Sistemas Distribuidos
Módulo 9

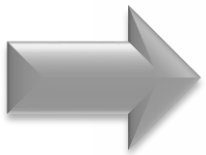
Objetos Distribuidos

Agenda

1. Conceptos de objetos y objetos distribuidos
2. Organización
3. Tipos
4. Procesos
5. Comunicación
    1. Modelo
    2. RMI
    3. Ejemplo: Java RMI
6. Sincronización, Consistencia y Replicación
7. Ejemplo Middleware: CORBA

Objetos Distribuidos - Conceptos

- Estado: encapsula los datos
- Métodos: operaciones sobre los datos
- Interfaz: se utiliza para la disponibilidad de los métodos



El *Estado* y la *Interfaz* se encuentran en máquinas distintas

Objetos Distribuidos

Objetos Distribuidos

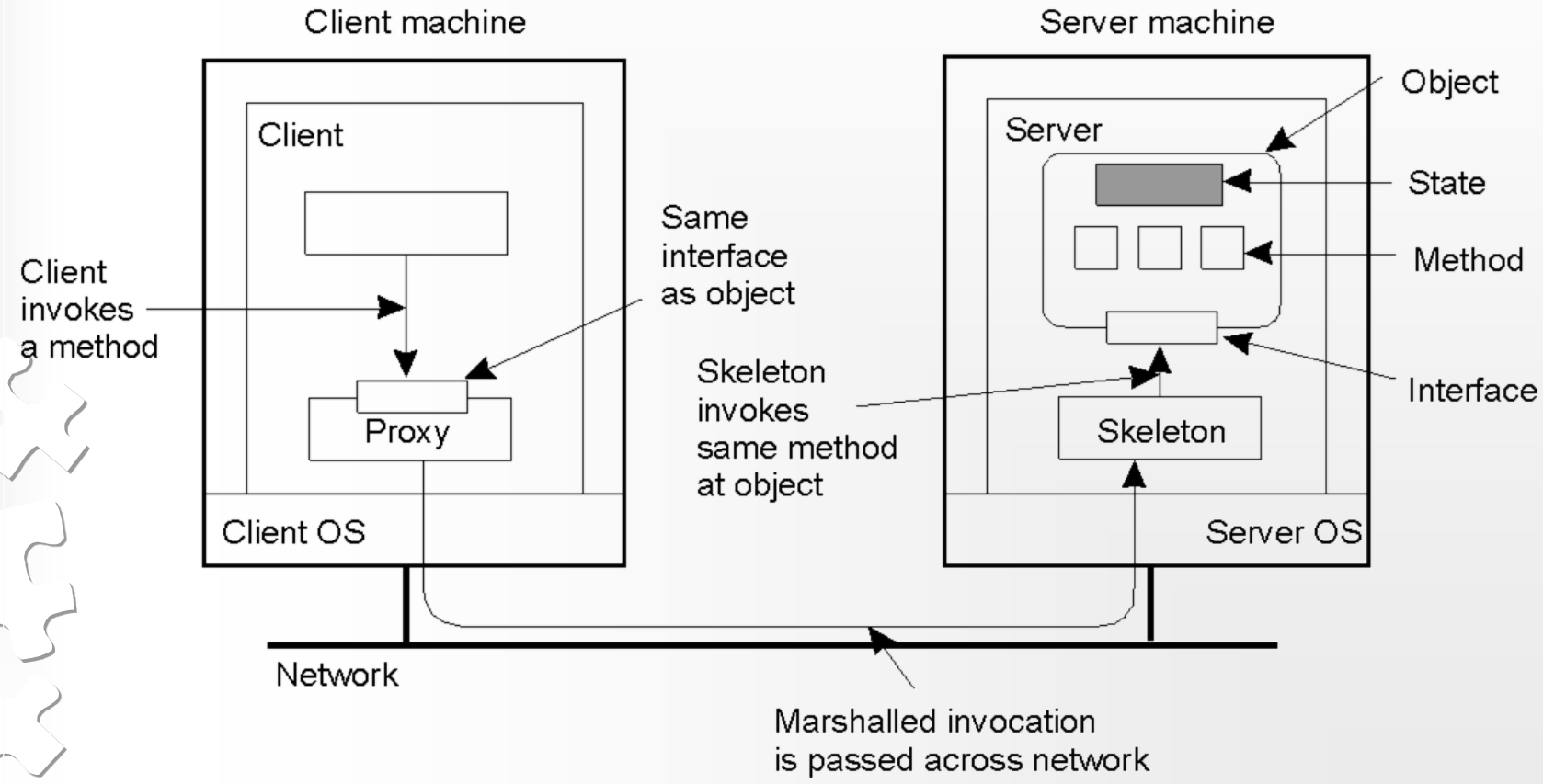
Objetos Distribuidos

- Referencias a Objetos Remotas
- Interfaces Remotas
- Acciones distribuidas
- Excepciones distribuidas
- Garbage collection distribuido

Objetos

- Referencias a Objetos
- Interfaces
- Acciones
- Excepciones
- Garbage collection

Objetos Distribuidos - Organización



Objetos Distribuidos - Tipos

- Tiempo de Compilación (Compile-Time)
- Tiempo de Ejecución (Runtime)
- Persistentes
- Transitorios

Ejemplos

- Enterprise Java Beans (*Objeto Remoto*)
- Objetos compartidos distribuido Globe (*Objeto Distribuido*)

Objetos Distribuidos - Procesos

- **Servidores de Objetos** – servidor diseñado para alojar objetos distribuidos.
  - Este tipo de servidores no proporciona, por sí mismo, un servicio específico.
  - Los servicios específicos son implementados por los objetos que residen en el servidor
- Invocación de Objetos
  - Qué código debe ejecutar
  - Qué datos debe operar
  - Si debe iniciar un nuevo thread para que se haga cargo de la invocación

Objetos Distribuidos - Procesos

Servidores de Objetos

- Invocación de Objetos
  - Soporte una única forma de invocación
  - Soporte diferentes formas de invocación
  - Adaptadores de objetos

Objetos Distribuidos - Comunicación

RMI (Remote Method Invocation) es una extensión de la invocación de métodos locales que permiten que un objeto que vive en un proceso invoque los métodos de un objeto que reside en otro proceso.

- El modelo de comunicación está construido sobre el protocolo **REQUEST-REPLY** y con semántica de llamada ***AL MENOS UNA VEZ*** o ***A LO SUMO UNA VEZ***.

Objetos Distribuidos – Comunicación

Binding un Cliente a un Objeto

(A) Un ejemplo con binding implícito utilizando solamente referencias globales.

```
Distr_object* obj_ref;          //Declaración de una referencia a un
                                // objeto del sistema (systemwide)

obj_ref = …;                    // Inicialización de la referencia al
                                // objeto ditribuido
obj_ref-> do_something();       // Bind implícito e invocación a un
                                // métdodos.
```

Objetos Distribuidos – Comunicación

Binding un Cliente a un Objeto

(B) Un ejemplo con binding explícito utilizando referencias globales y locales.

```
Distr_object objPref;          //Declaración de una referencia a un
                               //objeto del sistema (systemwide)
Local_object* obj_ptr;         //Declaración de un puntero a un
                               //objeto local
obj_ref = …;                   //Inicialización de la referencia al
                               //objeto distribuido.
obj_ptr = bind(obj_ref);       //Bind explícito y obtención de un
                               // puntero al proxy local.
obj_ptr -> do_something();     //Invocación de un método en el proxy
                               // local.
```

Objetos Distribuidos – Comunicación

Referencia a un Objeto

- Dirección de red de la máquina, dirección de Internet
- Identificación del servidor (puerto)
- Tiempo
- Identificación del objeto
- Interface del Objeto Remoto

32 bits	*32 bits*	*32 bits*	*32 bits*	
Dir Internet	Nro. puerto	tiempo	Nro.objeto	interface del Objeto remoto

Objetos Distribuidos – Comunicación

Forma de Invocación

- Estática

p. ej → Fobjeto.append(int**)**

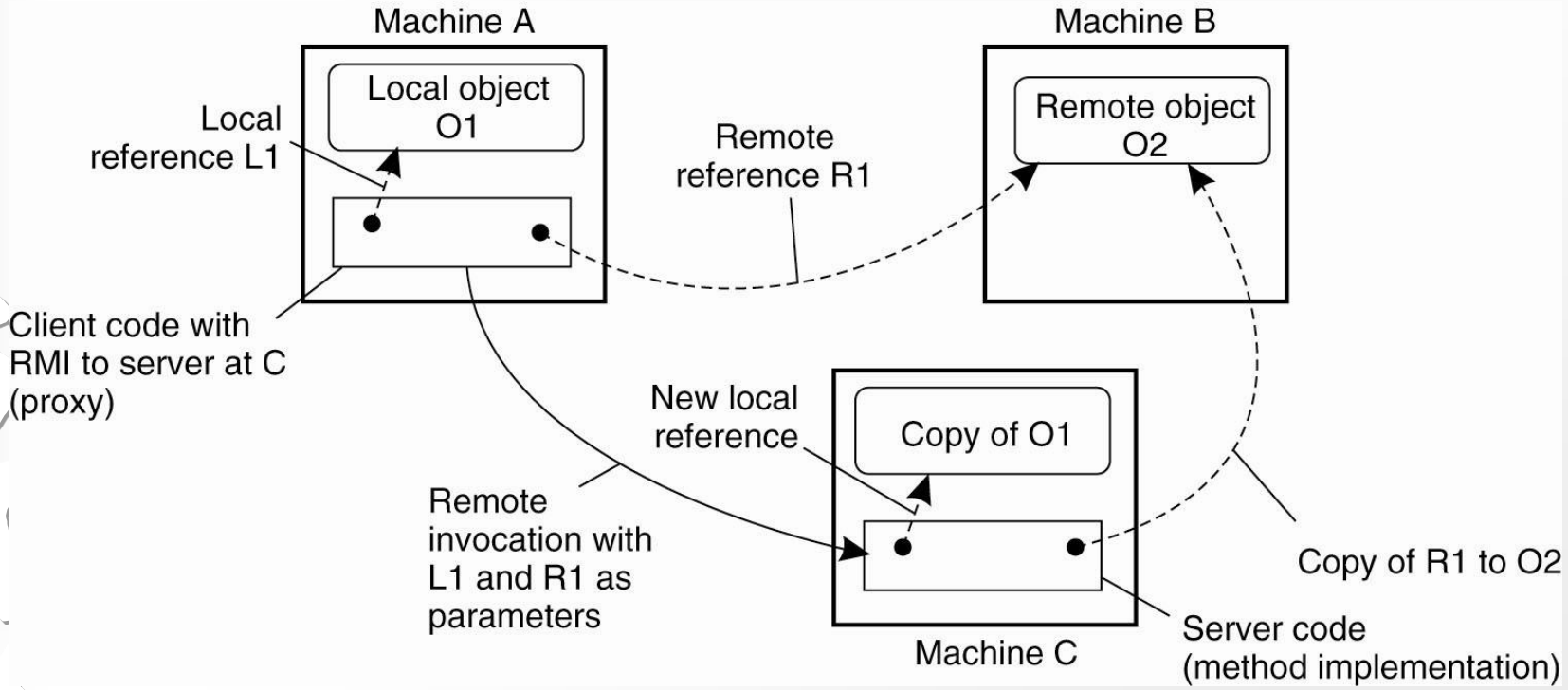
- Dinámica

→ Invoke(objeto, método, param entrada, param salida)

Invoke(fobjeto, id(append), int)

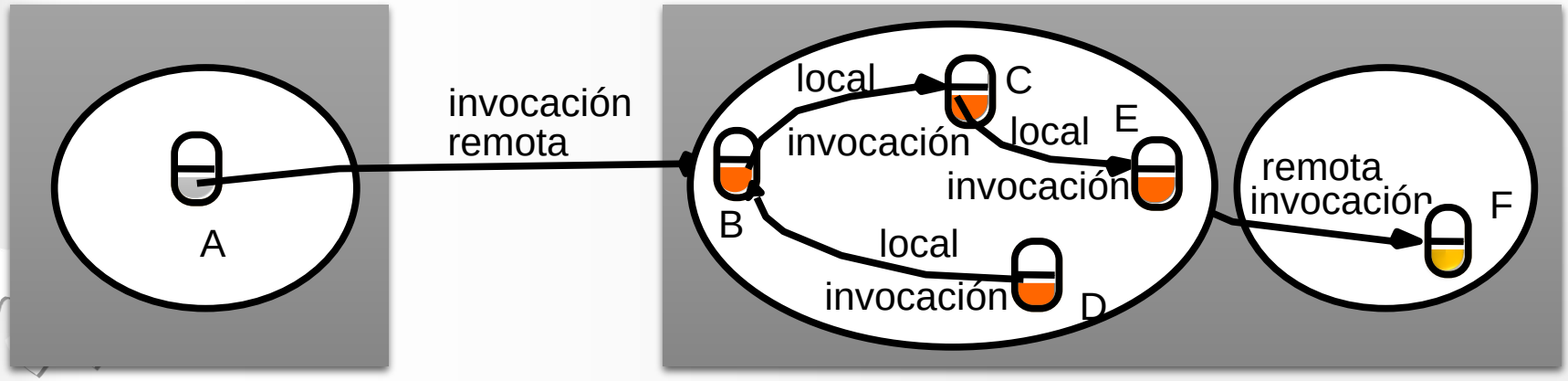
Objetos Distribuidos – Comunicación

- **Pasaje de Parámetros**



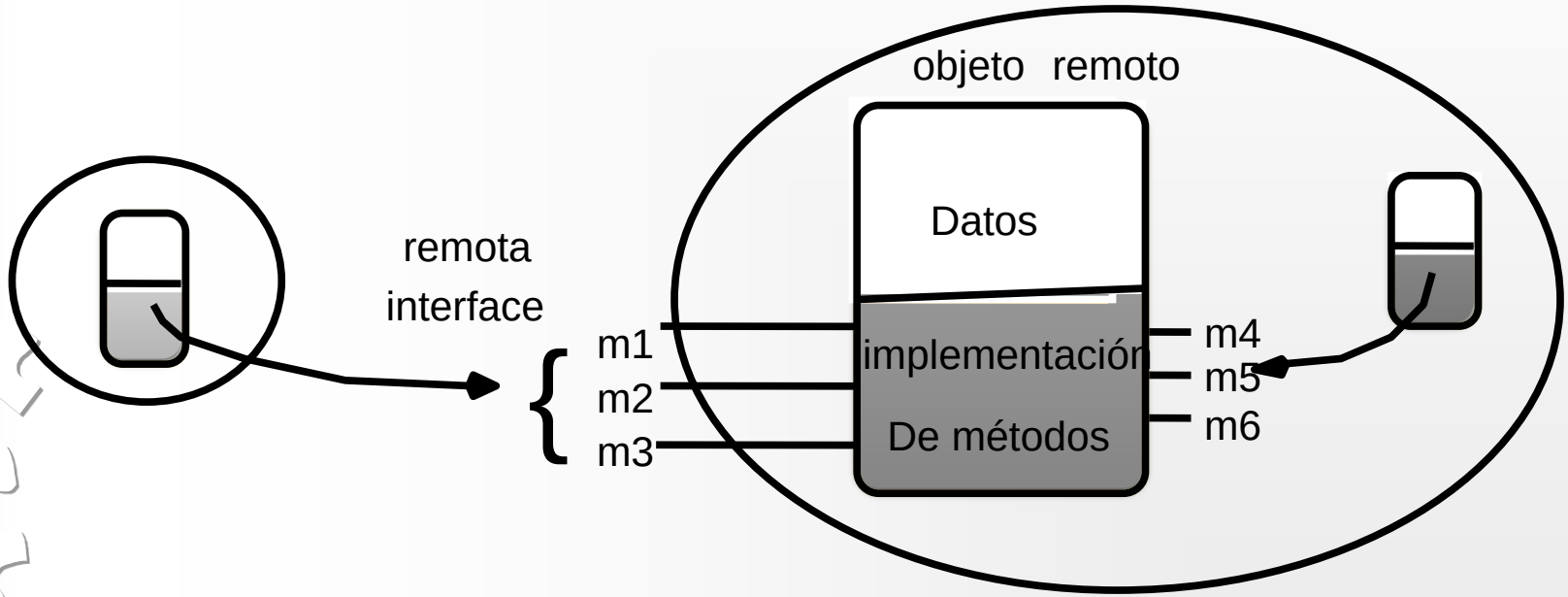
Objetos Distribuidos – Comunicación

Invocación de Métodos Locales y Remotos



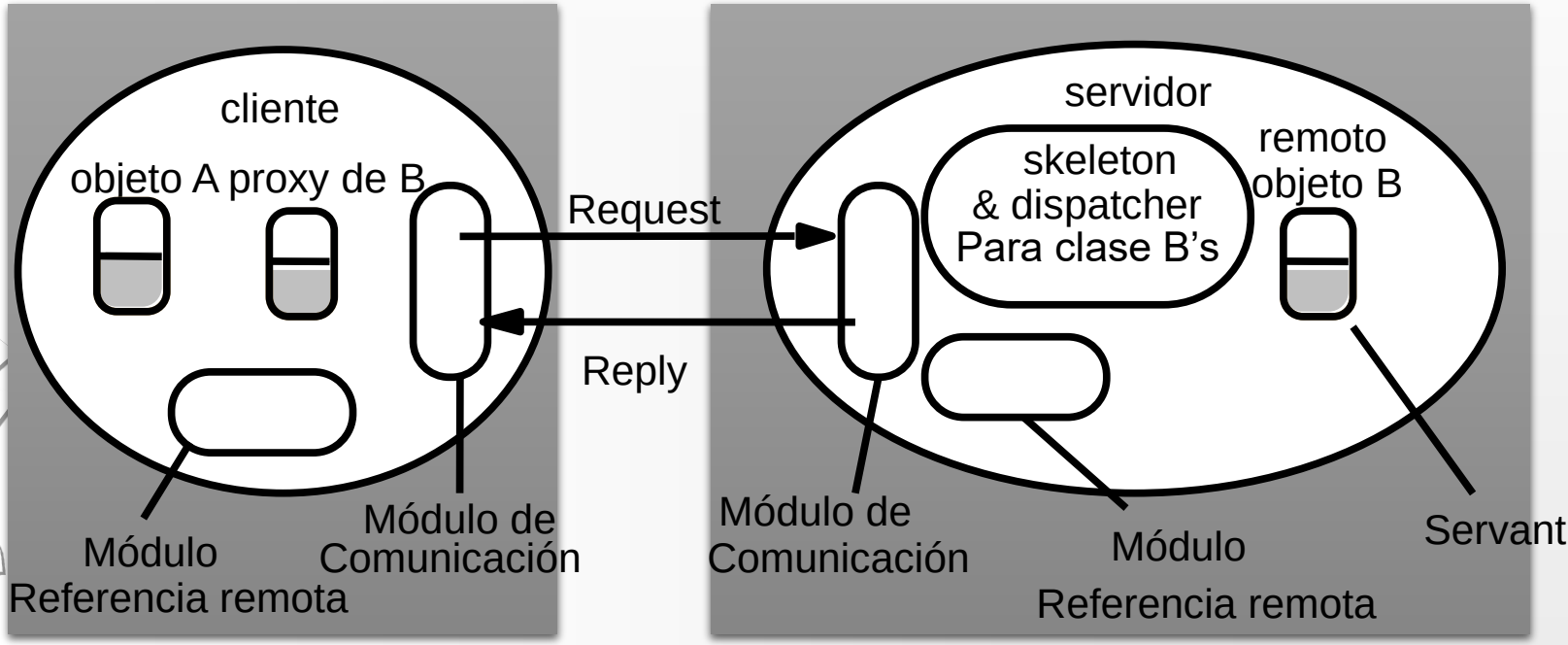
Objetos Distribuidos – Comunicación

- Objeto Remoto e Interfaces

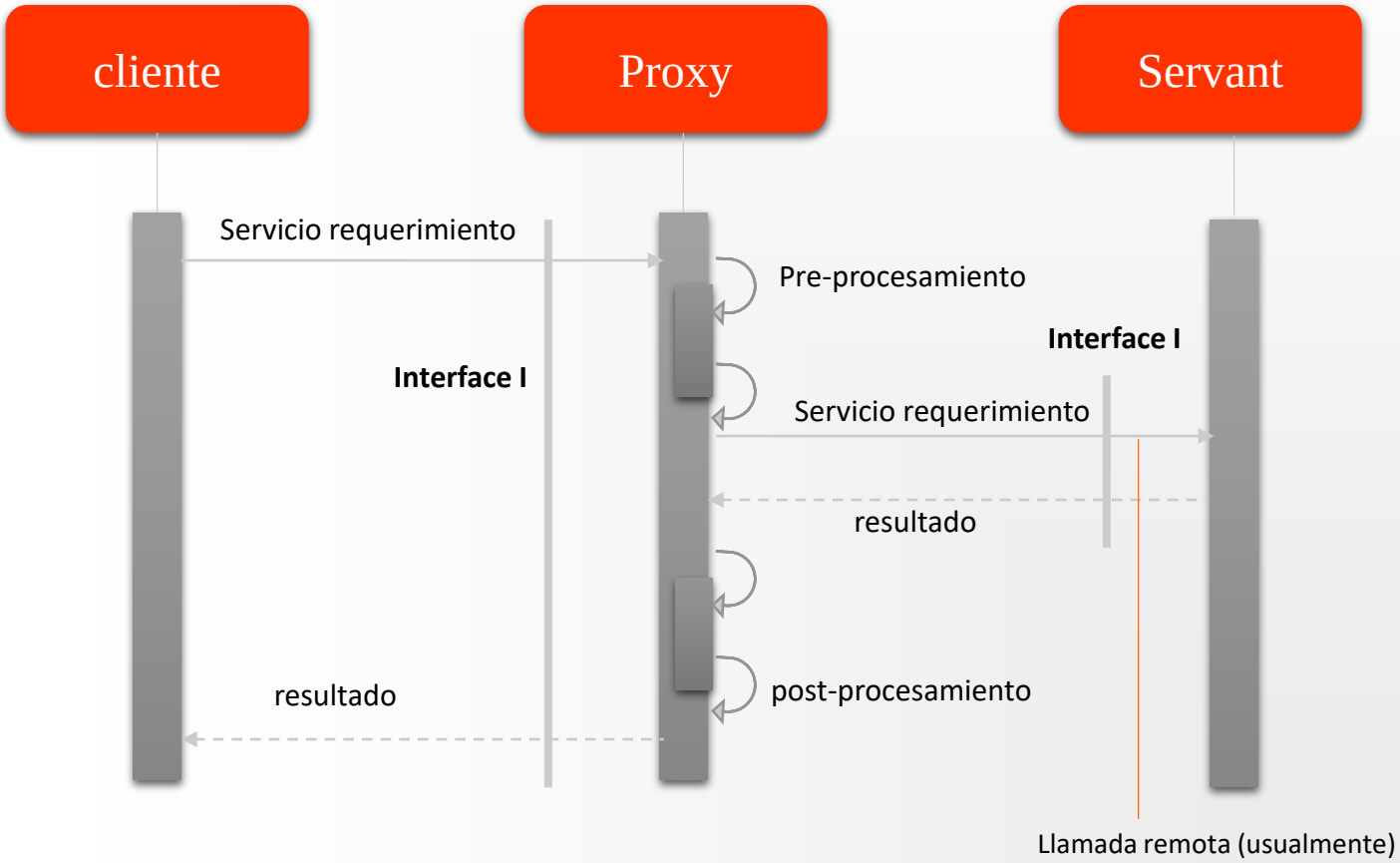


Objetos Distribuidos – Comunicación

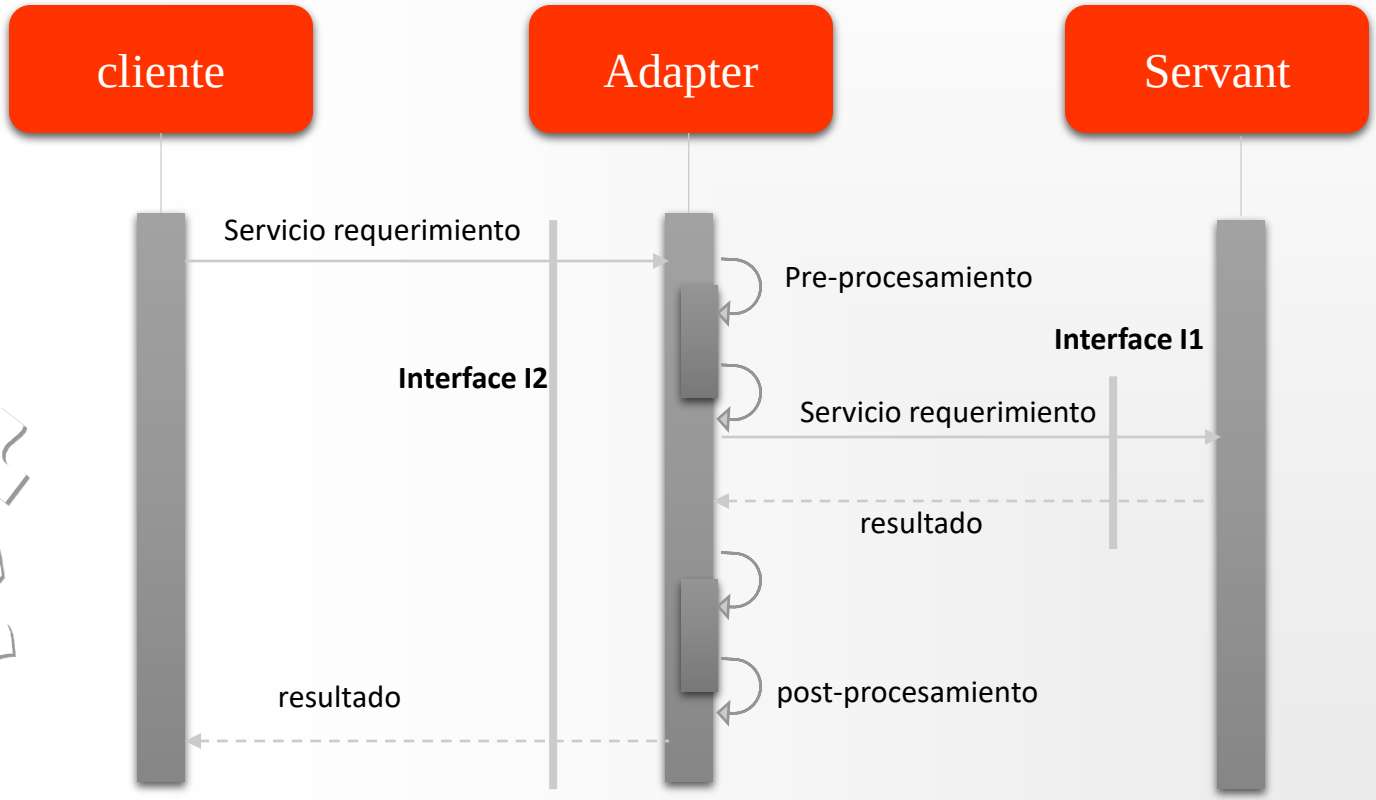
Arquitectura y Componentes



Objetos Distribuidos – Comunicación



Objetos Distribuidos – Comunicación



Objetos Distribuidos – Comunicación

Ejemplo - Java RMI

- Java Remote Method Invocation (Java RMI) permite crear aplicaciones distribuidas utilizando la tecnología basada en Java, permitiendo que los métodos de objetos remotos puedan ser invocados desde otra máquina virtual Java, ubicada posiblemente en otro nodo.
- RMI utiliza serialización de objetos para realizar el marshal y unmarshal de los parámetros y no truncar tipos, soportando el polimorfismo de la orientación a objetos.
- **Releases de Java RMI**
  - Java RMI está disponible para la Plataforma Java 2, Standard Edition (J2SE) y la Plataforma Java 2, Micro Edition (J2ME).

Objetos Distribuidos – Comunicación: Java RMI

- Pasos para realizar una aplicación distribuida
  1. Diseñar la interfaz remota
  2. Diseñar el programa servidor
  3. Diseñar el programa cliente
  4. Compilar los fuente y generar los stubs

Objetos Distribuidos – Comunicación: Java RMI

1.- Diseñar la Interfaz Remota

ReceiveMessageInterface.java

```
import java.rmi.*;
public interface ReceiveMessageInterface extends Remote
{
    void receiveMessage(String x) throws RemoteException;
}
```

Objetos Distribuidos – Comunicación: Java RMI

2.- Diseñar el programa Servidor

RmiServer.java

```
import java.rmi.*;
import java.rmi.registry.*;
import java.rmi.server.*;
import java.net.*;
public class RmiServer extends java.rmi.server.UnicastRemoteObject
implements ReceiveMessageInterface
{
   int     thisPort;
   String   thisAddress;
   Registry registry;    // rmi registry buscar los objetos remoto.
// Es la implementación de ***ReceiveMessageInterface***.
public void receiveMessage(String x) throws RemoteException
{
      System.out.println(x);
   }
```

Objetos Distribuidos – Comunicación: Java RMI

```
public RmiServer() throws RemoteException
  {
    try{
       // get the address of this host.
       thisAddress=
  (InetAddress.getLocalHost()).toString();
    }
    catch(Exception e){
       throw new RemoteException("can't get inet
  address.");
    }
  thisPort=15500;  // puerto seleccionado

  System.out.println("direccion="+thisAddress+",p
  uerto="+thisPort);
    try{
    // creación del registry y  bind del nombre y
  objeto.
    registry = LocateRegistry.createRegistry(
  thisPort );
    registry.rebind("rmiServer", this);
    }
    catch(RemoteException e){
    throw e;
    }
  } static public void main(String args[])
  {
    try{
    RmiServer s=new RmiServer();
  }
  catch (Exception e) {
       e.printStackTrace();
       System.exit(1);
  }
   }
}
```

Objetos Distribuidos – Comunicación: Java RMI

3.- Diseñar el programa cliente

RmiClient.java

```
import java.rmi.*;
import java.rmi.registry.*;
import java.net.*;
public class RmiClient
{
   static public void main(String args[])
   {
     ReceiveMessageInterface rmiServer;
     Registry registry;
     String serverAddress=args[0];
     String serverPort=args[1];
     String text=args[2];
     System.out.println(“Enviando "+text+" a  "+ serverAddress
   +":"+serverPort);
```

Objetos Distribuidos – Comunicación: Java RMI

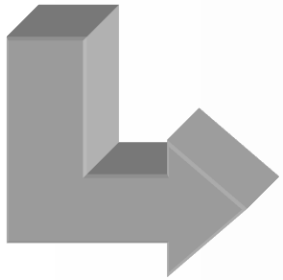
RmiClient.java (continuación)

```
try{
        // obtener el registry"
        registry=LocateRegistry.getRegistry(serverAddress,
           (new Integer(serverPort)).intValue()
        );
        // buscar el objeto remoto
        rmiServer= (ReceiveMessageInterface)(registry.lookup("rmiServer"));
        // call the remote method
        rmiServer.receiveMessage(text);
     }
     catch(RemoteException e){
        e.printStackTrace();
     }
     catch(NotBoundException e){
        e.printStackTrace();
     }
  }
}
```

Objetos Distribuidos – Comunicación: Java RMI

4.- Compilar los fuentes y generar los stubs

- javac RmiServer.java ReceiveMessageInferface.java
- javac RmiClient.java ReceiveMessageInterface.java
- rmic –classpath . RmiServer



RmiServer_Skel.class
RmiServer_Stub.class

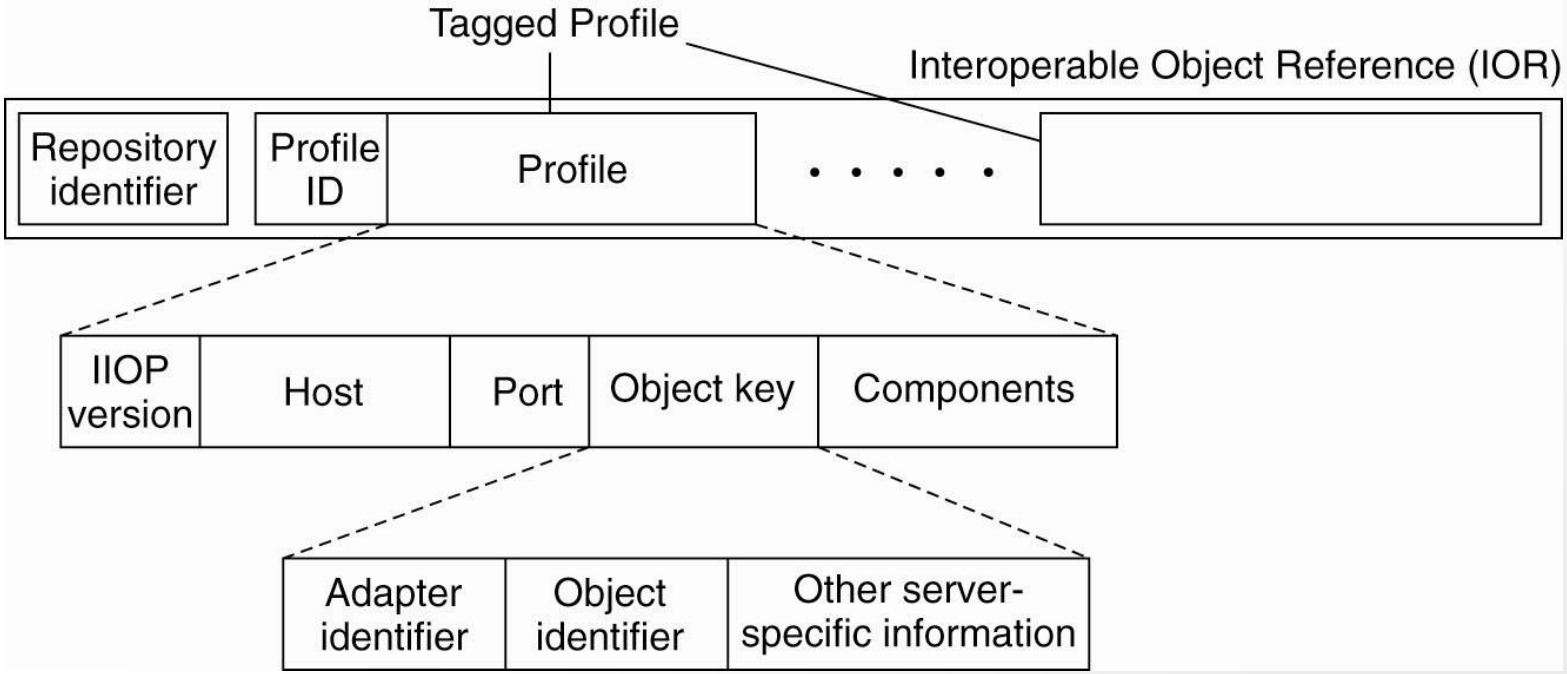
Objetos Distribuidos – Asignación de Nombres

- Vinculada con el lenguaje de programación. Por ejemplo: proxy de Java

- Independiente del lenguaje y la plataforma. Por ejemplo en CORBA, se utiliza una referencia a objeto interoperable (IOR).

Objetos Distribuidos – Asignación de Nombres

Organización del IOR



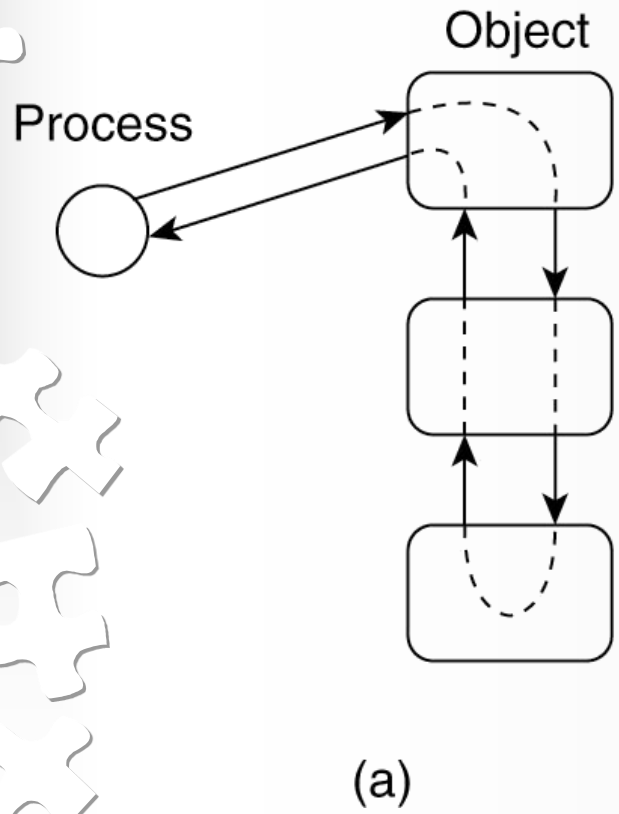
Objetos Distribuidos

Aspectos a tener en cuenta son:

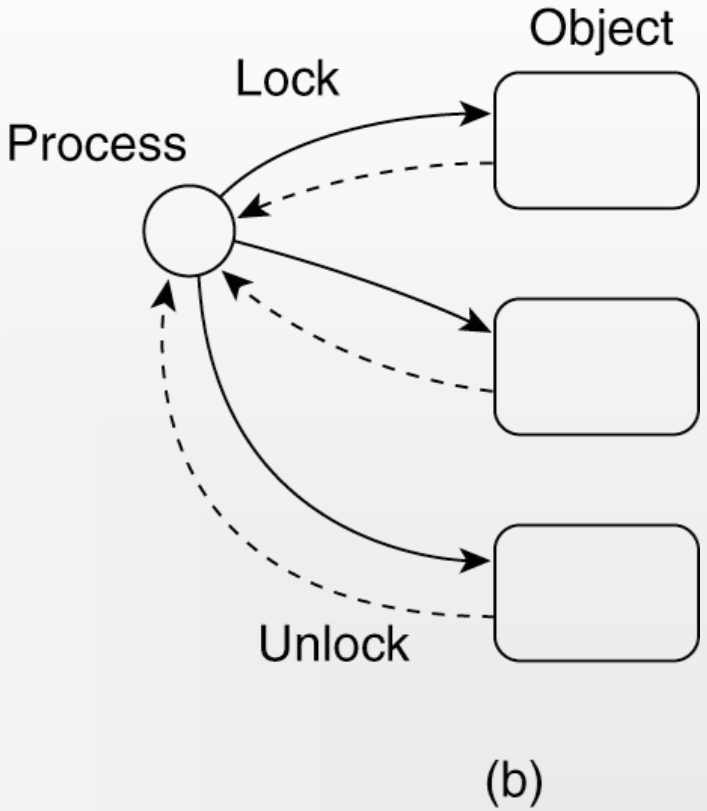
- **Consistencia**
  - Ejecución concurrente. Sincronización. Ordenamiento.
- **Replicación**
  - Atención de un requerimiento cuando hay réplicas entre una o varias invocaciones.

Objetos Distribuidos – Sincronización

- Diferencias en el control de flujo para bloquear procesos

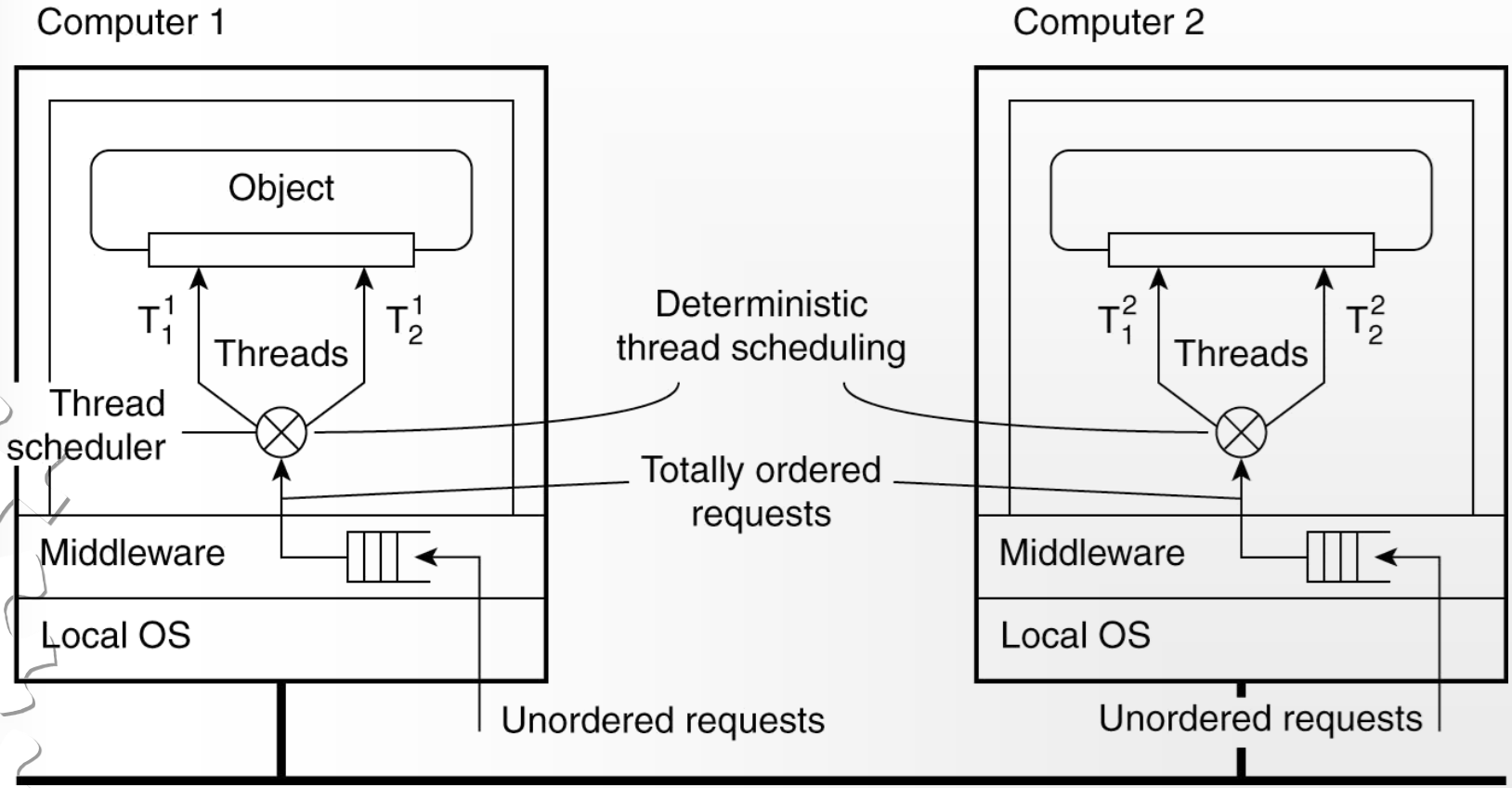


(a)



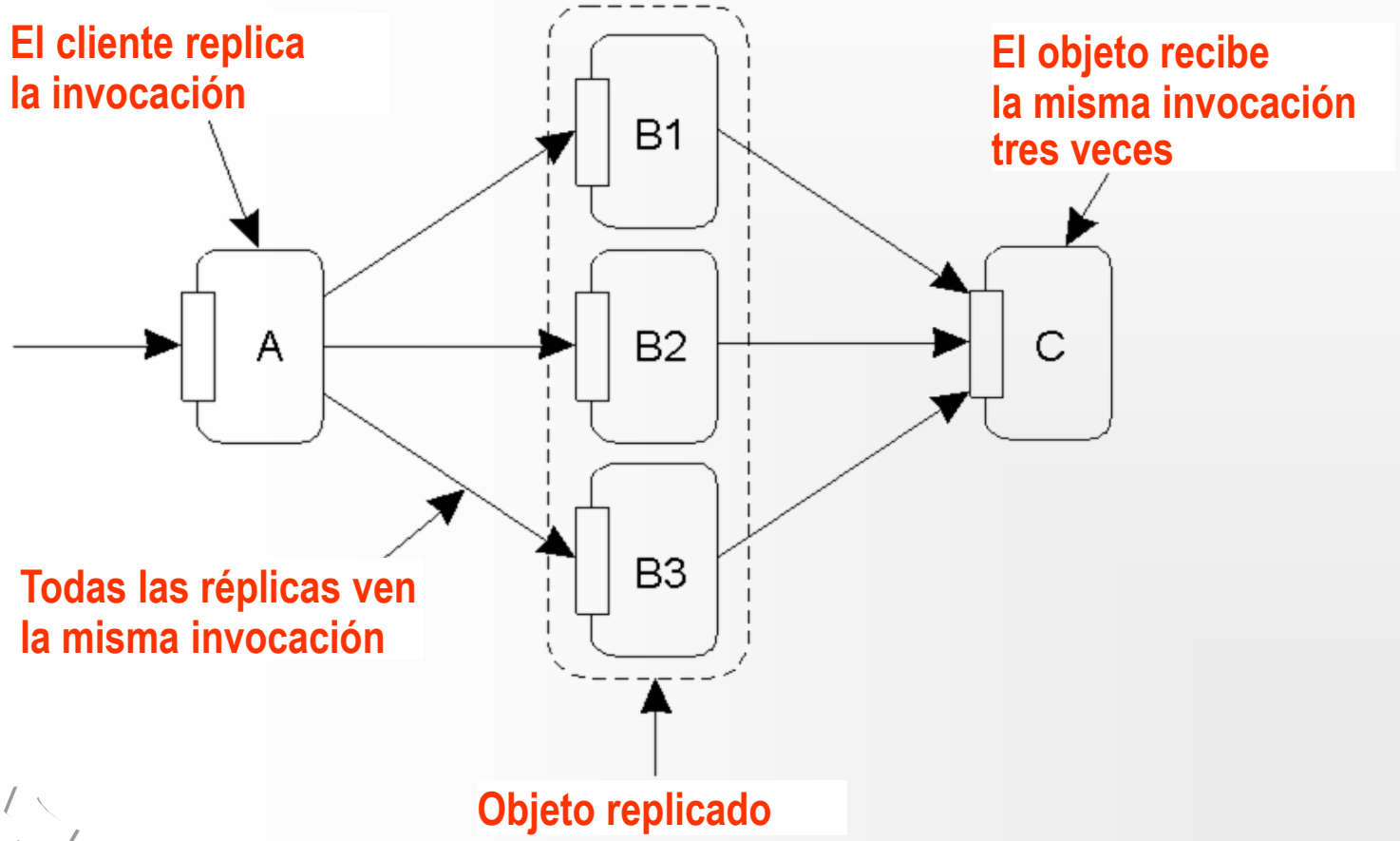
(b)

Objetos Distribuidos – Consistencia



Objetos Distribuidos – Replicación

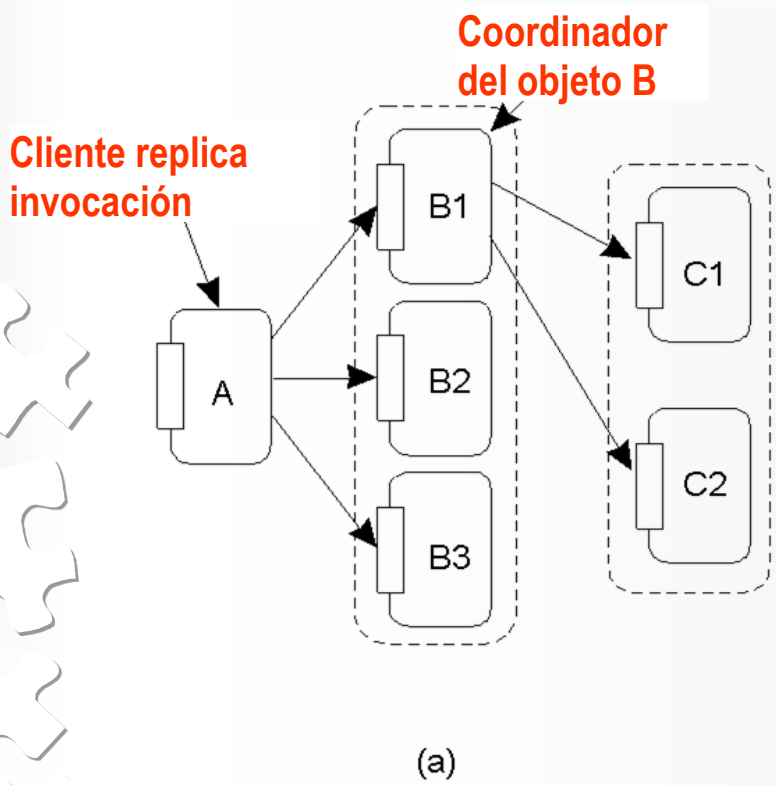
El problema de invocaciones replicadas.

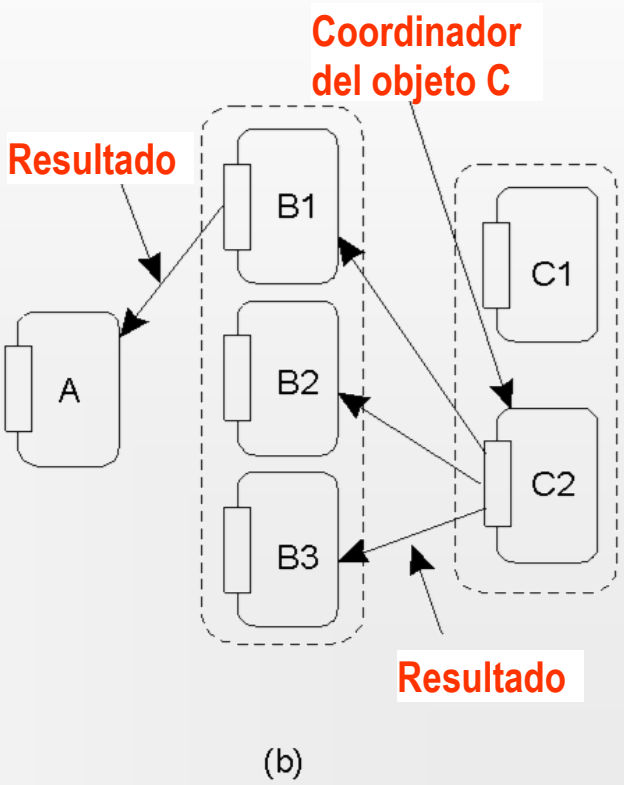


Objetos Distribuidos – Replicación

a) Reenvío de una invocación desde un objeto replicado.

b) Retorno de una respuesta al objeto replicado.





Objetos Distribuidos

Common Object Request Broker Architecture - CORBA

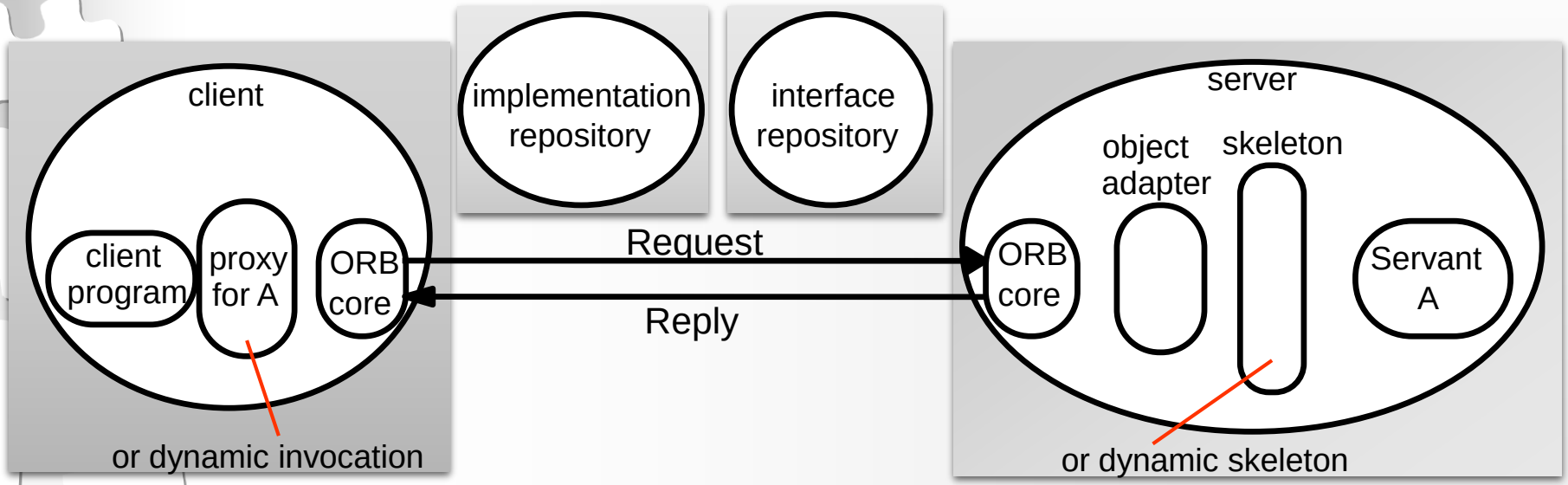
- CORBA es un diseño de middleware que permite que los programas de aplicación se comuniquen unos con otros con independencia de sus lenguajes de programación, sus plataformas hardware y software, las redes sobre las que se comunican y sus implementaciones.

Objetos Distribuidos - CORBA

Componentes independientes del esquema RMI

- Un lenguaje de definición de interfaces conocido como IDL
- Una arquitectura
- Estándares de comunicación. General Inter-ORB (GIOP)
- Internet Inter-ORB Protocol (IIOP)

Objetos Distribuidos - CORBA



Bibliografía:

- Coulouris,G.F.; Dollimore, J. y T. Kindberg; “Distributed Systems: Concepts and Design”. 5th Edition Addison Wesley, 2011.
- Tanenbaum, A.S.; van Steen, Maarten; “Distributed Systems: Principles and Paradigms”. 2nd Edition, Prentice Hall, 2007 and 1st Edition 2002.