Goal of the lecture

Students should understand

• What a distributed system is:
  • its basic characteristics and ingredients

• Why we are interested in their design:
  • motivation, benefits
  • design goals

• What types we do distinguish:
  • high-level classification
  • application domains
Material

Mandatory

Chapter 1 of

*Distributed Systems, 3rd edition, version 01*
Maarten van Steen, Andrew Tanenbaum

available here
Leslie Lamport:

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”
What is a distributed system

Coulouris, et al.

“A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages”
What is a distributed system

van Steen, Tanenbaum

“A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system”
Characteristics

Definitions may vary, but there is consensus on their main characteristics

• No global notion of time
• No global notion of state
• Heterogeneous resources
• Network communication
• Independent failures
Basic ingredients

• **Processing elements (nodes)**
  • autonomous (often geographically separated)
    – no global clock and no global state
  • full-fledged computers
    – desktop, laptop, smart phones, plug computers, ...
  • heterogeneous

• **Communication subsystem or network**
  • modeled as a graph
  • subject to dynamic changes

• **Software**
  • operating system(s), runtime systems
  • generic system services (middleware)
  • application/domain specific services (purpose of the system)
Middleware

A layer on top of the OS in every node providing generic services that make the distributed system appear as a single machine to the applications such as:

• communication, transactions, orchestration, reliability, security, ...
Distributed systems (why)

- **Unavoidable**
  - Inherent distributed environment/application
  - Multiple data/resources/users at separated physical location

- **Associated benefits**
  - Reduced development/maintenance cost through modularity
  - Reduced operational cost through resource sharing
  - Improved performance through replication
  - Improved dependability through redundancy

- **Design goals**
  - In general, realizing these benefits
Design goals

Since distributed systems are complex, expensive, exist for a long time, and span a large geographical range, their design needs to be

- **Efficient and effective**
  - resource combination and sharing
- **Transparent**
  - hiding their (internal) complexity
  - makes them easier to understand and use
- **Scalable**
  - coping with growth,
- **Open**
  - allowing usage by, extension with, integration into and built from 3rd party components and systems.
To obtain *distribution transparency*, a number of more specific transparencies all have to be achieved. In practice, especially failure transparency is extremely hard, if not impossible, to achieve (cf. Lamport definition).

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how objects are accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where an object is located</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that an object may be moved to another location while in use</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that an object may move to another location</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that an object is replicated</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that an object may be shared by several independent users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of an object</td>
</tr>
</tbody>
</table>
Scalability issues

The ability to cope with growth

• multiple dimensions:
  • size, geographical spread, administrative domains

• shift from a few dedicated high-performance resources to massive commodity resources:
  • vertical scaling versus horizontal scaling

• realized by a multitude of techniques:
  • hiding communication latency
  • partitioning and distribution
  • replication and load balancing

Remark: detailed discussion in separate lecture.
DNS scaling example: partition and distribute

DNS resolution (map domain name to IP-address)
- name space partitioned into zones
- administration differs per zone
- one authoritative server per zone
- system spans the entire globe
Openess issues

Allowing usage by, extension with, and integration into 3rd party components and systems.

- interoperability, composability, extensibility
  - services with well-defined interfaces and service level agreements (SLAs)
    - specifications neutral and complete
  - tool support for generating boiler-plate code
- separating policy from mechanism
  - rich facilities for (run-time) configuration
    - i.e., policy specification
  - self-configuration to assist users
Pitfalls

The 8 fallacies of distributed computing
Peter Deutsch (7) , Brian Gosling (1)

1. The network is reliable.
2. The network is secure.
3. The network is homogeneous.
4. Topology doesn't change.
5. Latency is zero.
6. Bandwidth is infinite.
7. Transport cost is zero.
8. There is one administrator.

A good architect determines:

- the importance of each of these properties for the system at hand
- provides mechanisms to deal with the absence of such a property

For background information see wikipedia page
Types of distributed systems

- **Systems for high performance computing**
  - cluster computing
  - grid computing
  - cloud computing

- **Distributed information systems**
  - distributed transactions
  - enterprise application integration

- **Pervasive systems**
  - ubiquitous computing systems
  - mobile computing systems
  - sensor networks
Systems for high performance computing

- **Cluster**
  - nodes run same OS and are connected by a high-speed network
  - master-worker nodes or
  - “fully” symmetric
    - process migration to achieve single-system image

- **Grid**
  - federation of systems; resources in multiple administrative domains
    - security issues,
    - virtual organizations through layered architecture

- **Cloud**
  - virtualized resources
    - pay per resource, guarantees through SLAs,
  - organized in service layers: IaaS, PaaS, SaaS
Distributed information systems

• **Distributed transaction processing**
  • nested transactions complicate maintaining transaction semantics
    – the ACID properties
  • coordination support by middleware
    – e.g. TP-monitor, special component for distributed commit

• **Enterprise application integration**
  • resource/service sharing between various applications in an enterprise
    – service-oriented approach (SOA)
  • message-oriented middleware (MOM) support
    – enterprise service bus (ESB)
Nested transaction processing

- nested transaction booking a city trip
- subtransactions

Diagram:
- Client application
- TP monitor
- Server
- Airline database
- City transport database
- Hotel database
- Middleware service/component
Multiple client applications built by combining services (server-side apps) offered at various branches of an enterprise
Pervasive systems

- **Ubiquitous computing systems**
  - devices are networked, context-aware, autonomous, intelligent,
  - interaction with users implicit and unobtrusive

- **Mobile computing systems**
  - wireless communication
  - changing topology and service availability

- **Sensor networks**
  - resource-constrained, specifically w.r.t. energy
  - programming support for communication
    - abstract neighborhoods, content-based addressing
  - emergent functionality through cooperation
    - aggregation, in-network data processing

Extensively treated in course 2IMN15, Internet of Things
The bus shelter ad uses facial recognition software with an HD camera to determine whether a man or woman is standing in front of the screen.

... 

Men are denied the choice to view the full 40-second ad in order to highlight the fact that women and girls across the world are denied choices and opportunities on a daily basis due to poverty and discrimination.
In network data processing (bottom)

Sensor data is sent directly to operator

Each sensor can process and store data

Sensors send only answers

Query

Operator's site
Distributed systems (where)

Everywhere
Application domains

- Financial applications
- Manufacturing
- Reservation
- Transportation / Traffic
- Telecom
- Multimedia
- Social media
- Health care
- Surveillance
- Smart environments
- Automotive / Aviation
- Webshops
- ...