Paradigmas avanzados de computación distribuida

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid

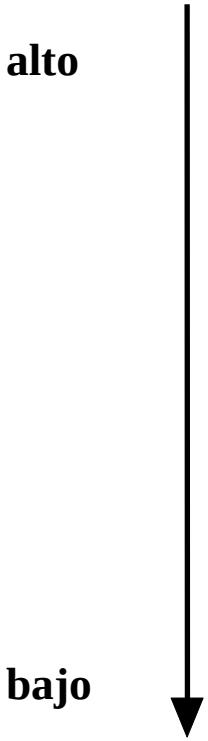
Contenidos

1. Sistema de colas de mensajes
2. Agentes móviles
3. Servicios de red
4. Espacio de objetos

Contenidos

1. **Sistema de colas de mensajes**
2. Agentes móviles
3. Servicios de red
4. Espacio de objetos

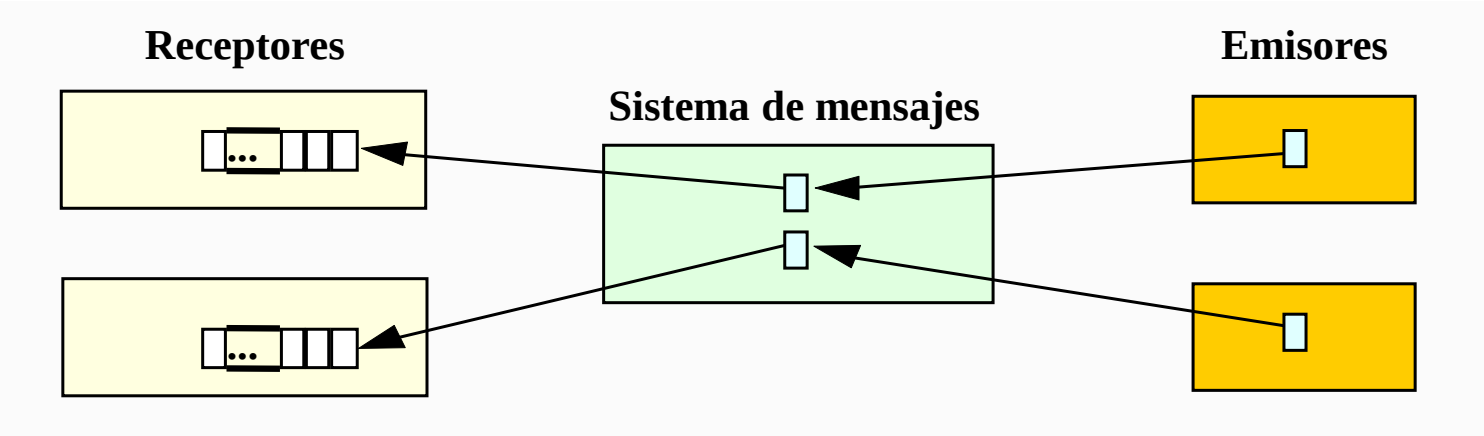
Paradigma de sistemas de colas de mensajes



Espacio de objetos, aplicaciones colaborativas
Servicios de red, *object request broker*, agentes móviles
procedimientos remotos, métodos remotos
Cliente-servidor, *peer-to-peer*, MOM
Paso de mensajes

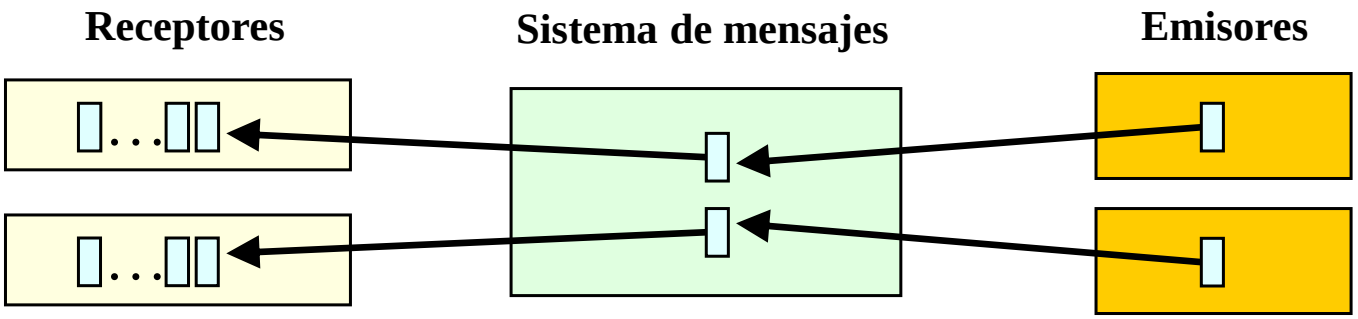
Paradigma del sistema de mensajes

- También denominado *middleware* orientado a mensajes (MOM)
- El sistema de mensajes actúa de intermediario entre los procesos que se comunican
- Proceso:
  - Emisión al sistema de mensajes
  - Almacenamiento en la cola asociada al receptor
  - Envío al proceso receptor

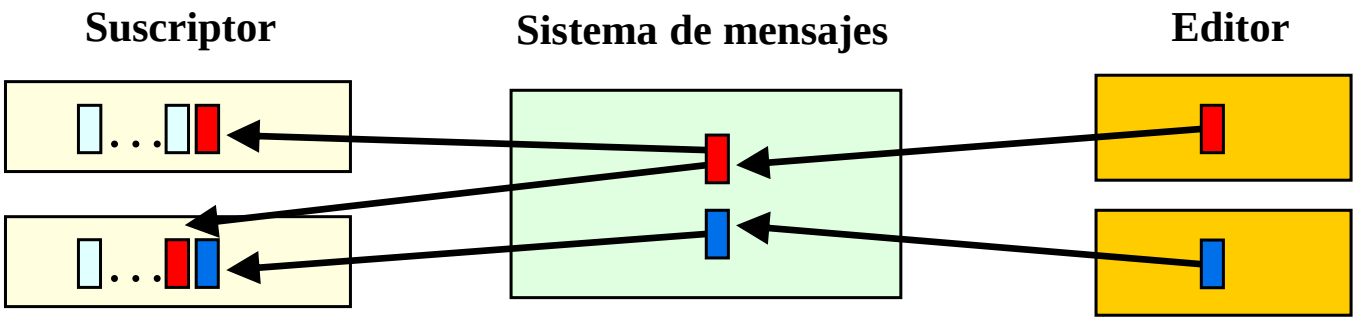


Paradigma de sistemas de colas de mensajes

- Clasificación:
  - Modelo de mensajes punto a punto

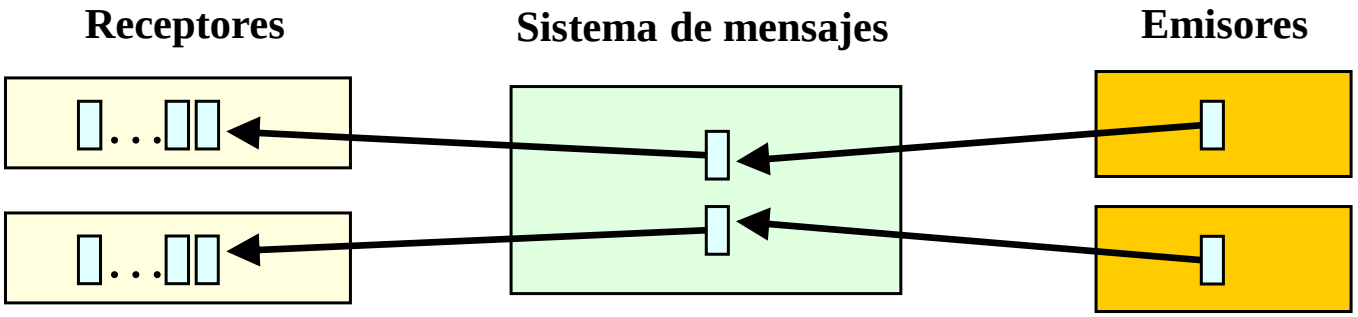


  - Modelo de mensajes publicación/suscripción

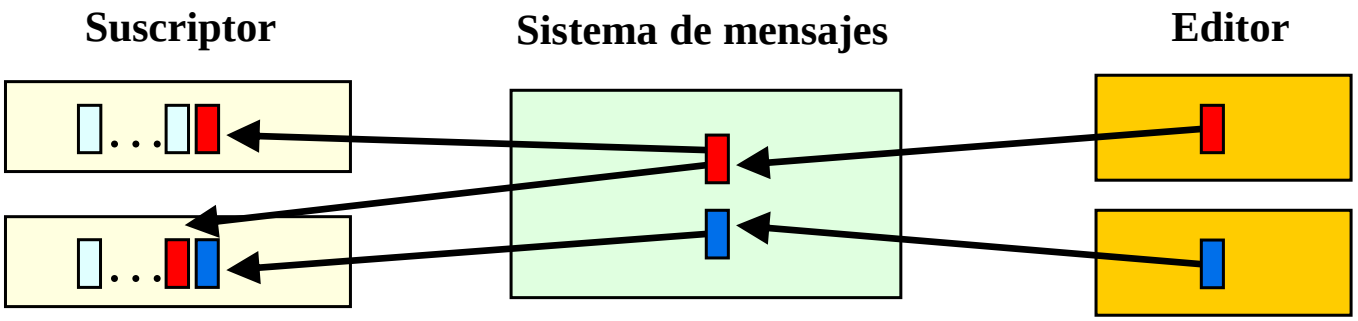


Paradigma de sistemas de colas de mensajes

- Clasificación:
  - Modelo de mensajes punto a punto



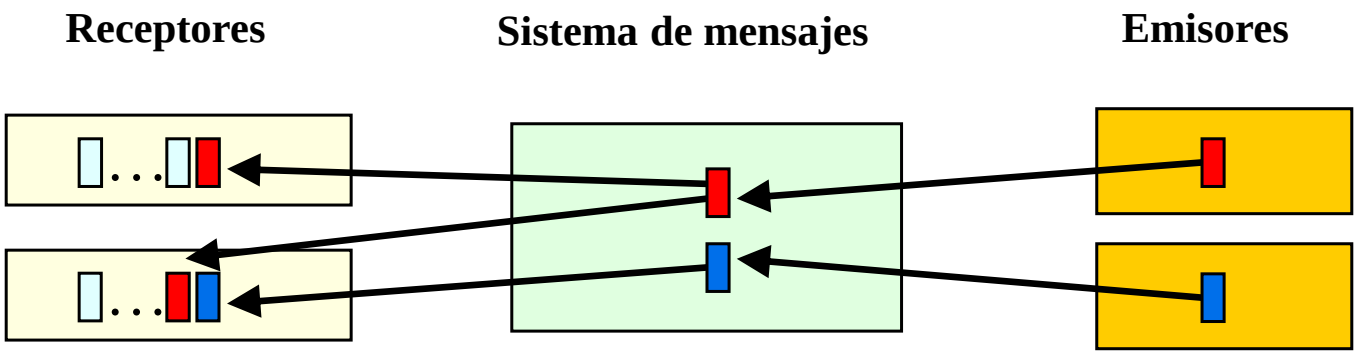
  - Modelo de mensajes publicación/suscripción



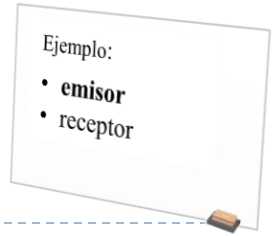
Paradigma de sistemas de colas de mensajes

- Modelo de mensajes publicación/suscripción:
  - *WebSphere MQ*
  - *IBM MQ*Series* (antes)
    - *http://www-01.ibm.com/software/integration/wmq/*
  - *Microsoft's Message Queue (MSQ)*
    - *http://msdn.microsoft.com/en-us/library/ms711472(VS.85).aspx*
  - *Java's Message Service*
    - *http://java.sun.com/products/jms/tutorial/*

Ejemplo con JMS

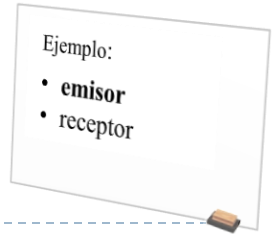


Ejemplo: emisor (1/4)




```
import javax.jms.*;
import javax.naming.*;

public class MessageSender
{

    public static void main(String[] args)
   {
        String                  queueName = null;
        Context                 jndiContext = null;
        QueueConnectionFactory  queueConnectionFactory  = null;
        QueueConnection         queueConnection = null;
        QueueSession            queueSession = null;
        Queue                   queue = null;
        QueueSender             queueSender = null;
        TextMessage             message = null;
        final int  NUM_MSGS;
```

Ejemplo: emisor (2/4)



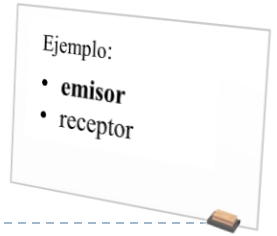

```
if ( (args.length != 1) ) {
    System.out.println("Usage: MessageSender ");
    System.exit(1);
}

queueName = new String(args[0]);
System.out.println("Queue name is " + queueName);

try {
        jndiContext = new InitialContext();
}  catch (NamingException e)
{
        System.out.println("Could not create JNDI " + "context: " + e.toString());
        System.exit(1);
}
```

Ejemplo: emisor (3/4)

Ejemplo:
- **emisor**
- receptor

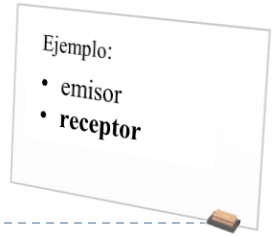
```
try
{
        queueConnectionFactory =  (QueueConnectionFactory)
                                  jndiContext.lookup("QueueConnectionFactory");
         queue = (Queue) jndiContext.lookup(queueName);
}
catch (NamingException e)
{
        System.out.println("JNDI lookup failed: " +  e.toString());
        System.exit(1);
}
```

Ejemplo: emisor (4/4)




```
    try {
            queueConnection = queueConnectionFactory.createQueueConnection();
            queueSession = queueConnection.createQueueSession(false,
                                                    Session.AUTO_ACKNOWLEDGE);
            queueSender = queueSession.createSender(queue);
            message = queueSession.createTextMessage();
            message.setText("Hello World!");
            System.out.println("Sending message: " +  message.getText());
            queueSender.send(message);
     } catch (JMSException e)  {
            System.out.println("Exception occurred: " +  e.toString());
     } finally {
            if (queueConnection != null) {
                try { queueConnection.close(); } catch (JMSException e) { }
            }
     }
    }
}
```

Ejemplo: receptor (1/4)



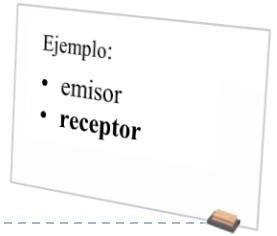

```
import javax.jms.*;
import javax.naming.*;

public class MessageReceiver
{
    public static void main(String[] args)
    {
        String                  queueName = null;
        Context                 jndiContext = null;
        QueueConnectionFactory  queueConnectionFactory  = null;
        QueueConnection         queueConnection = null;
        QueueSession            queueSession = null;
        Queue                   queue = null;
        QueueReceiver           queueReceiver = null;
        TextMessage             message = null;
```

Ejemplo: receptor (2/4)

Ejemplo:
- emisor
- **receptor**

```
if (args.length != 1)
{
    System.out.println("Usage: java " + "SimpleQueueReceiver <queue-name>");
    System.exit(1);
}
queueName = new String(args[0]);
System.out.println("Queue name is " + queueName)  ;

try
{
     jndiContext = new InitialContext();
}
catch (NamingException e)
{
    System.out.println("Could not create JNDI " + "context: " + e.toString());
    System.exit(1);
}
```

Ejemplo: receptor (3/4)




```
try
{
        queueConnectionFactory =  (QueueConnectionFactory)
                                        jndiContext.lookup("QueueConnectionFactory");
        queue = (Queue) jndiContext.lookup(queueName);
}
catch (NamingException e)
{
        System.out.println("JNDI lookup failed: " + e.toString());
        System.exit(1);
 }
```

Ejemplo: receptor (4/4)

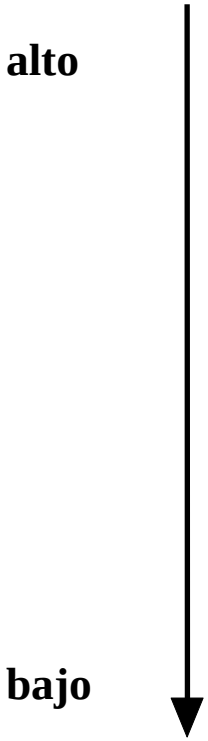
Ejemplo:
- emisor
- **receptor**

```
    try  {
       queueConnection =  queueConnectionFactory.createQueueConnection();
       queueSession =queueConnection.createQueueSession(false,
                                                  Session.AUTO_ACKNOWLEDGE);
       queueReceiver = queueSession.createReceiver(queue);
       queueConnection.start();

       Message m = queueReceiver.receive(1);
       message = (TextMessage) m;
       System.out.println("Reading message: " + message.getText());
    } catch (JMSException e) {
       System.out.println("Exception occurred: " +  e.toString());
    } finally {
        if (queueConnection != null) {
            try { queueConnection.close(); } catch (JMSException e) { }
        }
    }
}
```

Contenidos

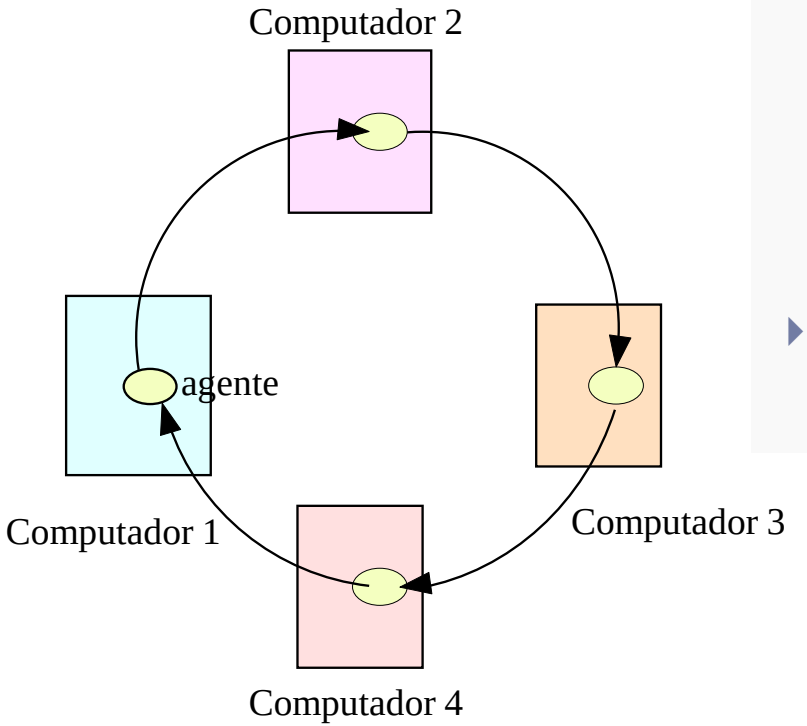
1. Sistema de colas de mensajes
2. **Agentes móviles**
3. Servicios de red
4. Espacio de objetos

Paradigma de agentes móviles



Espacio de objetos, aplicaciones colaborativas
Servicios de red, *object request broker*, agentes móviles
procedimientos remotos, métodos remotos
Cliente-servidor, *peer-to-peer*
Paso de mensajes

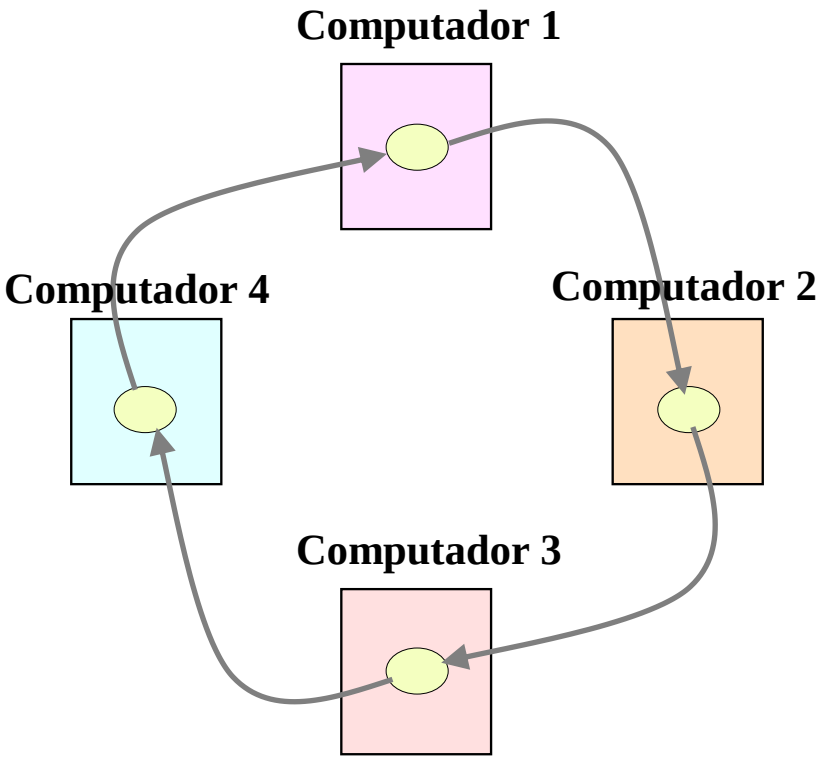
Paradigma de agentes móviles



- Agente móvil:
  programa u objeto transportable.
  - Un agente se lanza desde un ordenador
  - Viaja de forma automática de acuerdo con un itinerario
- Accede a los recursos o servicios de cada sistema que visita

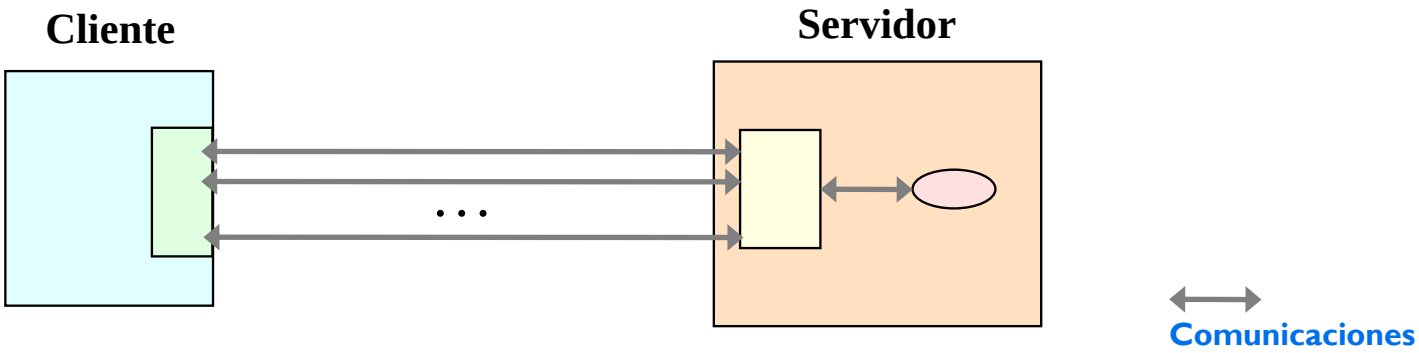
Paradigma de agentes móviles

- Agente: objeto serializable.
- Objeto activo *vs* agente móvil.
- Arquitectura:
  - Identidad.
  - Itinerario.
  - Datos de la entrega.
  - Código.

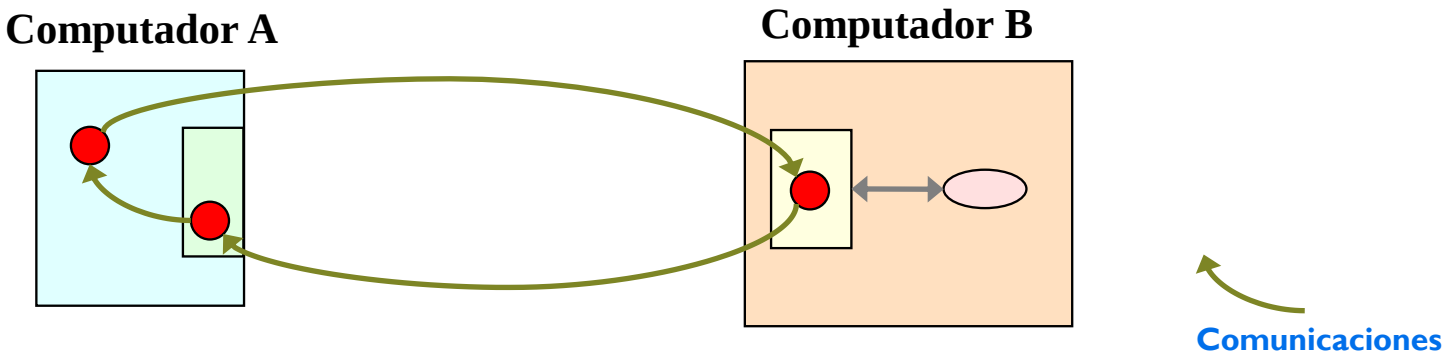


Cliente/servidor *vs* agentes móviles

- Paradigma cliente/servidor



- Paradigma de agentes móviles



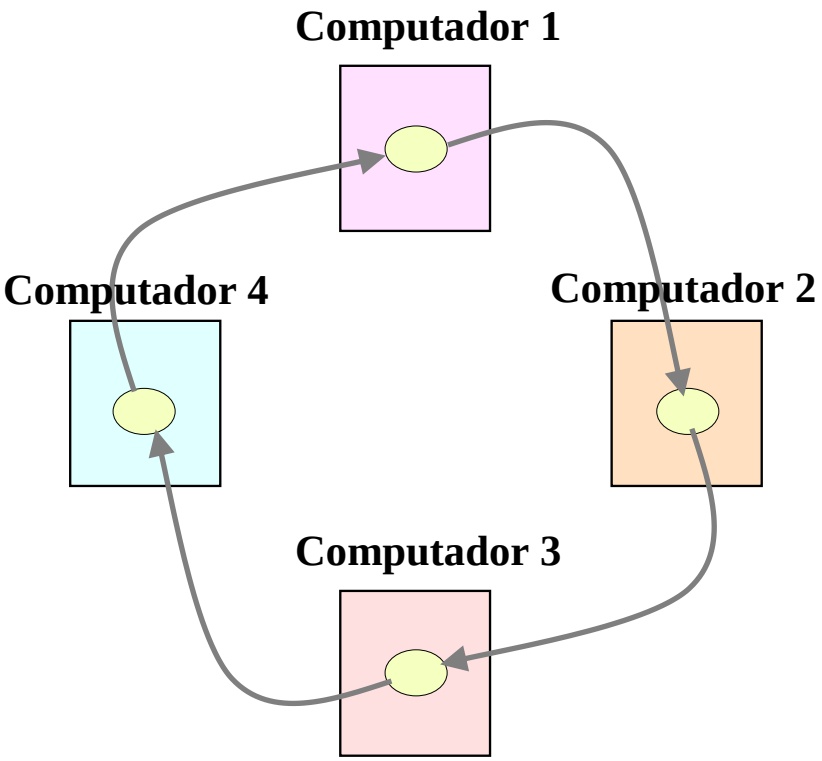
Paradigma de agentes móviles

- Ventajas en el empleo de agentes móviles:
  - Uso eficiente y económico de los canales de comunicación
  - Permite el uso de dispositivos portátiles de bajo coste
  - Permiten operaciones asíncronas y descentralizadas
- Riesgo de los agentes móviles: seguridad
- Técnicas de aumento del nivel de seguridad:
  - Autenticación
  - Acceso a recursos
  - Cifrado

Para más información…

- Mobile Agents Introductory
  http://www.docstoc.com/docs/13114805/Java-Mobile-Agents---Aglets
- The Mobile Agent List
  http://mole.informatik.uni-stuttgart.de/mal/mal.html
- Mobile Agents and the Future of the Internet
  http://www.cs.dartmouth.edu/~dfk/papers/kotz:future2/
- Mobile Agents
  http://cs.gmu.edu/~yhwang1/SWE622/Slides/MobileAgent.ppt

Ejemplo en Java



Ejemplo: interfaz para un agente


```
import java.io.Serializable;

public interface AgentInterface  extends Serializable
{
      void execute() ;
}
```

Ejemplo: agente (1/3)

Ejemplo:
- **agente**
- servidor
- cliente

```
import java.io.*;
import java.util.*;
import java.rmi.*;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;

public class Agent implements AgentInterface
{
        int  hostIndex;
        String name;
        Vector hostList;
        int RMIPort = 12345;

        public Agent (String myName,Vector theHostList,  int theRMIPort )  {
            name = myName;
            hostList = theHostList;
            hostIndex = 0;
            RMIPort = theRMIPort;
        }
```

Ejemplo: agente (2/3)

Ejemplo:
- **agente**
- servidor
- cliente

```
public void execute()  {
      String thisHost, nextHost;
      sleep (2);
      System.out.println("007 here!");
      thisHost = (String) hostList.elementAt(hostIndex);
      hostIndex++;
      if (hostIndex < hostList.size()) {
         nextHost = (String) hostList.elementAt(hostIndex);
         sleep (5);
         try {
                Registry registry = LocateRegistry.getRegistry ("localhost", RMIPort);
                ServerInterface h = (ServerInterface)  registry.lookup(nextHost);
                System.out.println("Lookup for " + nextHost + " at " + thisHost + " completed " );
                sleep (5);   // delay for visibility
                h.receive(this);
         }  catch (Exception e)  {
                System.out.println ("Exception in Agent execute: " + e);
         }
```

Ejemplo: agente (3/3)

Ejemplo:
- **agente**
- servidor
- cliente

```
        } else {
            sleep (5);
            System.out.println("Agent 007 has come home");
            sleep (5);
        }
    } // execute


    static void sleep (double time )  {
        try {
            Thread.sleep( (long) (time * 1000.0));
        } catch (InterruptedException e)  {
            System.out.println ("sleep exception");
        }
    }

}
```

Ejemplo: interfaz para el servidor

Ejemplo:
- agente
- **servidor**
- cliente

```
import java.rmi.*;

public interface ServerInterface   extends Remote
{
      public void receive(Agent h)
                  throws java.rmi.RemoteException;
}
```

Ejemplo: servidor (1/2)

Ejemplo:
- agente
- **servidor**
- cliente

```
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.net.*;
import java.io.*;

public class Server extends UnicastRemoteObject  implements ServerInterface
{
      static int RMIPort = 12345;

      public Server() throws RemoteException {
          super();
      }

      public void receive (Agent h) throws RemoteException {
          sleep (3) ;
          System.out.println ("*****Agent" + h.name + " arrived." ) ;
          h.execute();
      }
```

Ejemplo: servidor (2/2)

Ejemplo:
- agente
- **servidor**
- cliente

```
public static void main(String args[])  {
  InputStreamReader is = new InputStreamReader(System.in);
  BufferedReader br = new BufferedReader(is);
  String s;
  String myName = "server" + args[0];
  try {
     System.setSecurityManager(new RMISecurityManager());
     Server h = new Server();
     Registry registry =  LocateRegistry.getRegistry(RMIPort);
     registry.rebind( myName, h);
     System.out.println("**********************************");
     System.out.println("     Agent " + myName + " ready.");
     System.out.println("**********************************");
   } catch (RemoteException re) {
         System.out.println("Exception in AgentServer.main:“ + re);
   }
}

static void sleep (double time ) {
      try                                    { Thread.sleep( (long) (time * 1000.0)); }
      catch (InterruptedException e) { System.out.println ("sleep exception"); }
}
}
```

Ejemplo: cliente (1/2)


```
import java.io.*;
import java.util.*;
import java.rmi.*;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;

public class Client
{
    static int RMIPort = 12345;

    public static void main (String args[])
    {
        System.setSecurityManager(new RMISecurityManager());
        try {
         ...
```

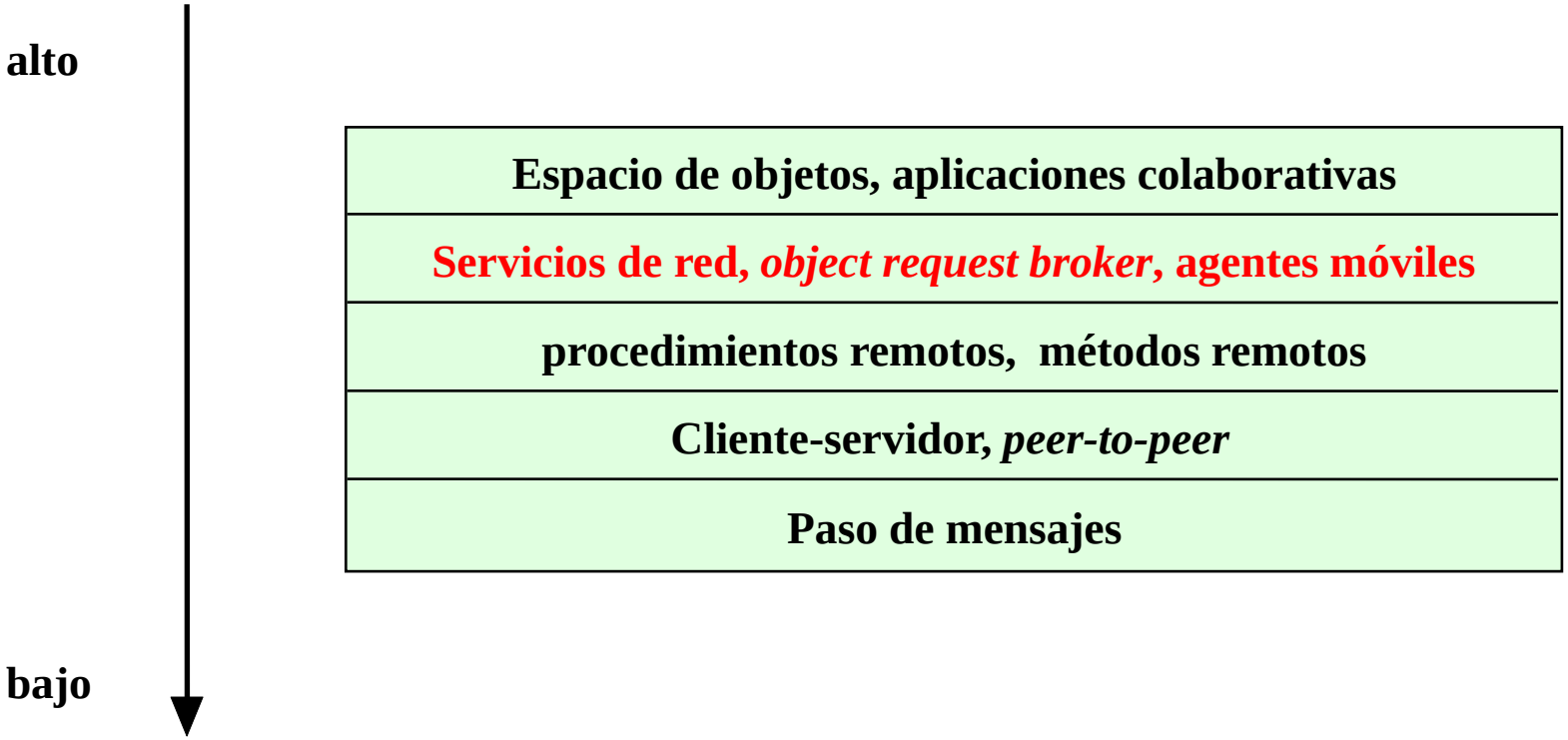
Ejemplo: cliente (2/2)


```
try {
        Registry registry = LocateRegistry.getRegistry ("localhost",  RMIPort);
        ServerInterface h = (ServerInterface)  registry.lookup("server1");
        System.out.println(" Lookup for server1  completed " );
        System.out.println(" ***Have a good trip, " + " agent 007.");
       Vector hostList = new Vector();
        hostList.addElement("server1");
        hostList.addElement("server2");
        hostList.addElement("server3");
       Agent a = new Agent("007", hostList, RMIPort);
        h.receive(a);
        System.out.println("***Nice job, agent 007");

    } catch (Exception e)  {
       System.out.println("Exception in main: " + e);
    }
 } // main

}
```

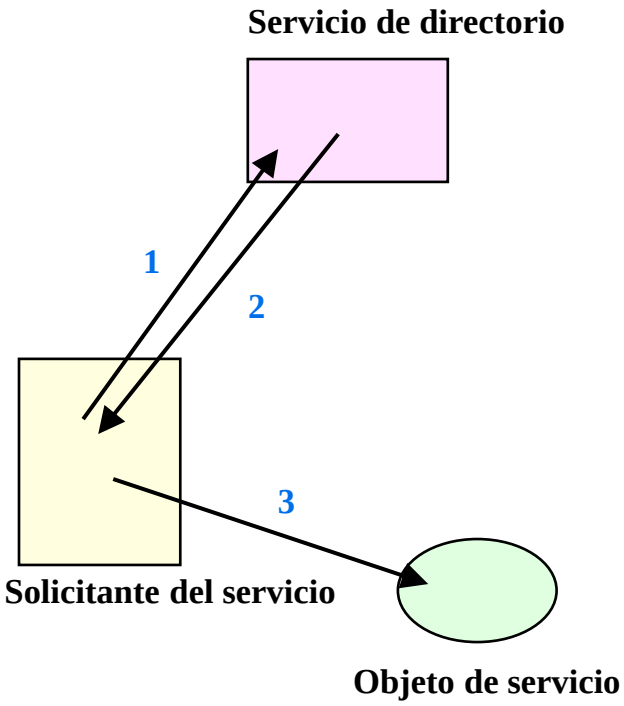
Contenidos

1. Sistema de colas de mensajes
2. Agentes móviles
3. **Servicios de red**
4. Espacio de objetos

Paradigma de servicios de red



Paradigma de servicios de red

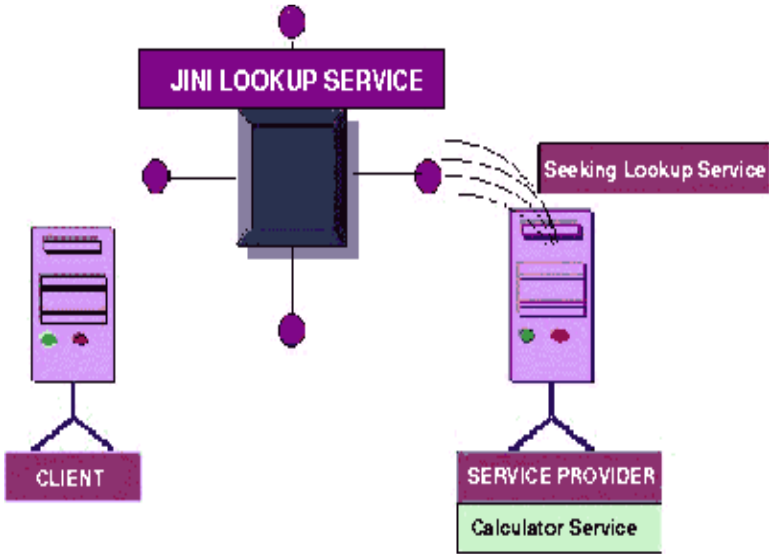


- Servicio de directorio: proporcionan la referencia a los servicios disponibles
- Pasos:
  1. El proceso solicitante contacta con el servicio de directorio
  2. El servicio de directorio devuelve la referencia al servicio solicitado
  3. Usando la referencia, el proceso solicitante interactúa con el servicio

Paradigma de servicios de red

- Red se presenta como una infraestructura con servicios
- Cliente accede a servicios:
  - Detección
  - Localización
- Servicios se añaden y retiran de forma dinámica
- Paradigma no centralizado

Servicios con Jini



- Servicio es objeto que implementa una API de servicios.
- Cliente buscar servicios que soportan una API.
- El objeto descubierto es recibido por el cliente.
- En cliente y el servicio únicamente deben coincidir en el API.
- Basado en Java RMI.

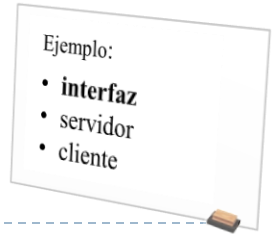
Para más información…

- Jini(tm) Network Technology
  http://www.sun.com/jini/
- Jan Newmarch's Guide to JINI Technologies
  http://jan.netcomp.monash.edu.au/java/jini/tutorial/Jini.xml
- Jini Planet
  http://www.kedwards.com/jini/
- Directory of Jini™ Resources on the net: Tutorials_and_Examples
  http://litefaden.com/sv/jd/Tutorials_and_Examples/

Para más información…

- Jini whitepaper
  http://wwwwswest.sun.com/jini/whitepapers/jini-execoverview.pdf
- California Software Lab | Jini by Example – Whitepaper
  http://www.cswl.com/whiteppr/tutorials/jini.html
- Noel Enete's Nuggets for Jini
  http://www.enete.com/download/index.html#_nuggets_
- Jini.org -- The Community Resource for Jini(tm) Technology
  http://www.jini.org/

Ejemplo:
interfaz de un servicio de red




```
import java.rmi.*;

public interface HelloServerInterface  extends Remote
{
    public String sayHello() throws RemoteException;
}
```

Ejemplo: servidor del servicio de red (1/3)


```
import net.jini.core.entry.*;
import net.jini.core.lookup.*;
import net.jini.core.discovery.*;
import net.jini.lookup.entry.*;
import com.sun.jini.lookup.*;
import com.sun.jini.lease.*;
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;

  public class HelloServer  extends UnicastRemoteObject
                                implements HelloServerInterface, ServiceIDListener
  {

      public HelloServer () throws RemoteException {
          super();
      }
```

Ejemplo:
servidor del servicio de red (2/3)

Ejemplo:
- agente
- **servidor**
- cliente

```
public String sayHello () throws RemoteException
{
      return ("Hello World!");
}


public void serviceIDNotify (ServiceID id)
{
      System.out.println (" ServiceID received: " + id);
}
```

Ejemplo: servidor del servicio de red (3/3)

Ejemplo:
- agente
- **servidor**
- cliente

```
public static void main (String[] args) {
    HelloServer server;
    Entry[] aeAttributes;
    JoinManager manager;

    try {
        System.setSecurityManager (new RMISecurityManager ());

        attributes = new Entry[1];
        attributes[0] = new Name("HelloServer");
        server = new HelloServer();
        manager = new JoinManager (server, attributes,  server,
                                   new LeaseRenewalManager() );
    } catch (Exception ex)  {
        ex.printStackTrace( );
    }
 }
}
```

Ejemplo: cliente del servicio de red (1/2)


```
import net.jini.core.entry.*;
import net.jini.core.lookup.*;
import net.jini.core.discovery.*;
import net.jini.lookup.entry.*;
import com.sun.jini.lookup.*;
import java.rmi.*;

  class HelloClient
  {
    public static void main (String[] args)
    {
        Entry[] attributes;
        LookupLocator lookup;
        ServiceID id;
        ServiceRegistrar registrar;
        ServiceTemplate template;
        HelloServerInterface helloServer;
```

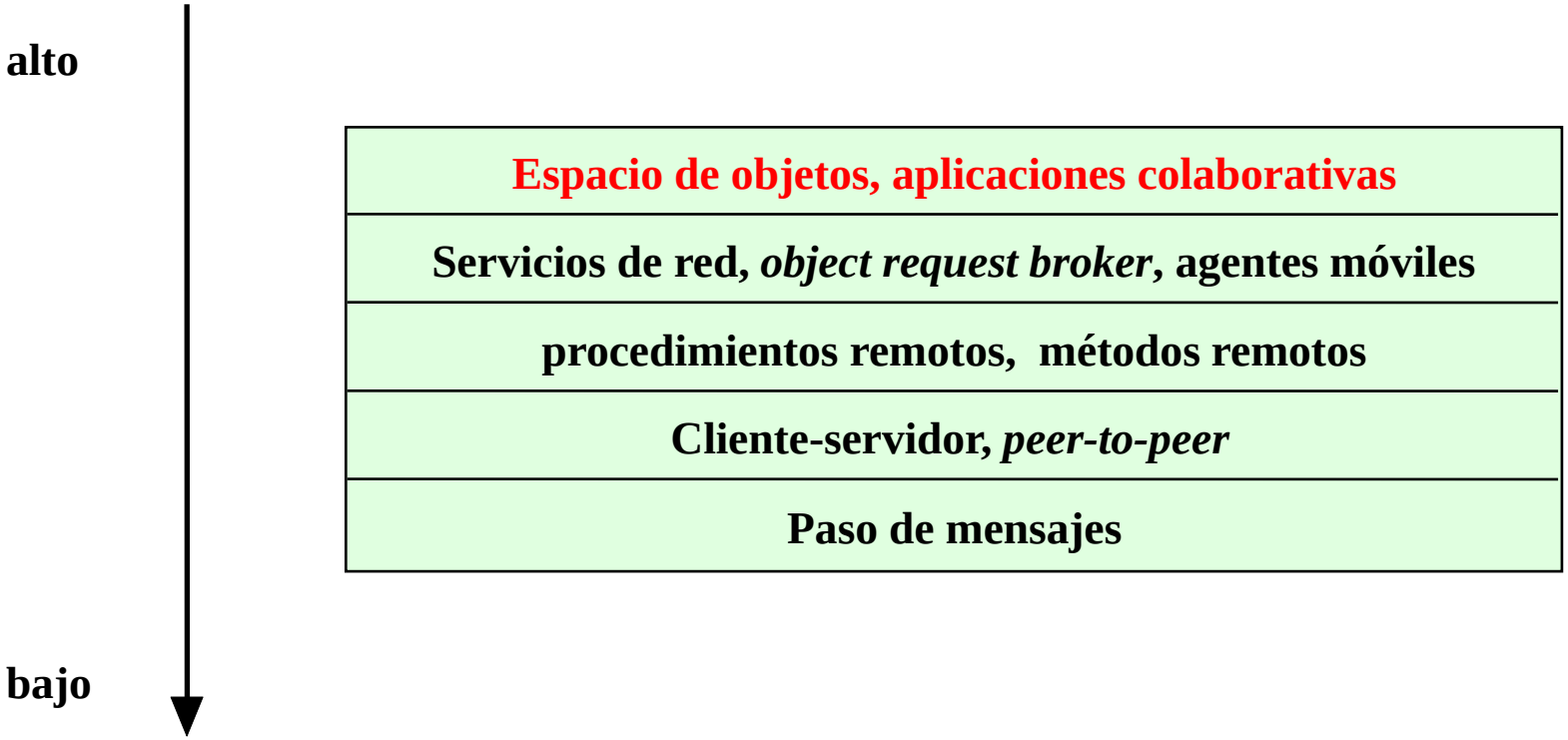
Ejemplo: cliente del servicio de red (2/2)


```
    try {
        System.setSecurityManager(new RMISecurityManager ());
        lookup = new LookupLocator("jini://localhost");
        registrar  = lookup.getRegistrar();

        attributes = new Entry[1];
        attributes[0] = new Name ("HelloServer");
        template = new ServiceTemplate (null, null, attributes);
        helloServer = (HelloServerInterface)  registrar.lookup(template);
        System.out.println(helloServer.sayHello());
    }  catch (Exception ex)  {
        ex.printStaceTrace( );
    }
 } // main

}
```

Contenidos

1. Sistema de colas de mensajes
2. Agentes móviles
3. Servicios de red
4. **Espacio de objetos**

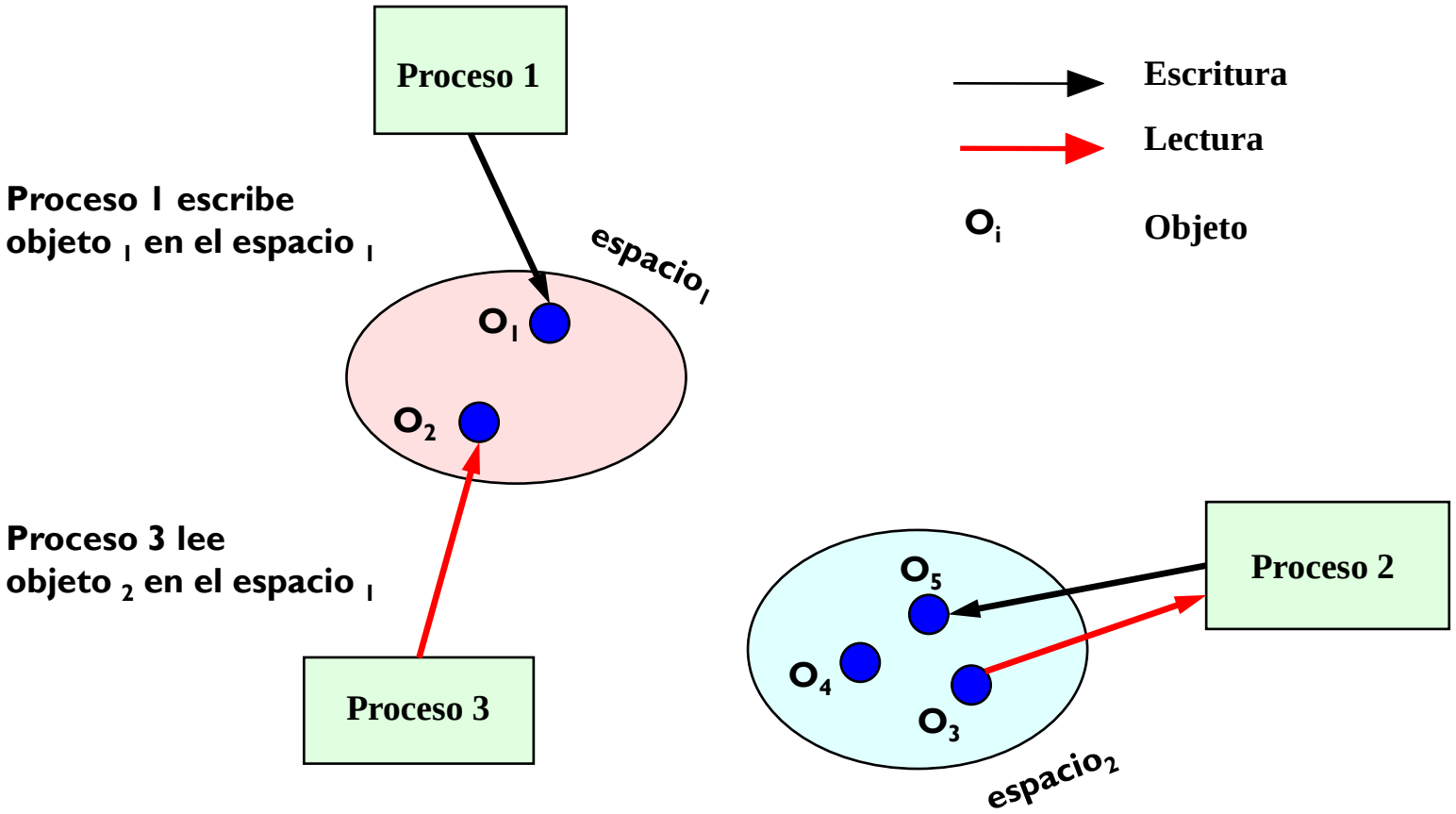
Paradigmas de
Espacio de objetos y aplicaciones colaborativas



Paradigma de espacio de objetos

- Base del sistema de pizarras
- Espacio de objeto: repositorio de objetos
- Los procesos se coordinan intercambiando objetos
- Los objetos no modifican los objetos en el espacio ni invocan los métodos directamente
- Operaciones de extracción y reinserción del objeto

Paradigma de espacio de objetos



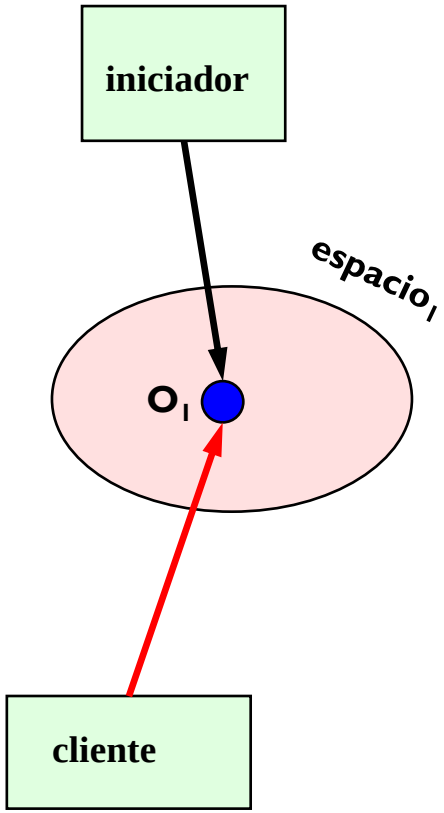
Paradigma de espacio de objetos

- JavaSpace:
  - Basado en Jini
  - Simple y potente:
    "*Using a small set of operations, we can build a large class of distributed applications without writing a lot of code*"
  - Da soporte a protocolos estrechamente relacionados
  - Simplifica el diseño del servidor

Para más información…

- The Nuts and Bolts of Compiling and Running JavaSpaces Programs: http://www.jiniworld.net/document/javaspace/The Nuts and Bolts of Compiling and Running JavaSpaces(TM).html
- JavaSpaces(TM) Principles, Patterns, and Practice: http://www.cswl.co.uk/whiteppr/tutorials/jini.html
- O'Reilly Network: First Contact: Is There Life in JavaSpace?

Ejemplo de espacio de objetos

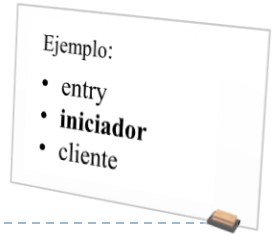


Ejemplo: espacio de objetos

Ejemplo:
- **entry**
- iniciador
- cliente

```
import net.jini.core.entry.Entry;

public class SpaceObject implements Entry
{
    public String message;
    public Integer counter;

    public SpaceObject( ) {    }

    public SpaceObject(String message) {
        this.message = message;
        counter = 0;
    }

    public String toString( )  {  return content + " read " + counter + " times.“;}

    public void increment( ) {  counter = new Integer(counter.intValue() + 1);  }
}
```

Ejemplo: espacio de objetos




```
import net.jini.core.lease.Lease;
import net.jini.space.JavaSpace;

public class HelloWorld  {
   public static void main(String[] args)  {
      try {
         SpaceObject msg = new SpaceObject("Hello World!");

         JavaSpace space;
         space = (JavaSpace)space();
         space.write(msg, null, Lease.FOREVER);

         SpaceObject template = new SpaceObject();
         while (true) {
            SpaceObject result = (SpaceObject) space.read(template, null, Long.MAX_VALUE);
            System.out.println(result);
            Thread.sleep(1000);
         }
      } catch (Exception ex) { ex.printStackTrace(); }
    }
}
```

Ejemplo: espacio de objetos

Ejemplo:
- entry
- iniciador
- **cliente**

```
import net.jini.core.lease.Lease;
import net.jini.space.JavaSpace;

public class HelloWorldClient
{
   public static void main(String[] args)
   {
        try {
          JavaSpace space = (JavaSpace)space();
          SpaceObject template = new SpaceObject();
          while (true)  {
            SpaceObject result = (SpaceObject) space.take(template, null, Long.MAX_VALUE);
            result.increment();
            space.write(result, null, Lease.FOREVER);
            Thread.sleep(1000);
          }
        } catch (Exception ex) { ex.printStackTrace();}
   }
}
```

Paradigmas avanzados de computación distribuida

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid