Wireless Communications

3. Data Link Layer

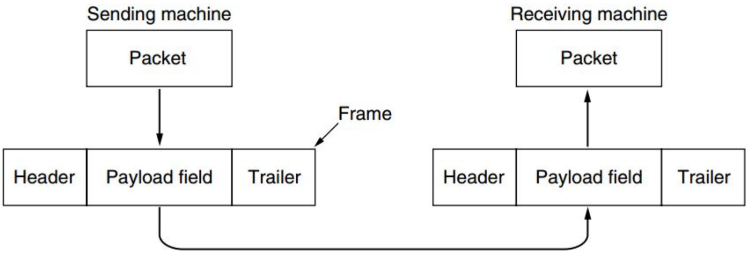
Elvio J. Leonardo

DIN/CTC/UEM

2018

Main Functions

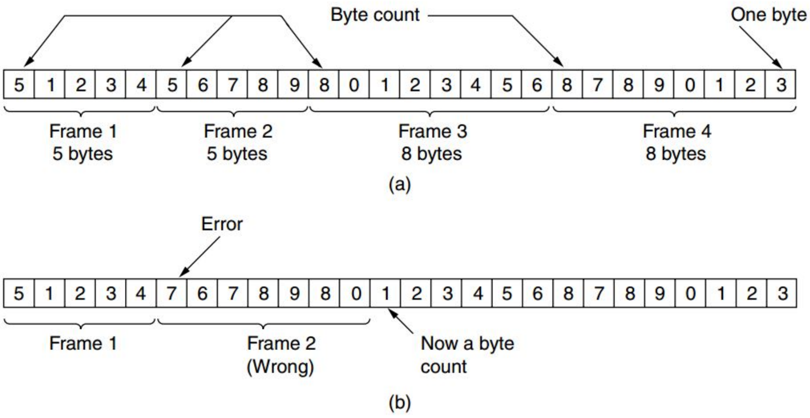
- Handle transmission errors
- Adjust the data flow
- Framing:



Main Functions

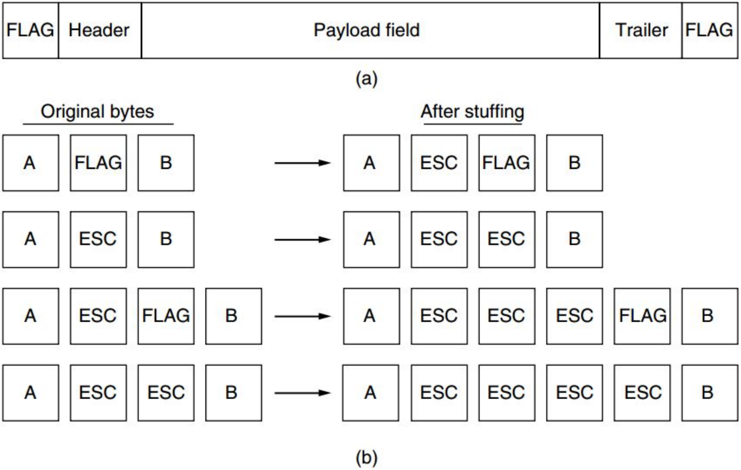
- Split information into frames:
  - Check if frames have arrived correctly
  - Otherwise:
    - Discard frame and
    - Request frame retransmission (not of all the information)
- Services offered to the Network Layer:
  - Service without connection and without confirmation
  - Service without connection but with confirmation
  - Connection-oriented service with confirmation

Frame with Counter



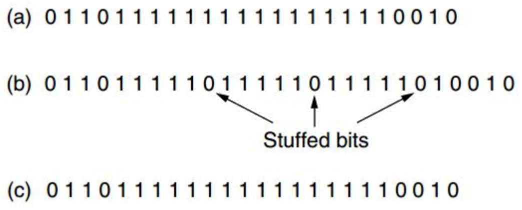
A byte stream... (a) without errors (b) with one error.

Frame with Byte Stuffing



Frame with Bit Stuffing

- Allows characters with any number of bits
- Each frame starts and ends with a standard sequence
  - Example: 01111110
- If the pattern happens in the data field, bits are included to break the sequence



(a) Original data. (b) Sequency after stuffing. (c) Sequency after destuffing.

Error Detection and Correction

- There are error detection codes and error detection and correction codes
- Code adds redundancy
  - Let k be the number of information bits
  - Let r be the number of bits added for protection
  - Let $n = k + r$ be the total number of bits
  - Then, of the 2^n possible words, only 2^k is used
- Hamming Distance $= D_H(w_1, w_2)$
  - Number of different bits between words w_1 and w_2
- The Hamming Distance of a code D_H is the smallest distance $\min[D_H(w_i, w_j)]$ for any two i, j codewords
- For the detection of n incorrect bits
  - It is needed $D_H \geq n + 1$
- For the correction of n incorrect bits
  - It is needed $D_H \geq 2n + 1$

Parity

- Adds one bit to the information word so that the number of bits with value 1 in the transmitted word is even (or odd)
  - Example: $10101 + 1 = 101011$ (for even parity)
  - Example: $10101 + 0 = 101010$ (for odd parity)
- At the receiver, the received information is wrong when the number of bits with value 1 in the received word does not correspond to the one expected
- The Hamming Distance of this code is $D_H = 2$
  - It is able to detect errors in one bit
  - It is unable to correct any error

Repetition

- Repeats every bit n times
- Example (with $n = 3$)
  - Coding: $1 + 11 = 111$
  - At receiver, if bits do not appear in groups of three, it means error
  - The Hamming Distance of the code is $D_H = 3$
    - It is able to detect errors in two bit
    - It is able to detect errors in one bit
  - Examples for the transmission of 111:
    - Error in one bit: receive 110 and it is correctly assumed that 111 was transmitted
    - Error in two bits: receive 100 and it is incorrectly assumed that 000 was transmitted
- Inefficient code (in the example, it reduces the bit flow to a third giving only $D_H = 3$)

Hamming Codes

- Block codes created by Richard Hamming, with distance $D_H = 3$ (thus correcting errors in one bit and detecting errors in up to two bits)
  - Redundancy bits $= r$ $(r \geq 2)$
  - Information bits $= k = 2^r - r - 1$
  - Total length $= n = k + r = 2^r - 1$
- Examples:
  - $r = 2, n = 3$ and $k = 1$ (redundancy of $2/3 \approx 67\%$)
  - $r = 3, n = 7$ and $k = 4$ (redundancy of $3/7 \approx 43\%$)
  - $r = 4, n = 15$ and $k = 11$ (redundancy of $4/15 \approx 27\%$)
  - $r = 10, n = 1023$ and $k = 1013$ (redundancy of $10/1023 \approx 1\%$)
  - Code efficiency increases with length
- Position of bits:
  - Redundancy bits = multiples of 2 = 1, 2, 4, 8, ...
  - Information bits = other positions

Hamming Codes

- Example:
  - Hamming Code with $n = 15$, $k = 11$ and $r = 4$

position $\rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bits $\rightarrow$	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}
information d_i $\rightarrow$			1		0	0	1		0	0	0	1	1	0	0
parity p_1 $\rightarrow$	X		X		X		X		X		X		X		X
parity p_2 $\rightarrow$		X	X			X	X			X	X			X	X
parity p_3 $\rightarrow$				X	X	X	X					X	X	X	X
parity p_4 $\rightarrow$								X	X	X	X	X	X	X	X
transmitted $\rightarrow$	1	0	1	1	0	0	1	0	0	0	0	1	1	0	0
recieved $\rightarrow$	1	0	1	1	0	0	1	0	0	0	1	1	1	0	0
parity p_1 $\rightarrow$	E		X		X		X		X		X		X		X
parity p_2 $\rightarrow$		E	X			X	X			X	X			X	X
parity p_3 $\rightarrow$				X	X	X	X					X	X	X	X
parity p_4 $\rightarrow$								E	X	X	X	X	X	X	X
corrected $\rightarrow$	1	0	1	1	0	0	1	0	0	0	0	1	1	0	0

Cyclic Redundancy Check (CRC)

- What is better: correct or just detect (and eventually retransmit)?
- Example:
  - Link with error rate of 10^{-6} (1 incorrect bit every $10^6 = 1{,}000{,}000$)
  - Communication in blocks of 1000 bits
  - Single error correction:
    - Using Hamming Code, $r = 10$, $n = 1023$, and $k = 1013$
    - Correction cost of approximately 1 %
  - Single error detection:
    - Using 1 parity bit per block, cost 0,1 %
    - Retransmission of incorrect blocks (1 each 1000 blocks), cost 0,1 %
    - Total cost of 0,2%

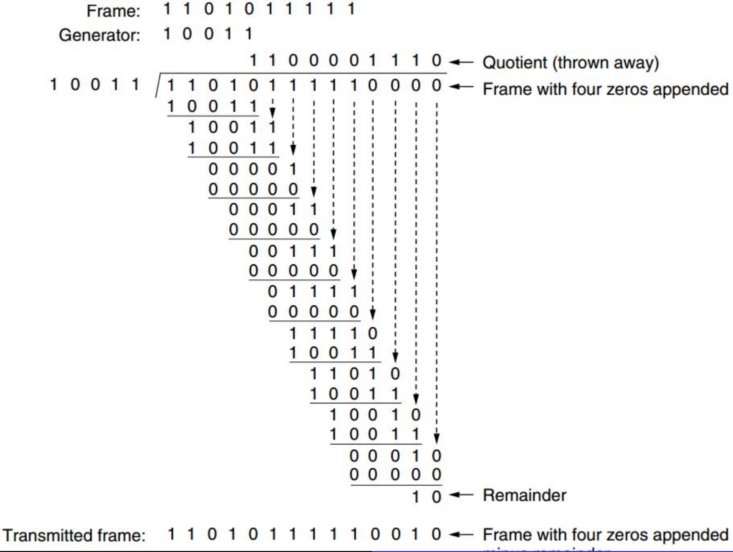
Cyclic Redundancy Check (CRC)

- Cyclic codes used in error detection
- Produces a checksum that is sent with the information
- It is able to detect:
  - Any error polynomial $E(x)$ that is not divisible by $G(x)$
  - Any single error if the generating polynomial $G(x)$ has 2 or more terms
  - Any double error whose distance is less than the order of $G(x)$
  - Any error of an odd number of bits if $G(x)$ is a multiple of $(x + 1)$
  - Any burst of errors with length up to the order of $G(x)$ if $G(x)$ has the term x_0

Cyclic Redundancy Check (CRC)

- Representation
  - Code described by the generating polynomial $G(x)$ of order r
  - Information described by the polynomial $M(x)$
  - Operations are in module 2, no borrowing in subtraction or carrying in addition
- Operation
  - Add r bits 0 to the polynomial $M(x)$, that is, $x^r M(x)$
  - Perform the division $x^r M(x)/G(x)$ and transmit the remainder of the operation together with the information

Cyclic Redundancy Check (CRC)



Cyclic Redundancy Check (CRC)

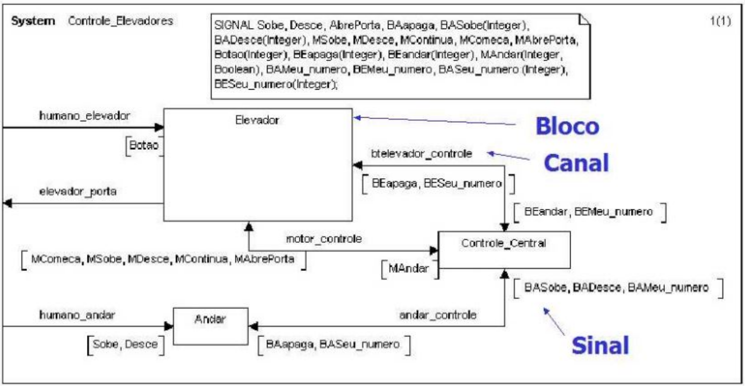
CRC-1	$x + 1$	Paridade
CRC-5	$x^5 + x^2 + 1$	Usado pelo USB
CRC-16	$x^{16} + x^{12} + x^5 + 1$	Definido pelo CCITT
CRC-16	$x^{16} + x^{15} + x^2 + 1$	Definido pela IBM
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$	Usado pelo 802.3
CRC-64	$x^{64} + x^4 + x^3 + x + 1$	Definido pela ISO 3309

Specification and Description Language (SDL)

- ITU Standard Z.100
- Specification language for distributed reactive systems, initially for telecommunications systems, but has found broader broader application today
- It has graphic and textual representations
- It is a formal language, with clear, precise and unambiguous specification
- It can be used in automatic code generation tools
- It can be used in systems simulation tools
- Defines a public (non-proprietary) standard

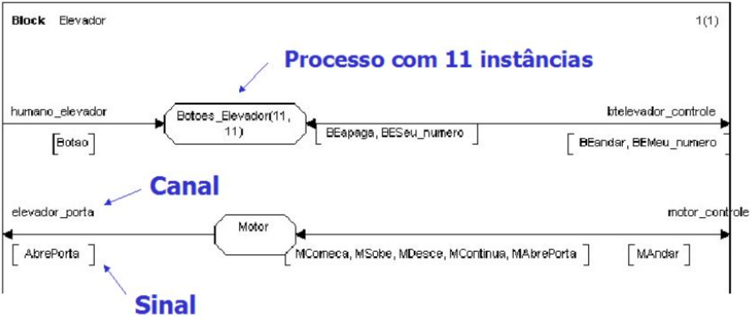
SDL: Sistema

- **System** is composed of **blocks** connected by **channels**
- Channels carry *signals* between blocks and to the outside world



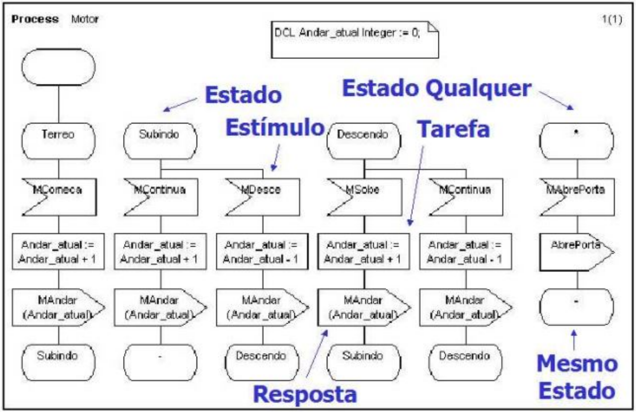
SDL: Block

- **Block** is composed of **processes** connected by **channels**
- Channels carry *signals* between processes and to the outside world



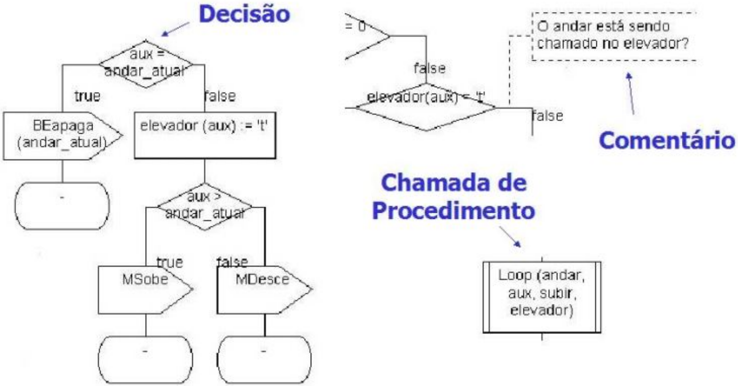
SDL: Process

- **Process** contains the state machine
  - Process consumes and produces signals (consumes **stimuli** and produces **responses**)



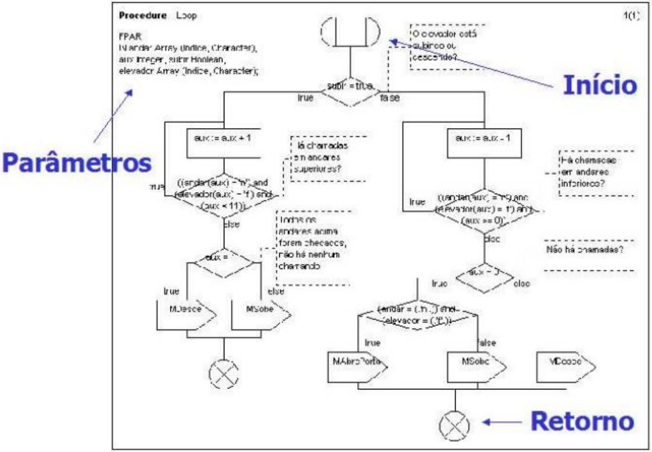
SDL: Process

- **Process** contains tasks, decisions and procedures



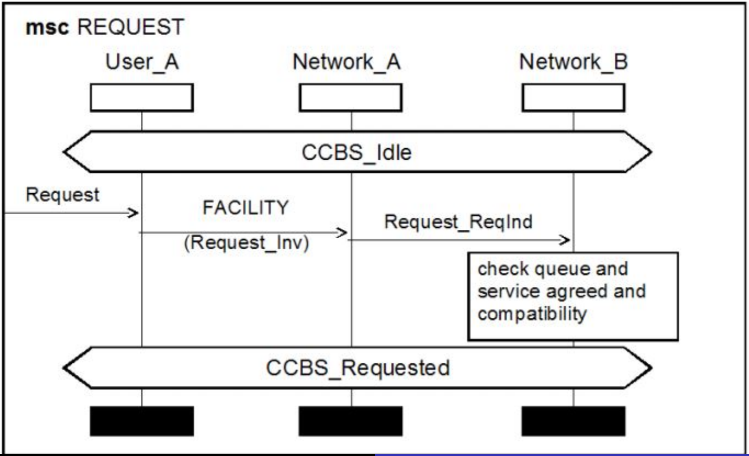
SDL: Procedure

- **Procedure** is equivalent to subroutine

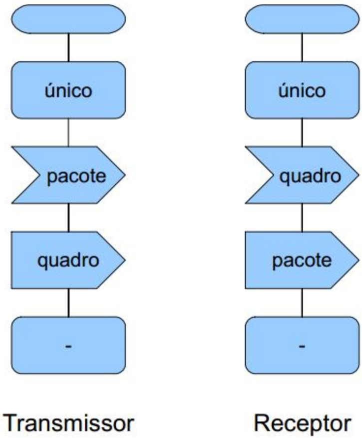


Message Sequence Chart (MSC)

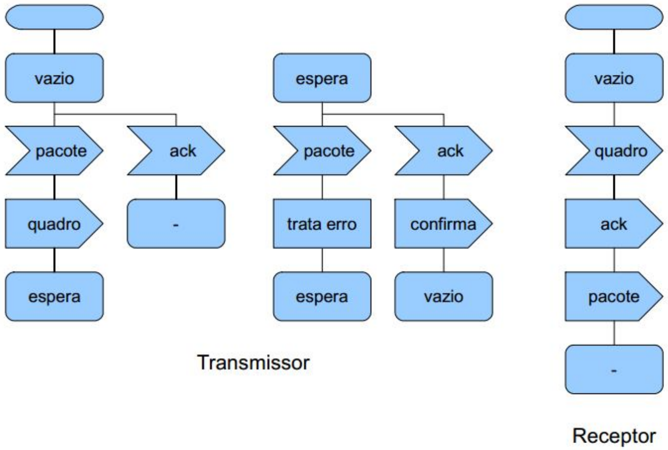
- Used to show the dynamic behavior of the system through message sequences



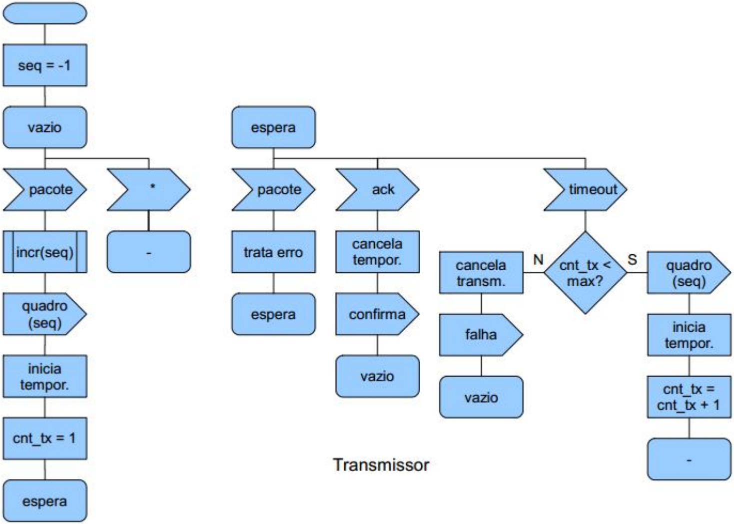
Simplex Protocol, No Errors



Simplex Protocol, Stop-and-Go, No Errors



Simplex Protocol, Stop-and-Go, With Noise



Improvements

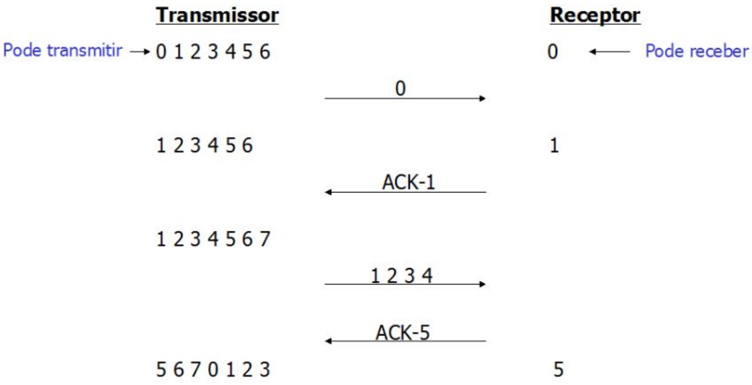
- Bi-directional transmission
- Piggybacking: waits a data packet to embed and send the ACK
  - How long to wait for this data packet?
- Make short ACK packet (without data field)

Sliding Window

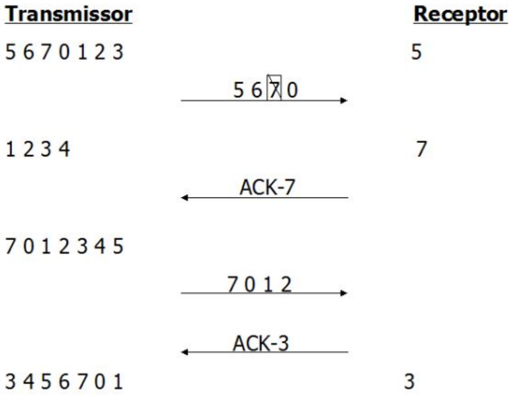
- Window of length n
- Transmitter needs a *buffer* with n positions for pending frames (those not yet ACKed)
- Transmitter sends at most n packets without receiving any ACK
- Send up to packet number (acked + n)
- Packets follow a circular numbering
  - Numbering from 0 to $n - 1$
- Receiving an ACK for package m implies *acknowledgment* of all previous packages

Sliding Window

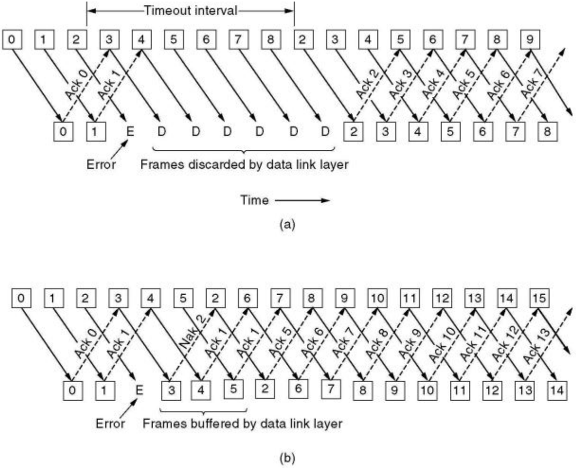
- Example for $n = 7$



Sliding Window



Go-Back-N and Selective Retransmission



(a) *Go-back-n* (b) Selective Retransmission