Práctica ORB 5. Servidor multiservicio multihilo

Objetivos específicos que persigue la realización del trabajo:

- Transformación de un programa que realiza una llamada a objetos locales de diferentes clases, a otro programa que realiza la llamada de forma transparente a los mismos objetos que residen ahora en un servidor remoto.
- Construcción de un servidor que soporte un número no limitado de instancias de diferentes clases.
- Construcción de un servidor multiservicio.
- Construcción de un servidor multihilo que atienda peticiones simultaneas de varios clientes.

Fecha	Grupo	Tiempo estimado resolución	Tiempo de resolución
		6 h	

Apellidos y nombre	Grupo	Calificación

1. Introducción

El servidor de Agenda implementado en las prácticas anteriores es un servidor multiobjeto multiservicio monohilo, puesto que el servidor atiende una petición cada vez.

En esta práctica se pretende construir un servidor multihilo, de tal forma que sea capaza de servir varias peticiones en paralelo.

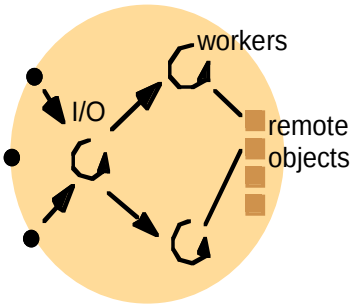
Para ello, cada vez que el servidor acepte una conexión, asignará a un thread la tarea de servir la petición.

El diseño del servidor puede seguir tres patrones, que se explican a continuación.

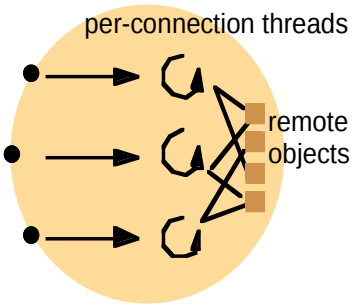
2. Diseño del servidor multihilo

El uso de servidores multihilo permite que éstos maximicen su rendimiento, medido como el número de peticiones procesadas por segundo. El servidor multihilo puede diseñarse de tres formas distintas:

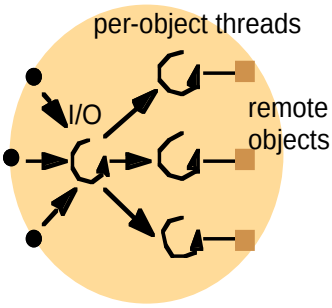
a) Thread por petición
b) Thread por conexión
c) Thread por objeto



a. Thread-per-request



b. Thread-per-connection



c. Thread-per-object

En los modelos anteriores existe un thread de entrada/salida (I/O) que recibe peticiones provenientes de un socket asociado a un puerto (o a un conjunto de sockets asociados a varios puertos).

En el modelo a) un thread por petición, el thread de I/O crea un nuevo thread trabajador cada vez que llega una petición. Un thread trabajador atiende una petición y muere cuando termina de atenderla. La ventaja de este modelo es que el número de threads a crear depende de las peticiones, pero por el contrario, en el tiempo de servicio de la petición hay que incluir el tiempo que se tarda en la creación del thread trabajador.

En el modelo b) un thread por conexión, se asocia un thread trabajador por conexión. Así, el thread de I/O crea un thread trabajador cada vez que el cliente establece una conexión. El thread trabajador atiende todas las peticiones de un cliente y dentro de la misma conexión a cualquier objeto y muere cuando el cliente cierra la conexión.

En el modelo c) un thread por objeto, cada objeto creado tiene asociado un thread que atiende peticiones a dicho objeto. En este modelo, es necesario contar con una cola de peticiones pendientes por objeto, si el thread del objeto recibe nuevas peticiones mientras está sirviendo una. Este tiempo de espera puede suponer que un cliente tenga que esperar a que el thread sirva las peticiones previas de otro cliente.

3. Se pide:

1) Construya el servidor multihilo multiservicio de Agenda y Tiempo eligiendo un modelo de los anteriores. Justifique su elección.
2) Indique cómo va a conseguir generar peticiones simultaneas.
3) Muestre una traza donde se vea que el servidor está atendiendo más de una petición simúltanea.