



OST

Eastern Switzerland
University of Applied Sciences

Distributed Systems (DSy)

Categorization

Thomas Bocek

29.03.2025

Learning Goals

- Lecture 1 + 2
 - Distributed systems add complexity. Avoid complexity!
 - Why do we need distributed systems?
 - Scaling (if one machine is not enough)
 - Location (to move closer to the user)
 - Fault-tolerance (HW will fail eventually)
- Lecture 7
 - What is a distributed system?
 - How can it be categorized?
 - What are transparencies?

Distributed Systems Definition

Definition: A distributed system in its simplest definition is a group of computers working together as to appear as a single computer to the user

Distributed Systems Definition



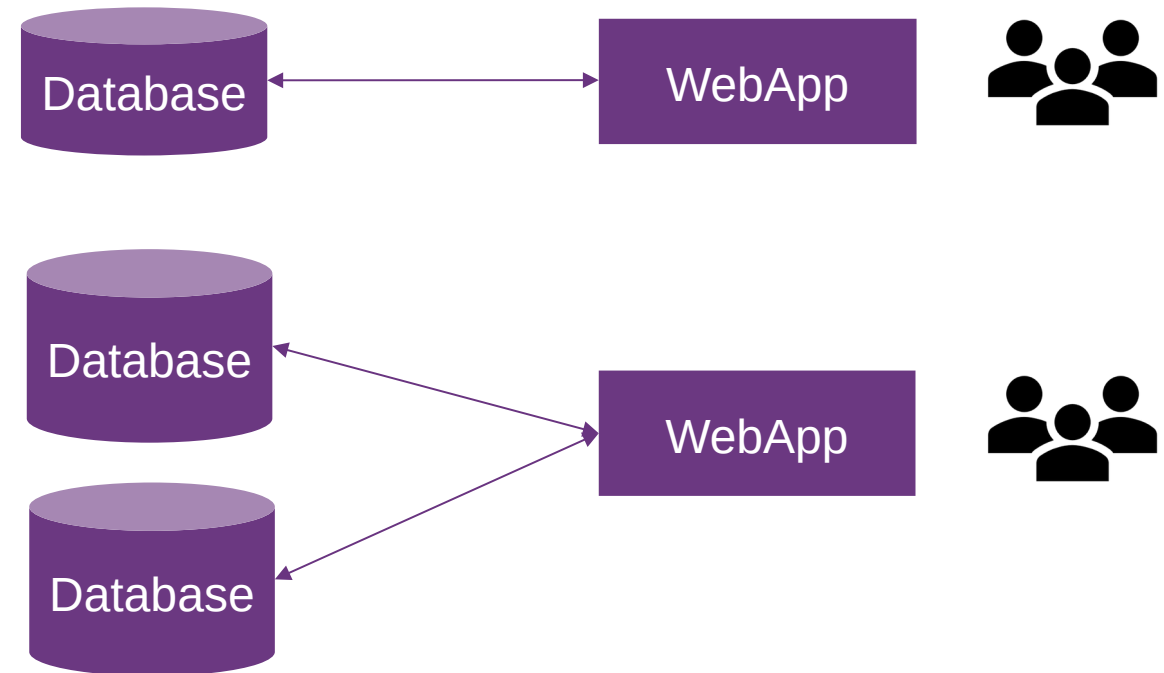
can you give me a definition of a distributed system



A distributed system is a network of independent computers that work together to achieve a common goal. In a distributed system, the computers communicate and coordinate their actions through message passing or other forms of communication, but they do not share a common memory or clock. Each computer in the system is typically responsible for its own processing and storage, and the computers may be geographically dispersed and connected through a network. The goal of a distributed system is to provide a higher level of performance, reliability, and scalability than can be achieved with a single, centralized system. Distributed systems are used in a wide variety of applications, including cloud computing, peer-to-peer file sharing, and distributed databases.

Distributed Systems Categorization

- It is useful to classify distributed systems as either tightly coupled, meaning that the processing elements, or nodes, have access to a common memory, and loosely coupled, meaning that they do not [[reference](#)]
- A homogeneous system is one in which all processors are of the same type; a heterogeneous system contains processors of different types
- Small-scale system: WebApp + database vs. large-scale with more than 2 machines
- Decentralized vs. distributed
 - Decentralized ~ distributed in the technical sense, but not owned by one actor



Distributed Systems Categorization

- Spring Term – Distributed Systems (DSy)

- Tightly/loosely coupled
- Heterogeneous systems
- Small-scale systems
- Distributed systems

(we will also talk about blockchains in this lecture)

- Fall Term – Blockchain (BlCh)

- Loosely coupled
- Heterogeneous systems
- Large-scale systems
- Decentralized systems

(we will also talk about distributed systems in this lecture, but DSy is highly recommended)

Distributed Systems Categorization

- Another classification
- **CAP theorem** - states that a distributed data store cannot simultaneously be consistent, available and partition tolerant
 - Consistency—Every node has the same consistent state
 - Availability— Every non-failing node always returns a response
 - Partition Tolerant—The system continues to be consistent even when network partitions
- With network partition - choose between consistency and availability
 - Is a system AP or CP?
- Blockchain and CAP
 - Both, if you wait for n blocks, CP, if you don't AP
- **Cassandra** AP
 - But can be configured CP
- **Discussion**
 - CAP theorem is too simplistic

Distributed Systems Categorization

- More Classifications
- Classification based on architecture
 - Client-server, peer-to-peer (P2P), or hybrid, or even more [\[link\]](#)
 - Software architecture [\[link\]](#)
- Classification based on communication model
 - Synchronous or asynchronous
- Classification: degree of transparency, fault tolerance, scalability, consistency (strong, eventual, and causal), data replication, data partitioning, heterogeneity
- No single universally applicable categorization of distributed systems

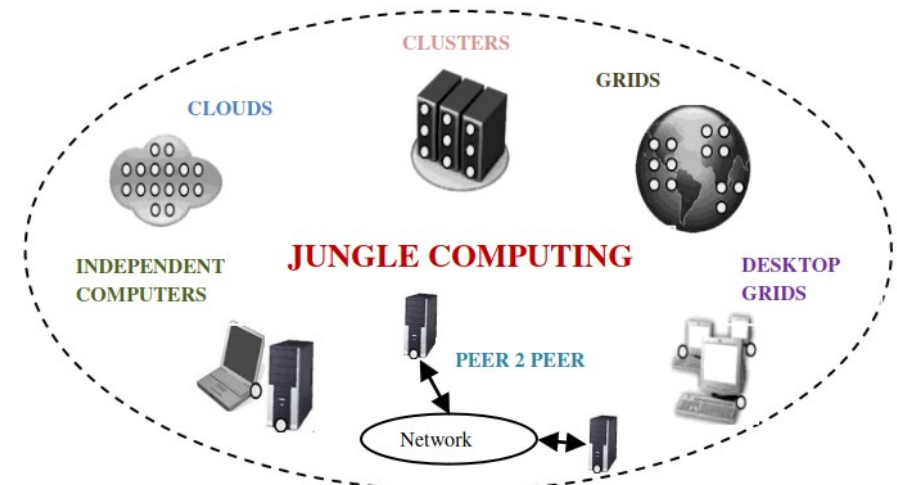


Figure 6.1: The jungle computing - a diverse collection of computing

[\[source\]](#)

Distributed Systems Categorization

“Controlled” Distributed Systems

- 1 responsible organization
- Low churn
- Examples:
 - Amazon DynamoDB
 - Client/server
- “Secure environment”
- High availability
- Can be homogeneous / heterogeneous

“Fully” Decentralized Systems

- N responsible organizations
- High churn
- Examples:
 - BitTorrent
 - Blockchain
- “Hostile environment”
- Unpredictable availability
- Is heterogeneous

Distributed Systems Categorization

“Controlled” Distributed Systems

- Mechanisms that work well:
 - Consistent hashing (DynamoDB, Cassandra)
 - Master nodes, central coordinator
- Network is under control or client/server → no NAT issues

“Fully” Decentralized Systems

- Mechanisms that work well:
 - Consistent hashing (DHTs)
 - Flooding/broadcasting - Bitcoin
- NAT and direct connectivity huge problem

Distributed Systems Categorization

“Controlled” Distributed Systems

- Consistency
 - Leader election (Zookeeper/[Zab](#), [Paxos](#), [Raft](#))
- Replication principles
 - More replicas: higher availability, higher reliability, higher performance, better scalability, but: requires maintaining consistency in replicas
- Transparency principles apply

“Fully” Decentralized Systems

- Consistency
 - Weak consistency: DHTs
 - Nakamoto consensus (aka proof of work, [\[link\]](#))
 - Proof of stake – Leader election, PBFT protocols
Is Bitcoin eventually consistent?
 - Some argue no, some argue it has even stronger guarantees [\[link\]](#)
- Replication principles apply to fully decentralized systems as well
- Transparency principles apply

Transparency in distributed systems

- Distributed system should hide its distributed nature
 - Location transparency – users should not be aware of the physical location
 - Access transparency - users should access resources in a single, uniform way
 - Migration, relocation transparency – users should not be aware, that resource have moved
 - Replication transparency – users should not be aware about replicas, it should appear as a single resource
- Concurrent transparency – users should not be aware of other users
- Failure transparency – users should **not** be aware of recovery mechanisms
- Security transparency – users should be minimally aware of security mechanisms
- More/other transparencies **here, here, here**
 - Depends on the context

Fallacies of Distributed Computing

- 8 fallacies to consider

- 1) The network is reliable

- Submarine cables

- 2) Latency is zero

- Ping to Australia is ~300ms, ping via Starlink +40ms

- 3) Bandwidth is infinite

- What is faster? Send a bike courier with an 8TB disk, that arrives 10h later, or send the data with a 1Gbit/s link? $8 * 1000 * 8 / (10 * 60 * 60) = 1.7\text{Gbit/s}$

- 4) The network is secure

- Assume someone is listening. Don't send sensitive data over the network

- 5) Topology doesn't change

- Ping to Australia, request can take different route than reply

- 6) There is one administrator

- Sometimes your route goes from one company to another rival company (UPC, Init7)

- 7) Transport cost is zero

- Someone build and maintains the network

- 8) The network is homogeneous

- From fiber to WiFi to cable, server, desktop, mobile