Grado en Ingeniería Informática

Práctica:

Diseño e implementación de un esquema editor/suscriptor con sockets

Sistema Distribuidos

Autores:
Juan José Garzón Luján (100363861)
Carlos García Cañibano (100363813)

Índice de contenidos

Descripción del código

Tenemos cuatro clases principales que son la de suscriptor, broker y editor y linked_list. Vamos a hablar de como funciona cada una y lo mas importante de ellas para la implementar todas las tres partes de la práctica.

I. Broker

Espera por parámetros el puerto y la dirección rpc a la que se tiene que conectar. Una vez recibido esos los parámetros correctamente, crea un socket para esperar conexiones de editores y subscriptores, y se conecta al servidor de rpc.

Cuando llegue una petición, se extrae la IP y el identificador del socket, con esos datos llamamos a un thread con la función “tratar_peticion” donde copiamos los datos para poder así atender a más peticiones y que no se produzca carrera de datos.

Una vez que tenemos una copia de los datos, leemos cual es tipo de operación a realizar. Si es **“PUBLISH”**:

- ➔ Leemos el tema y texto que envió el editor para realizar las operaciones pertinentes con la lista de los temas y el texto en el servidor rpc.
- ➔ Creamos una estructura tema para almacenar el tema en la lista, si ya existe el tema no se añade otra vez. Además, guardamos en la rpc el tema con su texto.
- ➔ A la hora de modificar la lista tenemos un mutex para así no se produzcan condiciones de carreras por el tratamiento de varias operaciones sobre la lista a la vez.
- ➔ Para poder repartir los temas a los suscriptores sacamos la lista de suscriptores del tema, siempre que haya suscriptores en la lista. Recorreremos esa lista y nos iremos conectando con los suscriptores enviándoles el tema.

Si es **“SUBSCRIBE”**:

- ➔ Leemos el tema al que se subscribe y el puerto donde estará escuchando. Comprobamos que si el tema al que quiere subscribirse y si no está, se crea el tema con el subscriptor en la lista de usuarios del tema.
- ➔ Si ya está creado, solo se añade el nuevo subscriptor a la lista de usuarios del tema. Dependiendo si ya está suscrito o no, se le enviará al suscriptor el 0 si fue todo bien y 1 si no se pudo realizar.
- ➔ Comprobamos en el servidor rpc, si se ha publicado algún tema para reenviárselo al nuevo subscriptor del tema. Si existe el tema, nos conectamos al puerto del suscriptor y le enviamos el tema con su último texto.

Si es **“UNSUBSCRIBE”**:

- ➔ Leemos el tema al que se quiere dar de baja y miramos si esta subscrito o no, si no está subscrito o ha habido algún fallo se le enviará un 1, sino un 0 ya que se ha hecho todo correctamente y se le quitará de la lista de usuarios del tema.

Si es **“QUIT”**:

- ➔ Leeremos el puerto que nos envíe el suscriptor que quiere darse de baja, y procederemos a eliminar a ese usuario de la lista de usuarios de los temas en lo que estaba subscrito.

II. Editor

- ➔ Leemos el host, el puerto, el tema y texto que nos introduce un editor. Todos estos datos son necesarios para la correcta publicación de un tema con su texto.
- ➔ Creamos un socket para poder conectarnos al servidor del broker, una vez conectado, le enviamos el tipo de operación que es “PUBLISH”, el tema y el texto para realizar la publicación.

III. Suscriptor

- ➔ Creamos un socket para conectarnos al broker y llamamos a la consola para esperar las peticiones que puede hacer el suscriptor con el broker.
- ➔ Dentro del shell (consola) imprimimos “c>” para que el suscriptor introduzca la operación que quiere realizar.
- ➔ Hay tres opciones posibles: “SUBSCRIBE”, “UNSUBSCRIBE” y “QUIT”, si no se introduce correctamente esas operaciones o se escriben otros comandos o cualquier otra cosa, se retornará que el comando no es válido.

Tenemos una función para cada posible operación. La de “SUBSCRIBE”:

- ➔ Creamos un thread con un socket servidor para poder leer las publicaciones de los tenemos a que nos subscribimos. Enviamos al broker la acción de subscribirse con el tema y el valor del puerto donde estaremos esperando las publicaciones.
- ➔ El broker nos retornará una respuesta de nuestra acción y así sabremos si hemos podido subscribirnos o no, e imprimiremos un mensaje con “SUBSCRIBE OK” o “SUBSCRIBE FAIL”.

Esta función se apoya en la clase **ClientThread**, que maneja la comunicación entre el broker y el suscriptor a la que se le envían temas.

- ➔ Hacemos un bucle con una variable booleana atómica que está por defecto a true, eso es para recibir todas las publicaciones de los temas a los que estamos subscritos.
- ➔ Dentro de este bucle creamos la URL para todo lo relacionado con el servicio web y creamos un nuevo servicio para esa URL y dentro de el creamos uno para nuestro puerto del subscriptor.
- ➔ Esperamos las conexiones y una vez tengamos una conexión, leemos el tema y el texto que viene separados por un “/”. Formateamos la respuesta con la función format() que hace las operaciones que corresponden para el formato web. Finalmente, imprimimos el tema con el mensaje formateado.
- ➔ Para parar el bucle se nos enviara una interrupción y pondremos la variable del bucle a false.

La función de “UNSUBSCRIBE”:

- ➔ Envía la operación al broker se darse de naja de un tema y para ellos tiene que tener un puerto abierto antes, ya que si no, no se habría subscrito a nada antes.
- ➔ Enviamos el puerto y esperamos la respuesta de broker, dependiendo de ella se imprimirá “UNSUBSCRIBE OK” o “TOPIC NOT SUBSCRIBED”.

La función de “QUIT”:

- ➔ Enviaremos la operación al broker de QUIT y el puerto, cerraremos el socket servidor ya que hemos cerrado sesión y enviamos una interrupción al thread creado del suscriptor para que se pare.

IV. **Linked_list**

Para realizar las operaciones con la lista que utilizamos para almacenar los temas nos hemos apoyado en las clases linked_list. Tenemos dos estructuras, una para el tema que consta de nombre, de lista de usuarios y un puntero al siguiente tema; y otra para el usuario que consta de la ip, puerto y puntero al siguiente usuario. Con estas estructuras podremos creamos listas enlazadas de usuarios y temas.

Las funciones más importantes utilizadas por el bróker son:

Insert_topic: Recibe una lista de tema y el tema a introducir, recorre la lista si no esta vacía y el tema no esta ya en la lista, para insertar el tema al final. Si está vacía, se inserta el tema en la lista. Se utiliza para ir agregando los temas a la lista del broker

Insert_user: Recibe una lista de usuarios y el usuario a insertar. El funcionamiento es igual que el anterior para insertar temas. Se utiliza para ir agregando los subscriptores a los temas y así enviarles los temas.

Insert_user_topic: Recibe una lista de tema, el nombre del tema, y los datos del usuario a insertar. Verifica que el tema exista e inserta en la lista de usuarios el usuario creado con los parámetros recibidos. Se utiliza para insertar los subscriptores en los temas.

Insert_user_notopic: Es igual que la anterior, la diferencia esta en que no existe el tema en la lista y tiene que crearlo e insértalo en ella.

Delete_user_topic: Recibe una lista de tema, el nombre del tema, y los datos del usuario a insertar. Verifica que el tema exista y que el usuario este en la lista de usuarios del tema. Después recorre la lista de usuarios hasta encontrarle y lo elimina.
Se utilizar para cuando se hace cuando un subscriptor cierra la sesión o se dé de baja en algún tema.

Verify_topic: Recibe una lista de temas y el nombre de tema a verificar. Recorre la lista para saber si el tema existe o no. Es apoyo de las funciones como **insert_topic.**

Verify_user: Recibe una lista de usuarios y los datos del usuario a verificar. Recorre la lista para saber si el usuario existe o no. . Es apoyo de las funciones como **insert_user_topic.**

Search_topic: Recibe una lista de temas y el nombre del tema a buscar. Busca en la lista el tema y devuelve un tema con los datos del tema buscado, si no se encuentra, en el nombre del tema estará “NOT_FOUND”-

Quit: Recibe la lista de temas y los datos del usuario. Se va recorriendo los temas de la lista y viendo en cuales esta el usuario para quitarle de la lista de usuario de esos temas. Se utiliza cuando el broker lee la instrucción “QUIT”.

V. RPC

Para todo lo relacionado con la parte de las rpc hemos utilizado sqlite3, es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. Hemos creado las operaciones pertinentes para poder iniciar un servidor y poder almacenar y recuperar el último texto de cada tema. Estas funciones las usa el broker.

VI. Detalles para compilar y obtener todos los ejecutables

El servicio RPC está en su totalidad incluído en el makefile. Solo habría que ejecutar el comando make para generar todos los ejecutables escritos en c. Para ejecutar el servidor RPC simplemente utilizamos ./topic_server

Para generar el servicio web nos aseguramos de estar en la carpeta raíz del proyecto. Después ejecutamos los siguientes comandos:

→ javac format/FormatService.java
→ javac format/FormatServicePublisher.java

Después ejecutamos la publicación del servicio web para que pueda ser utilizado por el suscriptor:
→ java format/FormatServicePublisher

Para ejecutar el broker tener en cuenta que el servicio web se ejecuta en el puerto 2000. Utilizar otro. La ejecución de todos estos servicios es necesaria para poner a funcionar la aplicación.

También cabe destacar que en el código hay bastantes trazas comentadas que podrían ayudar a seguir el flujo de ejecución. No queríamos que los programas saturasen mucho con los mensajes, así que, si se estima oportuno, se pueden utilizar.

VII. Batería de pruebas

Tener en cuenta que las pruebas se han realizado con el broker ejecutando por debajo, así como el servidor RPC y el servicio web de transformación de textos.

Descripción	Entrada	Salida	Conclusión
Como editor, intentamos publicar un tema con una longitud mayor a 128 bytes.	Tema: "prueba…" Texto: “probando”	Max topic name length is 128	Se detecta correctamente el rango máximo del tema.

Como editor, intentamos publicar un texto con una longitud mayor a 1024 bytes.	Tema: “prueba” Texto: “probando...”	Max topic text length is 1024	Se detecta correctamente el rango máximo del texto.
Como editor, intentamos public un tema y un texto dentro de los rangos de longitud permitidos.	Tema: “Basket” Texto: “EEUU finalista del mundial contra FRANCIA con un resultado de 103 a 84 puntos”	Texto insertado correctamente	El tema y el texto son insertados correctamente en la lista enlazada y en la base de datos remota mediante RPC
Como suscriptor, nos suscribimos al tema creado anteriormente “Basket”	SUBSCRIBE Basket	MESSAGE FROM Basket: EEUU finalista del mundial contra FRANCIA con un resultado de 103 (one hundred and three) a 84 (eighty four)	Se recibe correctamente el texto extraido del servicio RPC y formateado con el servicio web de reemplazamiento de números.
Como editor, realizamos una modificación al tema “Basket”. En esta ocasión, tenemos dos suscriptores.	Tema: Basket Texto: Canasta de 3 puntos por parte del numero 23 del equipo americano	Se recibe en los dos suscriptores: Canasta de 3 (three) puntos por parte del numero 23 (twenty three)	Ambos suscriptores reciben correctamente el mensaje. El funcionamiento es el esperado.
El primer suscriptor anula la suscripción al tema “Basket” y se realiza una actualización por parte del editor.	Suscriptor: UNSUBSCRIBE Basket Editor: Tema: Basket Texto: Fin del cuarto tiempo del partido entre los Cavaliers y los Yankees	Solo se recibe en el suscriptor que no hizo UNSUBSCRIBE el mismo mensaje: Fin del cuarto tiempo del partido entre los Cavaliers y los Yankees	Es el funcionamiento esperado. Solo recibe actualizaciones del tema aquel que sigue suscrito.

El primer suscriptor realiza la suscripción a un tema que no ha sido publicado anteriormente con el nombre “patinaje sobre hielo”. Más tarde el editor realiza una publicación sobre ese tema.	Primer suscriptor: SUBSCRIBE patinaje Como editor: Tema: patinaje sobre hielo Texto: El equipo chino hace alarde una técnica excelente	Se recibe por parte del suscriptor el mensaje: El equipo chino hace alarde una técnica excelente	Funcionamiento correcto y esperado. No importa que un tema no haya sido publicado con anterioridad.
El primer suscriptor realiza la operación “QUIT” que debe anular todas sus suscripciones hasta el momento. Para que sean múltiples, le volvermos a suscribir primeramente al tema “Basket”.	SUBSCRIBE Basket QUIT	El estado de la lista que muestra el broker con la operación “show” revela que se ha eliminado al suscriptor de todos sus temas.	El resultado es el esperado. Se elimina correctamente al suscriptor de todos los temas de la lista enlazada del broker.
Probamos a quitarnos la suscripción de algún tema que no nos hayamos suscrito. Por ejemplo “Recetas”.	UNSUBSCRIBE Recetas	TOPIC NOT SUBSCRIBED	Es el resultado esperado. Cuando no estamos suscritos a un tema el broker nos lo indica correctamente.

Cabe destacar que, para poder realizar estas pruebas con las tres partes de la práctica montadas, se ha utilizado el servidor RPC y el servicio web. Esto quiere decir que las tres operaciones de RPC (iniciar el servicio, guardar tema/texto y consultar texto) funcionan correctamente para llevar a cabo las operaciones descritas. Así mismo, los resultados formateados que se muestran en algunas pruebas son la garantía de que también funciona correctamente.

El servicio web de formateo no permite caracteres como el “.”, “,”, “!”, “¡”, etc. No se contemplan en el enunciado restricciones de entrada, así que nos tomamos la libertad de incluirlas aquí.

VIII. **Conclusiones**

Entre los principales problemas encontrados podemos destacar el manejo de envío de mensajes, que se enviara de forma correcta para una lectura correcta. Cabe destacar que la anterior práctica con la lista nos ha facilitado bastante la implementación de la lista de temas y ahorrando bastante trabajo. En la práctica lo realmente difícil es que no sabes como funciona todo lo que hay que implementar, pero una vez que entiendes su funcionamiento no es costoso. El uso de sqlite3 para el servidor rpc, nos facilitó bastante el trabajo en ese apartado.

Personalmente, la práctica nos ha ayudado a comprender en profundidad como funciona los sockets en un esquema de editor suscriptor. Nos ha parecido muy interesante todo lo aprendido, en general, una muy buena práctica para entender todos los conceptos relacionados con esta práctica.