5 Data Link Layer and Data Frames

5.1 Tasks of the data link layer

The main task of the data link layer (layer 2 according to the ISO/OSI protocol stack) is to detect and eliminate transmission errors in the bit stream. It structures the bit stream to be transmitted into individual data frames (frames, physical layer service data units) depending on its significance, inserts additional special bit patterns (headers and trailers) for detection, and ensures the correct sequential transmission of these data frames of the sender data as well as the acknowledgement frames constructed on the receiver side.

To ensure error-free data transmission, the following additional tasks may occur:

- Retransmission of data frames when they are destroyed by interfering signals,
- Regulation in case of "data flooding", i.e. overflow of the buffer area of the receiver,
- minimization of data flow between the physical layer and the network layer,
- "piggyback" control in bidirectional transmission lines,
- control of access to the common data channel in broadcast networks (introduction of the MAC sublayer).

In local networks, the data link layer is subdivided into two sublayers:

- **Medium Access Control Layer (Layer 2a, MAC layer):**
  - Lower sublayer of the data link layer,
  - structuring of data frames depending on the network technology (Ethernet, Token Bus, Token Ring, FDDI),
  - Preparation of a serial data transmission,
  - Execution of a collision test,
  - Receiving the data on the receiver side,
  - checking the received data frames for correctness,
  - decomposition of the data frames into its components.

- **Logical Link Control Layer (Layer 2b, LLC layer):**
  - Upper layer of the data link layer,
  - Provision of a universal data format for the adaptation of the different types of frames due to different network technologies (Ethernet, Token Bus, Token Ring, FDDI),
  - error monitoring,
  - flow control.

Fig. 5.1-1 shows the layer structure around layer 2 in the TCP/IP model, ISO/OSI model and Ethernet standard.

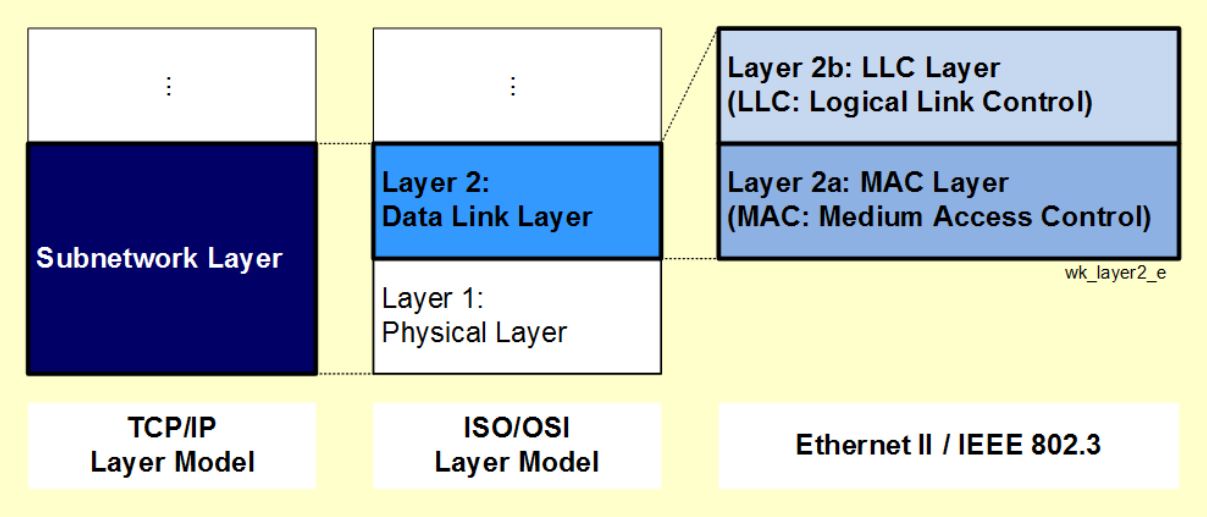


Fig. 5.1-1: Subdivision of the data link layer into MAC and LLC layers

5.2 Data frames

The structure of the data frames is to be explained by way of example using the Ethernet standard, version 2 (Ethernet II), which has established itself as the basis of data communication and is currently the most widely used method, and the international ISO standard IEEE 802.3 published in 1990.

For the transmission of data in digital communication networks, the bits of user data and control data to be sent over the transmission medium are packed by layer 2a (ISO/OSI notation) of the data link layer (Media Access Control layer MAC) in the form of data frames (frames, data frames). The data frames are constructed by software and then passed on by the network controller.

Both standards are compatible at the physical layer and layer 2a (MAC layer) of the data link layer. They therefore do not influence each other and can be used simultaneously in the same network (caution: division into two different subnets, communication only possible via gateways!).

The graph on the following page show the structure of the data frames according to the Ethernet II protocol and the IEEE 802.3 protocol (Fig. 5.2-1).

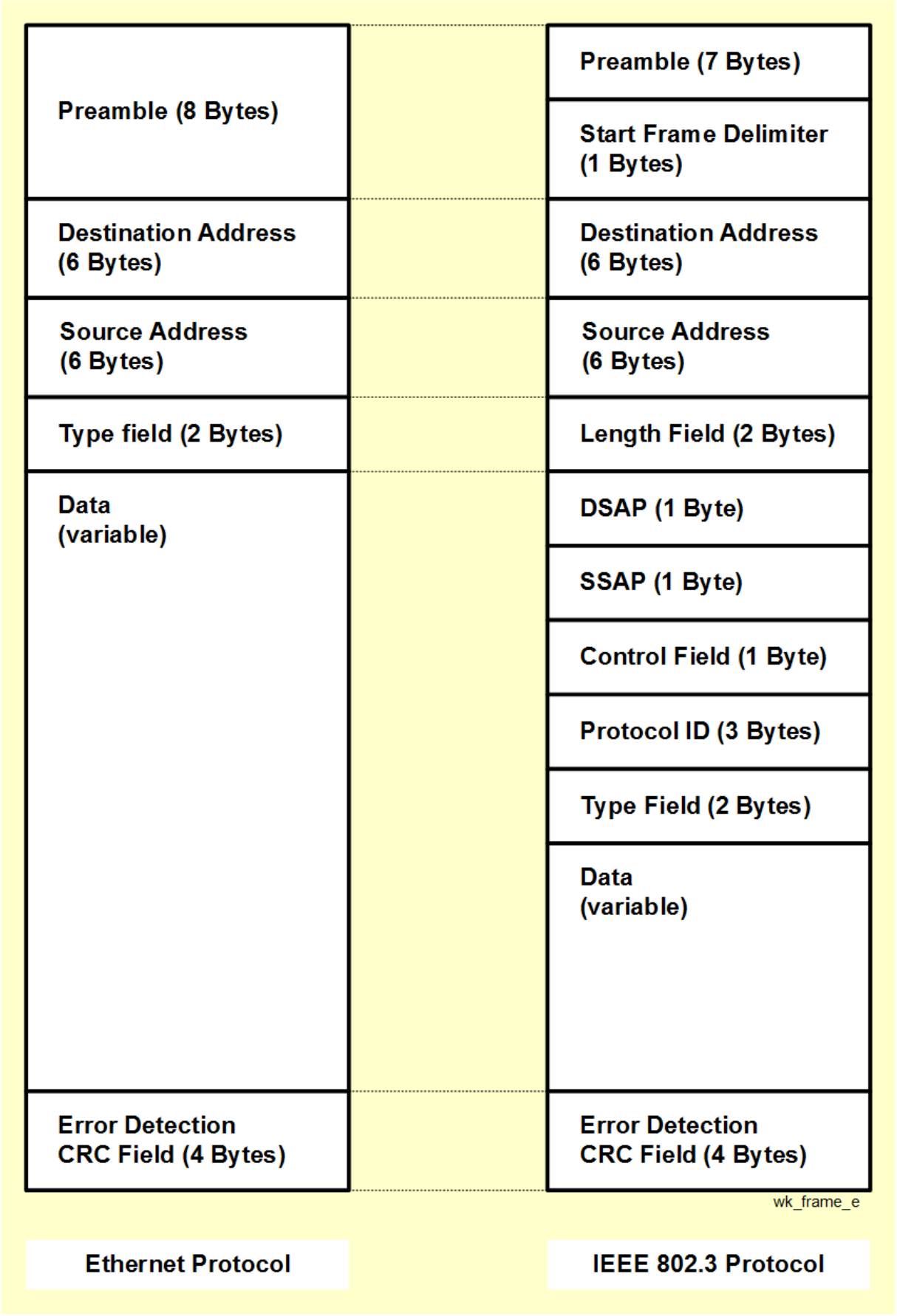


Fig. 5.2-1: Data frames according to Ethernet II and IEEE 802.3

The preamble is used for the time synchronization of transmitter and receiver in the case of asynchronous transmission methods (e.g. CSMA/CD methods). It is composed of eight (Ethernet) or seven (IEEE 802.3) 10101010 bit combinations. The timing of the receiver is defined from the bit sequence of the preamble.

In the IEEE 802.3 standard, the preamble is followed by a signaling byte (start frame delimiter) as a frame delimiter. The unique bit sequence (10101011) signals the start of the actual data packet to the receiver.

According to the specifications of the International Standardization Organization (ISO), each hardware unit (PC, workstation, file server, ...) in a communication network must be identifiable with a globally unique address. Addresses are assigned by the Institute of Electrical and Electronic Engineers IEEE in the USA.

Currently, six-byte wide hardware addresses (receiver and destination addresses) are used. The previously used two-byte addresses are still valid and are also accepted by the new standard.

Structure of the 48-bit (6-byte) Ethernet hardware address:

Bit 47 ***I/G-Bit (Indivdual/Group)*:**
- Distinction between individual address and group address (broadcast address, multicast address),

Bit 46 ***U/L-Bit (Universal/Local)*:**
- Distinction between locally assigned address and universal address (Local addresses are only allowed in isolated networks.),

Bit 45 … 0 **Actual hardware address:**
- Structure: Manufacturer identification (length: 3 bytes) + manufacturer-specific identifier (e.g. device, serial number, ...),
- storage (burn-in) in a ROM chip of the controller of the network interface card (NIC) by the manufacturer.

Examples of manufacturer identifiers:

00-00-0C	*Cisco*
00-00-C6	*HP*
00-AA-00	*Intel*
08- 00-20	*Sun*
08-00-5A	*IBM*
00-02-2D	*Lucent*

The two-byte wide type field of the Ethernet protocol uniquely identifies the protocol used in the data packets.

Examples for type field values (hexadecimal):

Type field value	Protocol
0800	*Internet Protocol IP*
0805	*X.25 Level 3* (öffentliches Paketnetz *Dx-P*)
6001	*DECnet*
809B	*AppleTalk*
8137, 8138	*Novell*

The length field of the IEEE 802.3 standard specifies the length (in units of bytes) of the subsequent data (protocol header and user data) in the data packet.

The data field contains all header and data information of the higher layer protocols (layers 2b to 7). If the length falls below the minimum length (Ethernet II: 46 bytes = 368 bits; IEEE 802.3: 38 bytes = 304 bits), the missing bytes are filled with padding bits of value zero.

The DSAP (Destination Service Access Point), SSAP (Source Service Access Point), the control field and the protocol ID field contain information of the LLC layer in the IEEE 802.3 standard.

A four-byte field for error detection (Cyclic Redundancy Check field CRC, Frame Check Sequence Field FCS) completes the frames. The sender enters a checksum in the CRC field that can be used by the receiver to detect transmission errors that occur in layer 1 or on the transmission link.

The 32-bit wide value G(x) of the CRC field is calculated with the following generator polynomial (CRC-32, IEEE 802):

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

The CRC calculation covers the sections destination address, source address, type/length field and data field.

If one of the cases listed below occurs during data communication, the received data frame is discarded:

- Error in CRC field,
- length of the data packet is not equal to an integer multiple of 8 bits,
- specification in the length field does not match the packet length
- (IEEE 802.3 only).

5.3 Logical Link Control

The organizational data for communication transmission between sender and receiver are inserted in the case of the ISO standard IEEE 802.3 on layer 2b (Logical Link Control Layer LLC) within the link layer.

According to the ISO/OSI convention, protocol data units (PDUs) are used to uniquely represent the individual fields. Each protocol data unit is made up of several elements:

- Identification (Frame Control),
- Destination Address,
- Source Address,
- data field with the data of the higher layers
- (LLC information, Logical Link Control Information),
- Control field for error control (Frame Check Sequence FCS).

The OSI model defines vertical access between the layers via addresses, the LLC Service Access Points (LSAPs). Each PDU contains a one-byte wide destination address (Destination Service Access Point DSAP) and a one-byte wide source address (Source Service Access Point SSAP).

The least significant bit of the DSAP address is the I/G bit (0: individual address, 1: group address). The least significant bit of the SSAP address is the C/R bit (0: command, 1: response).

During data communication, the least significant address bits of the LLC layer are always transmitted first.

Four different LLC services exist:

LLC type 1: connectionless service (datagram service) without error control and error control and without flow control (mostly used in LAN),

LLC type-2: connection-oriented service, virtual point-to-point connection with error correction,

LLC type-3: simple connection-oriented communication service with acknowledgement,

LCC type-4: fast point-to-point connection with full duplex communication.

5.4 Collision mechanisms

5.4.1 General

If data frames from several senders are sent simultaneously in communication networks, data collisions, i.e. overlapping of data frames, can occur, which destroy the payload data to be transmitted.
Depending on the network technology, various mechanisms exist for detecting collisions and repairing the incidents. These methods are implemented in the MAC layer (layer 2a).

In bus systems, e.g. in an Ethernet network, all stations are connected to a common transmission medium. A data frame generated by one station propagates bidirectionally over the bus. If a second station sends another data frame before the first frame has reached its destination station, a collision occurs.

The transmission system can be in one of three different phases: **Transmission Phase, Competitive Phase, Free Phase (Inertial Phase)**:

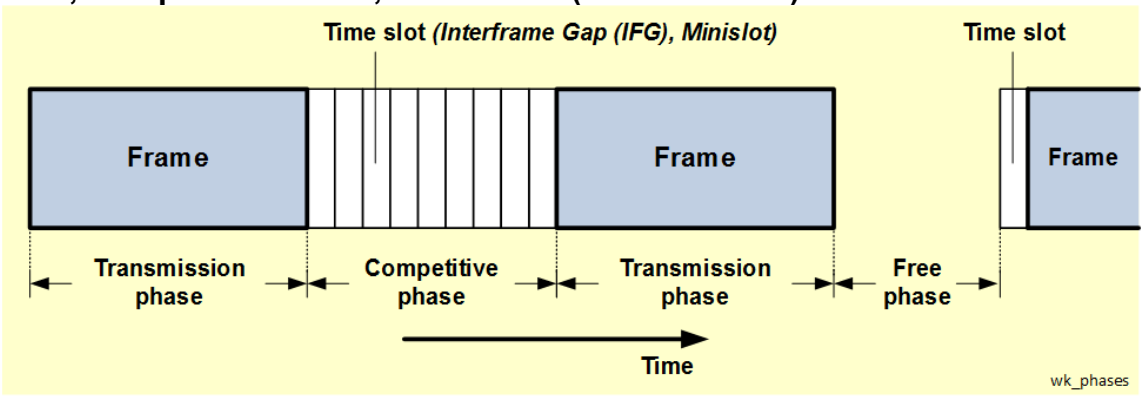


Fig. 5.4.1-1: Phases in the transmission of data frames

Various techniques are used to avoid collisions:

- **Selection techniques:**
  - Allocation of a transmission permission to a station by a central or decentralized control unit,
  - Time for the next transmission phase cannot be determined by the station itself determined by the sender,

- **Random access techniques:**
  - All stations have continuous transmission access to the transmission medium,
  - Occurrence of conflicts when two stations transmit simultaneously,
  - Implementation of methods for conflict detection and conflict conflict elimination.

As an example, two related random access methods used in computer networks will be discussed:

Carrier Sense Multiple Access CSMA,
Carrier Sense Multiple Access with Collision Detection CSMA/CD.

5.4.2 Carrier Sense Multiple Access CSMA

All stations interested in transmitting listen to the transmission medium **(carrier monitoring, carrier sensing)**. If there are no data frames in the transmission channel, any station can transmit. Three different strategies can be used for a transmission that is currently active:

- **non-persistent CSMA:**
  - Deferring the transmission request,
  - Retry of transmission after a randomly chosen delay time (in multiples of the slot time)
  - Delay time (in multiples of the slot time),
  - Behavior as in case of conflict,

- **persistent CSMA:**
  - Waiting for the end of the current transmission,
  - Immediate sending after the end of the current transmission (transmission with probability p = 1 in case of free carrier),
  - unavoidable conflict, if simultaneously several stations are waiting for a transmitting possibility,

- **p-persistent CSMA:**
  - Behavior according to the 1-persistent strategy with probability p ($0 \leq p \leq 1$) for transmitting,
  - Waiting for a short period of time (in multiples of the slot time) with the probability (1-p) and then repeated listening to the transmission transmission channel,
  - If channel then occupied, behavior as in case of conflict, otherwise transmission.

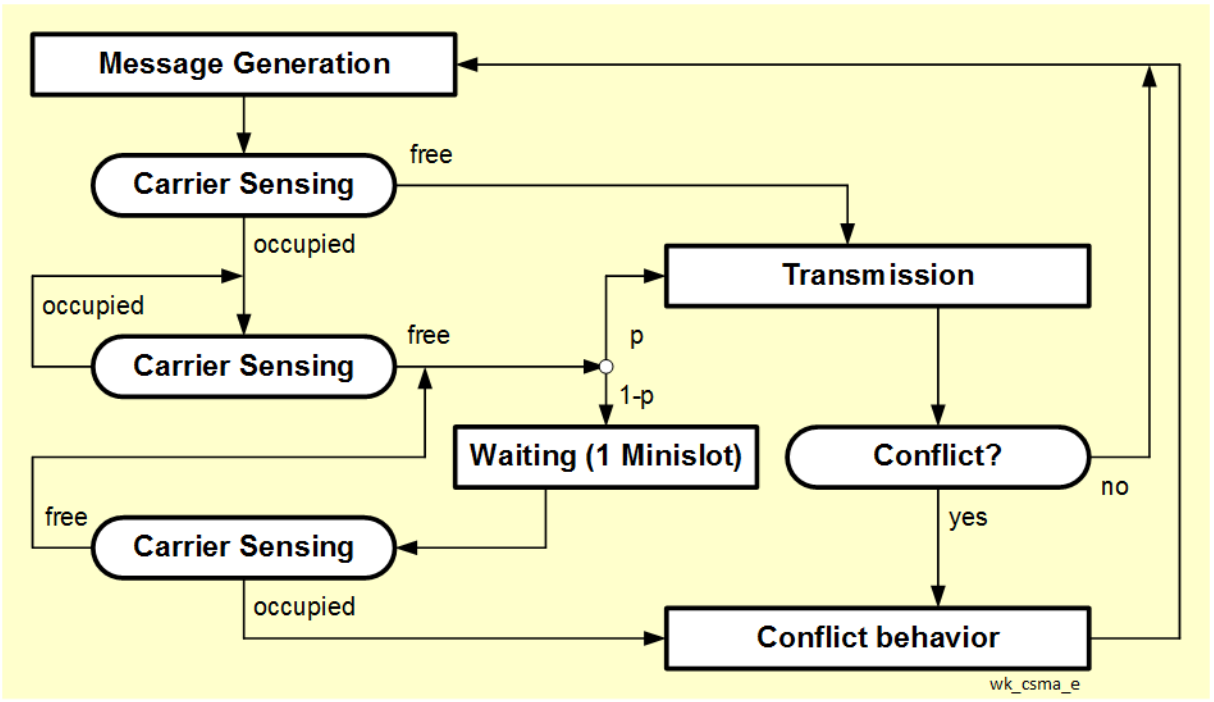


Fig. 5.4.2-1: Basic algorithm of the CSMA method

5.4.3 CSMA with collision detection CSMA/CD

Collision detection CD is introduced to improve the transmission rate. For this purpose, each station also listens to the medium during its own transmission (technical implementation by separating the transmitting and receiving units).
Initially, the CSMA/CD method behaves according to the 1-persistent strategy. When a collision occurs, all stations that notice the collision first send out an **interference signal** (**jamming signal**, 4 bytes with 16 1-0 bit combinations). All transmitting stations abort their transmissions immediately. After that, transmission attempts can be started again. The initial waiting interval (slot time) is 512 bit times (= 51.2 μs for Ethernet II). If collisions are detected repeatedly, the maximum waiting interval until retransmission is doubled in each case **(Binary Exponential Backoff BEB)**. The maximum number of attempts **(Attempt Limit)** is limited to 16. After the tenth attempt **(Backoff Limit)**, the waiting time remains constant.

5.5 Throughput and efficiency

Two basic variables are discussed to describe the performance of computer networks:

- bandwidth,
- latency.

5.5.1 Bandwidth

The bandwidth specifies the number of bits that can be transmitted within a defined period of time.

5.5.2 Latency

Latency T_L describes the amount of time it takes for a message to be transported from one end of the network to the other.

Round trip time (RTT) is the time required to send a message from one end of the network to the other end and back again:

$$RTT = 2T_L .$$

The latency T_L is composed additively of the propagation delay t_{Signal}, the transmission delay of the data packets t_{Packet} and the waiting time t_{Wait}:

$$T_L = T_{Signal} + t_{Packet} + t_{Wait} =$$
$$= \frac{L}{c} + \frac{P}{B} + t_{Wait}$$

with L: length dimension of the network,
c: phase velocity of the propagating wave ,
P: packet size,
B: bandwidth.

The waiting time describes the sum of all time subintervals for packing the necessary data frames in the higher layers, e.g. encryption or compression of the data, and the delays due to collision prevention.

5.5.3 Delay-Bandwidth Product

The delay-bandwidth product of latency T_L und bandwidth B

$$T_L B$$

is a measure of the data volume of a transmission line. It specifies the number of bits that a sender must transmit until the first bit arrives at the receiver.

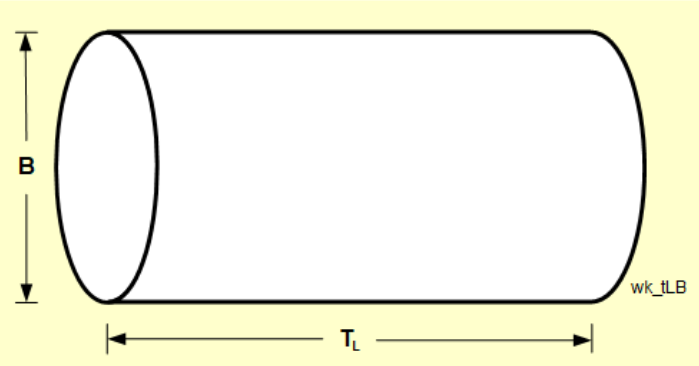


Fig. 5.5.3-1: Schematic view of the delay-bandwidth product

5.5.4 Information cube

In order to be able to transmit a given quantity of messages per time unit t, the mutual compensation of signal-to-noise ratio S/N and bandwidth B is possible. Error-free transmission becomes feasible even with a low signal-to-noise ratio if the bandwidth is selected to be sufficiently large:

$$B \cdot \frac{S}{N} \cdot t = \text{const}.$$

This situation can be visualized graphically using the example of the "information cube of constant volume" (according to C.E. Shannon):

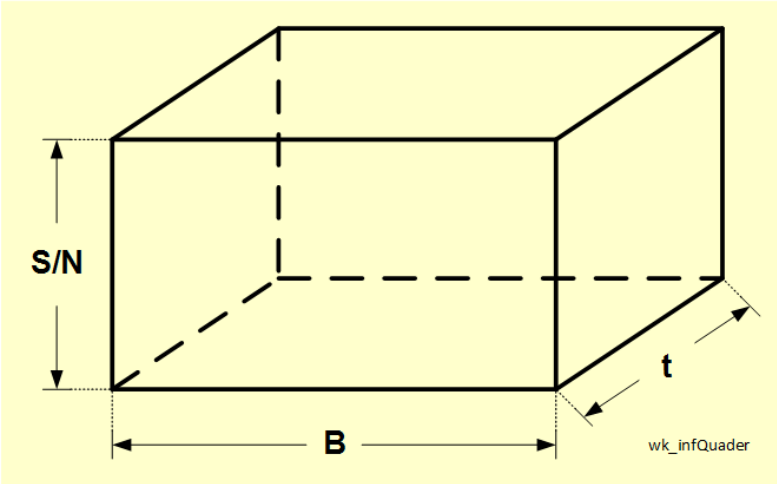


Fig. 5.5.4-1: Information cube according to C.E. Shannon

5.5.5 Throughput and transfer time

The effective end-to-end throughput S of a network is defined as the tatio of the transfer size (packet size) P and the associated transfer time $t_{Transfer}$:

$$S = \frac{P}{t_{Transfer}}.$$

For the transfer time applies in the case of a request message (with reconfirmation):

$$t_{Transfer} = RTT + \frac{P}{B} + t_{Wait}.$$

Exercises

E.5.5.5-1: Calculate the necessary transmission time for a smallest possible and a largest possible data frame in a network based on Ethernet, Fast Ethernet and Gigabit Ethernet technology.

Solutions:
Min. frame length: $67.20 \cdot 10^{-6}$ s, $6.720 \cdot 10^{-6}$ s, $4.256 \cdot 10^{-6}$ s,
Max. frame length: $1230.4 \cdot 10^{-6}$ s, $123.0 \cdot 10^{-6}$ s, $12.304 \cdot 10^{-6}$ s.

E.5.5.5-2: How many frames can be transmitted per second in each case (cases as in task A.5.5.5-1)?

Solutions:
Min. frame length (frames/s): 14 880, 148809, 234962,
Max. frame length (frames/s): 812, 8130, 81274.

E.5.5.5-3: What is the relative bandwidth utilization in the different cases (cases as in task A.5.5-1)?

Solutions:
Min. frame length (rel. bandwidth utilization): 0.5476, 0.5476, 0.0865,
Max. frame length (rel. bandwidth utilization): 0.9744, 0.9756, 0.9753.

E.5.5.5-4: A transcontinental computer network with a bandwidth of 1 Gbit/s has a round trip time of 100 ms. Files with the sizes 1 MByte and 10 MByte are to be transferred. Waiting times need not be considered. How large are the throughputs?

Solutions:
1 MByte file: 77.39 Mbit/s,
10 Mbyte file: 456.19 Mbit/s.

5.5.6 Considerations on efficiency

Various access protocols for multiple access are used at the data link layer in different network types.
Some protocols are to be quantitatively examined with regard to their efficiency.

5.5.6.1 Pure ALOHA technique

An early method for solving channel assignment, the ALOHA method, was developed by engineer Norman Abramson in 1970 at the University of Hawaii to implement a computerized, packet-switched wireless network. In this process, uncoordinated users compete for the use of a single channel. The method is considered basic because the core ideas developed can be applied to any other system. It later inspired Bob Metcalfe to develop Ethernet technology using CSMA/CD access methods.

The Pure ALOHA procedure (Pure ALOHA) is based on the following simple set of rules:

- Each data station sends its data frames without regard to already existing activities in the network.
- In case of collisions, the sent data frames are destroyed.
- By listening to the channel, the sender can detect the destruction of the data frames it has sent.
- In case of collision of data frames, the transmitting station repeats a new transmission attempt after a delay time randomly selected from a time interval.

For quantitative evaluation, consider a normalized throughput S:

$$S = \frac{\lambda P}{B}$$

with λ: flow rate (Frames/(time unit)),
P: packet length,
B: bandwidth of the data channel.

Assume that all packets have the same length and the transmission time t_P.

In the case S > 1, users produce frames faster than they can be sent over the data channel. Almost every frame suffers a collision. Reasonable normalized throughputs are in the range 0 < S < 1.

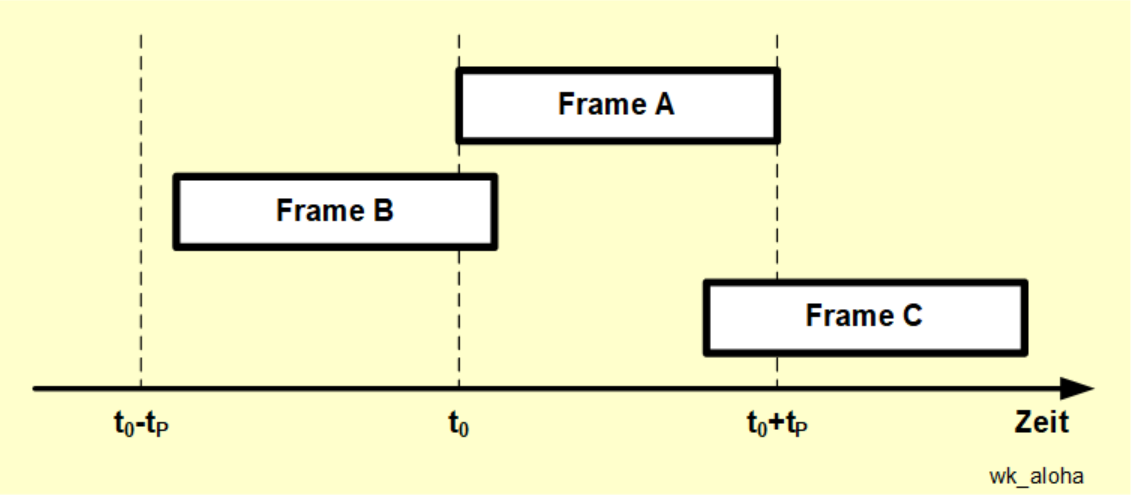


Fig. 5.5.6.1-1: Collision cases during frame transfer with Pure ALOHA

If the data packets overlap in time, the transmission process must be repeated. The probability p_k that k transmission attempts take place in the time span t_P is modeled in form of a Poisson distribution with the expectation value G:

$$p_k = \frac{G^k e^{-G}}{k!}.$$

G is the average number of frames per frame time t_P (average throughput) after summing up all originally sent and resent frames in case of collisions. It follows

$$G \geq S.$$

When the network load is low ($S \approx 0$), few collisions occur. The approximation $S \approx G$ applies.

The throughput is calculated from the product of the expected data frames and the probability that no additional transmission attempts occur in the time intervals $[t_0 - t_P, t_0]$ and $[t_0, t_0 + t_P]$ before and after the current time t_0:

$$S = G p_0(t_0 - t_P) p_0(t_0 + t_P)$$
$$S = G e^{-2G}.$$

Exercises

E.5.5.6.1-1: What is the mean throughput G in the case of maximum efficiency in the Pure ALOHA process? [0.5]

E.5.5.6.1-2: What is the maximum channel utilization at maximum efficiency in the Pure ALOHA process? [0.1839]

5.5.6.2 Slotted ALOHA technique

To increase the efficiency of the pure ALOHA procedure, the introduction of the subdivided ALOHA procedure (Slotted ALOHA) was proposed in 1972.

The following rules apply to the operation of the Slotted ALOHA procedure:

- The time axis is divided into time sections of fixed length (slots).
- The data frames must not exceed the length of a slot.
- Data frames may only to be sent at the beginning of a slot.
- Furthermore, the rules of the pure ALOHA technique apply.

In contrast to the Pure ALOHA method, collisions can now only occur within a time slot $[t_0, t_0 + t_S]$ of length t_S.

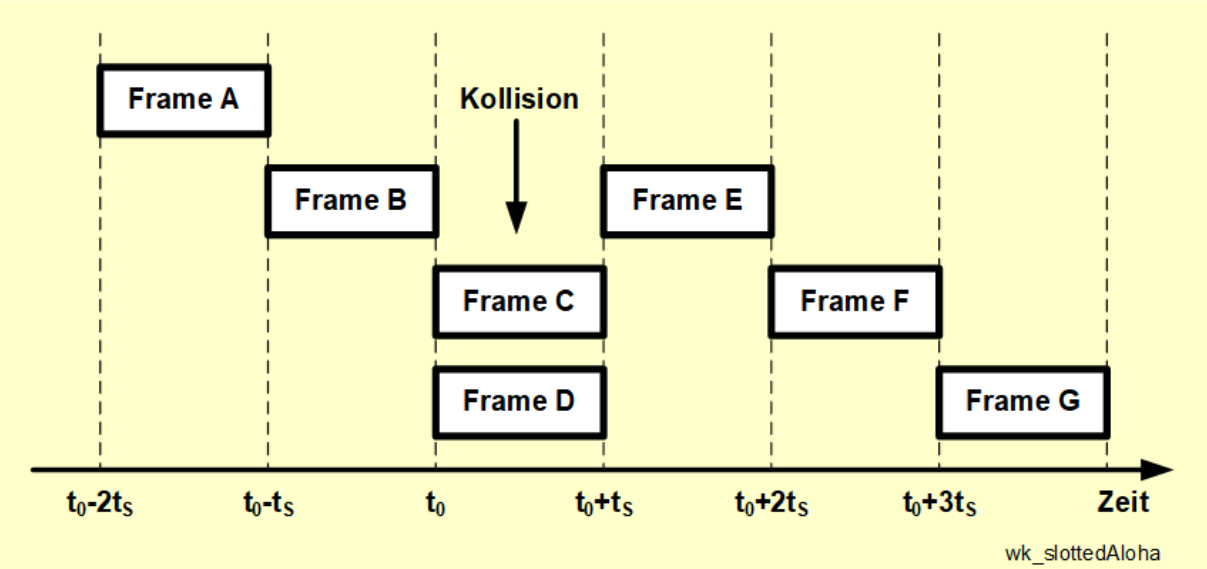


Fig. 5.5.6.2-1: Frame transfer with Slotted ALOHA

In analogy to the consideration of the Pure ALOHA process, the following follows for the throughput

$$S = Gp_0(t_0 + t_S)$$
$$S = Ge^{-G}.$$

Exercises

E.5.5.6.2-1: What is the average throughput G in the case of maximum efficiency in the Slotted ALOHA process? [1.0]

E.5.5.6.2-2: What is the maximum channel utilization in case of maximum efficiency in the Slotted ALOHA process? [0.3679]

5.5.6.3 Considerations for Ethernet Efficiency

Estimates are to be made on the Ethernet protocol (IEEE 802.3) using the CSMA method under high and constant utilization.

The following assumptions are made for the estimation:

- Let there be a total of k active stations.
- Due to the complex behavior of the exponential backoff algorithm, the assumption is to be made that in the case of a collision, each active channel attempts a repeated transmission with constant probability p.

Let the probability that a node transmits in a given time slot be p. At the same time, the probability that the other nodes in the same slot do not transmit is equal to

$$(1-p)^{k-1}.$$

The probability that a particular node is successful is thus given by

$$p(1-p)^{k-1}.$$

If there are k nodes in total, the probability A is that exactly one node in the network is successful,

$$A = kp(1-p)^{k-1}.$$

Exercise

A.5.5.6.3-1: Find the optimal transmission probability at which the probability that a node succeeds in a network with k nodes becomes maximum.

For networks with very many data stations follows

$$\lim_{k\to\infty} A = \frac{1}{e}.$$

The probability that a competition interval includes just j time slots is

$$A(1-A)^{j-1}.$$

The average number of time slots per competition interval is thus given by

$$\sum_{j=0}^{\infty} A(1-A)^{j-1} = \frac{1}{A}.$$

With the slot length t_{Slot}, the mean competition interval is

$$w = \frac{1}{A} t_{Slot}.$$

Assuming the optimal probability p = p*, the mean number of average competition intervals never exceeds e. For the mean competition interval, the result is as follows:

$$w = t_{Slot} e.$$

If the average transmission time for a frame is t_{Packet}, the efficiency E of the channel is given by

$$E = \frac{t_{Packet}}{t_{Packet} + w} = \frac{t_{Packet}}{t_{Packet} + \frac{t_{Slot}}{A}}.$$

Exercise

E.5.5.6.3-2: Determine the efficiency as a function of frame length, network bandwidth, spatial extent, and phase velocity of the signal in the optimal case within networks with very many active nodes.

5.6 Error detection with polynomial codes

5.6.1 Numerical calculation

For effective error correction, additional **error detection codes** are integrated into the data frames during digital data transmission and a retransmission is requested if necessary.

In practice, **polynomial codes (cyclic redundancy codes CRC)** are mainly used. A checksum is appended to the frame to be transmitted. It is calculated in such a way that the value of the frame represented by the bit sequence is divisible by a given generator polynomial known to the sender and receiver without a remainder. If a remainder occurs during the division, at least one transmission error is present.

In the calculation bit strings are considered as polynomials ("message polynomials") with the coefficients 0 and 1. In the presence of a data frame with a length of k bits, the description is made by a (k-1)-th order polynomial with the coefficients according to the bit sequence.
A bit sequence with k places is represented by a message polynomial M(x) with the coefficients $a_i \in [0, 1]$ and the place values x_i, $i \in [0, ..., k-1]$:

$$M(x) = \sum_{i=0}^{k-1} a_i \cdot x^i \; .$$

Example: Bit sequence of 7 bits: **1101011**

Polynomial of order 6: $\mathbf{M(x) = x^6 + x^5 + x^3 + x^1 + 1}$

The calculations are performed according to the rules of the algebraic field theory modulo 2. Carryovers and borrow bits in addition and subtraction are omitted. Addition and subtraction are both realized by the exclusive-or operation.

The CRC checksum for a data frame with m bits represented by a polynomial M(x) is determined according to the algorithmic steps below:

- Definition of a generator polynomial G(x):
  - most significant and least significant generator bit equal to 1,
  - order r of the generator polynomial G(x) < frame length with m bits,

- Multiplication of the polynomial M(x) by x^r: $x^rM(x)$
  - append r zero bits to the least significant side of the frame,
  - resulting total length of the frame: m+r bits,

- Execution of the modulo-2 division $x^rM(x)/G(x)$,

- modulo-2 subtraction of the resulting remainder from the bit string corresponding to $x^rM(x)$ yields the sum-set checked frame T(x), which is transmitted.

The receiver divides the received frame T(x) by the generator potential G(x). If a division remainder occurs, there is at least one transmission error.

In practice, checksum calculation is usually implemented by hardware units with feedback shift register circuits.

International standards define different generator polynomials.

Examples:

Norm	Generator polynomial	Use
CRC-8	$G(x) = x^8 + x^2 + x^1 + 1$	ATM
CRC-10	$G(x) = x^{10} + x^9 + x^5 + x^4 + x^1 + 1$	ATM
CRC-12	$G(x) = x^{12} + x^{11} + x^3 + x^2 + 1$	
CRC-16 (IBM)	$G(x) = x^{16} + x^{15} + x^2 + 1$	
CRC-CCITT (ITU)	$G(x) = x^{16} + x^{12} + x^5 + 1$	HDLC
Bluetooth	$G(x) = x^5 + x^4 + x^2 + 1$	Bluetooth
CRC-32 (IEEE 802)	$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$	Ethernet, FDDI

Exercise

E.5.6-1: Calculation of the CRC checksum for a data frame

Example: Frame: `11010111`
Generator: `1011`

The frame to be transmitted including the error detection code is searched for.

With the use of different generator polynomials, various transmission errors can be detected. The rules listed below can be proven in general:

- All single bit errors can be detected as long as the coefficients of the terms x^r and x^0 are different from zero.
- All double bit errors are detected if the generator polynomial consists of at least three terms.
- All odd errors are detected if the generator polynomial contains the term combination (x+1).
- All burst errors, i.e. sequences of consecutive single bit errors, are detected if the bursts consist of less than r bits.

5.6.2 Hardware realization

To optimize speed, CRC error checking is usually performed in practice on a hardware basis using k-bit shift registers in combination with XOR gates (Fig. 5.5.2-1).

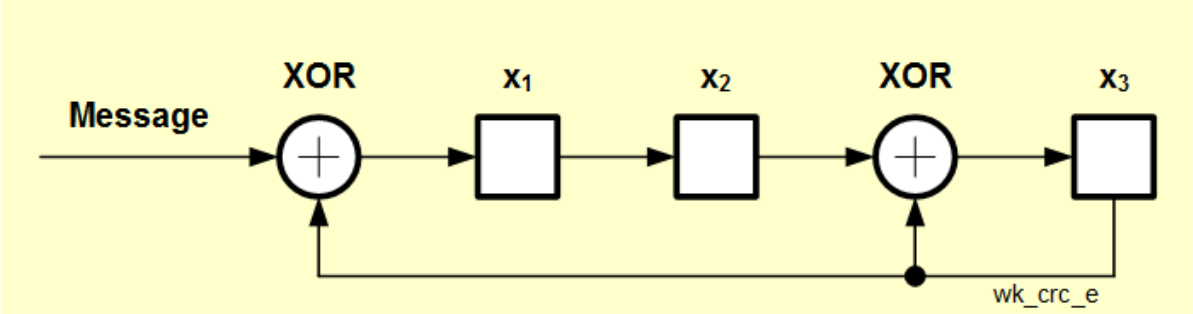


Fig. 5.6.2-1: Shift register for CRC calculation

The number of bits in the shift register is equal to the degree k of the generator polynomial.

The bit sequence of the message is appended with k zeros and inserted into the shift register. After the XOR operations have been executed, the shift register contains the remainder, i.e. the CRC code.

The individual bits in the shift register are indexed 1 to k from left to right. XOR gates are inserted before those bits where there is a term x^k in the generator polynomial.

Fig. 5.6.2-1 shows for example the structure of the shift register for the generator polynomial

$$G(x) = x^3 + x^2 + 1.$$

5.7 Protocol examples for the implementation of the data link layer

5.7.1 Introduction

The physical layer and data link layer protocols are sometimes integrated in input/output units of special microprocessors. In some cases, they are also implemented in software in a high-level language. The examples discussed in this chapter are intended to give a first insight into the programming technique for layer protocols.

The following conditions are assumed:

- The physical layer, data link layer and network layer are regarded as separate processes.
- Only reliable, connection-oriented services between two stations (sender and receiver) are possible.
- The data to be sent is unlimited.
- There are no waiting times for the provision of data.
- The data link layer waits for an event, e.g. the arrival of a frame, and is only activated afterwards.
- The length of the exchanged packets between the network layer and the link layer is constant (MAX_PKT).

The following header file applies to all models discussed below:

```
/*  Header file for the simulation of protocols for the      */
/*  data link layer                                          */
/*  Source: A.S. Tanenbaum, Computernetzwerke,               */
/*  Prentice Hall: Muenchen, London, u.a. 1998               */
/*  File:  protocol.h                                        */

#define MAX_PKT 1024                    /* Packet sizes
                                           in Bytes          */
typedef enum {false, true} boolean;     /* Boolean Type      */
typedef unsigned int seq_nr;            /* Sequence- and
                                           acknowledgement
                                           number            */
typedef struct                          /* Packet definition */
        { unsigned char data[MAX_PKT];
        } packet;
typedef enum {data, ack, nak} frame_kind; /* Frame type      */

typedef struct                            /* Frame format    */
        { frame_kind kind;                /* frame kind      */
          seq_nr seq;                     /* Sequence number */
          seq_nr ack;                     /* acknowledgement
                                             number          */
          packet info;                    /* data part       */
        } frame;
```

```
/* Wait for event, Return type of event */
void wait_for_event(event_type *event);

/* Retrieve packet from network layer for transmission */
void from_network_layer(packet *p);

/* Send information to network layer */
void to_network_layer(packet *p);

/* Retrieve input frame from physical layer and copy to r */
void from_physical_layer(frame *r);

/* Send data frame to physical layer for transmission */
void to_physical_layer(frame *s);

/* Start timer and enable timeout result */
void start_time(seq_nr k);

/* Stop timer und suppress timeout result */
void stop_timer(seq_nr k);

/* Start auxiliary timer and enable event ack_timeout */
void start_ack_timer(void);

/* Stop auxiliary timer and suppress event ack_timeout */
void stop_ack_timer(void);

/* Enable generation of event networ_layer_ready by the network
   layer */
void enable_network_layer(void);

/* Prevent the generation of the event network_layer_ready
   by the network layer */
void disable_network_layer(void);

/* Internal extension of the macro "inc":
   circular incrementation of k */
#define inc(k) if(k<MAX_SEQ)k++;else k=0;
```

5.7.2 Unrestricted simplex protocol

The following protocol 1 describes an ideal simplex connection with the following properties:

- Transmitter and receiver are ready at any time.
- All processing times are neglected.
- The buffer areas are sufficiently large.
- No errors and/or losses occur during data transmission.

Transmitter and receiver are each realized as functions.

Example:

```
/* Protocol 1: Utopia,
   Unrestricted simplex protocol                                    */
/* File: utopia.c (P1.c)                                            */

#include "protocol.h"

typedef enum {frame_arrival} event_type;

void sender1(void)
{
   frame s;           /* buffer for the outgoing frame             */
   packet buffer;     /* buffer for the outgoing packet            */

   while(true)
   {
      from_network_layer(&buffer);  /* get packet to send          */
      s.info = buffer;              /* copy for transfer           */
      to_physical_layer(&s);        /* send info                   */
   }
}

void receiver1(void)
{
   frame r;
   event_type event;

   while(true)
   {
       wait_for_event(&event);      /* only option:
                                       frame_arrival               */
       from_physical_layer(&r);     /* receive the incoming
                                       frame                       */
       to_network_layer(&r.info);   /* forwarding to
                                       network layer               */
   }
}
```

5.7.3 Finite resources of the receiver

In reality, the receiver needs a finite time for data processing. In addition, the data buffer has only a finite size. The sender must therefore only operate at a maximum data transmission speed that does not overload the receiver properties.

To optimize the transmission speed and to avoid data flooding in the communication network, the receiver, after forwarding the data frame to the network layer, sends an empty frame back to the sender: **acknowledgement frame**. Protocols of this type are called **stop-and-wait protocols**.
Technically, half-duplex operation is required for this protocol.

In the simplest version, the sender does not have to check the acknowledgement frame, since there is only one way of acknowledging.

Example:

```
/* Protocol 2: Stop-and-Wait-Protokoll                          */
/* File:  stopwait.c (P2.c)                                     */

#include "protocol.h"

typedef enum {frame_arrival} event_type;

void sender2(void)
{
   frame s;           /* buffer for the outgoing frame        */
   packet buffer;     /* buffer fot the outgoing package      */
   event_type event;  /* only option: frame_arrival           */

   while(true)
   {
      from_network_layer(&buffer); /* retrieve to send        */
      s.info = buffer;             /* copy for transfer       */
      to_physical_layer(&s);       /* send to physical layer  */
      wait_for_event(&event);      /* wait for frame_arrival  */
   }
}
```

```
void receiver2(void)
{
   frame r, s;
   event_type event;

   while(true)
   {
       wait_for_event(&event);     /* only option:
                                      frame_arrival           */
       from_physical_layer(&r);    /* retrieve incoming frame */
       to_network_layer(&r.info); /* forwarding to network
                                      layer                   */
       to_physical_layer(&s);      /* send an empty frame to
                                      sender                  */
   }
}
```

5.7.4 Consideration of noisy transmission media

Transmission media always contain noise sources. Data frames can be damaged during transmission, or they can be lost.

Exercise

E.5.7.4-1 Assuming that the receiver can evaluate a sent data frame for correctness after calculating the checksum, the following suggestion is proposed for data transmission with an unreliable communication channel:

- A timer is integrated to the sender, which is set when a message is sent.
- The receiver sends a confirmation frame if the message is received correctly.
- If the sender does not receive a confirmation frame within a specified time interval, the message is sent again and the timer is reset.

Is this method suitable for secure and error-free communication?

For error-free transmission, the receiver must be able to distinguish between the first reception of a message and the repetitions. To realize this, the sender provides each frame to be sent with a sequence number that the receiver can check. Only when frames with the correct sequence number are received a confirmation frame is sent back to the sender.

Protocols where the sender waits for a positive acknowledgement from the receiver before sending are called **PAR protocols (Positive Acknowledgement with Retransmission)** or **ARQ protocols (Automatic Repeat reQuest)**.

In the PAR protocol below, the sender stores a sequence number for the next frame to be sent (next_frame_to_send). The receiver works with a sequence number for the next frame to be expected (frame_expected).

```
/* Protocol 3: PAR-Protocol,
   Data transmission via an unreliable data channel          */
/* File:  noise.c (P3.c)                                     */

#include "protocol.h"

#define MAX_SEQ 1

typedef enum {frame_arrival, cksum_err, timeout} event_type;

void sender3(void)
{
   seq_nr next_frame_to_send;     /* sequence number for
                                     next frame              */
   frame s;                       /* buffer for outgoing
                                     frame                   */
   packet buffer;                 /* buffer for outgoing
                                     packet                  */
   event_type event;
   next_frame_to_send=0;          /* output sequence number  */

   from_network_layer(&buffer);   /* retrieve package from
                                     from network layer      */

   while(true)
   {
      s.info = buffer;            /* frame creation          */
      s.seq = next_frame_to_send; /* inserting the sequence
                                     number in frame         */
      to_physical_layer(&s);      /* send                    */
      start_timer(s.seq);         /* abort if waiting time is
                                     too long                */
      wait_for_event(&event)      /* wait for frame_arrival  */
```

```
      if (event == frame_arrival)
      {
         from_network_layer(&buffer); /* retrieve the next
                                         frame               */
         inc(next_frame_to_send);     /* increment sequence
                                         number              */
      }
   }
}


void receiver3(void)
{
   seq_nr frame_expected=0;
   frame r, s ;
   event_type event;

   while(true)
   {
       wait_for_event(&event);        /* "frame_arrival" or
                                         "cksum_err"         */
       if (event == frame_arrival)
       {
         from_physical_layer(&r);     /* get the incoming
                                         frame               */
         if (r.seq == frame_expected)
         {
           to_network_layer(&r.info); /* forward to network
                                         layer               */
           inc(frame_expected);       /* Increment
                                         sequence number     */
         }
         to_physical_layer(&s);       /* send an empty frame
                                         to physical layer   */
       }
   }
}
```

5.7.5 Protocol with variable window size

For bidirectional data transmission between two stations, only one data link should be used if possible to make optimum use of the transmission capacity. Data and acknowledgement frames can be mixed during transmission. Various techniques help to recognize the transmitted information:

- Inclusion of a identification field in the frame header,

- Delaying the sending of the acknowledgement frame by the receiver until the receiver's network layer wants to send new data, and incorporating the acknowledgement frame for the previous reception into the frame to be sent **(piggyback transport, piggybacking)**,

- Sending a separate acknowledgement frame if a previously defined waiting time is exceeded.

To maintain synchronization in case of timing errors, garbled or lost frames, **variable window size protocols (sliding window protocols)** with the following properties are used:

- Sender constantly keeps a list of consecutive sequence numbers:
  send window:
  - Number of sequence numbers = number of frames that may be sent,
  - sequence numbers in the current transmission window correspond to those frames that have been sent but not yet acknowledged,
  - each new packet from the network layer is assigned by the next free sequence number, upper limit of the send window is moved by one,
  - moving the lower limit when an acknowledgement is received,
  - transmission window always contains a list of unacknowledged transmitted frames,
  - blocking of the transmitter's network layer if the maximum size of the transmit window is reached,

- Receiver uses a r**eceive window** for the number of frames that may be received:
  - Destruction of each frame that is not received in the receive window,
  - forwarding to the switching layer of a frame received without errors and if its sequence number corresponds to the lower limit of the window,
  - in this case simultaneous sending of an acknowledgement frame to the sender and incrementation of the current receive window content,
  - the size of the receive window is always constant.

The function slide.c below shows a possible implementation for a **1-bit sliding window protocol**. In this case the protocol satisfies the stop-and-wait procedure. The sender waits for the corresponding confirmation after each sending of a frame.

```
/* Protocol 4: Bidirectional „Sliding window" protocol,
   1-Bit-sliding window protocol                              */
/* File:  slide.c (P4.c)                                      */

#include "protocol.h"

#define MAX_SEQ 1

typedef enum {frame_arrival, cksum_err, timeout} event_type;

void protocol4(void)
{  seq_nr next_frame_to_send;    /* sequence number for next
                                    frame: 0 or 1            */
   seq_nr frame_expected;        /* 0 oder 1                 */
   frame r, s;                   /* buffer for outgoing
                                    frame                    */
   packet buffer;                /* buffer for outgoing
                                    packet                   */
   event_type event;
   next_frame_to_send=0;         /* sequence number          */
   frame_expected=0;             /* number for expected
                                    frame                    */
   from_network_layer(&buffer);  /* get the first package    */
   s.info = buffer;              /* frame creation           */
   s.seq = next_frame_to_send;   /* insert sequence number
                                    into frame               */
   s.ack = 1 - frame_expected;   /* piggybacking
                                    confirmation             */
   to_physical_layer(&s);        /* send to physical layer   */
   start_timer(s.seq);           /* abort if waiting time is
                                    too long                 */


   while(true)
   {
      wait_for_event(&event);
      if (event == frame_arrival)
      {
         from_physical_layer(&r);       /* get frame form
                                           physical layer    */
         if (r.seq == frame_expected)   /* processing the
                                           incoming frame    */
         {  to_network_layer(&r.info);  /* forwarding to
                                           network layer     */
            inc(frame_expected);        /* increment
                                           sequence number
                                           of expected frame */
         }
         if (r.ack == next_frame_to_send)  /* processing the
         {                                 /* outgoing frame */
            from_network_layer(&buffer);   /* Retrieve next
                                              frame          */
```

```
            inc(next_frame_to_send);    /* increment
                                           sequence number
                                           of frame to send  */
         }
      }
      s.info = buffer;                  /* prepare packet
                                           information       */

      s.seq = next_frame_to_send;       /* add seqence number */

      s.ack = 1 - frame_expected;       /* update
                                           acknowlegement
                                           number            */

      to_physical_layer(&s);            /* send to physical
                                           layer             */

      start_timer(s.seq);                /* start timer      */
   }
}
```

5.8 Network hardware components on the data link layer

5.8.1 Bridges

Bridges and **switches** are used as active network components in the data link layer.

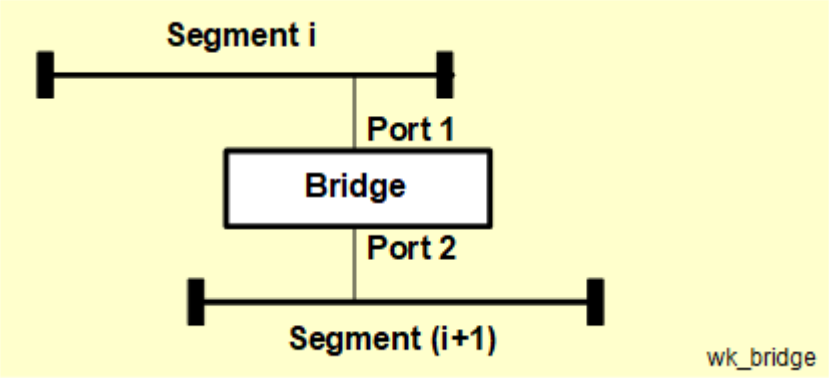


Fig. 5.8.1-1: Connection of two network segments with one bridge

Bridges fulfill a variety of tasks:

- (Theoretically infinite) expansion of a network in terms of its spatial extent and maximum number of stations,
- Enabling simple error mitigation by preventing the forwarding of data frames with erroneous MAC frames,
- Limitation of local traffic (filtering of frames with respect to their address: possibility of local restriction to LANs, reduction of total network load, channelization of security-related traffic).

Bridges operate as bidirectional receive/transmit devices on layer 2 according to the ISO/OSI protocol stack. They allow the connection of multiple local area networks (LANs). They can connect networks with different physical layers. However, layers 2 to 7 (ISO/OSI) must be identical for the connected subnetworks.

Depending on the protocol layer, a distinction is made between MAC bridges (actual standard bridges) on the MAC layer for coupling local networks and LLC bridges on the LLC layer for implementing LAN associations via wide area networks (WANs) or other backbone networks.

Other classification features are the number of ports (2-port bridges, multi-port bridges), configurability (configurable bridges, self-learning bridges), spatial coverage (local bridges, remote bridges), route discovery capability (transparent bridges, source routing bridges).

The table below provides an overview of the main differences between transparent bridges and source routing bridges:

Transparent Bridge	**Source Routing Bridge**
Networking *IEEE-802.3/Ethernet-LANs*	Networking LANs over WANs
Ability to lean	No own learning ability
Path finding is based on the MAC address and the spanning tree algorithm	Route selection is based on encapsulation method *(Tunneling)*

Filter functions in bridges are defined by entries, e.g. the MAC addresses, in a filtering database in the bridge. The filter mask entries can be made manually or automatically as a result of learning processes. If the data frames are forwarded when the frame properties match the database entries, this is referred to as a positive filter; if blocked, it is referred to as a negative filter.

In complex networks where multiple subnets are connected via bridges, MAC frames could circulate multiple times or permanently in topological loop arrangements, since hardware addresses do not provide information about local network membership.

A "loop suppression algorithm", the so-called **spanning tree algorithm** (ST algorithm, IEEE 802.1d) detects redundant bridges, separates them logically from the network and thus ensures loop suppression. The entire network structure is mapped as a data structure in the form of a tree. Minimum spanning trees are tree structures that connect all nodes of a graph with the smallest number of paths and minimum total weight of edges.

Information is exchanged between the bridges by sending so-called configuration information, the configuration BPDUs (Configuration Bridge Protocol Data Unit, hello packet) via multicast addresses. Each bridge has a six-byte unique identifier (bridge ID). Each bridge port is uniquely identified with a port identification number (port ID).

Bridges operate according to the store-and-forward principle. Complete data frames are read in sequentially and their checksum is calculated. Error-free frames are forwarded after the check, while faulty frames are discarded.

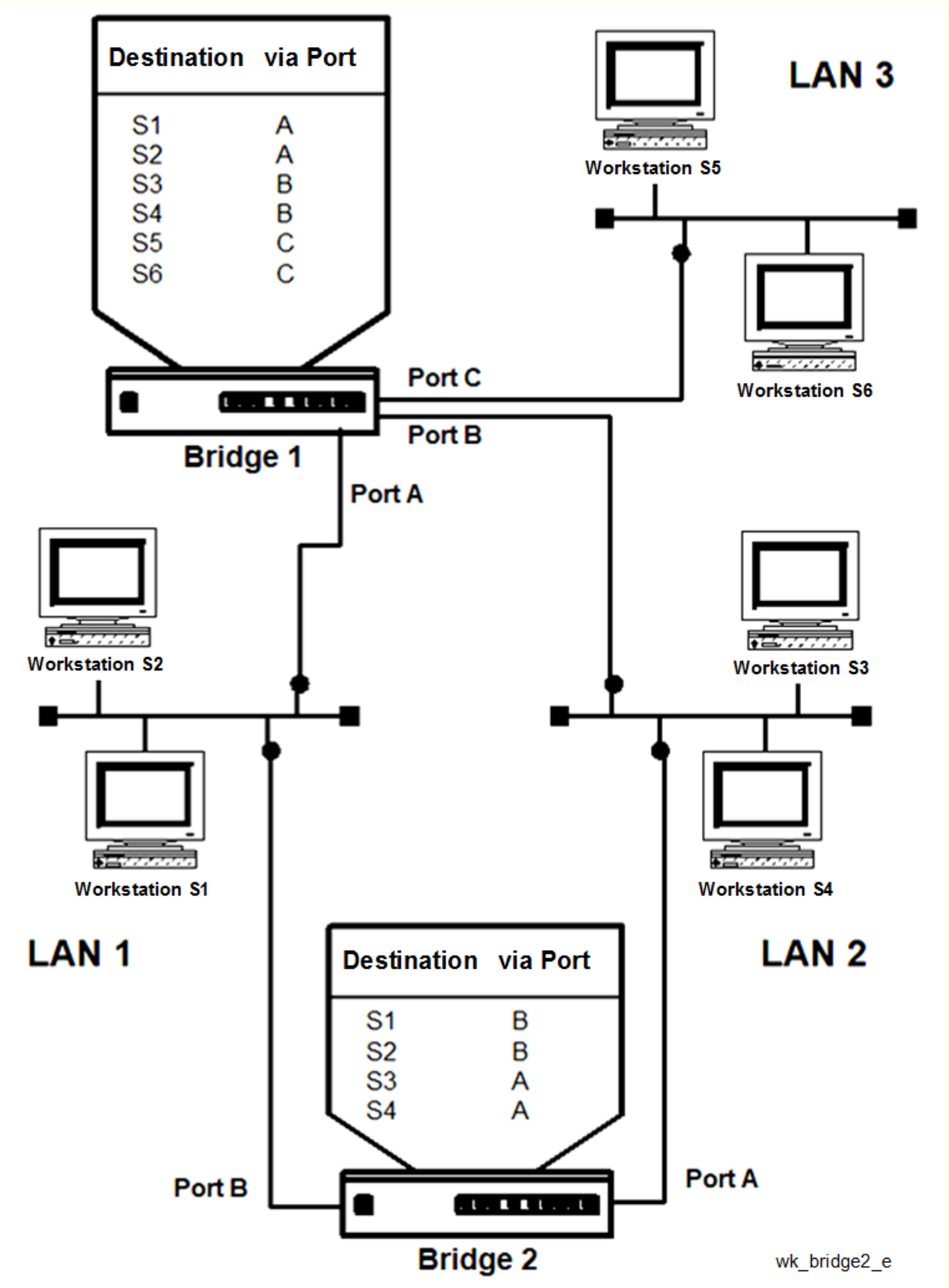


Fig. 5.8.1-2: Internetworking with Bridges
(Sx: MAC addresses)

5.8.2 Switches

5.8.2.1 Basics

Switches are network components that partially combine the properties of repeaters, bridges and routers. They serve to increase the transmission performance. The use of switches in Ethernet networks requires a star-shaped topography.

The most important properties of switches are summarized in the following list:

- High processing speed (almost "wire speed") due to advanced hardware components,
- Provision of the full bit rate of the transmission protocol for all connected devices,
- Support of full duplex operation with independent channels for transmit and receive directions,
- Switched LANs represent broadcast domains in which physical layer errors are restricted to the originator, in contrast to repeater technology.

Logically, switches simulate a bridged local area network with one computer per segment. Internal electronic circuits simulate the bridge function. The figure below illustrates the principle.

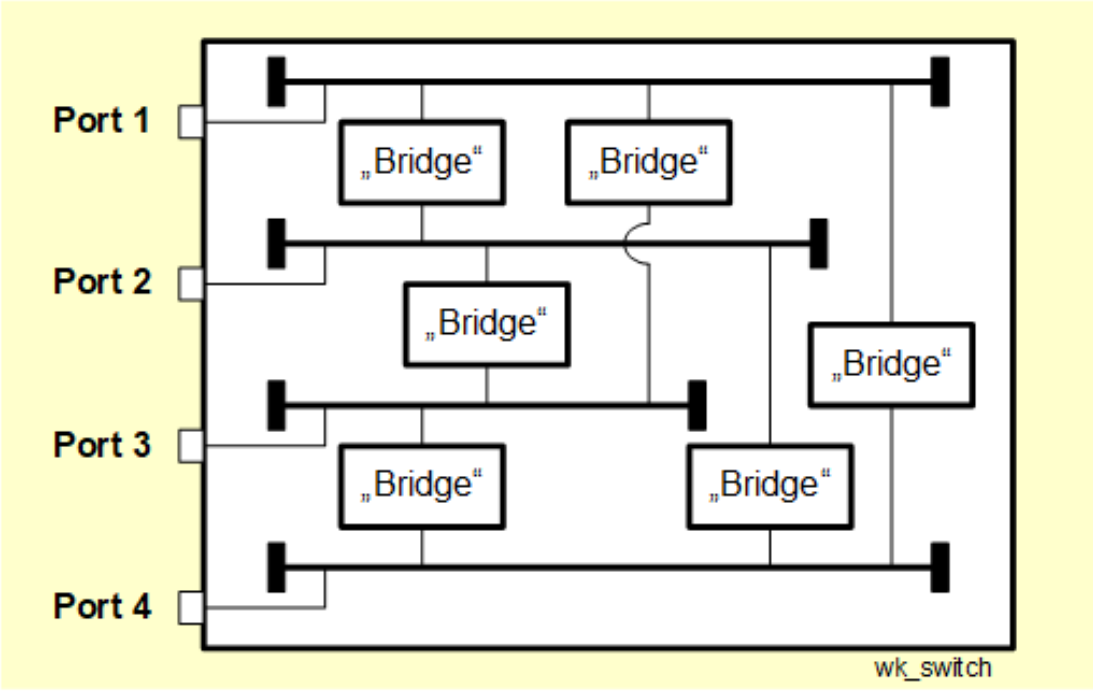


Fig. 5.8.2.1-1: Concept of building a switch consisting of several electronically simulated bridges

A network component, i.e. a computer, or another local network can be connected to each port.

Switches operate with two fundamentally different modes of operation:

- **Store-and-forward switching:**
  As in the case of bridges, entire frames are always read in, stored temporarily, and processed further after the checksum has been checked. Frames with errors are discarded.

- **Cut-through switching:**
  The header of a frame is read only up to the destination and source address, evaluated, and then immediately forwarded to the destination address. The throughput is increased.

5.8.2.2 Network topologies with switches

5.8.2.2.1 Hierarchical model

In a hierarchical tree structure with switches, there is a grouping into several levels, often with a gradation of the data rate.

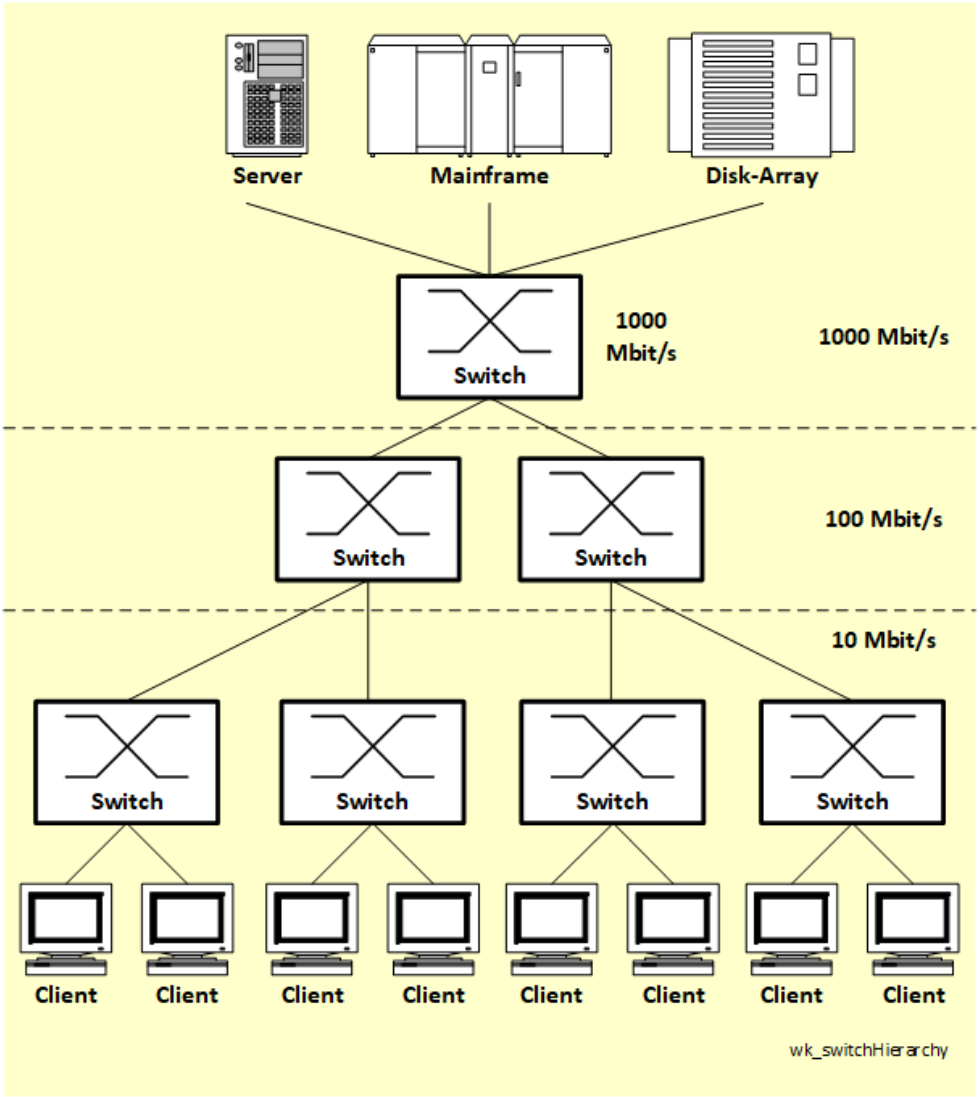


Fig. 5.8.2.2.1-1: Schematic of a hierarchically structured switch topology

Such a topological setup has the advantage that the central network components (servers, storage drives, fast processing units, etc.) used simultaneously by many clients are integrated in an environment with increased bandwidth. Access times in the central network area are drastically reduced.
In the lower client area, dimensioning adapted to the actual demand will be able reduce network costs.

5.7.2.2.2 Redundancy model

Mesh networks between the switches are used to increase the reliability of individual network connections. In the event of failure of individual connections, data exchange will continue to be possible via other paths.

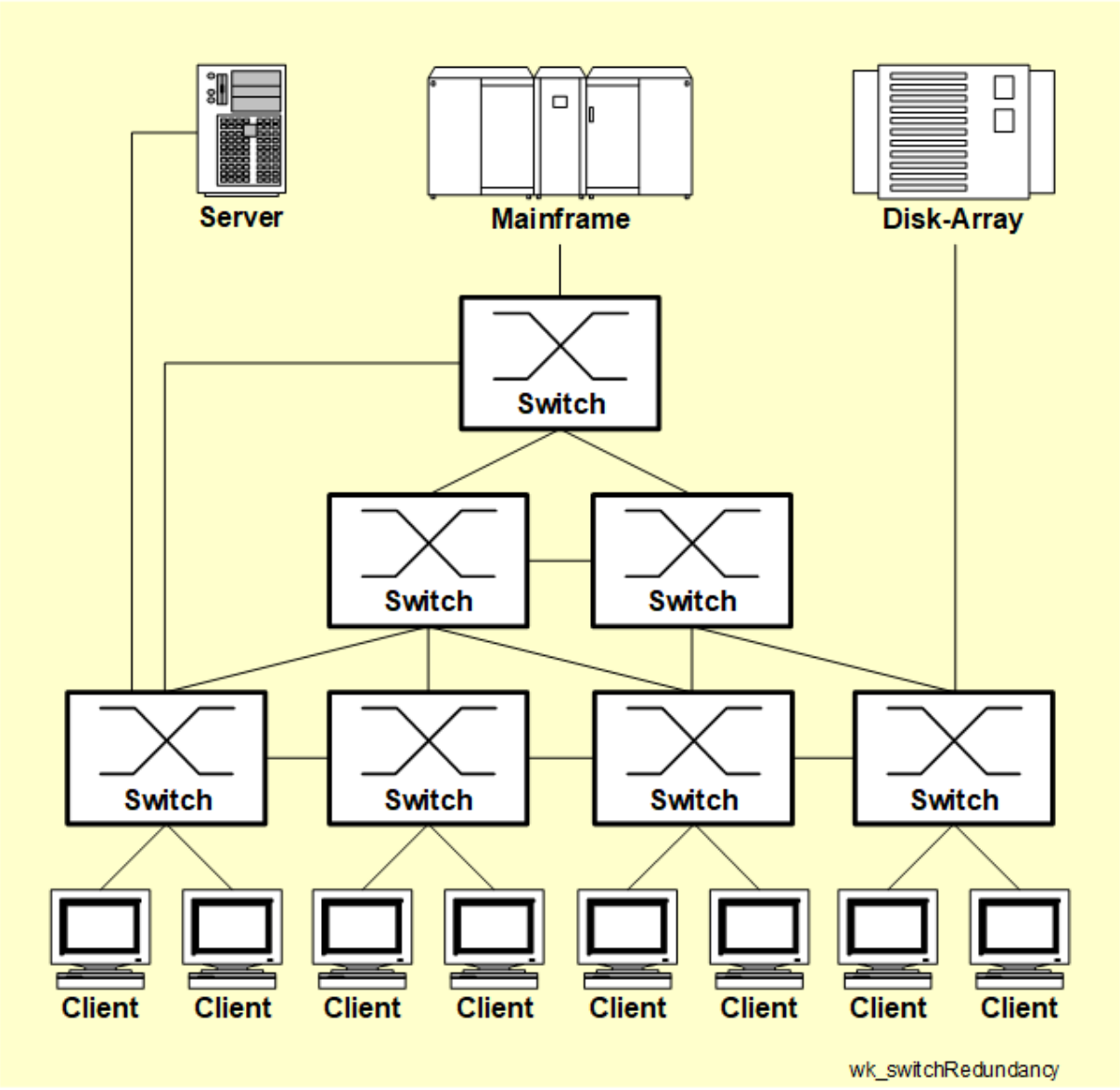


Fig. 5.8.2.2.2-1: Schematic structure of a redundant switch architecture

In meshed networks, loops inevitably occur. As in the case of brigde technology, the spanning tree algorithm (IEEE 802.1d standard) is used to suppress the loops and to clearly define the transmission paths.

The switches send spanning tree packets (Bridge Protocol Data Unit BPDU) at regular intervals via a reserved multicast address (01-80-C2-00-00-10) for topology control. The selection of the optimal connection paths is based on the lengths and the bandwidths of all network paths. IEEE 802.1d compatible switches therefore need their own bridge ID, their own MAC address. It must be possible to assign an individual priority and individual path costs (weights) to each port.

5.8.2.2.3 Link Aggregation

When using the link aggregation technique, several data lines are connected in parallel between the switches.
On the one hand, this results in an increase in bandwidth due to the bundling of several data lines. On the other hand, redundancy is improved. If one line fails, the parallel connections take care of the transport.

5.8.2.2.4 Formation of virtual local networks

The introduction of virtual local area networks, known as VLANs (Virtual Bridged Local Area Networks), leads to the optimization of network utilization. VLANs group the data flow in the entire network by limiting the broadcast domains. This relieves the data flow in the entire network.

In a simple version of VLAN realization, port-related grouping is performed. Ports of a switch defined by the administrator are assigned to a VLAN. Data exchange only takes place within a VLAN. In the process, network nodes in different physical segments can be grouped together.

The application of the IEEE 802.1q VLAN standard enables the exchange of data of a node in a VLAN beyond its boundaries. In the four-byte VLAN tag header of a data frame, which is embedded between the source address and the type field (Ethernet standard) or the length field (IEEE 802.3ac extension of the Ethernet standard), an identification code, the so-called VLAN ID, defines the assignment of a data frame to a VLAN. VLANs can be extended across multiple switches using this so-called tagging mechanism.

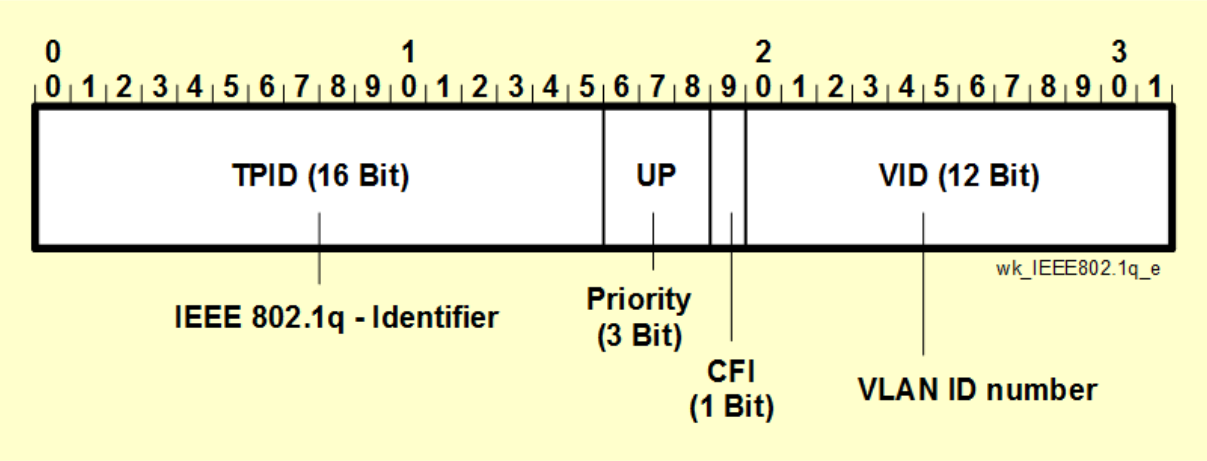


Fig. 5.8.2.2.4-1: VLAN header according to the IEEE 802.1q standard

The marking is done in the data link layer header (Ethernet header, IEEE 802.3 header):

- **TPID:**
  Tag Protocol Identifier TPID to identify the use of the IEEE 802.1q-Standard,
  Fixed value: 0x8100,
  Length: 16 Bit,

- **UP:**
  User Priority,
  Priority information for the user,
  Length: 3 Bit,

- **CFI:**
  Canonical Format Indicator,
  Length: 1 Bit,
  Possible values: 0: canonical value of the MAC address (LSB first),
  1: con-canonical format, Use in Token Ring systems, Source-Routed-FDDI-Media entries,

- **VID:**
  identification of a VLAN with a unique 12 bit long identification number VID (VLAN-Identifier, VLAN-ID),
  VLAN-IDs „0“ and „4095“ are reserved,
  VLAN ID numbers 1 to 4095 are allowed.

5.9 Network communication in the computer with the processor

5.9.1 Bus system in the Von-Neumann computer

Modern computers are originally based on the Von Neumann computer model published by the Hungarian mathematician John von Neumann. The four basic units are

- Artithmetic Logic Unit (ALU),
- Control Unit (CU),
- Memory,
- Input/Output unit (I/O).

The arithmetic unit and the control unit are combined to form the central processing unit (CPU).

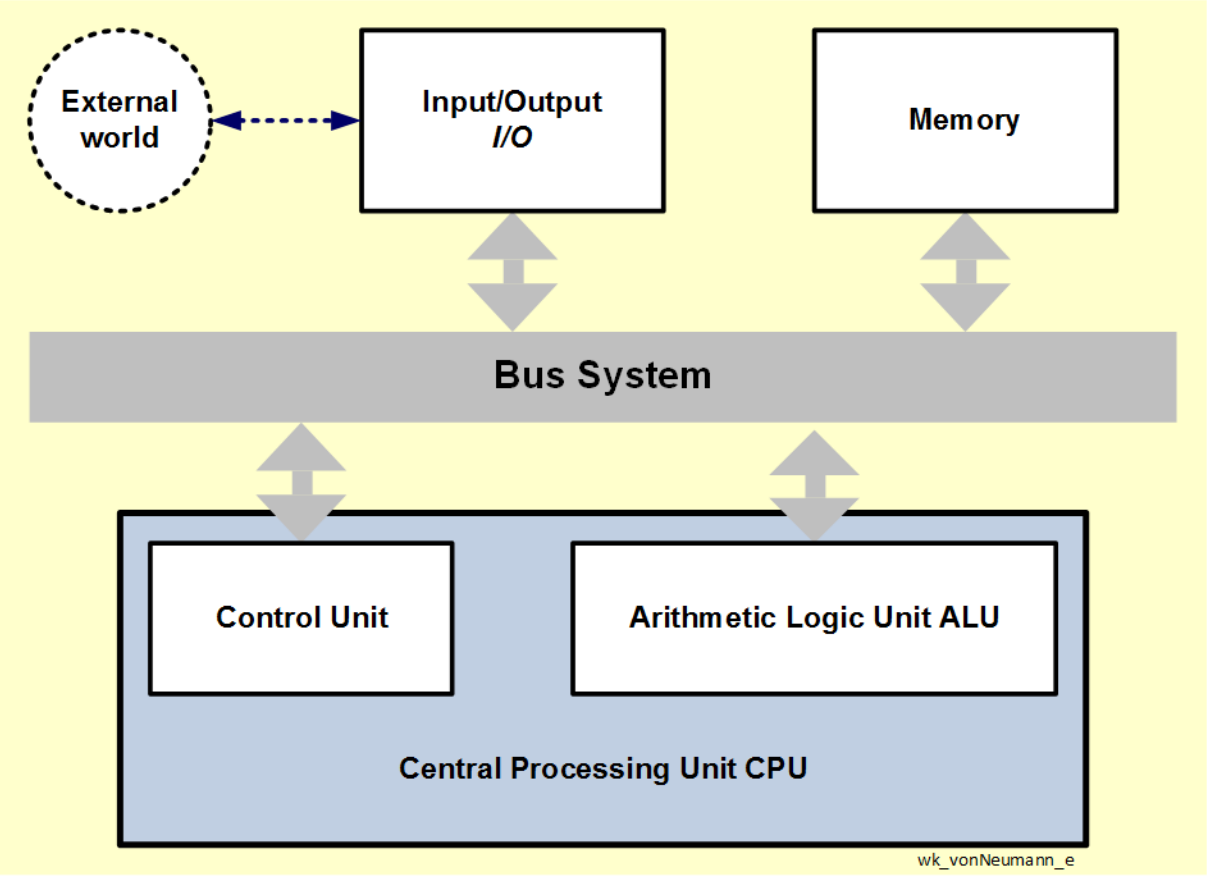


Fig. 5.9.1-1: Schematic of a Von Neumann computer system

A central bus system connects the four basic units to form a communication system. All instructions and data are transmitted via this bus system.

The CPU is responsible for the flow control and the execution of the commands. The memory stores both data and commands. Data and commands are fetched from the memory by the control unit. With the periphery (keyboard, monitor, sensors, etc.) is communicated via input/output unit.

5.9.2 PCI bus system

The most common bus system for the connection of peripheral systems to the CPU in standard computers is the PCI bus system (PCI: Peripheral Component Interconnect). The bus system operates on the data link layer.

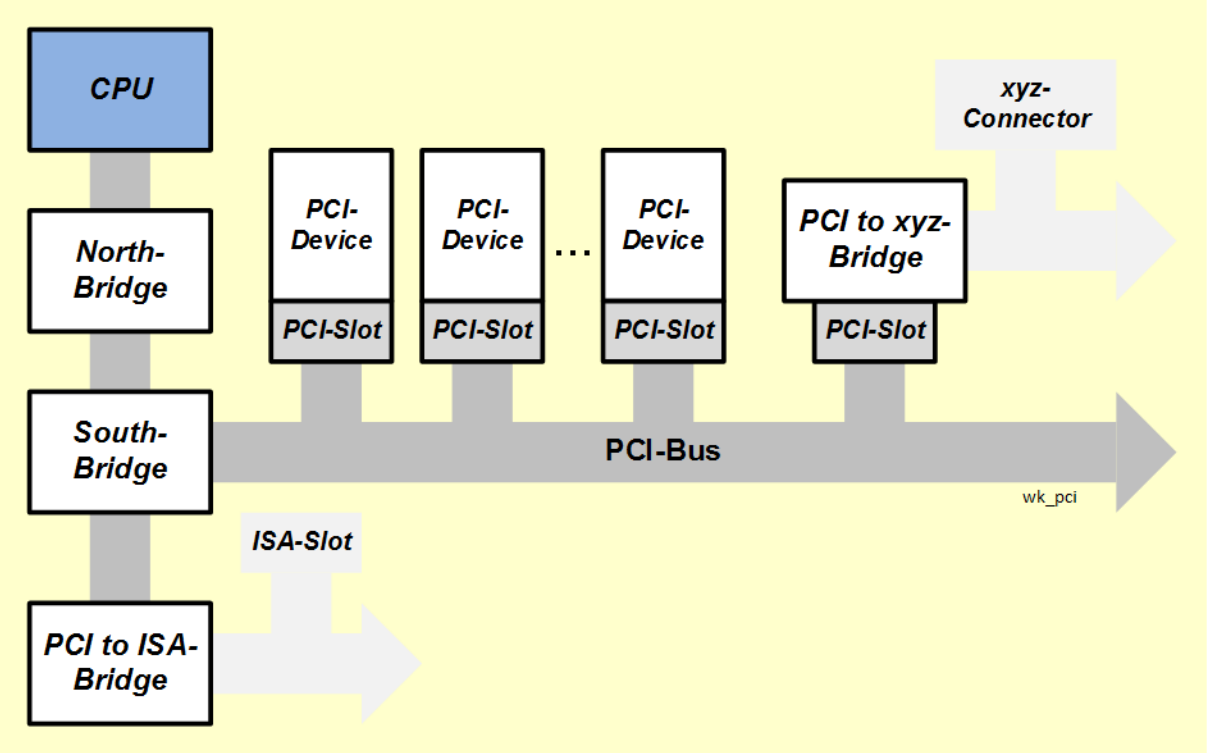


Fig. 5.9.2-1: Schematic structure of the PCI bus system

Expansion cards and devices of various manufacturers and types (network cards, sound cards, graphics cards, USB-based devices) can be plugged into the slots.

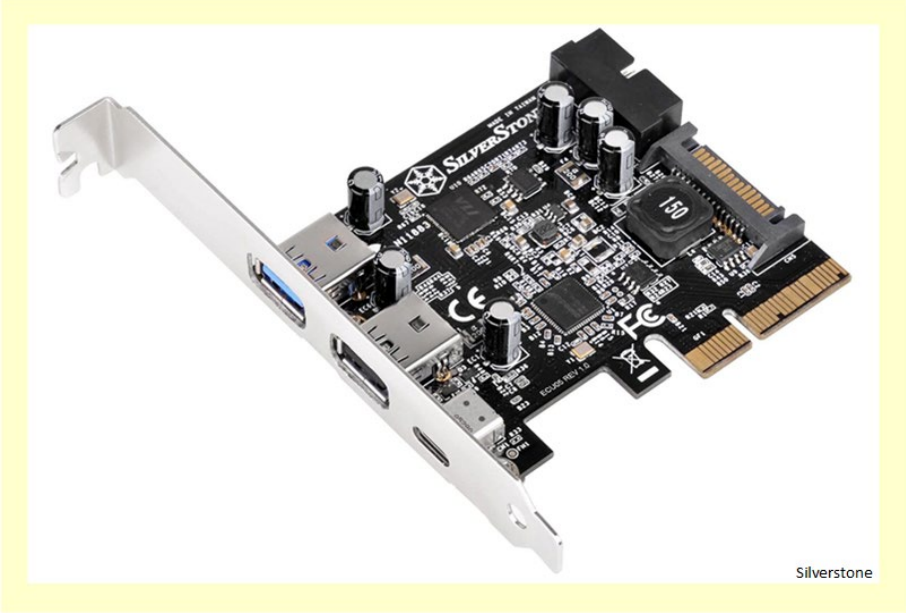


Fig. 5 9.2-2: Example of a PCI bus card (Silverstone USB 3.1 PCIe Card)

5.9.3 SATA bus system

Since the introduction of the i80486 processor in 1989, the ATA (Advanced Technology Attachment) bus system has been used to connect peripheral devices to the CPU. Originally equipped with parallel wires, the system migrated to serial data transfer technology (SATA: Serial ATA, Serial Advanced Technology Attachment) in the current millennium.

SATA is a technology for serial bidirectional data transfer between hard disks and SSDs (Solid State Discs) to mainboard components of a computer.

The table below gives an overview of the different versions:

Type	***SATA I***	***SATA II***	***SATA III***	***SATA Express***
Year of introduction	2002	2005	2009	2013
Max. velocity	1.5 Gbit/s	3 Gbit/s	6 Gbit/s	8 Gbit/s per PCI-Express-Lane
Max. data rate	150 MB/s	300 MB/s	600 MB/s	1600 MB/s
Commercial designation	*SATA 1.5* Gbit/s *SATA I SATA-150*	*SATA 3* Gbit/s *SATA II SATA-300*	*SATA 6* Gbit/s *SATA III SATA-600*	*SATA 3.2*

The graphic on the next page (Fig. 5.8.3-1) demonstrates the SATA connection of an external SSD hard disk (SSD: Solid State Disc) to a laptop.

The SSD hard disk (500 GB, 3D NAND SATA SSD, 560 MB/s read, 530 MB/s write) is installed in a SATA hard disk enclosure (6.35 cm, 2.5") with USB 3.0 port.

Control commands and data are physically transferred between laptop and hard disk in binary coding via a cable according to the USB 3.0 standard.

The solid state disk is controlled via the SSD controller including addressing and LLC layers. Both sublayers communicate with the corresponding layers of the laptop's data link layer by means of virtual connections: MAC layer to handle addressing, LLC layer for control.

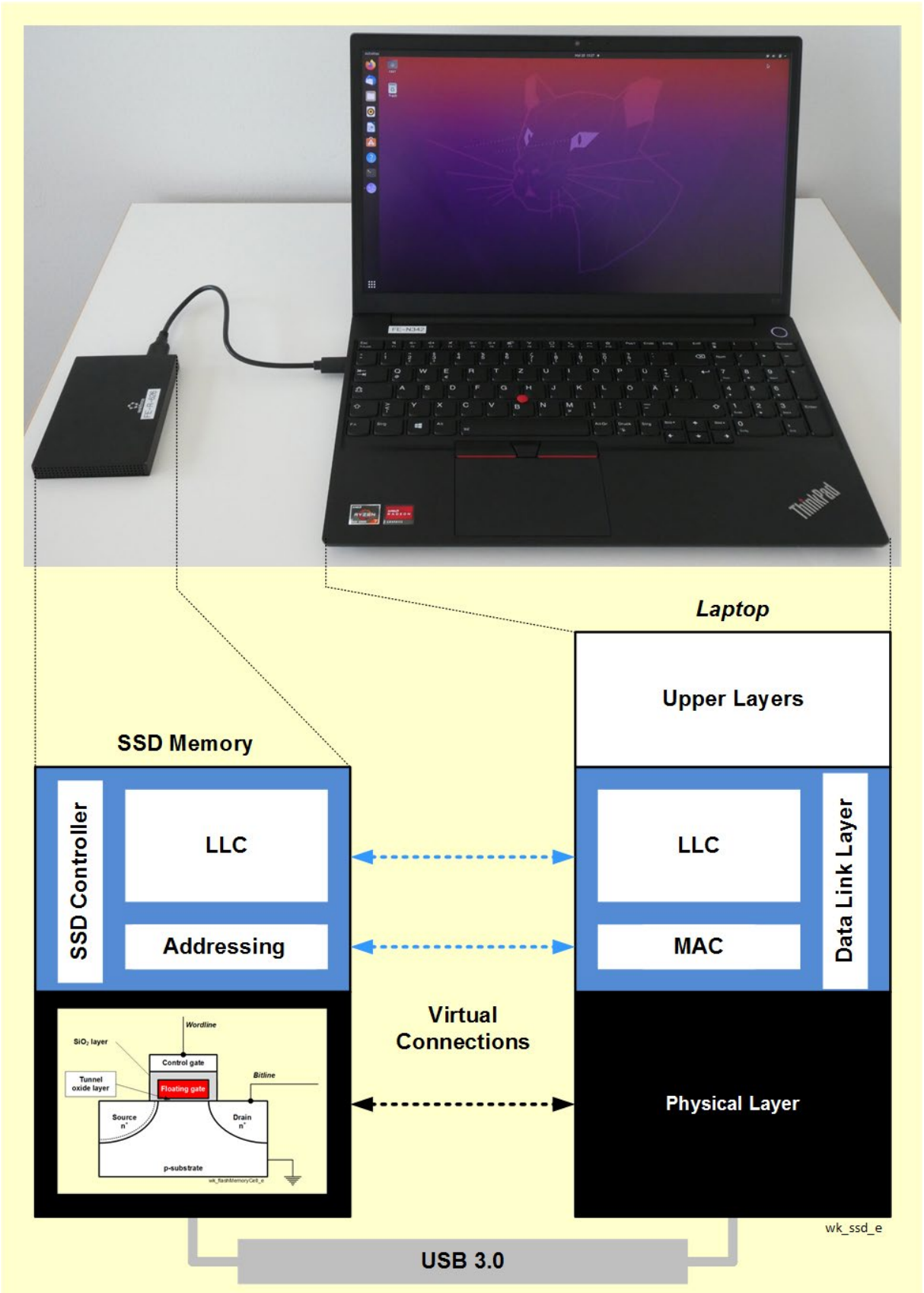


Fig. 5.9.3-1: Schematic of connecting an external SATA SSD hard disk to a laptop

5.10 Representation of data networks by graphs

5.10.1 Terminology

A **graph G** is a data structure consisting of two sets:

- a set of **vertices (nodes, points) V = V(G)**,
- a set of unordered pairs of certain nodes, the **edges E = E(G)**.

In data networks, the nodes represent the data stations, and the edges describe the network connections. The edges can be unidirectional (directed edge) or bidirectional (undirected edge). Unidirectional edges are indicated by an arrow, bidirectional edges by simple lines or double arrows.

A graph G in which all nodes are connected and in which no loops exist is called a **tree**. A subgraph T of a graph G in which all nodes are connected, which represents a tree and which contains all nodes of G, is called a **spanning tree**.

5.10.2 Representation of graphs

5.10.2.1 Introduction

Graphs can be represented in different forms. The visual method represents in graphical format the **connectivity of the network nodes via the edges**.

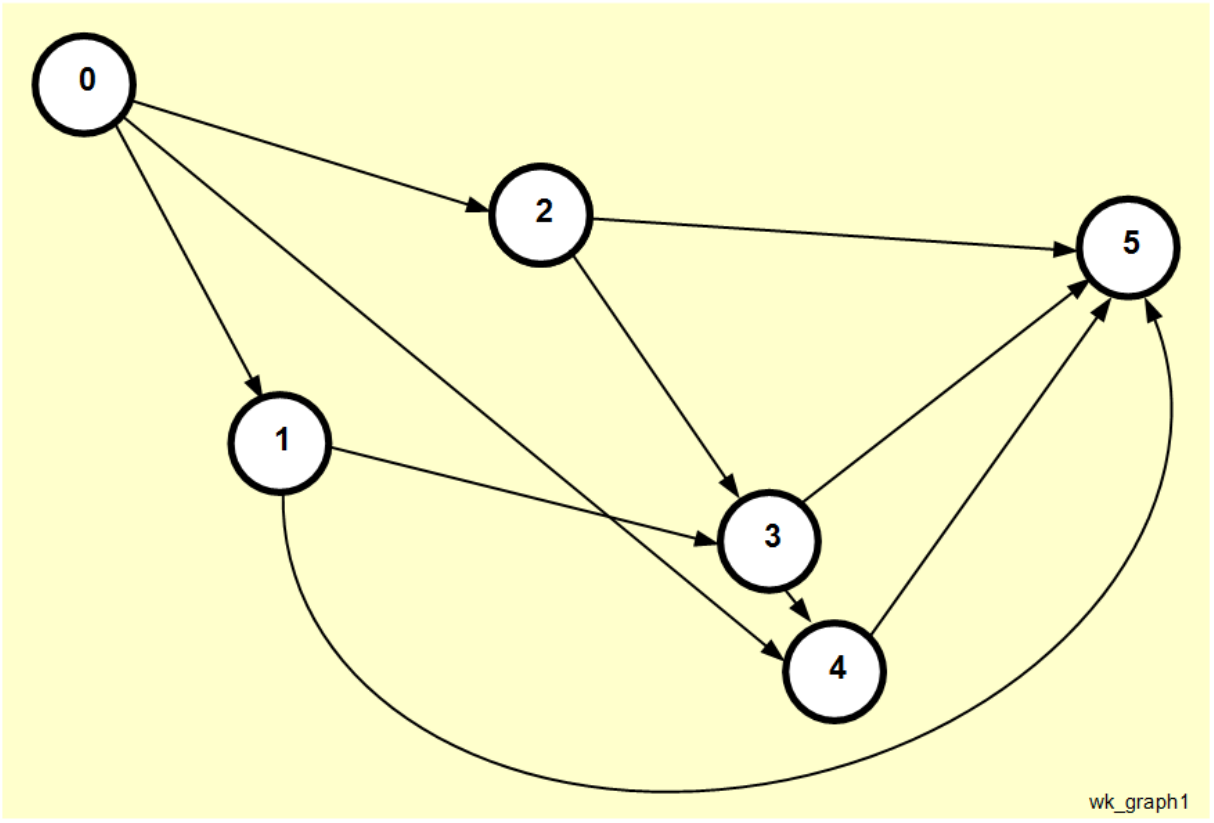


Fig. 5.10.2.1-1: Graphical representation of a network with unidirectional connections

5.10.2.2 Reduced graph

In the case where an edge between two nodes can also be replaced by another edge train within the graph that also connects the two nodes, the original edge can be omitted. The new graph is called a reduced graph. Reduced graphs lead to a reduction of complexity and to better clarity.

Exercise

E.5.10.2.2-1: Determine a reduced graph for the example network.

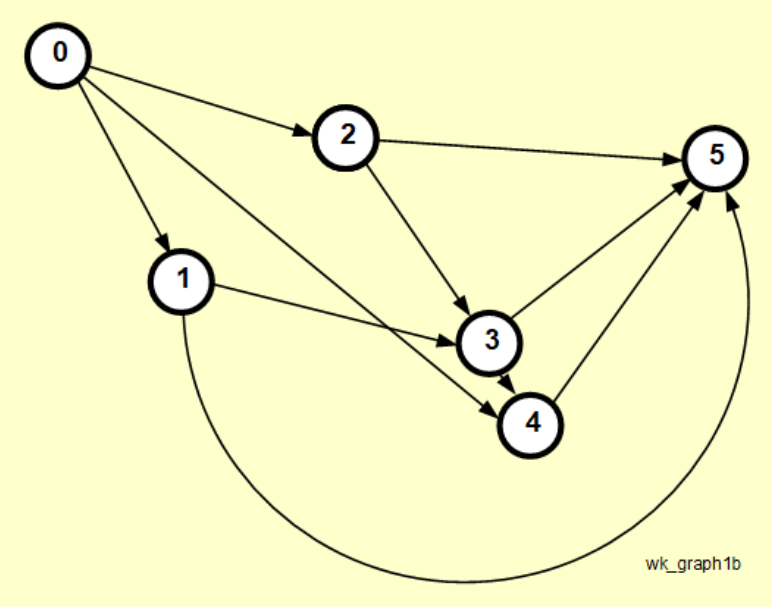


Fig. 5.10.2.2-1: Original graph of the example network Reduced graph

5.10.2.3 Adjacency matrix

An analytical representation is the two-dimensional representation in the form of the **adjacency matrix A**. Each element A_{ij} indicates whether node j is directly reachable from node i via an edge. In a network with N nodes, the following applies for
$0 \leq i, j \leq N-1$

- $A_{ij} = 1$, if node j is directly reachable from node i via an edge,
- $A_{ij} = 0$, otherwise.

A practically useful property of the adjacency matrix A is the fact that by exponentiation with exponent n a matrix A_n is obtained which represents all paths with n edges between two vertices:

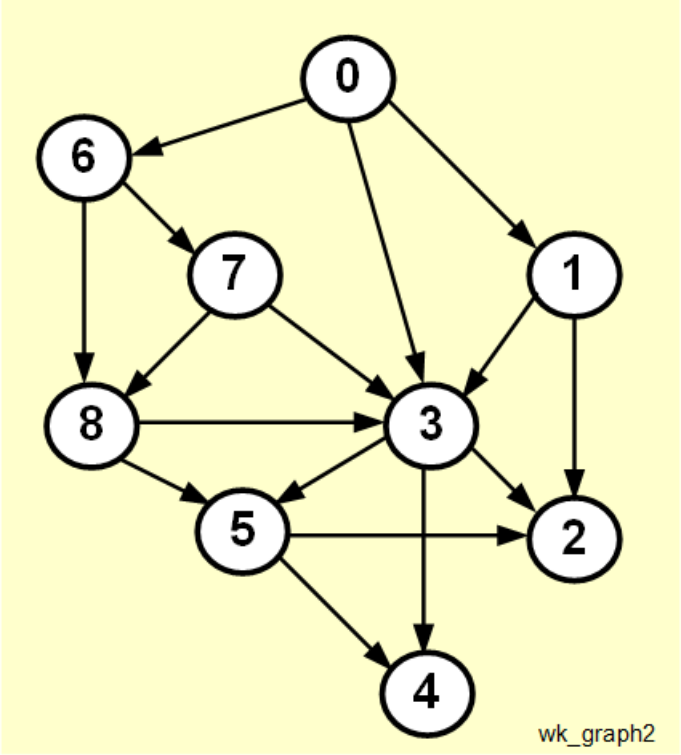
$$A_n = A^n .$$

Exercises

E.5.10.2.3-1: Develop the adjacency matrix for the example network above.

i/j	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

E.5.10.2.3-2: For the network shown below, construct the adjacency matrix and work out one possibility of a spanning tree.
(The first elements in the tree close to the root should be nodes 0 and 6).



E.5.10.2.3-3: Develop an executable computer program that multiplies a given adjacency matrix by itself and outputs on the screen all paths in the associated mesh that are composed of two edges.

5.10.2.4 Edge list

A third way of representing data networks is in the form of an **edge list**. Edge lists generally require less memory compared to the corresponding adjacency matrices.

Example for the above sample network:

```
(0, 1)    (1, 3)    (2, 3)    (3, 4)    (4, 5)
(0, 2)    (1, 5)    (2, 5)    (3, 5)
(0, 4)
```

Exercises

E.5.10.2.4-1: Write an executable computer program that creates the associated adjacency matrix from a given edge list.

E.5.10.2.4-2: Extend the program of the previous task (E.5.10.2.4-1) by a function which decides whether any two nodes in the network can communicate with each other. Display the result in matrix form on the screen.

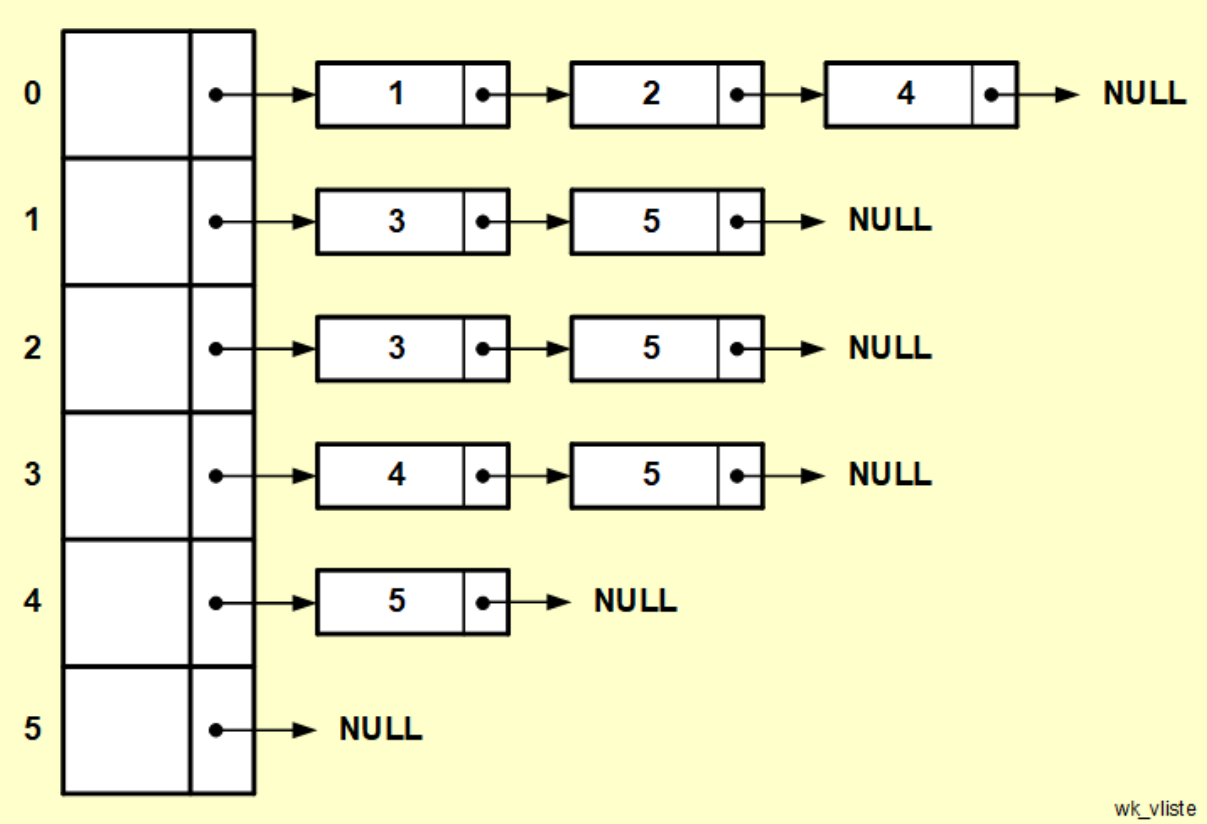
E.5.10.2.4-3: Integrate into the program from task E.5.10.2.4-1 a function that decides whether each node in the network describing the adjacency matrix is part of a loop.

5.10.2.5 Linked list

Alternatively, data networks can be represented by **linked lists**. Each node in the network has its own linked list, which contains all nodes that can be reached from this node.

The representation is relatively confusing, since first the node of interest must be searched for in the main list.

The advantage of this representation is that the graph can be created instantaneously while traversing the network. No static presets are required.



Fit. 5.10.2.5-1: Representation of the example network by the data structure of a linked list

5.10.2.6 Construction of minimum spanning trees

The construction of **minimum spanning trees** with the so-called **spanning tree algorithm** (ST algorithm, IEEE 802.1d) is often used for software-technical avoidance of loops in data networks.

Given a connected weighted graph G. All edges in G are labeled with non-negative weights. Each possible spanning tree T of G is assigned a total weight calculated from the sum of the weights of all edges. The minimum spanning tree of the graph G is the spanning tree whose total weight is as small as possible.

The weight of a minimum spanning tree is uniquely defined. The minimum spanning tree of a graph, on the other hand, is not uniquely defined. If two or more edges of the graph have the same weights, different minimum spanning trees exist.

Two different algorithms for constructing minimum spanning trees will be considered:

- **Algorithm 1:**
  Construction of a minimum spanning tree by sequential deletion of edges:

  - Given a connected weighted graph with n vertices,

  - Step 1: Order the edges of G by decreasing weights,
    (Assumption: all edges initially belong to the spanning tree).

  - Step 2: Sequential deletion of all those ordered edges which
    do not break the graph, until (n-1) edges remain,

  - Step 3: End of the algorithm.

- **Algorithm 2:**
  Construction of a minimum spanning tree by adding edges (algorithm after J. Kruskal):

  - Requirement is a connected weighted graph with n vertices,

  - Step 1: Order all edges of G with increasing weights,
    (Assumption: none of the edges initially belongs to
    the spanning tree).

  - Step 2: Sequential addition of that shortest edge,
    which does not form a cycle,

  - Step 3: End of the algorithm.

Exercises: Construction of minimum spanning trees

E.5.10.2.6-1: According to the two algorithms discussed, construct minimum spanning trees for the network shown below: