# Distributed Systems

## Introduction

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
`oliver.hahm@fb2.fra-uas.de`
`https://teaching.dahahm.de`

15.04.2024

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Motivation

Why do we need distributed systems?

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Semiconductor Technology: Performance and Costs

- Memory chips:
  - 1973: 4 kB,
  - 1993: 16 MB,
  - 2012: 16 GB
  - 2021: 512 GB (Samsung DDR5 DRAM)
- Moores's law (1965): The number of transistors in an integrated circuit (IC) doubles about every two years
- The costs per transistor function decrease to one tenth every four years

## Semiconductor Technology: Performance and Costs

- Memory chips:
  - 1973: 4 kB,
  - 1993: 16 MB,
  - 2012: 16 GB
  - 2021: 512 GB (Samsung DDR5 DRAM)
- Moores's law (1965): The number of transistors in an integrated circuit (IC) doubles about every two years
- The costs per transistor function decrease to one tenth every four years

$\Rightarrow$ Computers are becoming...

... More powerful
... Cheaper
... Smaller

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## The ARPANET

1962 The idea of the 'Internet' as 'tool to create critical mass of intellectual resources' (Licklider, Taylor)

1974 Basics of TCP/IP written on paper by Cerf/Kahn (IP=Internet Protocol, TCP=Transmission Control Protocol), standardization in the following years

1982 Transition towards IP version 4 (IPv4) [1]

from 1983 Dissemination of TCP/IP due to Berkeley UNIX 4.2 BSD, source code publicly available

2

Author: Gorthmog, CC BY-SA 4.0

[2]deprecated, but still widely used

## The World-Wide Web (WWW)

from 1970 Work about *hypertext systems* (i.e., distributed network of node documents connected by pointers with rudimentary navigation options) by Ted Nelson (Project Xanadu)

1990 Proposal of a hypertext project at *CERN* in Geneva by Tim Berners-Lee and Robert Cailliau: cradle of the world wide web

1992 Publication of an open version of a web server and browser (Unix based) by CERN, by the end of the year about 50 web servers are online

1996 First *search engines* with a site-scoring algorithm, e.g., Google search

2004 Start of *Web 2.0* brought up blogs and RSS as well as services like Facebook or Twitter

2011 The Websocket protocol is standardized, providing communication channels "over HTTP"

## Ubiquitous Networks

| | |
|---|---|
| 1982 | A Coca-Cola vending machine was *connected to the Internet* at Carnegie Mellon University |
| 1995 | The first specification of IPv6 is published |
| 1996 | Hewlett-Packard and Nokia release the OmniGo 700LX and the 9000 Communicator, first smartphone predecessors |
| 1997 | Kristofer S. J. Pister, Joe Kahn, and Bernhard Boser (Berkeley) preset a research project proposal called *Smart Dust* |
| 1999 | Kevin Ashton (P&G) coined the term Internet of Things |
| 2001 | Wikipedia goes online |
| 2004 | Facebook is founded |
| 2007 | Apple releases the first iPhone |
| 2014 | The IETF working group *CORE* publishes a first specification about the Constrained Application Protocol (CoAP) |

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Today's Classes of Computer Systems

- Personal Computer (PC, Desktop), Workstations
- Server, Mainframes
  - Highly reliable processing of mass data
  - High to ultra-high performance I/O-units
  - Server provide services in computer networks
- Supercomputer
  - Variety of processors/nodes
  - very high processing performance
  - Example: numerical calculations for weather forecasting
- Embedded Computer
  - Part of machines, devices, or facilities
  - The computing unit remains in the background compared to the (main) functionality of the surrounding system
  - Cyber-Physical System

## Current Development

- Today's computer become more and more powerful and they have an increasingly better price-to-performance ratio, but this is achieved only by gradual improvements of known techniques
  - Processors
    - Reduced development cycles due to improved design tools
    - Focus on processors with Intel instruction sets for office usage
    - various $\mu$Controller types for embedded Systems (ARM, MIPS, RISC V . . . )
    - Multicore processors
  - Systems
    - Increased use of systems with many nodes
    - e.g., blade server, HPC cluster
  - Networks
    - Increasing data rate
    - Manifold quality of service (QoS) requirements
    - Mobile nodes

## Current Development (2)

- Virtualization
  - Virtual machines (VMs)
  - Memory virtualization (Software Defined Storage)
  - Virtual networks (Software Defined Networks, SDNs)
- Virtual infrastructures (Cloud Computing)
- Internet of Things, Industry 4.0/Industrial Internet
- Big Data

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

What is a distributed system and what do we need to build one?

## Distributed Systems – A definition

A **Distributed System** is

- a collection of autonomous computing systems (nodes),
- coupled over a logical network, and
- appearing to its users as a single coherent system.

## Distributed Systems – A definition

**A Distributed System is**

- a collection of autonomous computing systems (nodes),
- coupled over a logical network, and
- appearing to its users as a single coherent system.

Lemma:

- We need a network (i.e., connect the nodes)
- Communication is usually based on some kind of **middleware** providing a consistent access to the nodes and a common semantics for operations and results
- Independent nodes may behave *erratically* and we need some mechanism to manage those

## More Definitions

### Coulouris

“A system in which hardware or software components located at **networked computers** communicate and coordinate their actions only by **message passing**.”

## More Definitions

### Coulouris

"A system in which hardware or software components located at **networked computers** communicate and coordinate their actions only by **message passing**."

### Tanenbaum

"A distributed system is a collection of **independent** computers that **appear to the users** of the system as a single computer."

## More Definitions

### Coulouris

"A system in which hardware or software components located at **networked computers** communicate and coordinate their actions only by **message passing**."

### Tanenbaum

"A distributed system is a collection of **independent** computers that **appear to the users** of the system as a single computer."

### Lamport

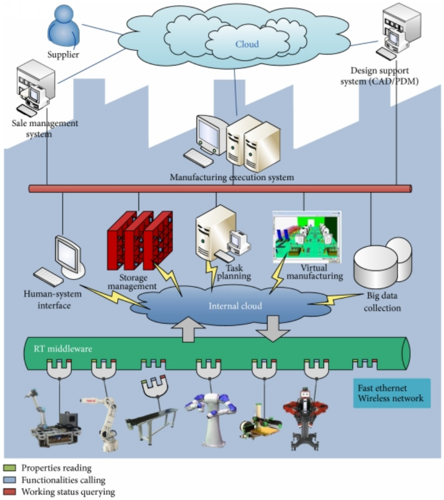
". . . a system in which the failure of a computer you didn't even know existed can render your own computer unusable."

## Brainstorming

Can you think of an example for a distributed system?

## Examples for Distributed Systems

- **Build systems**
- The **Domain Name System** (DNS)
- **Core Router infrastructure** on the Internet
- **Peer-to-peer** network applications (like BitTorrent for filesharing)
- Synchronized **calendars**, **task planners**, or **address books**
- **Automated production** lines
- Amazon Web Services (AWS) **cloud** solution



https://commons.wikimedia.org/wiki/File:System-architecture-of-the-smart-factory.jpg

## Basic Concepts of Distributed Systems

- Strong Coupling: Two software components are called **strongly coupled**, if they communicate with each other by sharing common resources, i.e.,
  - shared typed objects
  - shared memory segments
- Loose Coupling: Two software components are called **loosely coupled**, if they communicate with each other by message passing (increased autonomy of the components)
- Analogously, there are corresponding paradigms at the level of application programming paradigms that are based on sharing or message passing.

## Distributed Program/Distributed System

- A **distributed program** consists of a set of loosely coupled software components that cooperate (by message passing) with respect to a common problem solution
- A distributed program contains
  - a distributed state
    (in the respective software components)
  - distributed control/coordination, to accomplish joint problem solving
- A **distributed system** is a computing system that executes a distributed program

## Computer Networks vs. Distributed Systems

### Computer Network

The autonomous computers are explicitly visible (and have to be explicitly addressed)

## Computer Networks vs. Distributed Systems

### Computer Network

The autonomous computers are explicitly visible (and have to be explicitly addressed)

### Distributed System

Existence of multiple autonomous computers is not visible to the users ($\Rightarrow$ **transparency**)

## Computer Networks vs. Distributed Systems

### Computer Network

The autonomous computers are explicitly visible (and have to be explicitly addressed)

### Distributed System

Existence of multiple autonomous computers is not visible to the users ($\Rightarrow$ **transparency**)

- But many issues have to be tackled for both
- Every distributed system relies on services provided by a computer network

## Advantages and Challenges

Can you think of advantages or additional challenges to overcome?

## Advantages and Challenges

Can you think of advantages or additional challenges to overcome?

**Advantages:**

- Higher performance
- Higher availability
- Location independence

**Challenges:**

- Synchronization required
- More sources of failures
- Communication overhead

## Organizational

- HIS registration
  - For this module the exercises are **mandatory**, i.e., you are only allowed to take the exam if you successfully passed the exercises
  - You have to register for the exercises via the HIS until **April 29, 2024**!
  - Registrations are **binding** $\Rightarrow$ you cannot withdraw from a registration
  - If you do not register before the deadline you cannot pass the exercises $\Rightarrow$ you cannot take the exam

## Organizational

- HIS registration
  - For this module the exercises are **mandatory**, i.e., you are only allowed to take the exam if you successfully passed the exercises
  - You have to register for the exercises via the HIS until **April 29, 2024**!
  - Registrations are **binding** $\Rightarrow$ you cannot withdraw from a registration
  - If you do not register before the deadline you cannot pass the exercises $\Rightarrow$ you cannot take the exam
- No exercise sessions next week (April 30 and May 1)

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Transparency (User Perspective)

- Transparency means that certain things become **invisible** $\rightarrow$ in the case of a distributed system: certain properties
- Common types of transparency:
  - Location transparency: enables resources to be accessed without knowledge of their physical or network location, esp. the location is not part of its name
  - Access transparency: Enables local and remote resources to be accessed using identical operations
  - Migration or mobility transparency: The component can be moved without changing the user interface
  - Replication transparency: Enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas used by users

## Types of Transparency (Developer Perspective)

- More types of transparency:
  - Concurrency transparency: Enables several processes to operate concurrently using shared resources without interference between them
  - Scaling transparency: Allows the system and applications to expand in scale without change to the system structure or the application algorithms
  - Performance transparency: Allows the system to be reconfigured to improve performance as loads vary.
  - Failure transparency: Enables the concealment of faults, allowing users to complete their tasks despite the failure of hardware or software components

## How transparent are modern Distributed Systems?

### Making Distributed Systems Manageable

Transparency helps to simplify the management and programming of the system, since the aspect in question does not need to be considered by the user of the system.

- In order to appear as a uniform as possible from a system view, any distributed system strives to realize all transparency types
- Perfect distributed systems that abstract from all aspects do not currently exist
- Many modern systems and middlewares support particular transparency types (e.g., location transparency)

## Agenda

Motivation and History
- Semiconductor Technology
- Communication Technology
- System Technology

Basic Concepts of Distributed Systems
- Basic Concepts
- Types of Transparency
- Design Principles
- Operating System Support (LOS - NOS - DOS)
- Overview

## Principle: Robustness

**Robustness** of a distributed system requires its objects to...

- be available at *all* times,
- be agnostic to the topology, and
- behave fail-safe.

## Principle: Robustness

**Robustness** of a distributed system requires its objects to...

- be available at *all* times,
- be agnostic to the topology, and
- behave fail-safe.

In order to design a robust distributed system, it depends...

- on a well-chosen and qualified architecture,
- mature software on the nodes, i.e., possess as little bugs as possible,
- redundancy and fail-over mechanisms.

## Robustness versus Failure

Failures in a distributed system may (and will) happen…

- for the **entire** distributed systems ⇒ not usable any more.
- for a **few specific nodes and objects** ⇒ partial unusable *components*.

### Vital Actions for a robust distributed system

In order to maintain the functionality of a distributed system in case of failures of its components, some actions are crucial.

- Detecting failures (identifying component and perhaps reason),
- Marking failures (making it visible to others),
- Tolerating failures,
- Recovery from failures, and ideally
- setting up Redundancy.

## Principle: Scalability

A distributed system behaves **scalable**, if it supports significant increase of

- the number of users,
- the number of nodes, or
- the numbers of objects (resources).

## Principle: Scalability

A distributed system behaves **scalable**, if it supports significant increase of

- the number of users,
- the number of nodes, or
- the numbers of objects (resources).

In practice, the **scalability** of a distributed system depends on

- the **number of supported users and processes**, restricted by memory and computing power,
- the **physical distance** between nodes, introducing latency in information exchange, and
- the **domain-model** of the distributed system, confining the administrative growth.

## How to scale?

Several properties are required to make a distributed system scalable:

- Use **asynchronous** communication
- **Parallelization**, e.g., of request handlers
- Keep information **local** ($\Rightarrow$ latency is minimal)
- **Cache** as much as possible (local replication of data)
- Organize data **hierarchical** ($\rightarrow$ DNS)
- Reduce name lookups for resources; instead use an algorithmic scheme
- Move computation to clients

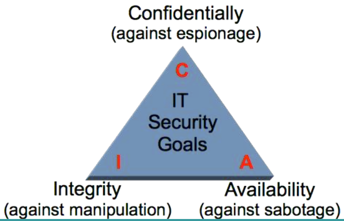
In other words: **reduce the need for communication** and **decouple components**.

### Challenges of replication

Replicating data may lead to inconsistencies among the different copies of a data set. Hence, global synchronization of objects and **time** on the nodes is required. However, strict synchronization and thus long-distance coherence is almost impossible.

## Principle: Security

Security is crucial for distributed systems
**IT Security** targets the following goals ⇒

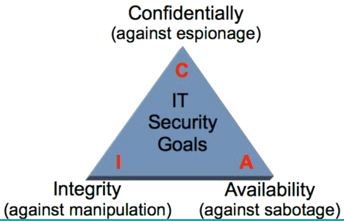


### A distributed system meets the **IT Security** goals, if it provides

- confidential access and storage of data (by means of en/de-cryption),
- integrity for data-in-rest and data-in-flight (and data-in-computation),
- availability of resources even under critical circumstances and failure conditions.

## Principle: Security

Security is crucial for distributed systems
**IT Security** targets the following goals ⇒



A distributed system meets the **IT Security** goals, if it provides

- confidential access and storage of data (by means of en/de-cryption),
- integrity for data-in-rest and data-in-flight (and data-in-computation),
- availability of resources even under critical circumstances and failure conditions.

Common failure conditions for distributed systems may be triggered from the outside:

- (Distributed) *Denial of Service* (DDoS)
- *Malware* infection.

## Security versus Ease of Use

The Where is the problem with **IT Security** and user acceptance?

- A common opinion is, that (IT) security is too complicated to handle by the user but **usable** security is required
- As a result, IT security concepts have been developed to be opaque (invisible) to the user and works 'automatically'.
- Any **IT Security** requires that the user of the system is *informed* about the current security level; otherwise may act irresponsible.

$\hookrightarrow$ In order to realize security, compromises with other quality features of software development are always necessary. However, usability is a key criteria to implement security.

## Network Security

Do you know a technical method to achieve end-to-end authenticity, integrity, and confidentiality?

## Network Security

Do you know a technical method to achieve end-to-end authenticity, integrity, and confidentiality?

Transport Layer Security (TLS) (formerly **SSL**) is a popular and wide-spread approach to establish a secure communication channel.

## Principle: Openness

### An open distributed system provides

- an uniform communication mechanism (interoperability),
- well defined and published APIs (Application Program Interface),
- ways to publish these interfaces to enable remote access,
- permitting the use of the shared resources,
- allows access independently from specific hardware, (computer) languages, and from heterogeneous sources (clients/users),
- and is well tested and verified regarding these requirements.

$\hookrightarrow$ Openness requires particular attention when taking care for robustness and security.

## Homogeneity and Heterogeneity

Distributed systems consist of a vast variety of heterogeneous components; moreover, different understanding of the shared **objects** due to

- different hardware platforms (big vs. little endian),
- different computing Languages (Java, C, Python),
- different integration mechanisms (middleware).

[3] see: `http://pubs.opengroup.org/onlinepubs/9699919799/`

## Homogeneity and Heterogeneity

Distributed systems consist of a vast variety of heterogeneous components; moreover, different understanding of the shared **objects** due to

- different hardware platforms (big vs. little endian),
- different computing Languages (Java, C, Python),
- different integration mechanisms (middleware).

### Some thoughts:

- In order to provide **openness** a qualified abstraction layer is required and proprietary solutions need to be avoided. A solid foundation to realize **openness** is the `POSIX`[3] standard, to be obeyed.
- On the other hand, homogeneity often yields a restricted view to the problem and is subject of inefficient legacy solutions which tend to simultaneously crash in case of a problem.

[3] see: `http://pubs.opengroup.org/onlinepubs/9699919799/`

## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## LOS (Local Operating System)

- Common OS for a single node (without support for distributivity)
- Examples:
  - IBM MVS,
  - UNIX System III,
  - DOS, Windows 3.1,
  - ...

## NOS (Network Operating System)

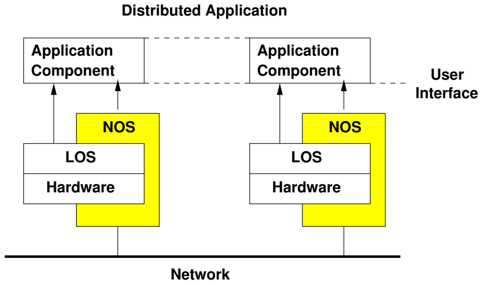
- OS extension of various LOS' for a multi-computer system to provide certain functions wrt.
    - File system,
    - Protection (user management),
    - remote program execution

  on a system level, more or less transparent

  Examples:
    - Novell NetWare,
    - MS Windows for Workgroups and basically all versions of Windows since Windows 98,
    - UNIX Yellow Pages (NIS) und Network File System (NFS)
    - Linux

## NOS (2)

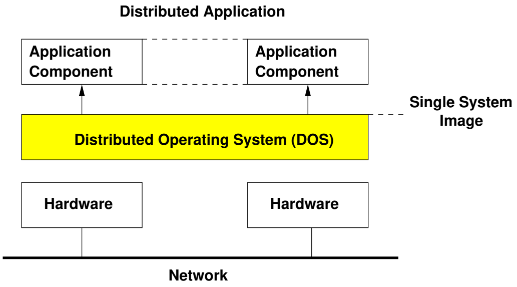
- Basic structure of a NOS:



- The underlying LOS may be the same or different
  Examples:
  - Netware Client for DOS, NT, . . .
  - NFS Client for UNIX, NT, . . .

## DOS (Distributed Operating System)

- A distributed operating system is a basic OS which
  - provides a unified system view of a multi-computer system to its users
  - is based on algorithms that run under distributed control and exchange of messages in order to implement transparency



## Agenda

- Motivation and History
  - Semiconductor Technology
  - Communication Technology
  - System Technology
- Basic Concepts of Distributed Systems
  - Basic Concepts
  - Types of Transparency
  - Design Principles
  - Operating System Support (LOS - NOS - DOS)
  - Overview

## Topics of this Lecture

Some topics that will be covered in this lecture:

- Network and Concurrent Programming
- Communication Patterns
- Remote Invocation
- Directory Services
- Security
- Global State and Time
- Fault Tolerance
- Distributed Filesystems
- Middleware
- *Service Discovery*
- Web Services and REST
- *Coordination and Transactions*
- Internet of Things
- *Information Centric Networking*

Important takeaway messages of this chapter

- Physical limits in semiconductor technologies require new approaches to boost performance
- The ubiquity of the Internet makes distributed systems increasingly important
- The underlying distributed nature of the components remains invisible to the user and programmer of a distributed system ($\rightarrow$ transparency)