IBD – Intergiciels et Bases de Données

Introduction

Fabien Gaud, fabien.gaud@inria.fr

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

Professors

- Middleware and distributed systems
  - Fabien Gaud (Fabien.Gaud@inria.fr)
  - Vivien Quema (Vivien.Quema@inria.fr)
- Database systems
  - Goran Frehse (Goran.Frehse@imag.fr)
  - Renaud Lachaize (Renaud.Lachaize@inria.fr)

Organization

Week date	Wednesday 13:30 – 15:00	Wednesday 15:15 – 16:45
03/02	Lecture 1 on Middleware	Lecture 2 on Middleware
10/02	Lecture 1 on DB	
24/02	Lecture 3 on Middleware	Practical work 1 on Middleware
03/03	Lecture 4 on Middleware	Practical work 1 on Middleware
10/03	Lecture 2 on DB	Lecture 5 on Middleware
17/03	Practical work 1 on DB	Practical work 2 on DB
22/03	Projet	Projet
31/03	Projet	Projet
07/04		
21/04	Projet	Projet
28/04	Soutenance Projet	Soutenance Projet

Organization

- Prerequisites
  - Java programming
  - Networking principles
  - Structured query language (SQL)
- Perspectives
  - Master 2 Professional
    - M2P – GI
  - Master 2 Research
    - M2R – Distributed Systems
    - M2R – Database systems
    - M2R – Information systems

Web site and evaluation

- Web site
  - http://www-ufrima.imag.fr/ ⇒ Intranet ⇒ Services pédagogiques ⇒ Placard électronique ⇒ M1 Info ⇒ IBD
- Evaluation
  - Mid-term evaluation
    - A project on middleware, distributed systems and database systems
    - Demonstration of project results
    - Evaluation of the demonstration
  - Final evaluation
    - Final exam

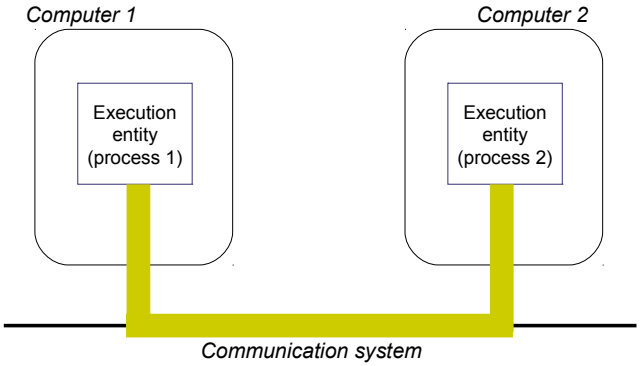
Outline of lectures and practical work on middleware

- Lectures
  - Introduction to distributed systems and middleware
  - RMI-based distributed systems
  - Servlet-based distributed systems
  - Introduction to multi-tier distributed Internet services
- Practical work
  - Programming distributed systems with RMI
- Project on multi-tier Internet services

Outline

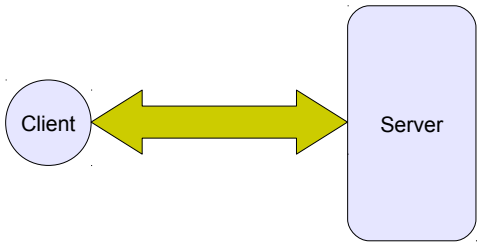
- Objectives and organization
- **What is a distributed system ?**
- What is a middleware ?
- Conclusion

What is a distributed system



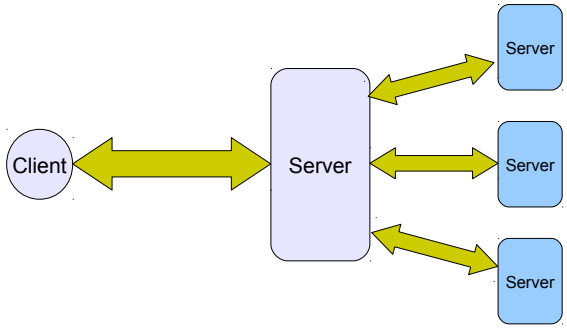
Example of distributed systems

- Client - Server



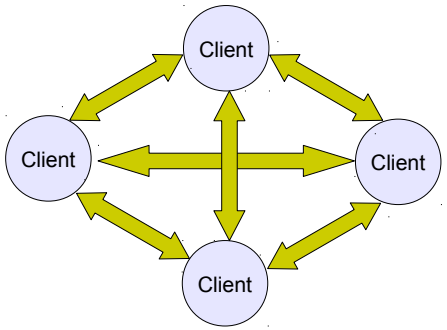
Example of distributed systems

- Multi-Tier



Example of distributed systems

- Peer to peer



Distributed systems

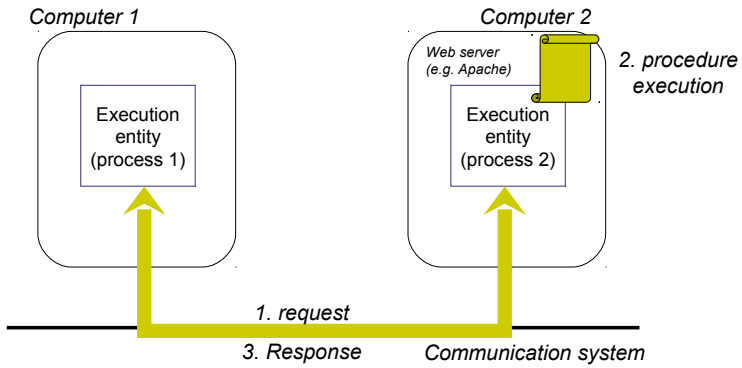
- Advantages
  - Storage/Computation capacity
  - Scalability
  - Fault tolerance
- Points to consider
  - Security
  - Failure
  - Synchronization
  - Heterogeneity

Communication mechanisms in a distributed system

- Direct
  - Program to program
    - E.g. remote procedure call
  - Program to database
    - E.g. distributed transaction processing
- Indirect
  - Message queuing

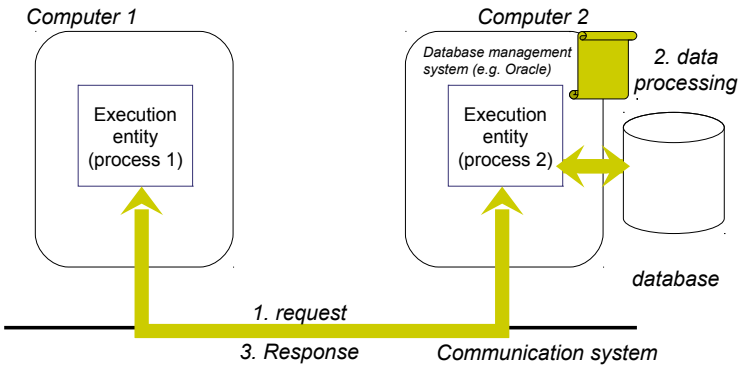
Communication mechanisms in a distributed system

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



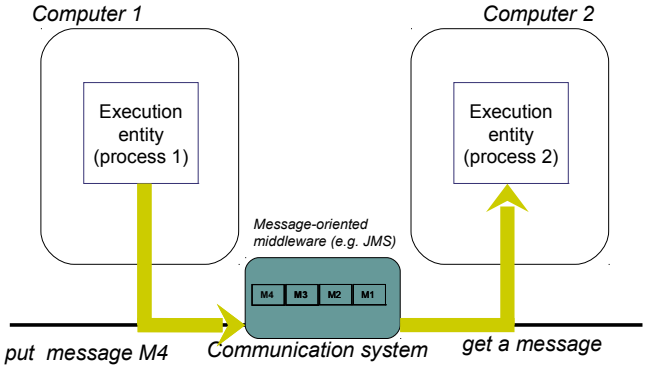
Communication mechanisms in a distributed system

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



Communication mechanisms in a distributed system

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



Communication mechanisms in distributed systems - perspectives

- Direct
  - Program to program
    - E.g. remote procedure call
  - Program to database
    - E.g. distributed transaction processing
- Indirect
  - Message queuing

M1 Info – IBD – "I" Part

M1 Info – IBD – "BD" Part

M2P GI – Dist. Sys.
M2R Info – sp. SAR – Dist. Sys.

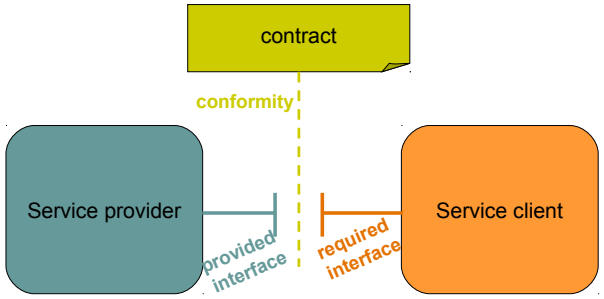
Outline

- Objectives and organization
- What is a distributed system ?
  - Communication mechanisms in distributed systems
  - **Services and interfaces in computing systems**
  - Client/server architecture
- What is a middleware ?
- Conclusion

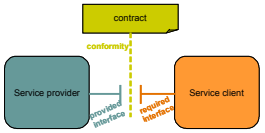
Services and interfaces in a computing system

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

Interfaces (1/2)

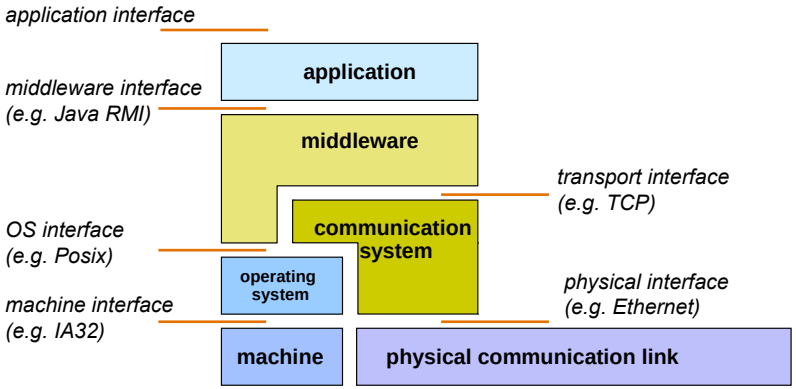


Interfaces (2/2)



- 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
- 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

- Objectives and organization
- What is a distributed system
  - Communication mechanisms in distributed systems
  - Services and interfaces in computing systems
  - **Client/server architecture**
- What is a middleware
- Conclusion

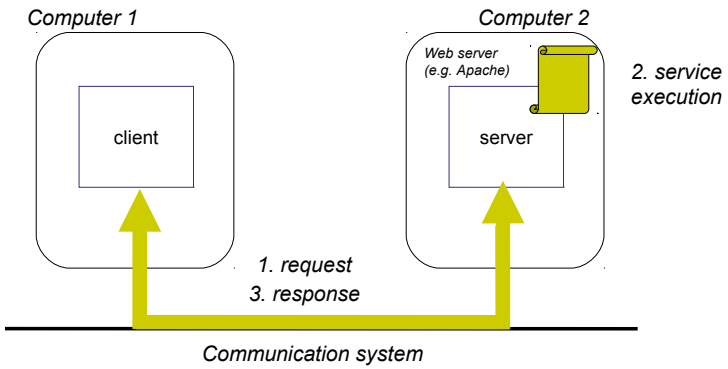
Client/Server architecture (1)

- A server provides some services
  - Examples
    - Processing database queries
    - Offering file system accesses
- A client uses some services provided by a server
  - Examples
    - Displaying database query results to the user
    - Requesting a file

Client/Server architecture (2)

- 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 (3)



Client/Server architecture (4)

- Communication between the server and the client:
  - Can be synchronous or asynchronous
  - Use a protocol defining
    - The 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
    - The response message:
      - Sent by the server to the client
      - Results of service execution, or error message
    - The interaction

Client/Server communication channel

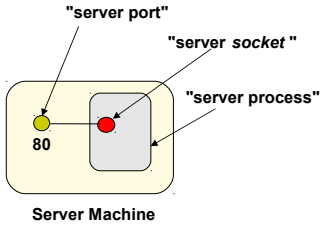
- With low level operations
  - Using functions of the communication system
  - Example: Sockets
- With high level operations
  - Using a middleware
  - Example: RMI in object-oriented middleware
    - Remote method invocation

Communication using Sockets

- Ports
  - Between two ports, allocated to two processes
  - Port numbers are managed by the operating system
    - Many important services have a standardized port
    - Example: port 80 for HTTP service
    - Port between 1 and 1023 are reserved
- Sockets
  - A programming model based on streams
    - Through a stream interface, one may send/receive bytes through a socket
  - A behavior semantics (UDP,TCP, ...)

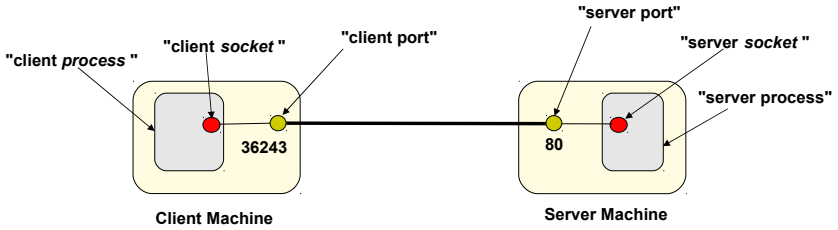
Communication using Sockets (2)

- Server side
  - Creation of the server process
    - Request a socket on a port
    - The local port is granted by the operating system
  - Server waits for incoming data

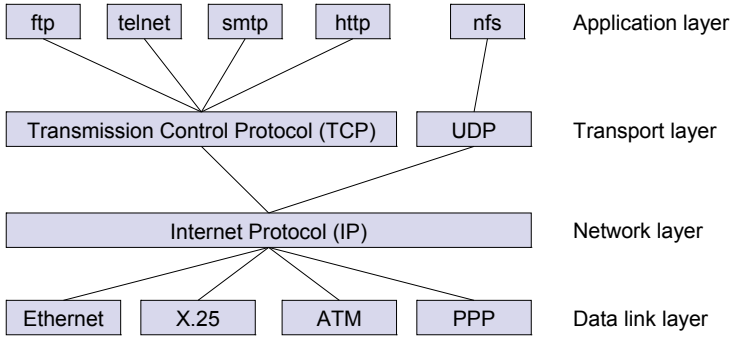


Communication using Sockets (3)

- Client side
  - Creation of the client process
  - Request a channel to the remote port
    - A local port is allocated
    - The communication channel is established
  - The two sockets are connected to each other
    - Client sends/receives data



Protocol layers



Transport level Protocol

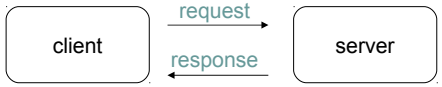
- UDP (User Datagram Protocol)
  - Over IP that routes data packets
  - Output stream is automatically sliced into data packets
  - Data packets may be lost or reordered
  - But very efficient (just an IP++)
- TCP (Transmission Control Protocol)
  - Also above IP, but it is lossless and FIFO
    - Lossless: data packets are not lost
    - FIFO: data packets are delivered in the order they were sent
  - A connection-oriented protocol
    - An actual point-to-point connection needs to be opened and closed
    - Connections introduce an overhead

UDP and TCP application example

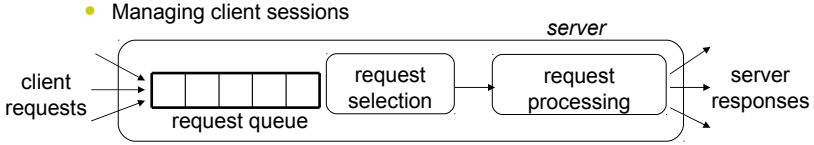
- Typical applications
  - UDP
    - Require high bandwidth, can accept loss or reordering
    - Examples:
      - Transmission of video/sound in real time
        - Out of sequence or incomplete frames are just dropped
      - Other more complex communication protocols
        - Such as totally-ordered multicast using Lamport's logical clocks
  - TCP
    - Transferring files (ftp for instance)
    - Downloading web pages or images

Server resource management

- 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)
    - Managing client sessions



Server resource management (2)

- 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, or
  - Finite State Machine (FSM)

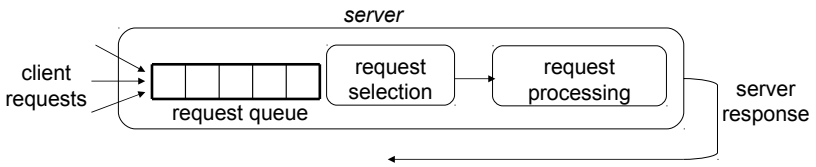
Service resource management (3)

- Threads
  - Advantages
    - Lighter than processes
    - Memory sharing
  - Drawbacks
    - Be careful with memory mutual accesses
- Processes
  - Advantages
    - Memory isolation
  - Drawbacks
    - Heavy
    - No shared memory

Single threaded server

- Server resource management – A unique thread

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

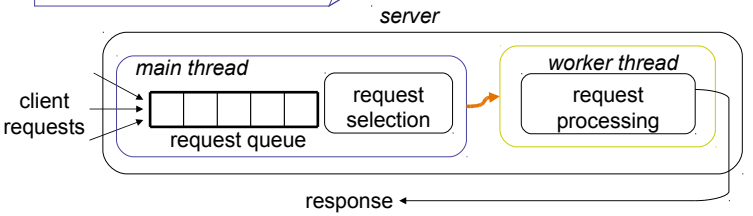


Multi-threaded server

- Server resource management – Multiple threads

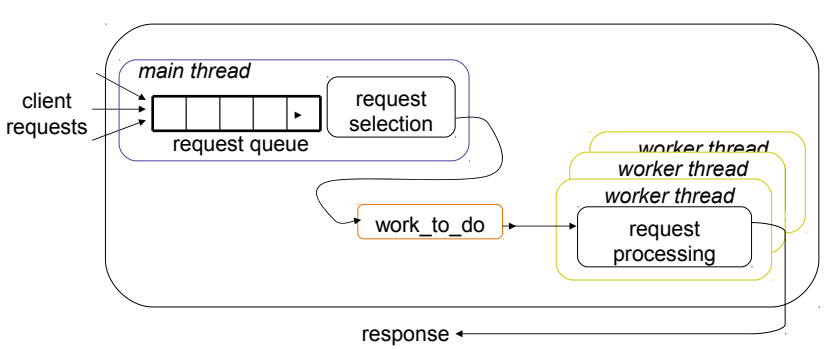
```
while (true) {
   receive(client_id,message);
   extract(message, service_id,
      params);
   thr = create_thread(client_id,
      service_id,params);
}
```

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



Multi-threaded server (2)

- Server resource management – A pool of threads



Multi-threaded server (3)

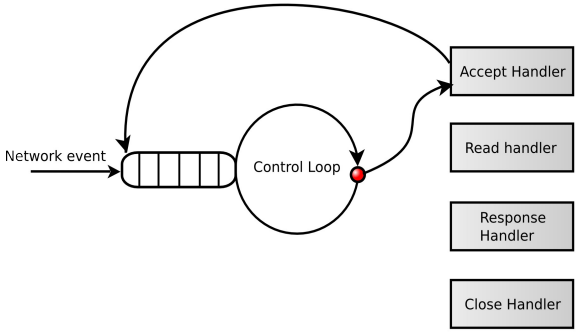
- Server resource management – A pool of thread

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

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


Event-based server

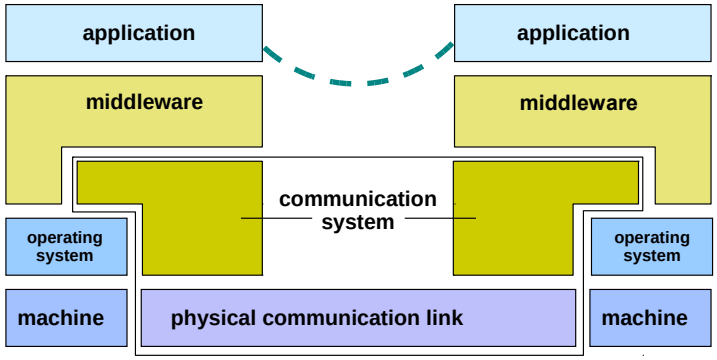
- Server resource management – A FSM



Outline

- Objectives and organization
- What is a distributed system ?
  - Communication mechanisms in distributed systems
  - Services and interfaces in computing systems
  - Client/server architecture
- **What is a middleware ?**
- Conclusion

What is a middleware



Functions of a middleware

- A middleware has mainly four functions :
  1) Make distribution as invisible (transparent) as possible
  2) Provide a homogeneous view of underlying heterogeneous hardware and software systems
  3) Provide services of common use for distributed systems
  4) 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

Middleware layers

- OS
- JVM
- RMI
- Servlet / JSP
- Sun J2EE / EJB

Incoming lectures and practical work on middleware

- Lectures
  - Introduction to distributed systems and middleware
  - *RMI -based distributed systems*
  - Servlet-based distributed systems
  - Introduction to multi-tier distributed Internet services