جامعة الأهرام الكندية
AHRAM CANADIAN UNIVERSITY

Lecture (08)
Distributed Objects

Dr. Ahmed ElShafee

Agenda

- Overview
- Introduction
- RMI & RPC
- RMI
- CORBA

Overview

What options do developer has for distributed application development?

Developer can choose several solutions for creating distributed applications programs.

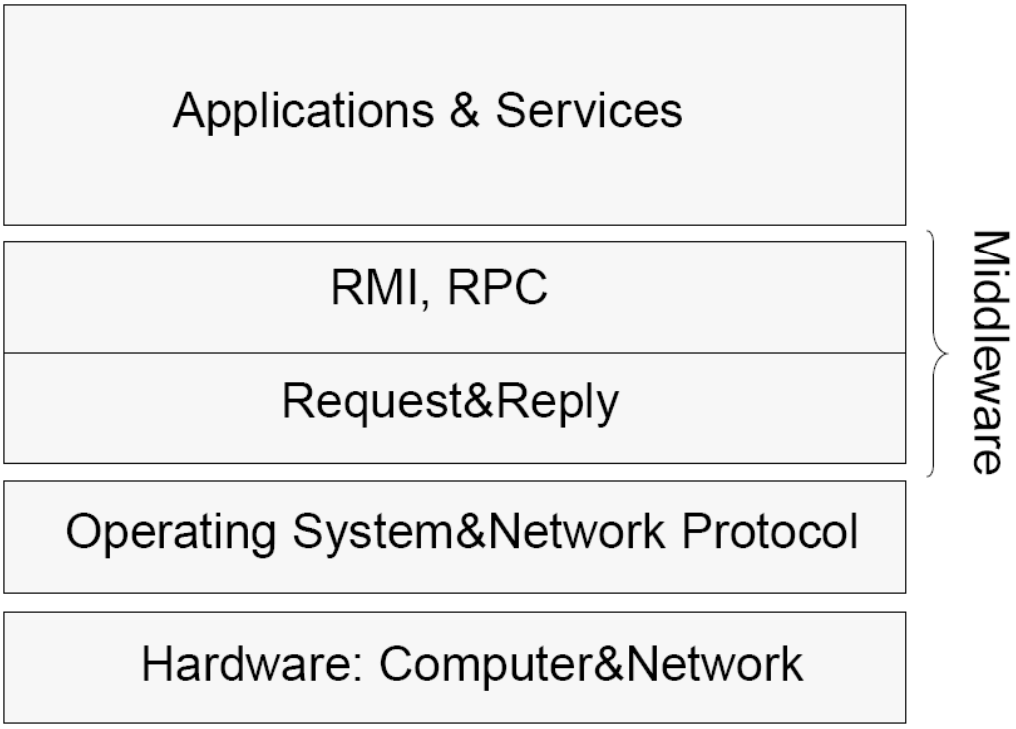
1) RMI/RPC technology

2) IDL technology (for CORBA programmers)

Introduction

- This lecture concentrates on communication between distributed objects by means of two models:
  - remote method invocation (RMI) and
  - remote procedure call (RPC).
- RMI, as well as RPC, are based on request and reply primitives.
- Request and reply are implemented based on the network protocol (e.g. TCP or UDP in case of the Internet).

Introduction (cont,..)



Introduction (cont,..)

The goal:

Make -for the programmer- distributed computing look like centralized computing.

The solution:

Asking for a service is solved by the client issuing a simple ***method invocation*** *or* ***procedure call****;* because the ***server*** can be on a ***remote machine*** this is a remote invocation (call).

RMI (RPC) is transparent*:*

the calling object (procedure) is not aware that the called one is executing on a different machine, and vice versa.

Introduction (cont,..)

Communication example:

The ***client*** writes:

- - - - - - - - - - - - - - - - - - -

server_id.service(values_to_server, result_arguments);

- - - - - - - - - - - - - - - - - - -

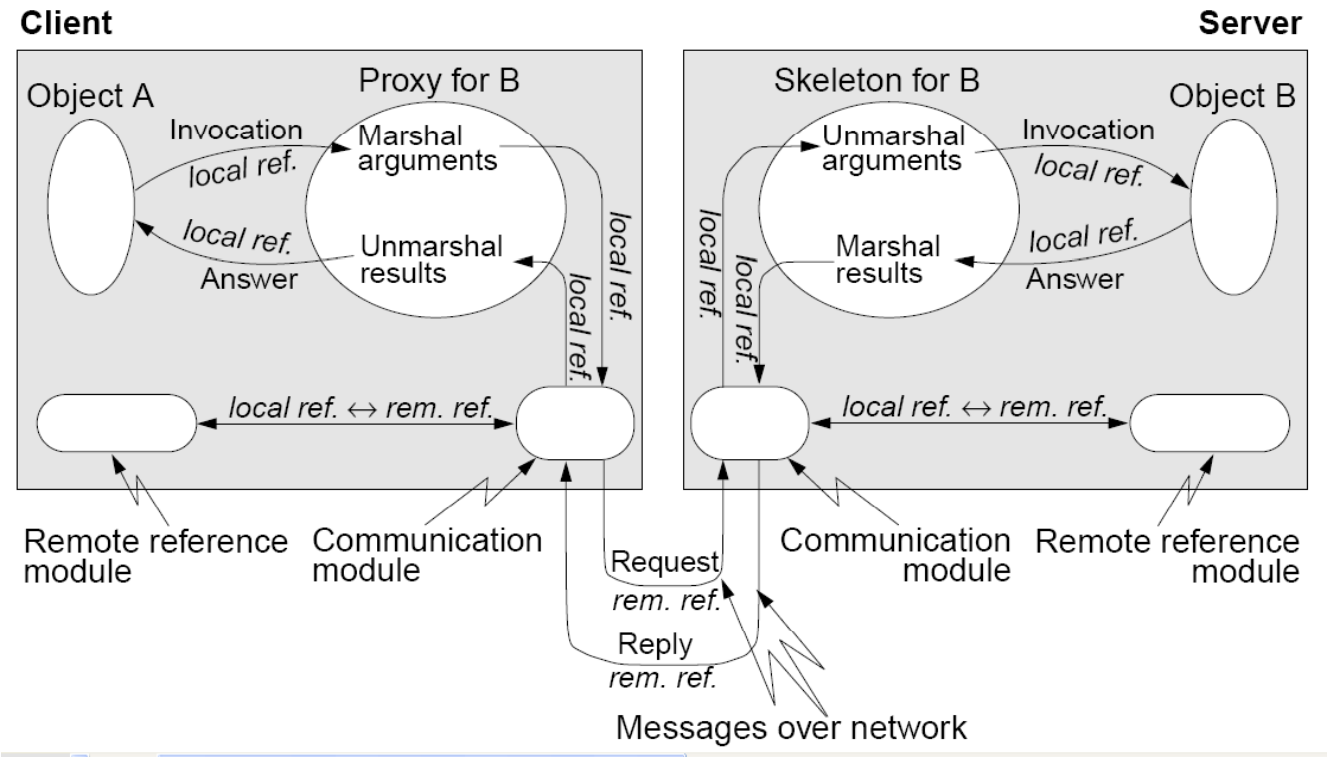
Introduction (cont,..)

The ***server*** contains the method:

```
public service(in type1 arg_from_client;
                  out type2 arg_to_client)
               {
               - - - - -
               - - - - -
               }
```

The programmer is unaware of the request and reply messages which are sent over the network during execution of the RMI.

Introduction (cont,..)



Introduction (cont,..)

Who are the players?

Object A asks for a service

Object B delivers the service

Who more?

The proxy for object B

The skeleton for object B

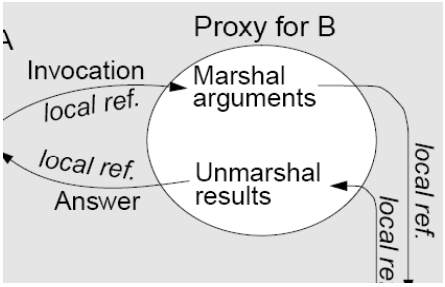
Communication module

Remote reference module

Introduction (cont,..)

1. The proxy for object B

- If an object A holds a remote reference to a (remote) object B, there exists a proxy object for B on the machine which hosts A.
- The proxy is created when the remote object reference is used for the first time.
- For each method in B there exists a corresponding method in the proxy.



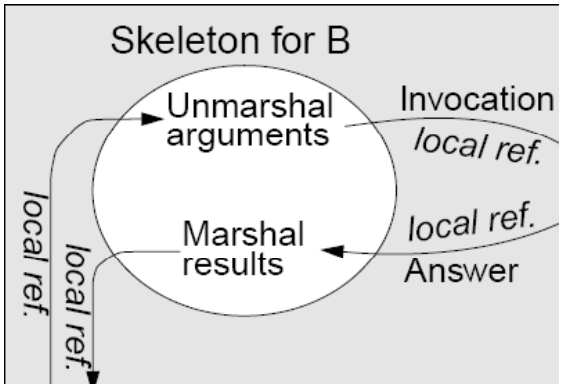
Introduction (cont,..)

- The proxy is the local representative of the remote object ⇒ the remote invocation from A to B is initially handled like a local one from A to the proxy for B.
- At invocation, the corresponding proxy method marshals the arguments and builds the message to be sent, as a request, to the server.
- After reception of the reply, the proxy unmarshals the received message and sends the results, in an answer, to the invoker.

Introduction (cont,..)

2. The skeleton for object B

- On the server side, there exists a skeleton object corresponding to a class, if an object of that class can be accessed by RMI.
- For each method in B there exists a corresponding method in the skeleton.



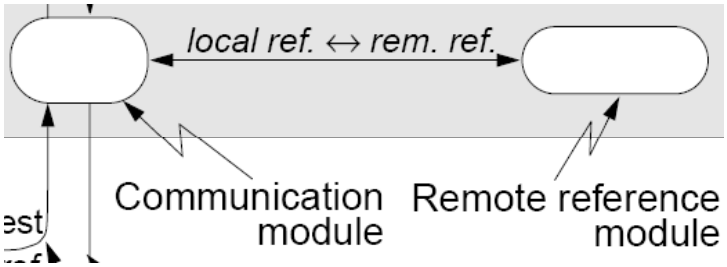
Introduction (cont,..)

- The skeleton receives the request message, unmarshals it and invokes the corresponding method in the remote object; it waits for the result and marshals it into the message to be sent with the reply.
- A part of the skeleton is also called dispatcher.
- The dispatcher receives a request from the communication module, identifies the invoked method and directs the request to the corresponding method of the skeleton.

Introduction (cont,..)

3. Communication module

- The communication modules on the client and server are responsible of carrying out the exchange of messages which implement the request/reply protocol needed to execute the remote invocation.
- The particular messages exchanged and the way errors are handled, depends on the RMI semantics which is implemented.



Introduction (cont,..)

4. Remote reference module

- The remote reference module translates between local and remote object references.
- The correspondence between them is recorded in a ***remote object table***.
- Remote object references are initially obtained by a client from a so called ***binder*** *that is part of* the global name service (it is not part of the remote reference module).
- Here servers register their remote objects and clients look up after services.

RMI & RPC

- Remote Method Invocation (RMI) technology was first introduced in JDK1.1.
- RMI allows programmers to develop distributed Java programs with the
- same syntax and semantics used for non-distributed programs.
- RMI is based on a similar, earlier technology for procedural programming
- called remote procedure call (RPC)

RMI & RPC (cont,..)

Disadvantages of RPC

- RPC supports a limited set of data types Therefore it is not suitable for passing and returning Java Objects
- RPC requires the programmer to learn a special interface definition language (IDL) to describe the functions that can be invoked remotely

RMI & RPC (cont,..)

The RMI architecture defines

1. How objects behave.
2. How and when exceptions can occur.
3. How memory is managed.
4. How parameters are passed to, and returned from, remote methods.

- The remote object model for Enterprise JavaBeans (EJB) is RMIbased.

RMI & RPC (cont,..)

- RMI is designed for Java-to-Java distributed applications.
- RMI is simpler and easier to maintain than using socket.
- Other options for creating Java-to-non-Java distributed applications are:
- a) Java Interface Definition Language (IDL)
- b) Remote Method Invocation (RMI) over Internet Inter-ORB Protocol
- (IIOP) -- RMI-IIOP.

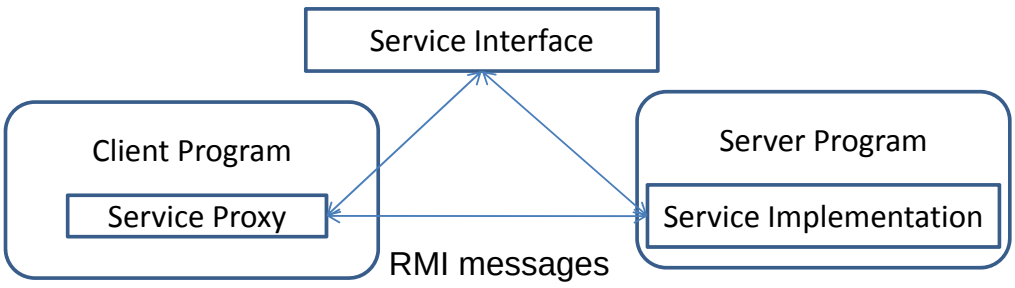
RMI

- RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.
- At client side, RMI uses interfaces to define behavior.
- At server side, RMI uses classes to define implementation.

RMI (cont,..)

The service interface is implemented by two classes.

1. The first one is at the server side which implements the behavior.
2. The second one is at the client side which acts as a proxy.

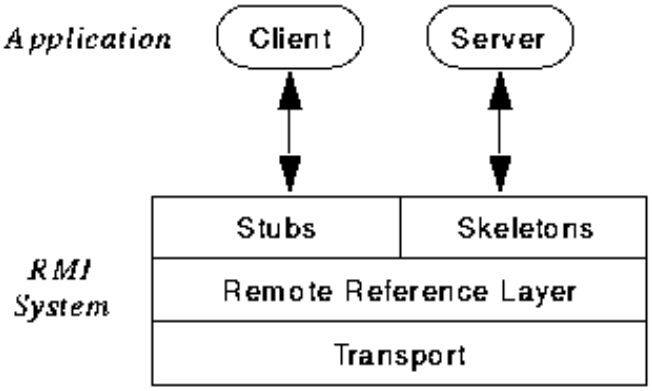


RMI (cont,..)

RMI layers

The RMI implementation is built from three abstraction layers.

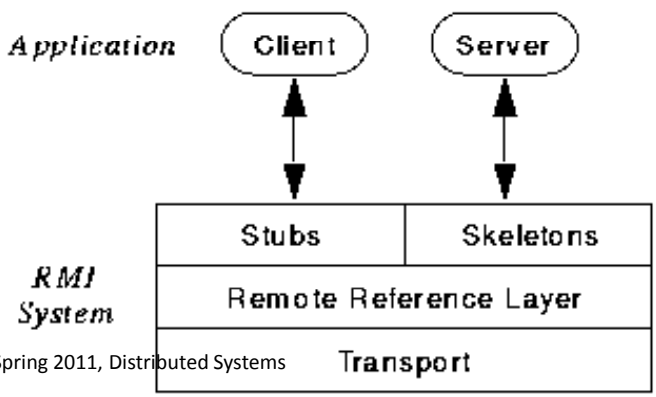
- a) The Stub (proxy) and Skeleton layer
- b) The Remote Reference
- c) The transport layer



RMI (cont,..)

Stub & Skelton layers

- The first layer lies beneath the view of the developer intercepts method calls made by the client to the interface reference variable and redirects these calls to a remote RMI service.
- RMI uses the Proxy design pattern in this layer.



RMI (cont,..)

- A skeleton is a helper class that is generated to communicate with the stub across the RMI link
- In the Java 2 SDK implementation of RMI, the new wire protocol has made skeleton classes obsolete.
- You only have to worry about skeleton classes and objects in JDK 1.1 and JDK 1.1 compatible system implementations.

RMI (cont,..)

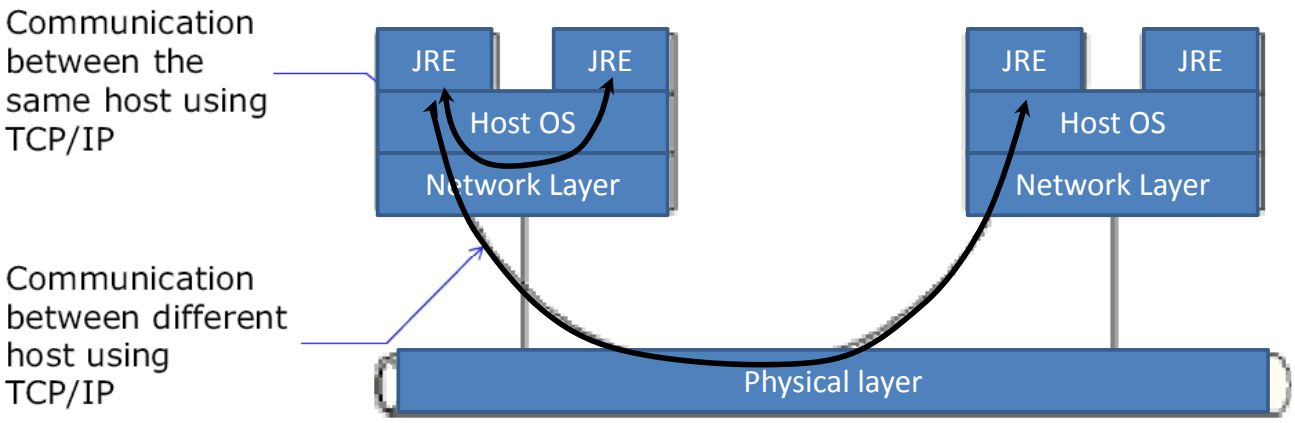
Remote reference layer

- This layer provides a RemoteRef object that represents the link to the remote service implementation object.
- In JDK 1.1, only unicast, point-to-point connection is supported.
- Before a client can use a remote service, the remote service must be instantiated on the server and ran all the time.
- In Java 2 SDK, client-server connection is added and activatable remote objects is supported .
- With the introduction of the RMI daemon, rmid, remote objects can be created and execute "on demand," rather than running all the time.

RMI (cont,..)

Transport layer

- The Transport Layer makes the connection between JVMs. All connections are stream-based network connections that use TCP/IP.



RMI (cont,..)

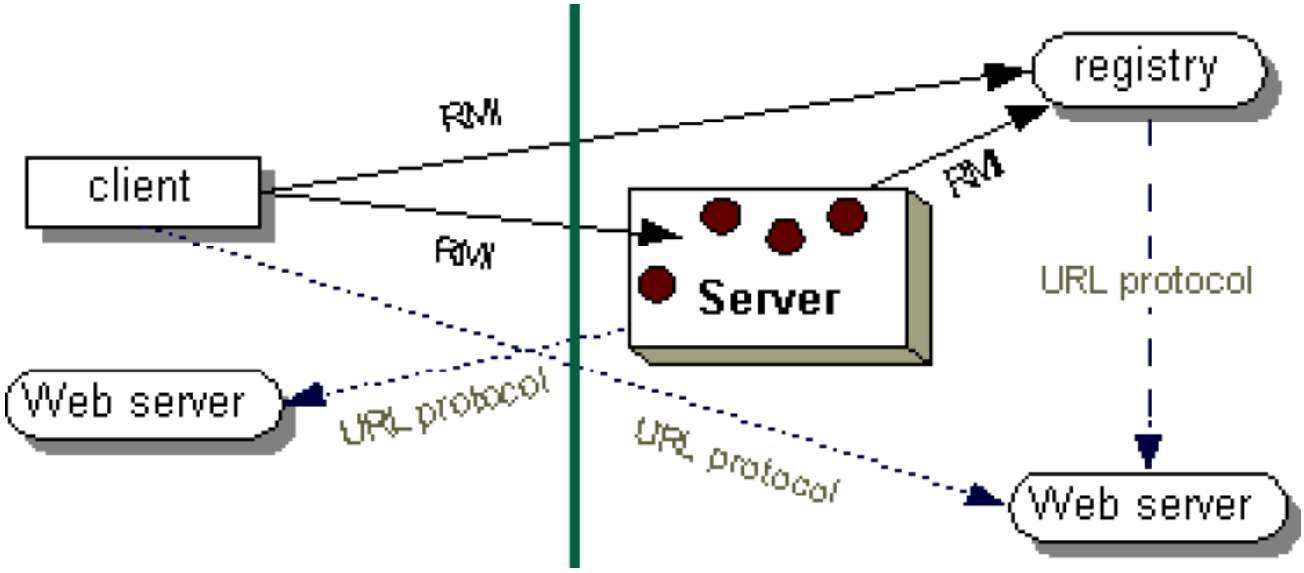
- On top of TCP/IP, RMI uses a wire level protocol called Java Remote Method Protocol (JRMP).
- JRMP is a proprietary, stream-based protocol.
- Sun and IBM have jointly worked on another version of RMI, called RMIIIOP(Remote Method Invocation over Internet Inter-ORB Protocol), which combines RMI-style ease of use with CORBA cross-language interoperability.
- The remote object model for Enterprise Java Beans(EJBs) is RMI-based.

RMI (cont,..)

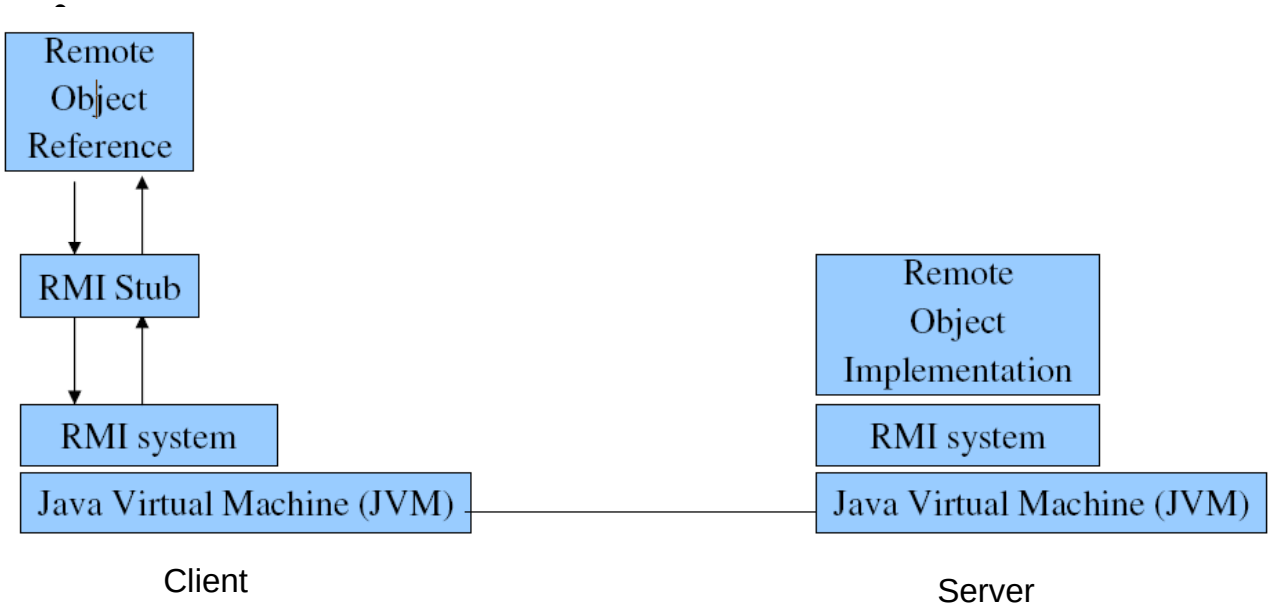
Naming Remote Objects

- In RMI, clients find remote services by using a naming or directory service.
- RMI can use many different directory services, including the Java Naming and Directory Interface (JNDI).
- RMI itself includes a simple service called the RMI Registry, rmiregistry.
- The RMI Registry runs on each machine that hosts remote service objects and accepts queries for services, by default on port 1099.

RMI (cont,..)



RMI (cont,..)



CORBA

Overview

- The main points in this section are:

1) The OMG and CORBA

2) basic concept of CORBA

2) CORBA Architecture

3) Object Request Broker (ORB)

4) Structure of a CORBA Application

5) Interface Definition Language

CORBA (cont,..)

The OMG

1) Object Management Group
    a) Founded in 1989
    b) Not-for-Profit organization
    c) Vendor neutral
    d) ~800 member companies
2) Key Specifications
    a) UML
    b) CORBA

CORBA (cont,..)

What is CORBA?

- Defines a framework for object-oriented distributed applications.
- Defined by a consortium of vendors under the direction of OMG.
- Allows distributed programs in different languages and different platforms nto interact as though they were in a single programming language on one computer.
- Brings advantages of OO to distributed systems.
- Allows you design a distributed application as a set of cooperating objects and to reuse existing objects

CORBA (cont,..)

Object Request Broker (ORB)

1) A software component that mediates transfer of messages from a program to an object located on a remote host.
2) Hides underlying network communications from a programmer.
3) ORB allows you to create software objects whose member functions can be invoked by client programs located anywhere.
4) A server program contains instances of CORBA objects.

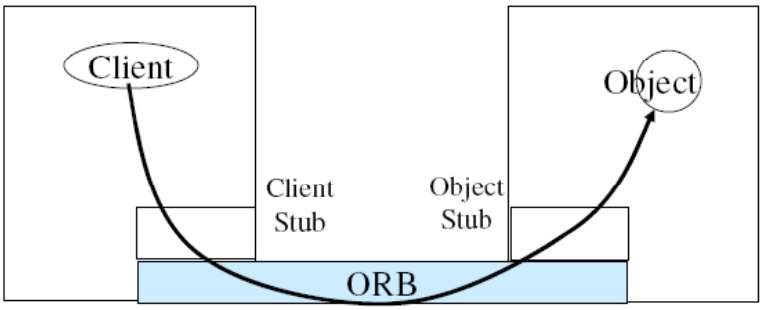
CORBA (cont,..)

ORB: Conceptual View

1) When a client invokes a member function on a CORBA object, the ORB intercepts the function call.
2) ORB directs the function call across the network to the target object.
3) The ORB then collects the results from the function call returns these to the function call.

CORBA (cont,..)

Implementation Details



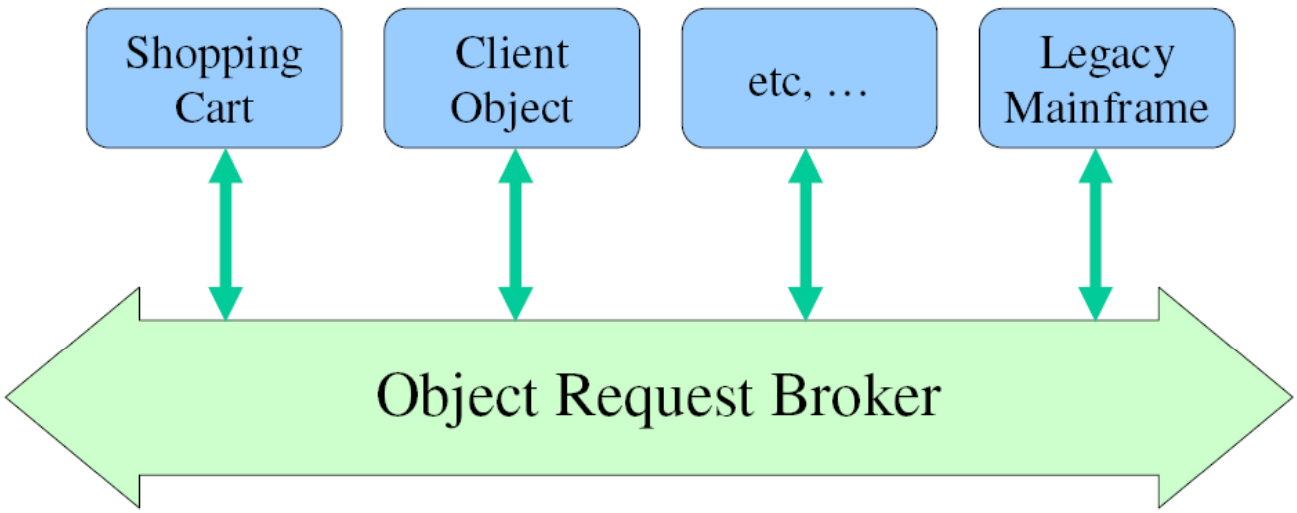
Access to the services provided by an Object

ORB : (Object-oriented middleware) Object Request Broker
ORB mediates transfer between client program and server object.

CORBA (cont,..)

CORBA: A "Software Bus"

- All CORBA objects connect to each other via ORB.



CORBA (cont,..)

CORBA IDL

1) Interface Definition Language
    a) used to generate application code (stubs/skeletons)
    b) language neutral (Ada, C++, Java, …)
2) IDL is NOT a programming language
    a) lacks control structures
    b) provides no implementation details
    c) a specification

CORBA (cont,..)

CORBA Objects and IDL

1) These are standard software objects implemented in any supported language including Java, C++ and Smalltalk.
2) Each CORBA object has a clearly defined interface specified in CORBA interface definition language (IDL).
3) The interface definition specifies the member functions available to the client without any assumption about the implementation of the object.

CORBA (cont,..)

Client and IDL

1) To call a member function on a CORBA object the client needs only the object's IDL.
2) Client need not know the object's implementation, location or operating system on which the object runs.

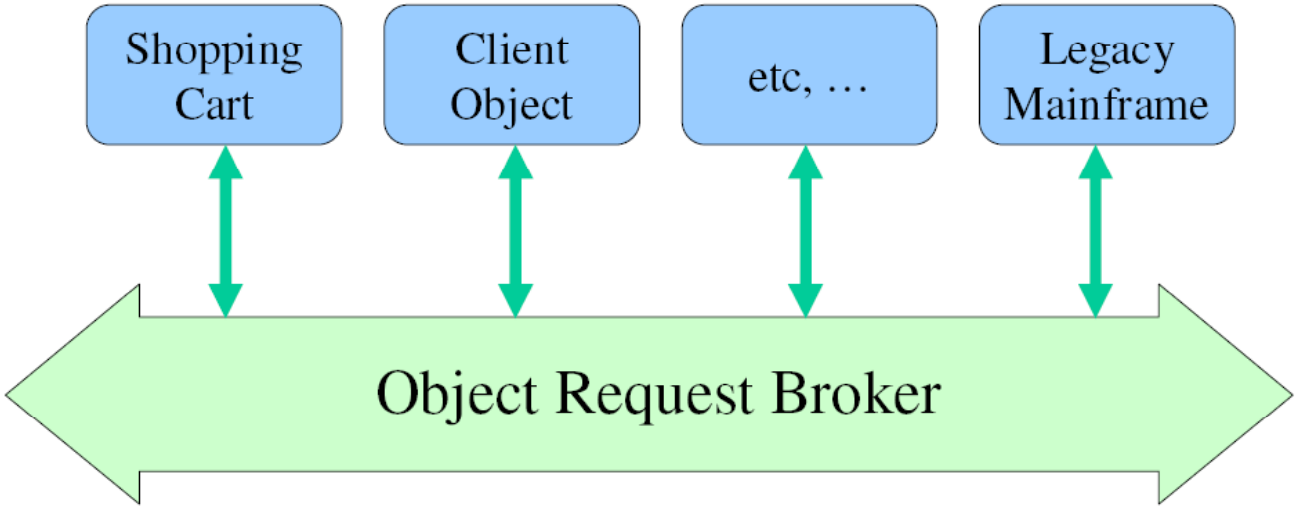
CORBA (cont,..)

Interface and Implementation

1) Interface and implementation can be in two different languages.
2) Interface abstracts and protects details (trade secrets) from client
3) Interface offers a means of expressing design without worrying about implementation.
4) Interface is separated from implementation

CORBA (cont,..)

CORBA Application Diagram



CORBA (cont,..)

CORBA Development Steps

1) Design the Application
2) IDL Specification
3) IDL Compilation (Code Generation)
4) Write the Client & Server implementation specific code
5) Compile the source code
6) Run the application

Thanks,

See you next Week, isA