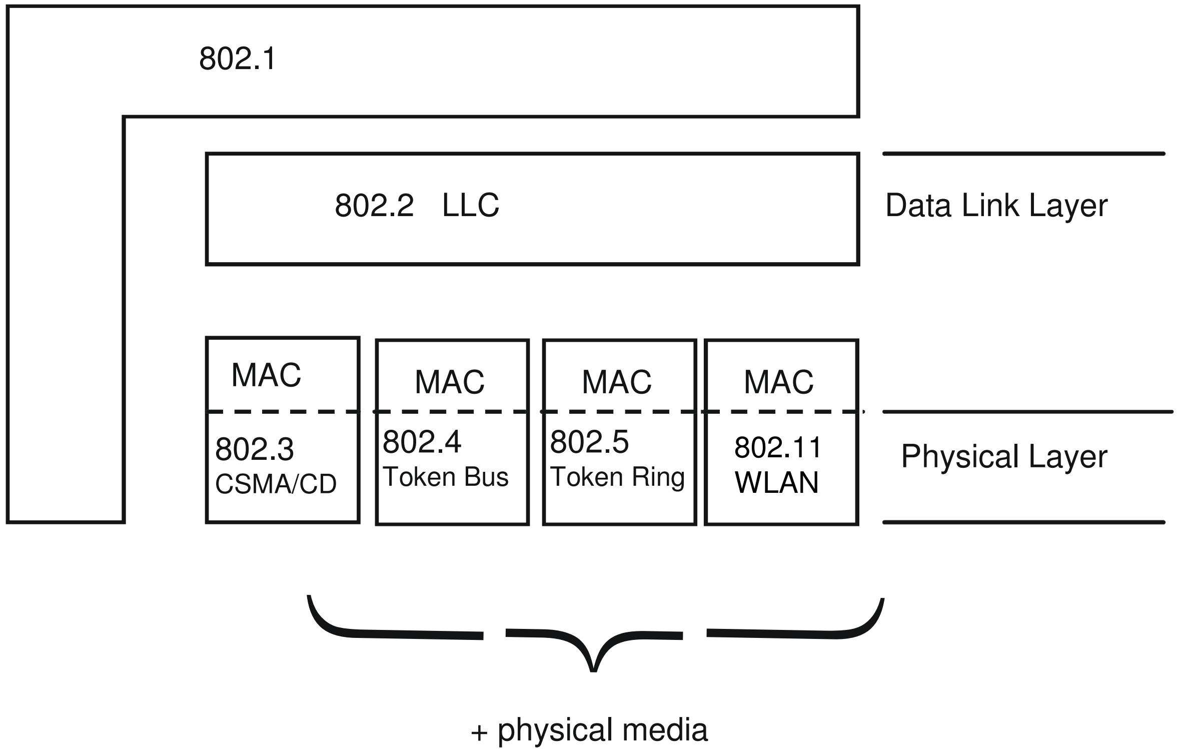
3. Data Link Layer

Functions of the Data Link Layer (Layer 2)

- Detection and correction of transmission errors between nighbors on the same physical link
- Flow control
- In LANs also: Medium Access Control (MAC)

The Data Link Layer in LANs (IEEE 802)

As standardized by IEEE and ISO



3.1 Transmission Errors

- White noise
- Signal distortion
- Cross modulation on cables
- Impulse noise
  - mostly caused by switching equipment
  - typically lasts approx. 10 ms (96 bits destroyed at 9600 bit/s)

Often, errors occur in clusters.

Advantage:	Only a few packets contain errors.
Disadvantage:	Clustered errors are difficult to recognize and correct.

Probability of Transmission Errors

Example of an empirically determined error rate

Probability for a faulty packet for telephone lines and packets of n bytes:

$$p(n) = 10^{-4} n^{0.8}$$

n	**p(n)**
8	5.278031E-04
16	9.189586E-04
32	0.0016
64	2.785761E-03
128	4.850293E-03
256	8.444851E-03
512	1.470334E-02
1024	0.0256

3.2 Error Detecting and Error Correcting Codes

Error: The data received is not identical to the data sent.

For error correction the user data must be supplemented by control information (redundancy).

The amount of redundancy determines the number of errors that can be detected or even corrected by the receiver.

The relationship between the amount and type of redundancy and error detection and error correction potential is the topic of **coding theory**.

A Simple Error Detection Mechanism: The Parity Bit

Bit 1	0	1	1		
Bit 2	1	0	1		
Bit 3	0	0	0		
Bit 4	0	1	0		
Bit 5	0	0	0		
Bit 6	0	1	0		
Bit 7	1	1	1		
Parity bit	0	0	1	even	Parity
	1	1	0	odd	

Properties of the Parity Bit

- The parity bit supplements the number of ones contained in the user data field to an even or odd total number.

```
1   0   0   1   1   1   0   0   0
1   0   1   1   1   1   0   0   1
        Information             ↑
                            Parity bit (for even parity)
```

- Used in asynchronous transmission. And in many other places in computers.
- Only a single bit error (or an odd number of bit errors) can be recognized, and none can be corrected.

Hamming Distance (1)

Hamming distance d between two code words

Number of bit positions in which two code words c_1, c_2 differ.

Example: d(10**001**001, 10**110**001) = 3

Can be computed as the number of bits of c_1 XOR c_2.

Hamming distance D of a complete code C

$$D(C) := \min \{ d(c_1, c_2) \mid c_1, c_2 \in C, c_1 \neq c_2 \}$$

Hamming Distance (2)

Theorem

The ability of a code to detect errors and correct errors is a function of its Hamming distance.

To **detect** e bit errors a Hamming distance of $e + 1$ is necessary.

To **correct** e bit errors a Hamming distance of $2e + 1$ is necessary.

How Much Redundancy Do We Need?

The code may consist of m character bits. How many check bits do we need in order to correct a **1 bit error**?

m	**r?**

$n = m + r$

- There are 2^m legal character codes.
- For every code word there must be n illegal code words around it with a distance of one bit.
- 2^n is the total number of code words we can form.
- $(n + 1)\ 2^m \leq 2^n$ **=>** $(m + r + 1) \leq 2^r$
- This gives the lower limit for r.

Example:

$m = 7$

$(8 + r) \leq 2^r$ **=>** $r \geq 4$

Disadvantage of Error Correcting Codes

Large overhead (much redundancy). Present in all transmissions, including the error-free ones.

Therefore:

Error detection and retransmission are more efficient in most classical computer network situations.

Examples

An Error-Detecting Code

A code with one parity bit (even or odd) per code word.

=> Hamming distance D = 2

=> Recognition of a 1 bit error is possible

(in fact, all errors with an odd number of bits)

An Error-Correcting Code

Four code words:

00000 00000, 00000 11111, 11111 00000, 11111 11111

=> Hamming distance D = 5

=> Correction of all 2 bit errors is possible

Example:	00000 **00**111	=>	00000 11111
	2 bit error		nearest code word

Cyclic Redundancy Check (CRC)

Basic Idea

- Consider the bit pattern as a representation of a polynomial with the coefficients 0 and 1.

  Example: 11001 = $x^4 + x^3 + x^0$

- Select a **generator polynomial** G(x) of degree g.
- Add a checksum to the message M(x) in such a way that the polynomial T(x), formed by the message with attached checksum, is divisible by G(x).
- Transmit T(x).

Algorithm CRC

We have G(x) of degree g and begin by attaching g 0-bits to the message, giving $x^gM(x)$.

1. Divide $x^gM(x)$ by G(x). Do the division modulo 2.
2. Subtract the remainder from $x^gM(x)$, modulo 2
3. The result is T(x), the message including the checksum, ready for transmission.

Example of the CRC Computation

```
Frame:    1101011011,  Generator: 10011. Attach four 0-Bits:
11010110110000 : 10011 = 1100001010
10011
 10011
 10011
  00001
  00000
   00010
   00000
    00101
    00000
     01011
     00000
      10110
      10011
       01010
       00000
        10100
        10011
         01110
         00000
Remainder:1110
```

(division modulo 2: no carry when subtracting)

Generator Polynomials in International Standards

CRC-12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$

Used for 6-bit character codes.

CRC-16 $= x^{16} + x^{15} + x^2 + 1$ (ISO)

CRC-CCITT $= x^{16} + x^{12} + x^5 + 1$ (CCITT)

Both used for 8-bit character codes.

Error recognition potential of CRC-16, CRC-CCITT (16 bits)

(results from coding theory):

- all single and double bit errors
- all errors with an odd number of bits
- all cluster errors with a length ≤ 16
- 99,998% of all longer cluster errors

Implementation of the CRC in Hardware

Implementation of the CRC

The CRC calculation can be implemented in hardware very efficiently (for example, on the HDLC chip) with a shift register and the bitwise XOR function!

3.3 Bit Stuffing

For the use of error detecting and error correcting codes, the data stream must be partitioned into **frames**.

Problem: How can we delimit these frames in the physical layer bit stream if arbitrary bit patterns can occur in the user data?

Solution: **Bit Stuffing**

We choose 01111110 as the frame delimiter („flag“). The sender **always** inserts a 0 after five 1’s in the user data stream, **before** passing the data on to the framing function.

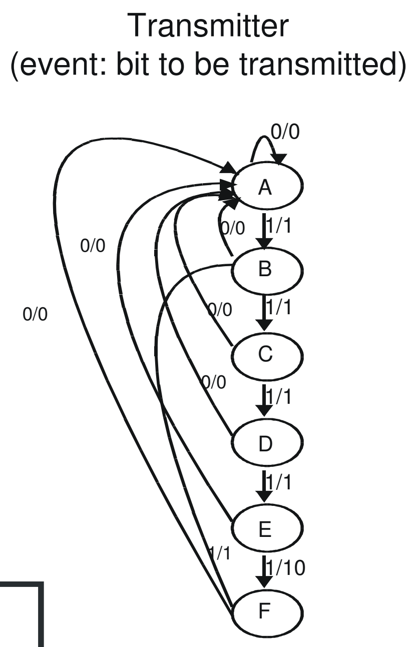
Bit Stuffing Example

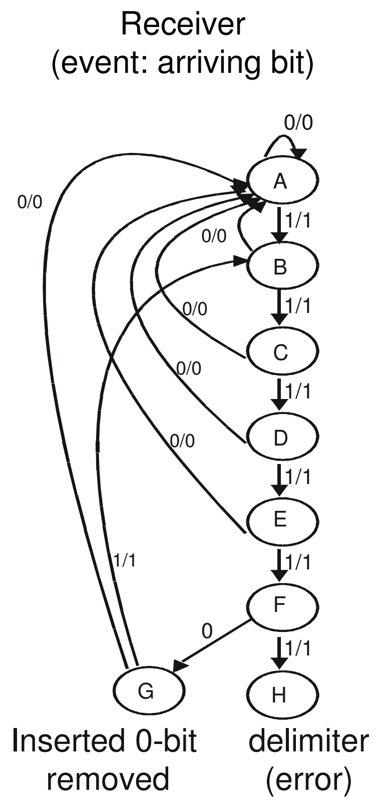
Transmitter	**0**	**1**	**1**	**1**	**1**	**1**		**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**		**0**	**1**	**0**	**1**
Line	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
Receiver	**0**	**1**	**1**	**1**	**1**	**1**		**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**		**0**	**1**	**0**	**1**

If the receiver sees a 0 after five 1's, it **always** removes it from the data stream before passing the data on to the user (higher layer).

Bit Stuffing: Finite Automata

Finite automata for transmitter and receiver for a transmission with bit stuffing.





Notation: e/o
e = event
o = output

Layer 2 Frame Format

8	8	8	>=0	16	8
Flag 01111110	Address	Control	Data	Checksum	Flag 01111110
↑ delimiter					↑ delimiter

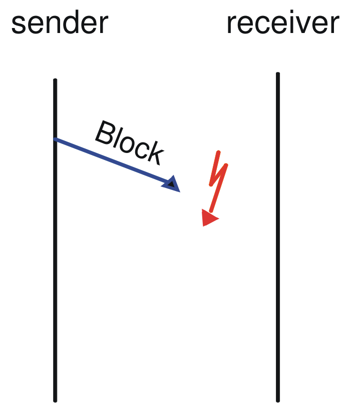
3.4 Acknowledgments and Sequence Numbers

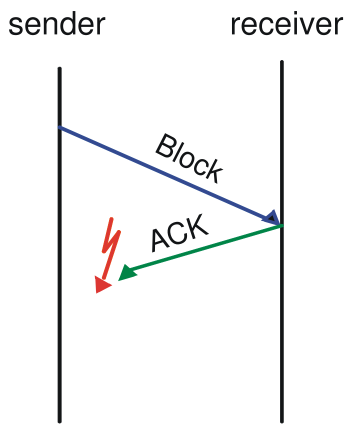
Acknowledgements and sequence numbers are used for

- **error control** (to detect lost blocks, to identify retransmitted blocks)
- **buffer management**
- **flow control**.

Acknowledgments

Two cases of loss:





a) loss of a data block in transit
- receiver waits for data
- sender waits for the acknowledgment

b) loss of an acknowledgment
- sender waits for the acknowledgment
- receiver waits for data

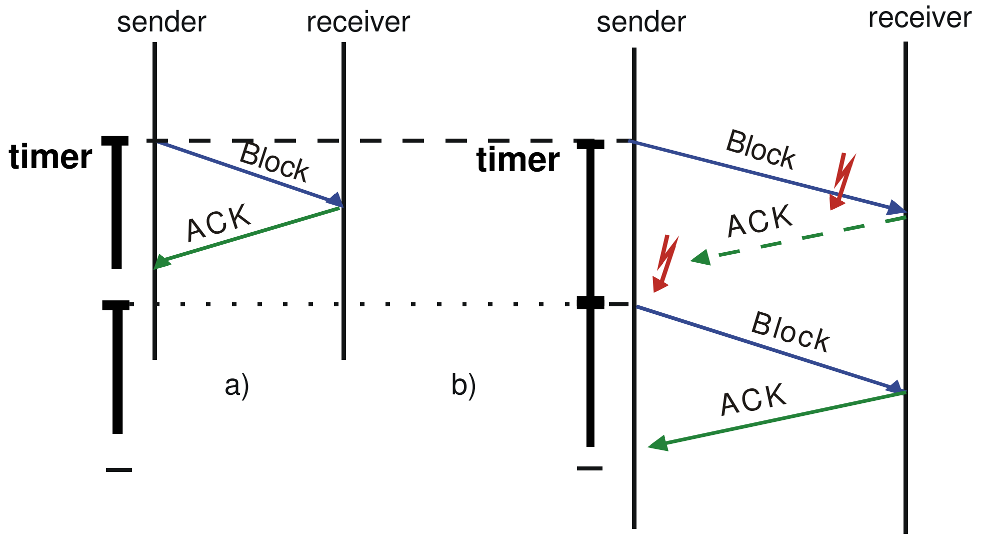
:

Without time out: the sender is blocked.

Solution: We introduce a **time out** at the sender.

Acknowledgments with Time Out

Timeout on the sender side



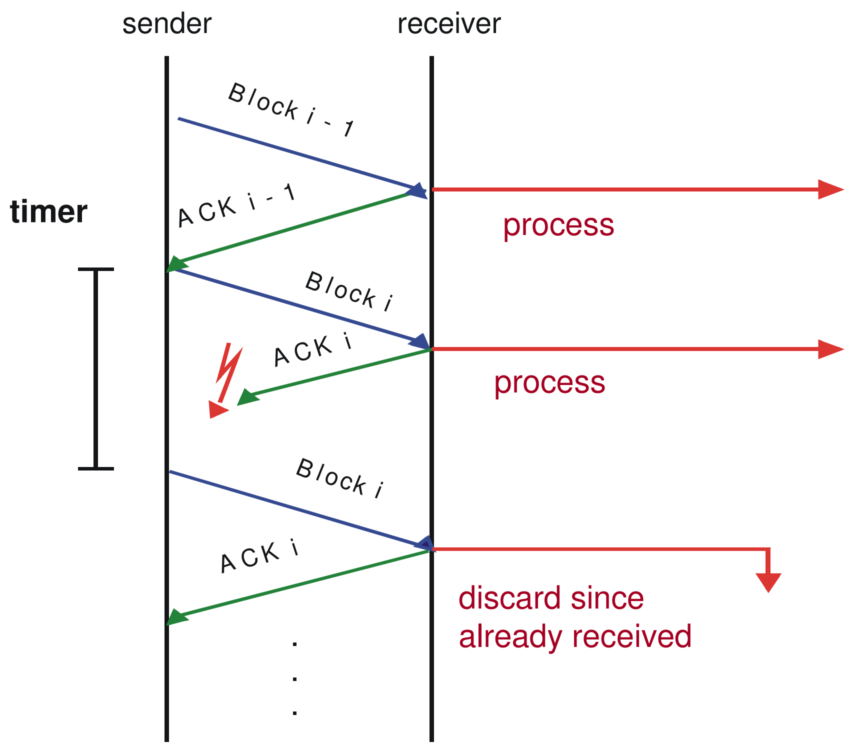
New Problem:

The same block is transmitted twice. The receiver does not know whether he should pass the second block up to the higher layer for further processing.

Acknowledgments with Time Out and Sequence Numbers

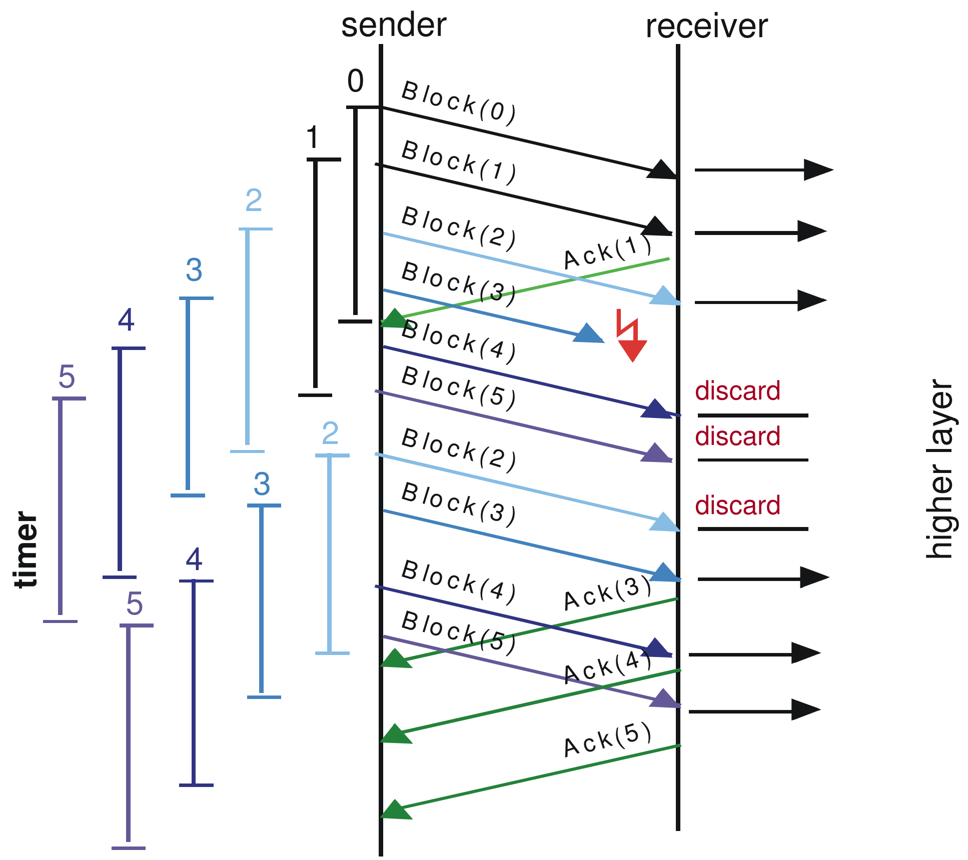
Solution: Sequence numbers

Each block gets a sequence number. When a block is retransmitted it will have the same the sequence number.



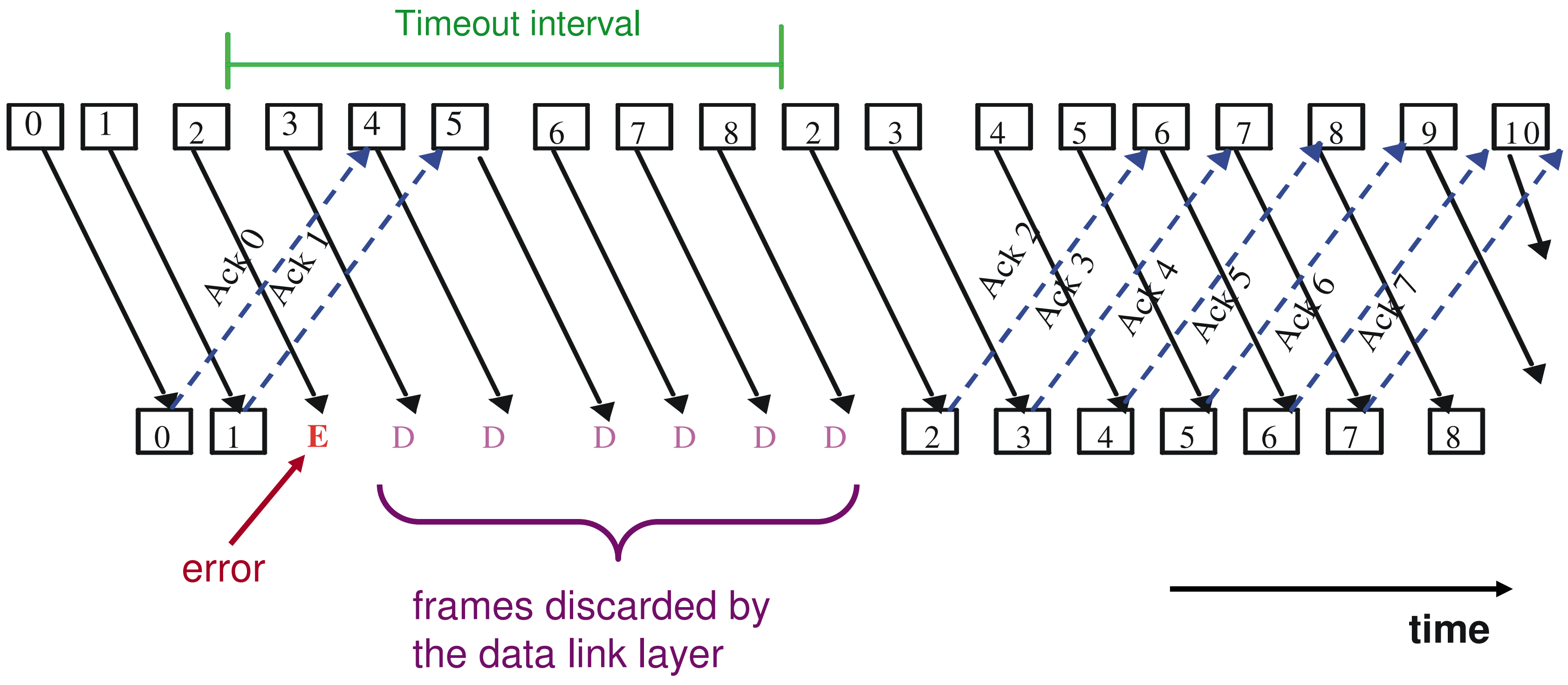
Sequence Numbers

Sequence numbers are also used in the acknowledgments. With one acknowledgment we can now confirm several blocks (frames) at once.



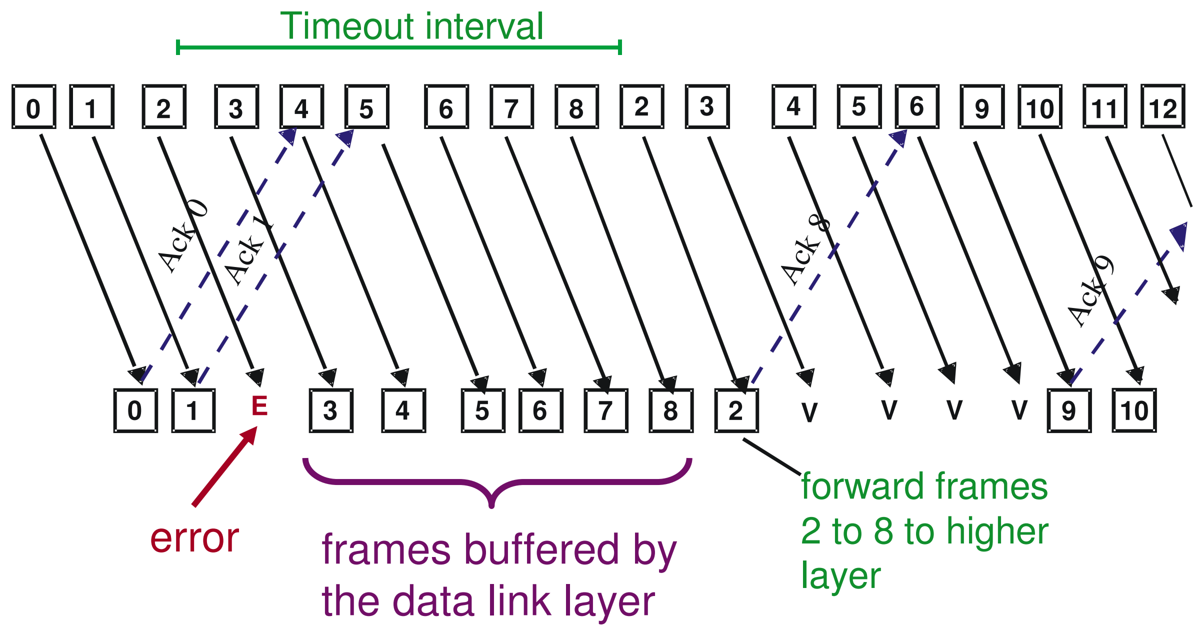
Error Control by "go-back-n" Without Buffering

In case of a bit error or frame loss no Ack is sent back. When the timer expires, the sender retransmits all frames starting with the unconfirmed one.



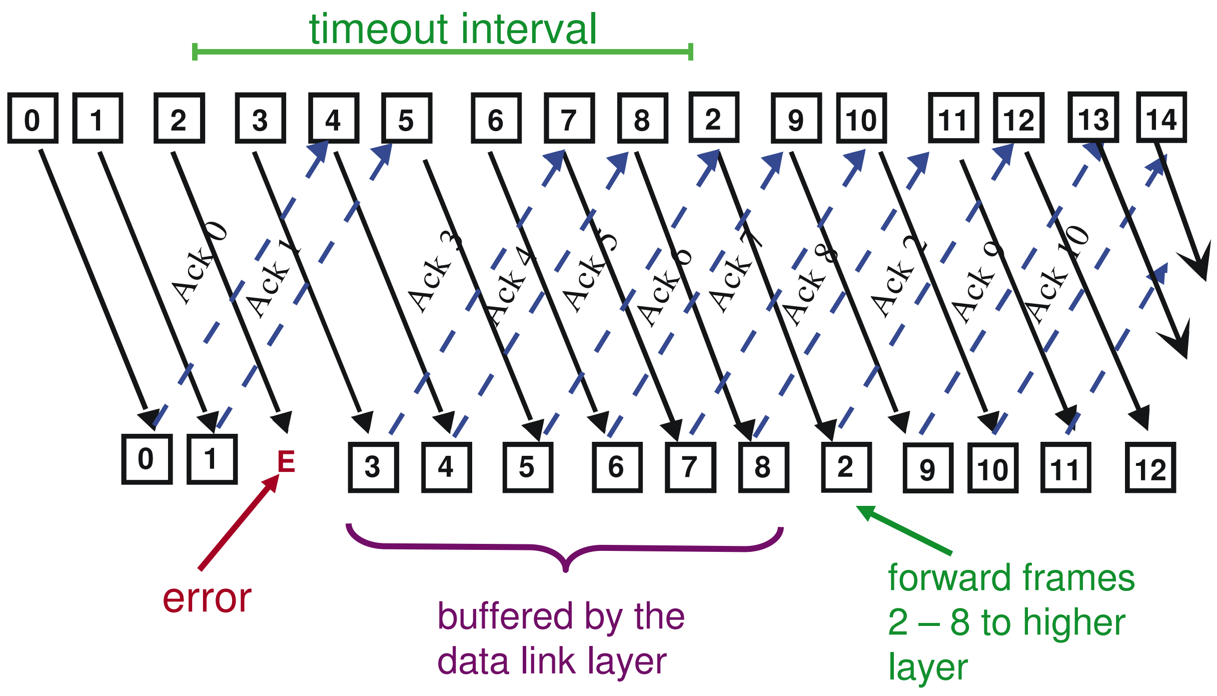
Error Control by "go-back-n" with Buffering

In the case of an error no Ack is sent back. The receiver buffers all frames received correctly. When the timer expires the sender begins to transmit all frames again, starting with the unconfirmed one, until he receives a cumulative Ack from the receiver. It then continues with the first frame that has never been transmitted.


Timeout interval
0 1 2 3 4 5 6 7 8 2 3 4 5 6 9 10 11 12
Ack 0
Ack 1
Ack 8
Ack 9
0 1 E 3 4 5 6 7 8 2 V V V V 9 10
error
frames buffered by the data link layer
forward frames 2 to 8 to higher layer

Error Control by „Selective Repeat"

The receiver confirms each correctly received frame immediately. It buffers all correctly received frames after an incorrect/lost frame. When the timer ex-pires the sender selectively retransmits the unconfirmed frames only.



Passive Error Control

No negative acknowledgements. Only positive acknowledgements (Ack’s) or timeout and retransmission.

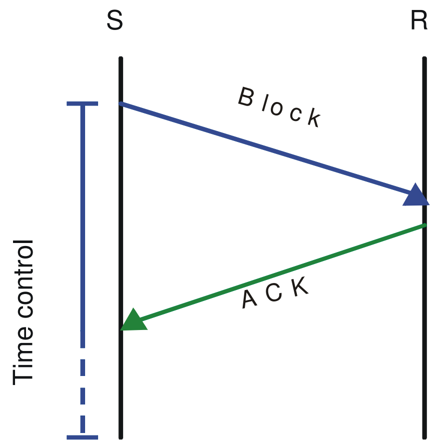
Disadvantage of passive error control:

- no distinction between missing and incorrect frames
- long delay

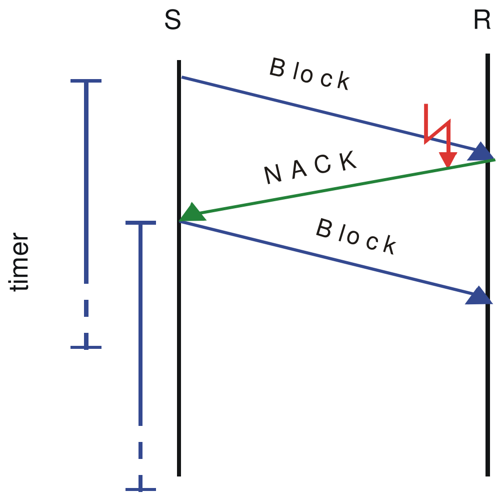
Active Error Control

Negative acknowledgments are also used (Nack's).

no error

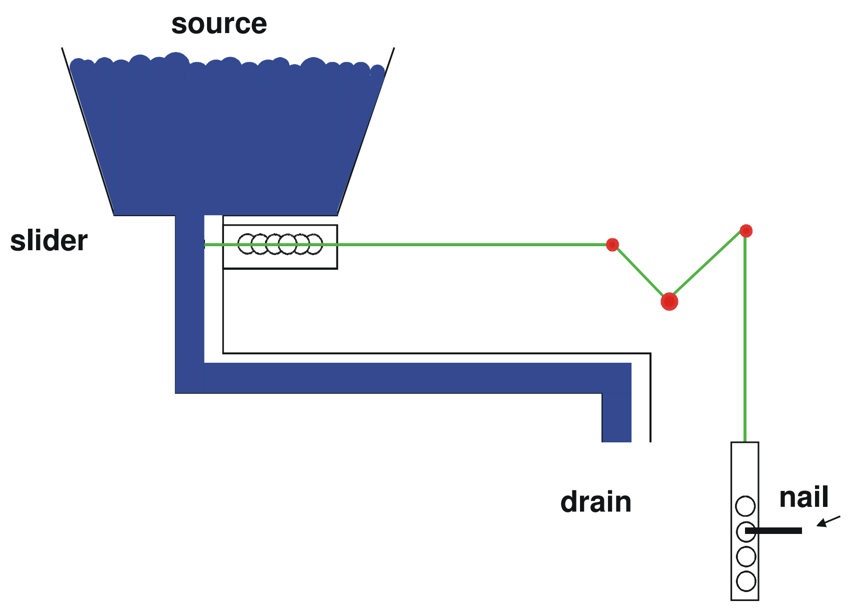


transmission error



3.5 Flow Control

The principle of flow control



Provides feedback from the receiver to the sender in order to avoid data overflow.

A Simple Stop-and-Wait Flow Control Protocol

Assumptions:

- error-free transmission
- limited number of buffers available
- processing speed of sender and receiver can differ

Procedure:

Use Ack’s for a simple flow control. After transmitting a frame, the sender waits for an Ack from the receiver. There is never more than one frame on the way.

Simple Stop-and-Wait Protocol (Sender)

```
procedure sender;
var s:       frame
    buffer: message
    event:  EvType
begin
   while (true) do
      begin
         FromHost(buffer);
         s.info = buffer;
         sendf(s);
         wait(event);       /*  wait for ack  */
      end;
end;
```

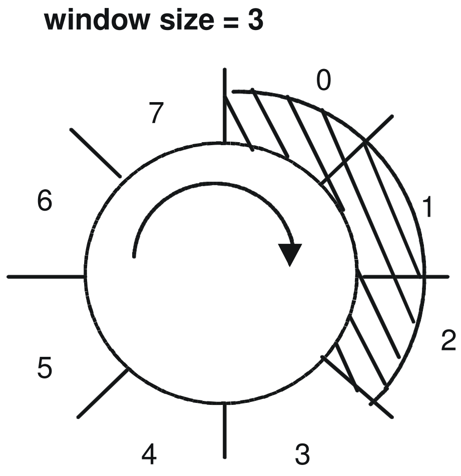
Simple Stop-and-Wait Protocol (Receiver)

```
procedure receiver;
var r,s:    frame;
    Event: EvType;
begin
   while (true) do
      begin
         wait(event);        /*  wait for frame arrival   */
         getfr(r);
         ToHost(r.info);
         sendf(s);           /*  send ack to sender       */
      end;
end;
```

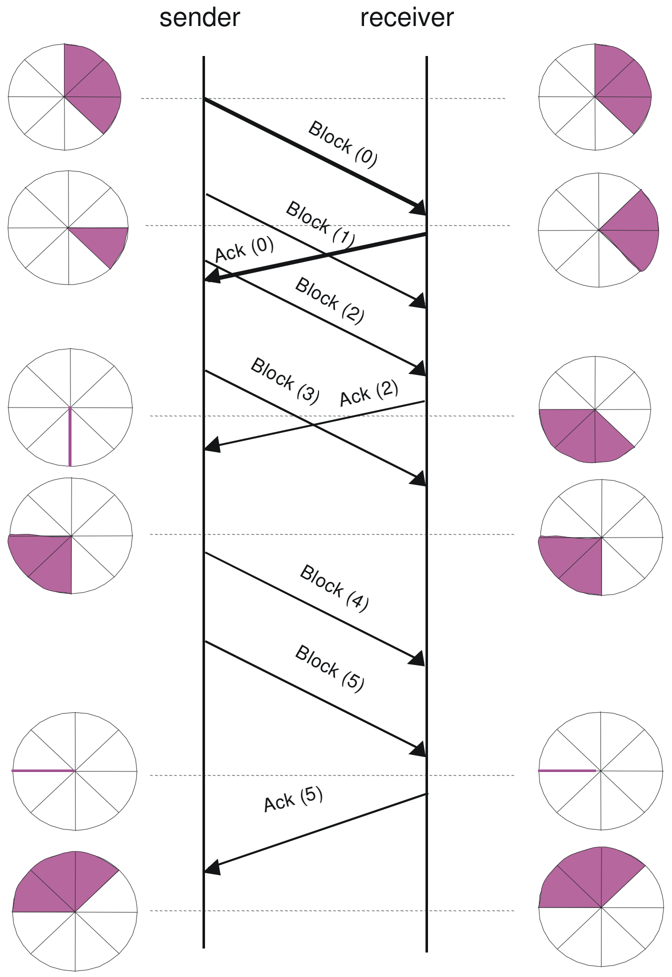
Sliding Window Flow Control

After connection setup the sender has the right to send as many frames as the window size permits. He mustthen wait for an ack. The receiver can send acknowledgements anytime, in particular before reaching the window size. The ack's open the window again.

Example

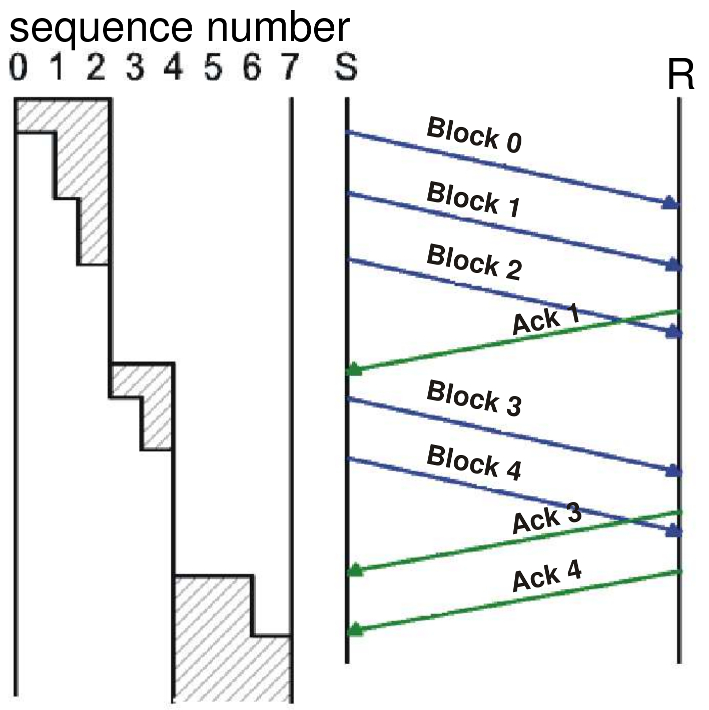


Sliding Window Example (1)



Sliding Window Example (2)

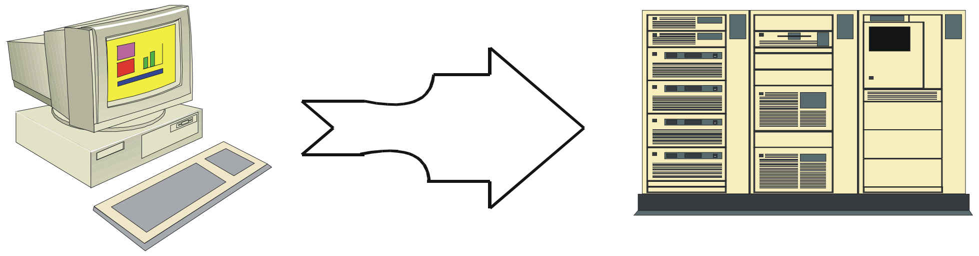
Opening and closing the window



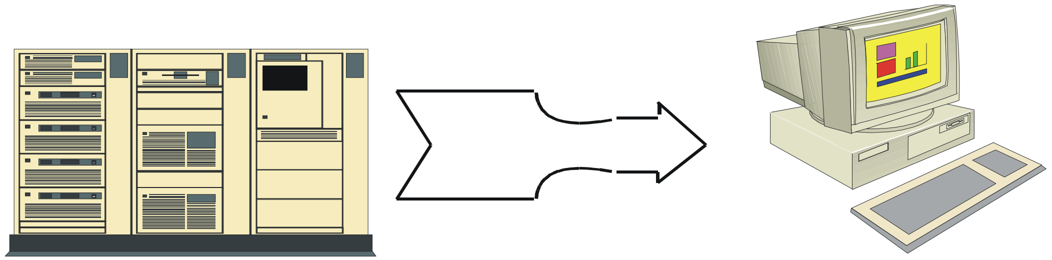
In most communication protocols the same sequence numbers are used for error control **and** for flow control!

Window Size and Buffers Needed (1)

a) Slow sender (PC) – fast receiver (mainframe): 1 buffer is sufficient

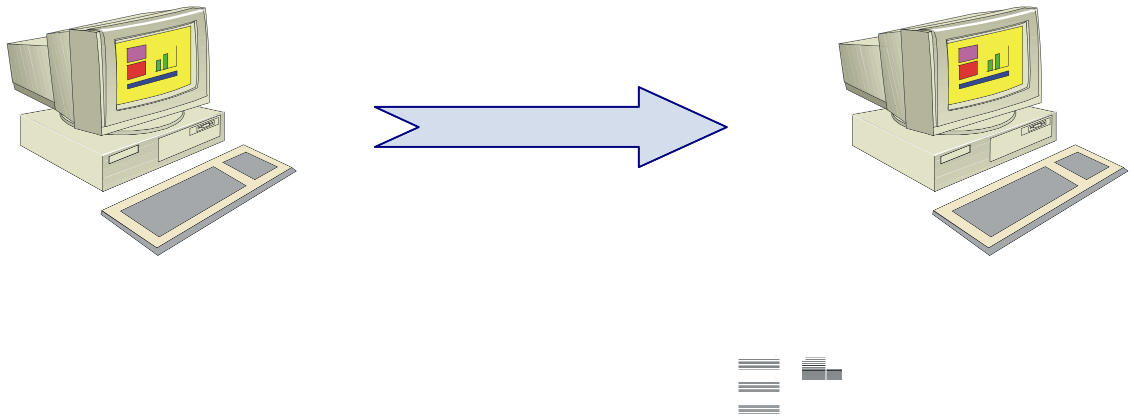


b) Fast sender (mainframe) – slow receiver (personal computer): 1 buffer is sufficient



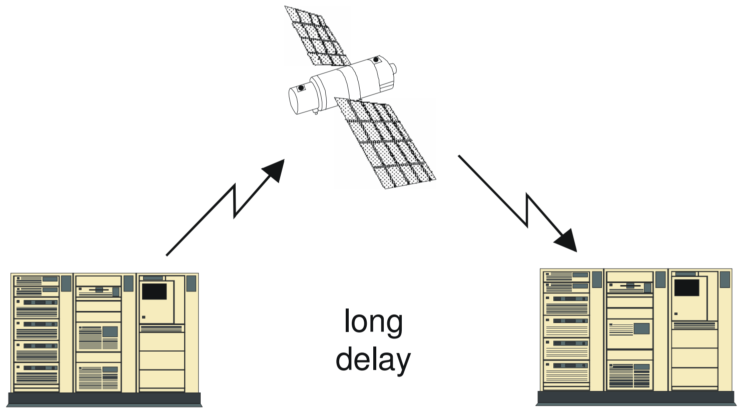
Window Size and Buffers Needed (2)

c) Balanced relationship, for example between two PCs: w buffers are useful to smooth the variance, e.g.: $2 \leq w \leq 7$



Window Size and Buffers Needed (3)

d) Transmission over a communications satellite: w should be large, e.g., $16 \le w \le 127$



Note: The longer the transmission delay, the larger the window size w should be, the more buffers will be needed on the receiver side. The receiver always needs at least w buffers.

3.6 Examples: HDLC, PPP

3.6.1 HDLC (High-Level Data Link Control)

The most important and influential protocol of the data link layer.

Closely related are :

LLC 2	Logical Link Control Type 2 in LANs
LAP-B	Link Access Procedure - Balanced (CCITT; Layer 2 of X.25)

HDLC – High-Level Data Link Control

Basic frame format

	Flag 01111110	Address	Control	Data	Frame Check Sequence	Flag 01111110
bits:	8	8	8	≥0	16	8

Transparency by bit stuffing.

Frame Check Sequence (FCS):

Cyclic Redundancy Check (CRC) of length 16 with

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

Procedure Classes (1)

Asymmetric configuration (unbalanced)

- Primary station: Management of the connection
- One or more secondary station(s)
- Point-to-point connections and multipoint connections
- Normal Response Mode: A secondary station can only send after it is asked to send by the primary station ("polling").
- Asynchronous Response Mode: The primary and the secondary station can send if the line is free.

Symmetric configuration (balanced)

- symmetry between stations, they are peers
- point-to-point connections only (no multipoint connections)
- asynchronous mode only

Procedure Classes (2)

- "Unbalanced" operation in normal response mode
- "Unbalanced" operation in asynchronous response mode
- "Balanced" operation in asynchronous response mode

Asynchronous Balanced Mode (ABM) is the basis of LAP B (Link Access Procedure - Balanced), the layer 2 of the X.25 standard, and also of the PPP (Point-to-Point Protocol) widely used in the Internet.

Address Field and Control Field

Address Field

Originally 8 bits long, for addressing the secondary station when polling

Extended addressing

The size of the address field can be a multiple of 8 bits.

Control field

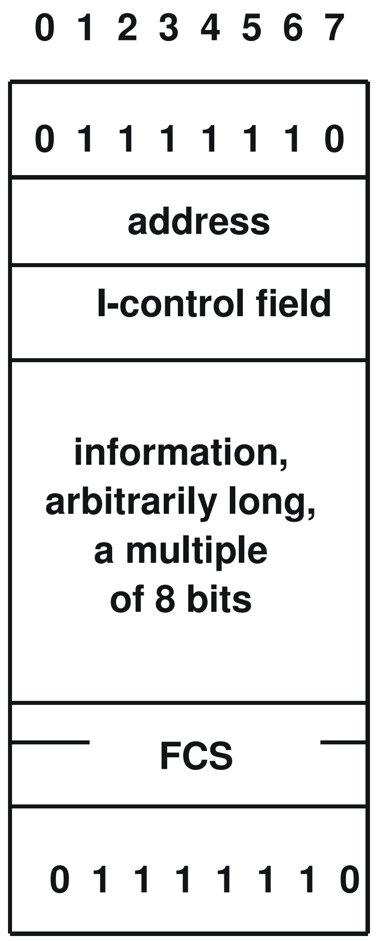
There are three different frame types in HDLC:

- I-frames (information frames)
  used for data communication
- S-frames (supervisory frames)
  used to control the data flow
- U-frames (unnumbered frames)
  used to control the connection.

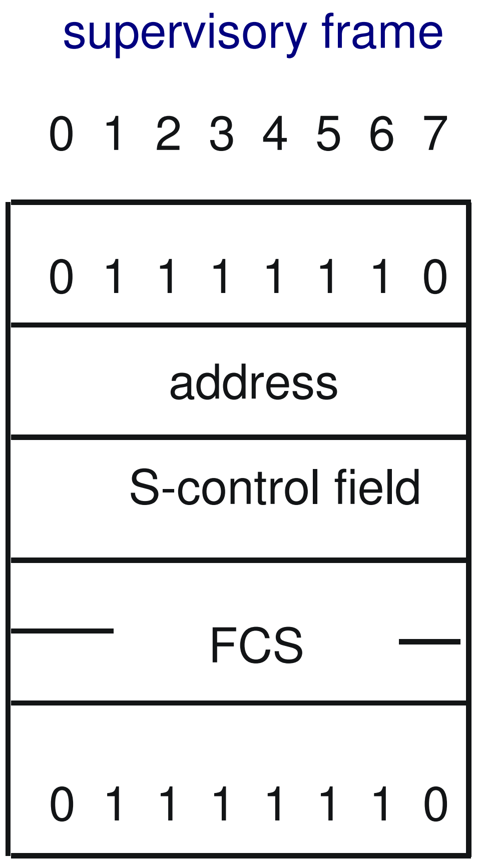
The frame type is indicated in the control field.

HDLC – Frame Formats (1)

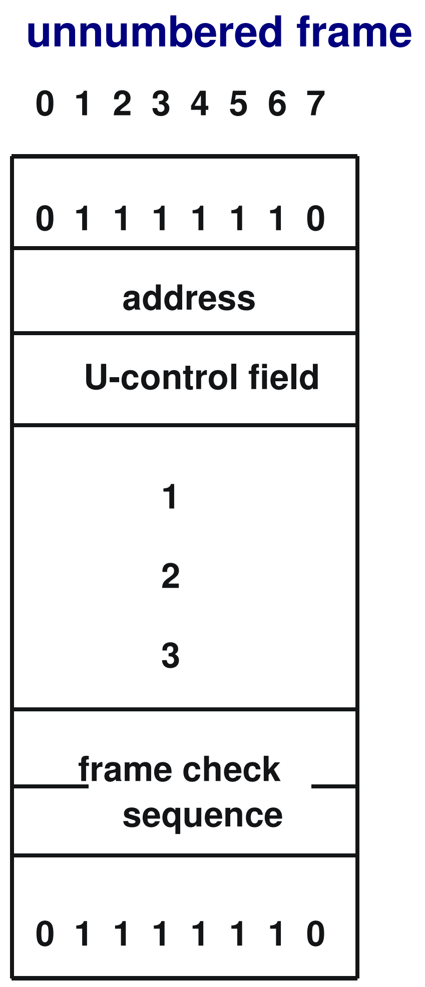
information frame



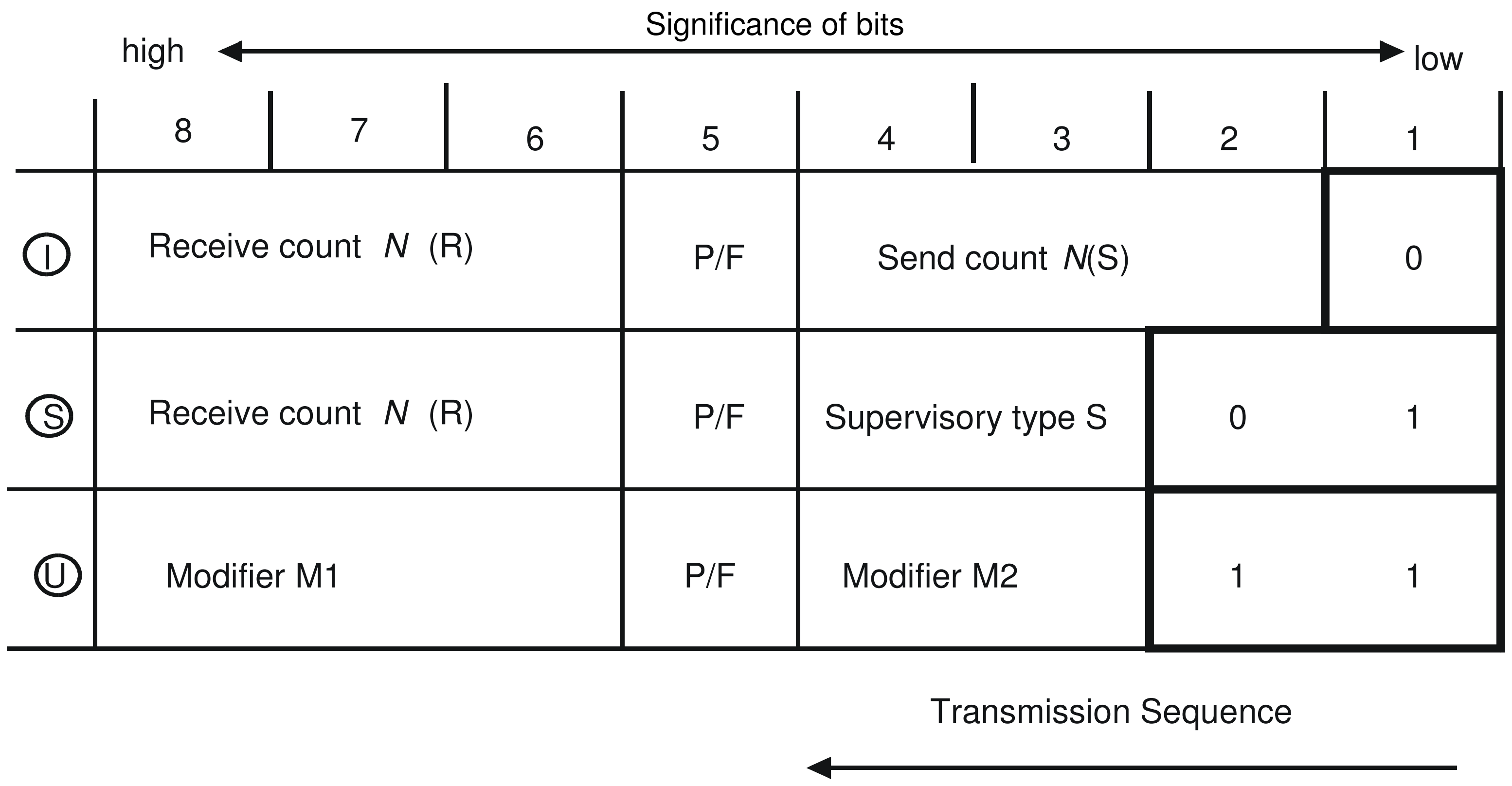
HDLC – Frame Formats (2)



HDLC – Frame Formats (3)



Control Field in the Basic Format



	8	7	6	5	4	3	2	1
Ⓘ	Receive count *N* (R)			P/F	Send count *N*(S)			0
Ⓢ	Receive count *N* (R)			P/F	Supervisory type S		0	1
Ⓤ	Modifier M1			P/F	Modifier M2		1	1

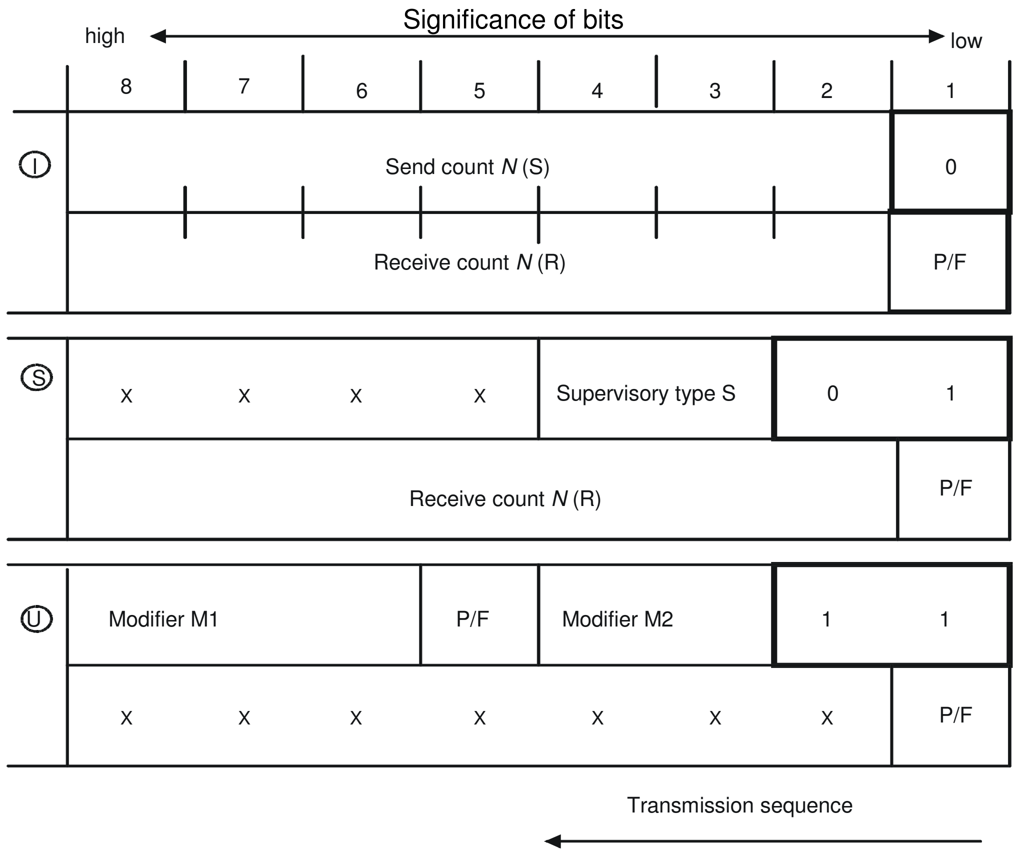
Sliding Window Flow Control in HDLC

HDLC uses a sliding window with a 3-bit sequence number

=> window size = 7

- **N(R)** receive counter
- **N(S)** send counter (sequence number of the frame)
- N(R) and N(S) are counters of the i-frames that were sent and received.
- ACK = number of **the first frame not yet received** (= number of the next expected frame)
- N(R) travels as a “piggybacked acknowledgment“: ACK can be trans-mitted by I-frames on the backward channel.

Extended Control Field (16 Bits)



N(S) and N(R) are now counters with 7 bits each, modulo 128 => window size = 127

Usage: satellite connections, etc.

Command / Response

Command	Frame of any type which is sent by a primary station.
Response	Frame of any type which is sent by a secondary station.

Combined stations can send both commands and responses.

P/F-Bit: Polling bit in the command frame. The receiver must confirm this frame by a Response frame with the final bit set.

Supervisory Frame (1)

We have two bits for the distinction => 4 different types of S-frames:

- **Receive Ready (RR)** 00
  Station is able to receive I-frames. N(R): next expected frame
- **Receive Not Ready (RNR)** 01
  A station indicates to the other one that it cannot receive at this time. The sender stops the transmission.
- **Reject (REJ)** (corresponds to NACK) 10
  Indicates that a transmission error has been detected.
- **Selective Reject** (SREJ) 11
  Indicates to the transmitter that a transmission error has been detected. Requests the retransmission of the selected frame only, specified by N(R).

Unnumbered Frame (1)

Five bits available => 32 different u-frames possible (not all are used)

At connection startup the mode/class of the connection is set as follows:

- Set Normal Response Mode (SNRM)
- Set Normal Response Mode Extended (SNRME)
- Set Asynchronous Response Mode (SARM)
- Set Asynchronous Response Mode Extended (SARME)
- Set Asynchronous Balanced Mode (SABM)
- Set Asynchronous Balanced Mode Extended (SABME)

They are used to set up a data link connection.

N(S), N(R) do not exist in the frame (thus the name “unnumbered“).

Unnumbered Frame (2)

- **Unnumbered acknowledgment (UA)**
  Used for the acknowledgment of a U-frame.
- **Disconnect (DISC)**
  Used by the primary station or by a combined station if it wants to close the connection. Acknowledgment: UA.
- **Disconnected Mode (DM)**
  Used to indicate that a station is logically disconnected. Only commands that set a mode are valid for a logically disconnected station.
- **Frame Reject (FRMR)**
  Sent if an invalid condition is discovered.

3.6.2 PPP (Point-to-Point Protocol)

A point-to-point protocol for accessing the internet over switched point-to-point lines (in particular modem lines) and over dedicated (permanent) point-to-point lines. Standardized in RFC‘s 1661 and 1662.

Characteristics

- Very similar to HDLC
- Framing corresponds to HDLC with certain restrictions
- The control protocol LCP (Link Control Protocol) serves particularly for the negotiation of parameter values.
- PPP can be used in both byte-oriented mode (with byte stuffing) and in bit-oriented mode (with bit stuffing).
- Layer 1 underneath has to be full-duplex.
- Layer 1 can be either synchronous or asynchronous (8 bits, no parity).

PPP Frame Format

8	8	8	16		16/32	8
flag	addr	ctrl	protocol	data	checksum	flag

flag	frame delimiter: `01111110`
addr	always set to `11111111` since in point-to-point traffic no address is needed
ctrl	„control“, always set to `00000011` (corresponds to HDLC-UI with P/F-Bit set to 0)
protocol	Qualifier for the protocol transported in the data field, e.g. IP, IPX
data	user data
checksum	FCS (frame check sequence. Calculated by CRC like in HDLC, either in hardware or software. 16 bits (default) or 32 bits long, the length is negotiated with LCP.
flag	If another frame follows immediately, there is only **one** “flag” (delimiter) between the two frames.

LCP (Link Control Protocol)

LCP is the control protocol of PPP. It serves to establish the connection, as soon as the carrier signal is detected, and to negotiate parameters.

LCP PDUs are carried as layer-3 PDUs in PPP frames! The control protocol thus works very differently from HDLC where there are separate frame types for control functions in layer 2.