Sistemas Distribuidos. Temario.

1. Introducción
2. Comunicación
3. Procesos
4. Nombrado y localización
5. Sincronización
6. Consistencia y replicación
7. Tolerancia a fallos

Tema 4.- Nombrado y localización

1. Introducción
2. Localización de entidades móviles
3. Recolección de residuos

Bibliografía: Capítulo 4 de Tanenbaum

1.- Introducción

1. Terminología
2. Espacios de nombres
3. Resolución de nombres
4. Aspectos de implementación

1.1.- Terminología

- Un **nombre** es una cadena de caracteres que se utiliza para referirse a una entidad.
- Una **entidad** puede ser cualquier recurso que sea deseable referenciar de forma remota: máquinas, impresoras, discos, ficheros, buzones, procesos, objetos, etc.
- Es posible operar sobre las entidades.
- Para operar sobre una entidad es necesario conocer su **punto de entrada**.
- El punto de entrada es una entidad especial, cuyo nombre es una **dirección**.
- Una entidad puede tener más de un punto de entrada.
- Ejemplo:
  - Entidades: personas
  - Nombres: nombre de las personas
  - Puntos de entrada: teléfonos.

  Una persona puede tener más de un teléfono.

1.1.- Terminología

- Una entidad puede cambiar su punto de entrada durante su vida.
- **Una dirección puede verse como un tipo especial de nombre: hace referencia a un punto de entrada de una entidad.**
- Un **identificador** es un nombre que tiene las siguientes propiedades:
  - Un identificador referencia como mucho una entidad.
  - Cada entidad es referenciado como mucho por un identificador.
  - Un identificador siempre referencia a la misma entidad (nunca se reutiliza)

1.- Introducción

1. Terminología
2. Espacios de nombres
3. Resolución de nombres
4. Aspectos de implementación

1.2.- Espacios de nombres

- Los nombres suelen organizarse en **espacios de nombres.**
- Un espacio de nombres puede representarse como un árbol:
  - Una **hoja** representa una entidad, y habitualmente contiene información sobre ella (por ejemplo su dirección, su estado, etc).
  - Un nodo interno es un **nodo de directorio**, con aristas salientes etiquetadas con un nombre.
  - Todos los nodos tienen identificador.
- Los nodos de directorio contienen una tabla de directorio (o simplemente directorio), con entradas del tipo (nombre de arista, identificador).
- Al nodo que solo tiene aristas salientes y ninguna entrante se le conoce como nodo raíz, o directorio raíz.
- Se emplea la terminología común empleada en los sistemas de directorios de ficheros:
  - Ruta relativa
  - Ruta absoluta

1.- Introducción

1. Terminología
2. Espacios de nombres
3. Resolución de nombres
4. Aspectos de implementación

1.3.- Resolución de nombres

- Al mecanismo que dada una ruta, proporciona una entidad, se le conoce como **resolución de nombres.**
  - La resolución puede proporcionar una entidad ‘hoja’ o un nodo directorio.
  - Es importante conocer desde dónde comenzar la resolución de nombres: dónde está el directorio raíz o el directorio actual.
- También debe considerarse el montaje de directorios (locales o remotos).

1.- Introducción

1. Terminología
2. Espacios de nombres
3. Resolución de nombres
4. Aspectos de implementación

1.4.- Aspectos de implementación

- Un servicio de nombres es un servicio que permite a sus usuarios añadir, eliminar y resolver nombres.
- Los servicios de nombres están implementados por servidores de nombres.
- Si el sistema distribuido al que sirven es grande, se suelen tener varios servidores.

1.4.- Aspectos de implementación

- Distribución de los espacios de nombres: casi siempre de forma jerárquica.
- Se suelen considerad tres niveles:
  - Nivel global: formado por nodos que casi nunca cambian. Suelen representar organizaciones.
  - Nivel administrativo: formado por nodos que cambian poco, pero más que los globales. Suelen representar unidades dentro de organizaciones.
  - Nivel de trabajo: formado por nodos que cambian con frecuencia, muchas veces gestionados por los usuarios finales.
- Conveniente alta disponibilidad del nivel global.
- Conveniente el empleo de caching y replicación.
- Conveniente alta disponibilidad del nivel administrativo dentro de la organización.

1.4.- Aspectos de implementación

Implementación de la resolución de nombres

- Aparte de aspectos de caching y replicación, la resolución se realiza en una serie de etapas:
  - De forma iterativa.
  - De forma recursiva.
    - Más carga a los servidores globales
    - Caching más eficaz.

Tema 4.- Nombrado y localización

1. Introducción
2. Localización de entidades móviles
3. Recolección de residuos

Bibliografía: Capítulo 4 de Tanenbaum

2.- Localización de entidades móviles

1. Introducción
2. Localización por difusiones
3. Localización por punteros hacia delante

2.1.- Introducción

- Los servicios de nombres tradicionales no resultan adecuados para soportar asociaciones nombre-dirección que cambian con frecuencia.
  - Si hay muchas consultas y pocas actualizaciones, replicación y caching eficiente.
  - Si hay muchas actualizaciones, replicación y caching se encarecen.
- Solución: distinguir nombrado de localización.
  - Se introduce un nivel de indirección: nombre-identificador-dirección.
  - El servicio de nombres retorna un identificador
  - El **servicio de localización**, dado el identificador busca su dirección

2.2.- Localización por difusiones

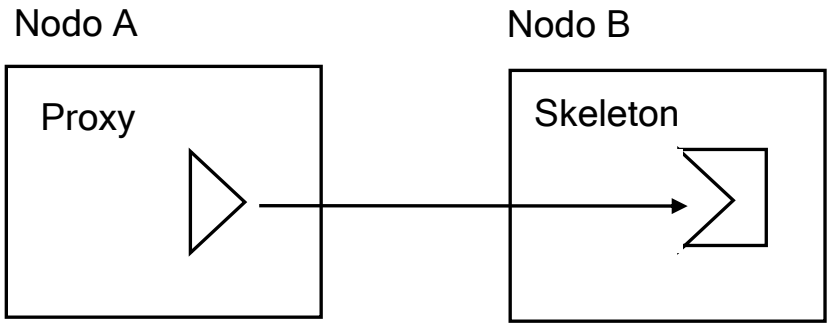
- Adecuado para redes locales conectadas en bus
- Se envía un mensaje de difusión al bus conteniendo el identificador de la entidad
- Sólo contesta el nodo que contenga el punto de entrada.
- Ejemplo: ARP

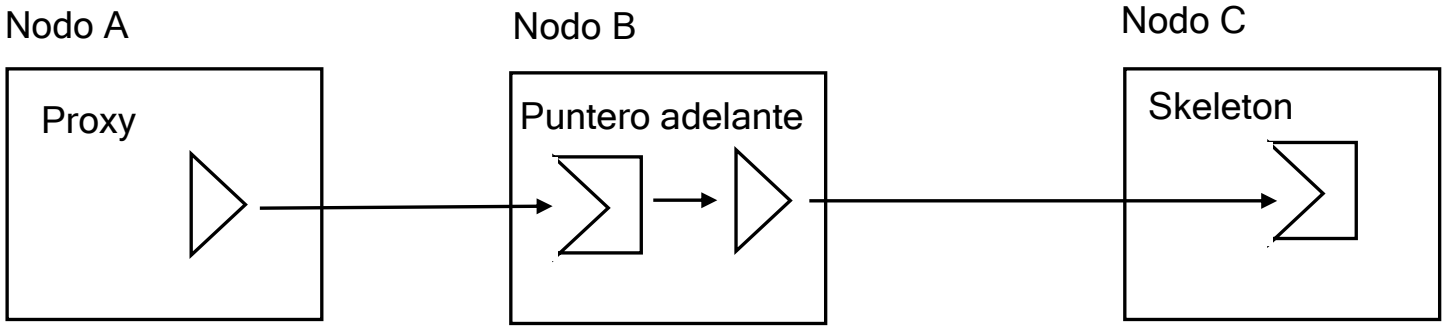
2.3.- Localización por punteros adelante

- Principio: cuando un nodo mueve de A a B, se deja un puntero en A hacia su nueva ubicación.
- Desventajas:
  - Las cadenas de punteros hacia delante pueden crecer mucho.
  - Si un nodo parte de la cadena falla, la entidad queda inaccesible
- Solución: intentar ir acortando las cadenas siempre que sea posible: necesario el empleo de técnicas de recolección de residuos.

2.3.- Localización por punteros adelante

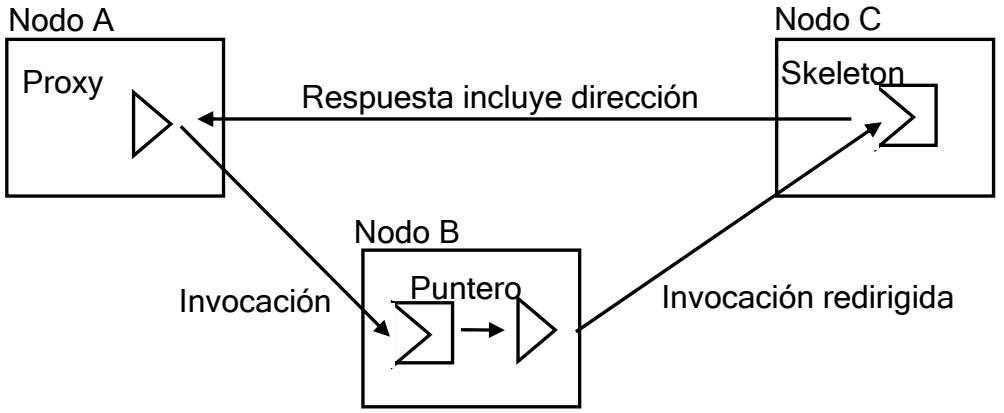
Ejemplo: Punteros adelante en objetos distribuidos. Un objeto migra del nodo B al nodo C. Se crea un proxy en el nodo B que apuntará al nuevo esqueleto.

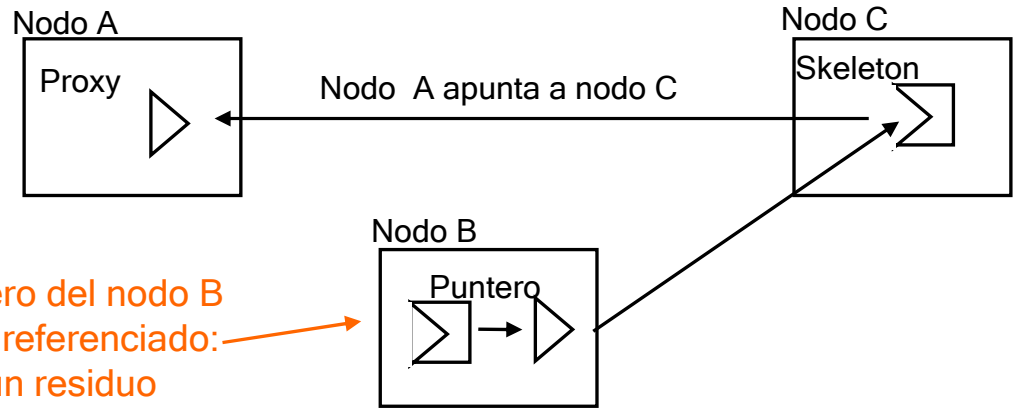




2.3.- Localización por punteros adelante

Ejemplo en objetos distribuidos (cont): Acortamiento de cadenas en invocaciones





Tema 4.- Nombrado y localización

1. Introducción
2. Localización de entidades móviles
3. Recolección de residuos

Bibliografía: Capítulo 4 de Tanenbaum

3.- Recolección de residuos

1. Introducción
2. Cuenta de referencias
3. Listas de referencias
4. Trazas de referencias

3.1.- Introducción

- El nombrado y la localización proporcionan una forma de referenciar entidades.
- Las entidades que no sea posible referenciar, deben eliminarse.
- Si no se eliminan entidades que sea imposible referenciar, el sistema se irá llenando de entidades no referenciadas, que irán consumiendo recursos.
- Saber si una entidad es referenciada es complejo: **algoritmos distribuidos de recolección de residuos**.
- Nos centraremos en recolección de residuos para sistemas de objetos distribuidos: estudiamos algoritmos para averiguar cuándo un objeto está no referenciado.

3.1.- Introducción

El problema de los objetos no referenciados

- Interesa saber cuándo no existen referencias que apunten a un objeto remoto (implementación).
- Si existe un objeto para el que no hay referencias, el objeto es un residuo.
- Además, que existan referencias a un objeto no implica que el objeto sea accesible: **ciclos de referencias o residuos cíclicos:**
  - Es posible que un objeto A tenga una referencia a un objeto B, que el objeto B tenga una referencia al objeto A, pero no existan más referencias que apunten ni a A ni a B.
  - Globalmente A y B son residuos, pues son inaccesibles.
  - Se establece un conjunto “raíz” de objetos que no necesitan ser referenciados y a través de los cuales se accede a los demás objetos.
  - Todo objeto que no sea accesible desde el conjunto raíz, es un residuo.

3.- Recolección de residuos

1. Introducción
2. Cuenta de referencias
3. Listas de referencias
4. Trazas de referencias

3.2.- Cuenta de referencias

Esquema básico

1. Todo objeto mantiene un contador en su skeleton, inicialmente a 0.
2. Cada vez que se crea una referencia al objeto, se incrementa el contador.
3. Cuando un nodo elimina su referencia, se decrementa el contador.
4. Cuando el contador pasa de 1 a cero, el objeto es un residuo.

- Para cada incremento y decremento, hace falta un mensaje que se envía al skeleton.
- Problema si un nodo cliente envía su referencia a otro nodo cliente y justo después elimina su referencia: el decremento puede llegar antes que el incremento!!

3.2.- Cuenta de referencias

Esquema avanzado

- El skeleton y los proxies mantienen un contador, inicialmente a 0. Cuando el contador del skeleton pase de 1 a 0 el objeto es un residuo.
- Casos:
  1. El servidor envía una referencia a otro nodo (nodo cliente)
  2. Un nodo cliente envía una referencia a otro nodo

- **CASO 1**.- El servidor envía una referencia: incrementa el contador del skeleton.
  - Si el nodo receptor ya tenía la referencia: se envía decremento al servidor.
  - Si el nodo receptor no tenía la referencia: no hace nada.

3.2.- Cuenta de referencias

Esquema avanzado (cont)

- **CASO 2**.- Un nodo cliente envía una referencia a otro nodo: incrementa su contador del proxy. Casos:
  1. Si el receptor es el servidor, el servidor envía decremento al cliente emisor.
  2. Si el receptor es un cliente:
     1. Si tenía la referencia, envía decremento al cliente emisor
     2. Si no tenía la referencia:
        1. Incrementa el contador de su proxy
        2. Envía incremento_especial al servidor, incluyendo la dirección del cliente emisor como argumento al mensaje
        3. Cuando el servidor reciba el mensaje, incrementará su contador y enviará sendos mensajes de decremento a los 2 clientes involucrados.

3.2.- Cuenta de referencias

Esquema avanzado (cont)

- Cuando un nodo cliente **elimine** su referencia, sólo se enviará decremento al servidor, si el contador de proxy es 0.
- Si no es 0, se enviará el decremento al servidor, cuando el contador llegue a 0.
- Si no es 0 y antes de que llegue a 0, se recibe una referencia, se actúa como si el nodo ya tuviera la referencia.

3.- Recolección de residuos

1. Introducción
2. Cuenta de referencias
3. Listas de referencias
4. Trazas de referencias

3.3.- Listas de referencias

- Cada skeleton mantiene una lista a las referencias que le apuntan.
- Cada vez que envía una referencia a otro nodo, incluye la dirección del cliente, si no existía.
- Cada vez que un cliente envía una referencia a otro cliente, antes de hacerlo, envía un mensaje al servidor, para que incluya la nueva entrada. (Nótese que se espera reconocimiento)
- Si un nodo elimina su referencia, envía un mensaje al servidor.
  - Ventajas: se toleran fallos si el servidor envía mensajes de ping a los clientes.
  - Desventajas: No escala bien.
- Ejemplo: JAVA RMI.

3.- Recolección de residuos

1. Introducción
2. Cuenta de referencias
3. Listas de referencias
4. Trazas de referencias

3.4.- Trazas de referencias

Esquema básico

- La cuenta de referencias y las listas de referencias no detectan ciclos de residuos.
- Para detectar ciclos de residuos, se comienza la búsqueda desde un conjunto de objetos raíz y se va construyendo el grafo de objetos alcanzables.
- Una vez se haya pasado por todas las referencias, los objetos restantes son residuos.

3.4.- Trazas de referencias

Esquema avanzado

- Todos los nodos tienen un recolector de residuos local a cada nodo, que se ejecutarán en paralelo.
- Los recolectores, marcan los proxies, skeletons y los objetos con uno de tres colores: blanco, gris y negro.

1. Inicialmente se marcan todas las entidades a blanco.
2. Todos los objetos alcanzables desde el conjunto de objetos raíz local a un nodo se marcan con gris.
3. Cuando se marca un objeto gris, se marcan todos los proxies contenidos en él a gris.
4. Cuando se marca un proxy a gris, se envía un mensaje a su skeleton, para marcarlo también a gris.
5. El objeto asociado al skeleton se marca con gris cuando se marque gris el skeleton.
6. De forma recursiva, se procede con los proxies asociados a este objeto, hasta que se llegue a un objeto que no tenga proxies, o cuyos proxies estén a gris.
7. Cuando se termine la exploración. Se marca el objeto a negro y después se marca a negro el skeleton asociado.
8. Cuando se marca negro el skeleton, se envía un mensaje a los proxies que le apuntan para volverlos negros.
9. Cuando estén marcados a negro todos los objetos del conjunto raíz, se habrá terminado.

3.4.- Trazas de referencias

- Esta fase de coloreado se llama **fase de marcado.**
- Una vez termina la fase de coloreado, comienza la **fase de barrido**.
- En la fase de barrido, se eliminan todas las entidades blancas.
- Desventajas:
  - Se debe detener el trasiego de referencias mientras dura la fase de marcado.
  - No escala bien por el número de mensajes necesarios.