\frac{S(x)}{C(x)}$$

Division:

$$
\begin{array}{r|rrrrrrr}
 & & & & \boxed{x^3} & \boxed{+x^2} & \boxed{+x} & \\
\hline
x^3+x+1 & x^6 & +x^5 & & & & & \\
 & x^6 & + & x^4 & +x^3 & & & \\
\hline
 & & x^5 & +x^4 & +x^3 & & & \\
 & & x^5 & + & x^3 & +x^2 & & \\
\hline
 & & & x^4 & + & x^2 & & \\
 & & & x^4 & + & x^2 & +x & \\
\hline
 & & & & & & \boxed{x} &
\end{array}
$$

= $q(x)$ quotient

dividend

divisor

= $r(x)$ remainder = 010

$$
\begin{array}{r|r}
 & 3 \\
\hline
35 & 122 \\
 & \underline{105} \\
 & 17
\end{array}
$$

$$T(x) = S(x) - R(x) \Rightarrow 1\ 1\ 0\ 0\ 0\ 1\ 0$$

Example (Receiver Side)

At receiver suppose M(x) = 1 1 0 0 0 0 0

C(x) = 1 0 1 1

R(x) = 100

=> error in the data received therefore, discard the frame.

CRC Properties

- Properties of Generator Polynomial
  - Let P(x) represent what the sender sent and P(x) + E(x) is the received string. A 1 in E(x) represents that in the corresponding position in P(x) the message the bit is flipped.
  - We know that P(x)/C(x) leaves a remainder of 0, but if E(x)/C(x) leaves a remainder of 0, then either E(x) = 0 or C(x) is factor of E(x).
  - When C(x) is a factor of E(x) we have problem; errors go unnoticed.
  - If there is a single bit error then E(x) = x^i, where *i* determines the bit in error. If C(x) contains two or more terms it will never divide E(x), so all single bit errors will be detected.
  - Therefore it is important to use code polynomial which satisfy the above properties and maximize the error detection power.

Standard Polynomials used for CRC

- Six generator polynomials that have become international standards are:
  - CRC-8 = x^8+x^2+x+1
  - CRC-10 = $x^{10}+x^9+x^5+x^4+x+1$
  - CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$
  - CRC-16 = $x^{16}+x^{15}+x^2+1$
  - CRC-CCITT = $x^{16}+x^{12}+x^5+1$
  - CRC-32 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

CRC Properties

- In general, it is possible to prove that the following types of errors can be detected by a C(x) with the stated properties:
  - All single-bit errors, as long as the x^k and x^0 terms have nonzero coefficients.
  - All double-bit errors, as long as C(x) has a factor with at least three terms.
  - Any odd number of errors, as long as C(x) contains the factor (x+1).
  - Any “burst” error (i.e., sequence of consecutive error bits) for which the length of the burst is less than *k* bits. (Most burst errors of larger than *k* bits can also be detected.)

Error Correction Technique

- Also known as forward error correction (FEC).
- FEC is used in transmission of radio signals, such as those used in transmission of digital television (Reed-Solomon and Trellis encoding) and 4D-PAM5 (Viterbi and Trellis encoding).
- Some FEC is based on Hamming Codes

Hamming Codes (for 1 bit error correction)

- Suppose the message consists of m bits and number of redundant bits r, required to implement the Hamming Code algorithm.
- How large does r need to be can be calculated by satisfying the following equality for the minimum value of r.

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

Sender’s Algorithm

- Bits are numbered from left to right from 1 to m+r.
- Power of 2 bit positions are reserved for redundant bits.
- Remaining bit position are filled with message bits.
- Each redundant bit will force a parity (even or odd) on the subset of message bits.

Sender’s Algorithm

- Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.

  a. Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).

  b. Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).

  c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).

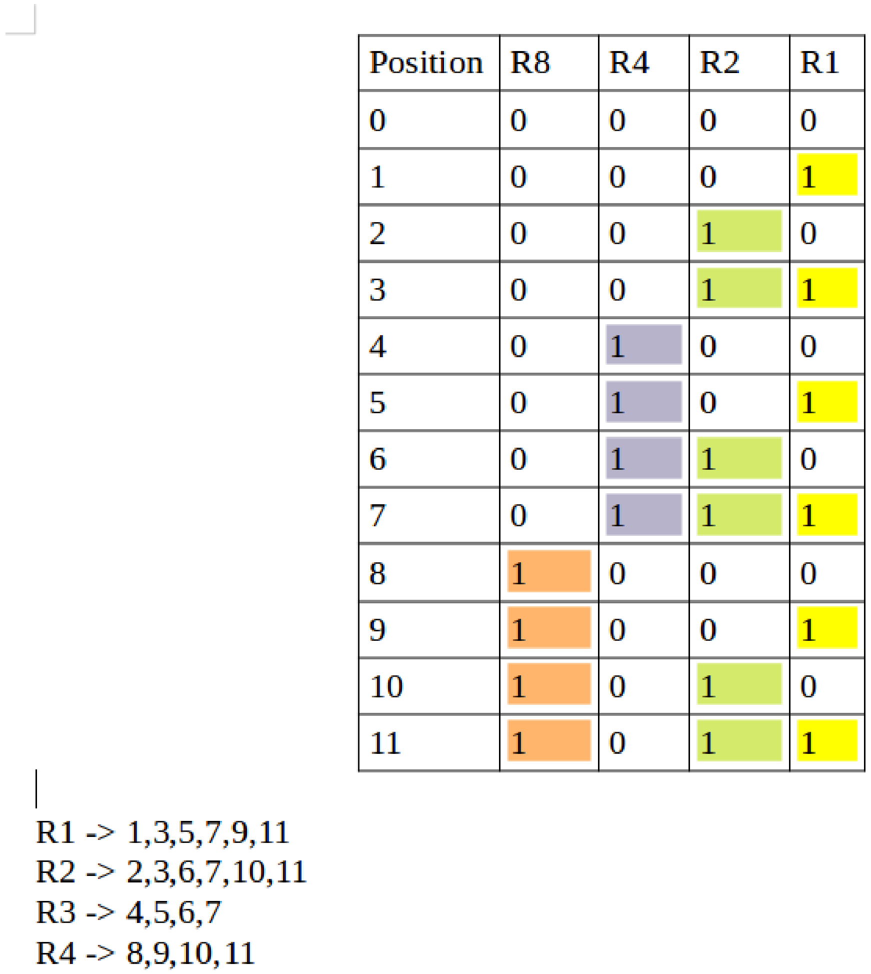
Sender’s Algorithm

d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).

e. In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

- Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
- Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Determination of Subset



Position	R8	R4	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

R1 -> 1,3,5,7,9,11
R2 -> 2,3,6,7,10,11
R3 -> 4,5,6,7
R4 -> 8,9,10,11

Example

- m=1011
- r=3
- m+r = 7

		1		0	1	1
r1	r2	d1	r3	d2	d3	d4
1	2	3	4	5	6	7

Subset of r1= 1, 3, 5, 7

Subset of r2 = 2, 3, 6, 7

Subset of r3 = 4, 5, 6, 7

Decimal Value	r3	r2	r1
	2^2	2^0	2^0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Example

- For implementing odd parity
- r1 = 1
- r2= 0
- r3= 1

- T(x) = 1 0 1 1 0 1 1

Receiver’s Algorithm

- If the parity is satisfied on the subsets then fill 1 in the table otherwise 0. The decimal equivalent of that binary value will give the position of error.
- For example suppose the received message =1 0 1 1 0 0 1

Evaluation of parity of corresponding subset of redundant bits	r3	r2	r1
	2^2	2^0	2^0
	0	1	0

- 0 1 0 => 2 is the position of error.