Introduction to Distributed Systems

Vania Marangozova-Martin

Vania.Marangozova-Martin@imag.fr

Web Site: http://ids.forge.imag.fr

Goals

- Get an insight on the complexity of distributed systems
- Introduction to middleware
  - Why do we need it?
  - What is inside?
- Practical work
  - This course is "practice-oriented"
  - Learn to use modern technologies: web servers, noSQL databases, cloud…

Agenda

Week		Tuesday, 09:45 – 11:15 (1h30)	Tuesday, 11:15 – 12:45
5	01-févr	Introduction to distributed systems. Java RMI.	RMI Tutorial
		Rattrapage Tuesday 17h: Distributed Chat with RMI	
6	08-févr	Distributed Web applications. Servlets.	Servlet Tutorial.
		Ratrrapage Tuesday 17h: Distributed chat RMI + demos	
7	15-févr	Servlets Lab	Servlets Lab + demos
8	22-févr	Interruption week	
9	29-févr	Coordination-Based Systems	Coordination Tutorial.
10	07-mars	No IDS (Vania away)	
11	14-mars	Coordination Lab	Coordination Lab + demos
12	21-mars	P2P System	P2P Tutorial.
13	28-mars	P2P Lab	P2P Lab
14	04-avr	Cloud Computing	Amazon, Google, Azure Cloud services presentations from students
15	11-avr	Cloud Lab (Part I)	Cloud Lab (Part II)
16	18-avr	Interruption week	

Administrativia

- Evaluation
  - RMI chat demo
  - Servlets web application demo
  - Cloud demo
  - Presentation of a cloud service
  - Final exam
- Web site: ids.forge.imag.fr
- Do not hesitate to mail me questions
  - tagged in the subject with [M1_MOSIG_IDS]

What is a distributed system

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

  Leslie Lamport, 1987.

Why distribution?

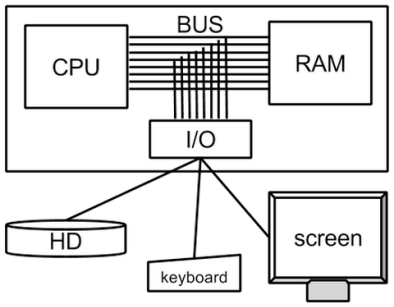
Distribution is everywhere!

- Systems need it
  - Integration of initially separated modules
  - Massive resource integration
    - Grids, clouds, data centers
  - New application domains
    - Ubiquitous computing
      - Surveilance, domotics
- Technical advances
  - Cost and performances
  - Generalized interconnection
    - Computers+TV+telecom
    - Sensor networks

Distributed Systems Characteristics

- Definition of a distributed system
  - Mutliple components interconnected with a communication system; components have computing (CPU), storage (memory, disks) or interface functions (sensors)
  - The components are not independent but collaborate towards a commin goal

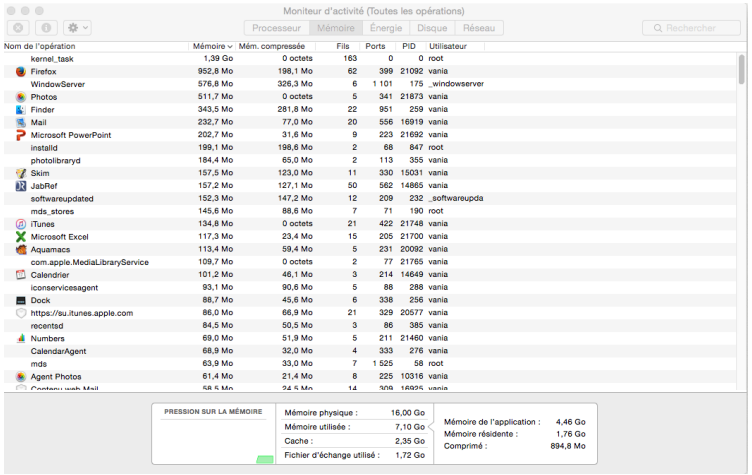
What is the difference between a centralized and a distributed system? 1/7



centralized system

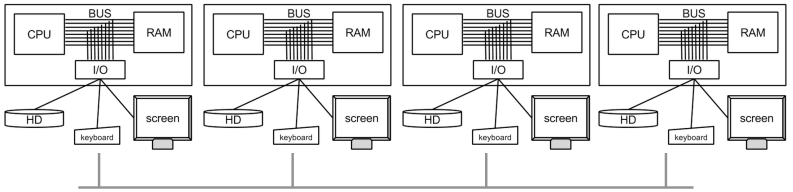
- All is accessible on the same machine
  - Memory
  - Data on disk
  - I/O
- Programs run locally
  - Can be monitored
    - Memory state
    - Process state
    - …

What is the difference between a centralized and a distributed system? 2/7



What is the difference between a centralized and a distributed system? 3/7

- What can be distributed?
  - The ressources are distributed
    - CPU
    - Memory (live, persistant) is distributed and not shared



What is the difference between a centralized and a distributed system? 4/7

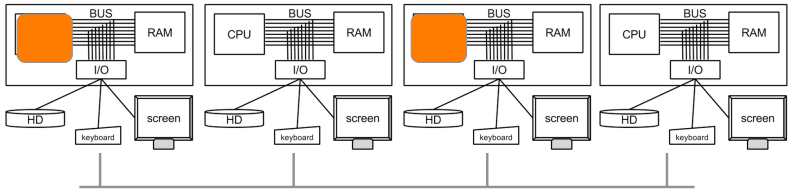
- What can be distributed?
  - The code

a variable here

is not accessible here

what we do here

is not what we do here



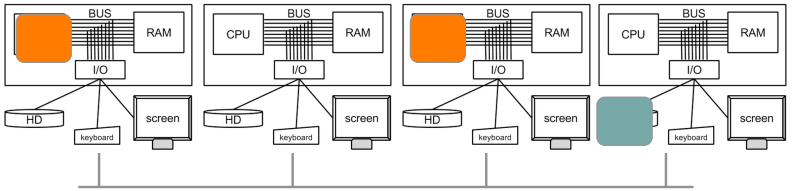
What is the difference between a centralized and a distributed system? 5/7

- What can be distributed?
  - The data

data is not here

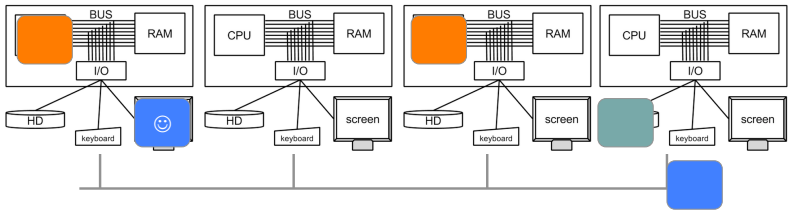
data is not here

data is not here



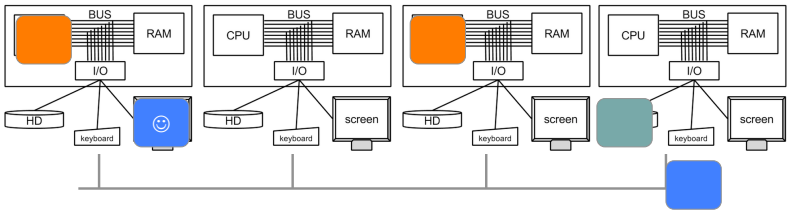
What is the difference between a centralized and a distributed system? 6/7

- What can be distributed?
  - Interactions



What is the difference between a centralized and a distributed system? 7/7

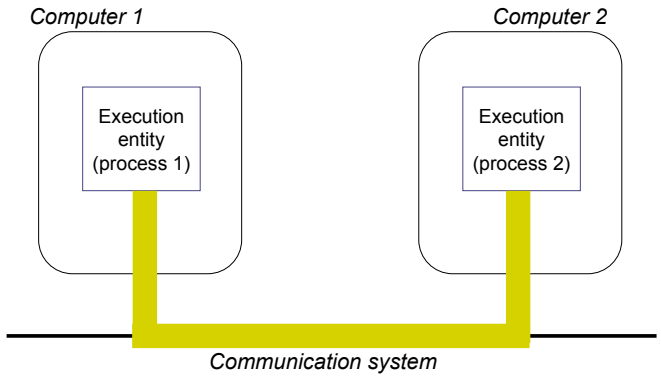
- No global time
- No global state
- … and it is supposed to work anyway ☺



Distributed Systems Characteristics (2)

- Desired properties
  - The system should be able to operate (at least in degraded mode) in the case of component failure
  - The system should support communication failures (message loss, disconnection, …)
  - The system should resist to secyrity attacks (confidentiality violations, integrity violations, denial of service, resource stealing, …)

A Distributed System

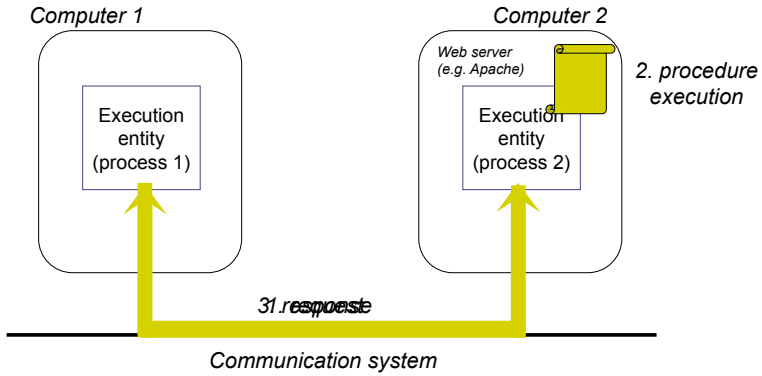


Communication mechanisms in a distributed system

- Direct (i.e. Synchronous) communication
  - Program to program
    - E.g. remote procedure call
  - Program to database
    - E.g. distributed transaction processing
- Indirect (i.e. Asynchronous) communication
  - Message passing

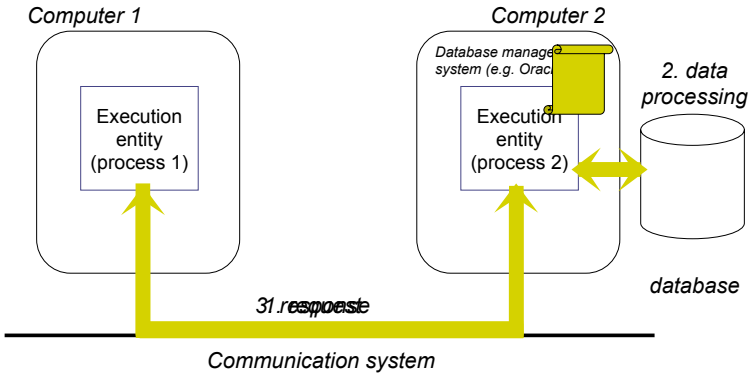
Communication mechanisms in a distributed system

- Remote procedure call (e.g. a web application)



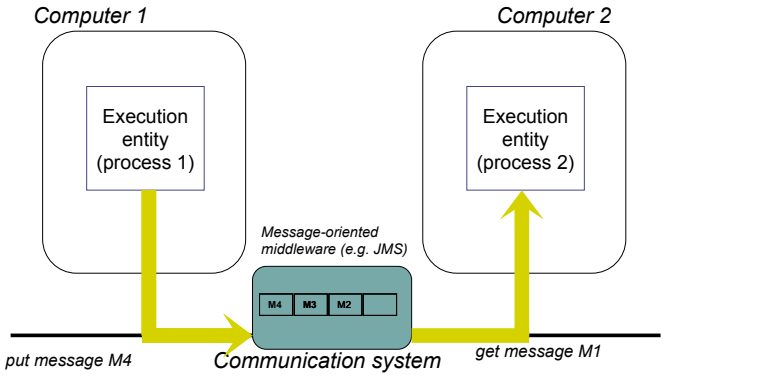
Communication mechanisms in a distributed system

- Distributed transaction processing (e.g. a database server)



Communication mechanisms in a distributed system

- Message passing (e.g. a chat system)



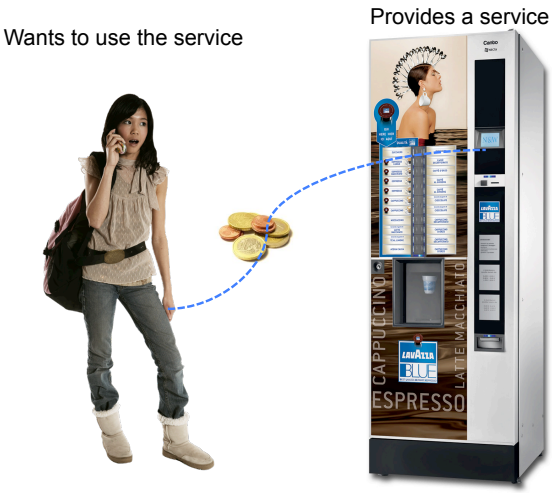
Outline

1. What is a distributed system
   - Communication mechanisms in distributed systems
   - **Services and interfaces in computing systems**
   - Client/server architecture
2. What is a middleware
3. References

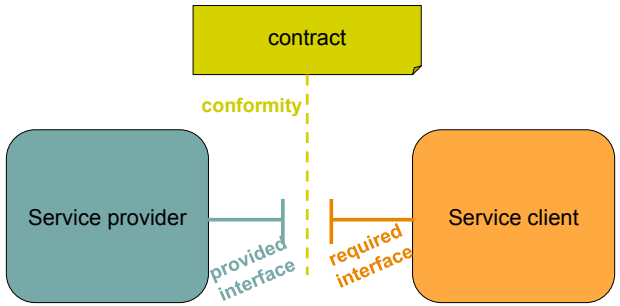
Services and interfaces in a computing system

- Service
  - A set of functions: implementing a given behavior, reusable, that are used in a predefined manner
    - Each SW or HW component provides a service
    - A service may be realized in different ways
    - "A service is a contractually defined behavior that can be implemented and provided by any component for use by another component, based solely on the contract",
      Bieber el. al., Service oriented programming, http://www.openwings.org/
- Interface
  - A service is accessible via one or several interfaces
  - An interface defines the possible interaction between a service provider and its user

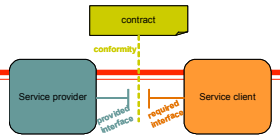
A Real-world Example



Interfaces (1/2)

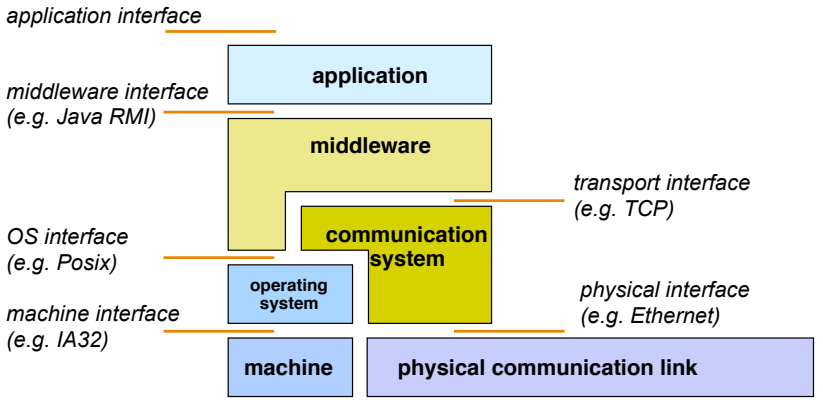


Interfaces (2/2)



- A service relies on two interfaces
  - Required interface (from the service client point of view)
  - Provided interface (from service provider point of view)
- Contract
  - The contract specifies the conformity between the provided and required interfaces
  - The service client and the service provider are considered as black-boxes; they might be replaced by other implementations as long as the contract is respected
- The contract may specify aspects that are not related to the interfaces
  - Non-functional properties related to QoS requirements

Examples of important interfaces in computing systems

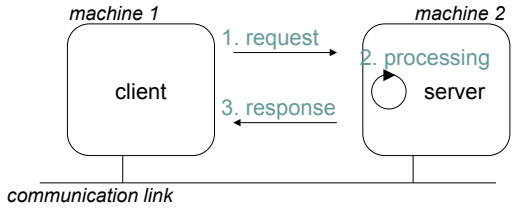


Outline

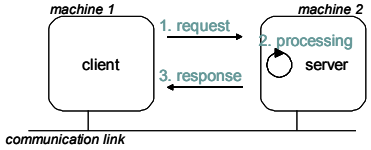
1. What is a distributed system
   - Communication mechanisms in distributed systems
   - Services and interfaces in computing systems
   - **Client/server architecture**
2. What is a middleware
3. References

Client/server architecture (1)

- Definitions
  - The client/server architecture is a general interaction model
  - The server provides a service
  - The client requests that service
  - The client and the server are usually (but not necessarily) hosted by two distinct machines
  - Examples of protocols based on the client/server architecture: RPC, Java RMI, Web Services, etc.



Client/server architecture (2)



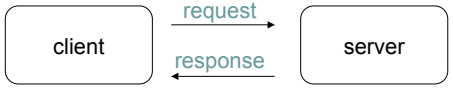
- Request message:
  - Sent by the client to the server
  - Specifies the requested service (a server may provide several services)
  - Contains parameters of the requested service
- Response message:
  - Sent by the server to the client
  - Results of service execution, or error message
- Synchronous communication between the server and the client:
  - When the client sends a request, it waits (it is blocked) until the server replies to its request

Client/server architecture (3)

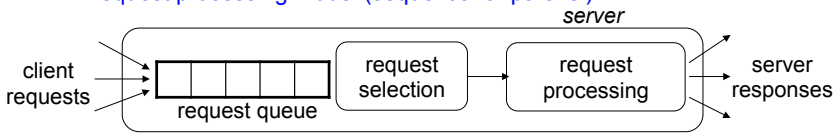
- Advantages of the client/server architecture
  - Structuring
    - Separation between the interface of a service and the implementation of that service
    - Based on this separation, the client and server implementations can be modified as long as the interface is kept unchanged
  - Protection/security
    - The client and server run in different protection domains
  - Resource management
    - A server may be shared by several clients

Client/server architecture (4)

- A server shared by several clients
  - The client point of view



  - The server point of view
    - Selecting a request among client requests
    - Request processing model (sequential or parallel)



Client/server architecture (5)

- Request selection (i.e. scheduling) model
  - First, the server selects one of the waiting (i.e. queued) client requests
  - Then, it process the client request and builds its response
  - Before it returns it to the client
- Different request selection strategies
  - First-In First-Out (FIFO)
  - Shortest first
  - Priority-based scheduling

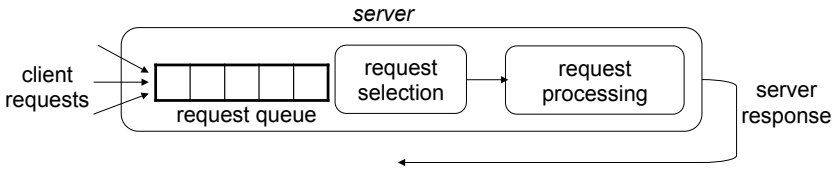
Client/server architecture (6)

- Request processing model (resource management)
  - The client and server are executed by two distinct processes (asynchronous call)
  - The client waits untils it receives a response to its request
  - Several requests may be processed concurrently by the server
    - real parallelism (e.g. multiprocessors, I/O)
    - pseudo-parallelism
  - Concurrency may take the form of:
    - multiple processes, or
    - multiple threads

Client/server architecture (7)

- Server resource management – A unique process

```
while (true) {
        receive(client_id,message);
        extract(message, service_id, params);
        results = do_service(service_id, params);
        send(client_id, results);
}
```

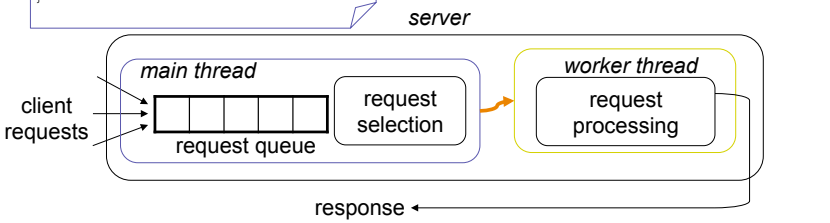


Client/server architecture (8)

- Server resource management – Multiple processes

```
while (true) {
    receive(client_id,message);
    extract(message, service_id,
       params);
    thr = create_thread(client_id,
       service_id,params);
}
```

```
Program executed by thread thr:
    results = do_service(
       service_id, params);
    send(client_id, results);
    exit
```



Client/server architecture (9)

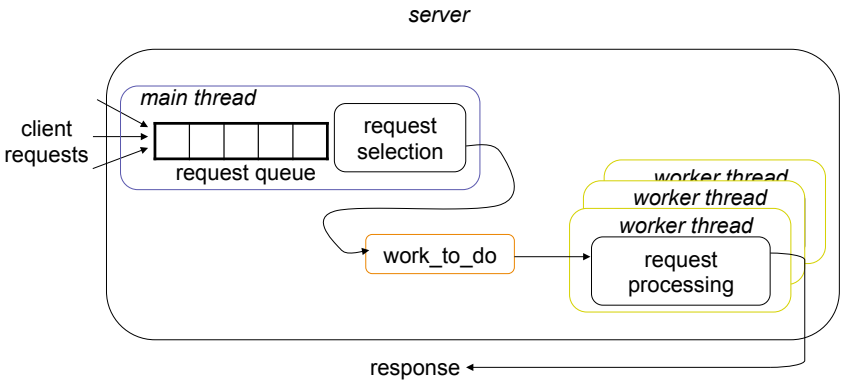
- Server resource management – A pool of processes

```
while (true) {
    receive(client_id,message);
    extract(message, service_id,
       params);
    work_to_do.put(client_id,
       service_id,params);
}
```

```
Pool of processes:
while (true) {
    work_to_do.get(
       client_id, service_id,
       params);
    results = do_service(
       service_id, params);
    send(client_id, results);
}
```

Client/server architecture (10)

- Server resource management – A pool of processes



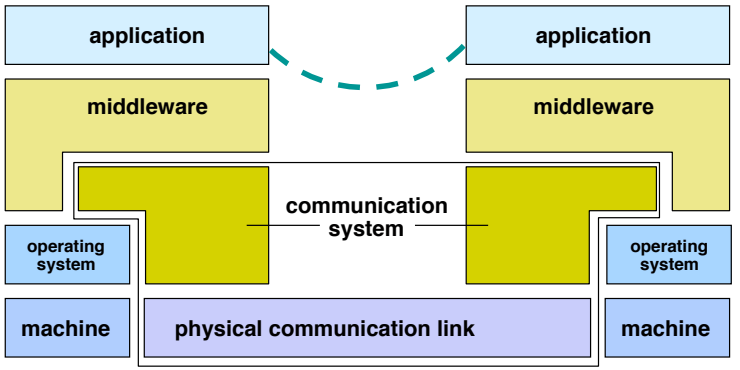
Client/server architecture (11)

- Application of the client/server architecture
  - With low level operations
    - Using functions of the communication system
    - Example: Sockets
      - TCP, connected mode
      - UDP, unconnected mode
  - With high level operations
    - Using a middleware
    - Example: RMI in object-oriented middleware
      - Remote method invocation

Outline

1. What is a distributed system
   - Communication mechanisms in distributed systems
   - Services and interfaces in computing systems
   - Client/server architecture
2. **What is a middleware**
3. References

What is a middleware

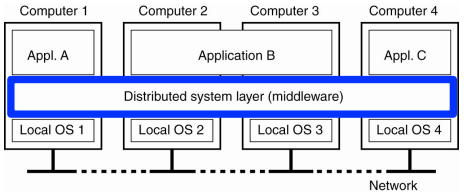


Functions of a middleware

- A middleware has mainly four functions
  - Make distribution as invisible (transparent) as possible
  - Provide a homogeneous view of underlying heterogeneous hardware and software systems
  - Provide services of common use for distributed systems
  - Provide a high-level interface or API (*Applications Programming Interface*) for programming distributed applications
- Middleware aims at simplifying programming distributed systems
  - Implementation, evolution and reuse of applications code
  - Inter-platform portability of applications
  - Interoperability between heterogeneous applications

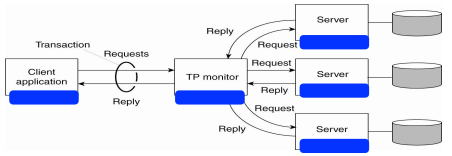
Examples of middleware solutions

- Computation intensive systems



MPI
OpenPBS
Globus
gLite
OpenStack
STORM
…

- Information systems



Oracle
MongoDB
pH1
Google Datastore
…

References

- Chris Britton, Peter Bye. *IT Architectures and Middleware: Strategies for Building Large, Integrated Systems (2nd Edition)*. Addison-Wesley, 2004.
- George Coulouris, Jean Dollimore, Tim Kindberg. *Distributed Systems: Concepts and Design (4th Edition)*. Addison Wesley, 2005.
- Arno Puder, Kay Römer, Frank Pilhofer. *Distributed Systems Architecture: A Middleware Approach*. Morgan Kaufmann, 2005.
- Andrew S. Tanenbaum, Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice Hall, 2006.
- This lecture is partly based on lectures given by Sacha Krakowiak, http://sardes.inrialpes.fr/people/krakowia/