Introduction to Distributed Systems and Middleware

Sara Bouchenak

Sara.Bouchenak@imag.fr
http://sardes.inrialpes.fr/~bouchena/teaching/

Objectives

- Introduction to distributed systems and middleware
- Conceptual and practical aspects of distributed systems and middleware
- Illustration through current distributed systems, e.g. web systems, database systems

Agenda

Week	Wednesday, 13:30 – 18:30
S6	Introduction to distributed systems and middleware (CM), S. Bouchenak, 13:30 – 15:00 Introduction to JDBC (CM), C. Labbé, 15:15 – 16:45
S7	RMI-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S8	Servlet-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S9	Interruption week
S10	Introduction to transactions (CM), C. Labbé, 13:30 – 15:00
S11	Multi-tier distributed systems (CM), S. Bouchenak , 13:30 – 15:00 Servlet-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S12	Presentation of the project (CM), S. Bouchenak , 13:30 – 15:00 Multi-tier distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S13	Support projet (TD), C. Labbé & D. Serrano, 15:15 – 18:30
S14	–
S15	Project, S. Bouchenak & C. Labbé & D. Serrano, 13:30 – 16:45

Additional information

- Evaluation
  - Mid-term evaluation
    - Demonstration and evaluation of practical work
  - Final exam
- Web Page
  - http://sardes.inrialpes.fr/~bouchena/teaching/IBD/

Contact

- Introduction to distributed systems and middleware
  - Sara Bouchenak (Sara.Bouchenak@imag.fr)
    Associate Professor, University of Grenoble I
    Researcher, LIG Laboratory, ERODS research group
- Transactional processing
  - Cyril Labbé (Cyril.Labbe@imag.fr)
    Associate Professor, University of Grenoble I
    Researcher, LIG Laboratory, SIGMA research group

Outline

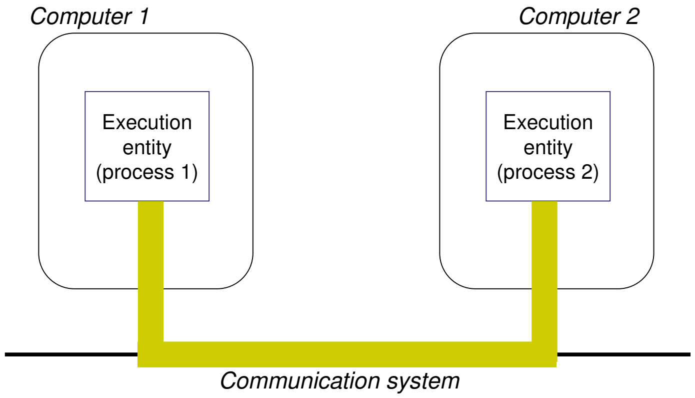
1. What is a distributed system
   - Communication mechanisms in distributed systems
   - Services and interfaces in computing systems
   - Client/server architecture
2. What is a middleware
3. References

What is a 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.

Distributed system

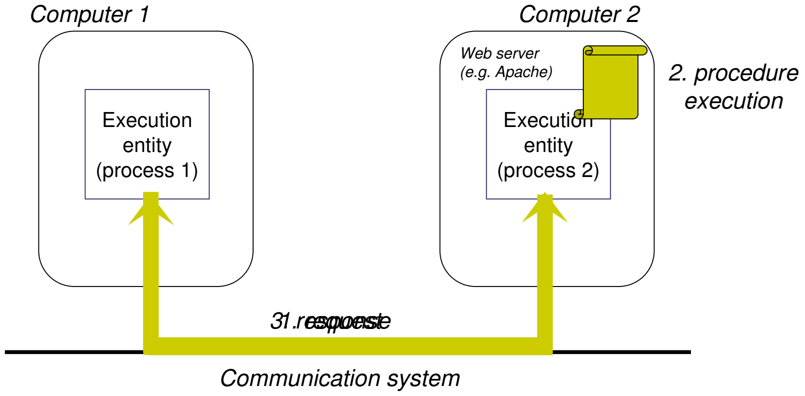


Communication mechanisms in a distributed system

- 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

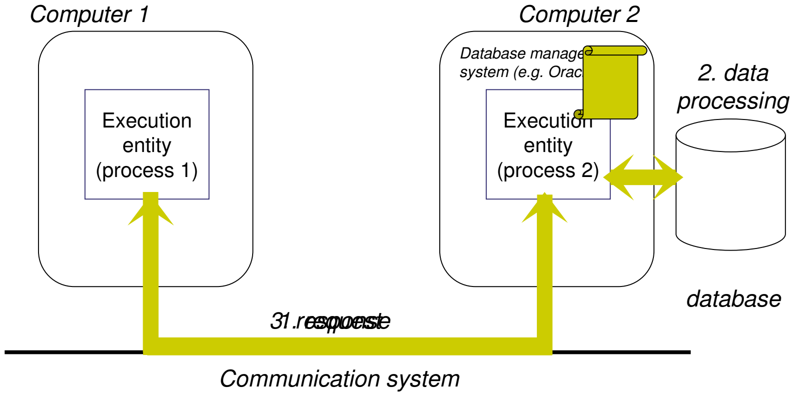
Communication mechanisms in a distributed system

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



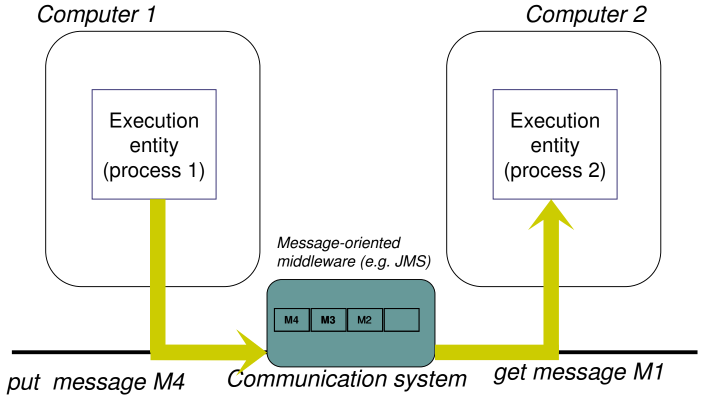
Communication mechanisms in a distributed system

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


Computer 1
Computer 2
Execution entity (process 1)
Database manage system (e.g. Orac
Execution entity (process 2)
2. data processing
database
1. request
3. response
Communication system

Communication mechanisms in a distributed system

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


Computer 1
Computer 2
Execution entity (process 1)
Execution entity (process 2)
Message-oriented middleware (e.g. JMS)
M4
M3
M2
put message M4
Communication system
get message M1

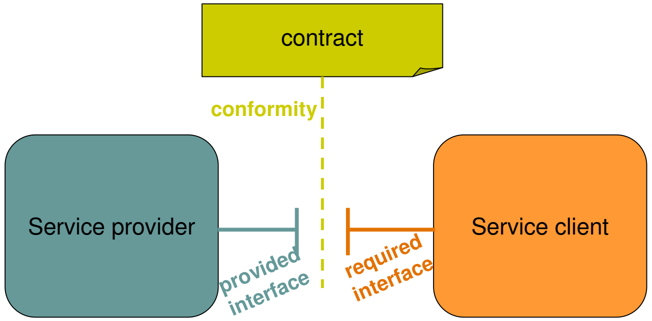
Outline

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

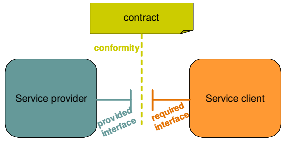
Services and interfaces in a computing system

- Service definition
  - A computing system is a set of (hardware and software) components
  - A component provides a service
  - "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 definition
  - A service is accessible via one or several interfaces
  - An interface defines the possible interaction between a service provider and its client

Interfaces (1/2)

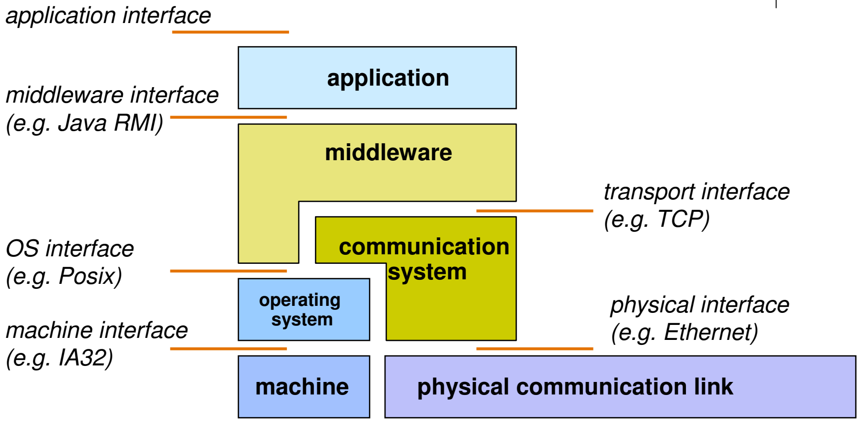


Interfaces (2/2)



- 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



application interface

application

middleware interface
(e.g. Java RMI)

middleware

transport interface
(e.g. TCP)

OS interface
(e.g. Posix)

communication system

operating system

physical interface
(e.g. Ethernet)

machine interface
(e.g. IA32)

machine

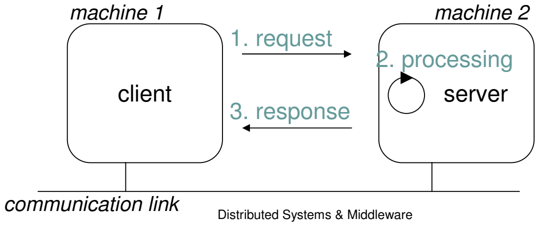
physical communication link

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.



machine 1

machine 2

client

1. request

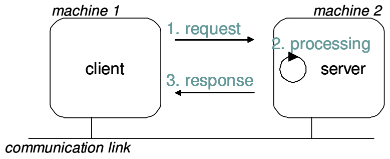
2. processing

server

3. response

communication link

Client/server architecture (2)



machine 1

machine 2

client

1. request

2. processing

server

3. response

communication link

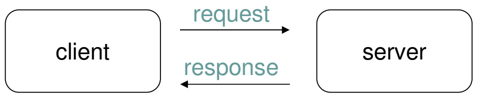
- Request message:
  - Sent by the client to the server
  - Specifies the requested service (a server may provide several services)
  - Contains parameters of the requested service
- Response message:
  - Sent by the server to the client
  - Results of service execution, or error message
- Synchronus 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
    - Seperation 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



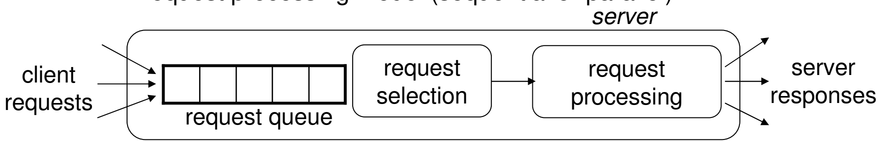
client

request

response

server

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



server

client requests

request queue

request selection

request processing

server responses

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

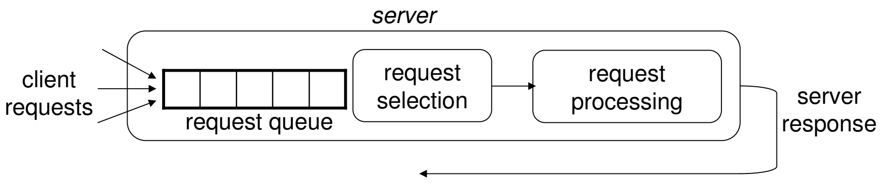
Client/server architecture (6)

- 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);
}
```

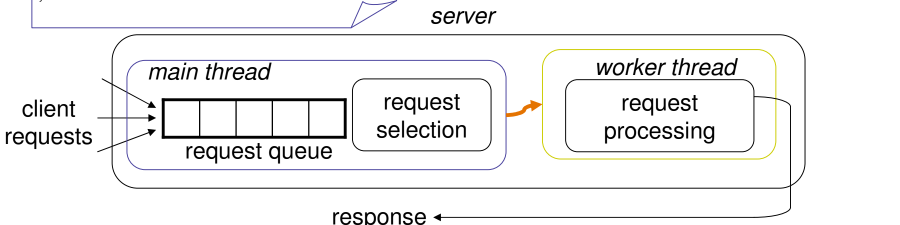


Client/server architecture (8)

- 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)

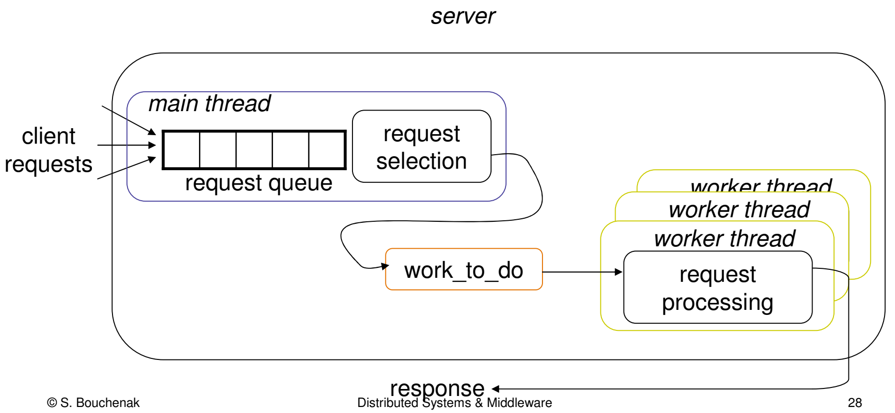
- Server resource management – A pool of processes

```
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



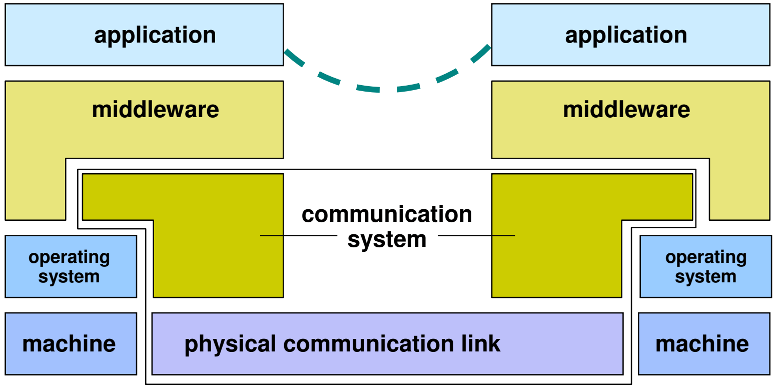
Client/server architecture (11)

- 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 for distributed systems

- 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

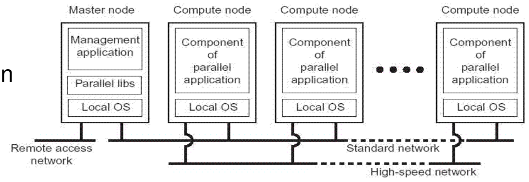
- Sun JVM
- CORBA
- Microsoft .NET
- Sun J2EE / EJB
- …

Types of distributed systems

- Distributed computing systems
- Distributed information systems
- Distributed pervasive systems

Distributed computing systems

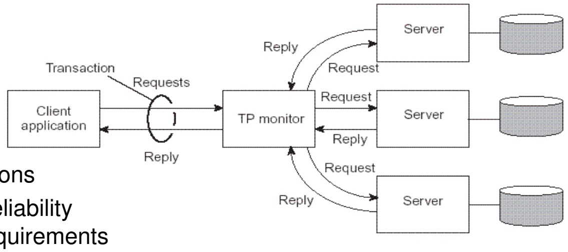
- Objective
  - Distributed systems configured for high performance computing
- Cluster computing
  - A group of high-end systems connected through a LAN
  - Homogeneous, i.e. same OS, hardware
  - Single managing node
- Grid computing
  - Heterogenneity
  - Geographical dispersion
- Applications
  - Video streaming
  - Web services
  - Scientific computing



M. van Steen, Lecture on Distributed Systems, Chapter 1, http://www.cs.vu.nl/~steen/

Distributed information systems

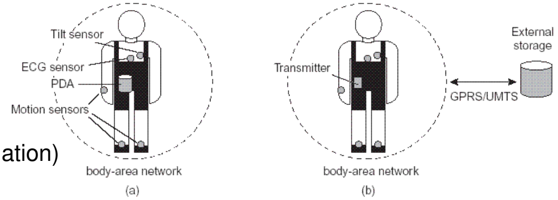
- Objective
  - Providing consistent access to (shared) data that can be distributed and accessed concurrenly
- Observation
  - Transactions
  - ACID properties
- Applications
  - Streaming applications
  - Data access with reliability and consistency requirements



M. van Steen, Lecture on Distributed Systems, Chapter 1, http://www.cs.vu.nl/~steen/

Distributed pervasive systems

- Objective
  - Providing consistent access to (shared) data that can be distributed and accessed concurrenly
- Observation
  - Contextual change
  - Ad-hoc composition
- Applications
  - Domotics (home automation)



M. van Steen, Lecture on Distributed Systems, Chapter 1, http://www.cs.vu.nl/~steen/

Outline

1. What is a distributed system
2. What is a middleware
   - What is a middleware
   - Functions of a middleware
   - Middleware for distributed systems
   - Examples of middleware solutions
   - Types of distributed systems
3. **References**

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/

Agenda

Week	Wednesday, 13:30 – 18:30
S6	*Introduction to distributed systems and middleware* (CM), S. Bouchenak, 13:30 – 15:00 **Introduction to JDBC** (CM), C. Labbé, 15:15 – 16:45
S7	RMI-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S8	Servlet-based distributed systems (CM), S. Bouchenak , 13:30 – 15:00 RMI-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S9	Interruption week
S10	Introduction to transactions (CM), C. Labbé, 13:30 – 15:00
S11	Multi-tier distributed systems (CM), S. Bouchenak , 13:30 – 15:00 Servlet-based distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S12	Presentation of the project (CM), S. Bouchenak , 13:30 – 15:00 Multi-tier distributed systems (TD), S. Bouchenak & D. Serrano, 15:15 – 18:30
S13	Support projet (TD), C. Labbé & D. Serrano, 15:15 – 18:30
S14	–
S15	Project, S. Bouchenak & C. Labbé & D. Serrano, 13:30 – 16:45