



Procesos

Prof. Mariela Curiel



Bibliografía

- A. Tanenbaum & M. Van Steen. “Sistemas Distribuidos. Principios y Paradigmas”. 2da. Edición.
- Smith & Nair. The Architecture of Virtual Machines. IEEE Computer. 2005.
- Fuggetta et al. Understanding Code Mobility. IEEE Transactions on Software Engineering
- Vivek, Aron, Banga. “Locality-Aware Request Distribution in Cluster Based Network Servers”



Contenido

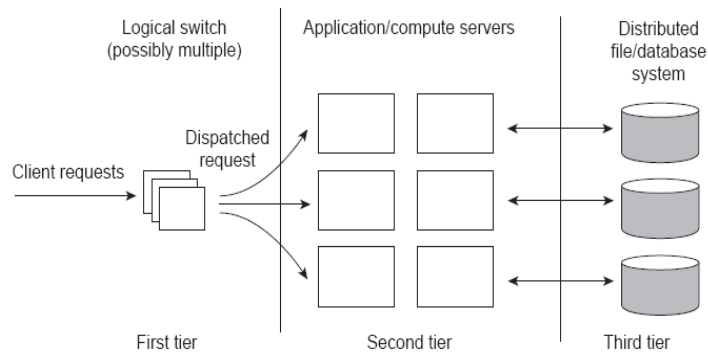
- Hilos
 - Repaso del concepto de hilos y procesos
 - Hilos en sistemas distribuidos (arquitectura C/S).
- Virtualización (Virtualización de Proceso, V. de Sistema)
- Clientes
- ▶ Servidores (servidores de clústeres, servidores distribuidos)
- Migración de Código



Servidores de Clústeres

- Es una colección de máquinas conectadas a través de una red (LAN), donde cada máquina ejecuta uno o más servidores de aplicación. La red tiene un gran ancho de banda y la latencia muy pequeña.
- La mayoría de los servidores de clusters están organizados dentro de tres niveles:

Servidores de Clústeres



Servidores de Clústeres

- El primer nivel es un switch o interruptor lógico que enruta las peticiones del cliente hacia el componente del cluster adecuado.
- En el segundo nivel se encuentran los servidores dedicados a procesar aplicaciones. Contienen Hw para computación de alto rendimiento.



Servidores de Clústeres

- En el caso de servidores empresariales, pudiera ser que el poder de cómputo requerido no sea el cuello de botella, sino que requieran gran capacidad de almacenamiento: El tercer nivel consta de servidores de procesamiento de datos (sistemas de archivos), servidores de BD.
- No todos los clústeres siguen esta separación, es frecuente que cada máquina tenga sus propias unidades de almacenamiento (arquitectura de dos niveles).



Servidores de Clústeres

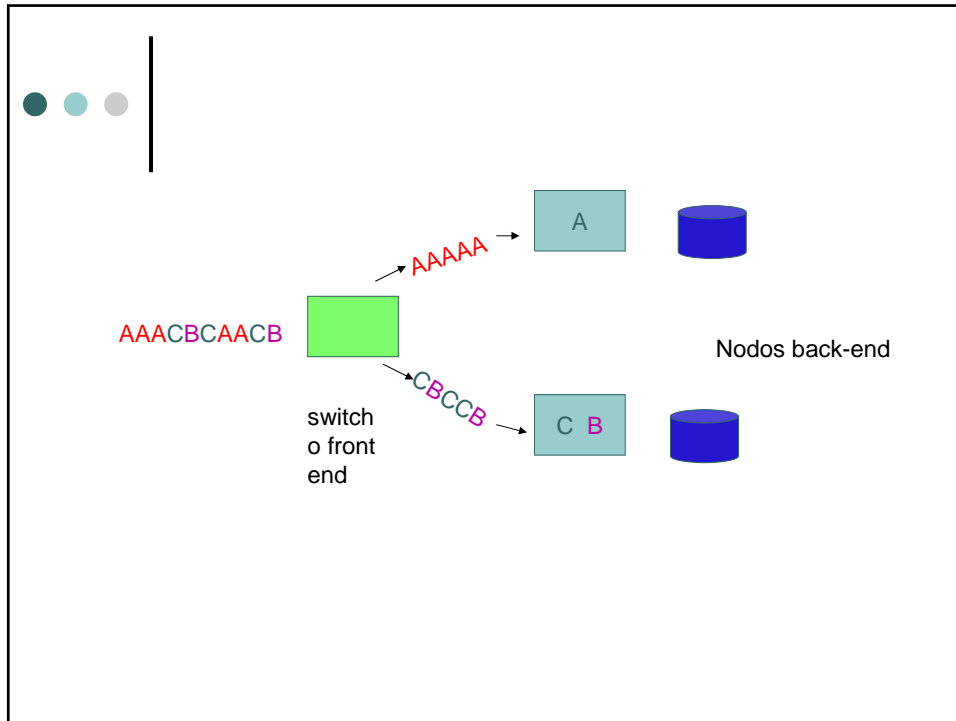
- Cuando el servidor de cluster ofrece múltiples servicios, pudiera ser que nodos diferentes dentro del cluster ejecuten servidores de aplicaciones distintos.
- En este caso el switch tiene que tener la capacidad de distinguir los servicios para enviar las peticiones a las máquinas apropiadas (sólo basta observar el número de puerto dentro de la petición)

● ● ● | Servidores de Clústeres: Estrategias para asignar tareas a los servidores

- La mayoría de las implementaciones son servidores de dos niveles que ejecutan únicamente un único servicio.
- La asignación de los requerimientos a los servidores busca principalmente balancear la carga. El switch envía los requerimientos a los nodos sin importar el tipo de servicio o el contenido que se está solicitando. Todos los nodos se consideran con igual capacidad de servir cualquier request. Una estrategia sencilla puede ser round-robin.

● ● ● | Servidores de Clústeres: Estrategias

- Hay otros enfoques: **Locality-Aware Request Distribution**: Se distribuyen las **peticiones de acuerdo al “contenido” junto con** información relacionada con la carga de cada nodo. La idea es enviar cada requerimiento al nodo que ya tiene los datos en su cache.
- Esta estrategia debe aumentar el rendimiento debido principalmente al aumento en la cantidad de hits en los caches.
- También se incrementa la escalabilidad porque los datos se distribuyen entre los distintos nodos del cluster.



● ● ● | **Servidores de Clústeres:
Estrategias**

- Migración de código de servidores más cargados a servidores ociosos...



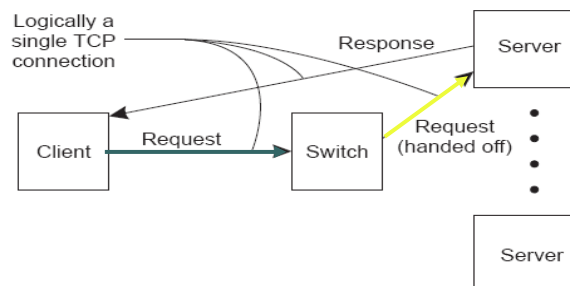
Servidores de Clústeres: El switch

- Una meta importante en el diseño de servidores de clústeres es ocultar el hecho de que existen múltiples servidores.
- Esta transparencia de acceso se ofrece dado que se tiene un único punto de acceso, el interruptor, que pudiera ser una máquina dedicada.
- Es el único punto de entrada, por lo cual se ofrece una única dirección de red.



Servidores de Clústeres: El switch

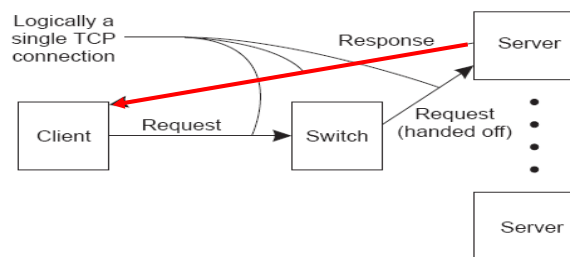
- El switch acepta peticiones entrantes de una conexión TCP y re-direcciona las peticiones hacia uno de los nodos. Para cada petición se re-envía el paquete con la misma, al nodo o servidor seleccionado.





Servidores de Clústeres: El switch

- El nodo seleccionado envía una respuesta al cliente, pero insertando la dirección IP del switch como campo fuente del encabezado del paquete.



Servidores de Clústeres: El switch

- Lo anterior es necesario para que el cliente pueda continuar con la ejecución del protocolo TCP: está esperando una respuesta por parte del switch, no de algún servidor aleatorio con el cual nunca se ha comunicado.



Servidores Distribuidos

- Los servidores de clústeres anteriores son configurados de forma estática. Ofrecen un conjunto de máquinas y un único punto de acceso.
- Cuando el punto falla, el cluster ya no está disponible.
- Para eliminar este problema se pueden proporcionar varios puntos de acceso. DNS (Servicio de Nombres de Dominio) pudiera devolver diversas direcciones, todas asociadas al mismo nombre.



Servidores Distribuidos

- Lo ideal es tener un punto de acceso con vida prolongada. Tener flexibilidad en la configuración del cluster: que tanto los nodos como el switch puedan variar.
- **Servidor Distribuido**: un conjunto de máquinas que cambian de manera dinámica y posiblemente también tengan más de un punto de acceso. Ante el mundo aparecen como un único y poderoso servidor.



Servidores Distribuidos

- Cómo se puede lograr:
 - IP versión 6: Nodo móvil, está asociado a una red doméstica donde reside normalmente y una dirección doméstica (HoA). Esta red tiene un enrutador (agente doméstico) que se encarga de enrutar el tráfico cuando el nodo móvil se sale de la red. Cuando el nodo se conecte a una red externa recibirá una dirección añadida cuidadosamente (CoA). Esta dirección se reporta al agente doméstico quien verá que todo el tráfico se re-envie a la nueva dirección. Todas las aplicaciones que se comunican con el nodo móvil sólo ven la HoA.

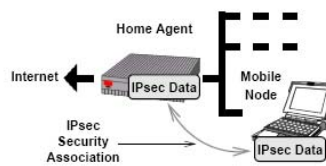


Fig. 1. Home network in Mobile IPv6

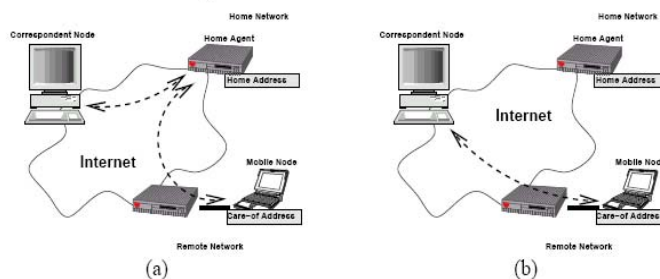


Fig. 2. Communication in MIPv6: tunneling (a), and route optimization (b)



Servidores Distribuidos

- Cómo se puede lograr:
 - Se asigna una dirección de contacto al punto de contacto del servidor distribuido (la que usa toda la vida y conocen todas las aplicaciones). Si el rol de servidor de contacto lo toma otro nodo, la dirección de este nuevo nodo se registra como una CoA (dirección añadida cuidadosamente) con el agente doméstico asociado al servidor distribuido.
 - El agente re-direcciona todo tráfico que viene a la dirección de contacto a la nueva dirección. El punto de acceso redirecciona las peticiones hacia los nodos que actualmente participan.



Servidores Distribuidos

- Para mejorar el desempeño se puede usar la el métodos de MIPV6 conocido como optimización de ruta:
 - Cada vez que un nodo movil con dirección doméstica HoA informa su CoA al agente doméstico. Este último puede re-enviar su CoA al cliente. A partir e ese momento el Cliente puede enviar directmanet los datos a la CoA. Aunque del lado del cliente la aplicación sigue usando la misma HoA (dirección doméstica), el software de soporte subyacente para MIPv6 traducirá dicha dirección a la CoA.



Contenido

- Hilos
 - Repaso del concepto de hilos y procesos
 - Hilos en sistemas distribuidos.
- Virtualización
- Clientes
- Servidores
- ▶ Migración de Código



Migración de Código

Hasta ahora sólo nos hemos preocupado por el intercambio de datos en SD. Sin embargo existen situaciones en las que el pase de un programa (de una máquina a otra), a veces incluso mientras se ejecutan, puede simplificar el diseño de un sistema distribuido.



Migración de Código

- Razones para la Migración de Código
- Tecnología y Mecanismos
- Paradigmas de Diseño
- Migración en Entornos Heterogéneos.



Utilidad de Migrar Código

- Inicialmente se hablaba de migrar procesos: trasladar un proceso de una máquina a otra (suspender la ejecución, migrar código y estado y reiniciar la ejecución en otro computador).
- La migración se hacía básicamente para balancear la carga, trasladando procesos desde nodos más saturados a nodos menos saturados. Indicadores de desempeño: longitud de cola del CPU, uso de la memoria, etc.



Utilidad de Migrar Código

- Hoy en día se habla de otros dominios de aplicaciones, por ejemplo:
 - ***Distributed Information Retrieval***: se trata de encontrar datos que satisfacen un determinado criterio en un conjunto de fuentes de información de tamaño importante y dispersas en la red. Las fuentes pueden definirse de forma estática o determinarse en forma dinámica durante el proceso de búsqueda. *La migración del código puede mejorar la eficiencia, migrando el código que ejecuta la búsqueda al mismo lugar (en la misma máquina) donde se encuentra la fuente de información.*



Utilidad de Migrar Código

- ***Migrar partes de un servidor al cliente (Documentos Activos)***: Las páginas estáticas se pueden mejorar con la capacidad de ejecutar programas que están relacionados con el contenido. Por ejemplo una aplicación que usa formas para componer y enviar queries a una BD remota. Este tipo de aplicación puede ser implementada usando una tecnología que haga “fetch” de fragmentos de código remoto (Java applets).



Utilidad de Migrar Código

- *Explotación de paralelismo*: Búsqueda de información en la WEB. Ejem: se implementa una consulta con un programa móvil, llamado agente móvil, que se traslada de sitio en sitio. Al hacer distintas copias de dicho programa y enviar cada copia a un sitio distinto, se aumenta el paralelismo y se mejora la eficiencia de la aplicación.



Utilidad de Migrar Código

- **Aplicaciones para monitorear/controlar dispositivos remotos**: La migración de código se puede usar para diseñar e implementar componentes de monitoreo que se *instalan* en los dispositivos a monitorear y reportan eventos que representen la evolución del estado del dispositivo.



Migración de Código

- Razones para la Migración de Código
- ▶ Tecnología y Mecanismos
- Paradigmas de Diseño
- Migración en Entornos Heterogéneos.



Enfoques más recientes...

- Hoy en día se habla de sistemas de código móvil: Mobile Code System (MCS), los cuales presentan ciertas innovaciones con respecto a los enfoques anteriores:
 - La movilidad se explota en la internet: heterogeneidad, anchos de banda diferentes.
 - El programa está consciente de su localización y la movilidad está bajo el control del programador.
 - La movilidad se realiza por otras razones, no necesariamente por balanceo de carga.

Mobile Code Technologies: Arquitectura para la Migración

The diagram illustrates the architecture of traditional systems. At the top, four circles represent 'Component' instances. Below them is a large box labeled 'True Distributed System'. Inside this box, there are three columns, each containing a 'Network Operating System' box above a 'Core Operating System' box. Below these are three 'Host' boxes. A vertical label 'Hardware' is on the right side of the host layer. Annotations with arrows point to the components, network OS, and core OS layers.

- Esconde que dentro de la red existen componentes remotos. Se perciben como componentes locales (CORBA)
- Servicios de comunicación no transparentes (sockets)
- Funciones tradicionales

Traditional systems

33

Mobile Code Technologies

Las tecnologías actuales que soportan la movilidad tienen una perspectiva diferente. La estructura de la red no se esconde al programados sino que se pone de manifiesto. Los TDS se reemplazan por los ambientes de Computación (CE). Estos retienen (conocen) la identidad del host donde se encuentran.

The diagram illustrates the architecture of mobile code technologies. It shows three columns, each representing a host. At the top, circles represent 'Component' instances. Below them is a box labeled 'Computational Environment'. Inside this box, there is a 'Network Operating System' box above a 'Core Operating System' box. Below these are three 'Host' boxes. A vertical label 'Hardware' is on the right side of the host layer. The number '34.' is at the bottom left.

34.

Mobile Code Technologies

Un CE ofrece a las aplicaciones la capacidad de relocalizar sus componentes dinámicamente en diferentes hosts.

Cada **componente** se divide en: **Unidades de Ejecución** (threads, procesos, etc) y **Recursos** (entidades que se pueden compartir entre múltiples UE).

Mobile Code Technologies

Unidad de Ejecución

Code segment

Execution state (stack and instruction pointer)

Data space

Computational Environment

Fig. 2. The internal structure of an executing unit.



Mobile Code Technologies

- *El segmento de código:* Es el conjunto de instrucciones. El Programa.
- *El estado de Ejecución:* datos privados, la pila, el PC
- *Espacio Datos:* referencias a recursos externos necesarios para el proceso tales como archivos, impresoras, dispositivos , otros procesos.



Mobile Code Technologies

- En sistemas convencionales el EU está asociada a un único ambiente computacional (CE) durante su tiempo de vida.
- En un MCS (Mobile Code System) el Segmento de Código, el Estado y el espacio de datos se puede re-asignar a un diferente CE.



Mecanismos

- **Movilidad Fuerte:** Se puede migrar tanto el código como el estado a otro CE. Se puede implementar mediante dos mecanismos: migración y clonación.
- **Migración:** Suspende el proceso (EU), transmite éste a su destino y luego lo resume.
- **Clonación:** se crea una copia del proceso (EU) en otro CE, el original EU sigue ejecutándose.
- **Movilidad Débil:** Se transfiere únicamente el segmento de código, junto con algunos datos de inicialización. Un proceso transferido siempre comienza en algunos puntos predefinidos. Por ejemplo los applets de Java siempre comienzan su ejecución desde el comienzo.



Mecanismos

- Sin importar si se trata de clonación o migración o si la movilidad es fuerte o débil se puede hacer una distinción adicional si la migración es iniciada por el remitente o por el destinatario.
- **Iniciada por el remitente (proactiva):** la migración comienza en la máquina donde reside o se ejecuta el código. Envío de agentes a realizar una determinada búsqueda .
- **Iniciada por el destinatario (reactiva):** la máquina destino toma la iniciativa para realizar la migración. Los applets de Java son un ejemplo. Programas para actualizar versiones del sistema de operación.



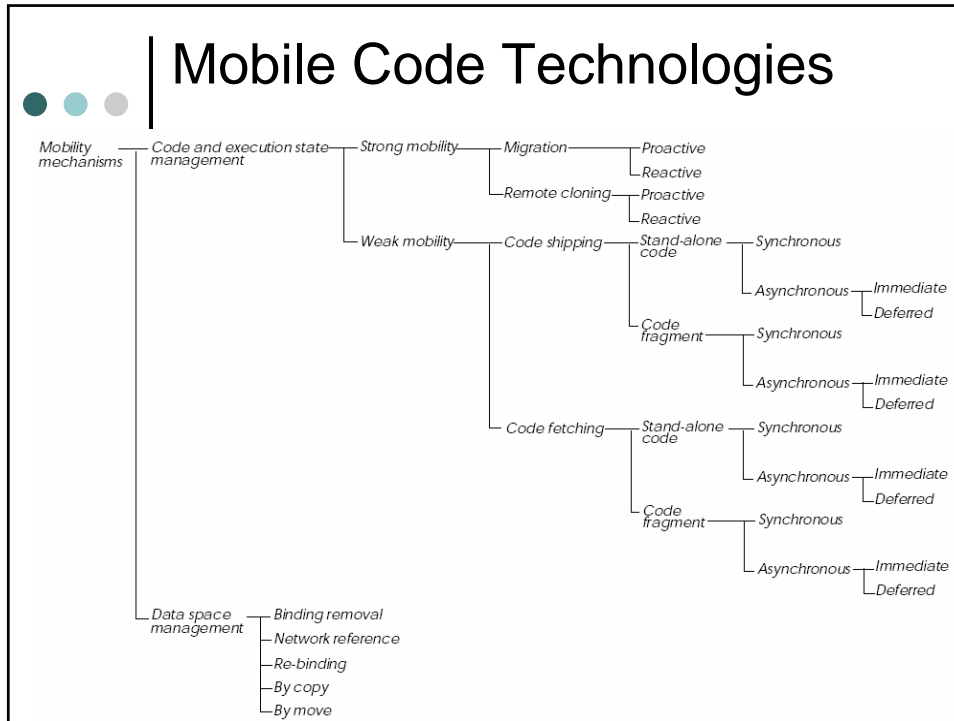
Mecanismos

- **Iniciada por el destinatario:** es más simple. Puede ocurrir entre un cliente y un servidor, y es el cliente quien toma la iniciativa. Se realiza de manera anónima, el servidor no está interesado en los recursos del cliente.
- La carga segura de un código hacia el servidor, con frecuencia requiere que el cliente se encuentre registrado y autenticado. El servidor debe conocer todos sus clientes. El cliente probablemente requiera acceder recursos del servidor y es necesario la protección de dichos recursos.



Mecanismos

- **Movilidad Débil:**
 - Una EU puede solicitar código (fetch) para ser enlazado/ejecutado dinámicamente o enviar el código a otro CE.
 - Se puede migrar un código o programa entero (se instancia una nueva EU en el destino) o fragmentos de código (clases) (pueden enlazarse al contexto de un código que ya se está ejecutando).
 - Síncrono o asíncrona: depende de si la EU requiriendo la transferencia se suspende o no hasta que llegada del código.
 - Ejecución diferida o inmediata: si el código que llega se ejecuta inmediatamente o posteriormente, cuando se satisface alguna condición.



- # Mobile Code Technologies
- Data Space Management
 - Mechanisms
 - Binding removal
 - Network reference
 - Re-binding
 - By copy
 - By move
 - Resource model
 - Resource = <I, V, T>
 - Binding: *By Identifier, By Value, By Type*
 - Transferable vs. Non-Transferable
- 44



Recursos: Tipo de asociación del proceso al recurso

- *Enlace por Identificador*: se hace referencia a un recurso mediante su identificador. Se requiere explícitamente el recurso al que se hace referencia. Ejem: un URL, un dirección IP de un servidor FTP.
- *Enlace por Valor*: se requiere solamente el valor del recurso. Ejm: rutinas de librería, no importa los archivos ni donde se encuentren sino el código o constantes que definan dichas librerías.
- *Enlace por Tipo*: se requiere un tipo de recurso específico, por ejemplo un terminal, una impresora.



Recursos

- Fijos: Están ligados a una máquina, no se pueden trasladar (dispositivos locales), recursos del SOP.
- Transferibles: Ejem. un archivo.
 - No adjuntos: se pueden trasladar fácilmente entre diversas máquinas.
 - Adjuntos: trasladarlo o copiarlo puede ser posible pero a un costo muy alto. Ejem: Una base de datos o un sitio WEB completo.



Mobile Code Technologies

Transferibles

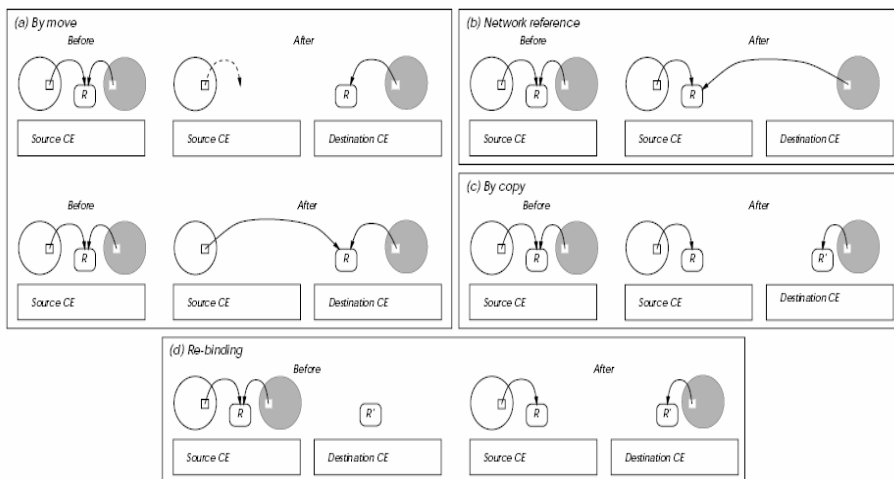
	No adjuntos	Adjuntos	Fijos
By Identifier	By move (Network reference)	Network reference	Network reference
By Value	By copy (By move, Network reference)	By copy (Network reference)	(Network reference)
By Type	Re-binding (Network reference, By copy, By move)	Re-binding (Network reference, By copy)	Re-binding (Network reference)

TABLE I

BINDINGS, RESOURCES AND DATA SPACE MANAGEMENT MECHANISMS.



Mobile Code Technologies



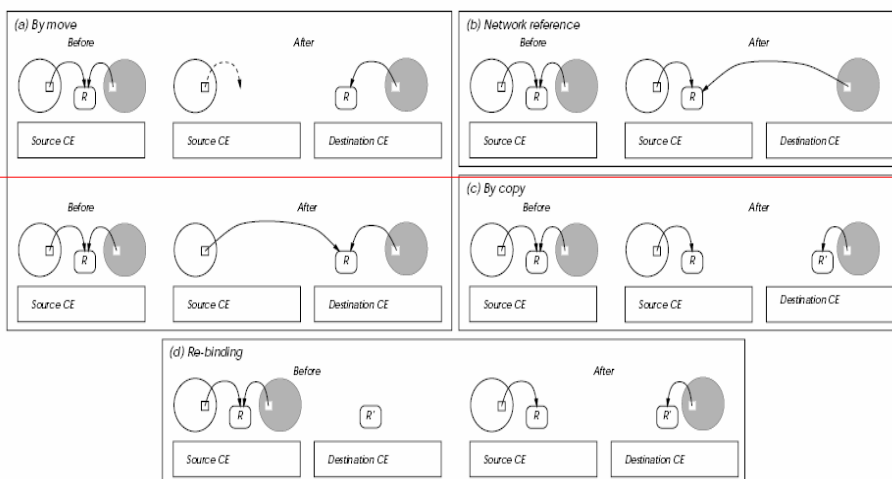


Recursos

- Retirar el enlace al recurso. La Unidad de ejecución U migra el CE.
 - Si U *estaba enlazado por identificador* al recurso: si se puede transferir el recurso, se mueve (se re-establece el enlace en la máquina destino) (a). Si el recurso no se puede mover se establece una referencia remota (b)



Mobile Code Technologies



OU

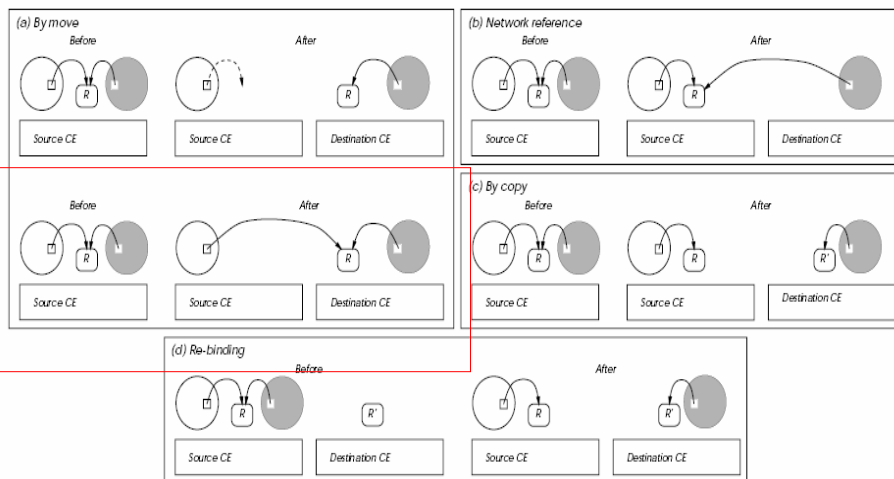


Recursos

- Retirar el enlace al recurso. La Unidad de ejecución U migra.
 - Mover el recurso de su CE puede causar problemas (a, abajo). No se traslada el recurso sino que se establece una referencia remota.



Mobile Code Technologies



32

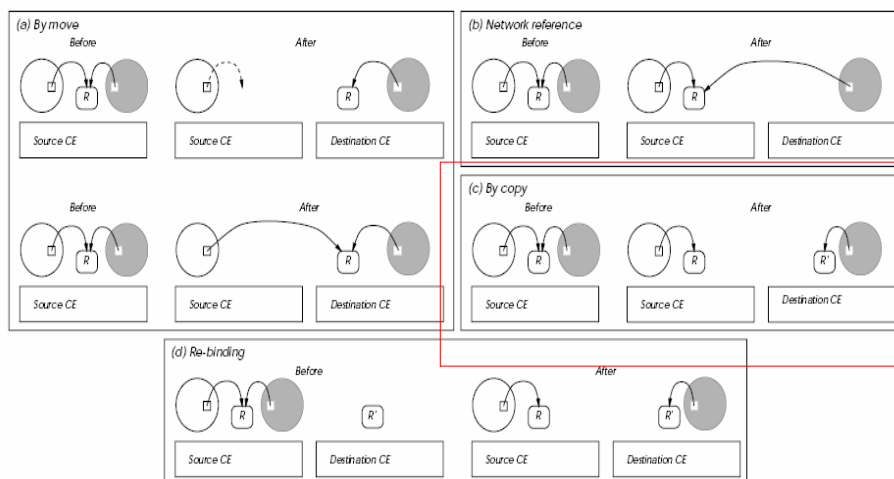


Recursos

- o Si B es por valor y el recurso es transferible, lo más conveniente es crear una copia del recurso en el destino (R') y en el destino re-establecer el enlace a R' (c). Si R no se puede transferir la única solución es una referencia remota. Si un recurso con el mismo valor ya está disponible en el CE destino, se evita la copia.



Mobile Code Technologies



34



Recursos

- o Si B es por tipo lo más conveniente es el re-enlace. El enlace se elimina y se establece, después de la migración, a un recurso R' en el CE destino, que tiene el mismo tipo de R (d)



Recursos

- o La naturaleza del recurso y el tipo de enlace vienen determinadas normalmente por el lenguaje más que por el programador de aplicaciones.



Mobile Code Technologies

- A Survey of Mobile Code Technologies
 - Agent Tcl
 - Ara
 - Facile
 - Java
 - Java Aglet
 - M0
 - Mole
 - Obliq
 - Safe-Tcl
 - Sumatra
 - TACOMA
 - Telescript



Tecnologías de Código Móvil

- Agent Tcl: una EU, es un proceso UNIX que ejecuta el interpretador del lenguaje (TCL). Como los procesos corren en espacio de direcciones separados sólo pueden compartir recursos del SOP. Tales recursos se consideran no transferibles.
- En Agent TCL los agentes pueden saltar a otro CE, clonar (fork) un EU a otro CE remoto o enviar un trozo de código a un remoto CE.



Tecnologías de Código Móvil

- En el primer caso se mueve el interprete completo con el código del procesos y su estado. En el segundo caso se hace una clonación remota (creación remota de copias de los procesos). En el tercer caso se crea un nuevo EU en el destino para ejecutar el script (shipping o envío, movilidad débil). El mecanismo es asíncrono e inmediato. Una copia de las variables de ambiente de la EU que solicita el envío del script se pueden enviar también al destino.



Tecnologías de Código Móvil

- Java: La máquina Virtual Java que interpreta el bytecode, constituye el CE.
- Java provee un mecanismo programable (class loader) para recuperar y enlazar clases dinámicamente a un programa en ejecución.
- El class-loader se invoca cuando el código que se está ejecutando contiene un nombre de clas eno resuelto.



Tecnologías de Código Móvil

- El class-loader recupera la clase, probablemente desde un servidor remoto y la carga en la JVM. El código de la clase comienza a ejecutarse.
- Este proceso puede ser activado de forma explícita por una aplicación, independientemente de la necesidad de ejecutar o no el código.
- Por lo tanto Java soporta: movilidad débil, usando un mecanismo para traer (fetch) de fragmentos de código. El código se ejecuta desde el comienzo, no se trabaja con el “estado” ni con asociaciones a recursos.



Tecnologías de Código Móvil

- Los applets pueden ser descargados con páginas HTML para permitir dinamismo en la presentación y acceso interactivo al servidor. La combinación de un browser con una JVM se puede ver como una tecnología por sí misma . En este caso los browsers son CE y los applets son EU ejecutándose concurrentemente. La descarga de applets se puede ver como un mecanismo que ofrece el browser para traer (fetch) de código stand-alone.



Paradigmas de Diseño

La meta del diseño es la creación de una arquitectura de software: el software se descompone en componentes y se indican sus interacciones.

Las arquitecturas de software con características similares se denominan estilos arquitectónicos o paradigmas de diseño.

Los enfoques tradicionales no son suficientes cuando se diseñan aplicaciones móviles, el concepto de localización es importante.

Es importante identificar paradigmas de diseño razonables en SD que exploten la movilidad del código.

63



Definiciones

- Los elementos de la arquitectura, se dividen en:
 - componentes de código (know-how)
 - recursos: que son los datos o dispositivos usados durante el cálculo (la computación).
 - Componentes computacionales: la entidad activa que lleva a cabo la ejecución



Conceptos

- Interacciones: eventos que involucran a dos o mas componentes.
- Sites: albergan o contienen los componentes



Paradigmas de Diseño

- Cliente-Server (CS)
- Evaluación Remota (REV)
- Código bajo demanda (COD)
- Agentes Móviles (MA)



Metáfora

Luisa y Cristina interactúan para hacer una torta.

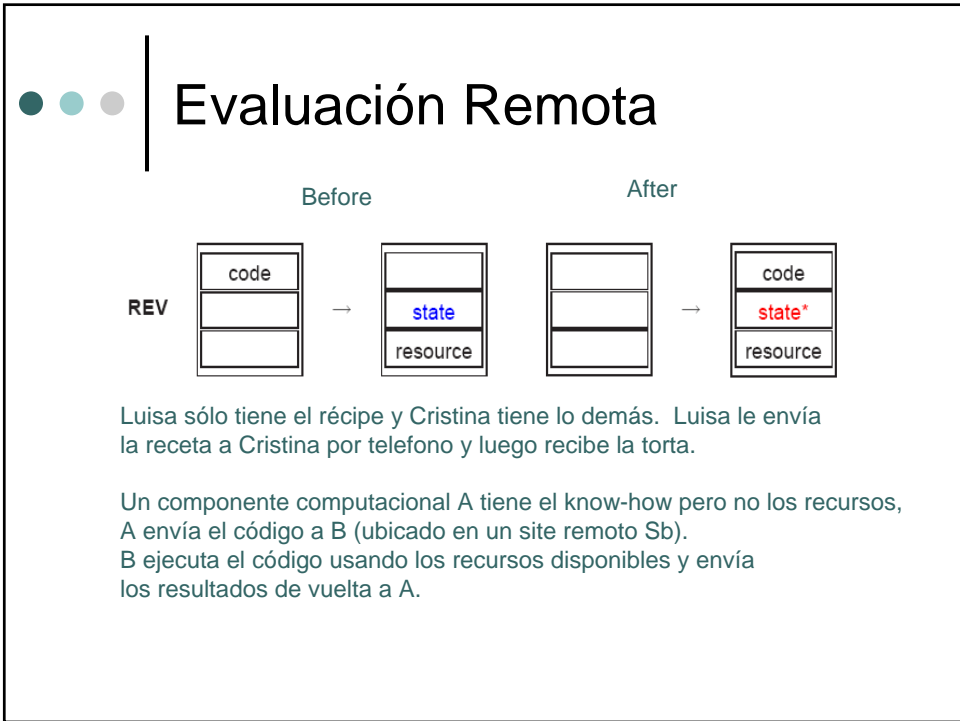
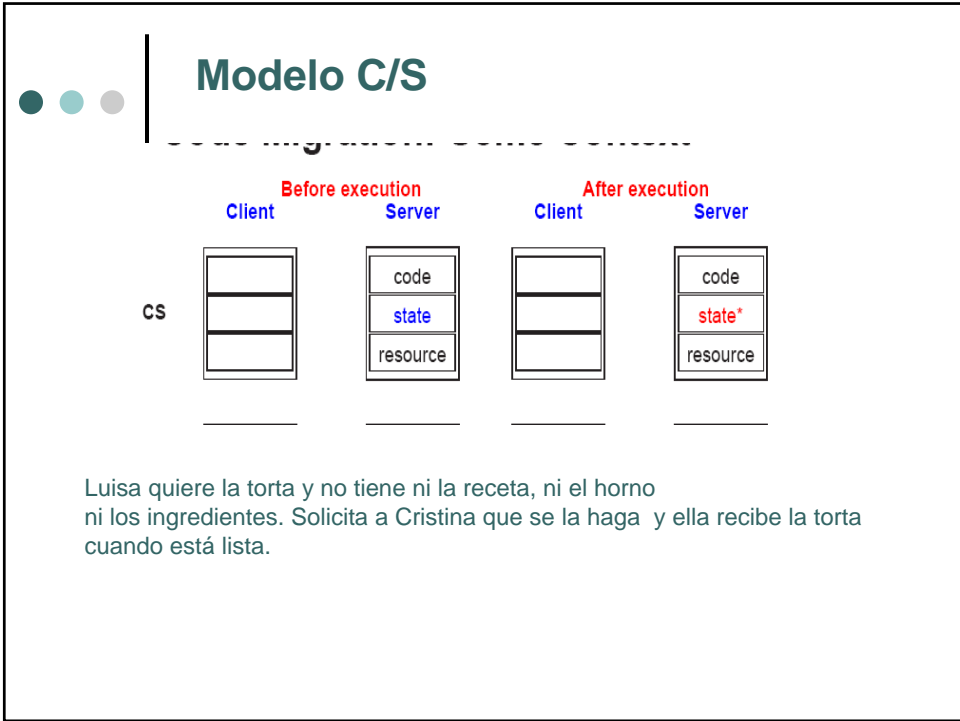
- Receta o algoritmo: *know-how* (componente de código)
- El Horno y los ingredientes son los recursos
- La persona que mezcla los ingredientes es el elemento o componente computacional.

Para preparar la torta todos los componentes deben encontrarse en el mismo Site.



Design Paradigms

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
Client-Server	A	know-how resource B	A	know-how resource B
Remote Evaluation	know-how A	resource B	A	<i>know-how</i> resource B
Code on Demand	resource A	know-how B	resource <i>know-how</i> A	B
Mobile Agent	know-how A	resource	—	<i>know-how</i> resource A



Code on Demand

Luisa tiene los Ingredientes y el horno pero no la receta. Luisa le solicita a Cristina la receta, y ella se la da por teléfono y Luisa prepara la torta en su casa.

A tiene los recursos pero ninguna información para manipularlos. A interactúa con un componente B en otro sitio (Sb) y le solicita el servicio del know-how. B le envía el código a A.

Mobile Agent

Luisa tiene el recipe y los ingredientes pero no el horno. Luisa va con la torta batida (o sin batir) a la casa de Cristina a hornear la torta.

A tienen el know-how e inicialmente se encuentra en Sa. Algunos recursos están en Sb. A migra a Sb, llevando consigo el know-how y algunos resultados intermedios. Una vez en Sb, A culmina los servicios. El componente computacional se desplaza (estado)



Design Paradigms

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
<i>Client-Server</i>	A	know-how resource B	A	know-how resource B
<i>Remote Evaluation</i>	know-how A	resource B	A	<i>know-how</i> resource B
<i>Code on Demand</i>	resource A	know-how B	resource <i>know-how</i> A	B
<i>Mobile Agent</i>	know-how A	resource	—	<i>know-how</i> resource A

73



Migración en Sistemas Heterogéneos

- Está la máquina destino preparada para ejecutar el código???? (sólo si hablamos de sistemas homogéneos).
- Solución: Máquinas Virtuales
 - Lenguajes Interpretados ejecutándose en una máquina virtual de procesos: Java
 - Máquinas virtuales de sistema que permiten la migración de sistemas operativos completos y aplicaciones.