Computer Engineering & Computer Networking Summary

1 - Basic Introductions

Current state of affairs

- Computers involved in every **daily activity**
- Computers are available in **vast quantities**
- Computers are **ubiquitous**
- Everything communicates with everything
- Integrated Circuits, chips and computer engineering, communication networks shape today's and tomorrow's economies and societies
    - Impacts Politics
    - Information and Communication Technology (ICT) impacts today society

Developments in Computer Science

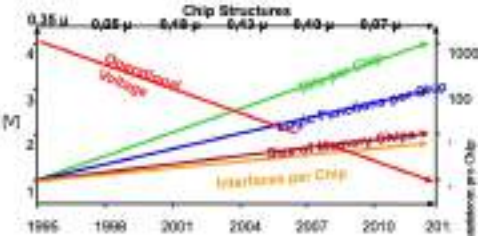
- **Abacus** - 3000 years ago in China (Addition, Subtraction, Multiplication and division, roots & squares)
- **Algorithms** - Persian Mathematician and astronomer
    - Processing Rules for humans to perform like machines
    - No room for interpretation

Steps from 1500 to 1930 go to Slides (p. 15)
1st Operational Computers (1940s) (p. 16)
Circuits, CPUs - at Scale

- 25 mn NAND Memory Chips
    - Surface Area: 167 mm^2 for 8GB
    - **Stackable** since 2010

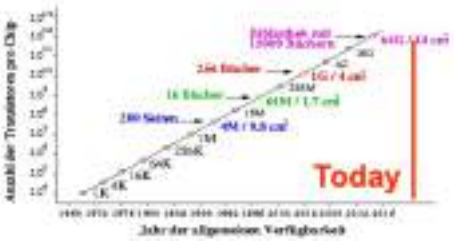
Moore's Law

- Number of transistors per (processor) chip doubles every 2 years.
- Computing power of high speed processors double every 18 months
- At the same costs microelectronincs deliver double the performance every 2 years.
- Only through cooperation of major companies can innovation thrive.



Trends

More Transistors per chip - according to SIA (American Semiconductor industry)
Enhanced Power at lower energy demand
Enhanced size of memory chips
Maybe completely new underlying technology?



Fundamental physics of new materials

- New theoretical methods
- Previous methods applied to new materials
- Design of new materials with specific functionality
- Synthesis of materials only for computer science purposes

Less energy demand on operation and during production?

Possible replacements by **Multiferroics**, (ferromagnetic, ferroelectric, ferroelastic, and/or ferrotoroidic)

2 - Arithmetics

Binary System might have originated in China

Arithmetics - handle arbitrary information and information units (Calculate, convert, save, communicate)

- Hexadecimal, Decimal - exists to compact long binary numbers
- Octal and Hexadecimal systems are easily converted into binary system

Numeral Systems

- Each separate binary position (0, 1) is called a **bit**
- Bit: smallest unit computer can handle
- Binary System is a **place value system** - each position equals the value of a power to 2

Fractional Numbers

A dot separated the integral and fractional part.

$$X_{10} = \sum_{i=-M}^{N-1} b_i \cdot B^i \quad \text{with}$$

- B = base of the number system $(B \in \mathbb{N}, B \geq 2)$
- b = numericals $\{b_i \in \mathbb{N}_0, 0 \leq b_i < B)$
- N = number of position in front of dot
- M = number of position after the dot

Euclid's Algorithm

Conversion from decimal system into a system with base b

Representation of a number:

$$Z = z_n\, 10^n + z_{n-1}\, 10^{n-1} + \ldots + z_1\, 10 + z_0 + z_{-1}\, 10^{-1} + \ldots + z_{-m}\, 10^{-m}$$
$$= y_p\, b^p + y_{p-1}\, b^{p-1} + \ldots + y_1\, b + y_0 + y_{-1}\, b^{-1} + \ldots + y_{-q}\, b^{-q}$$

Start with the highest valued position

- Step 1: Calculate **p** according to $\mathbf{b^p \leq Z < b^{p+1}}$ (define i = p)
- Step 2: Seek $\mathbf{y_i}$ and the remainder $\mathbf{R_i}$ by dividing $\mathbf{Z_i}$ by $\mathbf{b^i}$: $\mathbf{y_i = Z_i}$ **div** $\mathbf{b^i}$; $\mathbf{R_i = Z_i}$ **mod** $\mathbf{b^i}$;
- Step 3: Repeat step 2 for i = p-1, … and replace after ach step **Z** by $\mathbf{R_i}$, until $\mathbf{R_i = 0}$ or until $\mathbf{b^i}$ (the conversion error acceptable) will be small enough.

Horner Scheme
Conversion from decimal system into a number system of base b

Divide the decimal number successively by the value of base **b**.

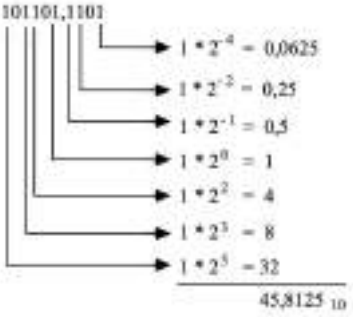
Integral remainders define the numericals of **X_b** in the order of least significant to highest significant position.

Convert 15741_{10} into the hexadecimal system:

$15741_{10} : 16 = 983$	Remainder 13	(D_{16})
$983_{10} : 16 = 61$	Remainder 7	(7_{16})
$61_{10} : 16 = 3$	Remainder 13	(D_{16})
$3_{10} : 16 = 0$	Remainder 3	(3_{16})

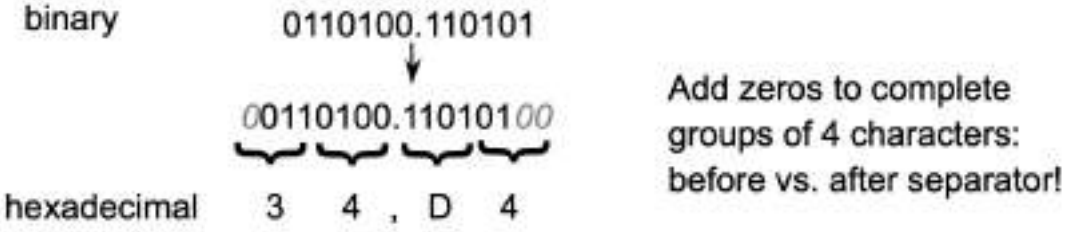
➔ **$15741_{10} = 3D7D_{16}$**

Base b -> Decimal System
Based on position multiply value times b^position



Conversion of arbitrary Place Value Systems
In case of base b being a power of another base, multiple places can be combined by a single numerical or vice versa.

$2^4 = 16 \Rightarrow$ 4 binary places $\rightarrow$ 1 hexadecimal place



Representation Options for Negative Numbers

Absolute Value Plus Algebraic Sign

One place used for algebraic sign (Vorzeichen)

MSB (Most significant bit, the leftmost) determines
MSB = 0 -> positive number
MSB = 1 -> negative number

Drawbacks:

- For additions and subtractions signs have to handled separately
- Two representation of number zero exists

Ones Complement for B-1 Complement

$$z_{ek} = (2^n - 1) - z$$

Defined as the complement of the positive number
Negate a number by negating each bit separately

Negative numbers are identified by a bit set on the first place

Example: $4 = 0100_2$ ➔ $-4 = 1011_{ek}$

$-4 = 2^4 - 1 - 4 = 11_{10} = 1011_2$

Advantages compared to 1:

- First place doesn't need to be regarded for add. & sub.
- Still 2 zero representations.

Two's Complement

$$z_{zk} = 2^n - z$$

Add to the Ones Complement the value 1
MSB defines the sign of the number

Drawback:

- Asymmetric number space, smallest negative number is in absolute value bigger than largest positive number

Example:

The number -77_{10} is supposed to be represented by 8 bit

$77_{10} = 0100\ 1101_2$

Absolute value/sign: $-77 = 1100\ 1101_2$ (Complement bit-by-bit)

Ones complement: $-77 = 1011\ 0010_2$ (Add 1)

Two's complement: $-77 = 1011\ 0011_2$

Offset Representation

- the same as using two's complement except that the most significant bit is inverted.

Used for the floating point number representation
The number space is shifted by an addition of a constant

- Such that the smallest (negative) number will be represented as 0..0
- For n places the offset is 2^(n-1)
- The number space is asymmetric
- number sequence from the most negative to the most positive is a simple binary progression, which makes it a natural for binary counters

Summary and Comparison of All Options

Representation as				
Decimal	Absolute value/sign	Ones Complement	Two's Complement	Characteristic
-4	---	---	100	000
-3	111	100	101	001
-2	110	101	110	010
-1	101	110	111	011
0	100,000	111,000	000	100
1	001	001	001	101
2	010	010	010	110
3	011	011	011	111

Fix Point Numbers

- Decimal separator position should never change
- Is placed after last bit
- Any number can be converted to this agreement by applying scale factors
- Negative Numbers are represented by a two's complements
- Not used anymore except for input output parts of hardware
- Integer defines a specific fix point format

Floating Point Numbers

Very large or very small numbers are represented as floating point numbers

- Subset of the rational numbers
- Representation in a semi logarithmic scale

$$X = \pm \text{Mantissa} \cdot b^{\text{Exponent}}$$

b is most often fixed to 2 or 16 by the system (so no need to represent in machine word)

Floating point numbers are represented often with absolute value and sign, but not as a two's complement!

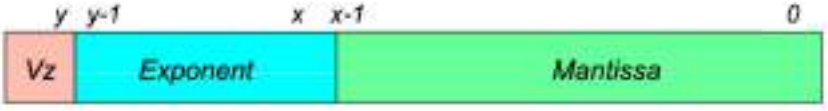
The **mantissa** defines the decimal separator at a fixed position typically agreed upon as left of the MSB.
The **exponent** is an integer with an offset

For the mantissa and the characteristic a **fixed number of memory bits** are defined within the systems architecture.

Characteristic's length y-x defines the size of the number space.

Mantissa's length x defines the precision of the representation

$$\text{Decimal number} = (-1)^{Vz} * (0,\text{Mantissa}) * b^{\text{Exponent - offset}}$$



Normalization

- Enforcing a unique representation style
- Zero will be a special sequence
- All representations start with "1.X"
- "1." will be implied by the system

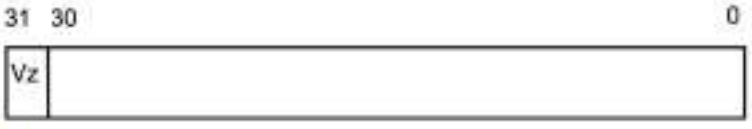
- In case of all arithmetical operations as well as conversions, this “missing” 1 have to be considered!

Examples of Representations and Number Spaces

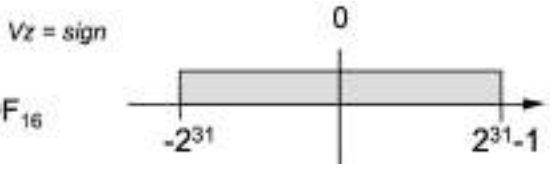
Example: 32 bit, base 2, decimal number 7135

The **amount of numbers** to be represented is for all three case identical: **2^32**
The **range** and **density** of the number spaces differ drastically!

Fix point Representation with Two’s Complement:

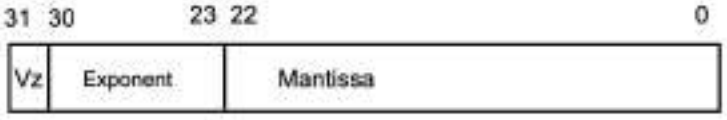


$0\ 000\ 0000\ 0000\ 0000\ 0001\ 1011\ 1101\ 1111_2 = 0000\ 1BDF_{16}$



- Numbers between -2^31 and 2^31-1

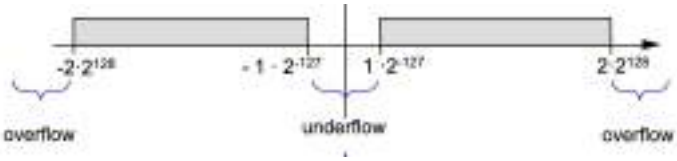
Floating Point Number, normalized:



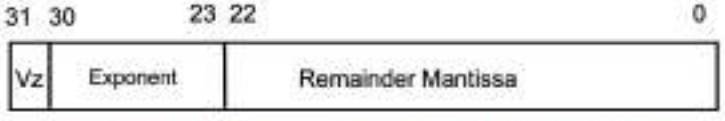
$0\ 100\ 0101\ 1\ 110\ 1111\ 0111\ 1100\ 0000\ 0000_2 = 46EF\ 7C00_{16}$

absolute value : $1 \cdot 2^{-127}$ ... $(1 + 1 - 2^{-22}) \cdot 2^{128} \approx 2 \cdot 2^{128}$

A zero cannot be represented!



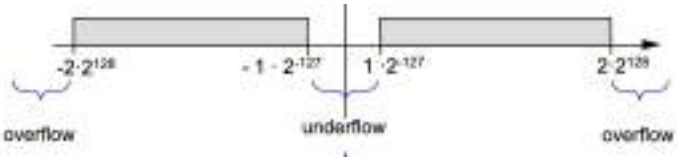
Floating Point Number, normalized, first "1" hold implicitly:



$0\ 100\ 0101\ 1\ 101\ 1110\ 1111\ 1000\ 0000\ 0000_2 = 46DE\ F800_{16}$

positive numbers: $1 \cdot 2^{-127}$ ... $(1 + 1 - 2^{-23}) \cdot 2^{128} \approx 2 \cdot 2^{128}$

A zero cannot be represented!


-2·2^128
-1·2^-127
1·2^-127
2·2^128
overflow
underflow
overflow

Characteristic Numbers
To be able to compare different floating point number representations 3 **characteristic** numbers are defined:

- **maxreal** the largest possible and representable normalized positive number
- **minreal** the smallest possible and representable normalized positive number
- **smallreal** the smallest number, which can be added to 1, to reach a representable number with a value different than 1.

Inaccuracy
The difference between two subsequent numbers in floating point representation increases exponentially with the size of these numbers

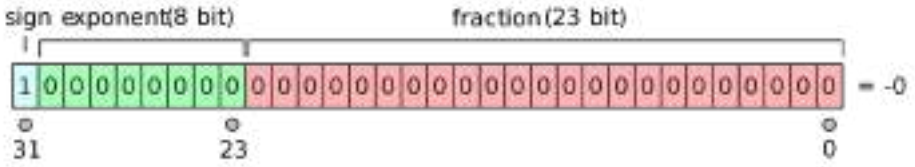
For fix point representation the difference remains constant

Not all rules for real numbers apply for floating point numbers (f.e. Associative law might not apply)

Standardization: IEEE-Standard
Defines following representations:

- IEEE single: 32 bit
- IEEE double: 64 bit
- IEEE extended: 80 bit

Representation of 0:



Addition of Binary Numbers

- Ensure negative numbers are represented in absolute value sign
- We reserve the first bit for sign
- Then we complement the representation so we have no problem with subtractions
- Every subtraction a-b can be represented as a + -b

If 2 Two's Complement numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs if and only if the result has the opposite sign. Overflow never occurs when adding operands with different signs.

i.e. Adding two positive numbers must give a positive result
Adding two negative numbers must give a negative result

Overflow occurs if

- (+A) + (+B) = −C
- (−A) + (−B) = +C

Multiplication

Two's complement advantage doesn't exist - need to convert back to absolute value and sign
Integer division and multiplication is repeated addition

If multiplier and divisor are powers of 2 then we can shift the binary numbers left and right to easily get our result.

- Numbers represented as absolute value/sign numbers are just multiplied.
  - Absolute values of those numbers are multiplied as if they were postive numbers.
  - The sign of the results will be calculate as an antivalence (XOR) of the two signs of both multipliers: sign (X•Y) = sign (X) XOR sign (Y)
- A multiplication floating point numbers means to multiply the mantissas and adding their exponents:

$$m_1\, b^{e1} \cdot m_2\, b^{e2} = (m_1 \cdot m_2)\, b^{e1+e2}$$

- If the mantissa is represented in the absolute value/sign representations, the regular multiplication method does apply
  - The result may be required to be normalized!
  - For the addition of the characteristics $c_1 = e_1 + o$ and $c_2 = e_2 + o$ the sum needs to be corrected to the right offset **o** to reach the correct result'scharacteristic $c = (e_1 + e_2) + o$.

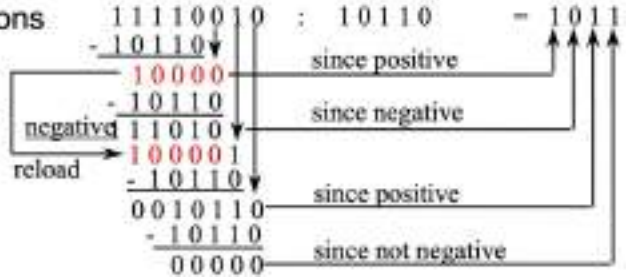
Division

Follows Multiplication Rules
Binary Division only knows 0 and 1 as outcome

- 0 if Divisor > current Dividend
- 1 if Divisor < current Dividend

- A computer-based recognition shows 3 alternatives:
  - 1. Comparator to compare divisor with current dividend
  - 2. Subtraction: In case of a negative result the old dividend's value is reloaded
  - 3. Subtraction: In case of a negative result, the divisor is added again (readdition)
- Shortenend division: Readdition and subtraction of a right shifted divisor (by 1 place) combined.

  + Divisor - ½ Divisor = + ½ Divisor
- Division by subtractions
  - Direct subtraction of divisors
  - Addition of the divisor's two's complement



- Can't divide by 0
- Periodic Fractions should be detected

Circuits for multiplications can be used for divisions if minor modifications are performed:

- Left-shift of dividend (instead of right-shift in case of multiplication)
- Subtraction Divisor (instead of addition of multiplier)

ASCII Code
American Standard for Coded Information Interchange

- Lower uppercase latin alphabet, arabic numericals and few special characters
- Coded in 8 bit characters - 256 different characters
- First bit is unused - 128 useable characters
- Some Extensions use the first bit to have 256 characters

End of strings can be put at the start or by a termination character at the end

Numerical as a Number matches the Number in Binary: 4 -> 00000100

Unicode is used for all known systems (2 byte characters)

UTF - Unicode Transformation Format

BCD Code
BCD codes binary numericals For each decimal numerical 4/8 bits used
Used for Calc in decimal system, storage of decimal numbers, LCD display controls

Decimal Number	Binary Number	BCD Representation
294	100100110	0010.1001.0100 2 9 4
16289	11111110100001	0001.0110.0010.1000.1001 1 6 2 8 9

Binary Scales
Vendors MB != MByte

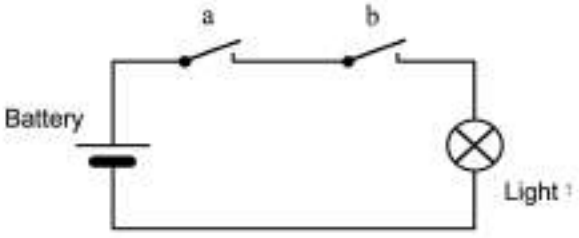
3 - Combinational Circuits

- Purely combinatorial, logic circuits
- No storage!
- Logical Functions

Boolean Algebra	Instantiation
V	{0,1}
⊕	∨
⊗	∧
n	0
e	1
$\bar{a}$	$\bar{a}$

Huntington Axioms:

H1: Commutative Law	$a \vee b = b \vee a$ $a \wedge b = b \wedge a$
H2: Distributive Law	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
H3: One Element	$a \wedge 1 = a$ $a \vee \bar{a} = 1$
H4: Zero Element	$a \wedge \bar{a} = 0$ $a \vee 0 = a$



A boolean expression by itself does not represent a Boolean value, since it may contain Boolean variables!

- Only an assignment of binary value with a Boolean value allows for the calculation of this Boolean expressions's Boolean truth value

Boolean Functions
Assuma a tuple of binary variables (x1,x2,...)

A (n-ary) Boolean Function assign to every possible Boolean value assignment of this Boolean variable exactly one Boolean value:

There exists 2^n assignments and 2^2^n functions
(foreach argument of a Boolean function two different function values exist)

16 Possible 2-ary Boolean Functions

x_1 0011 x_0 0101	Verbal Form	Symbolic Representation	Name
f_0 0000	constant 0	0	Contradiction, symbol: ⊥ (unfulfillable)
f_1 0001	x_1 und x_0	$x_1 \wedge x_0$	Conjunction
f_2 0010	not x_0, but x_1	$x_1 \wedge \bar{x}_0$	Inhibition
f_3 0011	identical x_1	x_1	Identity
f_4 0100	not x_1, but x_0	$\bar{x}_1 \wedge x_0$	Inhibition
f_5 0101	identical x_0	x_0	Identity
f_6 0110	x_1 unequal x_0	$x_1 \nleftrightarrow x_0$	Antivalence, XOR
f_7 0111	x_1 or x_0	$x_1 \vee x_0$	Disjunction
f_8 1000	not (x_1 or x_0)	$x_1 \bar{\vee} x_0$	NOR function, Peircean Arrow
f_9 1001	x_1 equal x_0	$x_1 \leftrightarrow x_0$	Equivalence
f_{10} 1010	not x_0	$\bar{x}_0$	Negation
f_{11} 1011	if x_0, then x_1	$x_0 \rightarrow x_1$	Implication
f_{12} 1100	not x_1	$\bar{x}_1$	Negation
f_{13} 1101	if x_1, then x_0	$x_1 \rightarrow x_0$	Implication
f_{14} 1110	not (x_1 und x_0)	$x_1 \bar{\wedge} x_0$	NAND function, Sheffer's Line
f_{15} 1111	constant 1	1	Tautology, symbol: T (generally valid)

Full Operators System - a system with which all Boolean functions can be represented (And, Or, Not)

Operators / Systems		Representation of ...		
		Negation	Conjunction	Disjunction
$(\wedge, \vee, \neg)$		$\overline{a}$	$a \wedge b$	$a \vee b$
$(\wedge, \neg)$		$\overline{a}$	$a \wedge b$	$\overline{\overline{a} \wedge \overline{b}}$
$(\vee, \neg)$		$\overline{a}$	$\overline{\overline{a} \vee \overline{b}}$	$a \vee b$
$(\overline{\wedge})$	NAND	$a \overline{\wedge} a$	$(a \overline{\wedge} b) \overline{\wedge} (a \overline{\wedge} b)$	$(a \overline{\wedge} a) \overline{\wedge} (b \overline{\wedge} b)$
$(\overline{\vee})$	NOR	$a \overline{\vee} a$	$(a \overline{\vee} a) \overline{\vee} (b \overline{\vee} b)$	$(a \overline{\vee} b) \overline{\vee} (a \overline{\vee} b)$
$(\wedge, \not\leftrightarrow)$		$a \not\leftrightarrow 1$	$a \wedge b$	$a \not\leftrightarrow b \not\leftrightarrow (a \wedge b)$

Normal Forms
A standardized representation of Boolean functions within a full operator‘s system such as (AND, OR, NOT) is defined by the **Conjunctive Normal Form (CNF)** – sometimes called “clausal normal form” – and the **Disjunctive Normal Form (DNF)**.

A **Literal L_i** is defined as either a positive variable xi or its negation ~xi (negative variable)

- Only positive and negative literals exist
- Placeholder for a boolean value basically

Literals & Clauses
A **conjunctive clause** (or product term) K(x1,...,xm) is defined as the conjunction of literals. Combined by **AND** operators

Every conjunctive clause may be represented in a form such that a single variable x only appears at most once in one literal.

Multiple appearances of x can be reduced due to the Idempotence Law (x & x = x).

Implicant and Minterm

A conjunctive clause $K(x_1,...,x_n)$ is termed implicant of a Boolean function $y(x_1,...,x_n)$, if $K \rightarrow y$

- K implies y (K is an implicant of y), if y also takes the value 1 whenever K equals 1
- *I.e.*, for every assignment $B \in \{0,1\}^n$ the following holds true:
  If $K(B) = 1$, $y(B) = 1$ holds as well.

Definition:

An implicant of a Boolean function $y(x_1,...,x_n)$ is termed minterm, if a literal of each variable x_i within the function y of implicants exists exactly once.

Disjunctive Normal Form
Assuming a Boolean function y(x1,...,xn), a Boolean expression is termed to be the **Disjunctive Normal Form (DNF)** of the function y, if it exists as a disjunction of all **minterms** Ki. Combined by **OR**

Implicat & Maxform

Definition:
A disjunctive clause $D(x_1,...,x_n)$ is termed implical of a Boolean function $y(x_1,...,x_n)$, if $D \rightarrow y$

- D implies y (D is an implicant of y), if $\bar{y}$ also takes the value 0 whenever D equals 0
- I.e., for every assignment $B \in \{0,1\}^n$ the following holds true:
  If D(B) = 0, y(B) = 0 holds as well.

Definition:
An implicat of a Boolean function $y(x_1,...,x_n)$ is termed maxterm, if a literal of each variable x_i within the function y of implicants exists exactly once.

Conjunctive Normal Form

Assuming a Boolean function y(x1,..., xm), a Boolean expression is termed to be **Conjunctive Normal Form (CNF)** of the function y, if it exists as a conjunction of all **maxterms** Di. Combined by **AND**

DNF and CNF

Any function with n variables **may** show up to **2^n** minterms and maxterms, respectively.

	Minterm	Maxterm
0	$\bar{a}\ \bar{b}\ \bar{c}$	$a \vee b \vee c$
1	$a\ \bar{b}\ \bar{c}$	$\bar{a} \vee b \vee c$
2	$\bar{a}\ b\ \bar{c}$	$a \vee \bar{b} \vee c$
3	$a\ b\ \bar{c}$	$\bar{a} \vee \bar{b} \vee c$
4	$\bar{a}\ \bar{b}\ c$	$a \vee b \vee \bar{c}$
5	$a\ \bar{b}\ c$	$\bar{a} \vee b \vee \bar{c}$
6	$\bar{a}\ b\ c$	$a \vee \bar{b} \vee \bar{c}$
7	$a\ b\ c$	$\bar{a} \vee \bar{b} \vee \bar{c}$

A Boolean function is uniquely specified by listing either all minterms or all maxterms, respectively.

c b a	y	Minterms	Maxterms
0 0 0	1	$\bar{a}\ \bar{b}\ \bar{c}$	
0 0 1	0		$\bar{a} \vee b \vee c$
0 1 0	0		$a \vee \bar{b} \vee c$
0 1 1	1	$a\ b\ \bar{c}$	
1 0 0	1	$\bar{a}\ \bar{b}\ c$	
1 0 1	0		$\bar{a} \vee b \vee \bar{c}$
1 1 0	0		$a \vee \bar{b} \vee \bar{c}$
1 1 1	1	$a\ b\ c$	

DNF: $y = (\bar{a}\ \bar{b}\ \bar{c}) \vee (a\ b\ \bar{c}) \vee (\bar{a}\ \bar{b}\ c) \vee (a\ b\ c)$

CNF: $y = (\bar{a} \vee b \vee c)(a \vee \bar{b} \vee c)(\bar{a} \vee b \vee \bar{c})(a \vee \bar{b} \vee \bar{c})$

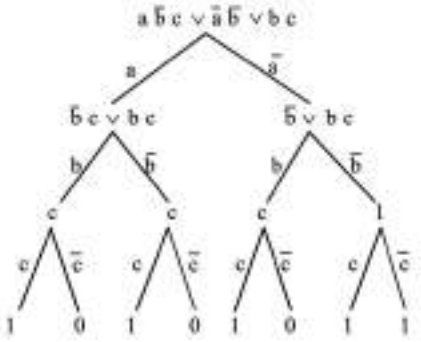
Deriving DNF or CNF from Function Tables

DNF:
The functions table contains all minterms of that function: For all rows, which show a function's value of 1, all input variables are concatenated with $\wedge$, while all of them with the value 0 are negated.
- The disjunction of all minterms leads to a Boolean expression in DNF.

CNF:
The functions table contains all maxterms of that function: For all rows, which show a function's value of 0, all input variables are concatenated with $\vee$, while all of them with the value 1 are negated.
- The disjunction of all minterms leads to a Boolean expression in CNF.

Disjunctive and conjunctive normal forms determine unique representations!

Shannon's Expansion



After the function had been expanded to all variables, minterms are identified by those leaves of the tree and their paths from the root, which show a value of 1.

Deriving Minimized Forms

- There is no optimal or unique algorithm of finding the normal form

Step 1: Derive the set of implicants and implicats, respectively, for a given Boolean function f, while ensuring that the number of literals is kept as small as possible.

Step 2: Select From These Sets of implicants and implicats, respectively, the smallest possible number of implicants and implicats, respectively, such that their disjunction and conjunction, respectively, determine this function f.

NAND NOR Conversion

Are good because they are very simple to implement in hardware.

If I receive a function f that is in disjunctive form (ORs):

- Double negation of entire f
- Application of DeMorgan law

Resulting In An Expression,which only contains NAND operators

Hierarchy of Combinatorial Circuits

Register transfer level: logic circuits as elements
- Multiplexer, shift registers, algebraic adders, …

Gate level: logic gates
- And, Or, Nand, …

Electronic switch level: Transistors
- Electrical engineering basics

Layout Level: Transistor technologies.
- Electrical Engineering basics

Gate Level
Full abstraction from technical implementation - Full Focus on logical behaviour

Design of Combinatorial Circuits
Design Limitations:
- Gates generate heat that needs to be diverted
- Gates need space (square micrometers)
- Gates have delay effects

Technical Criteria:
- Power consumption, switching time, space, material, durability
- Correct implementation with respect to static, dynamics (hazards)
- Constraints: #input lines, #output lines, maximal power of inputs/outputs

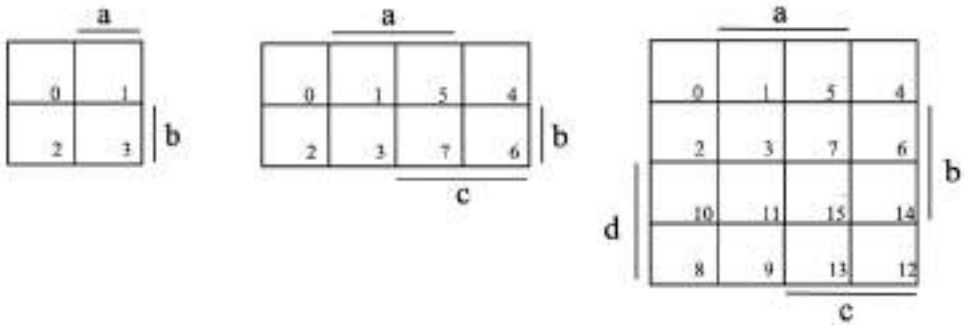
Economic Criteria:
- Design (salaries, simulations, databases)
- Implementation (costs per gate, costs per square micrometer)
- Operation (costs for tests, installation, maintenance at operation)

Criterias may contradict:
- Enhancing reliability -> Increased costs (redundancy, space)
- Lower design costs -> Increased design costs (handling design errors …)
- Higher density of functionality (less chips) -> Increased design costs

Find solutions that fulfill mandatory technical criterias at the lowest cost possible

Minimization
Find a representation of a function that minimizes the cost of the implementation, is the **minimal form** which you get through **minimization.**
Minimization can be done, algebraically, graphically, tale-based approach (1 & 2 only work for 5-6 var. max)

Graphical Approach - **KV Diagram** (Rectangle with right half belonging to a variable and left side belonging to the negation of that variable)



Field 11 is **a b ~c d**

The index with every field defines the index of this field's assignment of variables, while all variables are marked in alphabetically reverse order

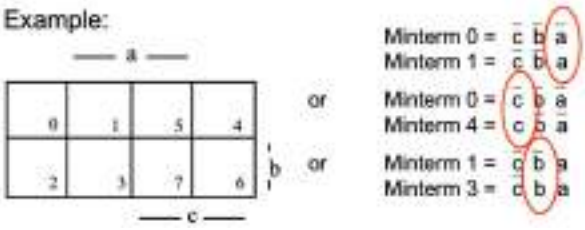
Field 11 = **1011** (binary) = **d ~c b a**

Function Table to KV Diagram - each row of function table corresponds to a single field

- Input all input variables in alphabetically reverse order into the table
- Now KV Diagram can be according to the index

Characteristics of KV Diagram

Minterms are located symmetrically to any center line / fold and only differ in a single value



Boolean terms differing in a single variable can be minimized.

- Minterms located symmetrically to a centerline can be combined into a simpler term

By knowing the minterms you can fill in ones and the rest must be zeros.

Prime Implicant

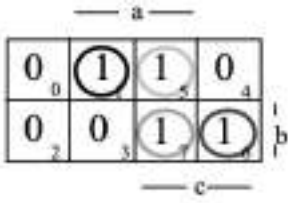
With a prime implicant we have a way of representing in a boolean algebraic expression the most number of minterms in a much simpler form.

An implicant x is a prime implicant if there is on other implicant that equals the implicant x

Every function can be represented as a disjunction of its prime implicants.

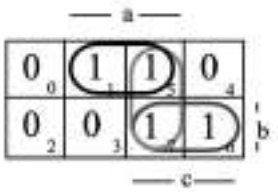
(Manually select the largest block of ones in KV Diagram)

$$f = \bar{a}\, b\, c \vee a\, c \vee a\, \bar{b}\, \bar{c}$$



4 minterms:
(a $\bar{b}$ $\bar{c}$, a $\bar{b}$ c, a b c, $\bar{a}$ b c)

3 prime implicants:
first order implicants
(a $\bar{b}$, a c, b c)



But (a $\bar{b}$, b c) would be sufficient!

$f = a\bar{b}\bar{c} \vee a\bar{b}c \vee abc \vee \bar{a}bc$
$= a\bar{b} \vee ac \vee bc = a\bar{b} \vee bc$

You could display a function as a series of minterm indices.

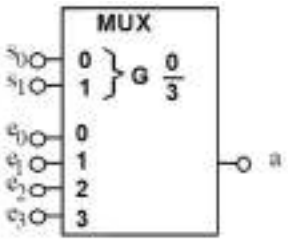
Alternatives for Function's Implementations

Are important because we would like when a function reappears to just instantiate that instead of rebuilding

- Multiplexer/Demultiplexer
- Memory Circuits

Multiplexer (MUX)

- Multiple Inputs n
- One output
- MUX are classified by their size a 2^n Multiplexer
- From the n control lines one of the 2^n inputs is selected and switched to be the output
- Control Dataflows
- A 2^n multiplexer can implement a boolean function with n+1 variables



s_1	s_0	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3

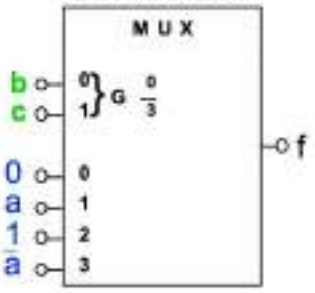
Implementing function f with a multiplexer:

$f = \bar{a}c \vee \bar{b}c \vee ab\bar{c}$

Implementation table in case of the selection of *b* and *c* as control inputs:

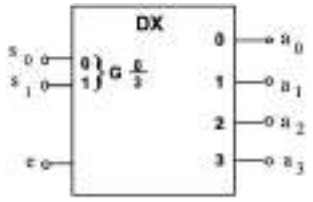
c b	00	01	10	11
a = 0	0	0	1	1
a = 1	0	1	1	0
f =	0	a	1	$\bar{a}$

Implementation



Demultiplexer/Decoders

Is a circuit that maps single input of n control lines to 2^n outputs.

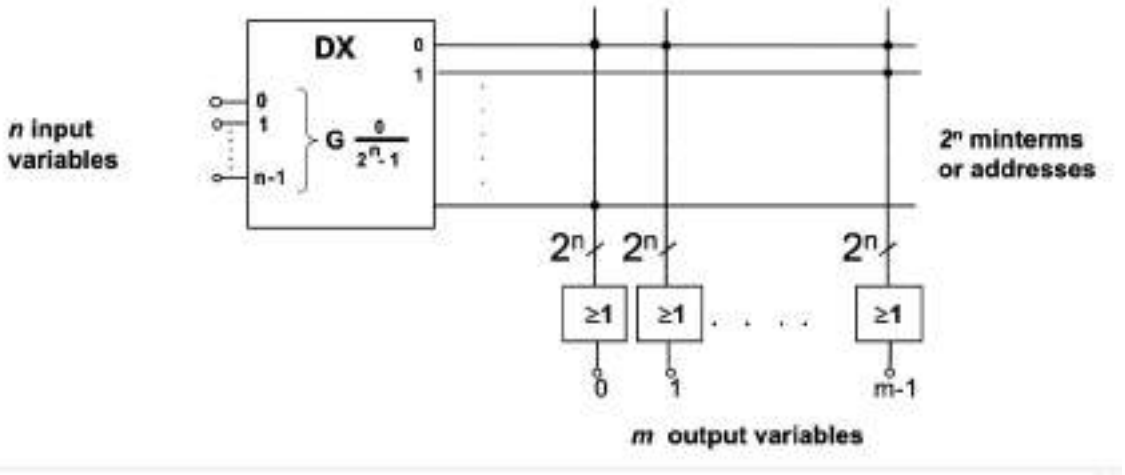


s_1	s_0	a_0	a_1	a_2	a_3
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

Memory-based Function's Implementation
Used if the function is not known before opposed to gates and multiplexers, decoders. You need higher flexibility for programming.

Scheme of Memory Cell

- Piece of hardware that is able to store information
- Big Decoder
- Imagine a big Table and based on your n inputs (control signals) that determine which memory cell you like to select as your output.



The input configuration determines the selector of a memory cell (addressing) and the value stored at this address will be offered at the output configuration.

The output configuration of the decoder determines the minterms of the n input configurations (the rows of the function table)

Memory Types
Rom (Read only Memory)

- Can only be read during operation
- Programming only done once by the vendor
- Storage Content will remain unchanged forever

PROM (Programmable Read Only Memory)

- Programmable by the user
- Happens by melting of fusible links
- Stored charge based on dedicated physical processes, that maintain that charge for years

EPROM (Erasable Programmable Read Only Memory)

- can be erased by applying UV Light
- can be reprogrammed
- Small Quartz Window provides access
- Electrical Erasing with high voltage - EEPROM: Electrically Erasable PROM, EAROM: Electrically Alterable ROM
    - Used fro Compact Flash, Memory Stick

RAM (Random Access Memory)

- Read and write
- Lose content if on power
- Dynamic Ram and Static RAM - DRAM & SRAM

DRAM

- Capacitor is used as a charger, a transistor is used to attach the control
- Memory needs to be refreshed regularly

SRAM

- Works without Power Supply
- Shorter Access time than DRAM
- Needs 6-8 transistors for flip flop, DRAM needs 1-2

Comparison

- RW: *read/write*
- Access Times (AT) for R/W
- Maintaining memory's content (operations OP) without power supply
- Memory size (MS)

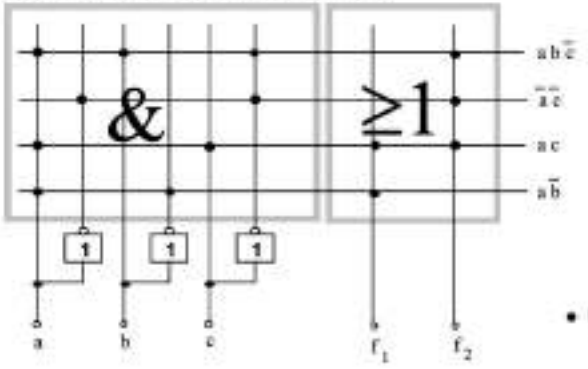
	ROM	PROM	EPROM	Flash-PROM	DRAM	SRAM
RW	R	R	R ((W))	R (W)	RW	RW
AT	+	+	+ / −	+ / −	++	+++
OP	+	+	+	+	−	−
MS	+	+	+	+	+	+

Programmable Logic Array (PLA)

- Personalized during production
- Input configuration prime implicants for covering the function, instead of minterm
- The decoder used up to now will be replaced by an AND matrix
- Field Programmable Logic Array (FPLA):
  - Fixed number of input configurations n
  - Product terms k
  - Output configurations m
- The role of PALs and FPLAs is determined by the demand for "similar" functions to be available in many devices, which may differ in their specific instantiations: reduction of costs, but no need to be fully flexible (programmable), but with fast operation

$f_1 = a\bar{b} \vee ac$ and $f_2 = \bar{a}\bar{c} \vee ac \vee ab\bar{c}$

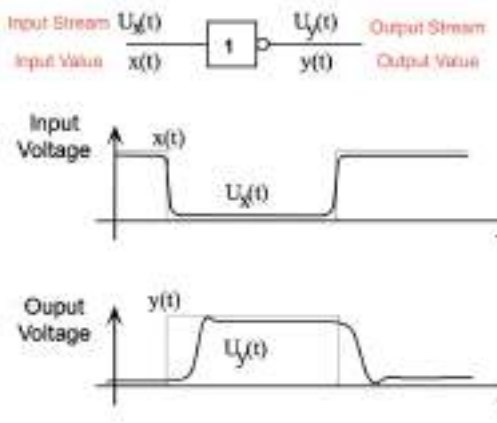
Their PAL implementation reads as follows:



• Connector

Delay Effects

Gates are implemented by transistors, resistors, and capacitors on the circuit's layout level.

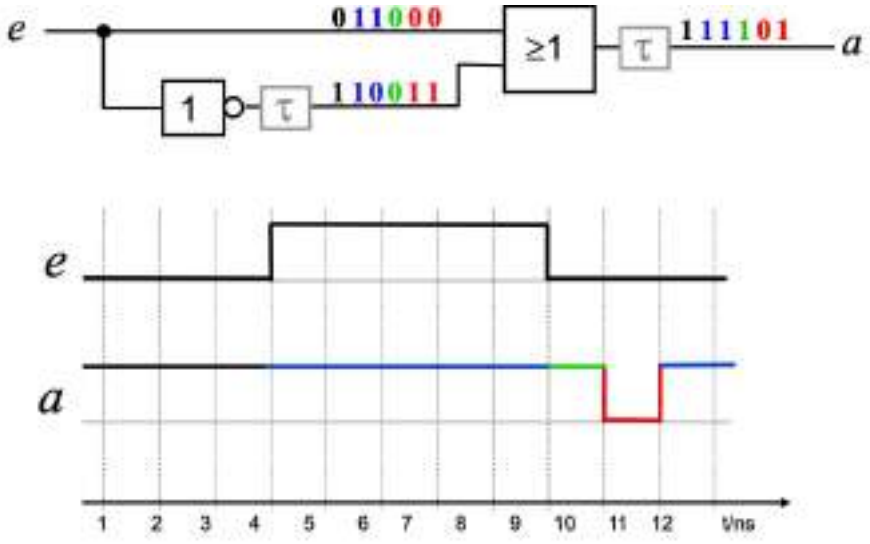


Delay Model

A real gate is modeled by:

- An ideal gate without any delays
- A delay Element with a defined delay
- Still to idealistic for real systems

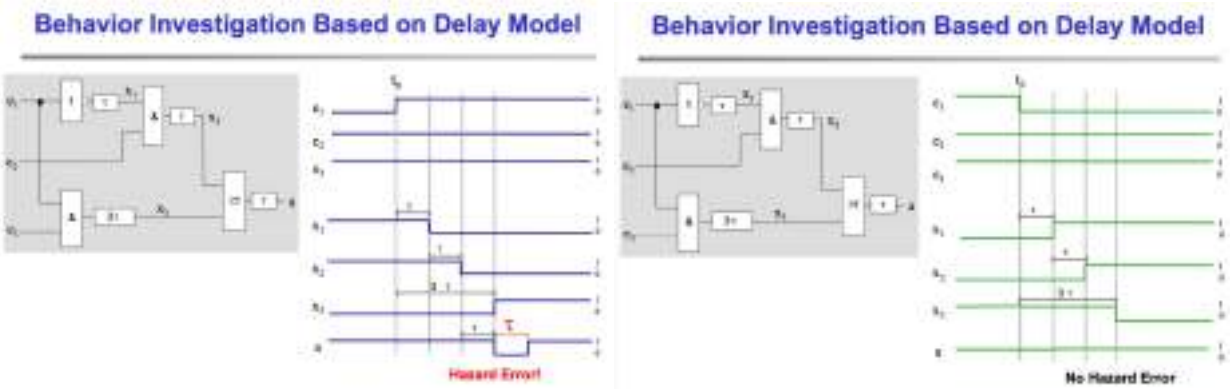
$$a = e \vee \overline{e} = 1$$



Multiple changes of the output configuration, and thus the output signal curve, may occur until a stable final value is reached.

Hazards

$$a = \overline{e}_1 e_2 \vee e_1 e_3$$



A **hazard error** is defined as multiple changes of an output configuration within a single transition
A **hazard** is defined as the logical-structural prerequisite for a hazard error, while no detailed delay values for gates implementing the function are considered
A **hazard-afflicted transition** is determined by

- Logical Function This Combinatorial Circuit Implements
- The Structure Of The Same In Terms of #gates and order of those (no delays)

Hazard Error -> Hazard

Hazard & unfavorable delay values -> Hazard error

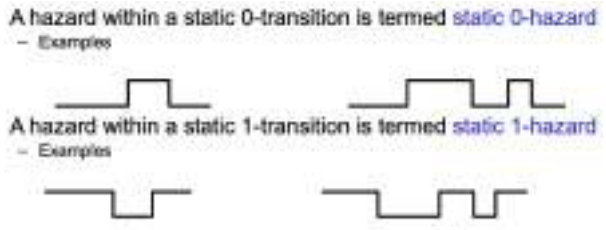
A functional hazard is due to the function itself

- Selection of deploy elements can prevent the functional hazard to occur but not the hazard itself

A structural hazard is due to the implementation of the circuit

- Can be prevented by developing a different circuit structure
- Select a different disjunctive/conjunctive normal form for the function

Static 0-Hazard and 1-Hazard



Static-1 Hazard: If the output is currently at logic state 1 and after the input changes its state, the output momentarily changes to 0 before settling on 1, then it is a Static-1 hazard.
Static-0 Hazard: If the output is currently at logic state 0 and after the input changes its state, the output momentarily changes to 1 before settling on 0, then it is a Static-0 hazard.

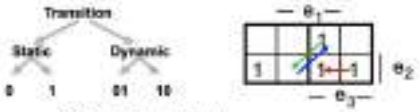
Dynamic 01-Hazard and 10-Hazard

A hazard within a dynamic 01-transition is termed dynamic 01-hazard

– Examples

A hazard within a dynamic 10-transition is termed dynamic 10-hazard

– Examples



Static 1-transition:

Transition (e_3,e_2,e_1): (1,1,0) → (1,1,1)

Dynamic 01-transition:

Transition (e_3,e_2,e_1): (0,1,1) → (1,0,1)

Dynamic 10-transition

Transition in inverse direction: (1,0,1) → (0,1,1)

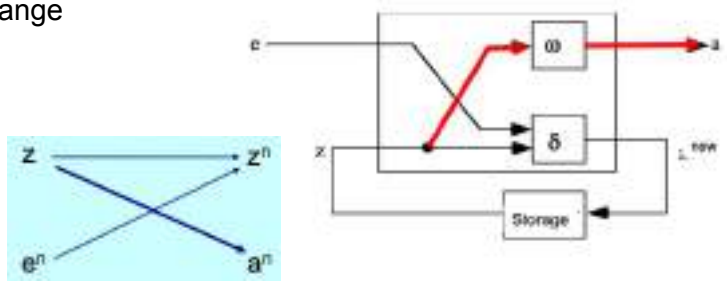
4 - Sequential Circuits

The output configuration depends also on values of input variables of past points in time!

Moore FSM

- A new state zNew is created
- The output is independent of the input
- The output assignment can only change after a state change
- **Synchronous and asynchronous sequential circuits**

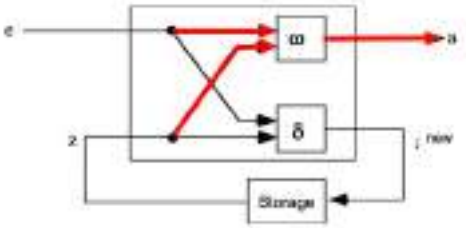
$a = \omega(z)$ $\quad a \in A;$ $\quad z^{new} = \delta(z,e)$ $\quad e \in E;$ $\quad z, z^{new} \in Z$

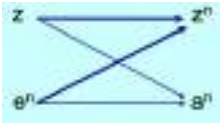


Mealy FSM

- A new state zNew is created but we use e for the output as well
- Output values can change immediately whenever input variables change

$a = \omega(z,e)$ $\quad z^{new} = \delta(z,e)$ $\quad e \in E;$ $\quad a \in A;$ $\quad z, z^{new} \in Z$



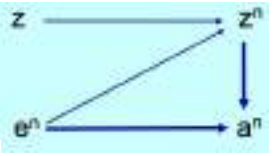


Mealy Type 1

- First: Based on new input output is calculated
- Second: A change into the next state is performed
- Synchronous Sequential circuits

Mealy Type 2

- First: change into next state is performed based on input
- Second: output is calculated while input is unchanged and still valid



Representation of FSMs

Time Chart of assignments
Sequence Table
FSM Table
State Graph (DFA)

z_k	x	z_{k+1}	$y_0 y_1$
z_1	0	z_1	0 0
z_1	1	z_2	1 0
z_2	1	z_2	1 0
z_2	0	z_3	0 0
z_3	0	z_3	0 0
z_3	1	z_4	0 1
z_4	1	z_4	0 1
z_4	0	z_1	0 0
nitial State z_k	Input Assignment	Next State z_{k+1}	Output Assignment

z_k	z_{k+1} x=0	z_{k+1} x=1	$y_0 y_1$
z_1	z_1	z_2	0 0
z_2	z_3	z_2	1 0
z_3	z_3	z_4	0 0
z_4	z_1	z_4	0 1

Moore FSM

z_k	$z_{k+1}/y_0 y_1$ x=0	$z_{k+1}/y_0 y_1$ x=1
z_1	$z_1/00$	$z_2/10$
z_2	$z_3/00$	$z_2/10$
z_3	$z_3/00$	$z_4/01$
z_4	$z_1/00$	$z_4/01$

Mealy FSM

Implementation of FSMs
State Memory is needed to store info - Feedback Loop can implement this very easily

Timing and Flip Flops
Synchronous sequential circuits - If all state memory is controlled within a sequential circuit by one or more synchronized clocks.

Synchronous sequential circuits
- Middle to large size
- Simple to analyze and design
- Synchronous sequential circuits are independent of (partially deployment-dependent) delays

Asynchronous sequential circuits
- Design is important
    - For bigger storage elements
    - For faster performance
- **If delay becomes smaller than the time for the clock signal to propagate on the circuit it makes sense to make operation asynchronous**

If the propagation Time of T is always larger than the maximum delay experienced within a sequential circuit
- Combinatorial circuits will have stabilized before influencing a new starter

There are two ways of taking the Clock into account:
- Level triggered
- Edge triggered

Level-triggered Control
Memory is transparent in one half of the clock period and it stores the value in the other half
The Input only influences the state if T=1, in case of T=0 the state is being stored

Implementation
- Combination of all input variables with T
- Level-triggered state memory is also called **Latch**

Drawback:
- The input may change during the active (transparent) clock period

Edge-triggered Control
Memory is only stored during a positive (0->1) clock edge or a negative (1->0) clock edge

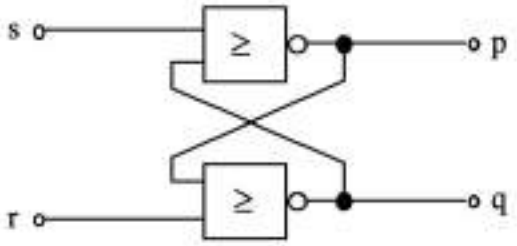
Advantage:
- Inputs only need to be valid for a short time
- They don't need to valid for the full half of the clock period's half (other than the level triggered case)

All "evaluation points" are determined exactly in time

Circuits symbols for the edge-triggered clock inputs

Asynchronous Rs Flip Flop

- Can't have r,s = 1,1 because p and q will be 0 which they can't be
- No Central Clock needed
- They are very sensitive - race conditions may appear
- Hazard error in transition of sequential circuits



- Reduce errors by following **normal fundamental mode**
- Only one input variable is allowed to change at the same time
- The change can only happen if all internal changes died out.

Problems known about asynchronous sequential circuits do not appear in synchronous sequential circuits

There might be oscillations but we wait long enough to stabilize so we don't care.

Synchronous sequential circuits always require a clock controlled state memory

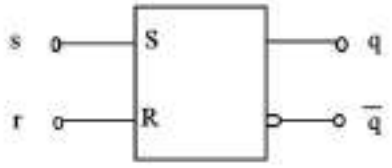
- A flipflop is used - there are different flipflop types

RS FlipFlop

Behavior

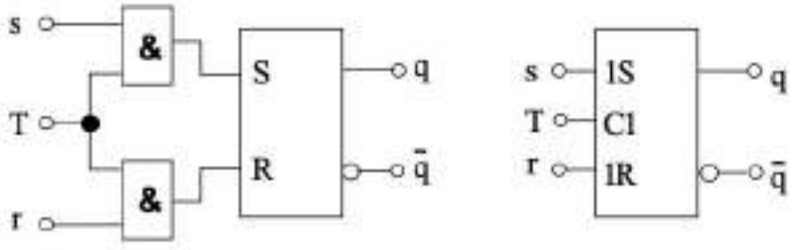
- Input *s* sets the storage ($s = 1 \Rightarrow$ output $q = 1$)
- Input *r* resets the storage ($r = 1) \Rightarrow q = 0$)
- Store: *r* and *s* both are set to $0 \Rightarrow q$ stores last value available
- The additional exit *p* is defined typically as the complement (negation) of *q*, as long as *r* and *s* do not equal 1 at the same time, thus; $p = \bar{q}$
- Impermissible: input (*r*, *s*) = (1, 1), thus not designed, $p = q = 0$ holds
- State variables *q* und its negation $\bar{q} = p$ (cf. earlier slide) are available as output

Symbol of the asynchronous RS flipflop:



From RS-FlipFlop to Level-controlled RS Latch

If we want to use a RS-Flip flop we need to add a clock like this:



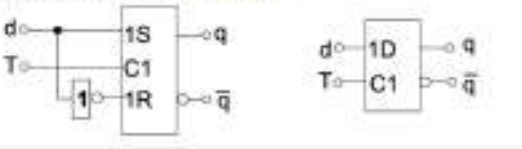
Adding a 1 before the S and R like this (1S, 1R) means that they depend on the same lock which is marked C1

If an input causes such a dependency the number succeeds the input variable, else the number precedes the variable.

Control table RS flipflop ("Triggering" Table

q^t	q^{t+1}	r	s	Interpretation
0	0	-	0	hold
0	1	0	1	set (S)
1	0	1	0	reset (R)
1	1	0	-	set

Circuit and symbol of the D latch:



D Flip Flop

Has a single input that also get imputed in a negated form

- Input config is **stored** for **one clock period**
- Input configuration is **delayed** by one clock period and accessible afterwards as the output configuration
- You don't see the output directly only one interaction later
- D for **Delay**

Function table

d	q^t	q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

Control table

q^t	q^{t+1}	d
0	0	0
0	1	1
1	0	0
1	1	1

Clock-edge Controlled D FlipFlop

To have a clock-edge controlled D Flip Flop you need:

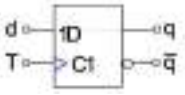
- Consists of 2 Flip Flops (Master & Slave)
- Clock given to master and same clock given to Slave but **inverted**
- Like this we get toggling
    - (No race Conditions)
    - If Master operational Slave is not - If Slave operational Master is not

https://www.youtube.com/watch?v=5ykewHgHYBI

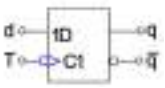
Variants of D Flipflops

- They are used as **clock-edge controlled memory elements**
- Small surface -> most integrated circuits for memory elements

Clock edged control:



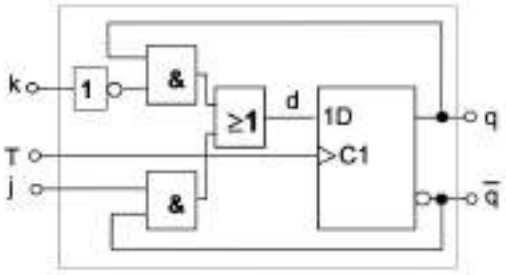
Negated Clock Edge Control:

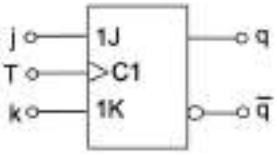


JK Flipflop

Similar to RS Flip Flop but can additional state to prevent error at r = s = 1

- States:
    - Store, Set, Reset
    - New **Complemented**





$d = q^t \bar{k} \vee \bar{q}^t j$

j	k	q^{t+1}	Function
0	0	q^t	store
0	1	0	reset
1	0	1	set
1	1	$\bar{q}^t$	change

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

T Flip Flop

- T for toggle
- We reuse a JK Flip Flop
- We give the same input to J & K
- To start the T flipflop, a set and reset input is required
- Synchronous set and reset of T Flipflop is not possible input e
- with input 1 output is complemented otherwise the value is stored

e	q^{t+1}	Function
0	q^t	store
1	$\bar{q}^t$	change

with

FlipFlop Control Summary

RS Flip flop (async)

- r=s=1 (not okay
- RS Latch (clocked, level triggered)

D Flip Flop, D Latch:

- r = 0 and s = 0 is okay

Edge triggered D flip flop implemented by two d latches

JK flipflop: r = s = 1 is a negating action

T Flipflop: with input 1 output is complemented otherwise the value is stored

Design of Synchronous Sequential Circuits

Design Procedure for a Serial Adder:

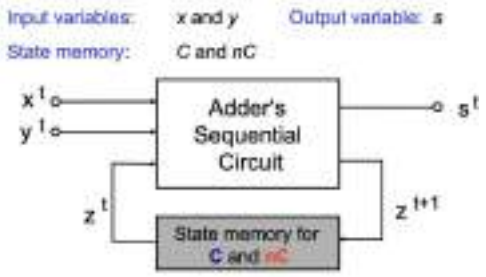
1. Definition of input and output variables

- Know what your variables are
- For the Serial adder you only need to know the carry from the previous operation

2. Determination of states

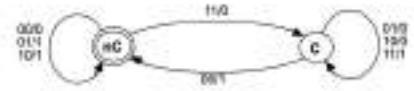
- You have to states carry - no carry

3. Drawing of block diagram

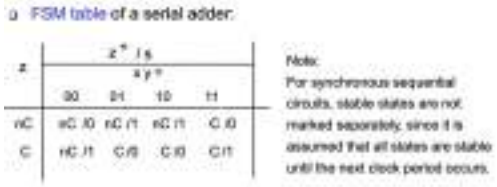


4. Design of finite state machine graph

Mealy sequential circuit is used since output depends on current input



5. Definition of finite state machine table

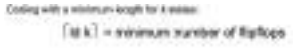


6. Determination of state coding

Asynchronous sequential circuits - very important, as state is crucial for the operation of the sequential circuit

Synchronous sequential circuits - state coding is not critical

Good state coding can reduce the amount of circuitry required for synchronous sequential circuits.



7. Generation of coded sequence table

Showing the State Variable variants with inputs and outputs

8. Extension of coded sequence table with flipflop types control

- Functions and terms can be read from the code sequence table
- Determine Flipflop type for the transition function's control circuit
- FlipFlop influences size of the transition function's control circuit

9. Minimization of output and control networks for flipflops

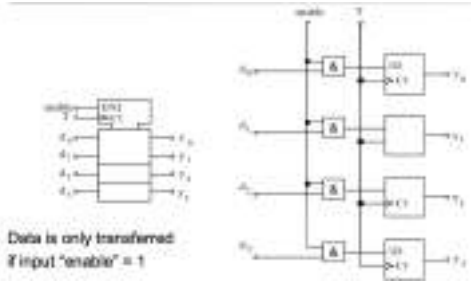
10. Drawing and implementation of sequential circuits.

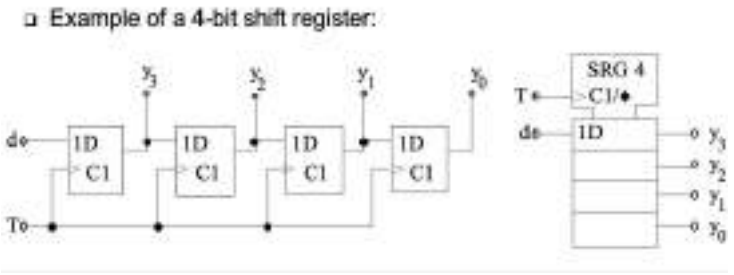
Registers
Registers are linear **arrangement** of flipflops for storing several bits, a **bit vector**

All flipflops for a register are controlled by a **single clock**

Usually all flipflops are influenced by additional joint controls

A 4 bit register with an enable Flag:





Shift Registers
Shift their content across interconnected memory elements
- Chain of D flip flops
- Output of one FlipFlop is the input of the next
- Useful for division by 2
    - Shifting a binary number to the right corresponds to a division by 2
    - Shifting a binary number to the left corresponds to a multiplication by 2

Ring Counter / Circular Buffer
- Connecting Output with first input for continuous passing of information

Counters
Do the counting of “events” - pulses / state changes
- Point them at successive addresses for a given memory for program counting
- Check successive work steps for control units

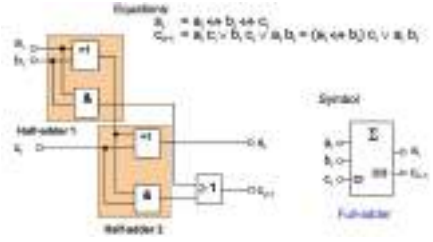
Can be used as **frequency dividers**
- A given pulse can be reduced in frequency

Asynchronous Counters
Size of control logic for asynchronous counter increases rapidly
Easier implementation is to just chain D Flipflops together
- But they are very slow
- And it’s output doesn’t change simultaneously

Half-adder
Adding two binary 1ary numbers, provide sum and carry but not final sum

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full-adder
Adding two binary n-ary numbers - created by two half adders

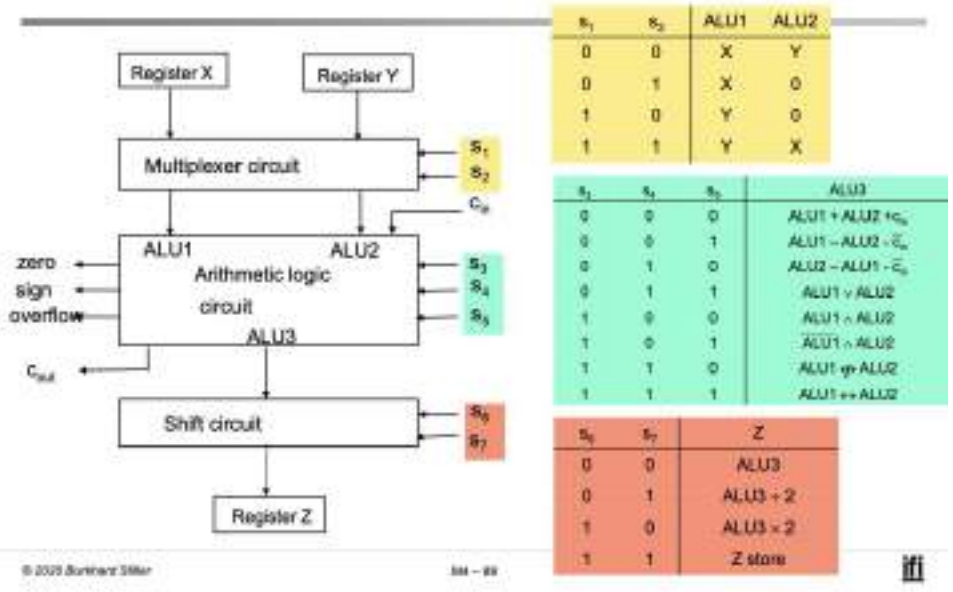


Carry-lookahead adder

- Control Logic size increases with the number of digits with Full-adder
- Parallel Carry Generation can solve this problem
- Hierarchy needed of carry-lookahead-adders

Arithmetic Logic Unit

- Functional **core** of digital computers
- Performs **logical** and **arithmetic** operations
- Input: Data and control signals from processor
- Output: Results and status signals to processor
- ALU often calculate only in fixed point numbers - floating point numbers need to be converted before



s_1	s_2	ALU1	ALU2
0	0	X	Y
0	1	X	0
1	0	Y	0
1	1	Y	X

s_3	s_4	s_5	ALU3
0	0	0	ALU1 + ALU2 + c_{in}
0	0	1	ALU1 − ALU2 − $\bar{c}_{in}$
0	1	0	ALU2 − ALU1 − $\bar{c}_{in}$
0	1	1	ALU1 ∨ ALU2
1	0	0	ALU1 ∧ ALU2
1	0	1	ALU1 ⊼ ALU2
1	1	0	ALU1 ⊽ ALU2
1	1	1	ALU1 ↔ ALU2

s_6	s_7	Z
0	0	ALU3
0	1	ALU3 ÷ 2
1	0	ALU3 × 2
1	1	Z store

- S1 & s2 decide in which order or how X & Y should be stored into the Arithmetic logic circuits 1-2
- S3- S5 decides the output operation to be performed
- S6 & s7 shift ALU3 before saving or just save

Components of ALU:

- Set of registers - storing input & output variables
- Multiplexer circuit - Determines which input variable will be stored in which ALU memory
- Arithmetic logic circuit - performs arithmetic and logical operations
- Shift circuit - performs multiplications, divisions, or leaves inputs unchanged

Inputs - Data Words X & Y, Control Signals s1….-s7 for selecting the right operation
Outputs:

- Status - zero, sign, overflow
- Control unit can detect ALU states and act accordingly

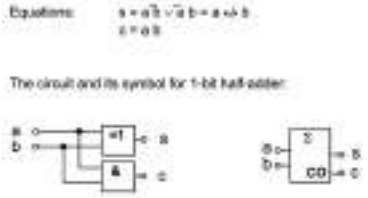
Bit-disc (bitslice-) ALU

A expandable structure on a device with a short word length - 4 or 8 bit

Bitslice: Concatenation of k Bitdisc blocks - for processing words of length k * m

You perform the same operation in parallel on different parts of the operand

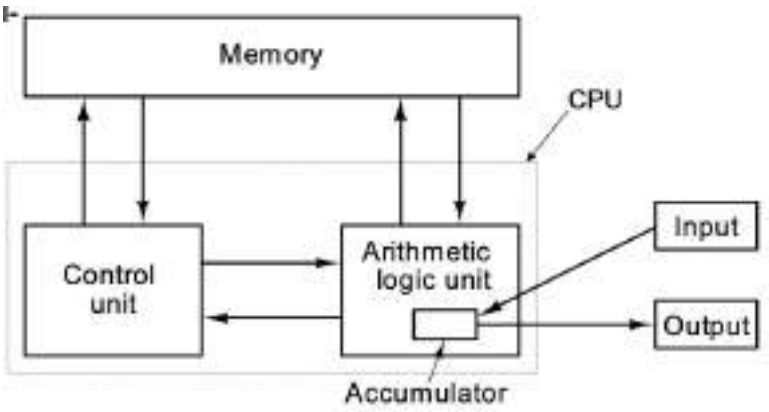
Common control is provided for example by a microprogram sequence and microprogram control unit



5 - Computer Engineering

von-Neumann Architecture
The von-Neumann architecture defines the **basics** of all **hardware architectures** today.

- CPU (Central Processing Unit)
    - Control Unit
    - Arithmetic Logic Unit
- Memory
- Input/Output Units



Motherboard / Mainboard
- Carries all components on a printed circuit board

Micro processor - an instance of a CPU
- Operated programs
- Controls and management of hardware

RAM memory (Arbeitsspeicher)
- Stores programs currently in operation
- And the belonging data

ROM memory - maintainer of programs
- Does hardware checks
- Boots operating system of the hard drive

Busses and interfaces - facilitate communication between all components
- Link between motherboard and periphery (graphic cards, network card, hard drives, printers)

Chip Set - maintains functionality of the mainboard
- Sequential circuits for the CPU & ALU
- Input / Output interfaces and their control
- Access to memory and storage units

Microprocessor
- Implemented as sequential circuits
- Millions of transistors

Microprocessor components:
- ALU - logic and math operations
- Control Unit - two Registers - instruction pointer and the instruction register (to control operation of programs)
- Registers - storage within the processor
- Busses - Connects the processor with is components
    - Data bus: RAM memory connection
    - Address bus: memory data
    - Control bus: Activated peripheral devices

Register

- Storage for inside of the processor
- Fixed length
- Operations are only possible when data is in these registers
- Short physical distance to control unit and ALU

Working Register - identified by IDs, data register type & address register type
Instruction pointer - contains address of the next instruction
Instruction register - stores a single binary machine instruction code
Stack Register - stores processor's state and the instruction pointer
Flag Register - contains execution states

Stack
LIFO Principle

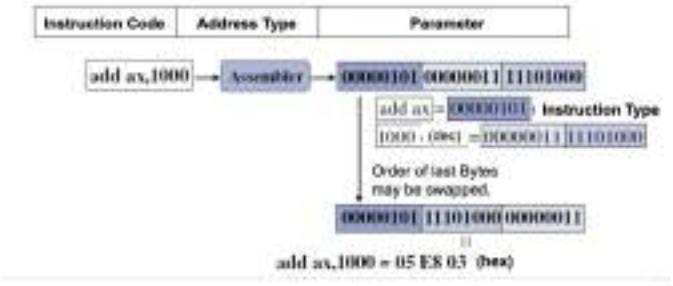
Operations of a Processor

1. The instruction counter of the control unit contains the address of the next machine instruction.
2. The address of this instruction will be received from the address register via the address bus.
3. The instruction will be transferred from the RAM memory to the instruction register.
4. The decoding logic analyzes this instruction's content and starts its execution.
5. During the execution of this instruction,additional data may be read from the RAM memory, indication signals may be issued to the periphery, a calculation of the ALU will be started, or a program counter's jump will be indicated. The state of each operation will become available in the flag register (Statusregister).
6. In case of a jump, the instruction pointer will be adjusted to the new address, otherwise the instruction pointer will be increased by 1.
7. The Processor Continues With step 1 above.

General Six Level Model of Instructions
Level 5 - **Problem-oriented language level**- Java, C#, C, ….
Level 4 - **Assembly language level** - Java Byte Code, MSIL/CIL
Level 3 - **Operating system machine level** - Unix, Windows, iOS
Level 2 - **ISA (Instruction Set Architecture) level** - x86, PPC, Sparc, ….
Level 1 - **Microarchitecture level** - Netburst, ISSE
Level 0 - **Digital logic level** - P4, P111, Sparcv9

Machine Instruction Codes

- Processed by the instruction pointer
- Not readable
- Symbolic machine instruction code - Assembler
- Every processor runs different decoder logic
- Syntax varies



Arithmetic Instructions - enable calculation and logic decisions - ADD, INC
Jump Instructions - enable deviations from linear processing of a program - JMP, CALL

Transport Instructions - enable transport of data between Processor, RAM, input/output units
Processor control instructions - prioritize tasks are applied to internally organize and maintain state of a processor

Addressing types

- Absolute addresses bad because need to know them when programming

Register-based addressing
Parameter is already available in register - no memory access needed

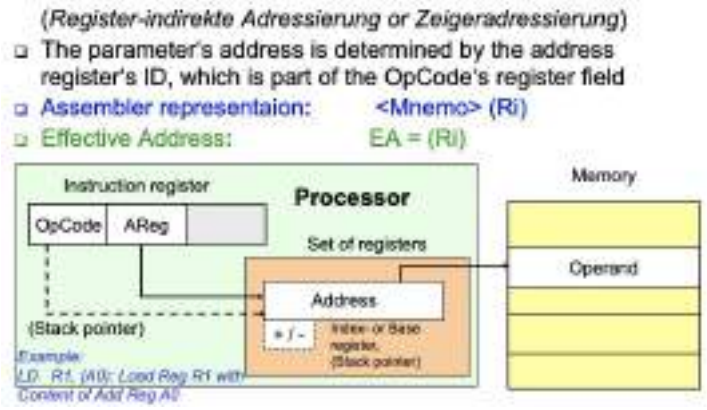
Single (one) step memory addressing
One calculation needed to determine the effective address

Two step memory addressing
Multiple sequential address calculations and memory accesses needed - first calculation leads to memory address which then again leads to a memory address

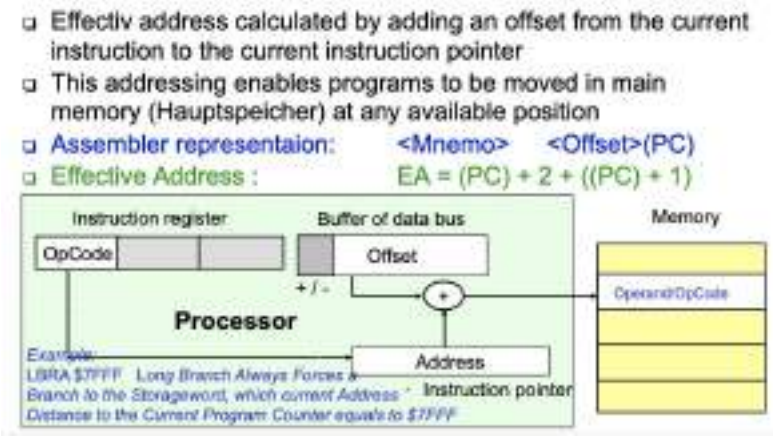
Implied Addressing
You provide the memory address in the operation code of the instruction

Register Operand Addressing
You provide the memory address in the Instruction Register Part of the instruction

Register-indirect Addressing



Program Counter-relative Addressing



Comparison

Type	Pentium II	UltraSPARC II	JVM
Immediate	x	x	x
Direct	x		
Register	x	x	
Register indirect	x		
Indexed	x	x	x
Based-indexed		x	
Stack			x

Memory

RAM memory (Arbeitsspeicher) and mass storage (Massenspeicher)

RAM memory - memory of a machine, stores programs and data - accessible at any time, immediately

RAM

- Temporary storage of data
- Short-term memory
- Read-write
- Volatile
- For applications and user-programs

Rom

- Non - volatile
- Long-term memory
- Operating System Kernel
- System tables

Storage element - 1 Bit memory

Memory cell - fixed number of storage elements, that is addressable by a single address

Storage word - Maximum number of storage elements which can be transmitted within a given bus cycle

Direct Access - each memory cell can be directly accessed - through address decoder

Organization - memory circuit organization depends on #memory cells and #storage elements per cell

Capacity - Information volume which can be stored within the memory - n*m Bit

Speed of memory depends on:

- Access time - maximum duration after feeding the address to a memory and the availability of the data at the output
- Cycle time - minimal duration between two subsequent memory accesses

Cycle time may be much longer than access time.

- Memory cell needs to recover first from access
- Sometimes a read process destroys therefore a write happens at access too

Ideally Cycle Time = Access Time - Reality: Cycle Time > Access time (up to 80%)

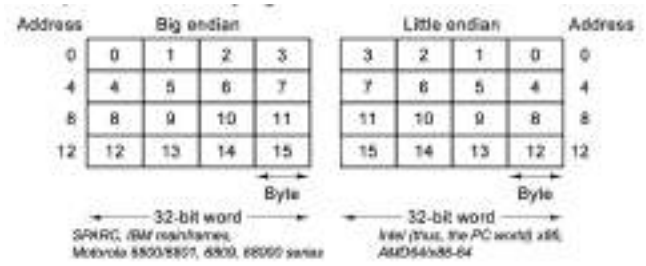
DDR3 - 2,7 Gbit/s, 1.5 V, refresh rate fixed

DDR4 - 3.2 MT/s, 1.2 V, dynamic refresh rate

Organization of Main Memory
Linear list of storage words

- Access times depend on chips in use
- Width of main memory typically the same as the data bus (8, 16, 32, 64)

Representations - Big Endian vs. Little Endian
Enndianess refers to the order of bytes within a binary representation of any digital data.



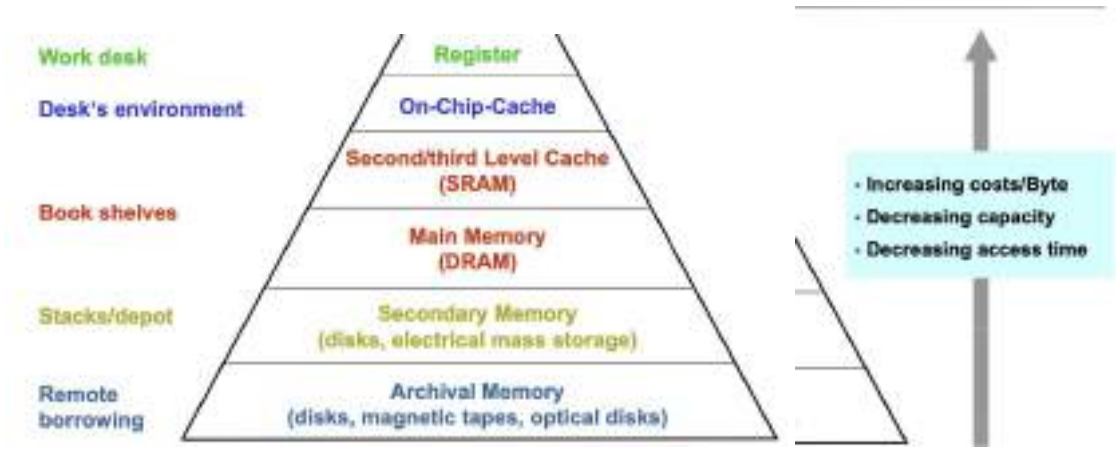
Bulk Storage
Magnetic data storage: Bits are represented by magnetic areas with polarity - HD, Floppy Disk, ZIP/JAZ drives

Optical Data storage: are stored via reflecting metal surfaces, sampled from a laser, bits are represented as pits or unchanged land.

Magnetic optical: a mix of magnetic and optical write and read operations is applied

Caches: short access times, improving the processors access time to the data
Virtual memory: Enlarging the existing main memory by a larger address space - multiple processes at the same time

For economic reason keep your data in different layers.



Cache
Problem:

- Cycle time of busses attached to modern processors are much shorter than the cycle time of large, cheaper DRAMs
- Requires to introduce wait cycles
- -> only small storage is implemented as cache

Idea:

- small , fast memory of SRAMs (Static Random Access Memory)

- Located between the processor the slower & cheaper main memory

Solution:
- Caches are small
- Fast buffer-style storage cells are located before larger and slower storage
    - To increase the likelihood of a shorter access time to the data

Principle of a Cache - buffer maintains copies of the main memory which the CPU is likely to access the most
- Cache is maybe implemented on the processor chip (on-chip cache)
- Or implemented as the fastest and expensive SRAM tech (off-chip cache)

Computer Components
Hardware - mechanical units, elements, and chips

Software - programs able to be operated on the computer

Firmware - microprograms (stored on ROMs), intermediary between hardware and software

BIOS - Basic Input/Output System
- Used to be implemented on the chip on the mainboard
- Today: flash PROM (a small battery powered RAM)
    - All configuration parameters and systems time is stored
    - No External power needed
- On startup various test are run to display messages - then operating system is loaded
- POST - Power on self tests (hardware components testing)
- Simple Communications with Hardware - systems clock - booting from a hardware, ….
- Control Handover to data storage - BIOS hands over to the booting storage
    - The Master Boot Record MBR (Start sektor) of the booting drive will be loaded into main memory and executed.

System Chip Set
- Chip Set is the physical interconnection between all hardware components
- Vary hard in performance
- Chip Set determines which hardware component should be used and how to communicate with it
    - System bus, Storage type, Interfaces, Type of processor
- Chips need to be embedded into a package against mechanical impact
    Materials: Plastic, Ceramics

Buses and Interfaces
Needed for **communication between all mainboard components** and peripherals (graphic cards, hard disks, printers)
All transport of data between mainboard components is operated by **internal busses**
For ultimate **speed** internal buses use multiple parallel lines
- One each per bit needed
- Chip set determines maximum number of bus lines
- #lines same as processors register width - > parallel processing of words

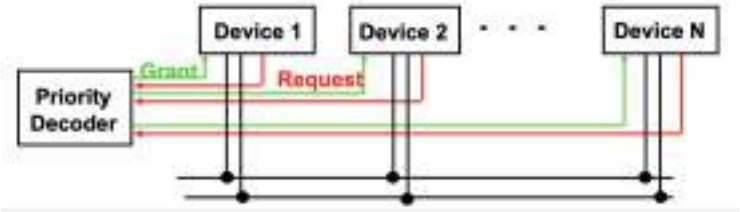
Data bus: bidirectional transmission between any number of devices
Address bus: unidirectional transmission of addresses to the storage or input/output devices
Control bus: Coordination of access rights to the data and address bus

Communications
Polling: In well determined time intervals we periodically check if there is data to be communicated, devices can't start communication themselves.

Interrupt Requests (IRQ): A communication with the processor may be requested or started by any device by issuing an interrupt request (a dedicated signal)

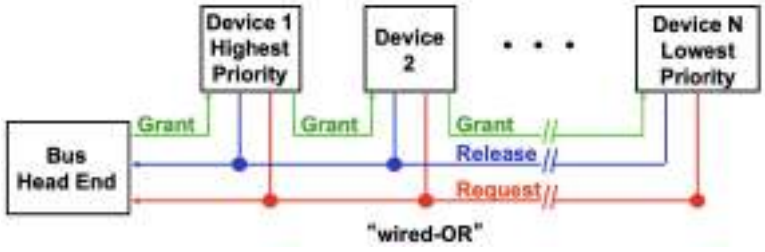
System Bus Arbitration
Centralized Approach (independent requests):

- Every device requests access to the system through their bus arbiter
- A **centralized priority decoder** decides if multiple requests come in the order based on preprogrammed rules
- Arbitration works with two messages:
    - Request = request for an arbitration
    - Grant = grating the arbitration

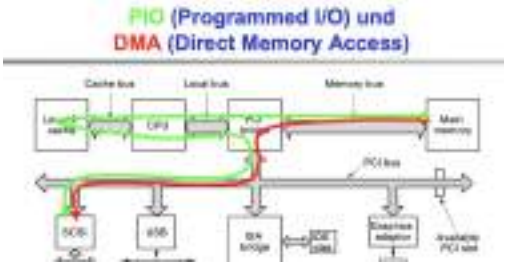


Decentralized Approach (Daisy Chaining)
All arbiters are chained based on their physical position

- Each arbiter forwards the Bus Grant messache (there are no local bus requests) to the next arbiter
- First arbiter in the chain belongs to the device with the highest priority
- A Release message is forwarding of a not needed grant



Interfaces for Internal Communications
Programmed I/O - pre-negotiated memory block to exchange data between processor and device



Memory-mapped I/O - addresses for the I/O devices are mapped to addresses used within the main memory

Direct Memory Access and Bus Mastering - Dedicated approach establishes direct transmission of data between device and main memory

Direct Memory Access (DMA)
Advantages:

- No memory access Load, Store and loop instructions necessary, because data transfer runds without any programmed control
    - Only two memory access per data required - processor load is reduced and can run tasks which do not need the system bus

Drawbacks:

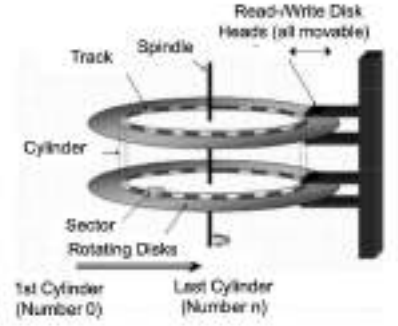
- Inconsistencies of the cache need to be handled

In general - it depends on the application

USB - simple plug and play, no configuration, shared bandwidth

Hard Disks - Bit is stored as a magnetic part on a spinning disk

- Bits are organized in tracks which is structured in sectors

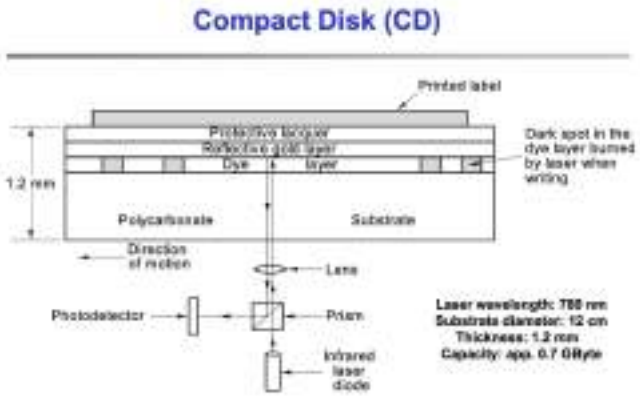


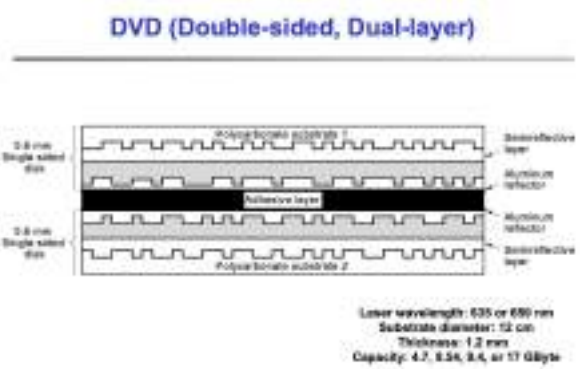
Solid State Drive (SSD)

- No movable parts
- Integrated circuits to store data persistently in flash memory
- Secondary storage in hierarchy of computer storage
- Resistant to shock
- Silent
- Faster access time
- Lower latency
- Capacity and costs

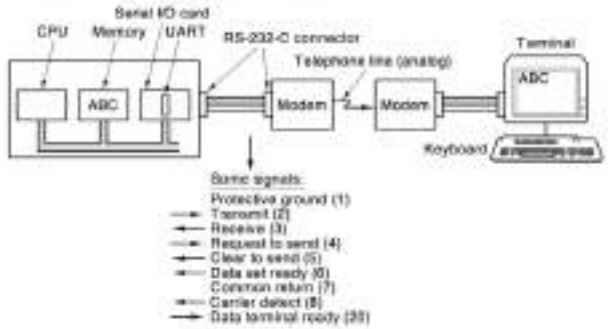
Optical Disks

- Optical storage data
- CD (compact disk)
- Thin metal layer with gets pits or lands and laser reads
- DVD (digital versatile Disk)

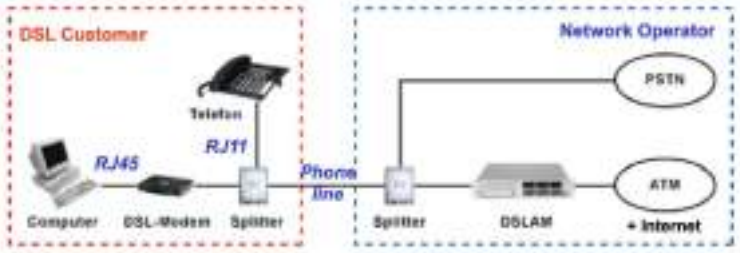




Modem – Interconnection of Computers (old)



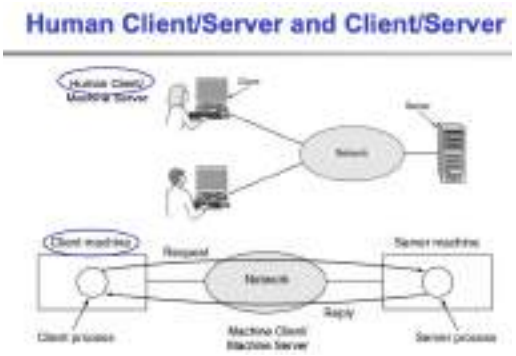
ADSL Modem – Interconnection of Computers (new)



(A)DSL: (Asymmetric) Digital Subscriber Line
DSLM: DSL Line Multiplexer
PSTN: Public Switched Telephony Network
ATM: Asynchronous Transfer Mode

6 - Computer Networking Introduction

Goal is the exchange of information

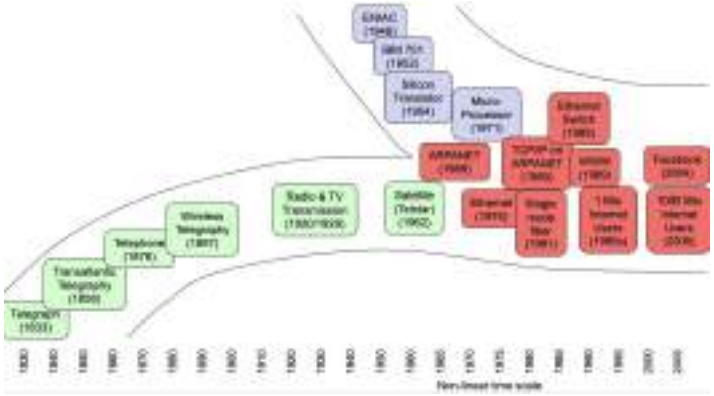


Purpose - human to human / human to machine / machine to machine communication

Achieved through Language, signals, characters, bits waves

Problems might semantics and mappics and the transmission of media

Evolution of Computer Networks



Major Network Components

End systems (OSI / IETF), terminals, computer, nodes, hosts, can be fixed or mobile

- Offer services to users (customers)
- Interfacing humans

Intermediate systems (OSI / IETF), bridges, routers, gateways, …

- Simple: Concentrators, bridges
- Complex: Switches, IP routers

Links, lines, cables, circuits, “radio links”, channels, ... , Access and backbone

- Phone lines
- Coaxial cables
- Fiber optics
- Radio links

OSI - Open System Interconnect, **IETF** - Internet Engineering Task Force, **IP** - Internet Protocol

Computer Network - set of autonomous computers exchanging information, each machine as memory, processor, and operating system

System - units that regularly interact with each other

Distributed system - geographically distant, autonomous interconnected computers, which offer a service based on cooperation

Application - single host based service (word)

Distributed Application - app running on a distributed system

Difference Network vs. Distributed System - Transparency of the distribution

Terms
Data - Facts, Digital data (discrete characters), Analog data (continuous function)
Information - Data that is processed, interpreted, structured, and organized-> useful information
Knowledge - Practical understanding of a subject, “justified true belief”

Signals - physical repr. Of data in space and time changes of physical parameters

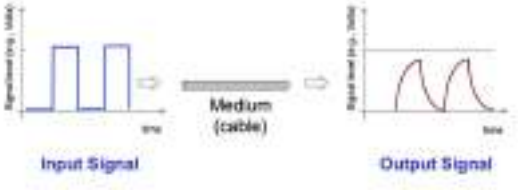
Messages - are digital data to be transmitted - structure is defined in communication protocols

Communication - to share in latin

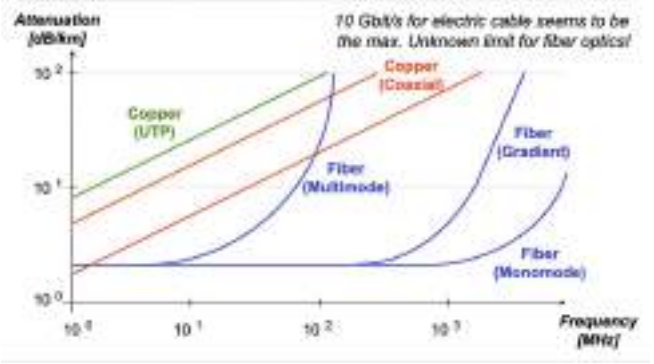
We transmit data as messages

Every data representation requires a signal representation

Attenuation



Transmission Links



Traditional Network Classification

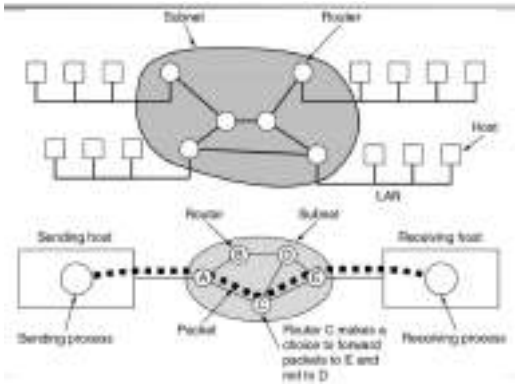
Distance	Coverage	Type	
1 m	Desk	PAN	Wireless keyboard and mouse
10 m	Room	LAN/SAN	
100 m	Building	LAN	
1 km	Campus	LAN	Interconnected LAN
10 km	City	MAN	
100 km	Country	WAN	
1,000 km	Continent	WAN	
10,000 km	Planet	GAN	The Internet

Personal Area Network: PAN
Storage Area Network: SAN
Local Area Network: LAN
Metropolitan Area Network: MAN
Wide Area Network: WAN
Global Area Network: GAN

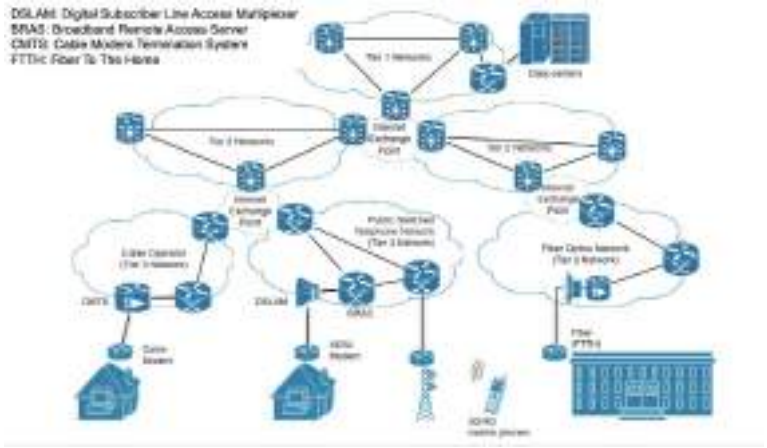
Local Area Network
Bus - one connected to the next to the next
Ring - they form a ring
Star - central entity connecting to everyone

Wireless Networks
Either you send out signals everywhere or you have a base station that transmits from a wired network locally.

Wide Area Network



The Internet - A LAN, MAN, WAN



Standardization
Based on joint and public agreements on norms and requirements to ensure interoperability
Standards don’t define how and why something works

De Facto Standard - informal convention or dominant use

De jure Standard - specified by a formal standardization body - is time consuming and costly

Standardization is a must for interoperability of communications

ITU-T (Telefax, JPEG, Async subscriber line), ETSI (digital cellular communications system), ISO (Paper size, Single byte character sets)

Standardization of protocols and components:
- IETF: Internet Engineering Task Force
- IRTF: Internet Research Task Force

Standardization of Local- and Metropolitan Area Networks
- IEEE-SA: Institute of Electrical and Electronics Engineers Standards Association
- IEEE-SA also works in other areas:

Standardization: Example Internet

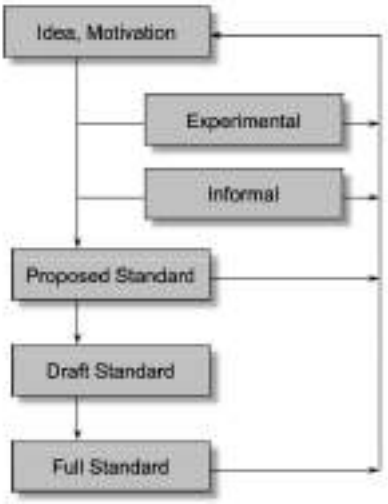
Process operated by the Internet Engineering Task Force (IETF):

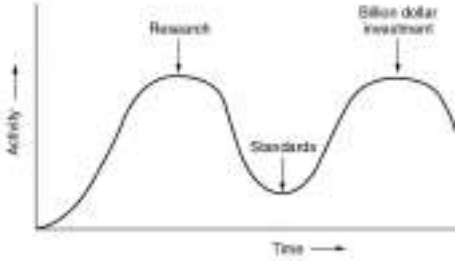
- Internet Engineering Steering Group (IESG) leads discussion

Two results:

- Standard
- Informal/Experimental

Already for a Draft Standard two independent and interoperable implementations are required





7 - Architectures and Service Protocols

Communication is the technical approach of communicating messages between participating partners

4. Major Characteristics

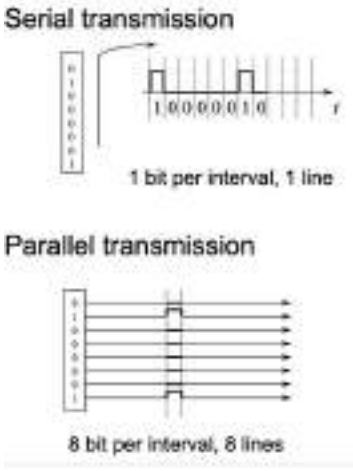
1 - Communication Partners:

Dialog - two hosts exchange data across a point to point link

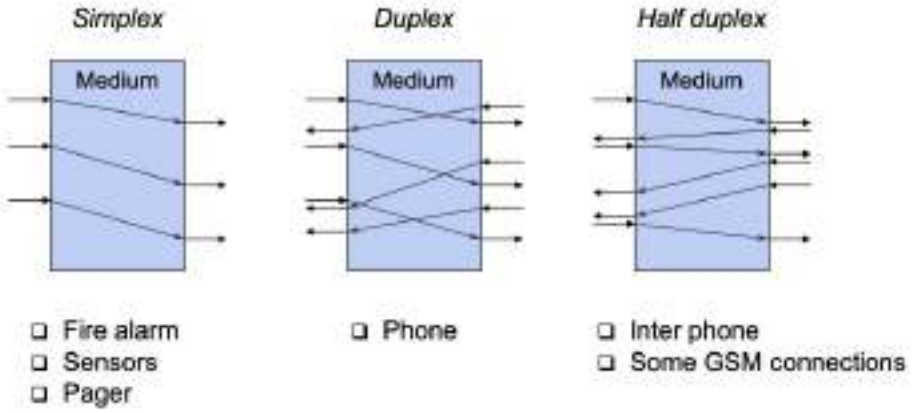
Multicast - single host sends to multiple other hosts

Broadcast - single host address many unknown hosts

2- Transmission Schemes:



3 - Directions of Use



4 - Quality of Communications:

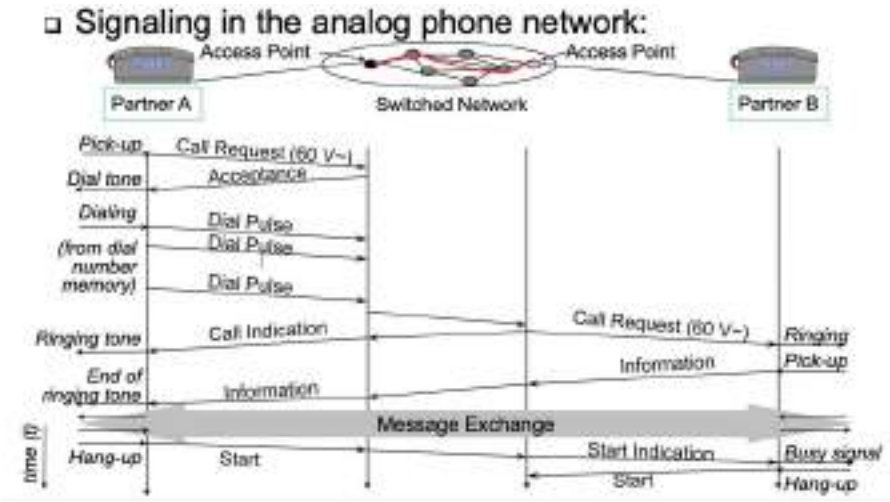
User requirements - ease of use, subjective ratings

Technical requirements

- Performance; Bandwidth or capacity, latency or delay, throughput, sending/receiving rate
- Reliability: Error tolerance, resilience, noise immunity, availability
- Security: Integrity, privacy, authentication, authorization, denial of service, non repudiation
- Safety
- Costs: investment, operation, maintenance costs

Throughput(or capacity) - Number of bits transmitted per second

Example Phone



ISO/OSI Basic Reference Model

Anwendungsschicht

Darstellungsschicht

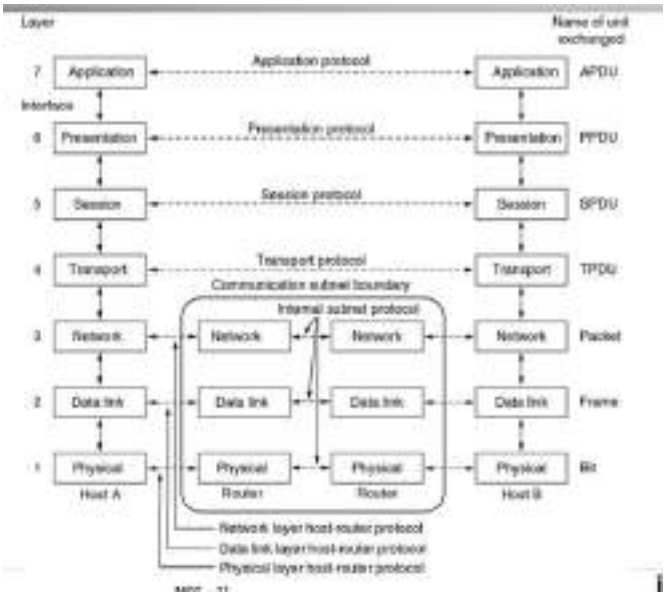
Kommunikationssteuerungsschicht

Transportschicht

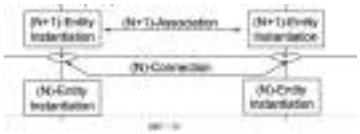
Vermittlungsschicht

Sicherungsschicht

Bitübertragungsschicht



Open System Interconnection (OSI) - Basic Reference Model

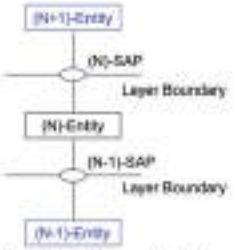
- Basis for enabling communication for systems of different vendors
- BRM is a guideline for all tasks for communication
- All Units are of an **(N)-Hierarchy** in all end systems
- Implementation of **(N)-Services** i an end-system
- There are many **(N)-Entities** types to implement different (N) Layer Protocol
- A copy of (N) Entity is an (N) Entity Instantiation
- **Peer Entities** - deliver functions of a service by data exchange
- **(N)-Connection** - Relation between two (or more) (N+1)-Entity Instantiations in an (N)-Layer being supported by an **(N)-Protocol**
- (N) Association: Relation between to NEntity Instantiations including the management of state information
    - Supported by (N-1) connections or by a N-1 connectionless service
    - May use multiple N-1 Connections time after time
    - 

N - Service-Access Point N-SAP

Within layered communication systems (N+1)-Entities communicate across (N)-SAPs (Service Access Points)

- An (N)-Entity offers its (N)-Service at the (N)-SAP
- An (N)-Entity utilizes services, which are offered at the (N-1)-SAP

Relations between (N-1)-SAP, (N)-Entity, and (N)-SAP



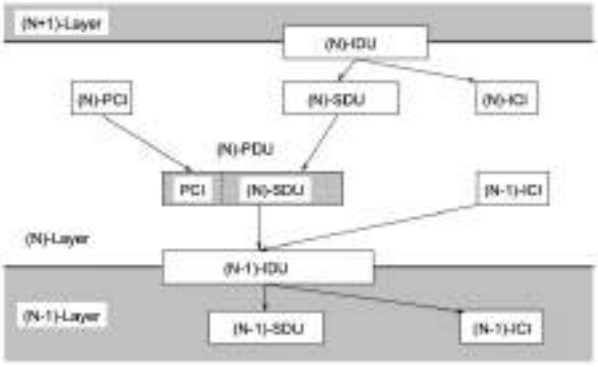
Internet example: A SAP is defined by an IP address

(N)-Connection End Point

An (N)-Connection End Point ((N)-CEP) determines the end point of an (N)-Connection within an (N)-SAP

Internet example: A CEP is defined by a port number

Generic OSI Communication Data Units



(N)-PDU
- Protocol Data Unit, PDU
- Data unit exchanged between (N)-Entities by applying a (N-1)-Layer service
- Combined out of (N)-PCI and (N)-SDU
- Equals an (N-1)-SDU

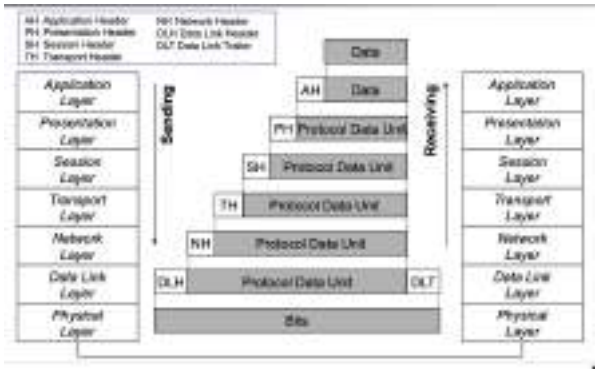
(N)-IDU
- Interface Data Unit, IDU
- PDU exchanged between (N+1)- and (N)-Entity at an (N)-SAP
- Combined out of (N)-ICI and (N)-SDU

(N)-SDU
- Service Data Unit, SDU
- Data being transmitted between (N)-SAPs transparently

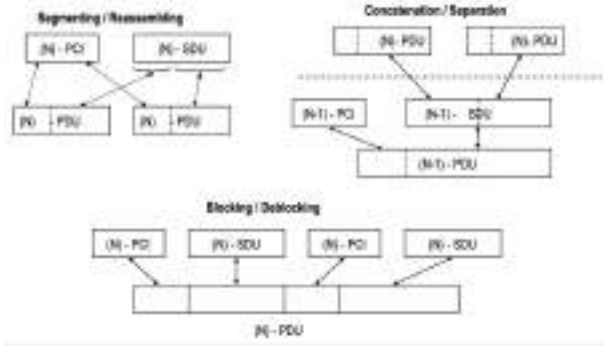
(N)-PCI
- Protocol Control Information, PCI
- Data exchanged between (N)-Entities to control the execution o operations, e.g., sequencing

(N)-ICI
- Interface Control Information, ICI
- Parameter exchanged between (N)-Layer and (N+1)-Layer to manage service functions, e.g., addresses

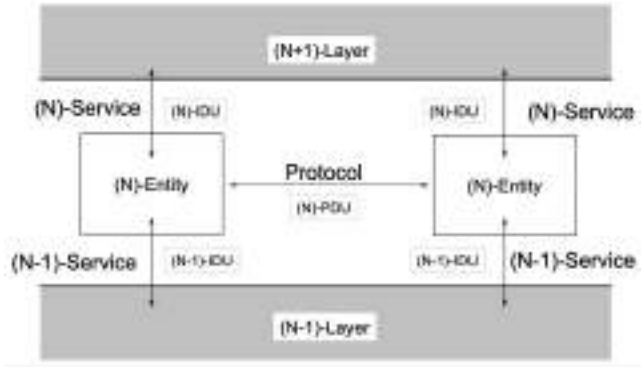
Encapsulation of Data



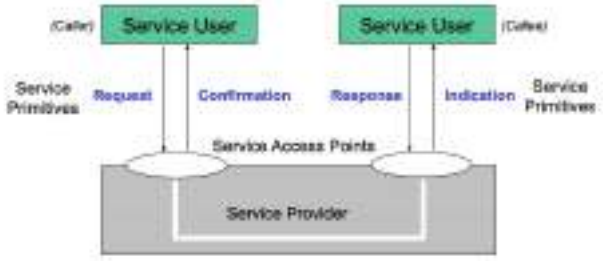
Handling of Data Units



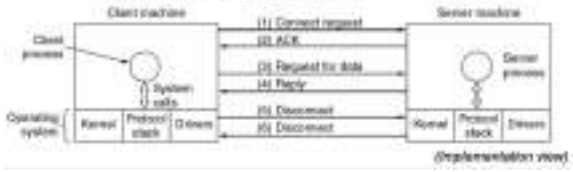
Communication in/between OSI Systems



Service and Protocol



Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

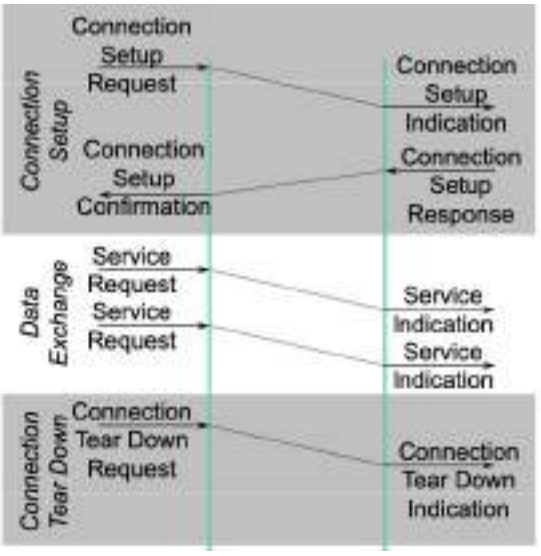


Layer/Application Name	Service	Service Type	Parameter
Physical (Ph) Data Link (DL) Network (N) Transport (T) HTTP FTP ...	Connect (Con) Data (Dat) Release (Rel) Abort (Abo) Provider Abort (PAbo) Disconnect (Dis) ...	Request (Req) Indication (Ind) Response (Rsp) Confirmation (Cnf)	(defined)

Examples:
- T-Con.Req(*Adresses*) = Connection request at the interface to the transport service
- HTTP-Get.[Req](*URL*) = Request of an HTML page being identified by the *URL*.

(Specific instances)

4 Service Types
Un-acknowledged Service - just sending off the request
Acknowledged Service - Sending Request and waiting for Response for Confirmation



Connection-oriented Services - consists of 3 phases

- Connection Setup - context building in network and host
- Data exchange - simplex, less context information required
- Connection tear down: context deletion and resource allocation

Connectionless Service

- No context of any relation between any transmission service
- No support of any sequence of delivery
- No negotiation between partners
- Implements unacknowledged service
- Entities do not store context about previous data exchanges

Addressing - unique identification of a communication partner's host

- Connection less service
  - Service request includes destination address, sometimes the source address
- Connection-oriented service
  - Context being established in setup includes addressing info
  - Multiple connections from the same SAP require a connection ID

Connectionless service - each data transfer is separate, no connection status is known
Connection-oriented service

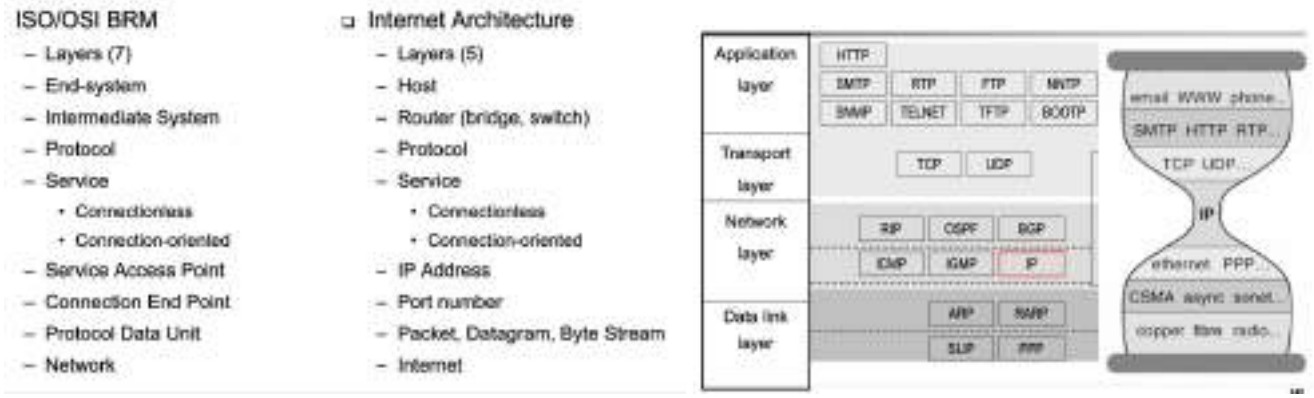
- Connection between user and service consists of N Layer and N-1 entities
- Request is performed by n-1 layer service primitives
- Based on protocol parameters can be used
- Data transfer possible if in connection state
- Tear down of connection is required

The internet Architecture

Application Layer	Application-specific functions supported by application protocols
Transport Layer	End-to-end data transmission between two hosts
Network Layer	Routing in the network, also named "Internet Layer"
Net-to-Host	Interface to the physical medium, "Network Interface Cards and Driver" (Access to medium and physical details)

Relation between OSI and the Internet
Differences:

- Layer 5 & 6 are included in the application layer
- Tasks of Layer 1 & 2 are included in host layer



Physical and Data Link Layer

Physical Layer:

- Is unsecured between hosts/routers
- Transmission of unstructured bit sequences
- Includes plugs, pins, mapping of scheme of data to signals

Data Link Layer:

- Error free data transfer
- Turns bits of layer 1 into frames
- Protocol Functions: Error detection, error handling, acknowledgements, time and sequence control, retransmission, flow control, and reset

Network Layer:

- Connects link layer to host-to-host connections
- Can be connectionless or connection-oriented service
- May include, inter-network, sub-network, or routing sub-layers
- Protocol functions: routing in case of switching, forwarding, congestion control

Transport Layer:

- Addressing of transport service users
- Data transfers between users in e2e systems
- Hides transmission and switching tech as well as any sub networks
- Can be connectionless or connection-oriented service
- Protocol functions: error control, flow control, acknowledgments

Application-oriented Layers:

Session Layer:

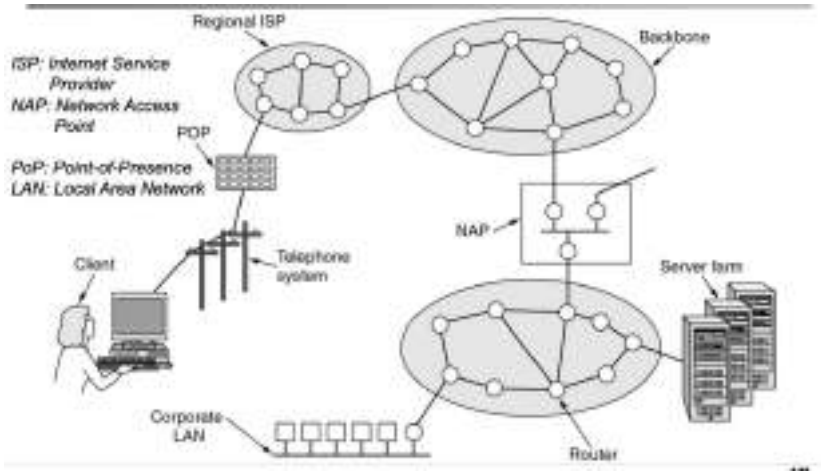
- Synchronization
- Defines sessions based on user requirements

Presentation Layer:

- Defines syntax for data to be transferred

Application Layer:

- Offers services to the user, email, file transfer, remote terminal login,...
- Offers generic services for applications: reliable message transfer, …


Regional ISP
Backbone
ISP: Internet Service Provider
NAP: Network Access Point
POP
PoP: Point-of-Presence
LAN: Local Area Network
NAP
Client
Telephone system
Server farm
Corporate LAN
Router

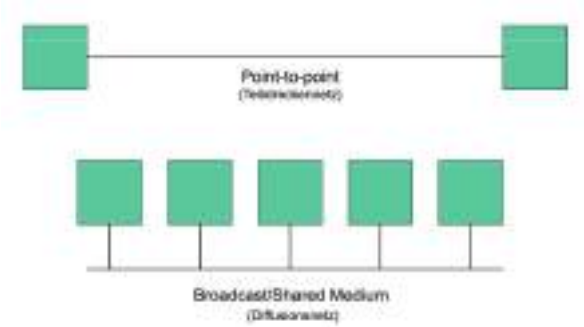
8 - Basic Concepts

A network is based off nodes (PC, router, switch, embedded system)

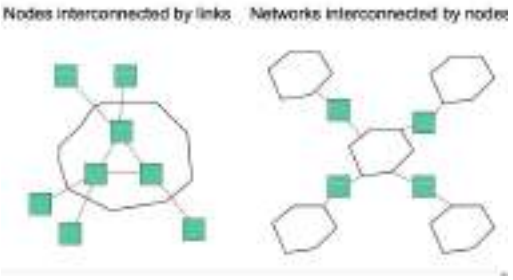
A Channel is an abstraction of a link

- Can be point to point or broadcast
- Propagation delay
- Has a capacity
- There are transmission errors, bit errors, error bursts
- Time dependent

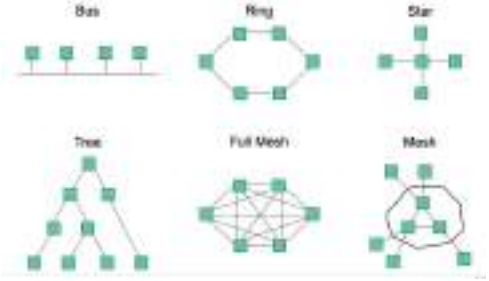
Point to point and broadcast links



Network is recursively defined:



Topologies



Important Aspects of Networks

- Resource sharing
- Inter-process communication
- Data transmission on a link
- Resilience (Ausfallsicherheit) against failures
- Error handling
- Coordination of access to shared resources
- Naming and addressing (identification of objects) Routing
- Fairness
- Charging and accounting

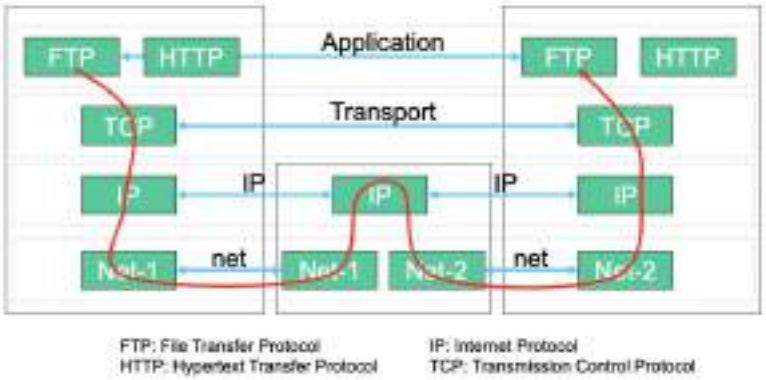
Multiplexing - transmitting more than one signal over one path

- Space-division - physical paths are grouped together

    - Analog transmissions can only be supported by this multiplexing
    - M input links, n output links
- Frequency-division - information transmitted in a certain frequency range (FM Radio)
- Time-division - Quantity information transmitted in a certain time range
    - Input is connected to output for a period of time
- Code-division - non inferring codes

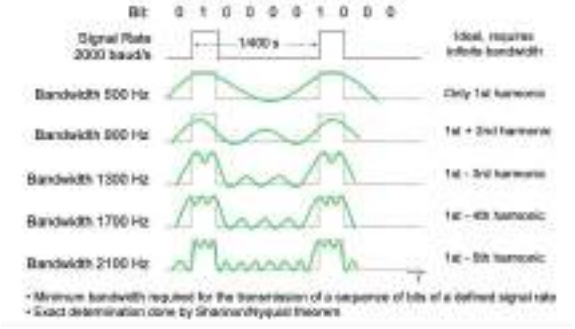
Switching: Network-wide Resource Sharing
Circuit Switching - management of e2e circuits with fixed resources
Packet switching - Management of packets Datagrams, virtual circuits
Message switching - Management of messages

Packet Switching on the IP layer

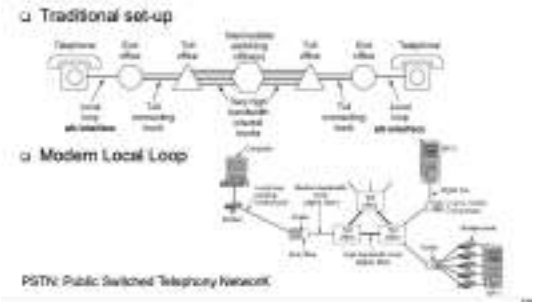


Physical Media
For any physical medium, frequency and length restrictions apply:
- Time * Bandwidth = const
- Length * Bandwidth = const

Bandwidth effects on digital signals



The PSTN



Phone: a/b Interface

- a/b stands for two cables
- These cable bring access to the phone network
- Phones used to need power supply

Pulse Code Signaling (deprecated)
- 100ms wait intervals for dialing to determine number

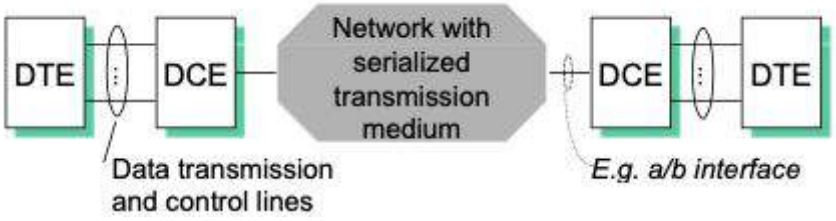
Multi Tone Dialing (today)
- 12 keys for dialing
- Each key encoded by two frequencies for redundancy

Frequency [Hz]	1209	1336	1447	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

697 & 1209 frequency identify the 1

Interface for Digital Data
A Data Circuit Terminating Equipment (DCE) (Datenübertragungseinrichtung) is located between Data Terminal Equipment (DTE) (Datenendeinrichtung) and the network

- The DCE adapts the signal to the access link (e.g. for a modem)
- DTE/DCE interfaces allows user to interface to a number of different end-systems



X Philosophie
- Minimized number of cables
- Because control and switching move into the DCE

Modulation - the transformation of a source signal into a different signal representation

FDM: Frequency Division Multiplexing
PCM: Pulse Code Modulations

Data	Transformation	Signal	Example
analog	Modulation	analog	FDM
analog	A/D transformation	digital	PCM
digital	D/A transformation	analog	Modem
digital	Recoding	digital	adaptation to other channels

Carrier is defined by a sinus oscillation wave

Amplitude Modulation - High amplitude 1 - low amplitude 0

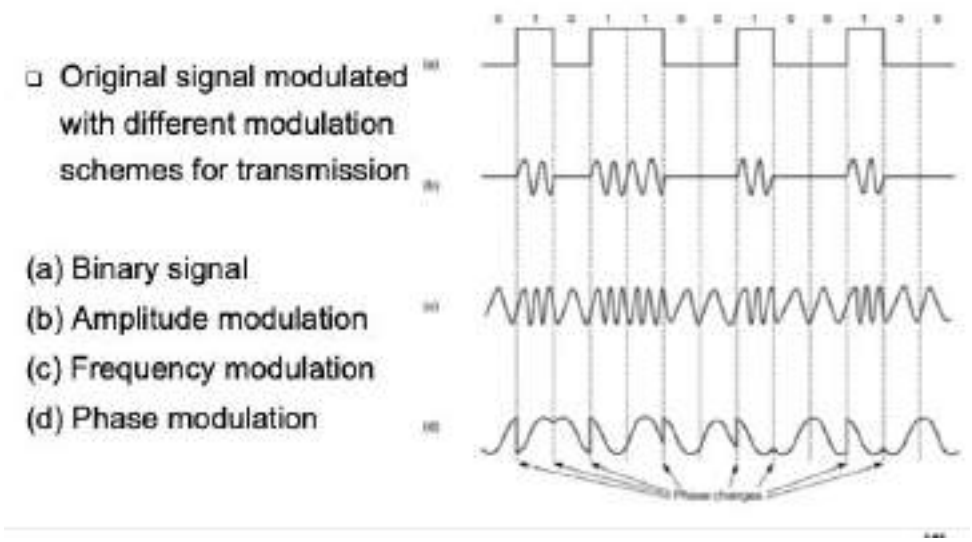
- Not that robust

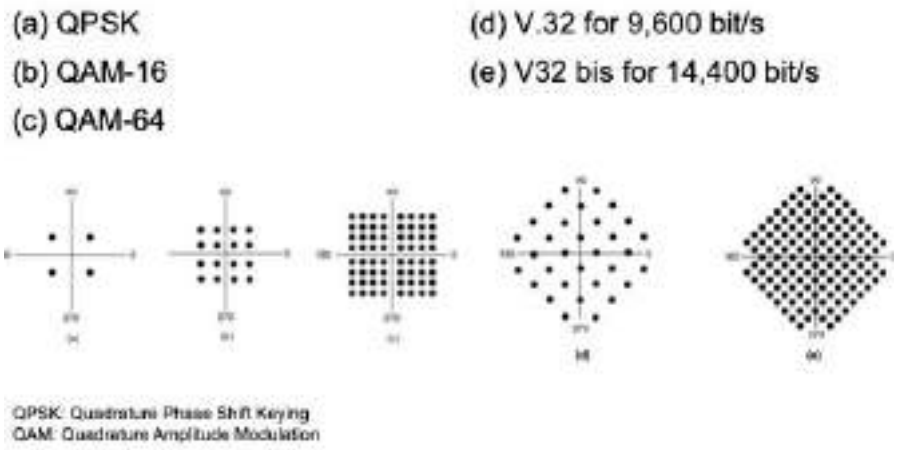
FM Frequency Modulation - High frequency 1 - low frequence 0

- Kinda robus

PM (Phasenmodulation) - 1: Phase 0 Degree, 0: Phase 180 Degree

- Robust, but expensive





Modem Technologies

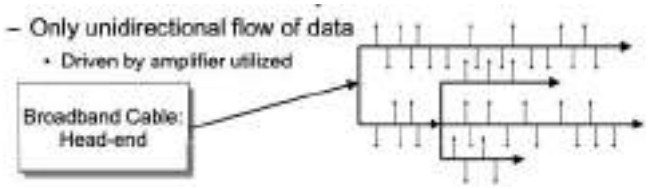
Cable modems - through broadband cables, data rates up to 2Gbit/s

Powerline communications - via power network - data rates up to 2Gbit/s, used when you have no space to deploy new cables or WLAN has to many outside inferences

xDSL modems - through traditional phone cables - 6-8 Mbit/s up to < 100 Mbits/s

Traditional Broadband Cable Network

- Tree based network
- Each leaf has a consumer (house/village)
- Operate at different bandwidth
- Use Frequency Division Multiplexing
    - Frequency for a dedicated service
    - #service is fixed by definition
- Here only unidirectional flow of data

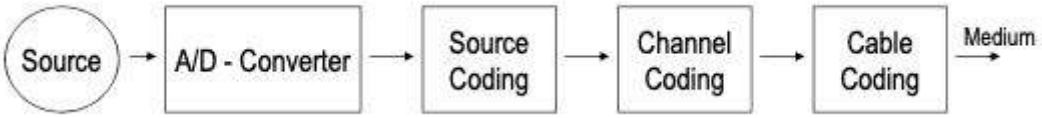


Link Encoding - Layer 1
Link Encoding helps to consider physical characteristics of the underlying network that help address a few errors

- Digital transmission capacity - how many bits per second we can transmit
- Energy level
- Noise level
- Error resilience (bit/burst)

Timing characteristic - We need to ensure that the senders clock is synced with the receivers clock
Transparency characteristic - we ensure consecutive 0 or 1 are not truncated or elongated

Encoding Principle



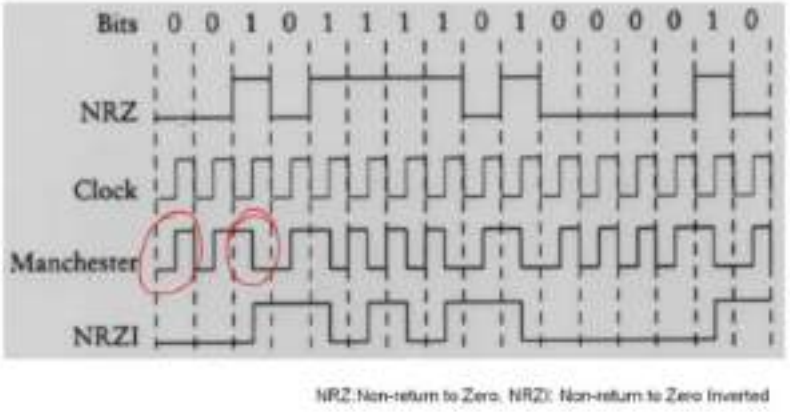
Prepare the sources signal such that the medium is capable of transmitting it

Source Coding - reduction of redundancy in signal (lossfree or lossy)

- Entropy coding, source coding, hybrid
- Get rid of redundant information

Channel Coding - Detects and corrects errors
Cable/Link (en-)coding - Assignment of bits to signals

- Depends on the underlying physical device



NRZ: Non-return to Zero, NRZI: Non-return to Zero Inverted

Manchester Code combines in an XOR the Clock and the NRZ signal

- Wanting to handle errors -> You need to transmit double the bits - more capacity -> adding more frequency

Framing - Layer 2
Frame - unit of processing at the link of sender/receiver

- To simplify storage management and operations on the bit stream
- Frame is not a packet

Framing Approaches:
Sentinel based - special bit pattern to show a start and end marker
Bit Stuffing

- Pay attention to not interpret data/content as a marker
- When sending modify the tag by inserting a zero after the 5th 1 if there are 6 consec. Ones
- When reading if you read 5 ones drop the consec 0
- Only when 6 ones are read a new frame starts

Counter based - include payload length in header
Clock based - each frame is a tim eunit long

Use a Frame Check Sequence to catch ill-recognized frames.

Character Stuffing
You don't want data to be interpreted as control signal
When seeing some control signal in data stuff a character in there and do like bit stuffing

Reliable Transmission on Unreliable LInks

- Physical media can have error, dusty cables

Guarantee a reliable transmission service
Error control:

- Detect and fix single bit errors
- Detect and fix burst errors
- Detect and retransmit lost frames
- Detect and eliminate duplicate frames
- Detect and fix wrong sequences of frames

Flow control:

- Adapt sending rate to receiver capabilities to avoid overflow situations

General concerns:

- Sometimes error and flow control are coupled
- Keep data in transit under control of some entity

Flow Control

- Enables continuous flow of data
- Avoids bursty data sending operations
- Avoids buffer overflows at receiver

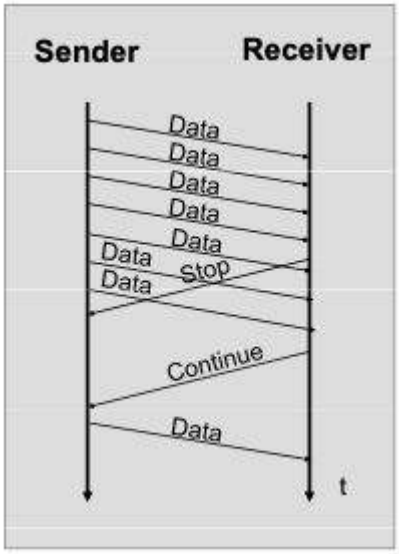
Credit or window based schemes

- Initial credit of N data units
- ACK i: Data units up to number i are acknowledged
- Credit j: Sender is allowed to send up to number i+j data units

If N is 1 you can send a request and need to wait for the acknowledgment then you can repeat
If N is > 1 we have a Window based Request sending style

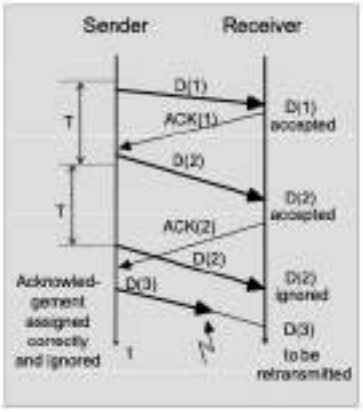
Rate based scheme - data units are paced - rates decided by timer or counter

Simple Flow Control



- Receiver should send Stop message early enough to still have space to fill up the cache with the incoming messages - till the receiver can react and stop sending
- Continue is able to accept further data
- Simple model - because we don't know the message size

Idle Repeat Request (IRQ)

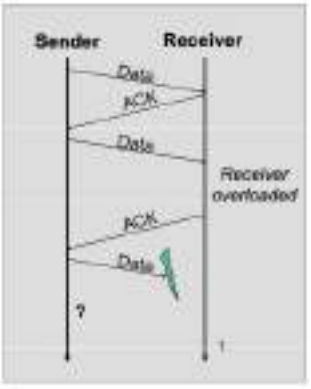


- Frame delineation and transparency
- Frame types:
  - Data D, acknowledgement ACK, and negative ACK
- Timeout T
  - At sender, receiver, or both
- Requires sequence numbers in data and acknowledgement frames

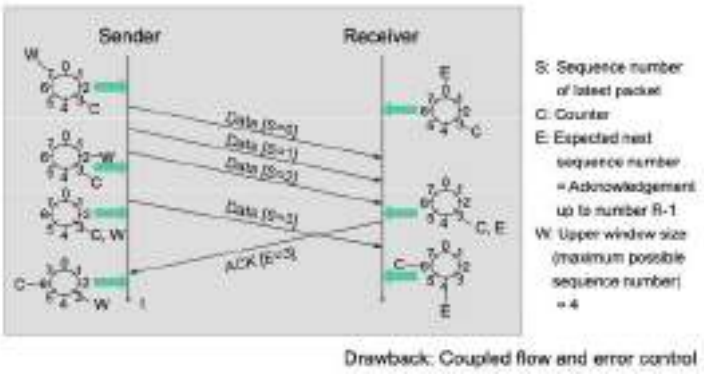
- If the acknowledgement comes after the timeout t interval to the sender will automatically send out the message again

Implicit Flow Control

- Operation
  - Withholding acknowledgement (ACK or NACK) may slow down the sender
  - Requires an error detection mechanism at the same time
- Problem: Sender is unable to distinguish cases
  - (a) Packet sent is lost
  - (b) Acknowledgement withhold due to overload



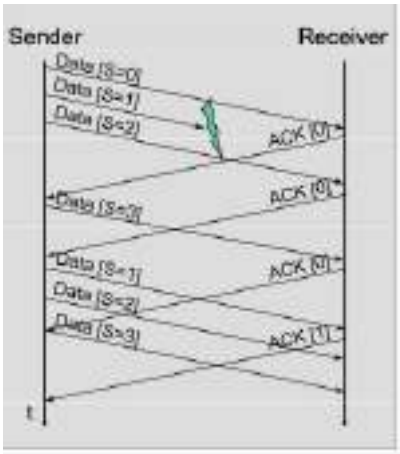
Sliding Window



- Both side have internal counters
- W indicates where the window size is

Jumping Window - makes the sliding approach an on off approach

Sliding Window



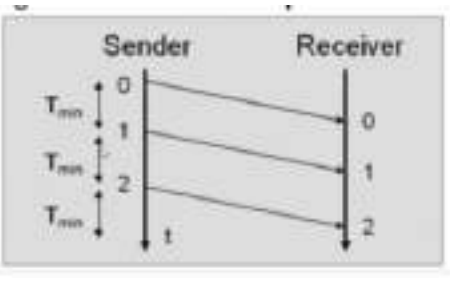
- You can have multiple packets “in flight”
- **Packet loss** - Multiple ACKs with same sequence number mean frame loss
    - Sender starts retransmitting from the first unacknowledged frame
- Even though s2 was received the sender doesn’t know about that - you have to go back to the point of failure
- **ACKs Loss**
    - ACK 0 and 2 come in
    - No retransmit required because 0-2 were received correctly just the ACK 1 got lost

Window based Flow Control Problems

- Decision when to offer a new credit
    - It’s a local decision - trade off between sending an acknowledgement every time or smooth sending
- Long RTT, high path capacity and small credits -> bursty traffic
- Amount of data in transit is large
- Once sent, data in transit cannot be influenced

A solution option is rate based schemes

- Avoid bursty traffic by controlling the burstiness
- Continuously send operation
- Regulating the send rate



Rate based Flow Control

- Limit max send rate due to small time interval rates

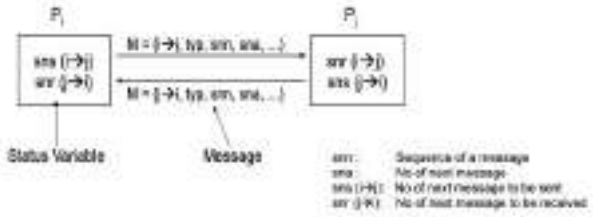
- For every sent data a timer is started
- Following data shall be send after time out at the earliest
    - Gives Continues flow of data units
    - But High resolution timer required

Error Control - consists of detection and correction

There's no brideye sender and receiver only have their POV

Protocols helping achieving reliability
- Sequence numbering
- Acknowledgement
- Retransmission
- Forward error correction
- Checksum calculation

Sequence Numbering - solve packet loss, duplication detection, sequencing and flow control
Every communication Partner P
- Numbers his messages M sequentially
- Based on his connection (differ between texts to wife, friends)



M = { i→j, typ, snn, sna, ... } defines

i→j : Connection and direction identifier
typ of { NM, ACK, NAK, SYN, CLS, ...}
snn : Sequence of a message
sna : No of next message
sns (i→j): No of next message to be sent
snr (j→i): No of next message to be received

NM normal M, positive ACK, NAK negative ACK,
SYN Connection set-up request, CLS connection release request

Solution: M of P_j arrives at P_i
Case A: snn = snr (j→i): no message is missing
P_i: (1) snr (j→i) +: = 1
(2) Sent ACK to P_j with sna = snr (j→i)
(3) Process messages
Case B: snn > snr (j→i): messages are missing
P_i: (1) NACK to P_j with sna = snr (j→i)
(2) delete messages

Note: Sender needs to store all messages until ACK arrives and not all error cases are considered.

Duplicates not detected
Some losses not detected
Case a)

Case b)

Case c)

- Note: ACKs always required, NACK may be omitted

Types of Acknowledgments

Sender-dependent – Sender requests for ACKs from a receiver.
Sender-independent – Receiver issues ACKs periodically or event-driven.
Positive ACKs – Correct reception of data is acknowledged.
Negative ACKs – Identification of incorrect received data.
Selective ACKs – Naming of a single explicit protocol data unit or segment.
Cumulative ACKs – Acknowledgements of one or many sequences of protocol data units or segments.

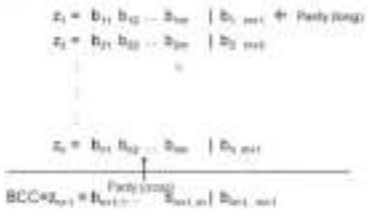
Types of Retransmissions
Selective – Retransmission of incorrect data only.
Go-Back-N – Complete retransmission of data after an incorrectly received data unit arrived.
Receiver-driven – Automatic Repeat Request (ARQ) as an explicit request.
Sender-driven – Timer-based request due to missing acknowledgement.
Sender stores data for retransmission purposes

- Buffer and memory requirements for high-speed multimedia data flows may be huge.

Parity Check

- Form of Check summing
- Message consists of a sequence of characters with m bit
- We add a m+1 or m+2 bit to it which is the parity
    - It helps to select per byte the information
- We have a cross parity and a long parity
- Help to identify if there are errors in the storing of those bits

To transfer sent a sequence of characters and parity
(BCC: Block Check Character):



- Typically you use an XOR - SUM is even or odd
- EXAMPLE of ASCII 7 bit code
    - 8th bit not used - in practice used for parity check
    - So we can detect if a single bit transfer occurred
    - Single and double errors detectable
    - Single errors correctable

Cyclic Redundancy Check (CRC)
Adds k bit of redundant information to an n-bit message

- Typically that message m is represented as an n-1 bit degree polynomial

M="10011010" corresponds to $M(x) = x^7 + x^4 + x^3 + x^1$

- Tracking the one positions

We then create a divisor polynomial of degree **k**

$$C(x) = x^3 + x^2 + 1$$

- If the transmitted polynomial (depending on the message polynomial and divisor polynomial) is easily divisible by C(x)
- And if the result of dividing the transmitted polynomial P(X) with the divisor C(x) = P(X) + E(X)
- Where E(X) = 0 - implying no errors
- The recipient divides (P(x) + E(x)) by C(x)

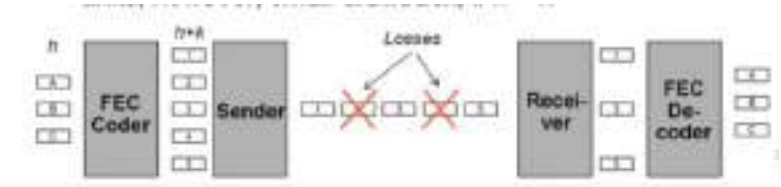
    - Remainder will be zero in only two cases
        - If E(x) was zero - no error
        - E is exactly divisible by C(x)
        - Therefore choose C(x) to make E divisible by C(x) extremely rare
- CRC have likelihoods and don't provide a 100% guarantee

Outcomes:

- Detects all single-bit errors, as long as the xk and x0 terms have non-zero coefficients
- Detects all double-bit errors, as long as C(x) has a factor with at least three terms
- Detects any odd number of errors, as long as C(x) contains the factor (x + 1)
- Detects any "burst" error (i.e., sequence of consecutive erroneous bits) for which the length of the burst is less than k bits
- Detects most burst errors of larger than k bits can also be detected

Forward Error Correction (FEC)

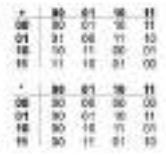
- If a retransmission makes no sense - audio, slides (because the retransmission will come out to late)
- FEC creates out of **h** original data units **h+k** redundant ones applying a **dedicated coding scheme**
- Any h number of data units can help to decode the original content
- Reduces delay in an error case
- But uses additional bandwidth for redundant data units, however small drawback h>>k



Example:

– Send 2 data units (A and B) of length 2 bit each
– Define *redundant data units* C and D:
  - C = A + B
  - D = A + 10*B
– A=10 and B=00 generates C=10 and D=10
  - Receive B and C: A = C0+10 = 10

Arrived	Calculation for A	Calculation for B
A, B	–	–
A, C	–	A+C
A, D	–	11*(A+D)
B, C	B+C	–
B, D	D+10*B	–
C, D	11*C+10*D	10*(C+D)



Go Back - N Advantages and drawbacks

- simple , efficient implementation
- Error indicating acknowledgments arrive on RTT later
    - Loss of bandwidth as high as twice the RTT
    - No buffering at receiver side

Selective scheme's advantages and drawbacks

- Complex algorithms
- Only sending back the incorrect data does provide better bandwidth
- In high-speed networks: Sequenced delivery of data units requires huge buffers, since twice the RTT indicates the delay of incorrect data

9 - Shared Direct List

Problem with shared media access is how do you coordinate independent senders and receivers using the same medium?

Some solutions

- Distributed or centralized coordination
- Preallocation of medium to each sender - Synchronous Time Division Access (TDMA)
- Allocation of medium on demand
- Constant frame lengths - call based approaches - Asynchronous Transfer Mode
- Variable frame lengths

Lan Properties

Access data on-demand on local network 10m - 2.5 km, 10-150 computers per connected LAN

- Cheap
- Build and operate it yourself
- Easily extensible

Contention-based Access Methods

Frames need to be transmitted in a time-based multiplexing scheme on a physical medium

- At time t only one station is allowed to send
- Collisions have to be detected

Coordination without centralized control

- Media states: free, busy, contended
- Collision (contention) is an interference during the sending operation
- Sending time is based on local information only
- Local observations cannot determine a media state, since
- electromagnetic waves travel with limited speed

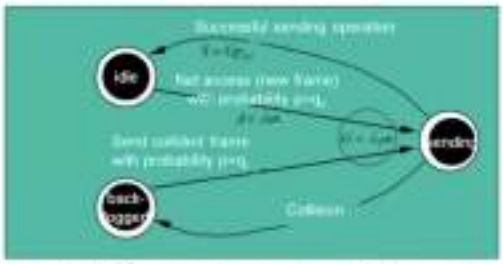
Aloha

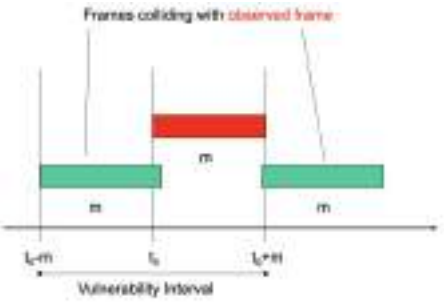
Radio transmission between links

```
/* send one frame */
repeat
    send(data)
    r=receive()
    if r <> ack then
        wait_random_time
    end
until r=ack
```

- Added Error Checksum correction
- Ignores errored transmissions
- You need to send an acknowledgement
- There is a time periodicity for when you can send packages
- If there is a collision each package will be retransmissioned separately order is based on luck
- Performance is determined by
    - Number of stations
    - Traffic load
    - Traffic characteristics
    - Parameters of medium
- Fairness is given but depends on configuration. Fair average/maximum access time

State Diagram for Aloha Station





Vulnerability Interval - Plus Minus m duration respective to t0

- Frames colliding with observed frame may only collide when they are back to back

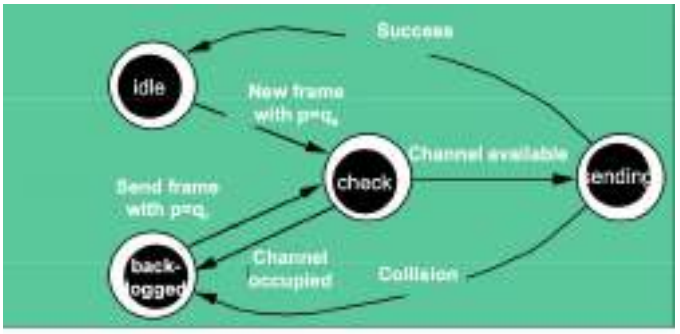
With LAN Systems you need to assume a cooperative behaviour

Carrier Sense Multiple Access

Solving the medium access problem by first checking for availability before sending data.

- Collisions are still possible

```
/* send one frame */
Repeat
    wait_channel_available()
    send(data)
    r=receive()
    if r <> ack then
        wait_random_time
    end
until r=ack
```

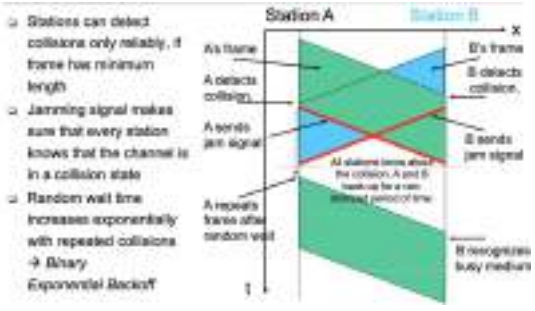


The "check" state determines the "CS" of the CSMA access scheme.

We throw a dice to ensure fairness with a probability p, the smaller the p the less aggressive we are

1. Always Listen into the channel
    a. 1 Persistent CSMA - If a channel is found you always send. If it's occupied you wait until it's free
    b. Non-persistent - if it's free send, otherwise wait random time and retest
    c. P-persistent - send with probability p, wait till free and send with probability p

With Collision Detection




```
/* send one frame */
t=1
Repeat
    wait_channel_available()
    r=monitor_while_sending(data)
    if r = collision then
        wait_random_time(t)
        t=2*t
    end
until r=success
```

- Jam Signal - dedicated bit sequence for a certain time unit

If the line is idle:
- Send immediately
- Messages of size 1500 byte

If line is busy:
- Wait until idle and transmit immediately
- (1-persistent)

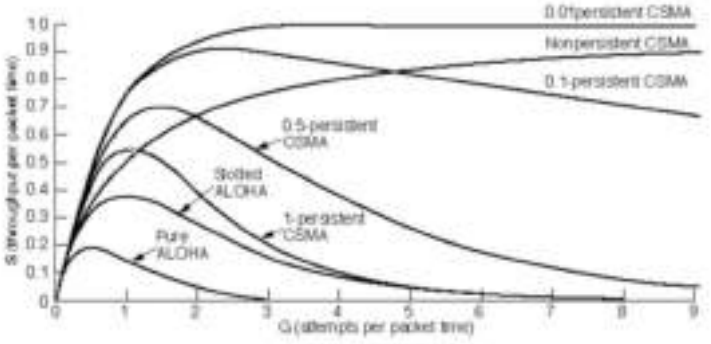
If there's a collision
- Jam for 512 bits, then stop transmitting
- A frame is 64 byte length
    - Header plus 64 byte of data

Delay & Try again - Exponential back-off
- First p calc for 0 -1, then 0 -3, 0 - 7 …..

Performance Tuning of CSMA/CD
The maximum throughput of CSMA-based systems is roughly indirectly proportional to ß
- B should be <= 0.01 to receive a good performance

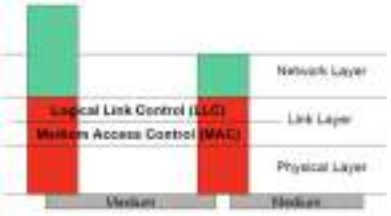
$$\beta = \tau / m = (\tau * C) / L$$

τ: Propagation delay [s]
m: Frame length [s]
L: Frame length [bit]
C: Transmission rate [bit/s]



If you need more attempts to get through without keeping the throughput high then the system isn't operationally secure.

Ethernet
- Developed in Xerox Parc
- 10, 100, 1000 Mbits/s variants
- 250 500 1000 lengths
- Medium part of Link and Physical Layer



Mac Layer
- Connectionless, unreliable (can get lost)
- Unicast, multicast, and broadcast addressing (doesn't prevent from listening, just a flag to pay attention)
- Frame Check Sequence checked - Cyclic redundancy check at hand so that data is error conform

LLC Layer (Not used as IP layer is placed on top of mac layer)
- Type 1: connection-less, unreliable, all addressing modes
- Type 2: connection-oriented, reliable, flow control
- Type 3: connectionless request, response type of service

How does the MAC work?

CSMA/CD Frame Format

Preamble:	Bit synchronization
SFD:	Byte synchronization (Start of Frame Delimiter)
DA:	Destination address
SA:	Source address
Length:	Length of payload
Payload:	Upper layer frame
PAD:	To fill up a short frame (padding)
FCS:	32-Bit CRC for error detection (Frame Check Sequence)

- Preamble: You need to make sure on the receiving side that your clocks are able to recover that signal

Ethernet MAC Addresses

- Every network capable device has a unique 48 bit MAC address
- Assigned by vendor
- All adapters in local network will receive all the frames and accept them passem them to the host depending on:
    - Unicast address
    - Broadcast address
    - Mutlicast address

Ethernet History & Evolution

Ethernet Today

- Likelihood of collisions is decreased due to the device and host being connected directly and alone
- Fast Ethernet - 100 mBits/s, CSMA/CD Algorithm, 100 m
- Gigabit Ethernet - CSMA/CD not used in Ethernet anymore

Nowadays Gigabit Ethernet Physical Layer

- You encode with a coding scheme that only adds 25% extra information
- Clock needs to regenerate with frames coming in

10 Gigabit Ethernet backwards compatible

- Full duplex mode -> no more Ethernet only Marketing

Token Ring Networks

- Frames flow in one direction upstream and downstream
- Bit pattern (token) rotates around ring
- Host first captures token and then allowed to transmit it further
- Hosts release the token after done transmitting - Immediate release, Delayed release
- Remove host’s frame when it comes back

You can have dual rings for reliability and directions, if a node fails you can reroute the traffic

- Each station creates a delay of 50 ms
- Max 500 stations

FDDI Access Algorithm

THT Token Holding Time - determines how long a station can hold the token

TRT Token Rotation Time - how long the token takes to traverse the ring

- TRT <= Active Nodes * THT + Ring Latency

TTRT Target Token Rotation Time - determines the agreed upper bound on TRT

- Each node send TRT
    - If TRT > TTRT - token is late so don' continue sending
    - TRT < TTRT - oken is early so send data
- Worst Case: 2x TTRT between seeing the token -

Token Management

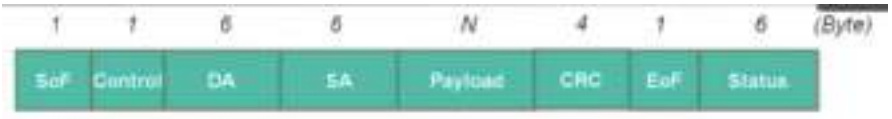
Tokens get lost:

- No token when initializing ring
- Bit error corrupts token pattern
- Node holding token crashes

Generate a new token (and agreeing on TTRT) when joining a ring or seeing a failure

- Each node send a claim frame for a new TTRT - I want to be the one who sets the timing
- Upon receiving a clam fraim you update the bid and forward it
- If the claim frame makes it all the way around the ring
    - This bid was the lowest
    - Everyone knows the TTRT
    - The hosts inserts new token which is then passed along and used

FDDI Frame Format - Token Ring Format



Control Field

- 1st bit: asynchronous (0) versus synchronous (1) data
- 2nd bit: 16-bit (0) versus 48-bit (1) addresses
- Last 6 bits: de-multiplexing key:
    - Includes reserved patterns for token and claim frame

Status Field

- From receiver back to sender
- Error in frame
- Address recognized
- Frame accepted (flow control)

10 - Packet Switching Networking

Switching - finding a temporary path across a network from sender and receiver for a package

This is done by connecting switching centers

Switching Centers should maintain the state of the switched connections

Connectionless switching in the internet is handled by **routing**

Signaling - is the terminology defining the communication that is needed for the switching to work

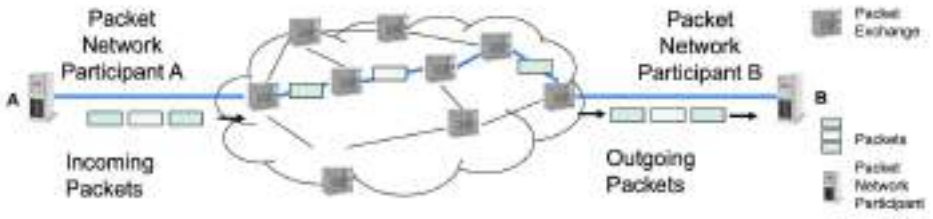
- DNS system
- End-systems and network
- Intermediate systems within the network

Message Switching

- Implemented on top of **packet switching** or **circuit switching** networks.
- A message is received completely before being forwarded (Store-and-forward)
- Header - source, destination and length information
- Message Text

Virtual Circuit (VC) Switching

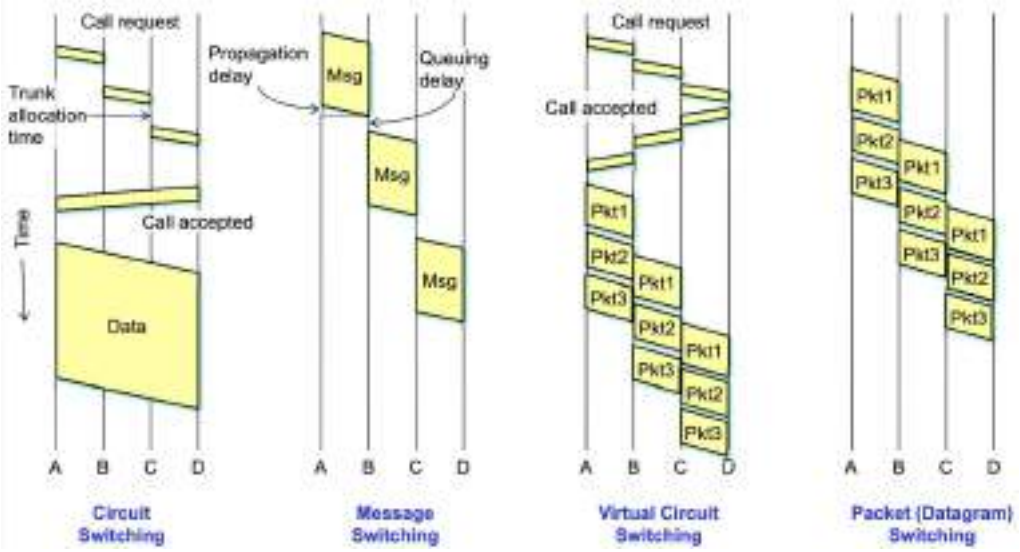
- An intermediate step between circuit switching and message switching.
- Enables a transmission over packet-switched networks as if there is a physical link between sender and receiver
- VC adheres to a signaling protocol where:
    - All packets follow the same path
    - All packets contain a virtual circuit ID
    - Path context available in all intermediate nodes



Packet (Datagram) Switching

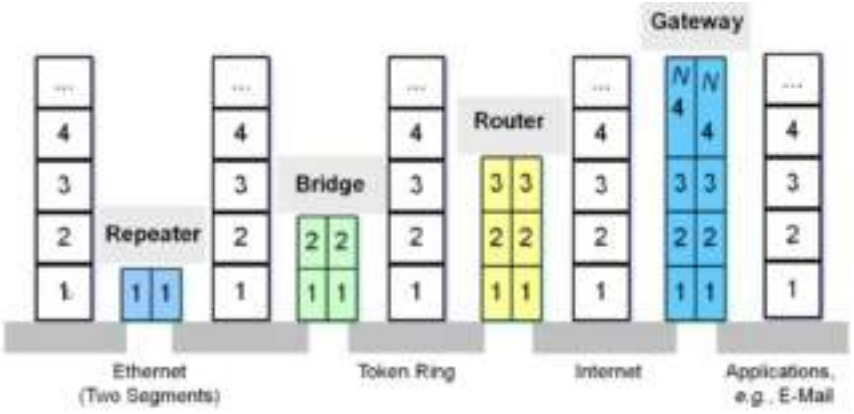
- The data you're sending has to be cut and sent in individual packets
- Packets have a limited length
- Packet length is variable
- Packets are received completely then sent
- Header contains switching and length information (so you know when to stop receiving)
    - Switching Info: source/destination address or virtual circuit ID
- Example: IP Protocol

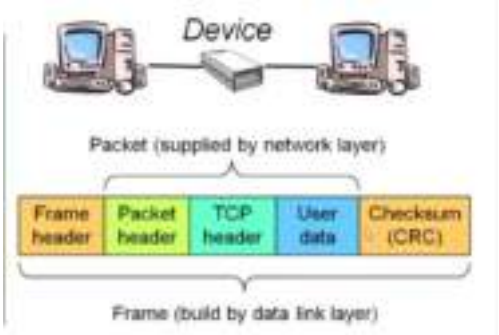
Comparison



Internetworking

- Set of subnetworks that are interconnected
- How do the intermediate system look like? - There are 4 types





Layer	Device
Application	Application gateway, e.g., mail forwarder, firewall
Transport	Transport gateway, firewall
Network	Router
Data link (MAC)	Bridge, switch
Physical	Repeater, hub

- For each corresponding layer we extend the header
- Checksum at the end
- After the data link layer builds the frame it's sent to the physical part

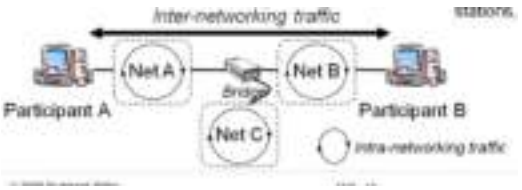
Repeater helps coupling media and reamplifying a signal

- They don't store anything
- They take and amplify
- Or They take and transform it
- They are very dumb - they just see bits
- Can extends LAN reachability
- Can change medium ex: copper to fiber

- Simple & cheap due to no data processing
- No filtering - all signals are forwarded
- No enhancement of computer network

Hub - A repeater with multiple ports
- layer 1
- Star topology - separate link to each device from hub
- It can aggregate multiple connections and forward as one connection
- No improved throughput

Bridges - devices coupling layer 2 networks



- Layer 2
- Require MAC address per port
- If the two connecting networks are of same type a bridge connects them in a **non-translating** manner
- If the two connecting networks are not of the same type a bridge connects them in a **translating** manner
- Bridge connects independent networks
- Is an interconnection device - forwards frame
- No need for additional frame headers
- Enhance network by having subnetworks be by themselves but able to interconnect

There are 4 types of bridges:
- Local/Remote
- 2 Port / Multiport - interconnects two or multiple networks
- Transparent/Source Routing - Bridge invisible/visible to end-systems
- Translating/Non Translating - Coupling of LANs of different/same MAC type

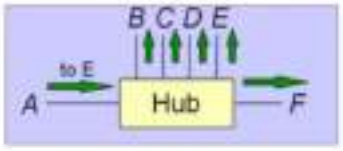
MAC Addresses

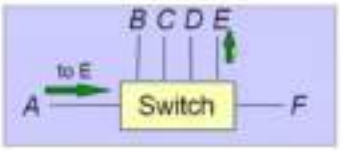
Medium Access Control unique identifier
- Any port of a communicating device is equipped with a unique identifier assigned to all network interfaces
- 48 bit address, 281 trillion options

Bridge Facts
- Any bridge holds as many MAC addresses as the have ports
- Forwarding decisions are done in the bridge
- Bridge needs to learn location of hosts by filtering the frames that pass through it
    - For Optimization
    - Using the source address
- Problem if you have multiple bridges in network -> leading to cycling and redundant paths
    - Solution make bridges smarter with **Spanning Tree Algorithm** ,....

LAN Switch
- Layer 2
- Powerful transparent bridge
- With many ports
- In charge of frame forwarding

Comparison: LAN Switches vs. Hubs
- Work at a similar location but in different layers
- Switches require to read the data to forwards





LAN Switch	**Hub**
Store and forward (small delay)	Cut through
Blocking	Non Blocking Architecture
Half duplex	Full duplex modes

CSMA/CD not used for full-duplex point to point connections

Spanning Tree Algorithm

Internet Design Principles

A lot of services and networks that are all separate

- Services are provided independent of the network
- Service providers are usually distinct from network providers

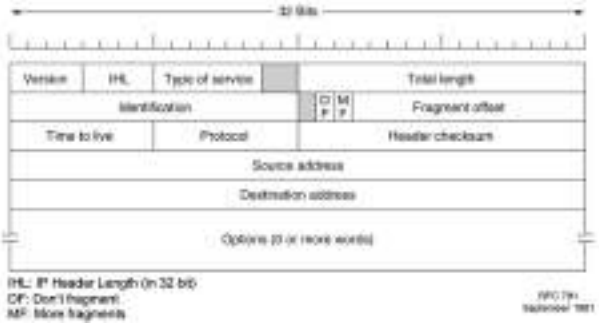
E2e architecture, end to end principle, network provides **best effort packet delivery (routing)**

Routers interconnect network via standardized protocol IP

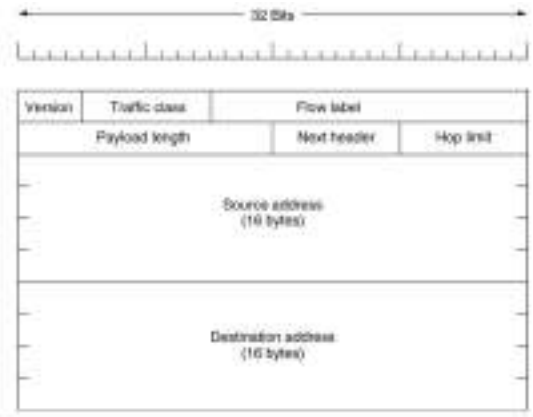
Processing happens at the edge (hosts) keep routers easy.

Scalability of distributed design and control is essential

IPv4 protocol



IPv6 Format - larger address space



IPv4 Address Structure
Bit address is split into groups of 8 raging form 0 to 255

IPv4 Forwarding & Routing

- **Forwarding** - the act of making a routing decision in a device
- We select the output port based on the destination address via the routing table
- **Routing**: Process by which the **routing table** is built
- If the destination is in the same subnetwork - Map IP destination address to MAC address and send directly

If destination outside:

- You find someone to help you: the router
- Identify matching router's IP address
- Map to routers MAC address
- Send to router

IP Routing Table

- It requires configuration based on the routing protocol
- Hosts IP addresses
- Network masks for all network interfaces being used
- It itself has a default IP address (additional ones if required)

The Host and the router maintain a **single routing table**

Destination - IP Address you want to ping/fetch
Gateway: the next IP address to hop to (router or device)
Netif: Network Interface Card locally maintaining that table

This knowledge is built overtime, if there's no information a default neighbouring device is being talked to.

This IP layer need to be matched to the underlying layer 2 - MAC address

IP address is assigned based on your infrastructure , where you are

We need to find a translation from IP address to MAC addresses

- Cache for efficiency
- Determine destination host or next hop router (leave task to next layer 3 router)
- How to do this: ARP

Address Resolution Protocol (ARP)

- Way to create a mapping table on the fly for IP -> MAC addresses
- You send a ARP Request for your IP on a local network
    - If the device is active it will answer the request coming in from the bus line

Routing Algorithms & Classification

- Topology (static or dynamic), it's always dynamic after a certain time interval
- Costs - traversing a link
- Operation of routing algorithm
    - Centralized - Big Brother View
    - Decentralized - useful for big networks
        - Non Adaptive - routing tables are constant
        - Adaptive - routing depends on network status

Decentralized Routing implemented by interacting autonomous routers exchanging information on control and topology

- High fault tolerance
- Used in smaller and larger networks

Flooding

- Simple - No routing table used
- Distributed
- Non-adaptive
- Every incoming packet of information forwarded on every link, except where it came from
- Problem: large flow of data traffic, redundant info
    - At the destination discard copies coming in after having received the first package
    - If the destination doesn't exist you continue flooding forever and overload the system
        - Therefore you need a time-to-live field to stop at some point in time, reduced by one at every step

Minimum Spanning Tree - connect all nodes in graph such that the sum of the link costs is minimized
Shortest Path Tree - SPT such that every path from node is the shortest path

Bellman Ford Algorithm
Dijkstra's Shortest Path Algorithm (greedy)

Distance Vector Routing

- This is the approach - using the dijkstra algorithm
- Each router has a set of triples - (destination, cost, nexthop)
- Everytime a routers table changes it triggers an update to the neighbouring routers
- If the cost is smaller we replace the next hop accordingly and calculate the new cost

Problem with Distance Vector Routing

Routers only knows the table and not the connectivity of the entire graph

- Therefore it could start routing paths through it self
- If such a link fails routers start to count costs to infinity
- Implement a threshold, the larger the threshold the slower the convergence, the smaller the threshold the smaller the network diameter
- To solve this: Link State Routing

Link State Routing
Routers distribute the local view (the graph topology) to other routers.
Each router contains a complete view of the topology.

Algorithm:
- Detect all neighbours and ping them
- Measure latency to neighbour to calculate cost
- Generate a Link State Package:
    - Sender, list of neighbours and latencies, timestamp
    - Generate periodically or triggered by an event
- Send Link State Package to all neighbours
    - Updating, Deleting old data
- Failure Detection if neighbours don't answer to ping
- Then run Dijkstra for shortest path

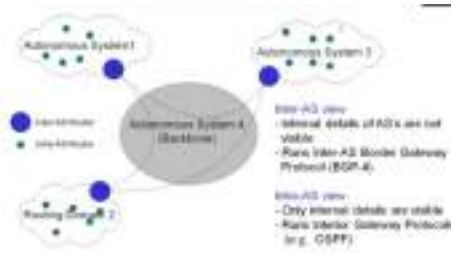
Advantages - fast convergence, no count to infinity problem, high stability routers having the topology

Disadvantages - storing topology is larger than vector, complexer implementation than distance vector

Scalability Issues in the internet
- Scalability of many users
- Scalability of all the subnetworks
- Solved by
    - Having flexible IP addresses with sub-netting
    - IPv6 for more addresses
    - Hierarchical and autonomous routing domains with centralized control

Autonomous System (AS)
Every AS has a unique identifier
- Offered to Universities, companies, backbone network

Types:
Stub AS - leave of the tree only a single connection - only for local traffic
Multi-homed - connects to another AS - no transit traffic
Transit AS - connections to multiple AS - for local and transit traffic



Border Gateway Protocol
- Is the recommended inter-AS routing protocol
- BGP is a **path vector**
    - **Distance vector** protocol with **loop protection**
    - It aggregates and distributes the reachability of a router (including the traversal path through all the AS's)
    - A lot of attributes are added

        - AS path, origin of route, next hop towards destination, multi-exit discrimination in case of parallel paths into an AS
    - Initialized once -> then updated
    - BGP uses TCP for reliable transport of routing updates

This wraps up Layer 3 and its elements

11 - End to End Protocols

Hop by hop vs.e2e Services

Hop by hop at its best effort:

- Messages can be dropped - router reaches hop limit
- Re-orders messages
- Delivers duplicate copies
- Limits messages to a finite size
- Delivers messages after some arbitrary delay

End 2 End Networks:

- Guarantee message delivery
- Deliver in correct order
- Deliver at most one copy
- Support custom large messages
- Allows the receiver to ask the sender to please not overload him with data
- It's supported on multiple applications running simultaneously on each host

Link Layer Protocol

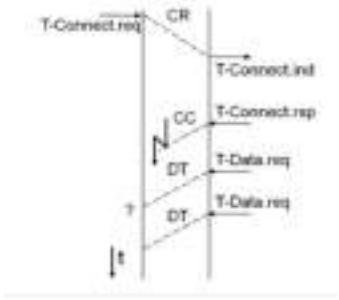
- Entities are connected via a single point to point link
- Mostly fixed and small Round Trip Times
- No change of package sequence - because you have neighbouring elements it is point to point
- Physical Link limits the sending rate

Transport Protocols

- Entities aren't neighbour systems - they can be anywhere
- Variable and larger RTT
    - -> larger bandwidth-delay product
- Order delivery isn't guaranteed
- Bottlenecks may exist on any intermediate link

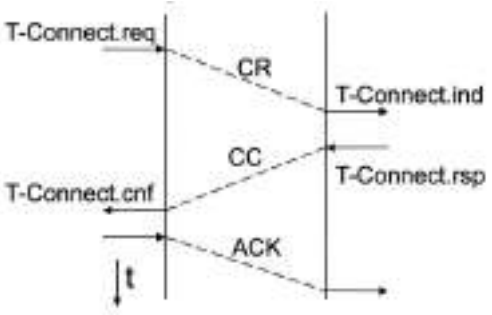
Transport Layer Latencies - 3 Types
Connection Setup Latency - Initial Ping setting up the connection
Transmission Delay - Transmission of a message just takes time
Connection Teardown Latency - It takes time for sender to receive teardown ping

Loss of a connection request might happen - can be solved by a **3-way-handshake**

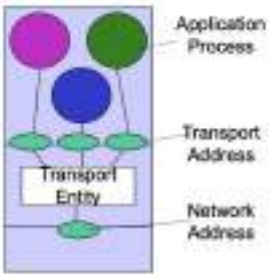


3-way-handshake

- Connection established as soon as CR and CC have been acknowledged
- Requires an additional message



Internet Transport Layer Addressing



We require a full **transport address** containing these globally unique values

- IP Address
- Port number - equals and end point of a communication system

There are reserved port numbers for popular services:

13: NTP (Network Time Protocol)
20: FTP (File Transfer Protocol) Data
21: FTP Control data
25: SMTP (Simple Mail Transfer Protocol)
53: DNS (Domain Name Service)
80: HTTP (Hyper Text Transfer Protocol)
119: NNTP (Network News Transfer Protocol)

UDP Protocol

- Is connection-less
- Unreliable and unordered datagram service
- Has multicast abilities
- No Flow Control
- Has an extended version of IP on layer 4

Transmission Control Protocol

- Between two sockets in full duplex mode
- Reliable - 3 way handshake, ordered delivery
- Error control: Sequence numbering, checksum, retransmissions
- Flow control: prevents sender from bombarding receiver
- Congestion control: prevents sender from blocking network

Transmission in CP is byte stream driven you send a byte a byte ….

- Breaks into segments and send via IP
- Segmenting Size according to MTU (maximum Transmission Unit)
- Receiving process reads some number of bytes

TCP Header

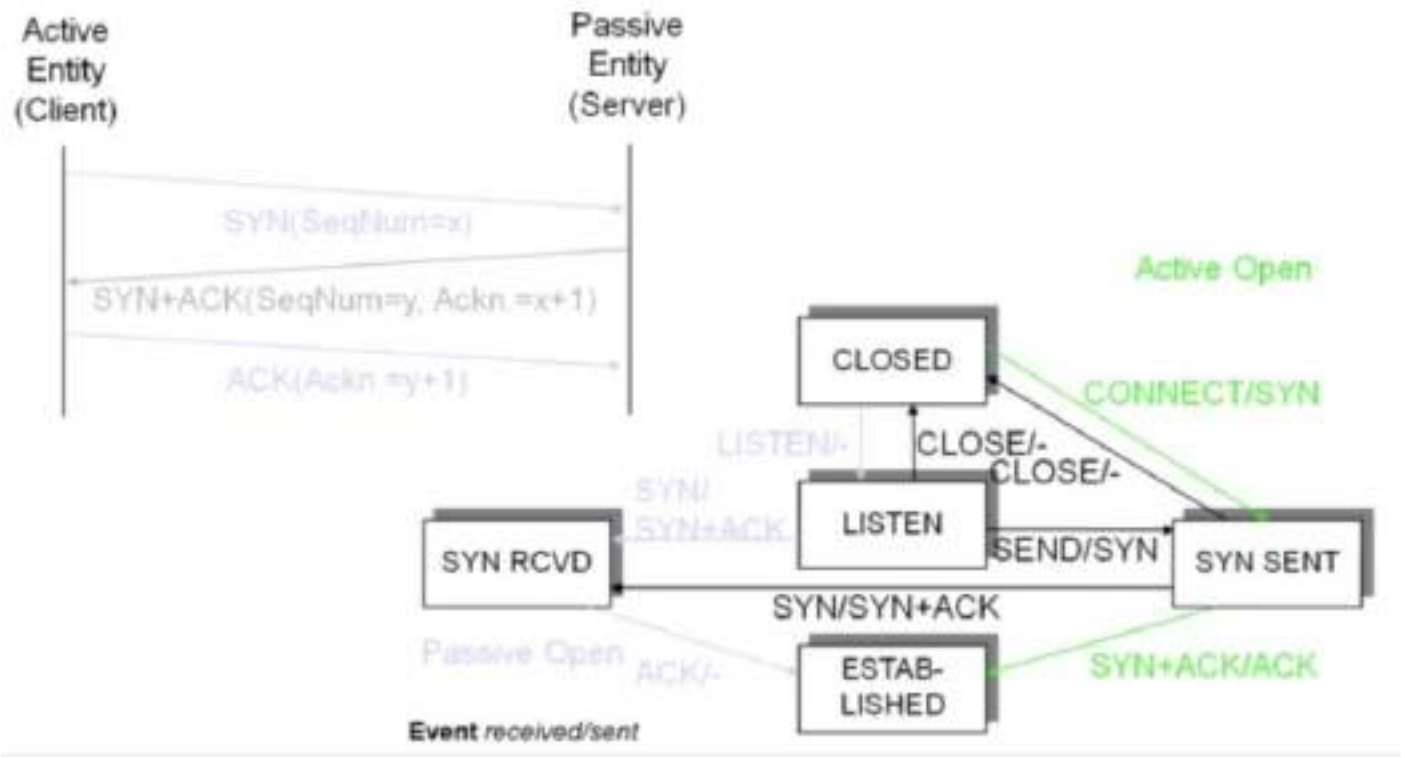
- Source and destination port
  - Part of the transport layer address
- Sequence number and acknowledgement
  - Sending and receiving sequence number
- TCP header length (HdrLen)
  - Number of 32 bit words in header
- Advertised window
  - Sending window of flow control
- Urgent pointer
  - Pointer to important data

- Flags
  - URG
    - Set to 1 in case the Urgent Pointer is used
  - SYN
    - Used at connection set-up
  - ACK
    - Signals the validity of the current acknowledgement field in header
  - FIN
    - Information that sender will not sent further data
  - RST
    - Restart of a connection
  - EOM or PSH
    - Information that the end of the message has been reached (End of Message or Push Flag)

TCP Connection Setup

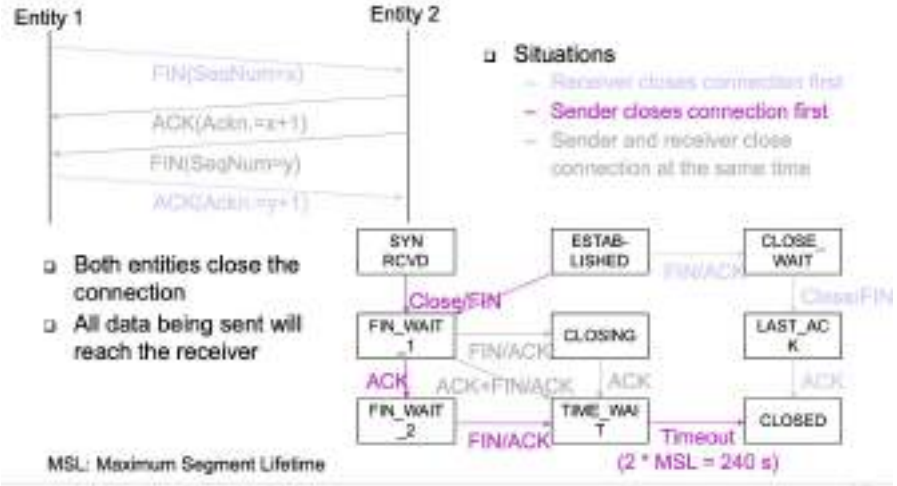
- Opening of a socket
  - Active mode - request a TCP connection from a specific socket (connect)
  - Passive mode - user informs TCP being able to accept and incoming connection (listen/accept)
- Functional View
  - 3 way handshake

- Exchange of initial sequence number to avoid conflicts with old connections on the same port and old packages in the network



- TCP you always send the next bite you'll expect

TCP Connection Teardown



- Should always have a teardown
- Erase their statemachine regarding to this process

TCP Error Detection and Correction

Piggybacking - 16 bit long window size (how many byte can be in transit) but if more data would need to be transferred the receiver window can be upgraded by a factor of 2 to 2^32 byte

Checksumming - Reliably securing TCP header, payload and TCP pseudo header

- By using the IP source and destination address, IP protocol field (6), and TCP segment size

Acknowledgement - help to identify that actions between sender and receiver are in place

- Cumulative (standard, Ack=y+1, everything up to y has been received)
- or selective (optional)

TCP Retransmission
How can we ensure that lost packages are retransmitted

Go Back N (Standard)
- Acknowledgements get sent for **n** meaning data till **n-1** has been received
- If now the number getting returned to the sender is too low he resends from where the receiver last acknowledged
- If there's no retransmission coming back we'll retransmit based on a timer (based on network type)
- During Connection Setup rules can be defined
    - Use a NAK that just sends erroneous segments

Selective ACKs
SACK acknowledge single, well received segments in a system that is rather sparsely sending data

Retransmission Timer
Starts if the timer timed out and no ACK was received it resends the segment and resets the timer (quite costly)

RTT Calculation

```
Diff = Sample_RTT - Estimated_RTT
Estimated_RTT = Estimated_RTT +
  (δ x Deviation = Deviation + δ(|Diff|- Deviation)
```

where δ is a fraction between 0 and 1

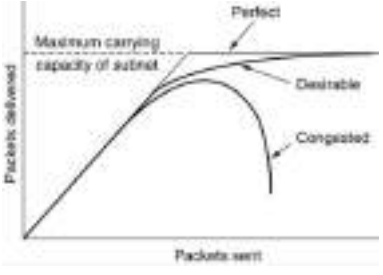
Consider variance when setting timeout value

- Time_Out = μ x Estimated_RTT + ϕ x Deviation
  where μ = 1 and ϕ = 4

- This information helps you have the retransmission timer should be configured
- How well this works depends on the clock quality of a computer
- Timeout mechanism is important for congestion control

TCP Flow Control
- Uses Flow Control with the sliding window mechanism
- ACK flag acknowledges receipt of all bytes numbered with a smaller number than the sequence number received
- Advertised Window determines the number of additional bytes, the receiver is able to accept
- Receiver allows sender to sent data up to the amount of:
    - Acknowledgement + Advertised Window - you don't want to overload the receiver

Internet Traffic Problem
- A Congestion collapse - network fully loaded, throughput was 0 - only retransmissions were observed

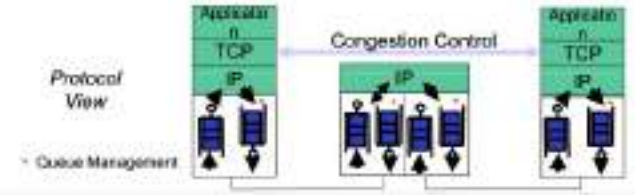


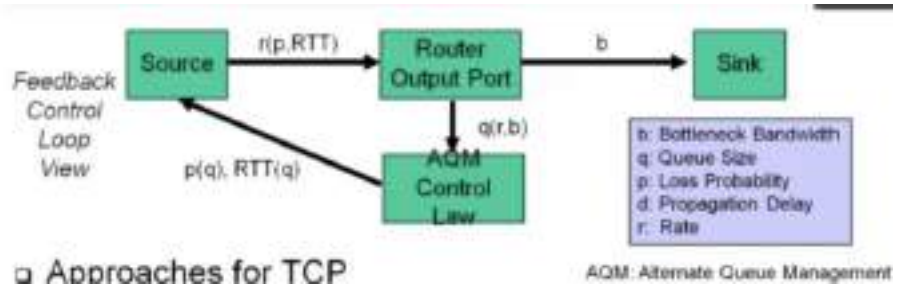
Congestion Control Principles & Prevention
Monitor the system, detect congestion, ensure information goes to a place where action can be taken, adjust system to correct the problem
- If this happens - bad -> Prevention is key

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

TCP Congestion Protocol

- Looks at the queuing situation in routers
- Congestion leads to retransmission in the transport layer
- This leads to an increase in congestion
- Congestion control needs to adapt sending rates
- Queue management within routers need to drop packets
    - Impacts negatively the communication but helps reduce load





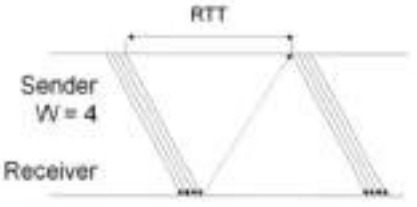
We have a feedback loop here that either gives the source or the AQM control on what to change

- We calculate the p and RTT based on AQM and these values are then used to calculate a new r, ….

Window based Congestion Control

- Inherits from our flow control window idea
- W is a window defining how many packets are allowed to be sent when receiving an ACK
- Push wait push wait

Advantage - small overhead due to ACKs
Drawbacks - bursty nature, loss of ACKs



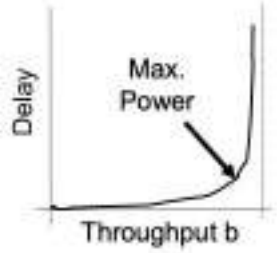
Window Sliding Control

- ACK step = 1

Advantage - not bursty, loss of ACKs is okay because I only use one
Drawbacks - large overhead due to many ACKs, we create a dependency of the window size and if a transmission error happened

- Control of transmission volume, bytes or packets
- Effective rate = W(t) / RTT(t)
  - W(t) = f(p(t))
  - RTT(t) = q(t) + 2*d
- Optimal Window: W = b * 2d
  - Optimal link utilization:
    - Effective rate = b*(2d/2d) = b
  - No queue at bottleneck:
    - Optimal network performance (power)

b: Bottleneck Bandwidth
q: Queue Size
p: Loss Probability
d: Propagation Delay



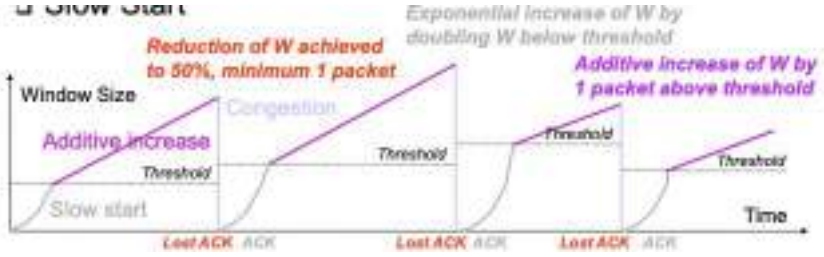
Power = Throughput / Delay

- Window size depends on time
- RTT depends on time
- Loss probability depends on time
- RTT depends on queues size + 2 * propagation deploy

Adaptation of TCP Window Size

Slow Start Algorithm

- Duplication of the window size every time unit until a threshold is reached then additively
- So we start exponentially till threshold and then +=1
- If you run into a congestion you start another slow start with a higher threshold (half of the window size at the time the ACK was lost)



Additive Increase Multiplicative Decrease (AIMD)

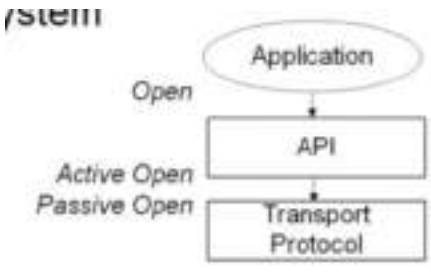
- Jigsaw looks
- Not used

Practical TCP Considerations

- Due to big transmission sizes the **Maximum Segment Life** may see an overflow due to reaching the max **TCP sequence numbers**

Protocol Implementations and Interfaces

- On the transport layer you need to think about the applications
- We separate protocol implementations form protocol interfaces being exported
- Interface is called an API
- API is defined by the OS
- Specific API: sockets



Socket Calls

socket	Create a new communications end point
bind	Bind server address to socket
listen	Listen to incoming connections
accept	Blocking at passive open, creates a new socket for a server
connect	Active connection setup
send	Send
receive	Receive
close	Connection teardown

```
socket  (domain, type, protocol)
bind    (socket, address, addr_len)
listen  (socket, backlog)
accept  (socket, address, addr_len)
connect (socket, address, addr_len)
send    (socket, message, msg_len, flags)
recv    (socket, buffer, buf_len, flags)
close   (socket)
```

Creating a socket

```
int socket(int domain, int type, int protocol)
domain=PF_INET, PF_UNIX
type=SOCK_STREAM, SOCK_DGRAM
```

Passive open on server

```
int bind(int socket, struct sockaddr *address,
         int addr_len)
int listen(int socket, int backlog)
int accept(int socket, struct sockaddr *address,
           int *addr_len)
```

Active open on client

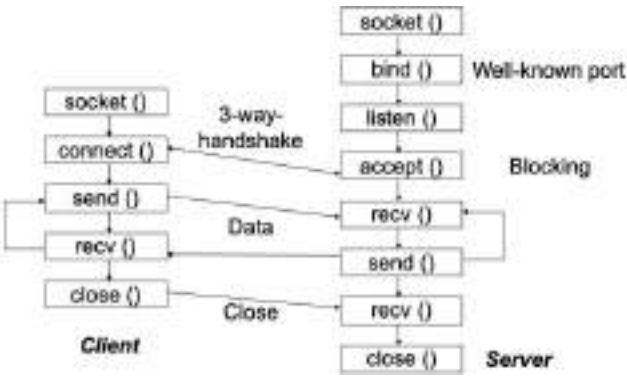
```
int connect(int socket, struct sockaddr *address,
            int addr_len)
```

Sending and receiving messages

```
int write(int socket, char *message, int msg_len, int flags)
int read(int socket, char *buffer, int buf_len, int flags)
```

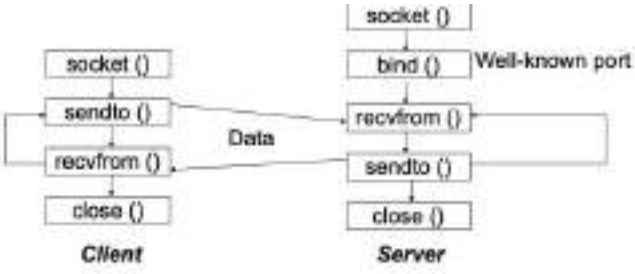
TCP Client and Server

- Use a 3 way handshake
- Client asks for a Connection (assuming he's the active guy)
- Server accepts
- ….



UDP Client and Server

- Simpler
- No state therefore simpler
-




```
recvfrom (socket, buffer, buf_len, flags, from, from_len)
sendto (socket, message, msg_len, flags, to, to_len)
```

The congestion window is flow control imposed by the sender, while the advertised window is flow control imposed by the receiver. The former is based on the sender's assessment of perceived network congestion; the latter is related to the amount of available buffer space at the receiver for this connection.

12 - Security Mechanisms and Protocols

Traditionally public networks were closed networks and maintained centrally - attacks weren't thought of as useful

Currently public networks are decentralized, especially the physical layer. Many interfaces to Thirdparties. Decentralized and anarchic.

Distributed Applications are vulnerable by definition

Vulnerability - a quality of a system that is an opportunity for misuse

Threat - a potential occurrence that can have an undesirable effect on the assets and resources of an IT system

Risk - Threat * Vulnerability || Likelihood * Impact

IT Security is a process of risk management

OWASP Risk Rating Methodology
Software life cycle

- Early stages
    - Identification of security concerns in the architecture or design using threat modeling
- Later stages
    - Code review
    - Penetration testing
- Operations
    - Some errors only appear in production

Steps:

1. Identify Risk
2. Factors for estimating likelihood of that risk
3. Determine factor of estimating technical and business impact
4. Deter-mining severity of risk at scale 0 … 9
5. Deciding on prioritized list what to fix
6. Customize risk rating model

Example Attacking the Data Transfer
Passive attack: Eavesdropping only, no change of data - threat of confidentiality
Active attack: Changing, Deletion, insertion - threat for confidentiality, integrity, authenticity

Example: Attacking Service Provisioning
Denial of Service (DoS) attacks of Web Servers

- Generate work for infrastructure in order to prevent from doing regular tasks
- SYN Flooding - TCP-SYN using spoofed IP addresses - server allocated resources for partially opened connections
- "Smurf" DoS attack: End System replies to ICMP (Internet Control Message Protocol) Echo Request messages - addresses are spoofed
- Distributed Denial of Service attacks (DDoS)

Threats - Examples/Countermeasures
Packet Snooper - reading of packet content -> Encryption

Packet Sniffer - reading source and destination addresses in the header -> Encapsulation of packet and encryption
Session Hijacking - gaining access & control to a multi message session -> Authentication
Date Tempering - like Session Hijacking but only part of the data is intercepted -> Authentication and encryption

Threats & Attacks

Intrusion and Data Loss
Intruder Classification

- Non tech person accessing information
- Internal Employees looking for information without authorization
- Mostly happens due to economic success
- Espionage, Military and industrial
- 50 % of attacks happen inside of a company

Data loss due to operation failures

- Sometimes it just happens
- Hard or Software errors
- Human errors

Major 7 Security Pillars
Authentication - enures partners involved in communications are who they claim to be
Authorization - ensure user has access to a service
Integrity - protects against the modification of a message along the transmission path
Privacy - defines the degree of publication of personal information and data
Confidentiality - protects data against eavesdroppers in com-channels only reaching authorized users
Non-repudiation - neither the sender or receiver can deny the transmission happened
Anti-replay protection - protects a receiver from the duplicated reception of an already authenticated and received message

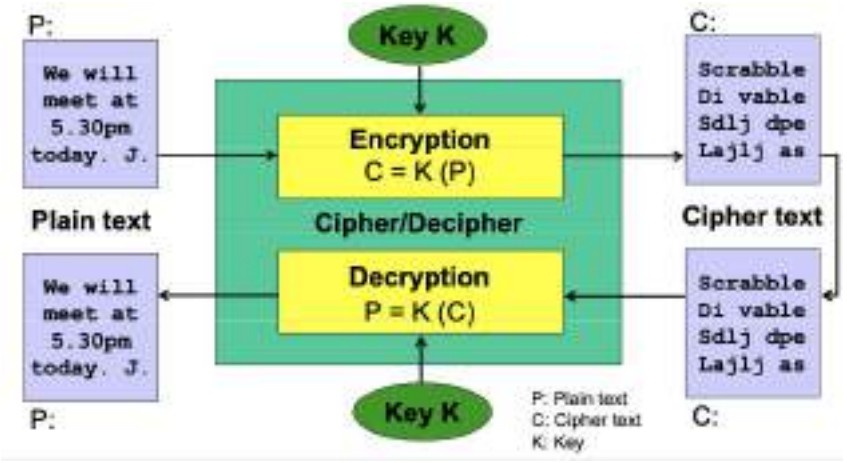
Additional Security Aspects
System security - entire system has to be protected by security mechanism and protocols
Anonymity - condition in which a person's identity is not known
Pseudonymity - condition in which a person has taken on an assumed identity
Auditing - process of collection unforged events and facts
Identity - set of characteristics by which a human is identifiable
Identity Management - Encompasses different mechanisms, password management, user provisioning, and access management.

- Maintains user access to services

- Single point of administration

Trust and Trusted Third Party - a tertiary party (entity, instance, administration) whom the other parties fully trust

Cryptography



Symmetric Cryptography

- Both entities owen a shared key
- Advantages:
  - Small overhead/calculations
  - Short keys
- Drawbacks
  - Key exchange is complicated

Asymmetric Cryptography

- Key pair of private/public parts
- Advantages
  - Public keys are easy to publish
- Drawbacks
  - Longer keys
  - Larger overhead/calculation
- Encryption is easy and publicly accessible
- Decryption hard unless your the intended recipient
- Minimum key length 1024 bit
- Secure Algorithm: RSA

Hybrid Approach (Symmetric and Asymmetric):

- User authentication and exchange of a session key which is public-key-based
- Establish a channel based on public key encryption - once
- Then you have a secure channel were you exchange the session key
- Used to encrypt and decrypt the data
- Session key changed every 10 min

Asymmetric Encryption Example

1. Both participants have private and public keys
2. Alice hands her public key to Bob / Or Bob finds her public key in a database
3. Bob Encrypt a document with her public key
4. Alice can decrypt the encrypted document with her private key

RSA

- RSA Function f maps plain text to ciphertext and the inverse of f ciphertext to plaintext

Function f need to adhere to these 5 properties

- F is one to one (uniquely invertible)
- F is easy to compute (encryption easy)
- f^-1 is difficult to compute (decryption difficult for the sender)
- F has a domain that is easy to sample from (Bob easily generates a key)
- Existence of an easy to compute function d of the input of f making computing f^-1 easy (Alice decrypts easily)

F is a **trapdoor function**

- You need a piece of information for it to work

$$f(x,e,p,q) = (x^e \bmod pq,\ pq,\ e),$$

- p, q are prime numbers
- x is plaintext
- x = x^e mod p*q is the ciphertext
- d(x,e,p,q)=e^-1 mod (p-1) * (q-1) - this is property 5
    - Or e*d*mod (p-1)*(q-1) = 1

F is the basis for the RSA Cryptosystem

- (e, pq) public encryption key (public key)
- (p, q, d) secret decryption key (secret key)

- Alice creates a private public key pair once
- Alice encrypts message with her private key
- Bob decrypt message with Alice's public key

RSA Example

- Take two very large prime numbers p,q
- Take their product N
- E is calculated 1 < e < (p-1) (q-1)
- D is calculated by d * e * mod (p-1)(q-1) = 1
- (e,N) is the public key
- Number triplet defines the private key (p,q,d)
- Plaintext P
- Ciphertext y = P^e mod N
- Ciphertext to plaintext = x = C^d mod N

Theoretical Security

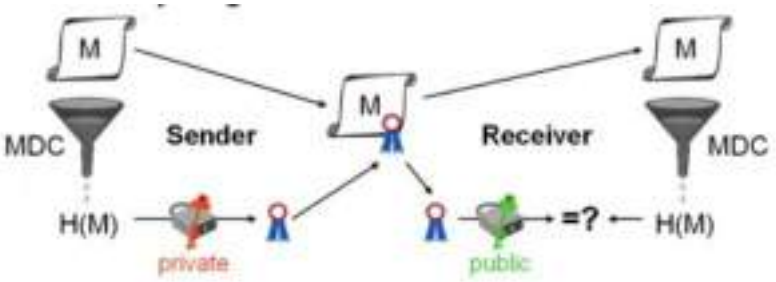
Prime number factoring can't be done in finite time, there are exponential algorithms but no polynomial ones (yet)

Signatures - Digital signatures are achieved by asymmetric encryption

Digital Signature

- Hash Value of message signed with private key of sender
- Receiver checks signature with public key of sender

- Guarantees the commitment

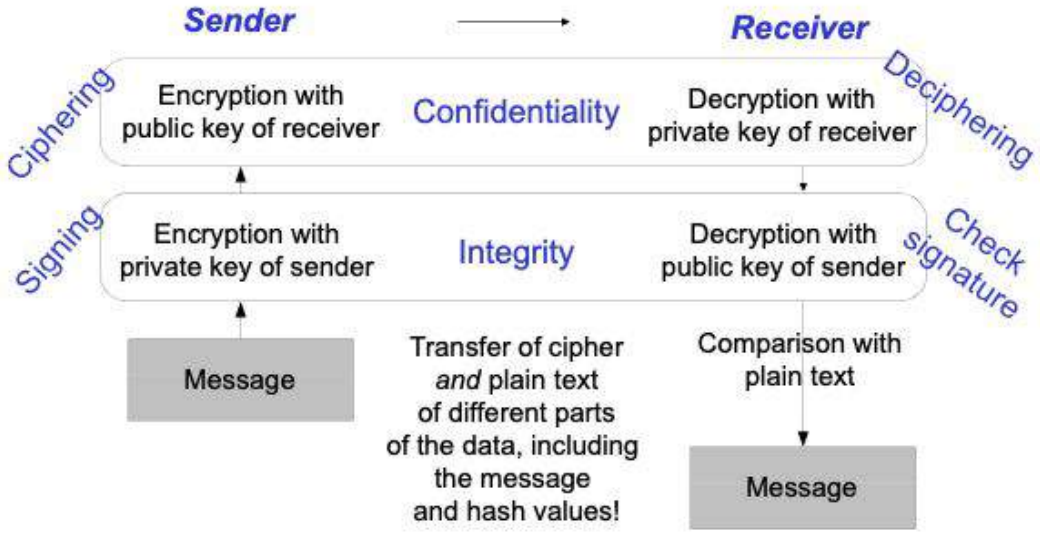


- M: Message
- MDC; Message Digest Code (crypto hash function)

Example Signature

1. Bob sends Alice his public key
2. Bob Signs the document
    a. Bob applies crypto hash function to document M
    b. Obtains a hash value HV1
    c. encrypts this hash value HV1 with his private key
3. Alice verifies the signature
    a. Applies same Hash Function to document M
    b. Obtains a hash value HV2
    c. Obtains the encrypted hash value HV1(enc)
    d. Decrypts HV1(enc) with Bobs Public Key
    e. Obtains HV1
    f. IFF HV1 and HV2 the signature on document M has been verified

Relationship between Ciphering and Signing



Certificates

- certify association between public key and person
    - Certified Authorities release certificates and function as a trusted third party
- Authentication of a sender for the singing process
    - Trusted mechanism to relate signature and signee
    - Checks to secure the signature not the confidentiality

A CA can generate and revoke a certificate

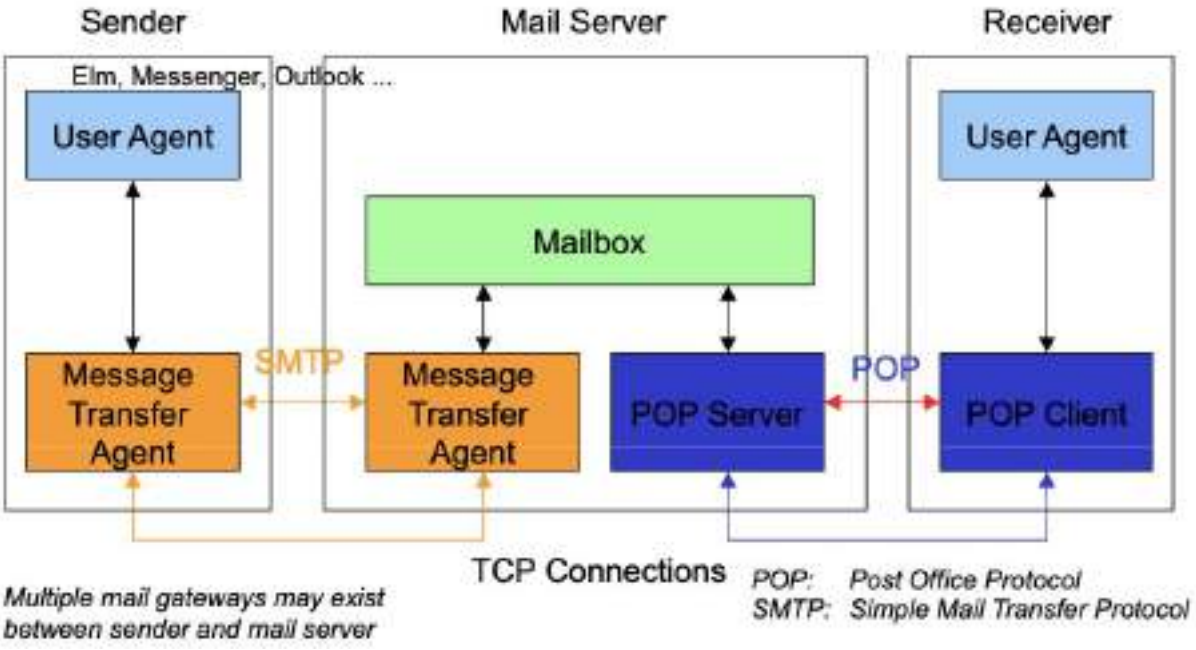
- If private key or CA was compromised
- Service the CA granted isn't used anymore

- CA may have faulty certificates

Therefore they add an Expiration time and keep track of revoked certificates in **Certificate Revocation List** (not expired)

13 - Applications Mechanism

Electronic Mail (E-mail)



- Sender runs a **User Agent** - your Mail Client (Outlook, …)
- Underneath that a **Message Transfer Agent** is talking from our local Machine the the Message Transfer Agent on the **Mail Server**
- **MailBox** contains all the data you've sent and received - maintained on the mail server
    - You might run a local copy
- **POP Protocol** helps you access the data in the Mailbox
    - You take a copy locally and it is then deleted on the mail server
- **IMAP** (popular) alternative to POP that helps syncing the local and server mailbox
- Facilitate through TCP Connections
- Mail Server is the endpoint of communications
- Reliability is only given for the user -> mailserver part not the user -> mailserver -> recipient part

Goal - international exchange of messages between humans

Major characteristics:

- Support of asynchronous sender and receiver behaviour
- Store-and-forward switching

Functions:

- Write E-Mails
- Transfer them to a destination
- Indication of success/errors cases or received messages
- Storing of messages

Simple Mail Transfer Protocol (SMTP)

- Direct TCP connection between mail server
- Able to send multiple messages within TCP connection
- TCP connection used for backward direction

Messages - contain ASCII text

- Max 64kByte
- Text or Multipurpose Internet Mail Extensions (MIME)

Type of Messages exchanged between client and server
- HELO: Introduction
- MAIL FROM: Information about sender
- RCPT TO: Information about receiver
- DATA: Sending messages
- QUIT: Finalization
- VRFY: Verification of user name
- EXPN: Expand, information about lists

Receiver acknowledges every message

SMTP-based Email Structure
Envelope
- Contains information to transport a message to the receiver
- Addressing based on DNS hadorn@acm.org
- Interpretation done by Message Transfer Agent (MTA)
    - Ensures data in envelope can be read - checks local operation

Header
- Subject, CC, BCC fields
- Interpretation of User Agent (UA)
- SMTP integrates envelope fields in header

Body - contains message

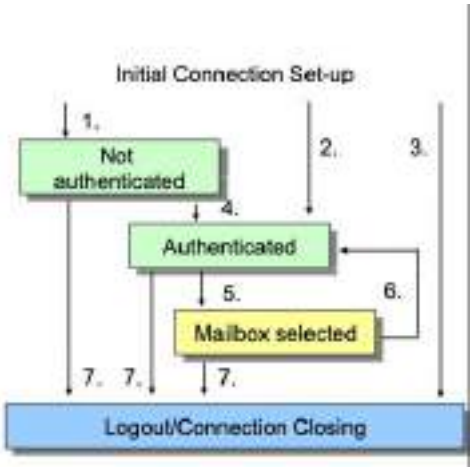
Client Operation
Problems
- Email server needs to be on - always
- PC and laptops are only connected temporarily

Solution - Emails are transferred from a server to a client based on a client protocol and stored over there

Client Protocols
- POP3 - Post Office Protocol 3
- Interactive Mail Access Protocol IMAP
- Hyper-text Transfer Protocol (HTTP) based
    - Can use the browser for this but is slow

Internet Mail Access Protocol (IMAP)
- Supports Nomadic Users - logging in from different places etc.
- We can have email directories, modify them, …
- Email can be stored on the server
- We can only download headers
- Service States
    - Non-authenticated -> Authentication Required
    - Authenticated -> Selection of directory/folder
    - Selected -> Manipulate and down-load of emails (mark as read, ...)
    - Logout: Closing down the session

Connection set-up
- Regular "OK Greeting"
- No authentication "PREAUTH Greeting"
  - Authentication performed external
- Set-up rejection
- Successful authentication
- Successful selection of mailbox
- Closing of mailbox
- Connection closing/logout



Multipurpose Internet Mail Extensions (MIME)

- 64kb, only asci, no äöü - very limited

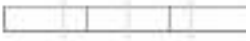
MIME has:

- Additional Header Information:
  - Contenttype:Text,multi-part,message,application,image,audio,...
  - MIME version
- Definition of content transfer encoding
  - Binary data needs to be mapped to ASCII characters

MIME Transfer Encoding

You can have no encoding and just have 7 or 8bit ASCII encoding

Base 64:
- Mapping of 8 bit binary values onto ASCII characters:
  - 24 Bits (3 Byte) → 4*6 Bits → 4 ASCII Char (restricted to values of 0 … 63)



Domain Name System (DNS) System
Resolving of names (domains) to IP addresses

Accomplished through recursion - too much information to store otherwise

Address
- Fixed length
- Tied to routing
- Often changed (when changing servers)
- Transport address is a tuple of Port and IP address

DNS hides addresses from Internet applications

Name
- Variable length
- Easy for humans to remember (hosts, mail boxes, servers)

Namespace - defines possible names
Flat name Space
- Is centralized coordinated - so no overlaps
- Not scalable
- All requests need be addressed to this central host - bad
- World wide consistency of copies is difficult to achieve

Hierarchical name space
- Due to hierarchy a structured namespace
- Localization of objects is made simpler
- Distribution of name assignments is easier
- Distributed storage of names is maintained

DNS defines syntax and rules for the delegation of a name authority and the implementation of a distributed system to map names onto addresses

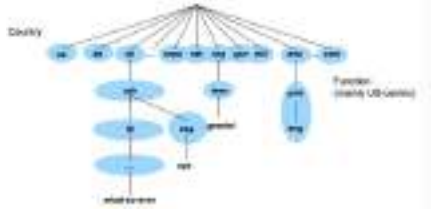
Top Level Domains - Unsponsered & Sponsored

.arpa arpanet Address and Routing Parameter Area
.biz business for companies, open to all
.com commercial for US companies, open to all
.info information for information providers, open to all
.int international for international organizations
.name names for private persons
.net network for network organization, open to all
.org organization for non-commercial organizations, open to all
.pro professionals for lawyer, tax accountants, doctors, engineers (only for U.S.A., Canada, Germany, U.K.)

.aero aeronautics for SITA only
.asia asia ICANN region Asia/Australia/Pacific
.cat catalan for katalan language and culture
.coop cooperatives, for .com LLC only
.edu educational for "post-secondary" accredited US
.gov government only for Government of the USA
.jobs jobs only for companies with job offers
.mil military for US military only
.mobi mobile for services within mobile devices
.museum museums, for association members only
.tel telephone simplified calling of persons/companies
.travel travel for travel organizations/companies
.post postal for all post and logistic companies (plan)

Top Level Domains - Country Codes
- 2 letter ISO code
- CH, DE, UK

DNS Name Space - soans a hierarchical tree with zones representing a sub tree



Every Zone maintains a **Zone File**

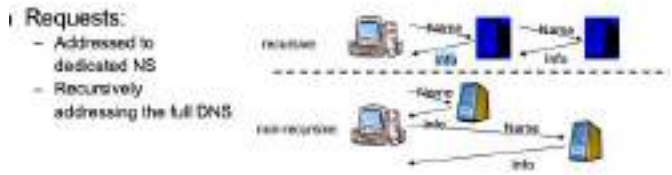
- A subset of its domains it belongs to
- One primary NS
- Minimum two secondary NSs

Operation:

- Every NS knows sub-part the namespace
- Every NS knows those IP addresses of all NS in the directly attached subdomains
- Every NS cache already known addresses
- Secondary NS maintain periodic updates (zone transfer) of local databases (driven by the primary NS)

Name Resolution

- Name resolution starts at tree root
- Name resolution initiated by DNS resolver within application



For incoming request NS checks local sub-tree, if not able to resolve, request addressed to higher level NS

- Resolver needs to know only “his” NS address (/etc/resolv.conf).

Format of DNS Messages

Query and responses apply similar format:

0	16	31
Identification	Parameter	
Number of Queries	Number of Responses	
Number of Name Authorities	Number of Additional Data	
Query Block ...		
Reply Block ...		
Authority Block ...		
Additional Data Block ...		

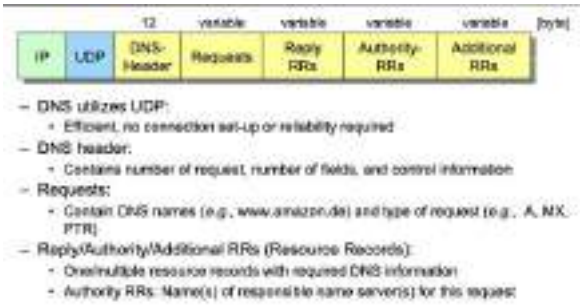
DNS Query

0	16	31
Queried DNS Name ...		
Query Type	Query Class	

DNS Response

0	16	31
DNS Name ...		
Type	Class	
Cache Validity	Length of Data	
Resource Record ...		

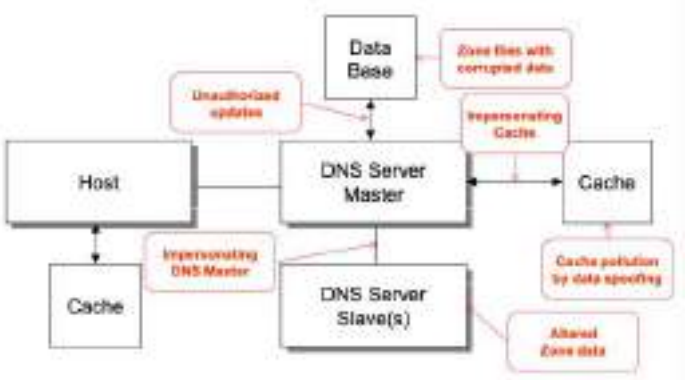
DNS Packet Format



Root Servers

- 13 Worldwide
- Root Server maintain top level domain only

Vulnerabilities



DNSSEC

DNS is plain text

- Simple UDP , no sessions
- Tree Structure With Delegations
- Each entity is responsible for a limited part of it
- Trust is Needed

DNS Security (DNSSEC) Extensions adds some security

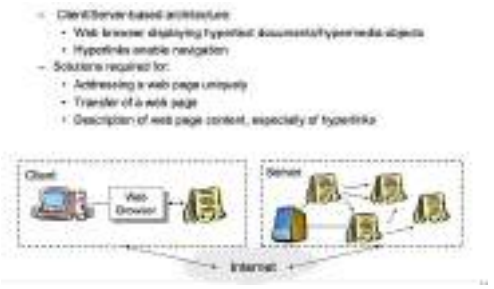
World Wide Web (WWW)

Goal - simple exchange of documents worldwide

Implementation - 1990 first prototype

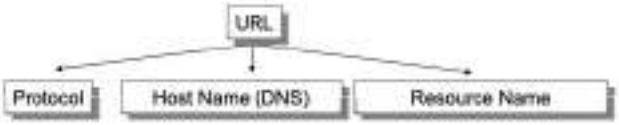
User interface: Mosaic -> Netscape -> …

Client/Server-based WWW Architecture



Addressing a Web Document

- A resource description identifies an object to be accessed within a server:
  - WWW: selected web page
  - FTP: to be transmitted file
  - Mail: Receiver of this message
- Uniform Resource Locator (URL):
  - Determines the resource's location as a name, the protocol to access this resource, and a name of the resource
    - E.g., http://www.ifi.uzh.ch/



WWW Application Protocol HTTP

- HTTP (Hypertext Transport Protocol):
  - Version 0.9/1.0 RFC1945, since January 1997 version 1.1 RFC2068
  - Mainly transfer of Web pages
- Characteristics:
  - ASCII-based application protocol (layer 7)
  - Utilized a reliable TCP connection at default port 80
  - Short-lived connection: HTTP server closes connection after a reply on a request of a client has been issued
- Sample commands:
  - `GET`: Request of a document
  - `HEAD`: Request of header information of this document
  - `POST`: Attachment of data to an existing document
  - `PUT`: Generation of a document

HTTP Messages

- Request Line:
  - Message type, requested resource, ...
- Status Line:
  - State information (error messages, HTTP version, ...)
- General Header:
  - Cache control, proxy information, ...
- Request Header:
  - Request- and client information, e.g., acceptable media/coding, ...
- Response Header:
  - Response information (location, server software, retry information, ...)
- Entity Header:
  - Information on resources and body (Compression, date of modification, language, length)
- Entity Body:
  - Message body

Example of a HTTP Request/HTTP Reply

Request:
- <Command><URL> <Version>
- Client requests non-cached, current version

TCP connection already established:
- Reply line
- Reply, date, server
- Encoding information
- Content type

HTTP Client

```
GET /index.html HTTP/1.1
Host: www.ifi.unh.ch
Pragma: no-cache
----
```

HTTP Server

```
HTTP/1.1 200 OK
Date: Fri, 24 Sep 1999 09:45:51 GMT
Server: Apache/1.3.6 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<HTML>
Structured document (HTML-based)
</HTML>
```

Surfin' … Surfin' … Surfin' the Net :-)