Tema 5:

Servicios de red

Servicios de red: Paradigmas de la computación distribuida

- Niveles de abstracción de los diferentes paradigmas

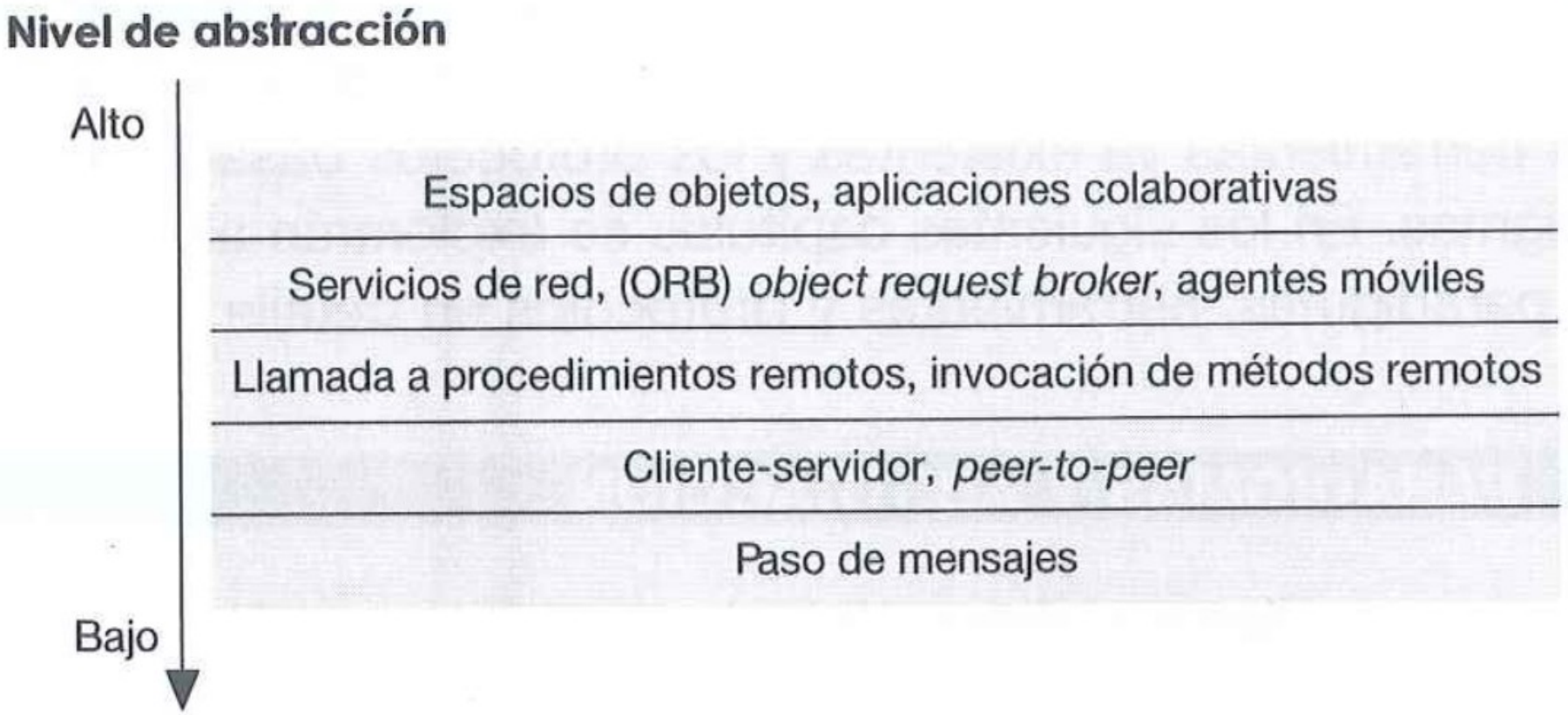


Figura 3.1. Los paradigmas de computación distribuida y su nivel de abstracción.

Servicios de red: Paradigmas de la computación distribuida

- Paso de mensajes
  - Los datos que representan mensajes se intercambian entre dos procesos un ***emisor*** y un ***receptor***

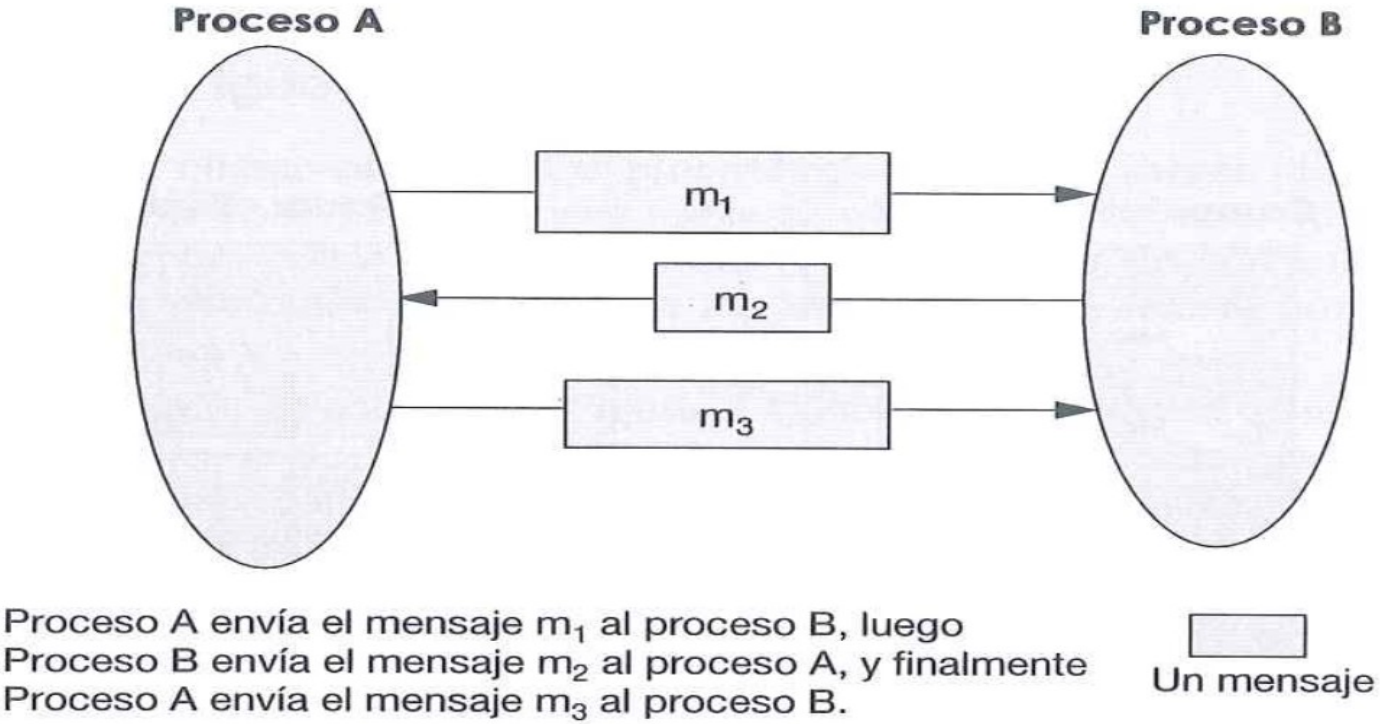


Figura 3.2. El paradigma de paso de mensajes.

  - Los procesos conectados realizan la entrada y salida de los mensajes de forma similar a la escritura y lectura de ficheros
  - La interfaz de programación de aplicaciones ***sockets*** se basa en este paradigma

Servicios de red: Paradigmas de la computación distribuida

- Cliente-servidor
  - El servidor es el proveedor de servicios y el cliente hace peticiones de los servicios.

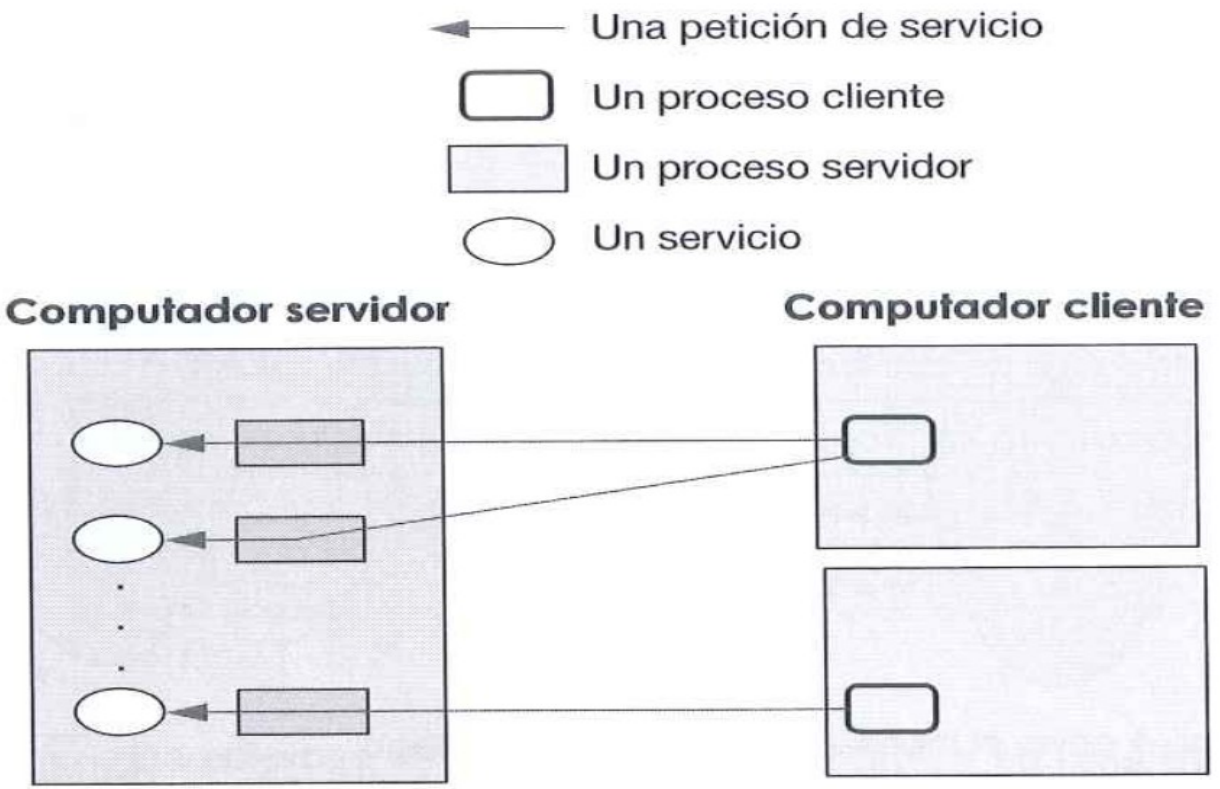


Figura 3.3. El paradigma cliente-servidor.

Servicios de red: Paradigmas de la computación distribuida

- Cliente-servidor
  - Proporciona una abstracción eficiente para proporcionar servicios de red
  - Ampliamente utilizado en la implementación de protocolos que proporcionan diferentes servicios de red como:
    - HTTP, FTP, DNS, NFS, AFS, CODA, etc
  - El paradigma cliente servidor es inherente a muchas aplicaciones distribuidas
    - El API de sockets orientados a conexión proporciona operaciones diseñadas específicamente para clientes y servidores
    - El API de ***llamadas a procedimientos remotos (Remote Procedure Call, RPC)*** y el API de ***Java de invocación de métodos remotos (Remote Method Invocation, RMI)*** se refieren a los procesos participantes como clientes y servidores

Servicios de red: Paradigmas de la computación distribuida

- De igual a igual o *peer-to-peer*
  - Los procesos participantes interpretan los mismos papeles, con idénticas capacidades y responsabilidades.

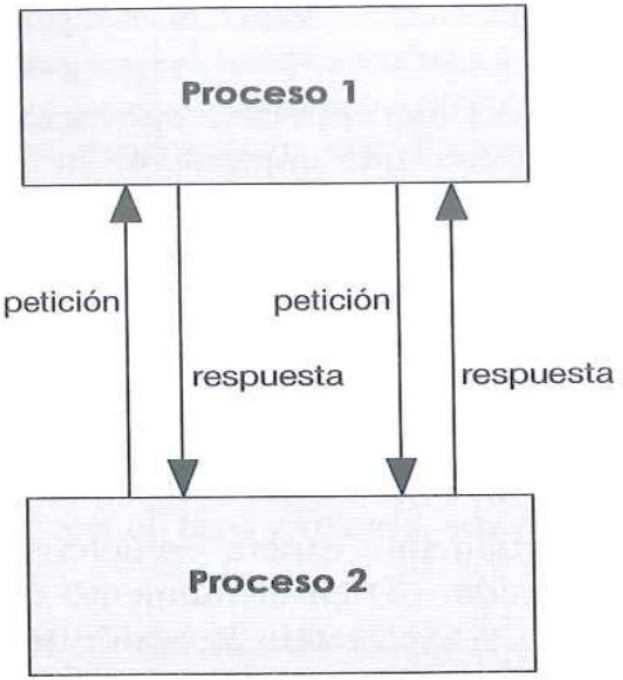


Figura 3.4. El Paradigma *peer-to-peer.*

  - Resulta muy apropiado para
    - `mensajería instantánea tipo chat`
    - `transferencia de ficheros grandes como vídeos, películas`
    - `videoconferencias`
    - `trabajo colaborativo`

Servicios de red: Paradigmas de la computación distribuida

- Modelo de llamadas a procedimientos remotos (Remote Procedure Call, RPC)
  - La comunicación entre dos procesos se realiza utilizando llamadas a procedimientos remotos
    - de forma similar a como un programador realiza la llamada a un procedimiento o función local que forma parte de un programa

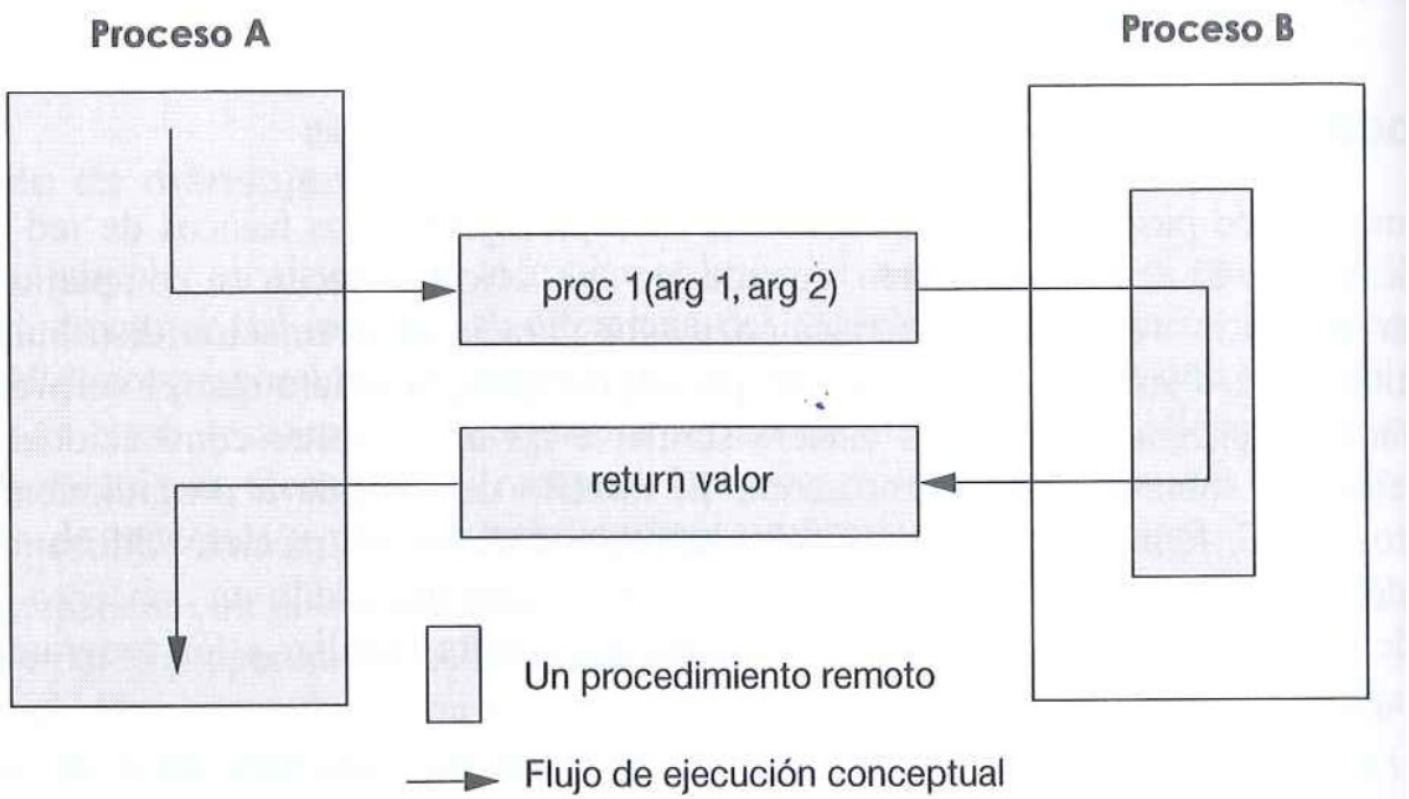


Figura 3.7. El paradigma de llamada a procedimientos remotos.

Servicios de red: Paradigmas de la computación distribuida

- Paradigmas basados en objetos distribuidos
  - Las aplicaciones acceden a ***objetos distribuidos*** sobre una red
  - Los objetos proporcionan ***métodos***, a través de cuya invocación una aplicación obtiene acceso a los servicios.
  - Existen varios paradigmas dentro de esta categoría
  - Paradigma basado en invocación a métodos remotos (***Remote Method Invocation, RMI***)
    - RMI es el equivalente de RPC en programación orientada a objetos.
      - Un proceso invoca métodos de un objeto que reside en un ordenador remoto

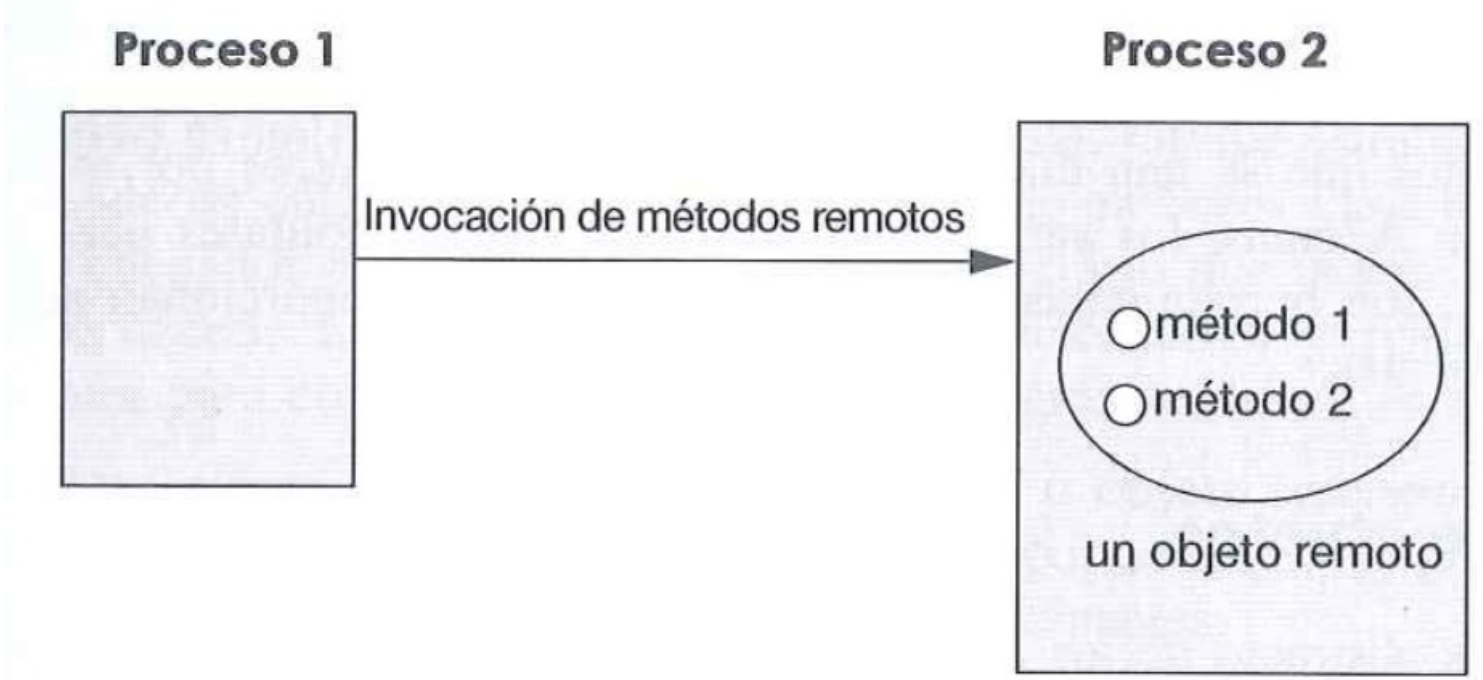


Figura 3.8. El paradigma de invocación de métodos remotos.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma basados en objetos distribuidos
  - Paradigma basado en ***Object Request Broker (ORB)***
    - Un proceso solicita una petición a un ORB (object Request Broker), que redirige la petición al objeto apropiado que proporciona dicho servicio

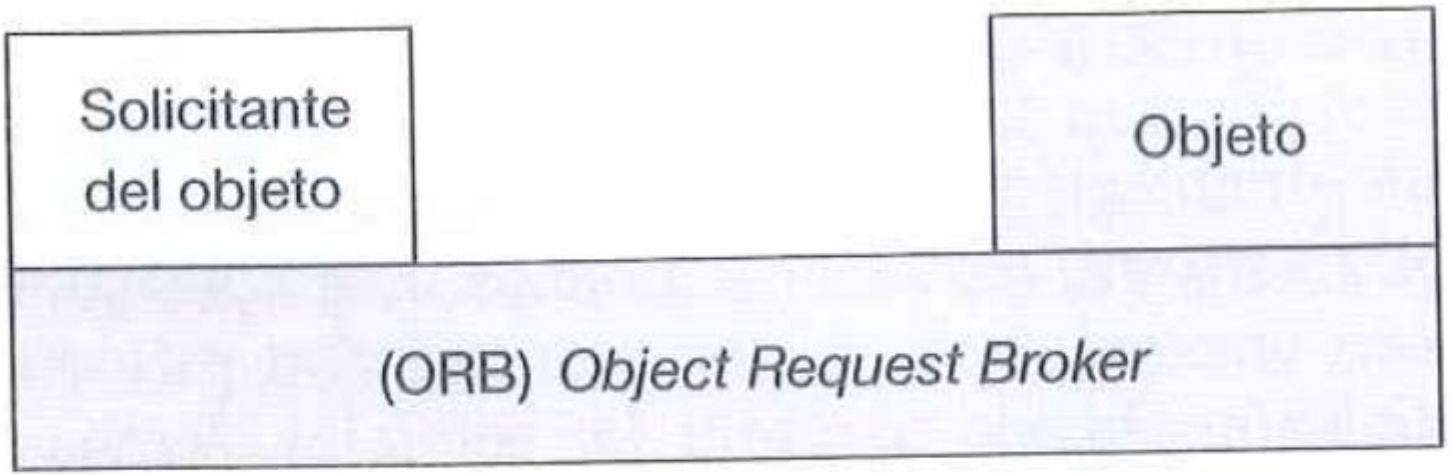


Figura 3.9. El paradigma basado en *Object Request Broker*.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma basados en objetos distribuidos
  - Paradigma basado en ***Object Request Broker (ORB)***
    - A diferencia de RMI, ORB funciona como un ***middleware***
      - Permite a una aplicación como solicitante de un objeto, acceder potencialmente a varios objetos remotos (o locales).
      - Funciona como mediador, permitiendo la interacción entre
        - Objetos implementados con diferentes API y/o
        - Objetos ejecutándose en diferentes plataformas
    - Este paradigma es la base de CORBA (Common Object Request Broker Architecture) de OMG (Object Management Group)
    - Herramientas basadas en CORBA son:
      - Visibroker de Inspire
      - Java IDL (Java Interface Definition Language)
      - Orbix de IONA
      - TAO de Object Computing
  - Otros paradigmas
    - Las ***las tecnologías basadas en componentes***, como:
      - Microsoft COM, Microsoft DCOM, Java Beans y Enterprise Java Beans
    - Los ***servidores de aplicaciones*** para aplicaciones empresariales se comportan como *middleware* que proporcionan accesos a objetos o componentes

- Paradigma de espacio de objetos
  - Los participantes en una aplicación convergen en un ***espacio de objetos***
    - `Los suministradores aportan objetos a este espacio`
    - `Los solicitantes que se suscriben al espacio de objetos puedne acceder a los objetos`
  - Un objeto en el espacio pude ser usado por un sólo participante a la vez
    - `Hay que garantizar la exclusión mutua`

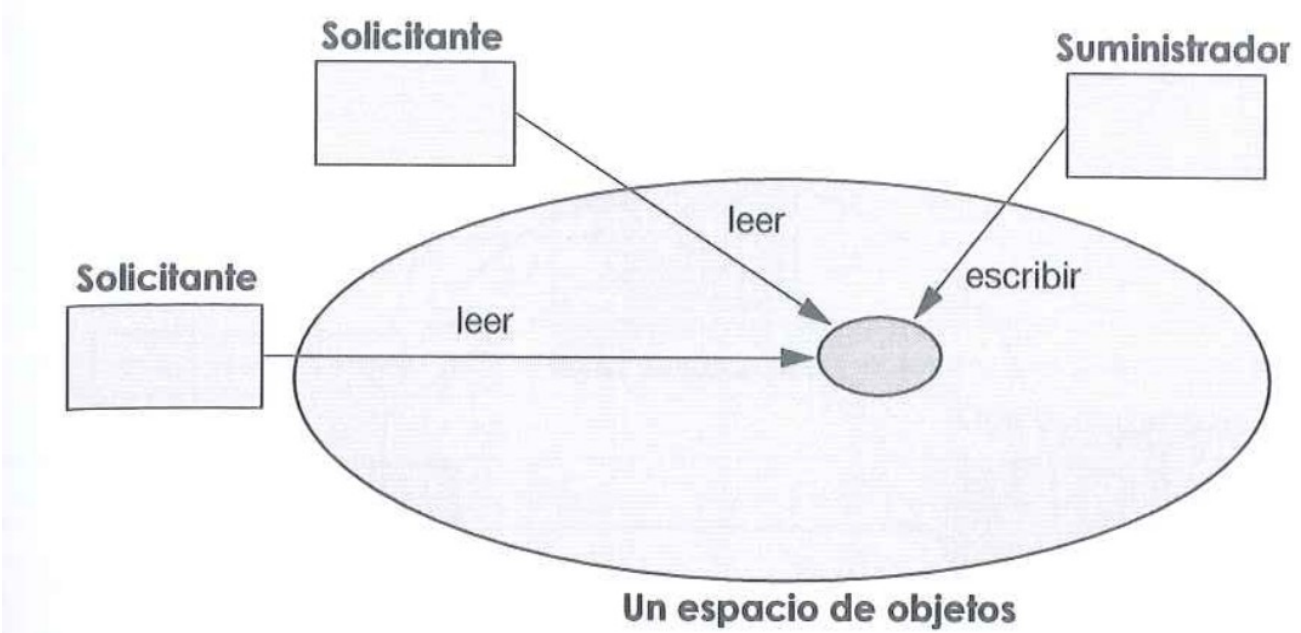


Figura 3.10. El paradigma de Espacio de Objetos.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma de agentes móviles
  - Un agente móvil es un ***programa o objeto transportable***
  - Filosofía
    - Un agente se lanza de un ordenador
    - Pasa por un itinerario fijado de ordenadores
    - En cada ordenador accede a los recursos o servicios necesarios y realiza las tareas necesarias para completar su misión

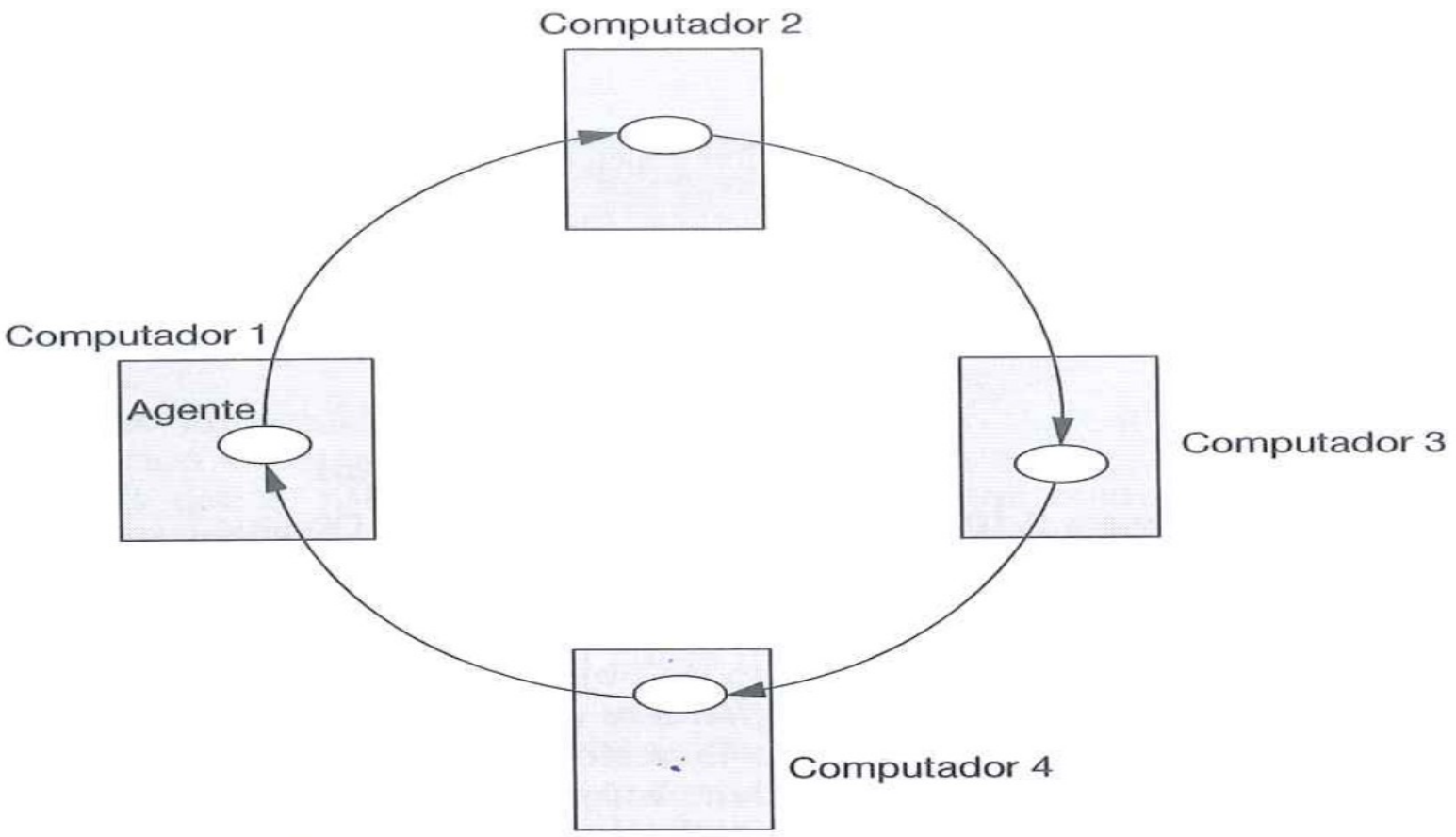


Figura 3.11. Paradigma de agentes móviles.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma de servicio de red
  - Los proveedores de servicios se registran en los servidores de directorio de una red.
  - Un proceso que desee un servicio particular *contacta con el servidor de directorio* en tiempo de ejecución, y si el servicio está disponible, al proceso *se le da la referencia* a dicho proceso
  - Usando esta referencia el proceso interactúa con el servicio

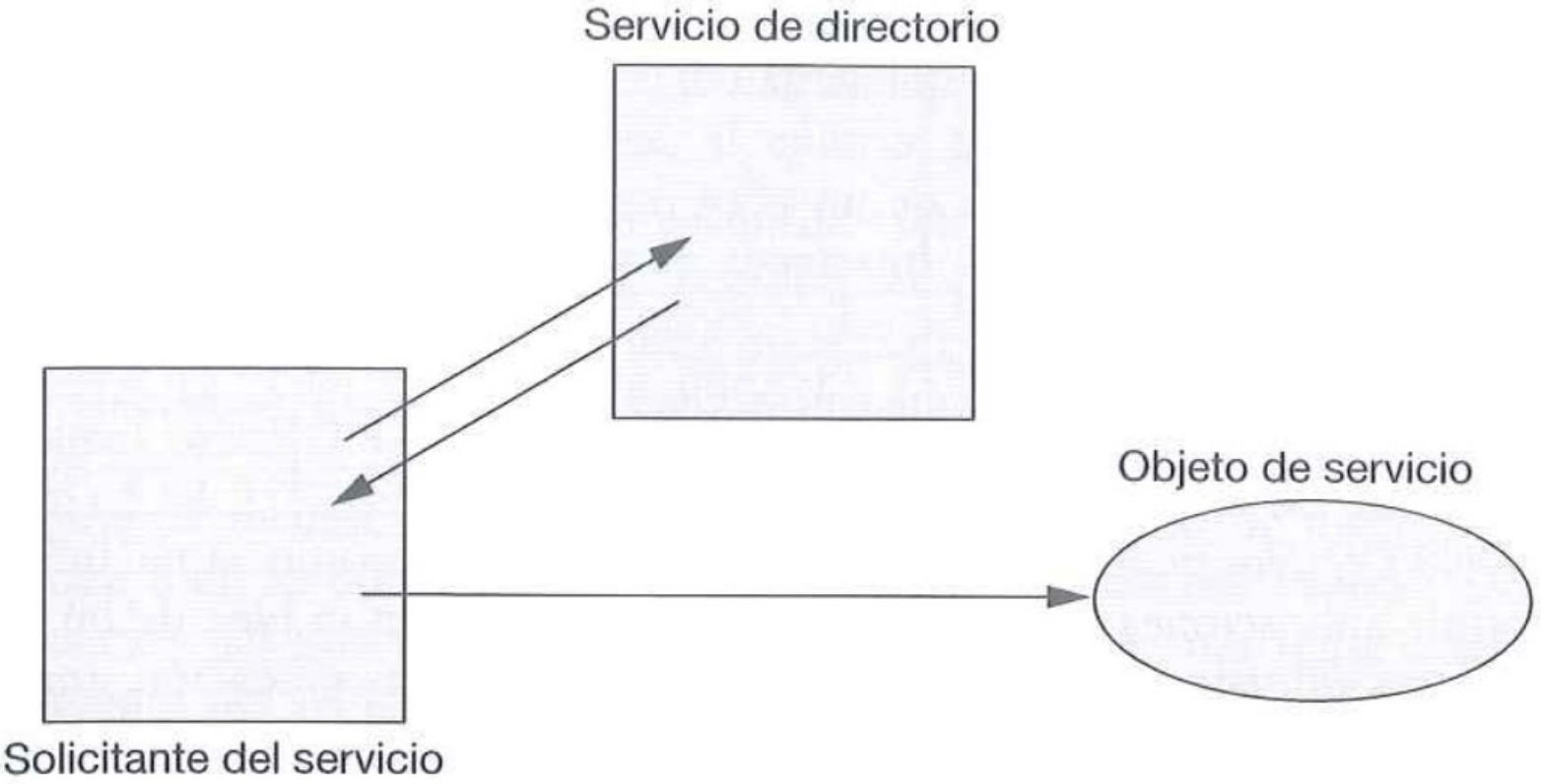


Figura 3.12. El paradigma de servicios de red.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma de servicio de red
  - El paradigma es esencialmente una extensión del paradigma de invocación de métodos remótos RMI
    - La diferencia es que los servicios se registran en un directorio global,
    - Así pueden ser localizados y accedidos por solicitantes de servicios de una red federada
  - Idealmente, los servicios pueden registrarse y ser accedidos con un ***identificador único global***
    - Ofrece al paradigma un nivel de abstracción extra: ***transparencia de localización***
      - De esta manera no se tiene que conocer la localización del objeto accedido
  - La tecnología Jine de Java se basa en este paradigama
  - El protocolo SOAP (Simple Object Access Protocol) aplica este paradigma a los servicios accesibles en la web.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma de aplicaciones colaborativas (groupware)
  - Es un modelo para trabajo colaborativo
  - Los procesos participan en grupo en una sesión colaborativa
  - Cada proceso puede hacer aportaciones a todo o parte del grupo
    - `Usando multidifusión para enviar datos`
    - `Usando pizarras o tablones de anuncios`

    lo que permite a cada participante leer y escribir datos sobre una visualización compartida
  - Los dos tipos de paradigma son:

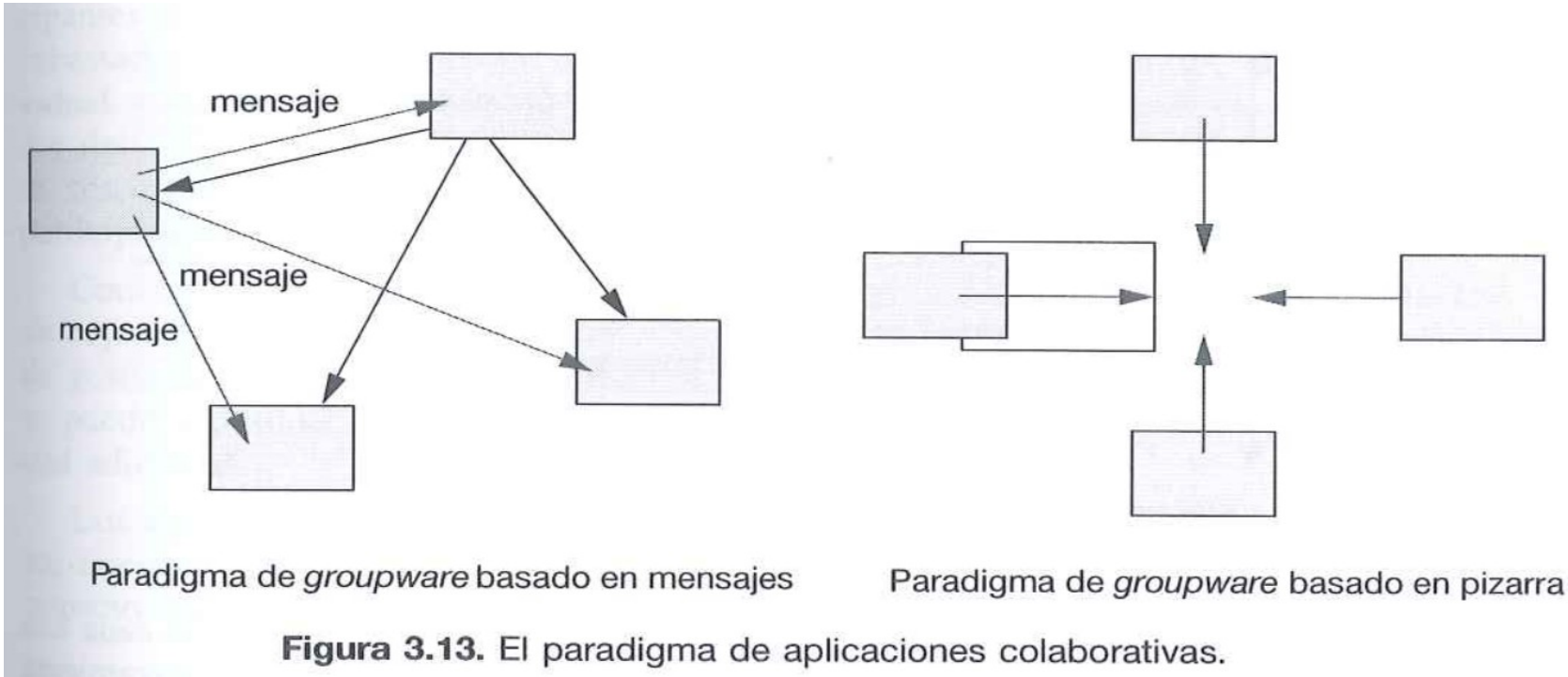


Figura 3.13. El paradigma de aplicaciones colaborativas.

Servicios de red: Paradigmas de la computación distribuida

- Paradigma de aplicaciones colaborativas (groupware)
  - En el paradigma de groupware basado en mensajes se basan
    - Programas groupware como Lotus QuickPlace
    - La API de Java de multidifusión y el Java Shared Data Toolkit (JSDT) vía intercambio de mensajes
  - En el paradigma de groupware basado en pizarra se basan
    - SMART Board
    - NetMeeting
    - Groove

Servicios de red: Servicios web

- Los servicios web proporcionan servicios de red transportados por HTTP
- Están siendo propuestos como una nueva forma de construir aplicaciones de red desde componentes distribuidos (servicios) independientes del lenguaje y de la plataforma
- Los protocolos y las API de la tecnología están evolucionando todavía
- Nos centraremos en:
  - El protocolo ***protocolo simple de accesos (Simple Object Access Protocol, SOAP)***
  - La ***API SOAP de Apache***
- Un servicio web se proporciona por un objeto servidor y se accede por un cliente.

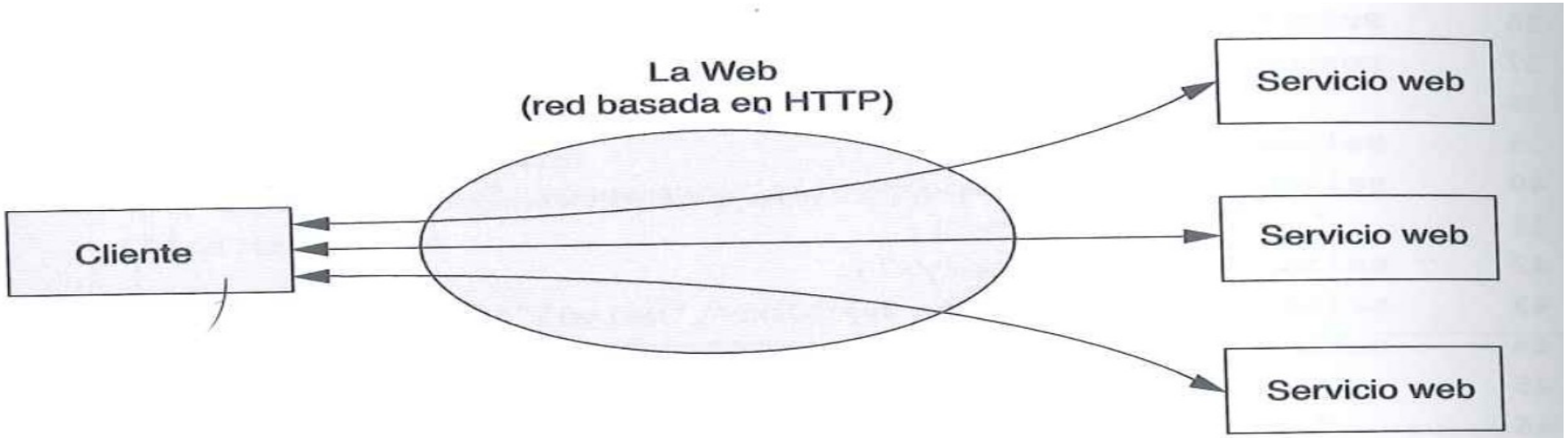


Figura 11.21. El modelo conceptual de servicios web.

Servicios de red: Servicios web

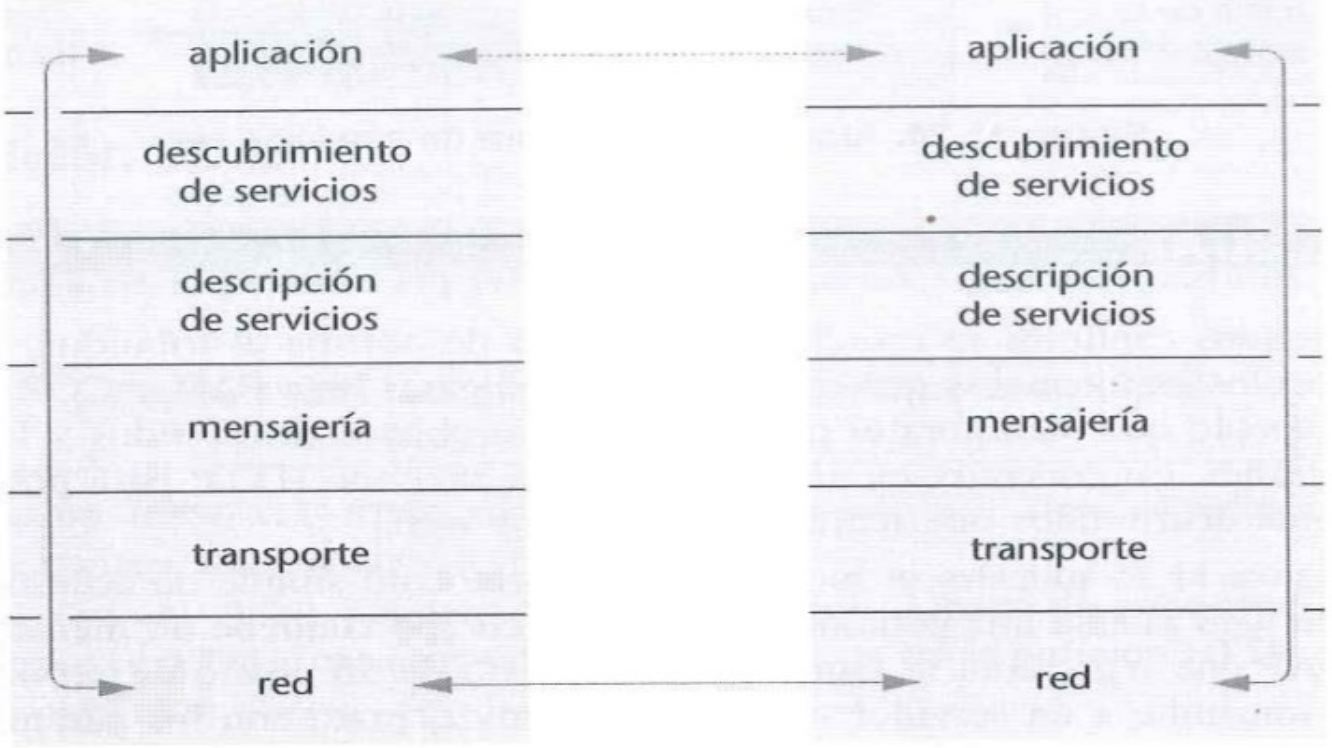
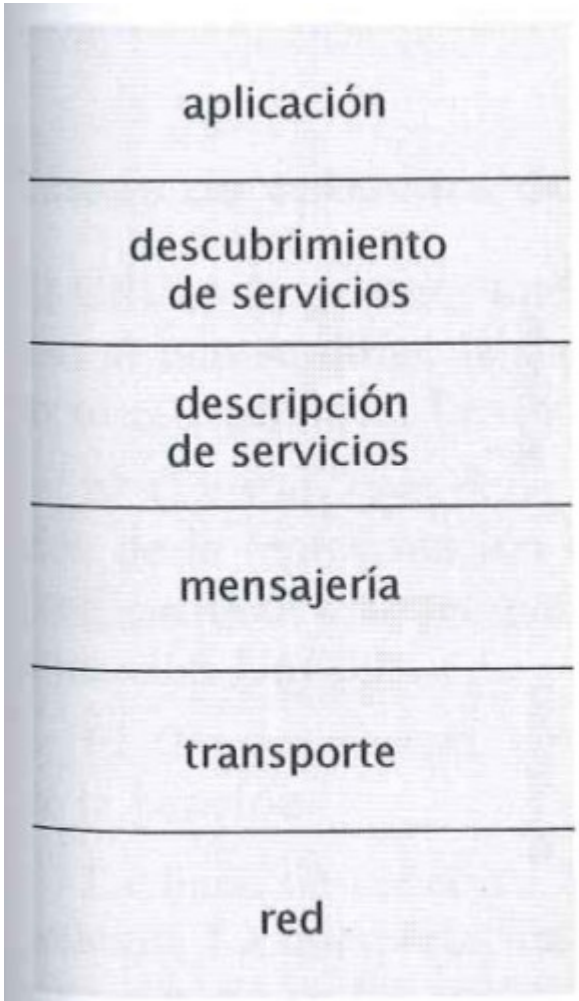


Figura 11.22. Jerarquía de protocolos de servicios web.

- El servidor y el cliente intercambian mensajes en la ***capa de aplicación*** en función del protocolo usado
- Un ***protocolo de descubrimiento de servicios*** permite al servicio ser registrado y localizado
- La ***capa de descripción del servicio*** permite que un servicio sea descrito en el directorio donde se registra
- La ***capa de mensajería*** proporciona los mecanismos para la intercomunicación de procesos
  - Incluyendo funcionalidades de empaquetamiento (marshaling) de datos
- La ***capa de transporte*** envía los mensajes
- La ***capa de red*** se ocupa de la transmisión física y el encaminamiento de paquetes

Servicios de red: Servicios web

- Protocolos predominantes para servicios web



UDDI (Universal Description, Discovery, and Integration, Descripción, Descubrimiento e Integración Universales).

WSDL (Web Service Description Language, Lenguaje de Descripción de Servicios Web).

XML, SOAP (Simple Object Access Protocol, Protocolo Simple de Acceso a Objetos).

TCP, HTTP, SMTP, Jabber.

IP.

Figura 11.23. Protocolos de servicios web.

- Arquitectura software de un servicio web
  - El ***escuchador del servicio*** de la máquina servidora recoge las peticiones de servicio transmitidas sobre la red
  - Cada petición se reenvía al ***proxy del servicio***
  - Éste invoca la ***lógica de la*** aplicación del objeto servicio y transfiere el valor devuelto al llamante

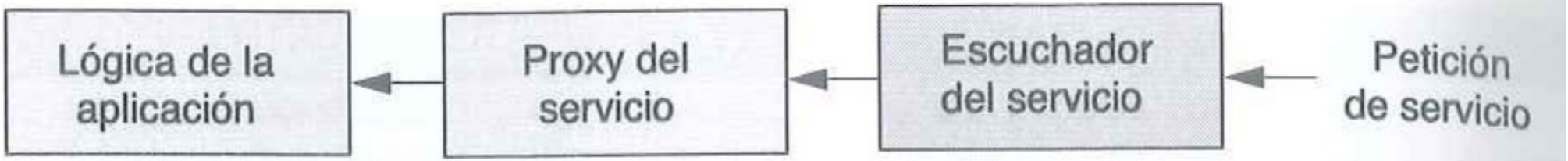


Figura 11.24. Arquitectura software de servicios web.

Servicios de red: SOAP (Simple Object Access Protocol)

- Es un protocolo que extiende HTTP para permitir acceso a *objetos distribuidos que representan servicios web.*

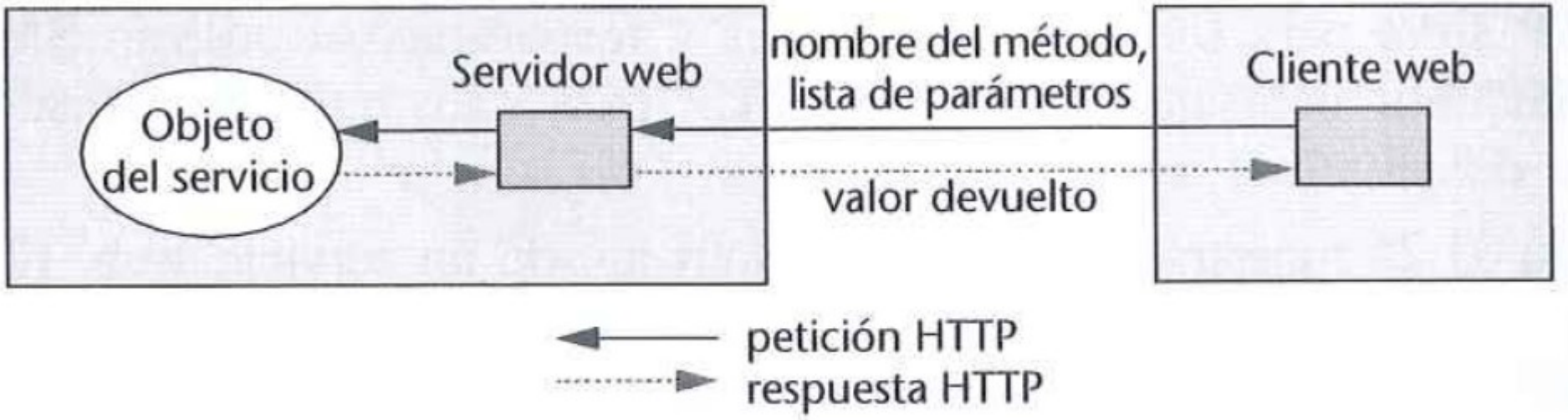


Figura 11.25. El modelo SOAP.

Servicios de red: SOAP (Simple Object Access Protocol)

- El mensaje SOAP se codifica en XML
- El formato de un mensaje SOAP es el siguiente

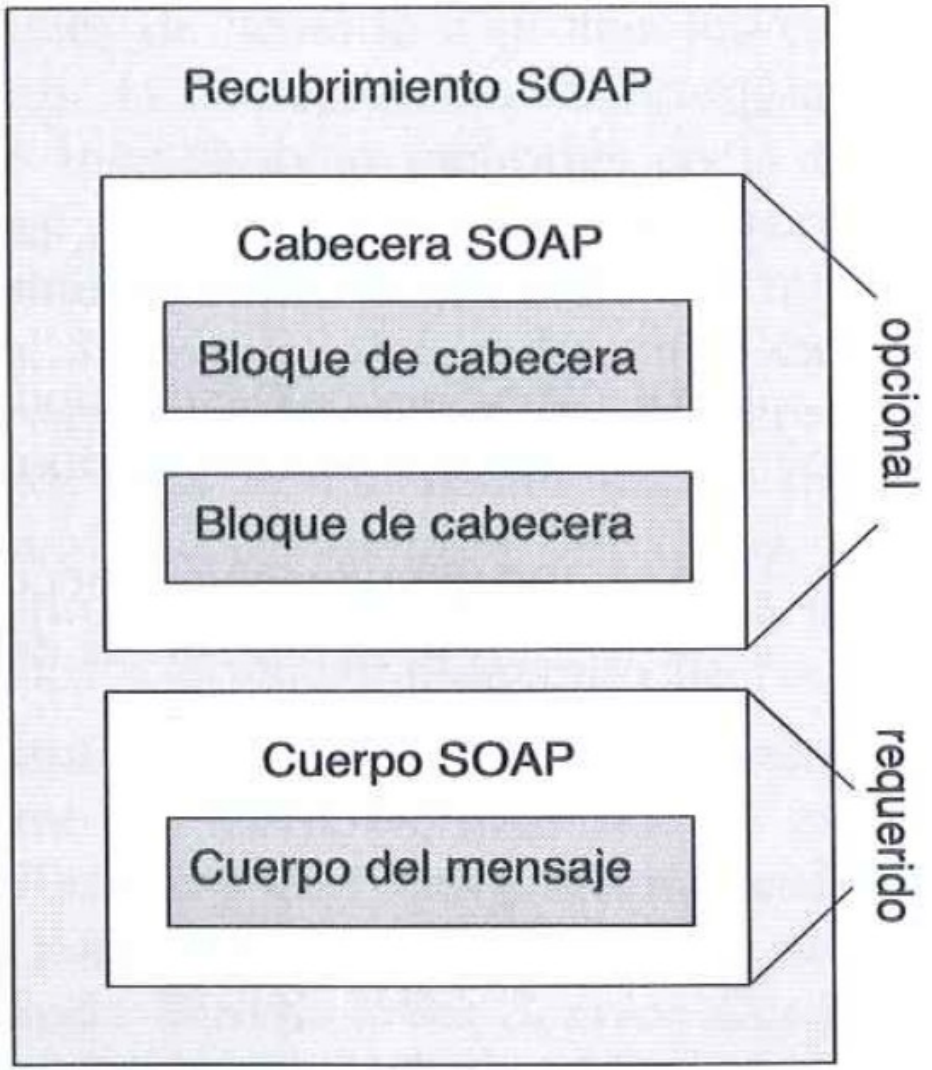


Figura 11.26. Esquema de un mensaje de petición SOAP.

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP
  - Cabecera

```
POST /examples HTTP/1.1
User-Agent: Radio UserLand/7.0 (WinNT)
Host: localhost:81
Content-Type: text/xml; charset=utf-8
Content-length: 474
SOAPAction: "/examples"
```

    - La primera linea se especifica el objeto remoto a través del URI (Uniform Resource Identifier o en español «identificador uniforme de recurso»): "/examples"

      URI = URL + URN

      (Uniform Resource Locator) (Uniform Resource Name)
    - User-Agent y Host deben especificarse
    - Content-Type debe especificarse como text/xml
    - charset puede ser
      - US-ASCII (opción por defecto)
      - UTF-8
      - UTF-16
    - Content-length, si está especificado, debe ser la longitud en byte del cuerpo de la petición
    - SOAPAction especifica el objeto remoto de la petición
      - Normalmente el URI y SOAPAction tendrán el mismo valor

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP
  - Cuerpo

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle =
    "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
    <SOAP-ENV:Body>
        <m:getStateName xmlns:m="http://www.soapware.org/">
            <statenum xsi:type="xsd:int">41</statenum>
        </m:getStateName>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

    - Recubrimiento SOAP (<SOAP-ENV:Envelope>)
      - Contiene una serie de atributos requeridos el esquema de codificación y el estilo del recubrimiento
    - Cuerpo SOAP (<SOAP-ENV:Body>)
      - Especifica el nombre del método a ejecutar: `getStateName`
      - Nombre del servidor: `www.soapware.org`
      - Por cada parámetro
        - Nombre: `statenum`
        - Tipo: `int`
        - Valor: `41`

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP

```
POST /examples HTTP/1.1
User-Agent: Radio UserLand/7.0 (WinNT)
Host: localhost:81
Content-Type: text/xml; charset=utf-8
Content-length: 474
SOAPAction: "/examples"
```

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle =
    "http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
    <SOAP-ENV:Body>
        <m:getStateName xmlns:m="http://www.soapware.org/">
            <statenum xsi:type="xsd:int">41</statenum>
        </m:getStateName>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP
  - Tipos de datos

Tabla 11.8. Tipos de datos escalares del esquema XML (Fuente: [soapware.org, 11]).

Valor del atributo	Tipo	Ejemplo
xsd:int	Entero con signo de 32-bit	-12
xsd:boolean	Valor booleano, 1 ó 0	1
xsd:string	Cadena de caracteres	Hola Mundo
xsd:float o xsd:double	Número de coma flotante con signo	–12,214
xsd:timeInstant	Fecha/Hora	2001-03-27T00:00:01-08:00
SOAP-ENC:base64	Binario codificado en Base64	eW91IGNhbid0IHJlYWQgdGhpcyE=

  - Estructuras

```
<param>
   <lowerBound xsi:type="xsd:int">18</lowerBound>
   <upperBound xsi:type="xsd:int">139</upperBound>
</param>
```

    - El orden de los elementos no es significativo, el nombre sí

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP
  - Tipos de datos
    - Arrays

```
<param SOAP-ENC:arrayType="xsd:ur-type[4]"
xsi:type="SOAP-ENC:Array">
<item xsi:type="xsd:int">12</item>
<item xsi:type="xsd:string">Egypt</item>
<item xsi:type="xsd:boolean">0</item>
<item xsi:type="xsd:int">-31</item>
</param>
```

      - El orden de los elementos es significativo, el nombre no
      - Cuando se mezclan los tipos de datos se expecifica `"xsd:ur-type[4]"`
      - Para declarar arrays con elementos de un solo tipo habría que sustituir por `SOAP-ENC:arrayType="xsd:int[4]"`
      - Para arrays con elementos de un solo tipo no es necesario especificar el `xsi:type` por atributo aunque es recomendable

Servicios de red: SOAP (Simple Object Access Protocol)

- Una petición HTTP con una petición SOAP
  - Tipos de datos
    - Objetos
      - Se puede transmitir un objeto si el proveedor del servicio define y registra el tipo del objeto como subtipo, y las dos partes proporcionan un serializador y deserializador apropiado.
      - El nombre del subtipo se declara como el atributo `xsi:type` del parametro
    - Valores nulos
      - Un valor puede ser también nulo
      - Se especifica como:

        ```
        <param1 xsi:null="1"/>
        ```

Servicios de red: SOAP (Simple Object Access Protocol)

- Una respuesta HTTP que contiene una respuesta SOAP

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 499
Content-Type: text/xml; charset=utf-8
Date: Wed, 28 Mar 2001 05:05:04 GMT
Server: UserLand Frontier/7.0-WinNT

<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
<SOAP-ENV:Body>
    <m:getStateNameResponse xmlns:m="http://www.soapware.org/">
        <Result xsi:type="xsd:string">South Dakota</Result>
    </m:getStateNameResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

  - La estructura es identica: cabecera, recubrimiento y cuerpo
  - Se le añade `Response` al nombre del método llamado
  - El `Result` de tipo `string` devuelve el valor, `Dakota del Sur`

Servicios de red: SOAP (Simple Object Access Protocol)

- Una respuesta HTTP que contiene una respuesta SOAP erronea

```
HTTP/1.1 500 Server Error
Connection: close
Content-Length: 511
Content-Type: text/xml; charset=utf-8
Date: Wed, 28 Mar 2001 05:06:32 GMT
Server: UserLand Frontier/7.0-WinNT

<?xml version="1.0"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance">
<SOAP-ENV:Body>
    <SOAP-ENV:Fault>
        <faultcode>SOAP-ENV:Client</faultcode>
        <faultstring>Can't call getStateName because there are too many parameters.</faultstring>
    </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

  - En el cuerpo se define un código error (`SOAP-ENV:Client`) y una cadena de error que describe el problema (`Can't call getStateName because there are too many parameters.`)

Servicios de red: AXIS

- Axis es una implementación Open-Source de un "SOAP Engine";
- A través de este componente es posible llevar acabo una comunicación mediante "Web-Services" una de las variaciones más prometedoras del lenguaje XML .
- En este link se explica como montar servicios web usando AXIS

  http://www.osmosislatina.com/axis/basico.htm