Technische Hochschule Rosenheim

Modul - IT Systems (IT)

Bachelor Programme AAI

12 - Summary

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Agenda

Technische Hochschule Rosenheim

Fast forward through the semester!!!

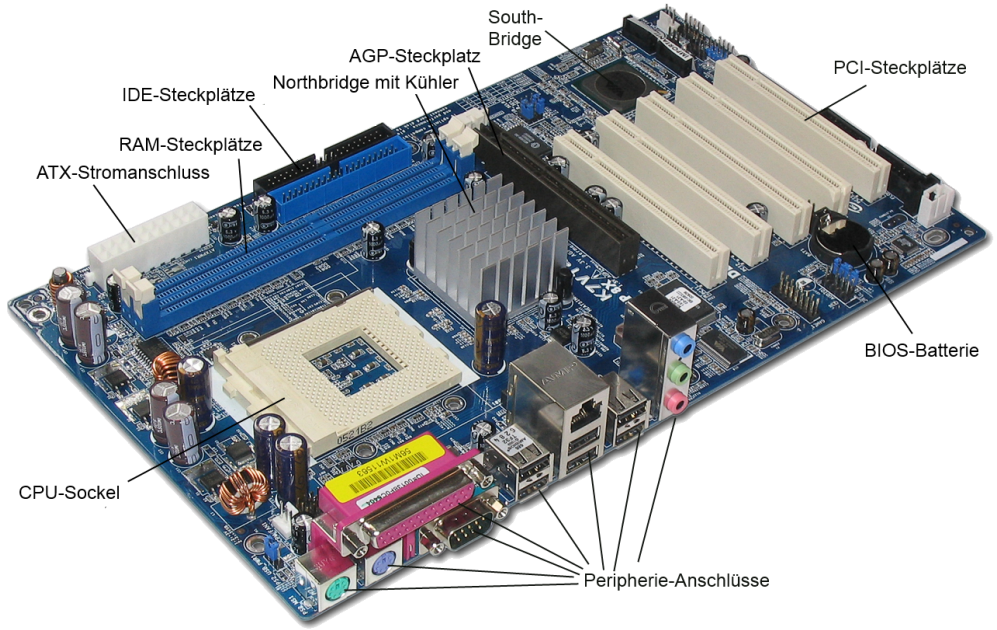
What is IT systems for AAI all about?

Technische Hochschule Rosenheim

AAI students should be capable of ...

- ... identify the components in a computer
- ... understand simple circuit networks
- ... know the difference between x86 and ARM
- ... to get along in different operating systems and not only with the mouse
- ... to know what IP means and how it is used
- ... to be able to explain the functionality of a client/server architecture
- ... to call remote methods using Remote Method Invocation (RMI)
- ... use HTTP and develop web applications
- ... put the topics of cloud, virtualization, and Docker into context.

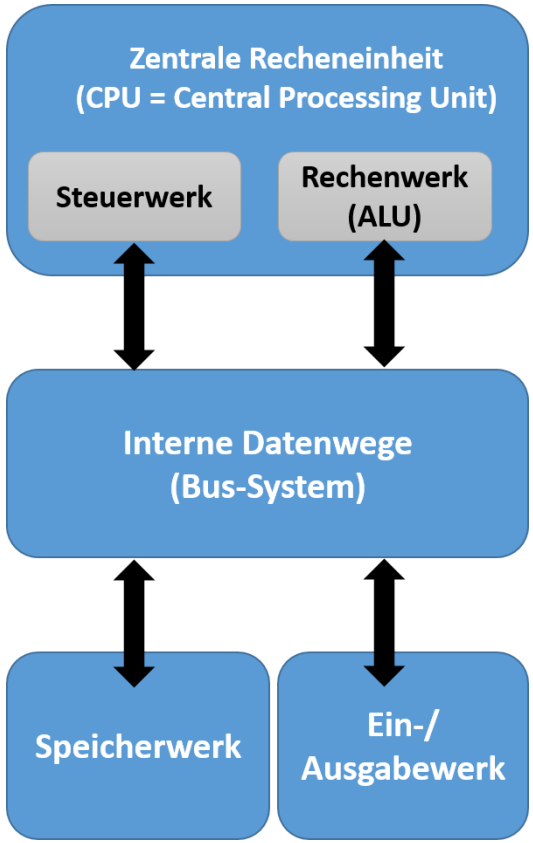
What is what?



taken from commons.wikimedia.org

v.-Neumann computer

- the calculator consists of 5 units:
  - Control unit, arithmetic unit, memory, input and output unit.
- structure of the calculator is independent of the problem to be solved
- program and data are stored in the same memory
- there is one set of instructions
  - arithmetic commands
  - logical commands
  - transport commands
  - branch commands
  - other commands
- all data are binary coded



Command Cycle (von-Neumann cycle)

Technische Hochschule Rosenheim

1. **Fetch**: The contents of the *PC* are loaded into the *MAR* and the contents of this address are fetched from memory via the *MBR* into the *IR*.
2. **Decode**: The decoder recognizes what the command is.
3. **Fetch Operands**: Fetch operands that are to be changed by the command.
4. **Execute**: The calculator executes the operation
5. **Update Program Counter (UPC)**: Increment of the instruction counter

```
MAR := PC;
MBR := S[MAR];
IR := MBR;
decode(IR);
IF NOT jump command THEN
BEGIN
    <provide operands>;
    PC := PC + 1;
ELSE
    PC := <jump destination address>;
END
```

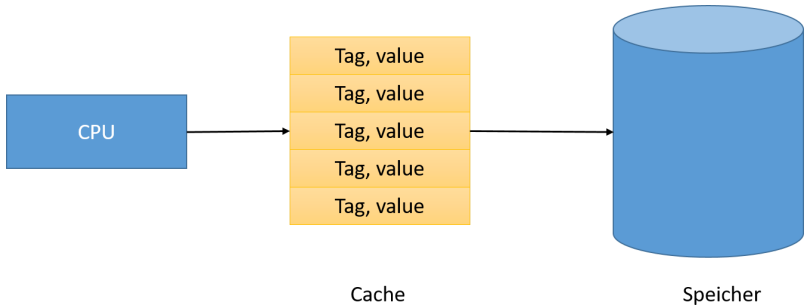
CPU Cache

Technische Hochschule Rosenheim

Basic idea of **cache**: Frequently used memory words/data should be cached to mitigate the von Neumann bottle neck problem.

Working principle of the cache:

- The CPU requests a sought datum or instruction in the cache.
- On a **cache hit**, the datum/instruction is in the cache.
- On a **cache miss**, a specific area containing the sought datum is loaded from main memory into the cache.
- **Write-Policies**: *Write-Through* and *Write-Back*.



Digital Processing

Technische Hochschule Rosenheim

The electronics in a computer works *digital*.

What does digital mean?

- in *digital electronic* it works with only two voltage levels
- that's why computers use '0' and '1' to transmit information
- a binary system exactly fulfills the requirements of *digital electronic* (abstract)
- '0' and '1' are the complementary or inverse values of each other

Boolean algebra

Technische Hochschule Rosenheim

Basis of the processing of an information on bit level is the Boolean algebra. (Origin: George Boole (1815-1864))

Purpose: To determine true and false statements beyond doubt: Boolean algebra operates on the two truth values: true (*true*) and false (*false*).

> **Note**: Boolean algebra is closely related to propositional logic. Two states false and true → two-valued (binary). These can also be described by 0 and 1.

In application (digital electronics), Boolean algebra is also called switching algebra.

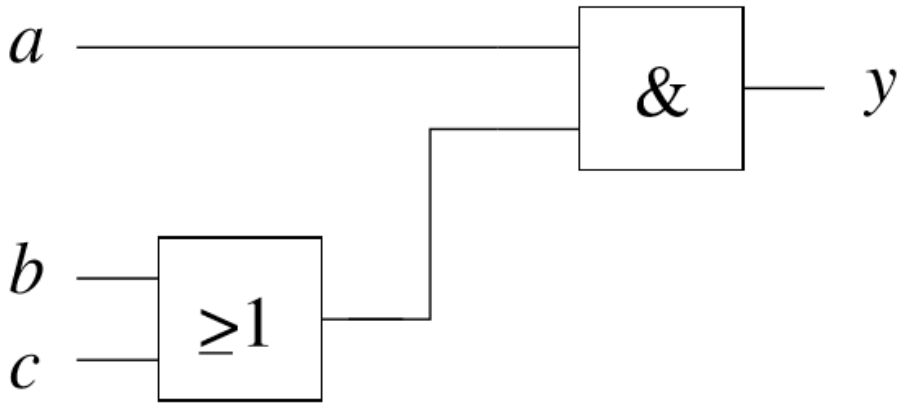
- True (W): T (True), H (High), 1 (bit set).
- False (F): F (False), L (Low), 0 (Bit not set)

Logic gates - example

Technische Hochschule Rosenheim

Any switching functions and switching networks can be composed from the logical gates.

Example: y = a ∧ (b ∨ c)



What does this circuit do?

Logic gates - example

Technische Hochschule Rosenheim

Truth table

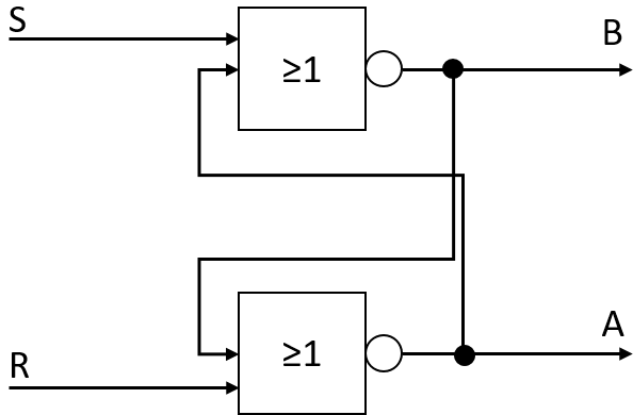
bit a	bit b	bit c	b ∨ c	a ∧ (b ∨ c)
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Realization of 1-bit memories

A memory is used to store instructions and data to be executed

How can such a gate-level memory work?

To create a 1-bit memory, we need a circuit that remembers input values. Such a circuit can be built from two NOR gates.



Alternative: **flip-flop memory devices**

Random Access Memory (RAM)

Technische Hochschule Rosenheim

- Memory devices that can read and write are called RAMs (Random Access Memory):
  - Static RAMs (SRAMS) are internally implemented with circuits that structurally correspond to *flip-flop memory devices*. They have the ability to retain their contents as long as current is flowing. SRAM access time in nanosecond range.
  - Dyanmic RAMS (DRAMS) consist of a *transistor* and *capacitor* which can then be charged and discharged. So zeros and ones can be stored. However, here each bit must be *refreshed* every few millliseconds. Have a high storage density

DRAM is slower than SRAM.

Other memories

Technische Hochschule Rosenheim

- ROMs (*Read-only Memory*): Since RAMs lose their memory contents without power, there are ROMs whose contents *are* burned in as a memory mask.
- PROMs (*Programmable ROM*): This chip works like a ROM. Here, the chip can be programmed **once** by *burning* fuses.
- EPROM (*Erasable PROM*): Can be programmed as well as erased. If a quartz window of an EPROM is exposed to a strong ultraviolet trick for about 15 minutes, all bits are set to 1. (Extension is an EEPROM = *Extended EPROM*).
- The **Flash memory** can be erased and reprogrammed block by block.

CISC

Technische Hochschule Rosenheim

CISC = Complex Instruction Set

- In a CISC model the CPU usually has few registers (memory locations in the CPU)
- Complex instruction set (e.g. process several registers at once in one loop)
- The instructions usually have unequal instruction lengths. This allows the code to be kept compact.
- Usually these instructions are realized by microcode. This is a decomposition of the instructions into smaller units, a kind of "interpreter" on the CPU.
- In modern CPUs, this microcode is even modifiable, allowing the manufacturer to introduce bug fixes.
- More instructions make the CPU design more complex, however programming more easily complex what CISC makes.

RISC

Technische Hochschule Rosenheim

RISC = Reduced Instruction Set

- RISC goes another way.
- One limits oneself to the really necessary instructions.
- As compensation you have much more registers (up to 256) on the chip, so you have much more fast register-to-register operations than slow memory-to-register operations.
- The few instructions make the design simpler and the processor cheaper to produce.
- Instead of microcode, one uses direct wiring of the instructions in the decoder, which makes execution faster than with CISC.
- At the same time, the data format is standardized.

RISC vs CISC

Technische Hochschule Rosenheim

CISC

```
01 ;CISC addition
02 ADD M3 M1 M2 ; M1 + M2 -> M3
```

RISC

```
01 ;RISC addition
02 LD R1 M1 ; M1 -> R1
03 LD R2 M2 ; M2 -> R2
04 ADD R3 R1 R2 ; R1 + R2 -> R3
05 ST M3 R3 ; R3 -> M3
```

What is an OS?

Technische Hochschule Rosenheim

Definition

An operating system (OS) is the link between the hardware of a computer on the one hand and the user or a program on the other. It includes programs that, together with the properties of the computer, "form the basis of the possible operating modes of this system and, in particular, control and monitor the execution of programs."_

Requirements

- "Optimal" utilization of the available operating resources
- fulfillment of user requirements, e.g. observance of priorities

Tasks of an OS

Technische Hochschule Rosenheim

- The operating system extends the hardware functionality, e.g. by file management
- allocation of user jobs to appropriate execution units (e.g. processes)
- appropriate scheduling, taking into account possible interactions between execution units
- management and, if necessary, appropriate allocation of resources to different execution units
- control and enforcement of protective measures (e.g. access rights), especially to ensure the integrity of data and programs, especially in multi-user operations
- Logging of the entire sequence of events in the system, collection and analysis of performance data for system optimization

Main components

- Job management, program execution
- program creation
- Process management and coordination
- Communication: I/O, file access, network communication
- User management, access control
- Error detection and handling
- Logging (monitoring, accounting)
- Resource and file management

Definition

System software comprises all programs that enable efficient and convenient use of a computer. In addition to the operating system, this includes supplementary, hardware-independent service and utility programs._

Classification according to operating modes

Technische Hochschule Rosenheim

- **Batch processing (batch processing)**: Processing a sequence of batch jobs. A batch job is assembled by the user with all necessary programs, data and Job Control Language (JCL) instructions. It is processed completely without user interaction.
  - Examples: *IBM OS/370, OS/390, MVS, BS 2000*.
- **Dialog mode (interactive processing)**: Constant change between actions of the user (e.g. command inputs) and those of the system (e.g. command execution). The user can influence the workflow in the dialog at any time.
  - Examples: *MS-DOS, MS Windows 95/98/ME/NT/2000/XP, UNIX/Linux*.

Process concept

Technische Hochschule Rosenheim

Definition

A **Program (Program)** is a static sequence of instructions in a programming language using data. It is used to code an algorithm and is generally in the form of a file._

Definition

*A (sequential) **process (also: task)** is a dynamic sequence of actions (changes of state) that comes about through the execution of a program on a processor. A process is characterized in particular by its time-varying state. It is generated in the operating system as a result of a task.*

File commands

Technische Hochschule Rosenheim

Interaction with the OS

Activity	Linux	Windows
create file	touch	echo >
list directory contents	ls	dir
Copy (copy)	cp	copy
move	mv	move
rename	mv	ren
Delete (delete)	rm	del
File output	cat	type, echo
Change directory	cd	
create directory	mkdir	mkdir
print working directory	pwd	echo %cd%

Networks

Technische Hochschule Rosenheim

- **Computer networks** are used to exchange data between computers.
- The data can be transmitted via cable, fiber, electromagnetic radiation, ... (there are also somewhat more unusual transmission media - see RFC 1149).
- One of the first large networks was ARPANET, the forerunner of today's Internet. The US government (more precisely, DARPA) wanted to create a decentralized and fault-tolerant network for military research.
- In the early 1970s, Vint Cerf and Robert Kahn finally created the basis for today's Internet, and then in the early 1980s the new Internet replaced the old ARPANET.

OSI layers

Technische Hochschule Rosenheim

Layers and meaning

#	Layer	Meaning	Units
7	Application Layer	Application Layer	
6	Presentation Layer	Data	
5	Session Layer	Communication Layer	
4	Transport Layer	TCP, UDP	
3	Network Layer	Network Layer	Packets
2	Data Link Layer	Link Layer	Frame
1	Physical Layer	Physical Layer	Bits, Symbols, Packets

Nyquist Theorem

Technische Hochschule Rosenheim

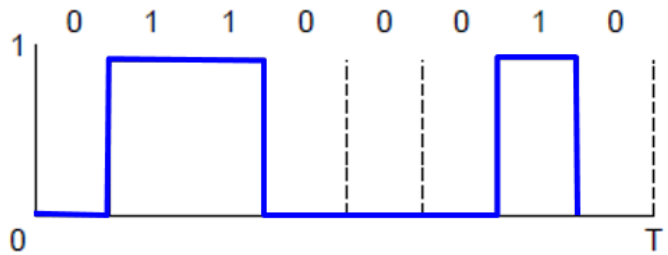
Data rate *D* with unnoised channel

- Data rate D for **unnoised** channel depends on:
  - Bandwidth B (size of the transmittable frequency range)
  - Number of signal stages used V

Nyquist theorem : $D = 2Blog_2V$

High bandwidth → high data rate

- **Question: what is V for the mapped signal?**.
  - B=4 kHZ, (channel unnoised!)
  - How many signal levels *V* are there?
  - What is the maximum data rate?



Shannon theorem

Technische Hochschule Rosenheim

Maximum data rate D with noisy channel

Shannon Theorem: $D = B * log_2(1 + S/N)$

- Delimitation
  - Applies in addition (!) to Nyquist.
  - Limit for data rate when noise is present.
- S/N: Signal-to-noise ratio (S/N)
  - Power of the useful signal S / power of the noise N
- S/N usually expressed in decibels (dB).
  - dB value: $10 * log(S/N)$
  - Example: S = 100mW, N = 1 mW, S/N = 100 → ?? dB!
- Noise sources
  - Intermodulation, crosstalk, thermal noise.

Framing

- Physical Layer receives and transmits **bitstream**.
- Error handling by link layer only possible
  - if bits are split into finite sequences (=frame).
  - Frame has redundancy (e.g. checksum), see next section.
- Problems
  - How does receiver recognize *frame start and end* from bit stream?
  - How to transmit arbitrary bit and character combinations?
- **Solutions**
  - Byte Count
  - Byte Stuffing
  - Bit Stuffing

Technische Hochschule Rosenheim

Checksum (in IP and TCP headers)

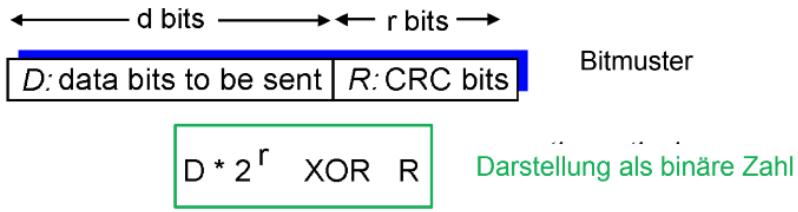
- Idea: **Addition**
  - Consider bits in groups of 16-bit words
  - Sum all 16-bit words considering the carry
  - 1's complement of the result is the checksum
- Check at receiver relatively simple
  - Add all transmitted words AND checksum
  - Result must consist of 1s bits, otherwise error

1. Wort		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
2. Wort		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Übertrag	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
Summe		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
Checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Technische Hochschule Rosenheim

Cyclic Redundancy Check (CRC)

- Consider data bits *D* to be transmitted as an unstructured bit sequence (binary number).
- Approach
  - **Generator G**: Sender and receiver know common bit pattern of $r + 1$ bits.
  - **R**: Sender determines r additional bits and appends them to data *D*.
  - The resulting bit pattern of length *d+r* is then transmitted.
  - Important: The bits *d+r* must be divisible by *G*.
  - Receiver makes sample and checks if divisible by G $(D * 2^r XOR\ R) = nG$
- Ethernet and WLAN: Implementation of CRC in hardware possible

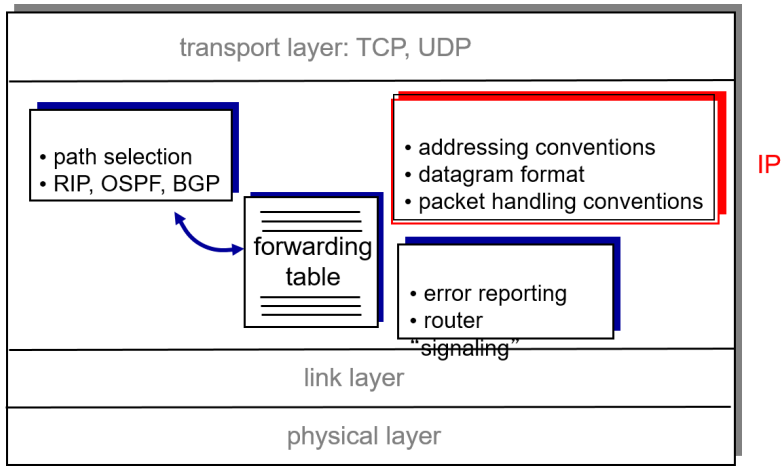


Network Layer

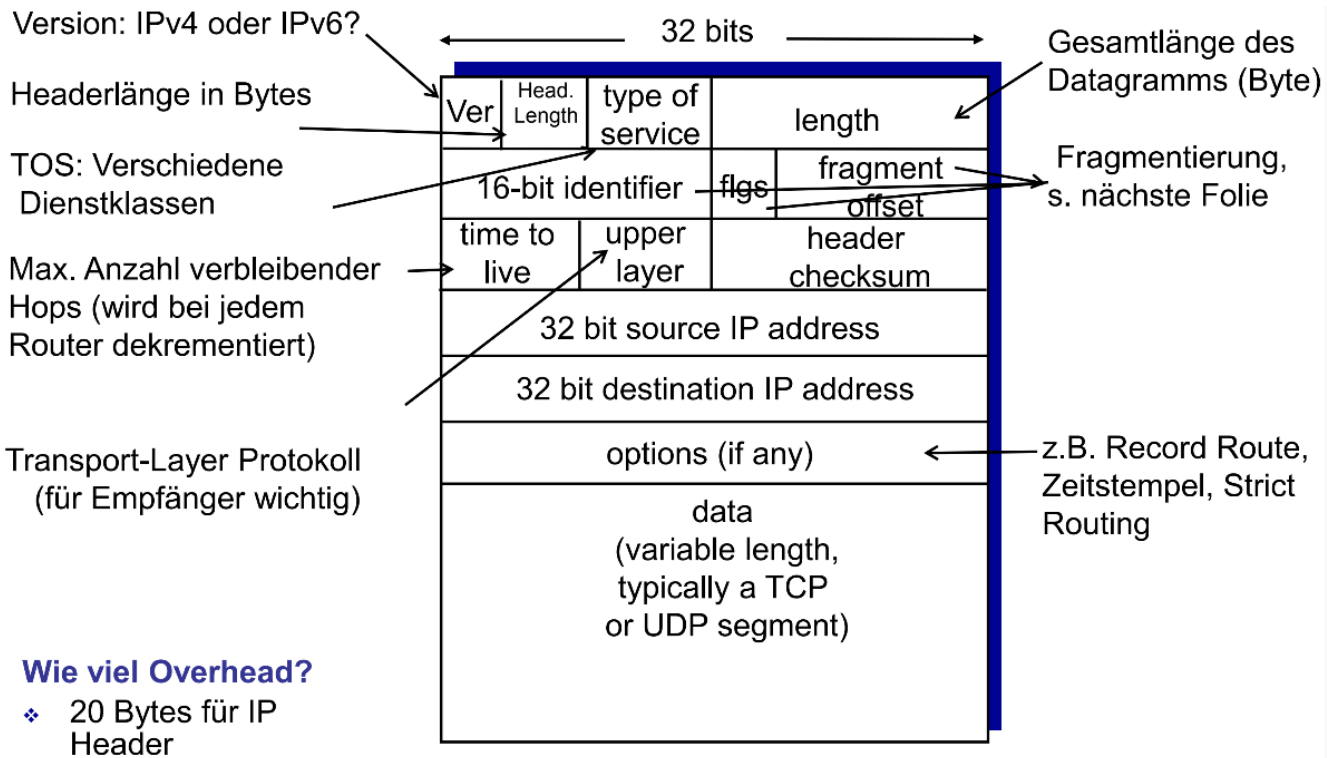
Technische Hochschule Rosenheim

Protocols

- Actually Network Layer consists of several protocols.
- But the most important one is the Internet Protocol (IP)
  - versions IPv4 and IPv6



IPv4 Datagram

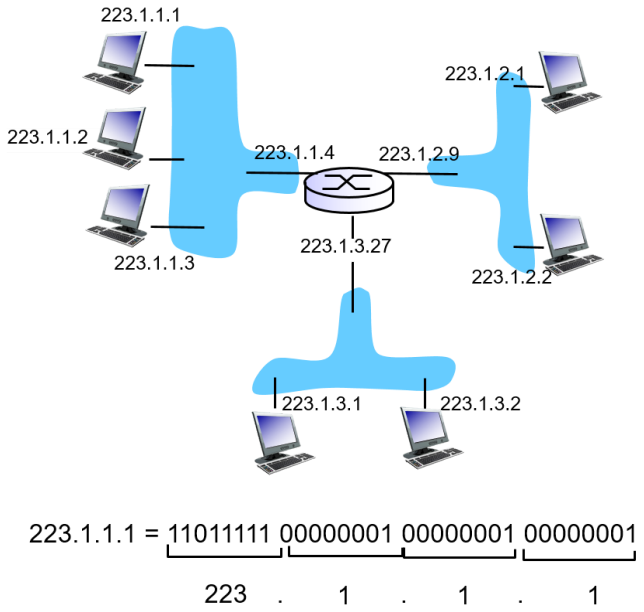


Wie viel Overhead?

- 20 Bytes für IP Header

IP addressing

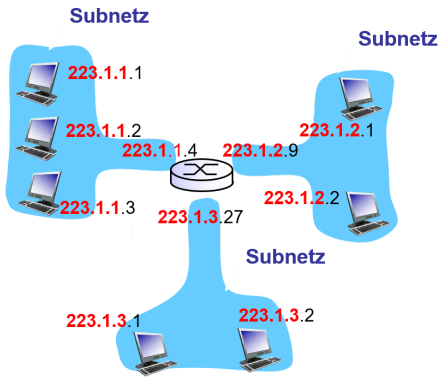
- IP address
  - 32 bit
  - Identifies host on the Internet
  - But logically belongs to Interface.
- Interface
  - Connection between host/router and link
  - Routers have several interfaces.
  - Each interface needs 1 IP address.



- Notation IPv4 address: Decimal numbers separated by dots

Subnets

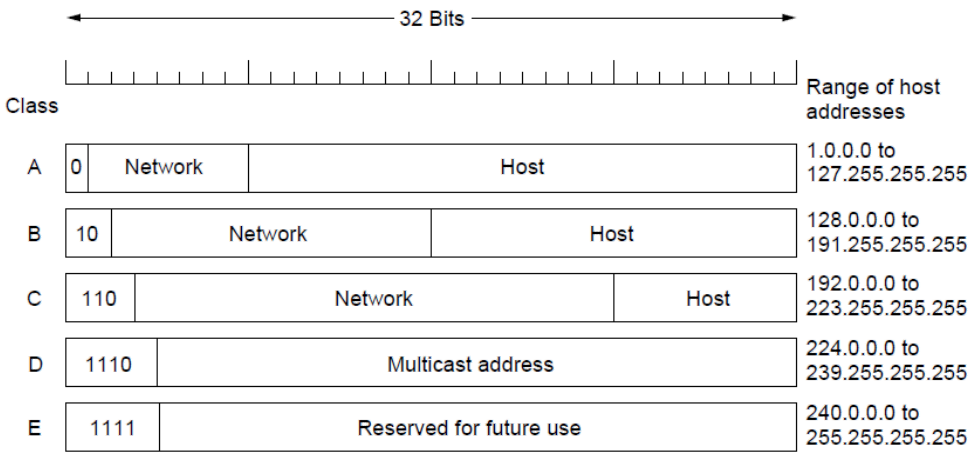
- What is an IP subnet?
  - Hosts share the same IP address prefix
  - Hosts can reach each other without router.
  - Ex: Ethernet, WLAN, etc.
- Address of a subnet
  - Subnet is addressable via common prefix!
  - Subnet mask (red): Length of the common prefix (e.g. /24)
  - Host part: Bits of the IP address which are different for each host.



- Notation, example
  - 223.1.3.0/24
  - The first 24 bits are the same for all hosts of the subnet.
- Advantage: You only have to keep subnet addresses in the routing tables.

Class Addressing

- Address ranges are passed to companies, universities, etc.
- Previously: Subnet prefixes must have mandatory length /8, /16 or /24.
- How many hosts can a Class /24 or Class /16 network have?



Special IPv4 addresses

Technische Hochschule Rosenheim

- Localhost, own PC
  - 127.0.0.1
- Private IPv4 addresses
  - globally not visible, to be used only locally in own administrative network.
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- Special addresses in a subnet
  - Example: 192.168.0.0/16 (netmask: 255.255.0.0)
  - Broadcast address: 192.168.255.255
    - For messages to all hosts of the subnet
    - All bits of the host part are set to 1
  - Network address identifies the subnet: 192.168.0.0
    - All bits of the host part are set to 0
    - Cannot be configured to interface.

UDP

- **UDP** is a *connectionless* protocol, i.e. no connection is established, data is simply sent.
- No other guarantees are made either, i.e. you have the same problems here as with *IP datagrams*:
  - The order of the arriving packets can change.
  - Packets can arrive multiple times or not at all.
  - Applications, which use UDP, must be thus robust opposite such problem, have however the advantage that data can be sent faster (the connection structure is dropped completely).
- For example, the following applications use UDP: **DNS**, **DHCP**, **SNMP**.

Technische Hochschule Rosenheim

TCP

- **TCP** is the namesake for the entire model (TCP/IP model) in addition to *IP*.
- The protocol allows the reliable exchange of data (**Reliable**).
  - Protection against transmission errors
  - Protection against packet loss
  - Protection against change of sequence
- **Connection oriented**: Establish connection before data transmission
- The following applications, for example, use TCP: **HTTP**, **FTP**, **SSH**

Distributed system

Technische Hochschule Rosenheim

A more general definition:

- A **distributed system** (VS) is a system in which.
  - Hardware and software components,
  - which are located on networked computers,
  - communicate with each other and coordinate their actions
  - by exchanging *messages*.
- A **distributed application** is an application,
  - that uses a distributed system to solve an application problem.
  - It consists of various components that communicates with the components of the VS as well as the users.
- Important aspect here: transparency, i.e., the distributed nature is hidden from the user.

Interaction Models

Technische Hochschule Rosenheim

- There are most different possibilities, how the software components on the computers interact with each other, in order to solve an application problem.
- In the last years some basic models (architectures) developed for it:
  - Client-Server (also Multi-Server)
  - Message-oriented architectures
  - Service-Oriented Architecture (SOA)
  - Multi-Tiered
  - peer-to-peer
  - grid computing
  - Cloud Computing
  - Microservices

Implementation via HTTP

- In practice, the *REST* paradigm is preferably implemented using HTTP/S.
- Services are addressed via **URL/URI**.
- The HTTP methods (`GET, POST, PUT, DELETE`) specify which operation a service should perform.

Since the World Wide Web already provides the necessary infrastructure for REST, you can already take your first steps with the corresponding interfaces using a browser. A possible starting point for this is the Fake Online REST API for Testing and Prototyping available at http://jsonplaceholder.typicode.com/.

URL/URI

Technische Hochschule Rosenheim

- URI = Uniform Resource Identifier (RFC: 3986): A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource.
- URL = Uniform Resource Locator: The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location")

```
          foo://example.com:8042/over/there?name=ferret#nose
          \_/ \______________/\_________/ \_________/ \__/
           | | | | |
       scheme authority path query fragment
          | _____________________|__
          / \ / \
          urn:example:animal:ferret:nose

authority = [ userinfo "@" ] host [ ":" port ]
```

Technische Hochschule Rosenheim

Client Request

```
GET /docs/index.html HTTP/1.1
Host: www.nowhere123.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
(blank line)
body
```

- Line 1: *Request Line* `request-method-name request-URI HTTP-version`
- line 2-5: *Request Header* `request-header-name: request-header-value1, request-header-value2, ...`
- Line 7: *Body*

Technische Hochschule Rosenheim

Server response

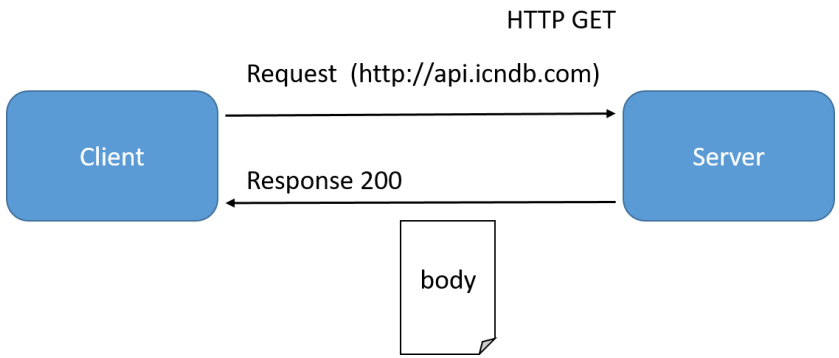
```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>
```

- Line 1: *status line* `HTTP-version status-code reason-phrase`
- Lines 2-10: *Response Headers* `response-header-name: response-header-value1, response-header-value2, ...`
- Line 12: *Body*

Making requests

REST requires the client to submit a request so that the server can respond with a response. A *Request* usually consists of:

- **HTTP verb** (standard operation): Defines the semantic of the *Request*.
- Header**: Allows the client to send further information (format, etc.).
- **Path**: Path to the resource (URL/URI)
- **Body** (optional): a message to be sent



cURL

Technische Hochschule Rosenheim

- Client for URLs -

... is a program library and a command line program for transferring files in computer networks

- cURL is available on almost all OS
- Supports the HTTP commands: Option -X [GET|POST|PUT|DELETE] -d : Transfer data in HTTP body --data "@path_of_file": transfer the contents of a file

GET

```
curl -X GET https://jsonplaceholder.typicode.com/todos/1
```

POST

```
curl -d '{"key1": "value1", "key2": "value2"}'
     -H "content-type: application/json"
     -X POST http://localhost:3000/data
```

What is JSON?

Technische Hochschule Rosenheim

- JSON = *JavaScript Object Notation*.
- Specification is available here: JSON
- JSON is a lightweight format for storing and transmitting data
- JSON is an open "quasi-standard" (used very often with RESTful services)
- JSON is *almost* "self-writing" and easy to understand
  - No schema validation, (see XML and XMLSchema).
  - but there is JSON Schema

```
{
    "employees":[
        {"firstName": "John", "lastName": "Doe"},
        {"firstName": "Anna", "lastName": "Smith"},
        {"firstName": "Peter", "lastName": "Jones"}
    ]
}
```

Cluster

Technische Hochschule Rosenheim

A computer cluster describes a usually large number of individual computers networked with each other, which are used to process individual subtasks belonging to an overall task in parallel.

- From the outside, a computer cluster looks like a single system.
- The respective nodes are connected to each other via a fast network.
- The reliability of such a computer cluster through redundancy.
- Computer clusters can be best used for processing batch jobs where many parallel partial calculations are performed.

ABER: However, if the processing involves subtasks that must be synchronized to a high degree, computer clusters are not suitable for this purpose because the communication overhead between the individual systems puts the performance gain that results from parallel processing into perspective again!

Cluster

Technische Hochschule Rosenheim

The goal of cluster computing is to provide very high computing power or a particularly fail-safe computing environment.

- **Homogeneous clusters** are characterized by the fact that the respective computers belonging to the cluster all use the same operating system and the same hardware.
- **Heterogeneous clusters** belong, may have different operating systems and hardware.

Cluster types

- **High Availability Clusters**: High availability clusters are used to increase availability and provide better resilience (Active Directory, ...); avoid a **single point of failure**.
- **Load-Balancing Cluster**: Load balancing (WebSites, ...).
- **High Performance Computing Cluster**: Performing computations (simulations, ...).

Amdahls' Law

Technische Hochschule Rosenheim

Amdahl's Law deals with the question of how much a certain program can be accelerated by parallelization.

It assumes that every program can be parallelized to a certain extent, and to a certain extent it must run sequentially, i.e. cannot be parallelized at all.

- The fraction that can be executed in parallel is P
- The runtime L can be normalized to 1, then the sequential part is (1-P)

Total runtime: $T_0 = (1 - P) + P$

With *N* processors the parallel part is shortened, so that applies

$$T(N) = (1 - P) + P/N$$

and as time gain *G* (Gain) results $G = \frac{L}{T(N)} = \frac{1}{(1-P)+P/N}$

Gustafson's law

Technische Hochschule Rosenheim

So the question is what problem size we can hand in the original runtime L=1 (normalized)on an N-processor system:

$$runtime L = 1$$

$$1 = (1 - P') + \frac{K * P'}{N}$$

$$P' = \frac{K * P'}{N}$$

$$N = K$$

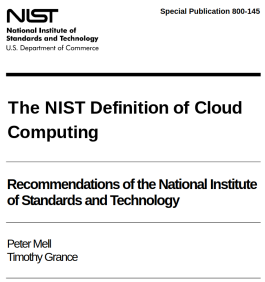
According to our model, we can always compute N times the problem size, regardless of the parallelizable part P'

The NIST Definition of Cloud Computing

Technische Hochschule Rosenheim

National Institute of Standards and Technology

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.


NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-145

The NIST Definition of Cloud Computing

Recommendations of the National Institute of Standards and Technology

Peter Mell
Timothy Grance

https://csrc.nist.gov/publications/detail/sp/800-145/final

Deployment Models

Technische Hochschule Rosenheim

- **Private Cloud** : In the case of "Private Clouds", the focus is on the fact that both the provider and the user are located in the same company, which means that, for example, all the problems from the area of data security are more or less invalid.
- **Public Cloud**: A "public cloud" is a "cloud" which is public, i.e. can be used by any person or company and is no longer restricted to internal applications of a single institution/company. In this case, problems related to data security also take effect, and each actor must decide for itself how much and which data it wants to keep outside its immediate control.
- **Hybrid Cloud**: A company operates its own "private cloud" and also uses a "public cloud" as a failover strategy or for peak loads.
- **Community Cloud**: The cloud infrastructure is provided by a "community" for a "community."

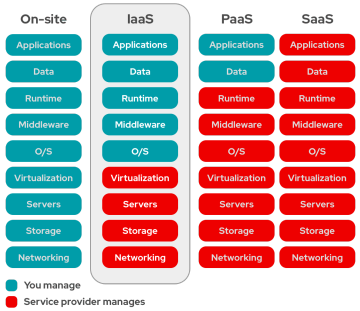
Characteristics

Technische Hochschule Rosenheim

- **On-demand self-service**: A consumer can unilaterally provision computing capabilities.
- **Broad network access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resource pooling**: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.
- **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability1 at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

Service Models

- **IaaS (= Infrastructure as a Service)**
  - Clouds provide usage access of virtualized computing resources such as computers, networks, and storage.
  - With IaaS, users freely design their own virtual computer clusters and are therefore responsible for the selection, installation, operation and functioning of their own software.
- **PaaS (= Platform as a Service)**
  - Clouds provide usage access of programming or runtime environments with flexible, dynamically adaptable computing and data capacities.
  - With PaaS, users develop their own software applications or have them run here, within a software environment provided and maintained by the service provider.
- **SaaS (= Software as a Service)**.
  - Clouds provide usage access to software collections and application programs.
  - SaaS providers offer special selections of software running on their infrastructure.
  - SaaS is also referred to as software on demand.



taken from https://www.redhat.com/de/topics/cloud-computing/what-is-iaas

Virtualization

Technische Hochschule Rosenheim

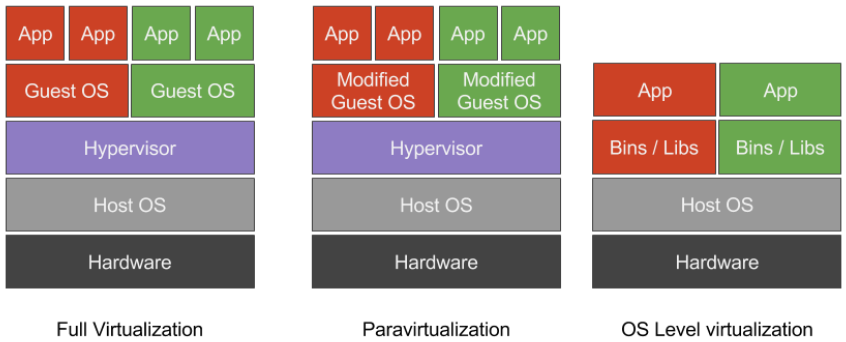
There are several concepts of virtualization:

- Software side virtualization
  - OS level virtualization (containers)
  - Application virtualization
- Hardware-side virtualization
  - Full virtualization
  - Paravirtualization
  - emulation

OS level virtualization

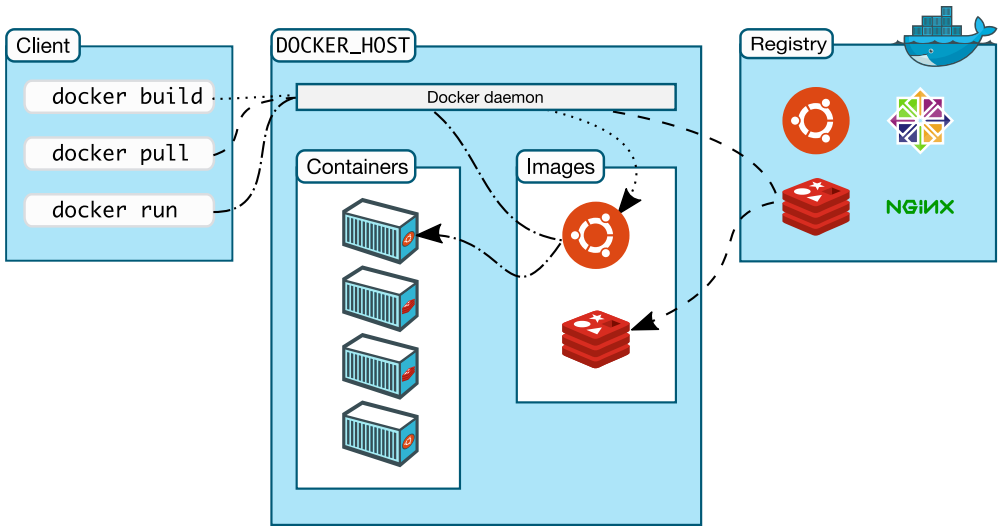
Technische Hochschule Rosenheim

- Lightweight approach: there is no hypervisor
- Each app runs as a process in the host (isolation by OS mechanisms)
  - Isolation of the process by kernel namespaces
  - Isolated file system
- almost no performance loss



A bit about Docker

- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure.
- With Docker, you can manage your infrastructure in the same ways you manage your applications.



taken from https://docs.docker.com/engine/docker-overview/

What is important?

Technische Hochschule Rosenheim

- von Neumann calculator
- Nyquist / Shannon theorem
- switching networks
- memory types
- OSI layers
  - checksum
  - MAC address
  - IP
- TCP/UDP
- Shell Coding
- HTTP and REST
- Architecture types
- Cloud/ Virtualization