Distributed Systems
“Introduction to distributed systems”

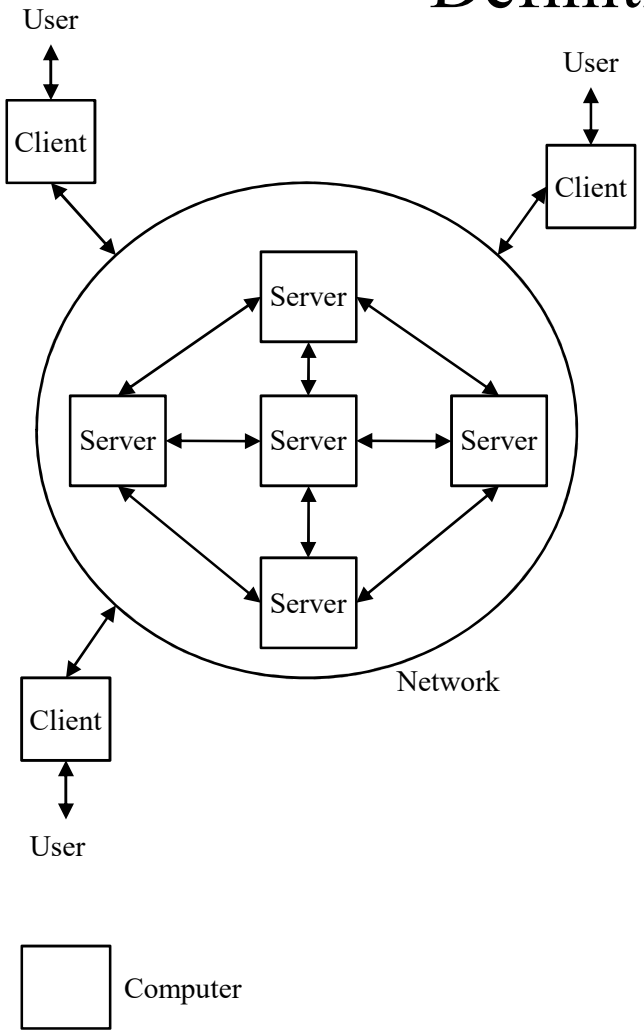
Mathieu Delalandre
University of Tours, Tours city, France
mathieu.delalandre@univ-tours.fr

Lecture available at http://mathieu.delalandre.free.fr/teachings/dsystems.html

Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Definitions and motivations (1)



"A distributed system is a collection of independent computers that appear to the user as a single computer." [A. Tanenbaum]

"A distributed system is one in which components located at networked computers communicate and coordinate their actions by passing messages. This definition leads the following characteristics of distributed systems: concurrency of components, lack of global clock and independent failures of components." [G. Coulouris]

"A distributed system is one on which I can't do my work due to a computer that has failed and I never heard of." [L. Lamport]

Definitions and motivations (2)

A distributed system has the following features:

No common physical clock: this is an important assumption because it introduces the element of distribution in the system and gives rise to the inherent asynchrony amongst the processors.

No shared memory: this is a key feature that requires message-passing for communication. This feature implies the absence of the common physical clock.

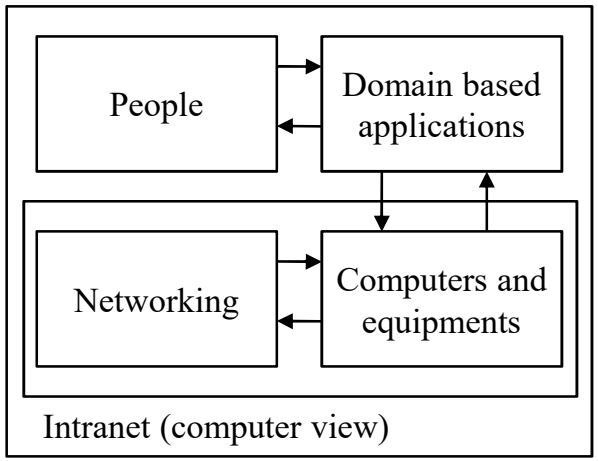
Geographical separation: the geographically wider apart that the processors are, the more representative is the system of a distributed system.

Autonomy and heterogeneity: the processors are loosely coupled in that they have different speeds and each can be running a different operating system. They are usually not part of a dedicated system, but cooperate with one another by offering services or solving a problem jointly.

Definitions and motivations (3)

Distributed systems are taking part of Intranet/Internet networks.

Information system (public view)



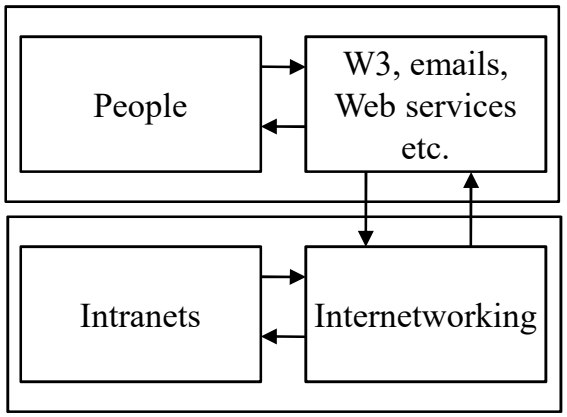
An Information System (IS): is any combination of information technology and people's activities using that technology to support operations, management, and decision-making. In this sense, the term is used to refer not only to the Information and Communication Technology (ICT), but also to the way in which people interact with this technology.

An intranet: is a private computer network that uses the Internet Protocol (IP) technology to securely share any part of an organization's information or network operating system within that organization. The term is used in contrast to Internet, a network between organizations, and instead refers to a network within an organization.

Definitions and motivations (4)

Distributed systems are taking part of Intranet/Internet networks.

Internet or web (public view)



Internet (computer view)

The World Wide Web: (i.e. WWW, W3 or Web) is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.

Internet: is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP). It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope. These networks are linked by a broad array of electronic, wireless and optical networking technologies. The Internet carries a vast range of information resources and services, such as the W3 and web services.

Definitions and motivations (5)

The motivations for using a distributed system are listed below.

Inherently distributed computation: in many applications such as money transfer in banking, or reaching consensus among parties that are geographically distant, the computation is inherently distributed.

Resource sharing: resources such as peripherals, databases, libraries, etc. cannot be fully replicated at all the sites because it is often neither practical nor cost-effective. Further, they cannot be placed at a single site because access to that site might prove to be a bottleneck. Therefore, such resources are typically distributed across the system.

Access to geographically remote data and resources: special resources such as supercomputers exist only in certain locations, and to access such supercomputers, users need to log in remotely.

Enhanced reliability: a distributed system has the inherent potential to provide increased reliability because of the possibility of replicating resources and executions, as well as the reality that geographically distributed resources are not likely to crash/malfunction at the same time under normal circumstances.

Increased performance/cost ratio: by resource sharing and accessing geographically remote data, the performance/cost ratio is increased. Such a configuration provides a better performance compared to special parallel machines.

Scalability and networking: as the processors are usually connected by a wide-area network, adding more processors does not pose a direct bottleneck for the communication network.

Heterogeneity and incremental expandability: heterogeneous processors may be easily added into the system without affecting the performance, as long as those processors are running the same middleware algorithms. Similarly, existing processors may be easily replaced by other processors.

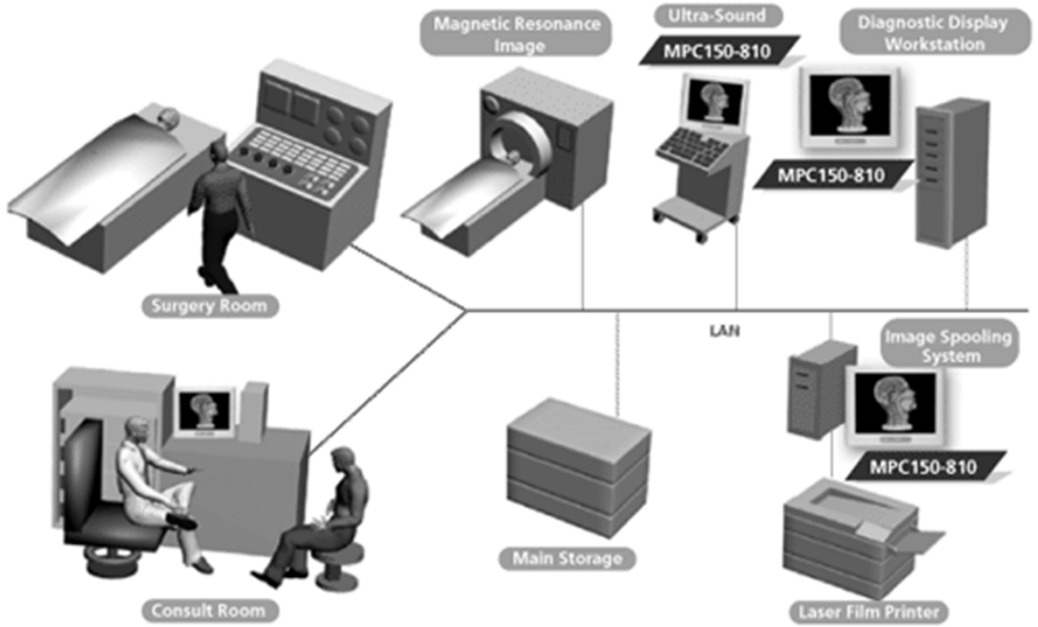
Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Use-cases and application domains (1)
“Hospital Information System”

The domain point of view:

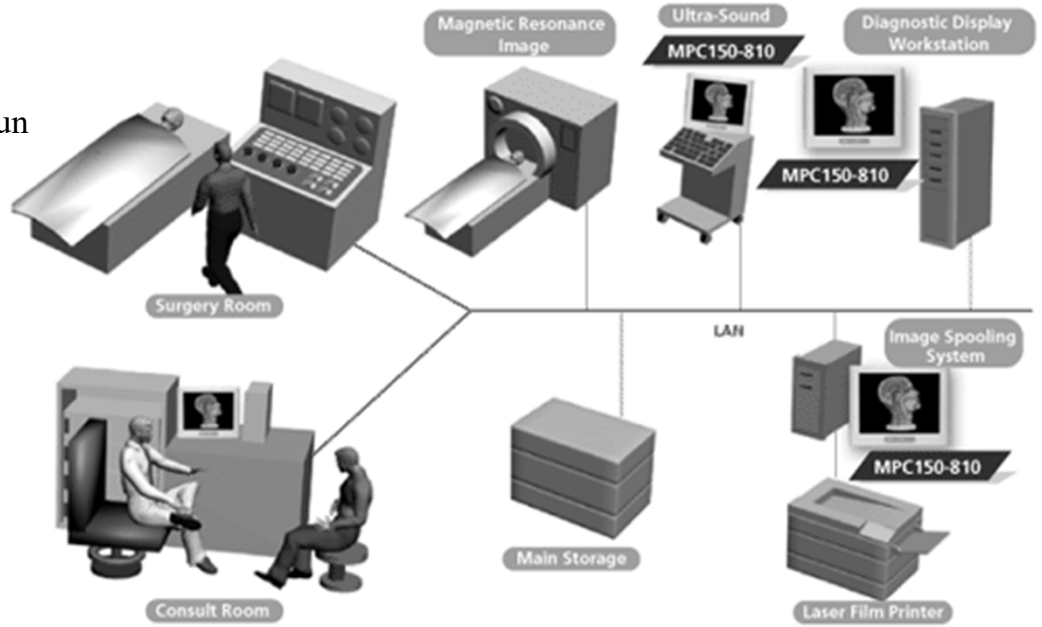
- resource management (room places, medicaments stocks and equipments, etc.),
- planning (medical & administrative staffs, medical interventions, etc.),
- information (medical and administrative files, people communication, newsletter and reports, etc.).



Use-cases and application domains (2)
“Hospital Information System”

The Intranet point of view:

- computer (desktops “Mac, PC”, embedded systems “sun station, mainframes”, etc.),
- equipment (scanners, monitoring, etc.),
- databases (administrative and medical files, etc.),
- communication & network (phone & beeper, Ethernet and serial bus, gateway to Internet, etc.).



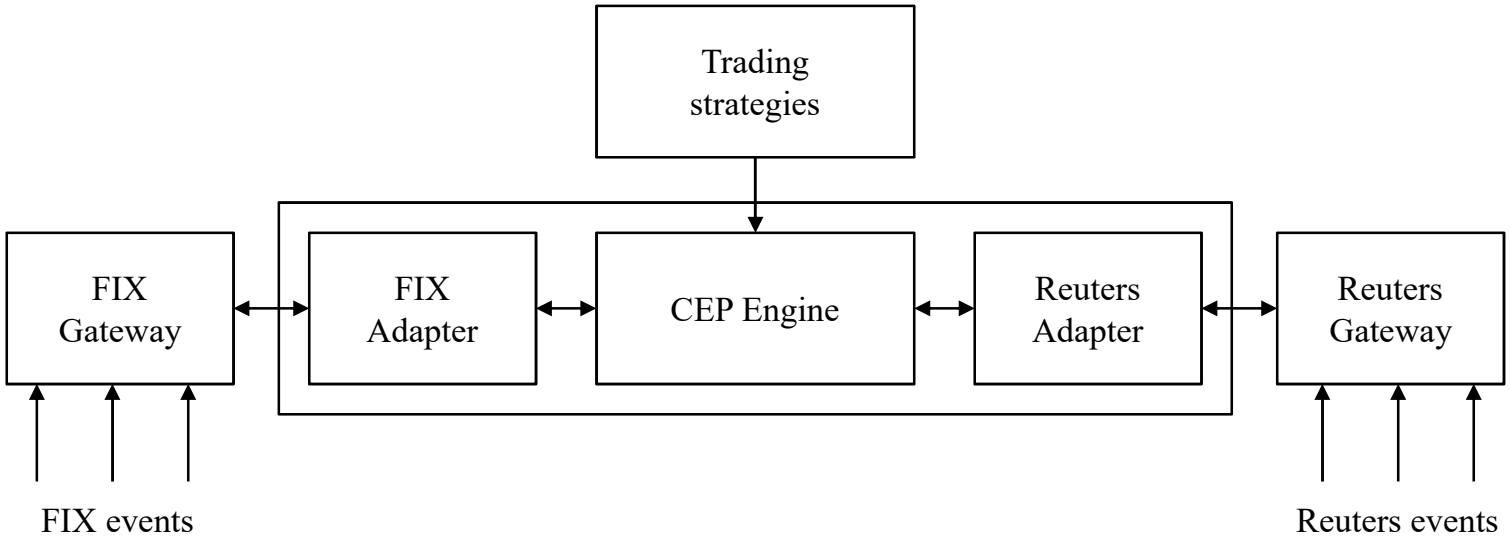
Use-cases and application domains (3)
“Financial trading”

Within a typical financial trading system, we have:

- a series of event feed into a given financial institution, in a variety of formats (e.g. Reuters data events, FIX events, etc.),

- the use of adapters which translate heterogeneous formats into a common internal format,

- the trading system must deal with a variety of event streams, requiring real-time processing to detect patterns that indicate trading opportunities; the Complex Event Processing (CEP) engine offers a way of composing event occurrences together into logical, temporal or spatial patterns.



Use-cases and application domains (4)
“Application domains”

The application domains are listed below.

The information society: the growth of the W3 as a repository of information and knowledge, the development of web search engines (e.g. Google, Bing).

Finance and commerce: the growth of the eCommece as exemplified by companies such as Amazon and eBay, and underlying payments technologies (e.g. PayPal).

Creative industries and entertainment: the emergence of online gaming as a novel and highly interactive form of entertainment with Massively Multiplayer Online Games or MMOGs (e.g. WoW).

Education: the emergence of e-learning through for example web-based tools such as virtual learning environments (e.g. Moodle).

Transport and logistic: the use of location technologies such as GPS in route finding systems and more general traffic management systems (e.g. Waze).

Science: the emergence of the Grid as a fundamental technology for eScience, including the use of complex networks of computers to support the storage (e.g. SETI project).

Healthcare: the growth of health informatics as a discipline with emphasis on online electronic patient records and related issues of privacy (e.g. smartwatch)

Environmental management: the use of (networked) sensor technology to both monitor and manage the natural environment (e.g. smart farming).

Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Trends in distributed systems

Pervasive networking and the modern Internet: the modern Internet is a vast interconnected collection of computer networks of many different types increasing all the time (WiFi, WiMAX, Bluetooth, etc.). The net results is that networking has become a pervasive resource and devices can be connected at any time and in any place.

Distributed multimedia systems: the benefits of distributed multimedia services are considerable in that a wide range of new applications can be provided (television broadcast, video-on-demand, music libraries, audio and video conferencing, etc.). The crucial characteristic of continuous media is that they include a temporal dimension and need to preserved their real-time relationships.

Distributed artificial intelligence: is an approach for complex learning, planning and solving of decision making problems. It is designed with parallel and distributed systems, able to exploit large scale computation and spatial distribution of computing resources. This allows to solve problems that require the processing of very large data sets with a huge complexity.

Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Relations to parallel systems (1)

Flynn's taxonomy identifies four processing modes. It is instructive to examine this classification to understand relation of distributed systems to parallel systems.

Single instruction stream, single data stream (SISD): this mode corresponds to the conventional processing in the von Neumann paradigm with a single CPU, and a single memory unit connected by a system bus.

Multiple instruction stream, single data stream (MISD): this mode corresponds to the execution of different operations in parallel on the same data. This is a specialized mode of operation with limited but niche applications, e.g., visualization.

Single instruction stream, multiple data stream (SIMD): this mode corresponds to the processing by multiple homogenous processing units which execute in lock-step on different data items. It targets applications that involve operations on large arrays and matrices, such as scientific applications.

Multiple instruction stream, multiple data stream (MIMD): in this mode, the various processing units / processors execute different code on different data. This is the mode of operation in distributed systems as well as in the vast majority of parallel systems.

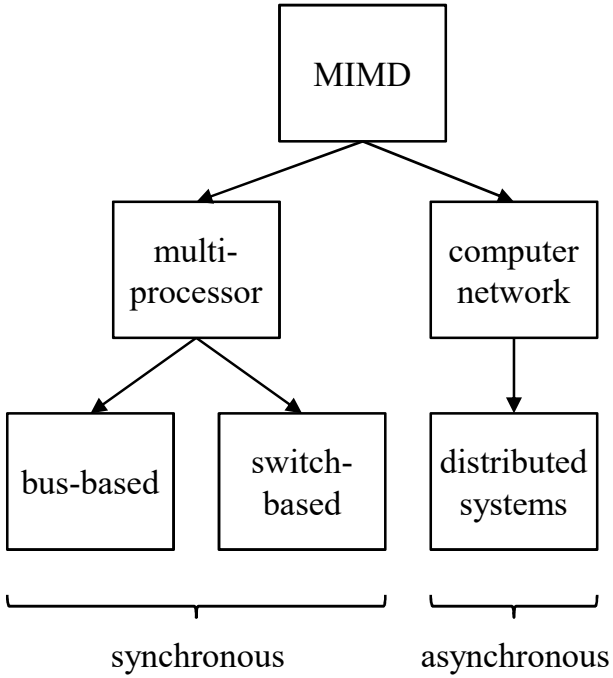
Data \ Instructions	Single Instruction (SI)	Multiple Instructions (MI)
Single Data (SD)	SISD uni-core processor	MISD processor with pipeline architecture
Multiple Data (MD)	SIMD e.g. uni-core processor with large register	MIMD e.g. multiprocessor

Relations to parallel systems (2)

Multiple instruction stream, multiple data stream (MIMD): in this mode, the various processing units / processors execute different code on different data. This is the mode of operation in distributed systems as well as in the vast majority of parallel systems.

- **Synchronous systems:** are systems where operations (instructions, calculations, etc.) are coordinated by one, or more, centralized clock signals.

- **Asynchronous systems:** in contrast, has no global clock.

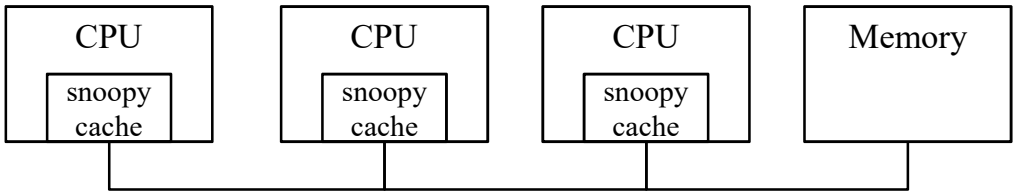


Relations to parallel systems (3)

	Processing mode	Hardware				Software
		Clock	Commu-nication	Memory		
bus-based multiprocessors	MIMD	synchronous	bus	shared memory	tightly coupled	tightly coupled
switched multiprocessors			switch			
network operating system		asynchronous	LAN	shared files	loosely coupled	loosely coupled
distributed system			WAN	message exchange		tightly coupled

synchronous systems

e.g. bus based multiprocessor



Memory	Description	Global memory state
cache	store reading / writing events of CPU only	incoherent
snoopy cache	monitor all events in the bus	coherent

Relations to parallel systems (4)

	Processing mode	Hardware				Software
		Clock	Commu-nication	Memory		
bus-based multiprocessors	MIMD	synchronous	bus	shared memory	tightly coupled	tightly coupled
switched multiprocessors			switch			
network operating system		asynchronous	LAN	shared files	loosely coupled	loosely coupled
distributed system			WAN	message exchange		tightly coupled

asynchronous systems

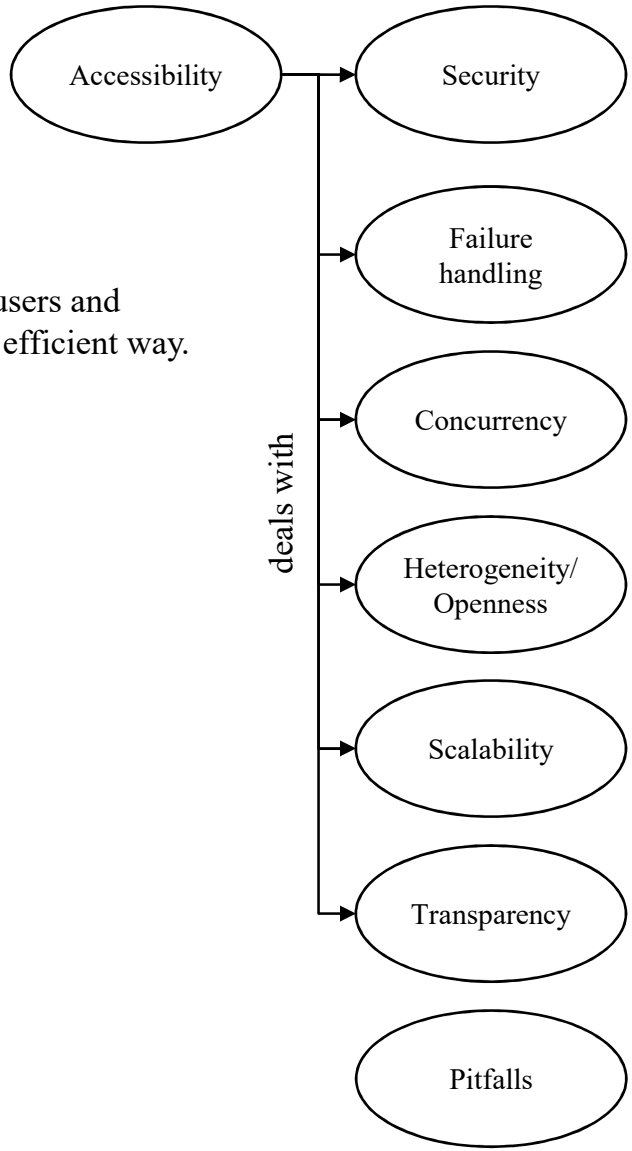
		Network operating systems	Distributed systems
Software	IPC	no	yes
	Interoperability		
	Shared data		
	Time synchronization		
	System coordination		

Introduction
to distributed systems

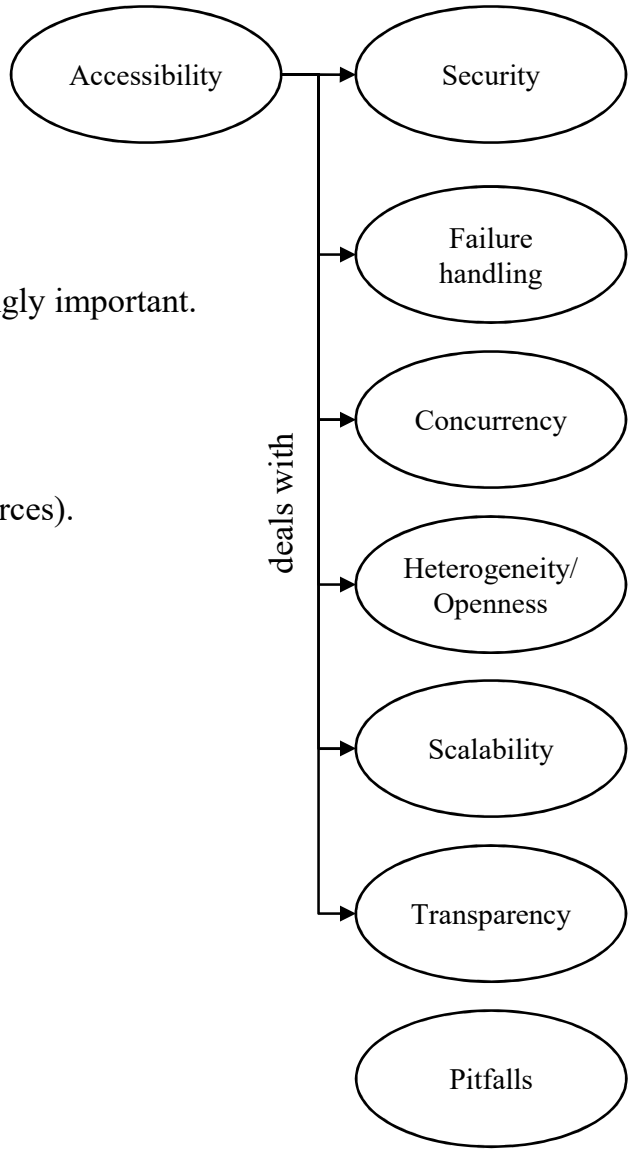
1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Goals and challenges of distributed systems (1)

Accessibility: the main goal of distributed systems is to make it easy, for the users and applications, to access remote resources and to share them in a controlled and efficient way.



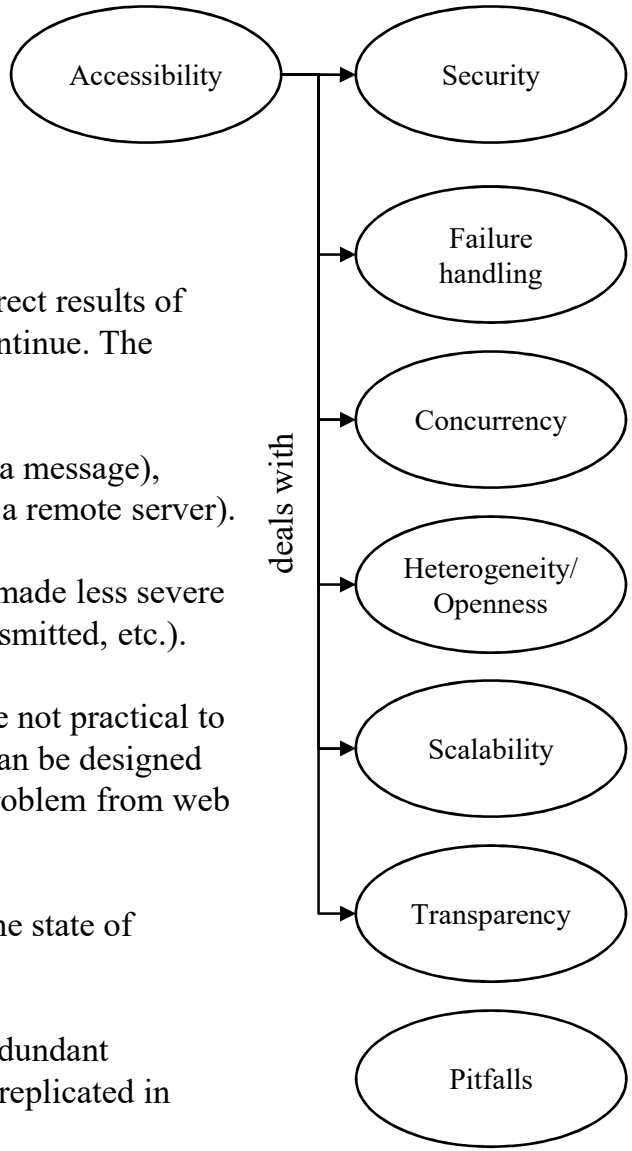
Goals and challenges of distributed systems (2)



Security: as connectivity and sharing increase, security is becoming increasingly important.

Security for information resources has three components:
- confidentiality (access control and protection).
- integrity (protection against alteration or corruption).
- availability (protection against interference with the means to access resources).

Goals and challenges of distributed systems (3)



Failure handling: computer systems sometimes fail that could produce incorrect results of programs. These failures are partial i.e. some components fail while others continue. The handling of failures is difficult and relies on different techniques.

- **Detecting failures:** some failures can be easily detected (e.g. checksum of a message), some other failures are difficult or even impossible to detect (e.g. crashed of a remote server).

- **Masking failures:** some failures that have been detected can be hidden or made less severe (e.g. messages can be retransmitted, file data can be "rolled back" and retransmitted, etc.).

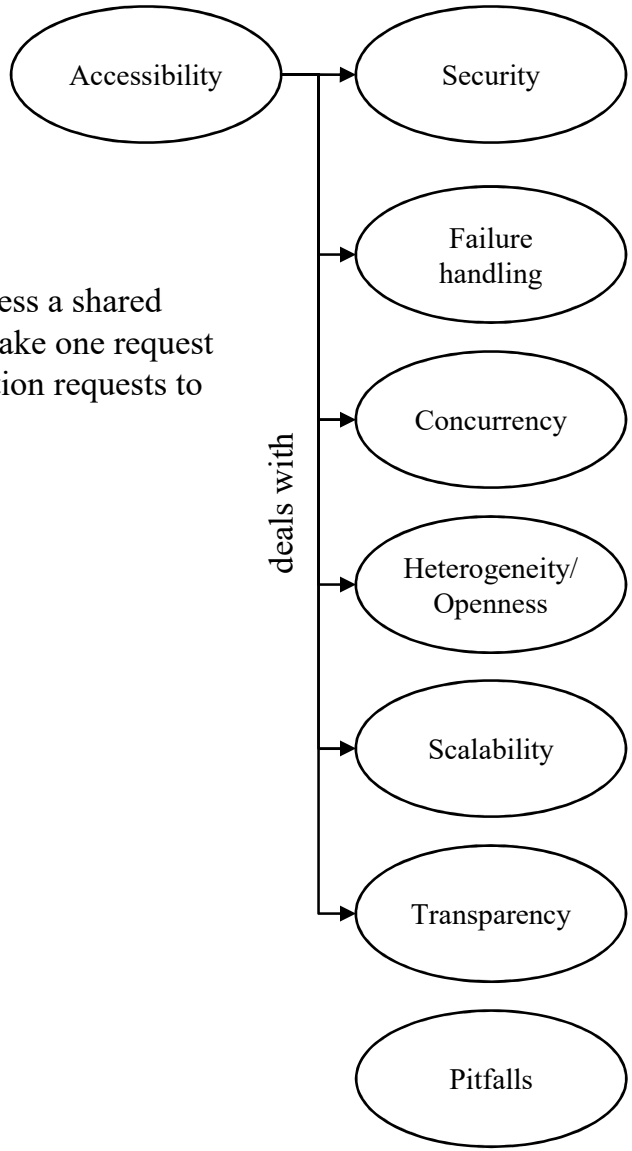
- **Tolerating failures:** for most of the applications in the Internet, it would be not practical to attempt to detect and hide all the failures that might be large. Applications can be designed to tolerate failures involving the users tolerating them as well (e.g. access problem from web browser to a web server, etc.).

-**Recovery from failures:** recovery involves the design of software so that the state of permanent data can be recovered or "rolled back" after a server has crashed.

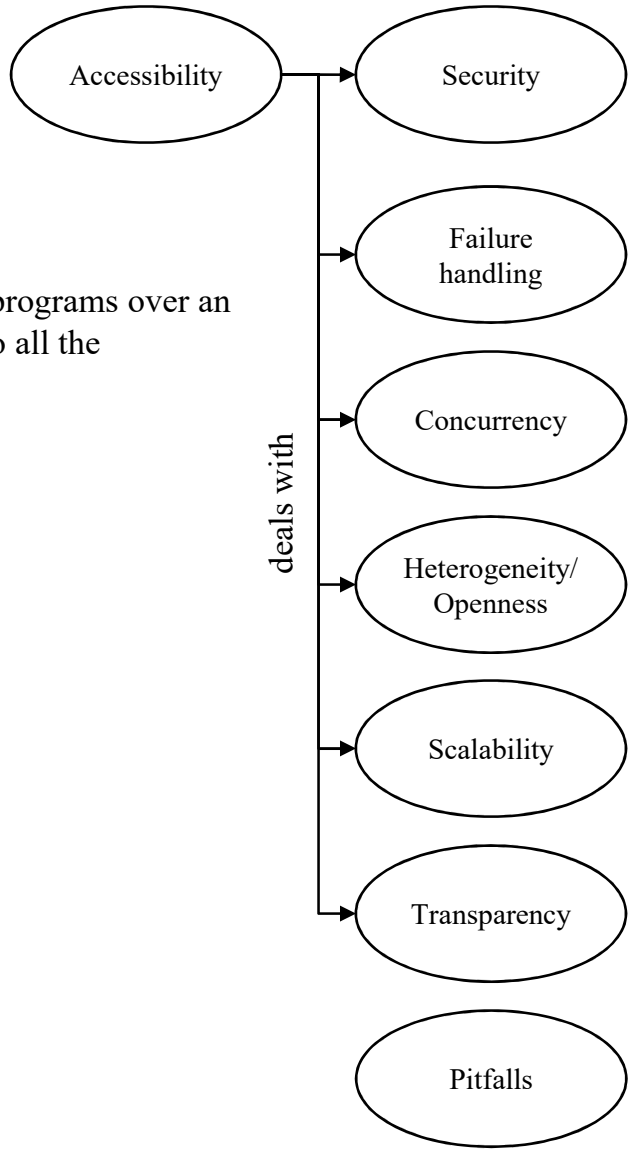
- **Redundancy:** applications can be made tolerate to failures by the use of redundant components (e.g. different routes to access a same router, a database can be replicated in several servers, etc.).

Goals and challenges of distributed systems (4)

Concurrency: there is possibility that several applications will attempt to access a shared resource at the same time. The process that manages a shared resource could take one request at a time. Therefore, distributed components generally allow multiple application requests to be processed concurrently.



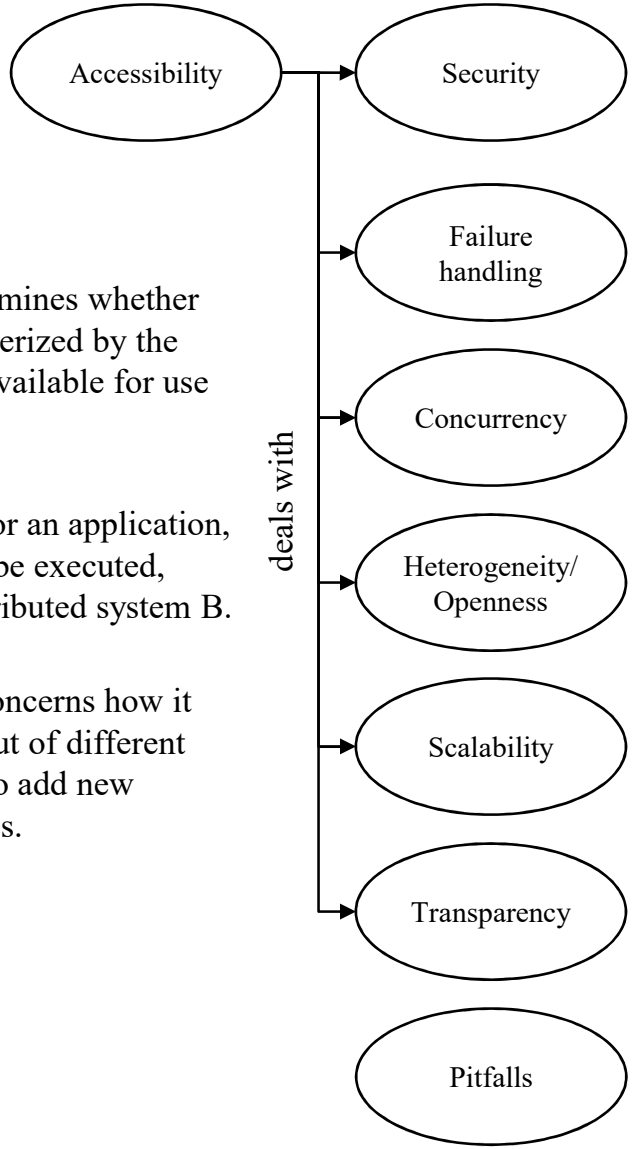
Goals and challenges of distributed systems (5)



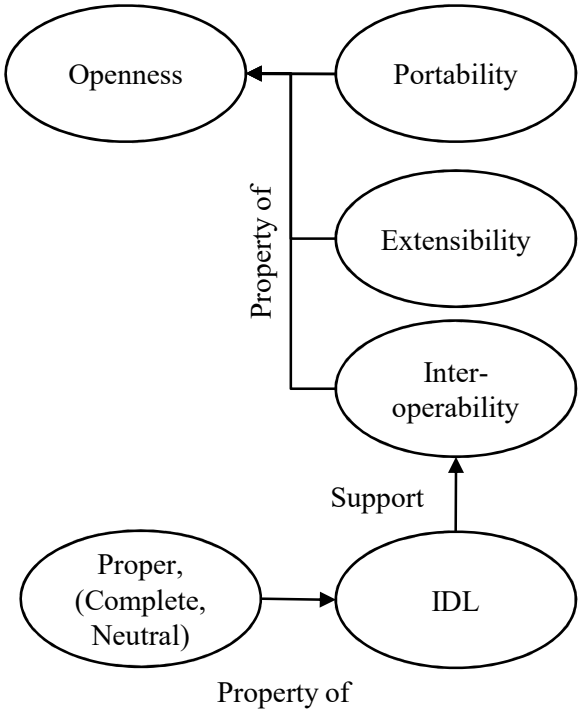
Heterogeneity: the Internet enables users and applications to access and run programs over an heterogeneous collection of computers and networks. Heterogeneity applies to all the following:

- networks,
- computer hardware,
- Operating Systems,
- programming languages,
- design issues (i.e. implemented by different developers),
- etc.

Goals and challenges of distributed systems (6)



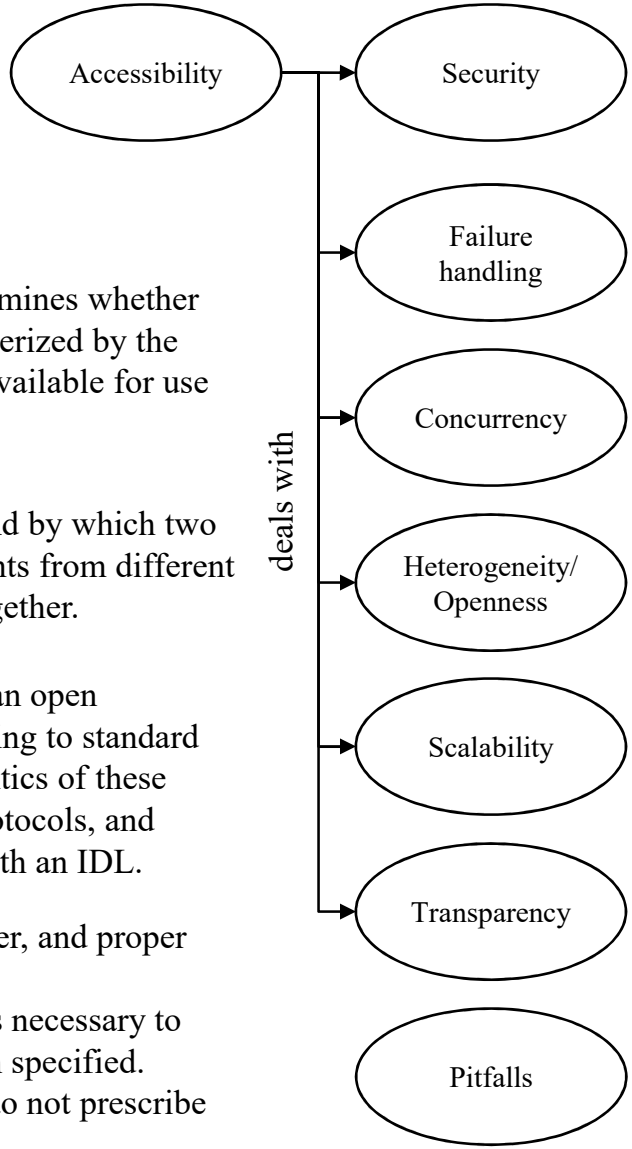
Openness: the openness in a distributed system is the characteristic that determines whether the system can be extended and re-implemented in a various way. It is characterized by the degree to which new resources-sharing and services can be added and made available for use by a variety of applications.



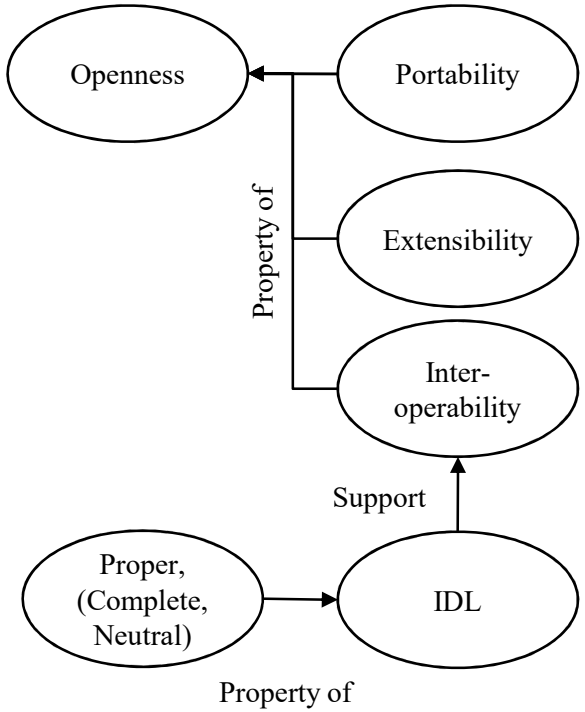
Portability: characterizes the property for an application, developed for a distributed system A, to be executed, without modifications on a different distributed system B.

Extensibility: for a distributed system concerns how it should be easy to configure the system out of different components, and how it should be easy to add new components or replacing the existing ones.

Goals and challenges of distributed systems (7)



Openness: the openness in a distributed system is the characteristic that determines whether the system can be extended and re-implemented in a various way. It is characterized by the degree to which new resources-sharing and services can be added and made available for use by a variety of applications.

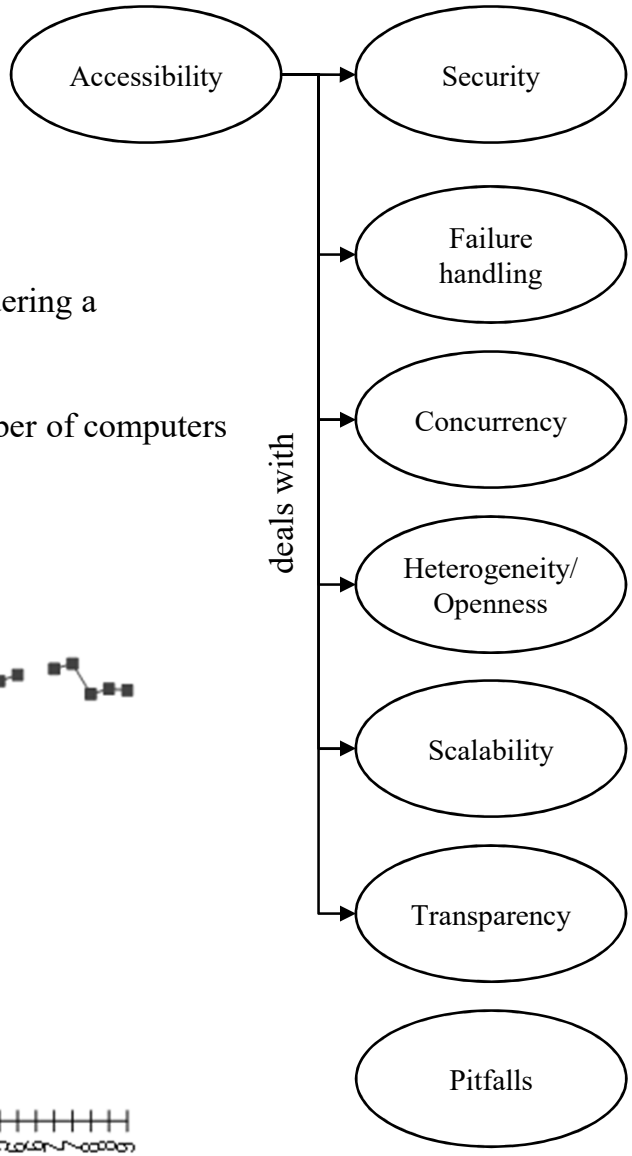


Interoperability: characterizes the extend by which two implementations of systems or components from different manufacturers, can co-exist and work together.

IDL (Interface Definition Language): an open distributed system offers services according to standard rules that describes the syntax and semantics of these services. Such rules are formalized in protocols, and specified through interfaces described with an IDL.

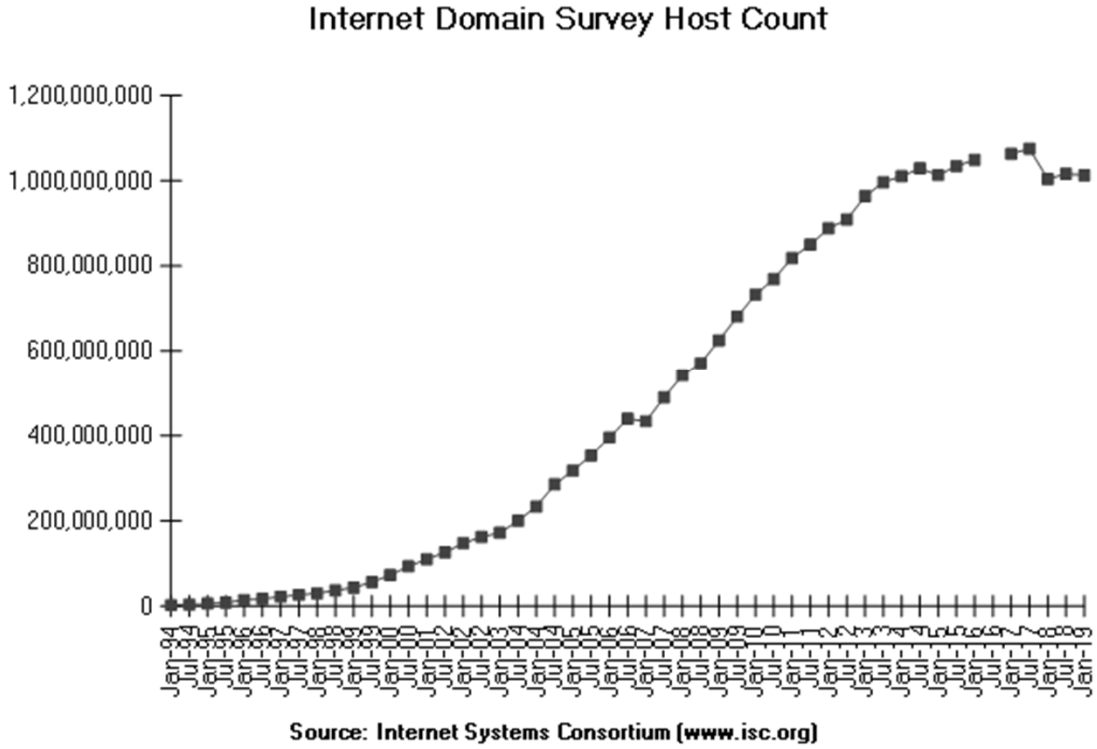
Proper: the IDL definition must be proper, and proper specifications are complete and neutral.
-**Complete:** means that everything that is necessary to make an implementation has indeed been specified.
-**Neutral:** means that the specifications do not prescribe what an implementation should look like.

Goals and challenges of distributed systems (8)

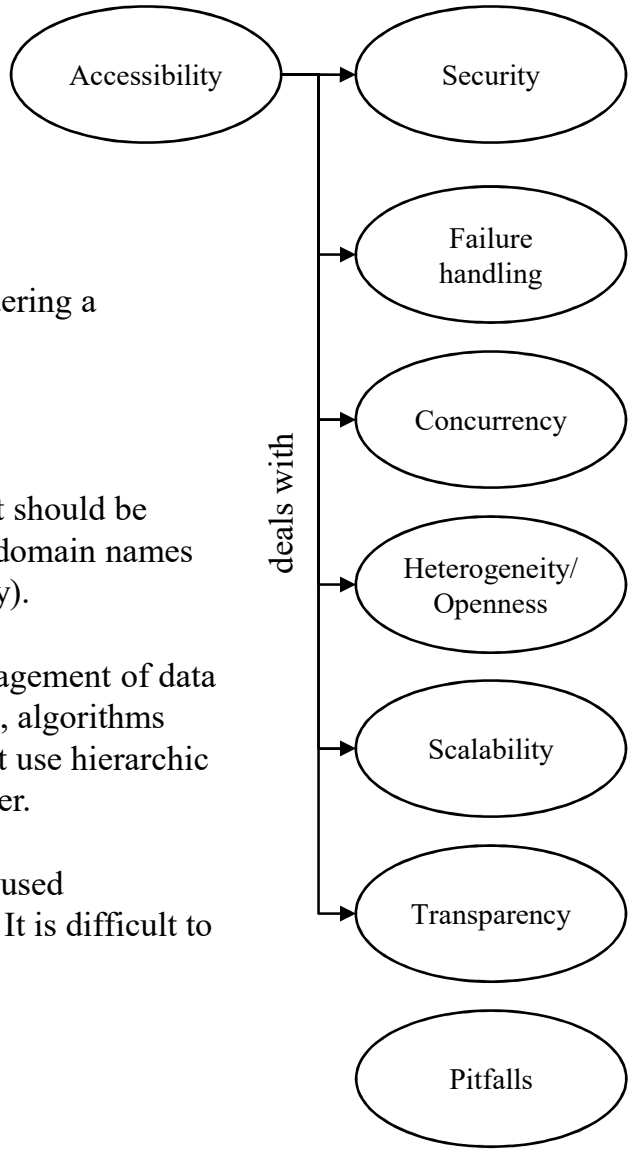


Scalability: a system is described as scalable if it will remain effective considering a significant increase in the number of resources and users.

The Internet provides an illustration of a distributed system in which the number of computers and services has increased dramatically.



Goals and challenges of distributed systems (9)



Scalability: a system is described as scalable if it will remain effective considering a significant increase in the number of resources and users.

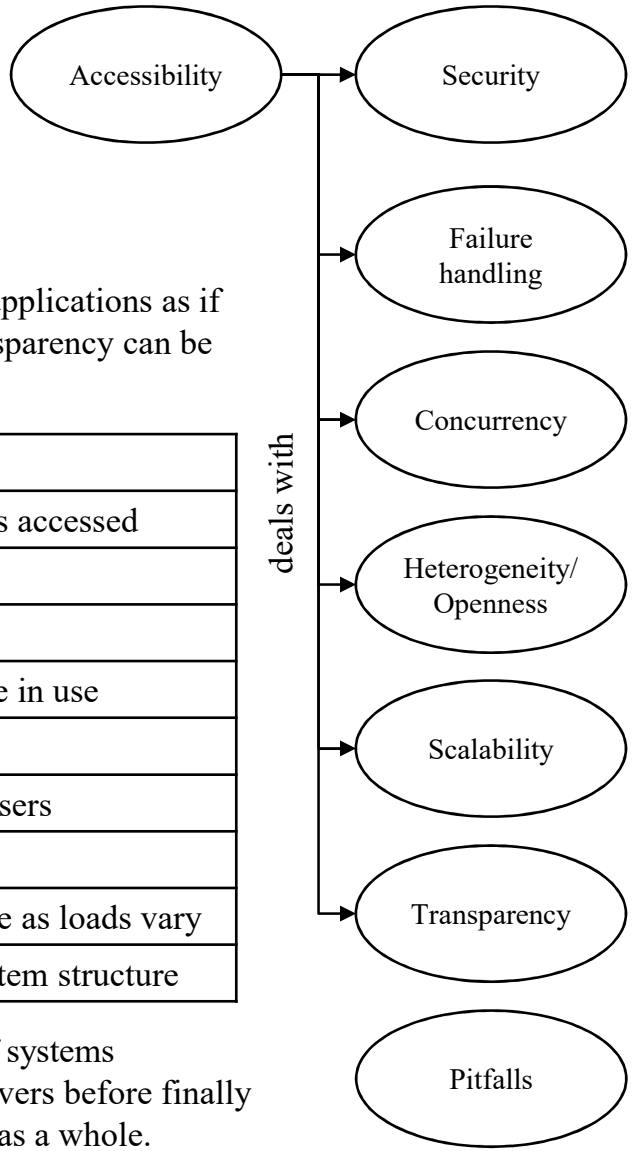
The design of scalable distributed systems presents the following challenges.

- **Avoiding performance bottlenecks:** as the demand for a resource grows, it should be possible to extend the system to meet it (e.g. bottle necks is supported with domain names by partitioning the name table between different servers administered locally).

- **Controlling the performance loss:** as an example, if we consider the management of data (e.g. corresponding table between domain names and IP addresses) of size n, algorithms that use a linear structure scale with a complexity of O(n) whereas those that use hierarchic structure scale in O(log(n)). Considering scalability, hierarchic ones are better.

- **Preventing software resources running out:** as an example, the data size used to store Internet addresses was 32 bits late 1970s, then 128 bits early 2000s. It is difficult to predict the demand that will be put on a system years ahead.

Goals and challenges of distributed systems (10)

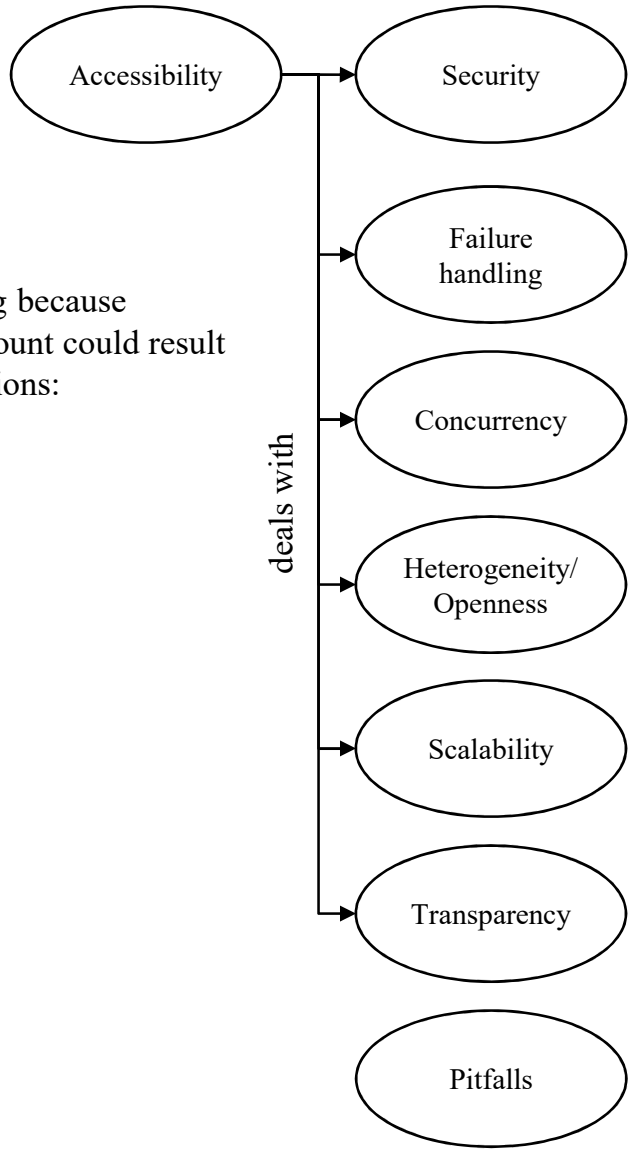


Transparency: a distributed system that is able to present itself to users and applications as if it were only a single computer system is said transparent. The concept of transparency can be applied to several aspects of a distributed system:

Transparency	**Description**
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is duplicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Performance	Allow the system to be reconfigured to improve performance as loads vary
Scalability	Allow the system to expand in scale without change the system structure

There is a tradeoff between a high degree of transparency and performance of systems e.g. when connection failures occur, applications repeatedly try to connect servers before finally giving up. Consequently, to mask a connexion failure slows down the system as a whole.

Goals and challenges of distributed systems (11)



Pitfalls: developing a distributed system differs from traditional programming because components are dispersed across network. No tacking this dispersion into account could result in mistakes. We can formulated these mistakes as the following false assumptions:

1. the network is reliable,

2. the network is secure,

3. the network is homogeneous,

4. the topology does not change,

5. the latency is zero,

6. bandwidth is infinite,

7. transport cost is zero,

8. there is one administrator,

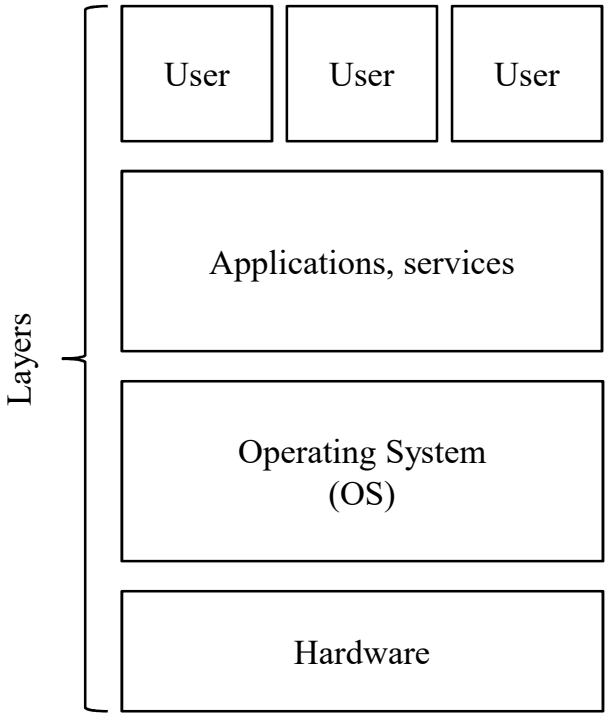
9. …

Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

The layered model and design issues (1)

The layered model at the OS level



Users: could share a same computer (through session / terminal).

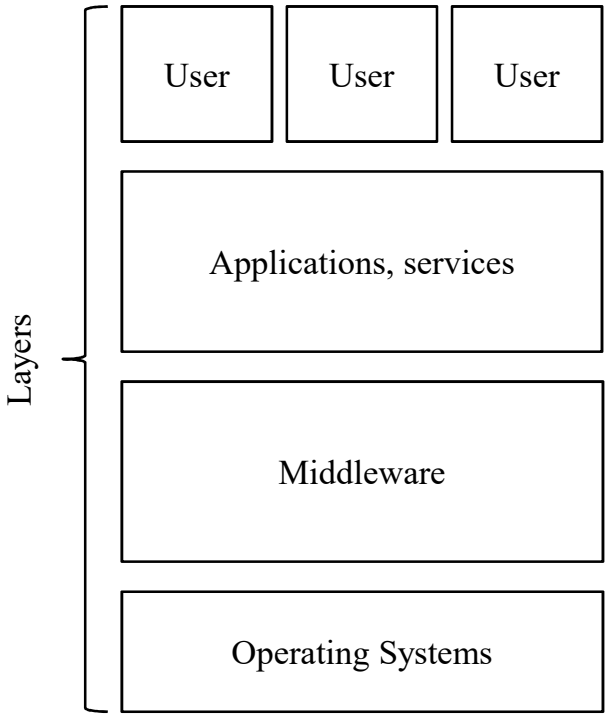
Application: designed to help the user to perform a singular or multiple related specific tasks (e.g. office, programming toolkit, web browser, etc.).

Operating system: interface between hardware and user, which is responsible for the management and coordination of activities and the sharing of the resources of a computer, that acts as a host for computing applications run on the machine.

Hardware: physical electronic components and mechanical parts that make up a piece of computer equipment (keyboard, disk drive, CPU, motherboard, etc.).

The layered model and design issues (2)

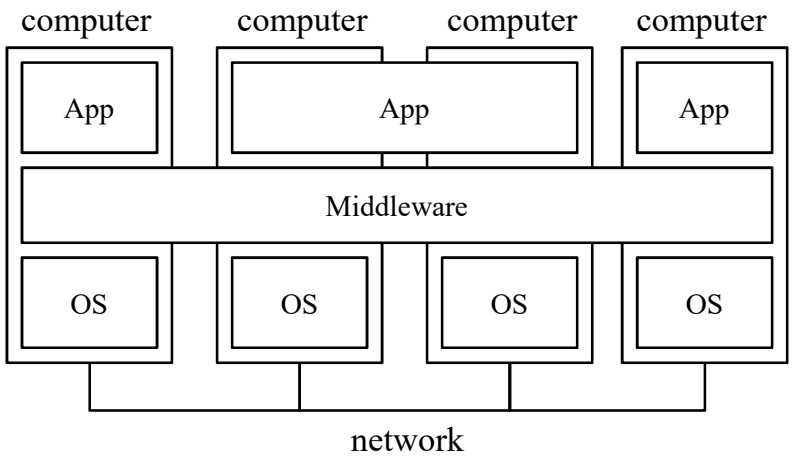
The layered model at the distributed system level



Users: could share a same computer (through session / terminal).

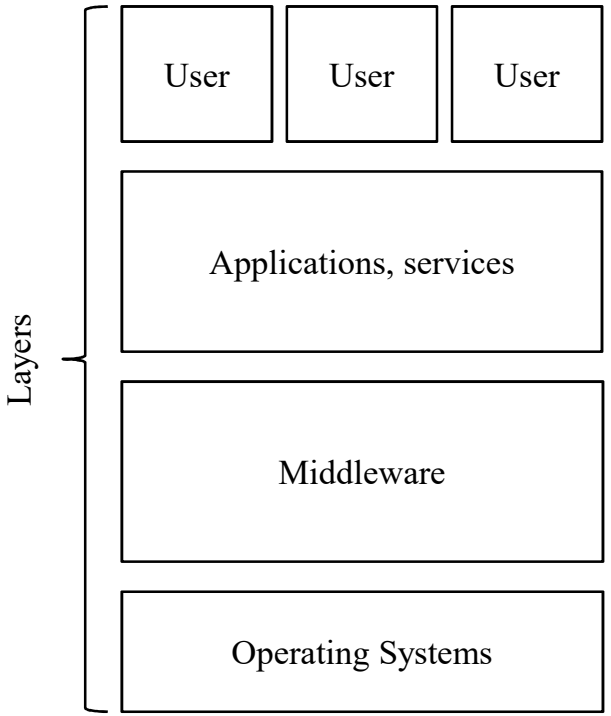
Applications, services: designed to help the user to perform a singular or multiple related specific tasks (e.g. office, programming toolkit, web browser, etc.).

Middleware: is an application that logically lives (mostly) in the application layer, but which contains many general-purpose protocols that warrant their own layers, including the session and presentation layers. Some of the middleware protocols could equally belong to the transport protocol.



The layered model and design issues (3)

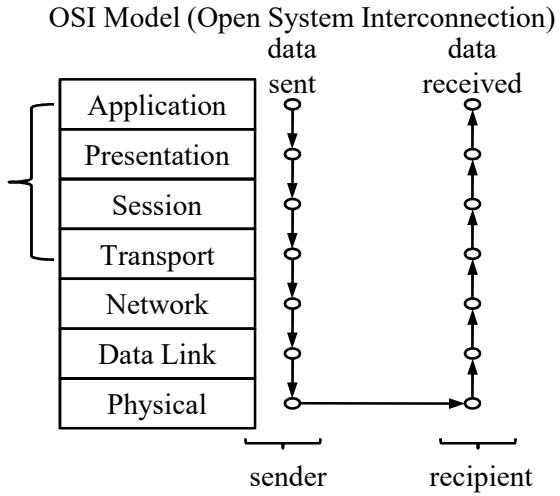
The layered model at the distributed system level



Users: could share a same computer (through session / terminal).

Applications, services: designed to help the user to perform a singular or multiple related specific tasks (e.g. office, programming toolkit, web browser, etc.).

Middleware: is an application that logically lives (mostly) in the application layer, but which contains many general-purpose protocols that warrant their own layers, including the session and presentation layers. Some of the middleware protocols could equally belong to the transport protocol.



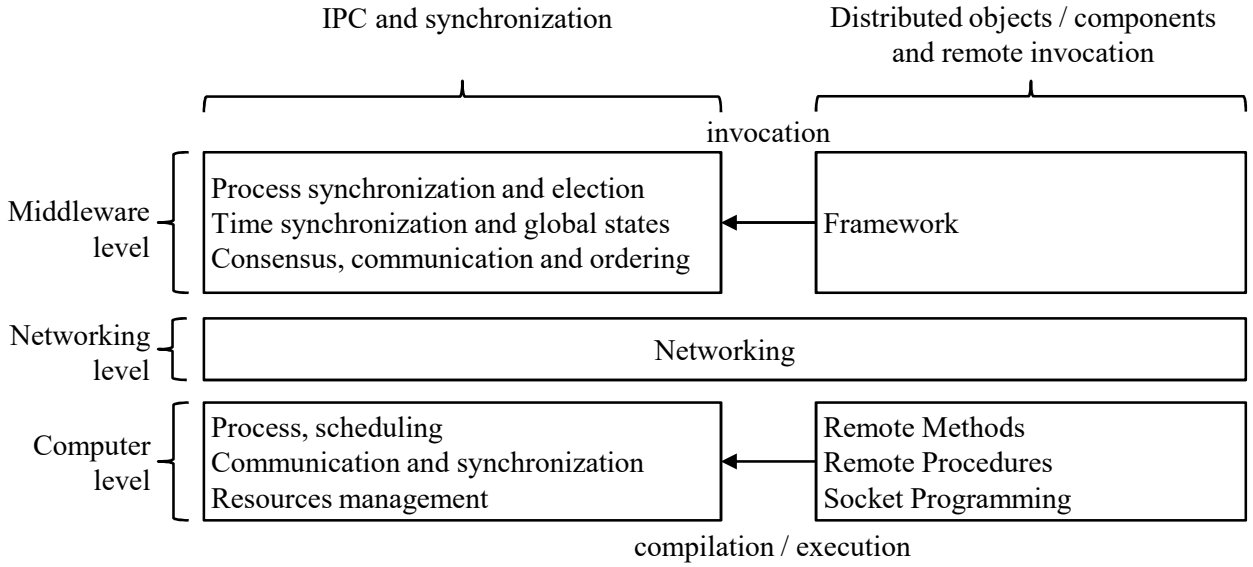
	Protocols, e.g.
Application	HTTP, FTP, SMTP, etc.
Presentation	SSL, CORBA, etc.
Session	RPC, RMI, etc.
Transport	TCP, UDP, etc.

The layered model and design issues (4)

Two main programming approaches for middleware design exist.

Inter-Process Communication (IPC) and synchronization: is related to a set of methods for the exchange of data among multiple threads and/or processes. Processes may be running on one or more computers connected by a network. The methods of IPC may vary based on the bandwidth and latency of communication, and the type of data being communicated. Synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, so as to reach an agreement or commit to a certain sequence of action.

Distributed objects / components and remote invocation: is concerned with programming models for distributed applications. That is, these applications are composed of cooperating programs running in several different processes. Such programs need to be able to invoke operations in other processes, often running in different computers.



Introduction
to distributed systems

1. Definitions and motivations
2. Use-cases and application domains
3. Trends in distributed systems
4. Relations to parallel systems
5. Goals and challenges of distributed systems
6. The layered model and design issues
7. Types of distributed systems

Types of distributed systems (1)
“Distributed information systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed information systems: are found in organizations that were confronted with a wealth of networked applications, but for which interoperability turned out to be a painful experience.
We can distinguish several levels at which integration took place.

Distributed transactions: appear with server connected to a database and made available to remote programs called clients. Such clients could send a request to the server, after which a response would be sent back. Clients can wrap a number of requests for different servers, into a single largest request executed as a distributed transaction.
A transaction guaranties that all, or none of the requests, would be executed.
Programming using transactions requires special primitives:

Primitive	**Description**
Begin_transaction	Make start of transaction
End_transation	Terminate the transaction and try to commit
Abort_transaction	Kill the transaction and restore the old value
Read	Read data from a file, a table or otherwise
Write	Write data from a file, a table or otherwise

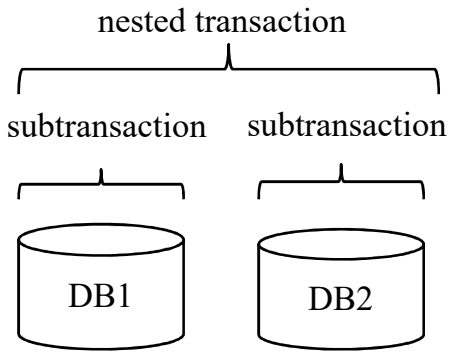
Transactions and primitives follow the **ACID** properties:
1. **A**tomic (transaction is indivisibly)
2. **C**onsistent (it does not violate system invariants)
3. **I**solated (transactions do not interfere each other)
4. **D**urable (once it commits, changes are permanent)

Types of distributed systems (2) “Distributed information systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed information systems: are found in organizations that were confronted with a wealth of networked applications, but for which interoperability turned out to be a painful experience.
We can distinguish several levels at which integration took place.

Distributed transactions: appear with server connected to a database and made available to remote programs called clients. Such clients could send a request to the server, after which a response would be sent back. Clients can wrap a number of requests for different servers, into a single largest request executed as a distributed transaction.
A transaction guaranties that all, or none of the requests, would be executed.
Programming using transactions requires special primitives:



two different
independent databases

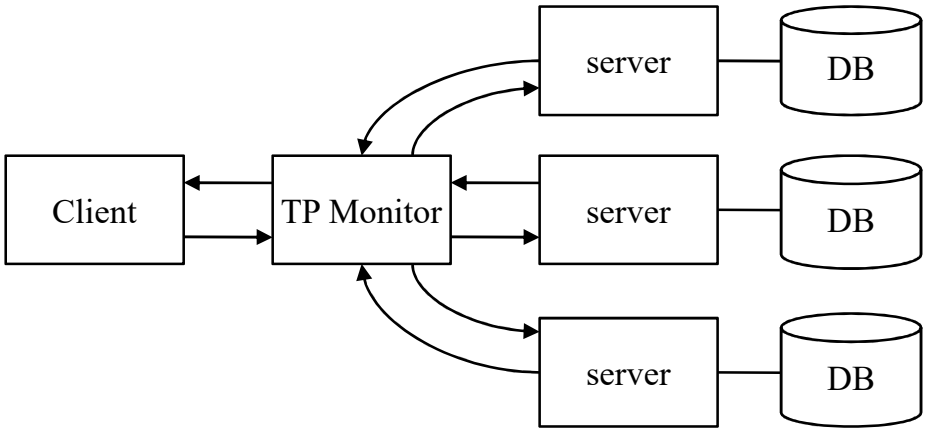
Nested transaction: is constructed from a number of subtransactions. The top-level transaction may fork off children that run in parallel, on different machines, to gain performance or simplify programming. Each of these children may also execute one or more subtransactions, or fork off its own children.

Types of distributed systems (3) “Distributed information systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed information systems: are found in organizations that were confronted with a wealth of networked applications, but for which interoperability turned out to be a painful experience.
We can distinguish several levels at which integration took place.

Distributed transactions: appear with server connected to a database and made available to remote programs called clients. Such clients could send a request to the server, after which a response would be sent back. Clients can wrap a number of requests for different servers, into a single largest request executed as a distributed transaction.
A transaction guaranties that all, or none of the requests, would be executed.
Programming using transactions requires special primitives:



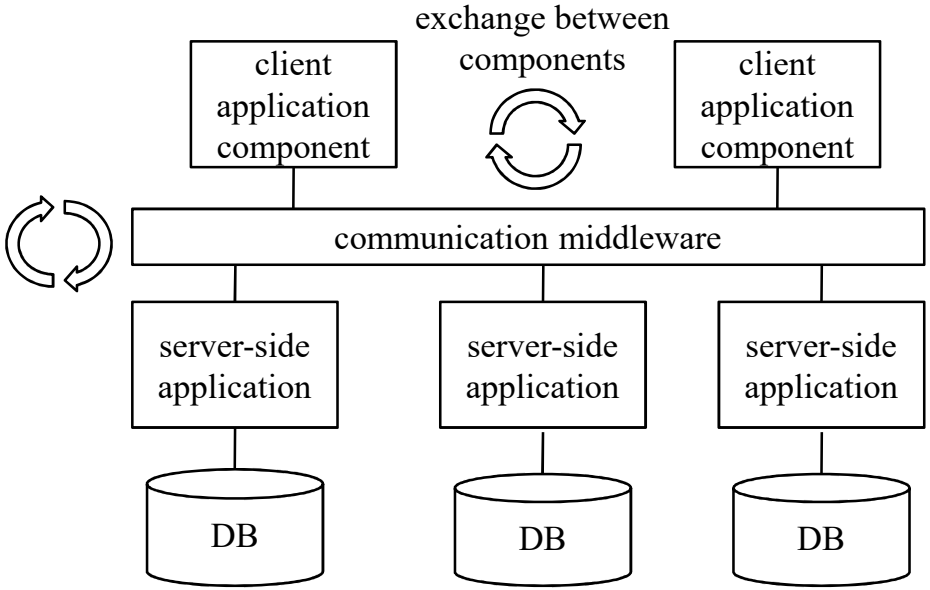
Transaction processing monitor (TP Monitor): handles distributed (or nested) transactions. Its main task was to allow an application to access multiple servers / databases by offering it a transactional programming model.

Types of distributed systems (4) “Distributed information systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed information systems: are found in organizations that were confronted with a wealth of networked applications, but for which interoperability turned out to be a painful experience.
We can distinguish several levels at which integration took place.

Enterprise Application Integration (EAI): the more the applications became decoupled from their databases, the more it became evident to integrate them. In particular, application components should be able to communicate directly with each other and not merely by means of the request/reply behavior or transactions with the database. Then, the main idea was that existing application components could directly exchange information.

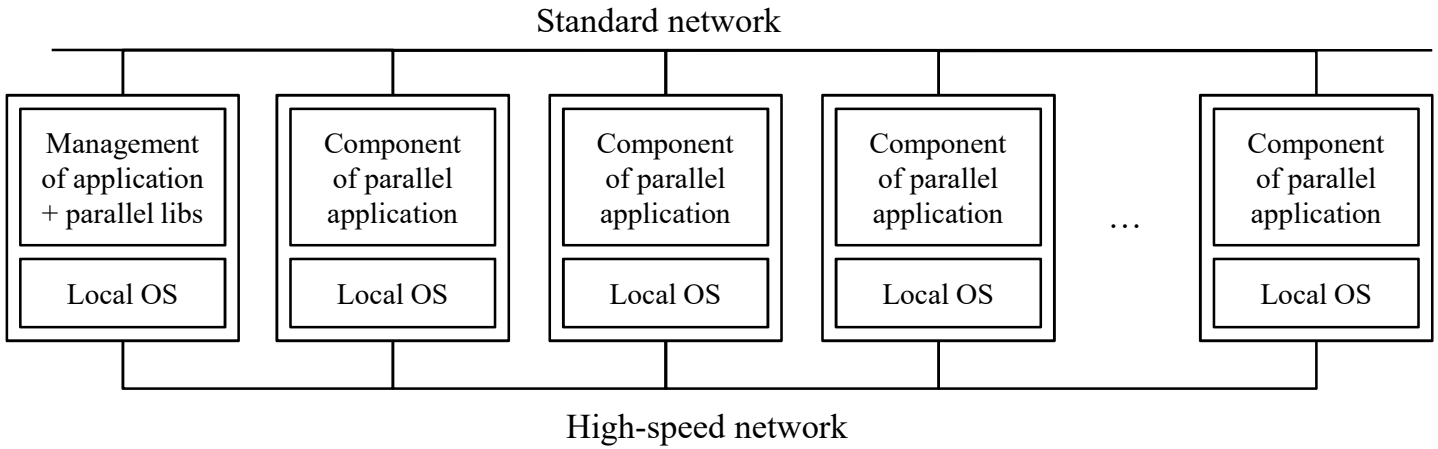


Types of distributed systems (5) “Distributed computing systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed computing systems: are used for high performance computing tasks. One can make difference between two subgroups:

Cluster computing: the underlying hardware consists of a collection of similar computers, closely connected by means of a high-speed local network. In addition, each node runs the same operating system.



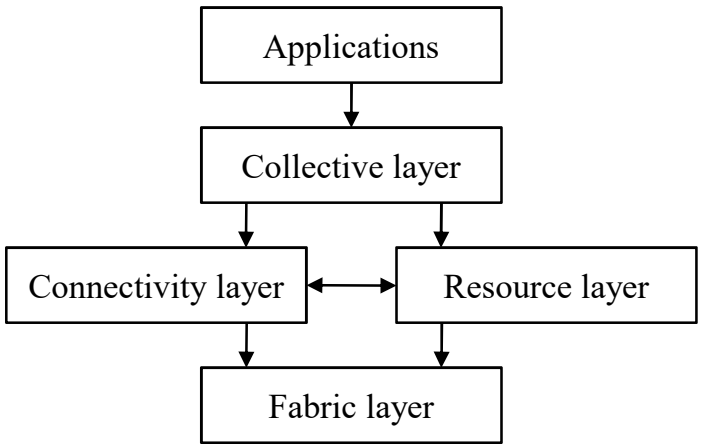
Types of distributed systems (6) "Distributed computing systems"

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed computing systems: are used for high performance computing tasks. One can make difference between two subgroups:

Grid computing: consists in a distributed system constructed as a federation of computers presenting different hardware, software and network technologies. Much of the software for realizing grid computing evolves around providing access to resources from different systems through interfaces. Focus is often on architectural issues.

e.g. the Foster's architecture



Layer	Description
Applications	Applications that operate within a virtual organization and which make use of the grid computing environment.
Collective layer	It deals with the access to multiple resources (discovery, allocation, scheduling, replication).
Connectivity layer	It groups communication protocols to support grid transactions and security access.
Resource layer	This layer is responsible of the access control, it will rely on transaction and authentication of the connectivity layer, and the management functions of the fabric layer.
Fabric layer	It contains functions for local resource management (state checking, synchronization, etc.).

Types of distributed systems (7) “Distributed pervasive systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed pervasive systems: are concerned with mobile and embedded computing devices. Within, we are confronted to systems in which instability is the default behavior. Devices are small, mobile, battery powered and have, in most of the cases, only wireless connexion.

An important feature of pervasive distributed systems is the general lack of human administrative control. At best, devices can be configured by their owners, otherwise they need to automatically discover their environment and **nestle in**. This nestle in is concerned with three requirements:

1. **Embrace contextual changes:** devices must be aware that his environment may change all the time.

2. **Encourage ad hoc composition:** devices can be used in very different ways by users, as a result it should be easy to configure the suite of applications running on a device.

3. **Recognize sharing as the default:** devices generally join the system to access and share information. This calls for means to easily read, store, manage and share information. In light of the intermittent and changing connectivity of devices, the space where accessible information resides will most change all the time.

Types of distributed systems (8) “Distributed pervasive systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed pervasive systems: are concerned with mobile and embedded computing devices. Within, we are confronted to systems in which instability is the default behavior. Devices are small, mobile, battery powered and have, in most of the cases, only wireless connexion.

Home systems consist of:
- one or more personal computers.
- typical consumer electronics such as TVs, audio and video equipments, gaming devices, smartphones, PDAs, etc.,
- misc devices such as kitchen applications, surveillance cameras, clocks, lighting controllers, etc.

Main challenges are:
- these systems should be completely **self-configuring** and **self-managing** (e.g. Universal Plug and Play “UPnP”).
- within these systems, we must manage what is known as “**personal space**” (e.g. single machine acts as a master to collect and synchronize data).

Types of distributed systems (9)
“Distributed pervasive systems”

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

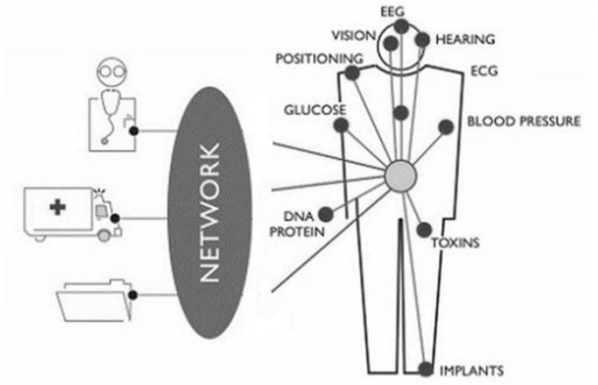
Distributed pervasive systems: are concerned with mobile and embedded computing devices. Within, we are confronted to systems in which instability is the default behavior. Devices are small, mobile, battery powered and have, in most of the cases, only wireless connexion.

Electronics Health Care systems are sets of devices being developed to monitor the well-being of individuals and to contact physicians. These systems consist of:

- various sensors (ECG, motion) organized in a **Body Area Network (BAN)**, preferably wireless.
- a central hub that manages the BAN and collects data from sensors, from time to time, the collected data are then offloaded to a larger storage device.

The main challenges are:

- BAN should be able to operate when a person is moving (i.e. wireless only).
- for reason of efficiency, BAN will be required to support **in-network data processing**, meaning that monitoring data will have to be aggregated before permanently storing it or sending it to a physician.



Types of distributed systems (10) "Distributed pervasive systems"

Distributed information systems	Transaction processing systems
	Enterprise Application Integration (EAI)
Distributed computing systems	Cluster computing
	Grid computing
Distributed pervasive systems	Home systems
	Electronic health care systems
	Sensor networks

Distributed pervasive systems: are concerned with mobile and embedded computing devices. Within, we are confronted to systems in which instability is the default behavior. Devices are small, mobile, battery powered and have, in most of the cases, only wireless connexion.

Sensor Networks: most of the sensor networks use wireless communication, and the nodes are often battery powered. They consist of:

- tens to thousands of relatively small nodes, each equipped with a sensing device.
- each sensing device should supports storage capabilities, otherwise a distributed database must be around to collect sensor data through the network, which waste network resources and energy.

The main challenge is:

- sensor networks will be required to support **in-network data processing**, meaning that a query must be forwarded to all the sensor nodes (e.g. along a tree encompassing all nodes) and to subsequently aggregate the results, as they are propagated back to the root, where the initiator is located.