Distributed Computing

1

Why distributed systems: Benefits & Challenges

- The Sydney Olympic game system: see text page 29-30
- Divide-and-conquer
- Interacting autonomous systems
- Concurrencies
- Transactions

2

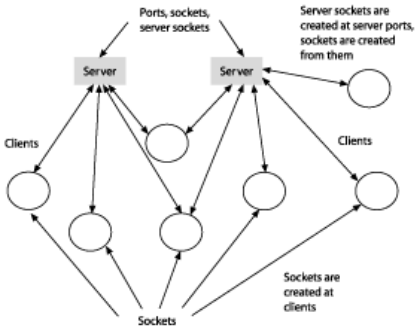
How to get it working

- Open systems, standards, protocols
- Message passing
- Distributed objects technology:
  - e.g. CORBA (multi-language technology)
  - RMI (Java-based technology)
  - COM (Microsoft)
  - future technologies and standards (?)

3

Reminder: Client-Server Example

- Server program which instantiate server **socket** that listens to a specific **port**
- Client program creates socket which connects to the server socket of the specified URL
- Exchanging string messages following a specific format (**protocol**)
- Synchronous vs. asynchronous message passing (see threads bellow)

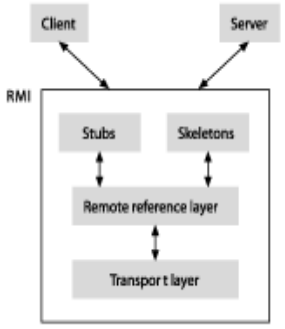


Ince, Figure 4.1

4

RMI: Remote Method Invocation

- See Ince, chapter 9
- Java technology that hides the details of communication
- In object-oriented computing we generate objects and call methods for objects. It should not matter where the objects and methods reside: the system should send objects over the network or execute methods on a different computer if necessary.
- → java.rmi package
- Extending RemoteObject
- RMI compiler → _Skel.class & _Stub.class



5

Developing a remote object system using RMI involves the following steps:

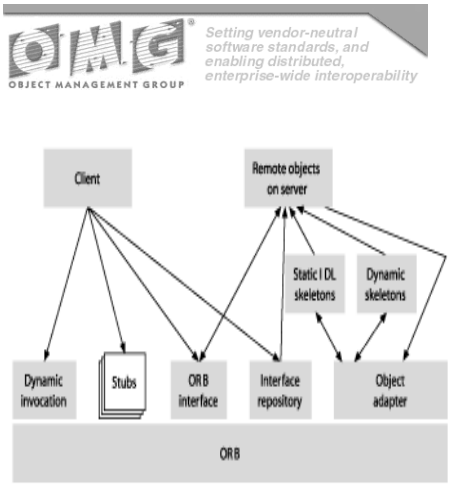
1. Develop the interface that describes the services provided by the remote object(s).
2. Implement the server code by developing a class which implements the remote interface. This class can contain the code which registers the object with the RMI registry as described here; however, the code can be placed in another class. Compile the code.
3. Implement the code for the client which uses the RMI registry to connect to the remote object. Compile the code.
4. Run the RMI compiler on the remote class in order to generate the stubs and skeletons and then make them available to the client and the server.
5. Run the RMI registry on the server.
6. Execute the class file associated with the server. The remote object that is associated with the server is now ready for receiving messages from code running on other computers.
7. Execute the class file associated with any client. This will hook into the remote object that was made available and carry out the processing that was required.

Ince, chapter 9

6

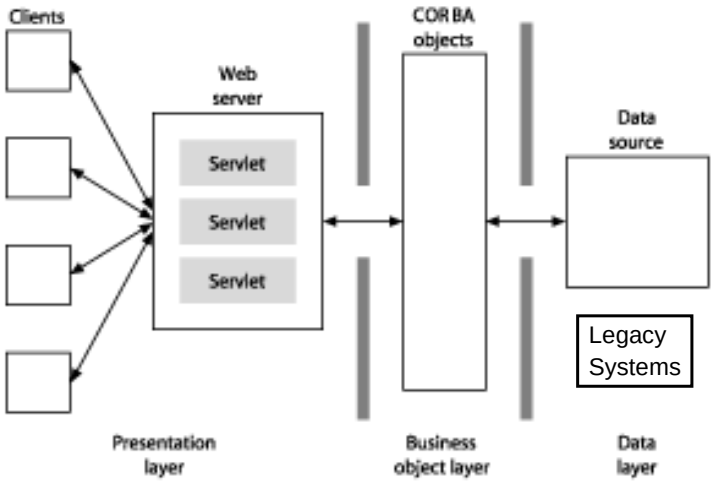
CORBA
(Common Request Broker Architecture)

- See Ince, Chapter 10
- OMG: Standardization of distributed object technology
- Is around for many years but has not become the widely supported standard as hoped for
- Language independent
  - IDL: Interface Definition Language
- CORBA services: name, concurrencies, security, time, etc.
- Commercial & free ORB products



7

A contemporary tiered e-commerce architecture



8

Concurrencies

- **Thread:** chunk of code that can be executed in parallel (independent) of other chunks of code. A **scheduler** supervises the execution of threads.
- In a distributed system we relay on independent processes, for example, we might have to wait for some requested data. In the meantime we could use the processor time for other tasks that might be pending.
  - Good distributed systems are multi-threaded
  - Parallel computing: `utilization of multiple processors'

9

Concurrency Control Problem

User A	Joint Account	User B
Read total: $100	$100	
		Read total: $100
Take out $50		
		Take out $50
Write total	$50	
	$50	Write total

This wouldn't be very good for the Bank!

10

Concurrencies and Locks

- If a thread depends on some data that should not be modified by another thread during its processing, then we have to **lock** that data.
- Client-servers: Write locks, read locks, … and dead locks (time limits, lock manager, …)
- Database servers: Row-, page-, table-, database-locking

11

Example: Distributed Database

- Reminder:
  - Relational database: series of tables that can contain links
  - Functions of a database server:
    - To interpret SQL statements
    - To optimize queries
    - To prevent the errors from concurrent access
    - To ensure internal consistency (referential integrity)
    - To detect and act upon deadlock
    - To administer security
    - To administer backup and recovery.
- A distributed database is one whose tables are distributed among a number of networked computers
- Issues: downloading, data replication, fragmentation

12

Transaction Processing

- A transaction is defined as a series of reads and writes of database objects" (Ramakrishnan, pg. 524)
- How a DBMS handles transactions is important for concurrency control and recovery

13

ACID Transaction Model

- Four important properties of transactions:
  1. Atomic: each transaction is either carried out completely or not at all
  2. Consistent: each transaction must preserve consistency of the database
  3. Isolated: transactions are not affected by concurrent transactions
  4. Durable: once a transactions has been completed, its effects should persist, even if the system crashes

14

Transaction Concurrency

- Why do we want concurrent transactions?
- What are the problems with concurrent transactions?
- What is the simplest way to handle concurrent transactions?
- A schedule is a potential execution sequence for the actions in a set of transactions

15

Transactions

- Sequence of operations
- ACID: Atomicity, Consistency, Isolation, Durability
- Server transactions vs. concurrent transactions
- Transaction monitors
  - e.g. Sun's Enterprise JavaBeans
    - interfaces
    - classes
    - containers
    - servers
    - transaction monitor
    - …

16