Networks and IT security Cours

2nd year professional license

Data Link Layer

BENAMEUR Sabrina

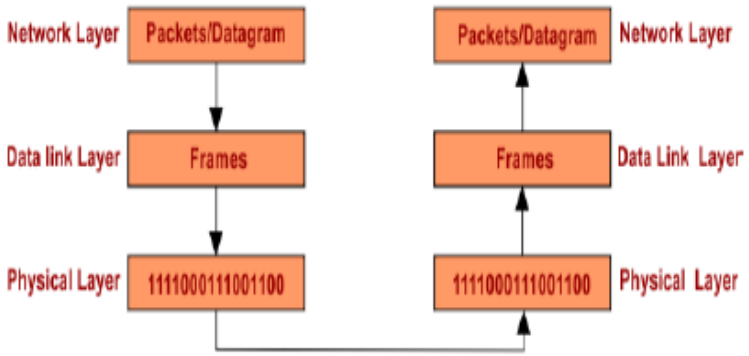
sabrina.benameur@univ-biskra.dz

Department of Computer Science

Mohamed Khider University of Biskra

Data Link Layer

- Physical layer : transmission of bits.
- The data link layer (layer n2) : ensure the correction of transmitted bits.
- The data link layer (layer n2) : retrieves data packets from the network layer, wraps them in frames, sends them one by one to the physical layer.

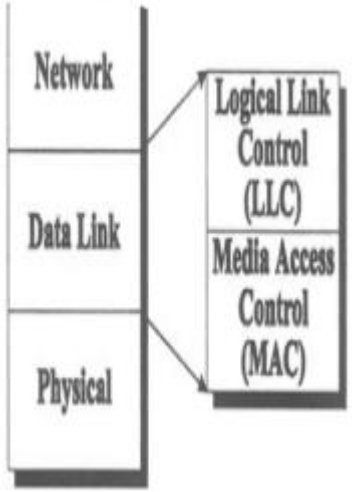


Data Link Layer

- The link layer must ensure :
  - The delimitation of the data blocks exchanged ;
  - Checking the integrity of received data ;
  - The organization and control of the exchange.
  - Shared channel access control

Data Link Layer

- The link layer consists of two sub-layers :
  - LLC (Logical Link Control) and
  - MAC (Media Access Control)



Frame delimitation

- Two methods :
  - Character count
  - Using Flags

Frame delimitation
Character counting

- A field in the frame header is used to indicate the number of characters in the frame.

- Problem : if the value of the added field is modified during transmission !

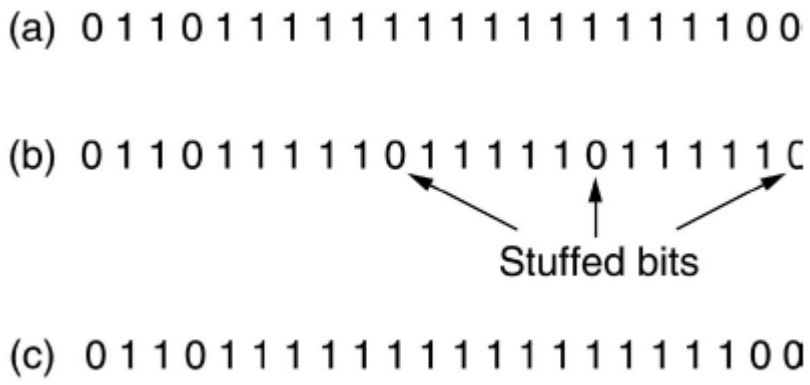
Frame delimitation
Use of Flags

- The frame is delimited by a particular sequence of bits called flag.

- Problem : if the value of the flag appears in the data ! !
- Solution : Binary Transparency (bit stuffing)

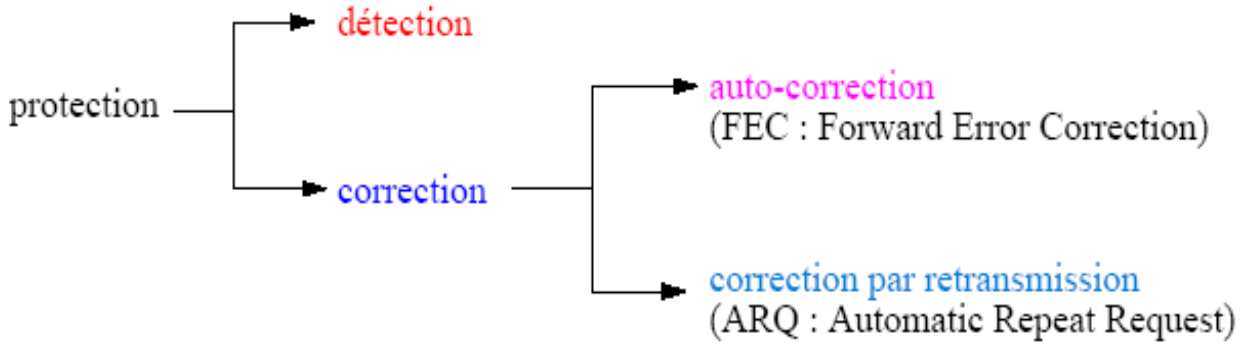
Frame delimitation Using bit stuffing

- Need for transparency bits when the value of the flag (01111110) appears in the data;
- Principle of bit stuffing :
  - On send : add to each five successive bits at 1 a bit at 0
  - On reception : remove for each sequence of five successive bits at 1 a bit at 0
- Advantages :
  - Synchronization always
  - frames of any size.



Transmission reliability

- Regardless of the communication media and transmission techniques used, errors may occur.
- Under these conditions, the binary sequence received will not be identical to the sequence transmitted.
  - Implementation of techniques for protection against transmission errors
- Strategies for protection against transmission errors:



Protection Detection

Principle:

- A transmitter wants to transmit a message (any binary sequence) to a receiver
- The transmitter transforms the initial message using a specific calculation process that generates a certain redundancy of information within the coded message.
- The receiver checks using the same calculation process that the message received is indeed the message sent according to these redundancies.
- Example: the repetition detection technique
  - the coded message is a duplicate copy of the initial message, the receiver knows that there has been an error if the copies are not identical.
- Note: some errors are undetectable!
  - e.g.: the same error on both copies simultaneously

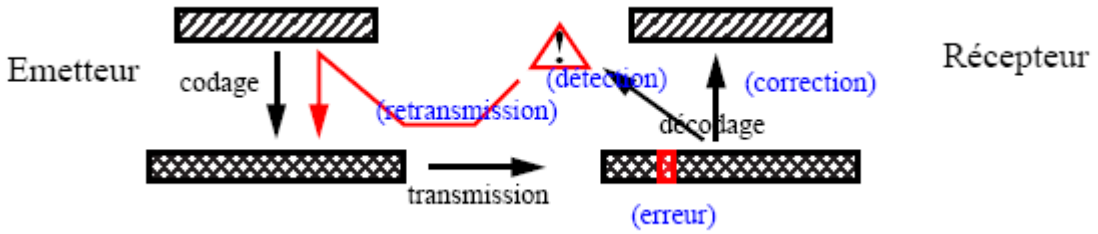
Auto-correction

Principle:

- After detecting an error, the redundancy in the transmitted message is sufficient to allow the initial message to be found.
- Example: the repetition detection technique
  - the coded message is a triple copy of the initial message, the receiver assumes that the initial message corresponds to the two copies which are identical.
  - Some detected errors are not correctable!!
    - e.g.: a different error on at least two copies
  - some errors are detected and incorrectly corrected!!
    - e.g.: the same error on two copies simultaneously

Protection
Correction by retransmission(1)

- After detecting an error, the receiver asks the transmitter, implicitly (timer) or explicitly (nack), to retransmit the message (encoded).
- Example: many telecommunications protocols: HDLC, X25, TCP, IP.



Protection
Correction by retransmission(2)

- Retransmission correction is preferred in networks where the loss rate is low and the retransmission delay is acceptable, because its overhead is generally lower than that induced by auto-correction codes.
- Estimated overhead (le surcoût) (average message length = 1000 bits):
  - typical error rate = 10^{-9} , retransmission rate → 1000. $10^{-9} = 10^{-6}$
  - typical overhead of a auto-correction code: a few bytes per message → 8/1000 ≈ 10^{-2}

Error protection codes

Classification of codes

- Two major families of codes:
  - Block codes: the coding/decoding of a block depends only on the information in this block.
  - Convolutional (or recurrent) codes: the coding/decoding of a block depends on the information in other blocks (generally previously transmitted blocks).
- Block coding is generally preferred in classic teleinformatics applications: the coding/decoding is simpler and there is less delay

General definitions

- The Hamming weight of a word is the number of 1 bits it contains.
- The Hamming distance between two words of the same length is defined by the number of binary positions that differ between these two words.
- It is obtained by the Hamming weight of the binary sum of the 2 words.
- 1 0 0 0 1 0 0 1

$\oplus$ 1 0 1 1 0 0 0 1

= 0 0 1 1 1 0 0 0 ; Dist(x,y) = 3

- The Hamming distance of a code is the minimum distance between all the words of the code.
  - Let C be a code (detector and/or corrector); Dist(C) = min { Dist(x,y) $\mid$ $x \in C \wedge y \in C$ }

Examples of block codes
Parity check(1)

- Even (odd) parity: the Hamming weight of the code words is even (odd)
- It is a systematic code (k, k+1) in which a bit (the parity bit) is added to the initial word to ensure parity. Its efficiency (rendement) is low when k is small.
- Example:
  - Transmission of characters using a representation code (the 7-bit ASCII code).

Lettre	Code ASCII	Mot codé (parité paire)	Mode codé (parité impaire)
E	1 0 1 0 0 0 1	1 0 1 0 0 0 1 1	1 0 1 0 0 0 1 0
V	0 1 1 0 1 0 1	0 1 1 0 1 0 1 0	0 1 1 0 1 0 1 1
A	1 0 0 0 0 0 1	1 0 0 0 0 0 1 0	1 0 0 0 0 0 1 1

- This code is able to detect all odd number errors. It does not detect even number errors!
- It can detect a parity error, but not locate it.

Examples of block codes
Parity check(2)

- Example: even parity

	Without error	With 1 error	With 2 errors	With 3 errors
We want to transmit	1000001	1000001	1000001	1000001
Calculating the parity bit	01000001	01000001	01000001	01000001
Transmission	01000001	01000001	01000001	01000001
Reception	01000001	01010001	01010000	01001111
Delete parity bit	1000001	1010001	1010000	1001111
Calculation of parity bit	01000001	11010001	01010000	11001111
Les nombres sont égaux	OK	Error	OK	Error

Examples of block codes
Longitudinal and vertical parity (1)

- For each character, a bit is added (vertical redundancy bit or parity bit, VRC: Vertical Redundancy Check).
- When the number of bits at 1 is even, an even parity code is used (for asynchronous transmission), otherwise it is odd parity (for synchronous transmission).
- For each character block, an additional control field is added (LRC: Longitudinal Redundancy Check)

Examples of block codes
Longitudinal and vertical parity (2)

- The data block is arranged in a matrix form (k=a.b).
- Parity is applied to each row and each column. We obtain a matrix (a+1, b+1).

	VRC (parité paire)
	1 0 1 0 0 0 1 1
	0 1 1 0 1 0 1 0
	1 0 0 0 0 0 1 0

LRC =	0 1 0 0 1 0 1 1

- The yield (le rendement) is very low:

  a.b/ (a+1).(b+1).

Examples of block codes
Longitudinal and vertical parity (3)

- The combination of LRC and VRC allows to detect 2 bit errors in a single word or to correct 1 error.
- Detection and self-correction capacity:
  - Principle: A single error simultaneously modifies the parity of a row and a column.
  - Correction: invert the bit located at the intersection of the row and the column with incorrect parity.
- Example:

VRC (parité paire)

1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
1 0 0 0 0 0 1 0
---
0 1 0 0 1 0 1 1

1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
1 0 0 0 1 0 1 0
0 1 0 0 1 0 1 1

Hamming Code

- The Hamming code is used to detect and correct an error that occurs in a transmitted block.
- To the d data bits, we add c parity check bits.
- We therefore have d+c = n bits..

Hamming Code

- The bits of each code word are numbered from right to left starting at 1, the control bits are placed in positions representing powers of 2 (1,2,4,8,...) and the data bits are interspersed (intercalés).
- Example:

					1	0	0	1	0	0	0
							+	c_3	c_2	c_1	c_0
=	1	0	0	c_3	1	0	0	c_2	0	c_1	c_0
	11	10	9	8	7	6	5	4	3	2	1

Hamming Code

- The list of parity bits controlling a given bit is provided by the 1 positions in its binary representation. For example:
  - bit number 11 = (1011)2 is controlled by bits c3, c1 and c0
  - bit number 7 = (0111)2 is controlled by bits c2, c1 and c0
  - bit number 4 = (0100)2 is controlled by bits c2
  - ...
- Thus, the list of bits controlled by each parity bit is:
  - c0 : {1,3,5,7,9,11}
  - c1 : {2,3,6,7,10,11}
  - c2 : {4,5,6,7}
  - c3 : {8,9,10,11}

Hamming Code

- In the previous example and for an even parity, the ci can be calculated as follows:
  - c0 = 0(0 + 0 + 0 + 1 + 0 + 1)
  - c1 = 0(0 + 0 + 0 + 1 + 0 + 1)
  - c2 = 1(1 + 0 + 0 + 1)
  - c3 = 1(1 + 0 + 0 + 1)
- And the code word sent will therefore be: 10011001000
- On reception, a binary word is constructed of length c which can be deduced from the length of the received code word (to encode 11 positions we need 4 bits). Each bit is equal to 1 if the parity check is OK and 0 otherwise. If the word is equal to 0, we conclude that there was no error, otherwise the value of the word represents the number of the erroneous bit.

Hamming Code

- For example, if we receive 10011001000, we calculate the ci:
  - c0 = 0+0+0+1+0+1 = 0
  - c1 = 0+0+0+1+0+1 = 0
  - c2 = 1+0+0+1 = 0
  - c3 = 1+0+0+1 = 0
- We therefore conclude that the word received is correct.
- If, on the other hand, we receive 11011001000, we find:
  - c0 = 0+0+0+1+0+1 = 0
  - c1 = 0+0+0+1+1+1 = 1
  - c2 = 1+0+0+1 = 0
  - c3 = 1+0+1+1 = 1
- The control word is therefore (1010)2 = (10)10, we conclude that bit number 10 is erroneous.

Hamming Code

Simplified calculation of the Hamming code

- Code 10101011001 with even parity: d = 11, therefore k = 4. The message to be transmitted therefore contains n = 15 bits and is equal to:

1	0	1	0	1	0	1	?	1	0	0	?	1	?	?
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

- In the message to be transmitted, we have bits at 1 in the following positions: 15, 13, 11, 9, 7, 3. We transform these positions into their binary value and add them modulo 2: We put 1 when we have an odd number of 1s and 0 for an even number of 1s.

15	=	1	1	1	1
13	=	1	1	0	1
11	=	1	0	1	1
9	=	1	0	0	1
7		0	1	1	1
3	=	0	0	1	1
c_i		0	1	0	0

- The coded message is therefore 101010101001100.
- The same process can be done at the reception to verify the received word.

Redundant cyclic codes

- These codes represent the message using a polynomial.
- The general formula for a binary message is a polynomial of the form:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots\ldots + a_2 x^2 + a_1 x^1 + a_0 x^0$$

- The coefficients a_n can only take the values 0 or 1 in binary, and the x^n are successive powers of 2.
- If we divide this polynomial by another binary polynomial, the remainder of the division constitutes the check character.

Redundant cyclic codes

- CRC (Cyclic Redundancy Check) Principle
  - We choose a polynomial G(x) of degree r: ex: $G(x) = x^4 + 1$
  - Let m be the message to send, we transform it into m(x)
  - We calculate: $M(x) = x^r \; m(x)$
  - $M(x) = Q(x)\, G(x) + R(x)$
  - $T(x) = M(x) + R(x)$
  - Sends the message T corresponding to the polynomial T(x)

Redundant cyclic codes

- How does the receiver know if there has been an error or not?
- If the remainder of the division of T(x) by G(x) is zero,
  - it means that there have been no errors,
- otherwise there has been an error
- $M(x) = Q(x)\,G(x) + R(x)$
- $T(x) = Q(x)\,G(x) + R(x) + R(x) = Q(x)\,G(x)$

Redundant cyclic codes

Example

- Let the message 1011011 be transmitted.
- The polynomial corresponding to the form:
  - $P(x) = x^6 + x^4 + x^3 + x^1 + x^0$
- If the divisor (generator) polynomial is equal to: $x^4 + x + 1$
- Which gives in binary: 10011
- We must therefore divide 1011011 by 10011.

Redundant cyclic codes

- This division is performed as follows:
- We add to the dividend, to its right, as many 0s as the divisor has bits minus one (here 5-1 => 4)
  - which gives 10110110000.
- Then we perform the division by successive subtractions (subtractions using the exclusive-OR) "Arithmetic modulo 2 (i.e. without carryover)"

```
 1 0 1 1 0 1 1 0 0 0 0
-1 0 0 1 1
 0 0 1 0 1 1 1
   -1 0 0 1 1
     0 0 1 0 0 0 0
       -1 0 0 1 1
     R: 0 0 0 1 1 0 0
```

Redundant cyclic codes

- Using polynomial division:

$$P(x) = x^6 + x^4 + x^3 + x^1 + x^0$$

- If the divisor polynomial is equal to: $x^4 + x + 1$

$$\begin{array}{l|l}
x^{10} + x^8 + x^7 + x^5 + x^4 & x^4 + x + 1 \\
\underline{x^{10} + x^7 + x^6} & \\
x^8 + x^6 + x^5 + x^4 & x^6 + x^4 + x^2 \\
\underline{x^8 + x^5 + x^4} & \\
x^6 & \\
\underline{x^6 + x^3 + x^2} & \\
x^3 + x^2 &
\end{array}$$

Redundant cyclic codes

- The last line of the division will give the remainder, therefore the CRC; it is this which will be added to the message which we are going to transmit.

M: 1 0 1 1 0 1 1

$M(x) = x^6 + x^4 + x^3 + x^1 + x^0$

$G(x) = x^4 + x + 1$

$R(x) = x^3 + x^2$

T: 1 0 1 1 0 1 1 **1 1 0 0**

- At reception, a similar calculation is performed on the received word,
  - but here the remainder must be zero.
  - Otherwise, an error has occurred along the way.

Redundant cyclic codes

- To carry out the verification we do as follows:

a) Example without error:

```
  1 0 1 1 0 1 1 1 1 0 0
-1 0 0 1 1
---------
  0 0 1 0 1 1 1
    -1 0 0 1 1
    ----------
      0 0 1 0 0 1 1
        -1 0 0 1 1
        ----------
          0 0 0 0 0 0 0
```

Redundant cyclic codes

- At reception we have:

  T= 1 0 1 1 0 1 1 1 1 0 0 => T(x) = $x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2$

- If the divisor polynomial is equal to: $x^4 + x + 1$

$$x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 \quad \Big| \quad x^4 + x + 1$$

$$x^{10} + x^7 + x^6 \quad \Big| \quad x^6 + x^4 + x^2$$

$$x^8 + x^6 + x^5 + x^4 + x^3 + x^2$$

$$x^8 + x^5 + x^4$$

$$x^6 + x^3 + x^2$$

$$x^6 + x^3 + x^2$$

$$0$$

Redundant cyclic codes

b) Example with error:

1 0 1 1 0 1 0 1 1 0 0

- 1 0 0 1 1

0 0 1 0 1 **1 0**

- 1 0 0 1 1

0 0 1 0 1 **1 1**

- 1 0 0 1 1

0 0 1 0 0 0 0

- If the remainder is different from 0, the transfer was not successful.

Redundant cyclic codes

- At reception we have:

  T= 1 0 1 1 0 1 0 1 1 0 0 => T(x) = $x^{10} + x^8 + x^7 + x^5 + x^3 + x^2$

- If the divisor polynomial is equal to: $x^4 + x + 1$

$x^{10} + x^8 + x^7 + x^5 + x^3 + x^2$ | $x^4 + x + 1$

$x^{10} + x^7 + x^6$ | $x^6 + x^4 + x^2$

$x^8 + x^6 + x^5 + x^3 + x^2$

$x^8 + x^5 + x^4$

$x^6 + x^4 + x^3 + x^2$

$x^6 + x^3 + x^2$

x^4

Note: The position of the error is indicated by the division remainder

Redundant cyclic codes

- The main divisor polynomials are:

LRCC-8 : $x^8 + 1$
LRCC-16 : $x^{16}+1$
CRC 12 : $x^{12}+x^{11}+x^3+x^2+x+1$

CRC 16 Forward : $x^{16}+x^{15}+x^2+1$
CRC 16 Backward : $x^{16}+x^{14}+x+1$
CRC CITT Forward : $x^{16}+x^{12}+x^5+1$
CRC CITT Backward : $x^{16}+x^{11}+x^4+1$

Communication procedures (protocols)

1. Data Link Management
2. The BSC Procedure
3. Bit-Oriented Procedures: The HDLC Procedure

Data Link Management (1)

- Communication procedures are level 2 software
- They are responsible for the error-free transmission of information blocks over one or more physical links (level 1)

Data Link Management (2)
Data Link Layer Functions

- Establishment and release of the data link on previously activated physical connections;
- Detection of transmission errors and activation of error recovery procedures;
- Higher level warning in case of unrecoverable error.

Data Link Management (3)
Data Link Layer Functions

- Supervision of the operation of the data link, according to
  - The transmission mode (synchronous or asynchronous),
  - The nature of the exchange (unidirectional, bidirectional alternating or simultaneous),
  - The type of link (point to point, multipoint, loop, etc.);
- Definition of
  - The syntactic structure of valid messages,
  - How to link emissions and receptions according to a specific protocol.

Data Link Management (4)

We distinguish 2 main families of procedures:

- Character-oriented procedures (for example BSC developed by IBM)
  - They generally operate alternately and are therefore send and wait type procedures.
- Bit-oriented procedures (such as HDLC)
  - They were designed for simultaneous bidirectional transmissions at high speeds.

The BSC procedure

- The BSC (Binary Synchronous Communications) procedure studied and implemented from the 60s.

- BSC is based on a synchronous transmission of character blocks, whatever the coding used (ASCII, EBCDIC, …).

The BSC procedure
Type of links used

The BSC procedure mainly allows two types of links to be used:

- Point-to-point links:
  - the stations communicate in half duplex mode,
  - In the event of conflicts, the primary station takes control of the secondary station;
- Multipoint links:
  - the primary station represents the central node in relation to the secondary stations.

The BSC procedure
Multipoint connections

- On multipoint links, data transfer has two modes, depending on the direction of data transfer:
  1. Polling mode
  - The primary station invites the secondary stations to transmit, each in turn.
  - The polled station responds with a message if it has something to transmit, otherwise it responds negatively.

The BSC procedure
Multipoint connections

2. The selecting or addressing mode.
   - The primary station wants to send data to a secondary station.
   - First, the primary station sends a selection message to the secondary station to inform it that it wants to send a message;
   - the secondary station responds positively if it is ready to receive, negatively if it is not ready.
   - In the case of a positive response, the primary station sends its data message. This will be acknowledged or refused by the secondary station.

The BSC procedure Characters used (1)

- The message encoding used by BSC is based mainly on the main international alphabets: the 7-bit ASCII code or the 8-bit EBCDIC code
- These codes are rich enough to allow the use of special characters for data link management.
- Some characters have a universally recognized function, such as:

The BSC procedure Characters used (2)

- **SYN (SYNchronous idle):**
  - used to ensure character synchronization.
  - the transmitter begins by sending a sequence of SYN characters (prefix sequence) before any message or supervision sequence.
- **ENQ (ENQuiry):**
  - placed in a supervision sequence. It invites a station to transmit or receive.
- **SOH (Start Of Heading):**
  - signals a start of header.

The BSC procedure Characters used (3)

- **STX** (Start of TeXt) :
  - delimits both the end of the header (when it exists) and the start of the text.
- **ETB** (End of Transmission Block) :
  - can be used to distinguish an end of a data block from an end of a message.
- **ETX** (End of TeXt) :
  - indicates the end of a text and the start of control characters used for error detection.

The BSC procedure Characters used (4)

- **ACK** (ACKnowledgement) :
  - appears in a sequence sent by the receiver to positively acknowledge the message he has just received.
- **NAK** (Négative AcKnowledgement) :
  - is in a sequence sent by the receiver to reject the message it has just received.
  - This message must then be re-sent.

The BSC procedure Characters used (5)

- **DLE** (Data Link Escape) :
  - used during transparent mode transmissions (when all configurations of the transmission alphabet, including procedure characters, are likely to be found in the data field).
  - indicates that the following character must be interpreted as a procedure character and not as a data character (STX, ETB, ETX...).
  - When the character to be transferred is itself DLE, two consecutive DLEs must be sent (known as a technique of doubling the DLE).

The BSC procedure Characters used (6)

- **EOT** (End Of Transmission) :
  - signals the end of a data transfer,
  - It is used in particular in multipoint configurations to indicate the end of exchanges between a secondary station and the primary station.
  - The sequence containing this character causes a general return to listening of the secondary stations of the data link.

The BSC procedure: Character synchronization and padding (1)

- In BSC, character synchronization consists of a series of "SYN" characters.
- Since messages are of arbitrary length, the transmitter must send, every second,
  - The sequence "SYN SYN" in normal mode,
  - "DLE SYN" in transparent mode, to avoid a possible loss of synchronization between transmitter and receiver.

The BSC procedure
Character synchronization and padding(2)

- At the beginning of the transmission, the number of SYN characters is greater than 2 (most often 4 to 7).
- The padding sequence (PAD) prevents the transmission from being cut off before the last character of the block is sent.
- It is composed of either a full byte 1, or an alternation of "0" and "1".

The BSC procedure: Timing

- In a half-duplex transmission, the receiver is always supposed to respond after a finite time
  - (for example, a receiving station must respond within 3 seconds, regardless of the operating mode, point-to-point or multipoint).
- In order to protect against any failure of the link or the correspondent, a timer is set at the end of which (time out) the decision is made to switch to degraded mode
  - (in general, 2 to 3 attempts are made before declaring the link out of service).

The BSC procedure Message format (1)

- The message formats used by BSC are based on the concept of a block.
- A message is a data field of any size, which can be divided into blocks of optimal size.
- BSC allows the transmission of many types of messages in normal mode:
  - header only: SYN, SOH, …{header}..., ETX, BCC, PAD ;
    (BCC: Block Check Character)
  - A single message without header: SYN, STX, ... ETX, BCC, PAD ,
  - A single message with header: SYN, SOH, ..., STX,…, ETX, BCC, PAD ,

The BSC procedure Message format(2)

- multiple blocks for the same message, without header:

  SYN, STX,…,ETB, BCC, PAD for the first blocks, SYN, STX,..., ETX, BCC, PAD for the last block of the message;

- multiple blocks for the same message, with header:
- SYN, SOH, ..., STX,…,ETB, BCC, PAD for the first blocks,
- SYN, SOH, ..., STX, ..., ETX, BCC, PAD for the last block.

The BSC procedure
Error detection

- Error detection depends on the transmission mode,
- Several techniques are used in BSC:
  - transversal parity (odd parity) and longitudinal (LRC) for ASCII in normal mode.
  - For EBCDIC and ASCII in transparent mode, use of the normalized generator polynomial V41 ($x^{16}+x^{12}+x^{5}+1$), defined as a CRC 16, i.e. a cyclic code on 16 bits.

Bit-oriented procedures: The HDLC procedure

- These procedures were designed to overcome the shortcomings of character-oriented procedures:
  - Specificity of a type of application,
  - use of a determined transmission alphabet,
  - transmission in alternation, etc.
- HDLC (High level Data Link Control) defines a set of procedure classes, standardized by ISO.
- The data link is made between a primary and one (or more) secondary.
- Transmissions can be point-to-point or multipoint, in alternation or in full duplex.

The HDLC procedure
Operating modes of secondary stations

There are three operating modes for the secondary:

- Normal Response Mode (NRM)
  - The secondary station must wait for an explicit command from the primary before it can transmit.
- Asynchronous Response Mode (ARM)
  - The secondary station is allowed to transmit data without waiting for the primary's invitation.
  - There is a possibility of conflict at link initialization, if both the primary and secondary want to transmit data simultaneously.
  - It assumes that both stations have both primary and secondary status.

The HDLC procedure
Operating modes of secondary stations

- The balanced or symmetrical asynchronous response mode (Asynchronous Balanced Mode or ABM).
  - The link is necessarily point-to-point;
  - The two stations have both primary and secondary status.
  - This mode of operation is known as LAP-B (Link Access Protocol-Balanced).

The HDLC procedure
Frame format

Flag	Address	control	Information	FCS	flag

- Flag (8 bits = 01111110) .
- Address (8 bits).
- Control (8 bits).
- Information (>= 0 bit).
- FCS (16 bits).
- Flag (8 bits = 01111110).

Opening and closing flags

- They are used to delimit the frame and ensure synchronization.
- A single flag can close a frame and open the next one.
- When there is no data to transmit, a continuous series of flags can be sent in order to keep the link open.
- To prevent a binary configuration from being understood as a flag, there is a mechanism on transmission to insert a 0 after five consecutive 1s for all bits between the flags (transcoding).

The address field

- Identification of the secondary station involved in the exchange
- Command frame: the destination secondary station
- Response frame: the transmitting secondary station

The control field

- It identifies the frame type and the frame number. There are three types of frames, which are command and/or response frames:
  - Information frames (I): whose INFO field contains the data coming from (to) the upper layers
  - Supervisory frames (S): which allow the supervision of the data link
  - Unnumbered frames (U): which are mainly used for initialization and for error recovery problems that cannot be recovered at level 2.

	1	2	3	4	5	6	7	8
Information	0	Ns			P/F	Nr		
Supervision	1	0	S	S	P/F	Nr		
Unnumbered	1	1	M	M	P/F	M	M	M

The HDLC procedure
Frame format

The protocol uses a window mechanism with 3-bit sequence numbers:

- N(S) is a modulo 8 counter which is the sequence number, from 0 to 7, of the transmitted I frame;
- N(R) is also a modulo 8 counter which means that the frames up to R-1 have been received. In the absence of frames received, the N(R) field remains at 0.
- P/F (Poll/Final bit): This bit has the P direction for primary stations and the F direction for secondary stations. Its use depends on the response mode of the secondary station.
- S and M are the bit patterns that differentiate respectively the different supervision frames (4 different types) and unnumbered frames (32 different types). It is called the type field.

Supervision Frames

S	S	Command	Meaning
0	0	RR (Receiver Ready)	The station is ready to receive the frame number Nr and positively acknowledges the reception of frames up to (Nr - 1)
0	1	RNR (Receiver not Ready)	The station is not ready to receive frames but and positively acknowledges the reception of frames up to (Nr - 1)
1	0	REJ (Reject)	The station rejects the frames from number Nr. The sender is obliged to retransmit (P/F = 1)
1	1	SREJ (Reject)	= REJ but only for frame number Nr.

Unnumbered frames

Frame	Command	Meaning
11111100	SABM	Set ABM requests establishment in ABM mode
11110000	DM	Disconnect Mode indicates station is located in offline mode
11001010	DISC	Disconnect releases link
11000110	UA	Unnumbered Acknowledge indicates reception acceptance of an unnumbered order

The information field

- It includes the data to be transmitted in the form of binary sequences,
- The number of bits in this field is arbitrary, it depends on the hardware used. In practice, it is a multiple of 8 because the data is made up of bytes.
- This optional field can only be present in I frames and some U frames.

The FCS (Frame Check Sequence)

- This is the 16-bit frame control field. It is used to detect information that has been transmitted incorrectly within a frame.
- It is obtained from the principle of division, by a polynomial generating the content of the address, command and information fields.