Javier Gómez Martínez
Carlos Li Hu

Memoria Práctica 2 de Redes 2

Introducción: una descripción de lo que se pretende realizar en la práctica

En esta práctica se pretende realizar un cliente IRC que siga el protocolo establecido en los RFC's de forma que sea funcional para los servidores IRC disponibles.

Este cliente debe tener ciertas funcionalidades imprescindibles como la conexión, mediante el protocolo TCP a un servidor IRC.
Así como ciertas funcionalidades básicas del protocolo IRC como permitir al usuario unirse a canales, mandar mensajes tanto a canales como a usuarios, cambiar el modo de los canales y/o de los usuarios, salir de canales, desconectarse del servidor, cambiar el topic de un canal, etc.
Realizar todas estas operaciones deberá ser posible mediante comandos de usuario. Sin embargo, para mejorar la experiencia del usuario, ciertos elementos de la interfaz (como botones etc.) tendrán el mismo propósito que ciertos comandos en particular.
Además de esto, se requiere que la interfaz facilite al usuario el manejo de este cliente, mostrando datos necesarios como la lista de usuarios en un canal, el topic del canal o las conversaciones (ya sean con un usuario por privado o en un canal) separadas por pestañas distintas.

Aparte de todas las funcionalidades comentadas anteriormente, nuestro cliente las expandirá en cierta medida añadiendo un sistema de envío y recepción de ficheros y de grabación, envío y reproducción de audio.

Diseño: una explicación de los módulos de los que se compone el programa, así como de las decisiones que se han tomado (procesos o hilos, sincronización, etc.)

La parte de mostrar una interfaz se realiza mediante el uso de las librerías de metis. Dado que la interfaz bloquea el proceso principal, la mayoría de las operaciones serán realizadas o bien mediante hilos o bien mediante callbacks resultantes de que el usuario interaccione con la interfaz (ya sea pulsando algún botón o escribiendo algo en el cuadro de texto).

Para empezar, el callback de conexión tratará de conectarse correctamente al servidor (especificado en un campo de texto de la ventana de conexión), y, enviando comandos NICK y USER a dicho servidor, también tratará de iniciar sesión.
Una vez esto suceda correctamente, se lanzará un hilo.
Este hilo se encargará de escuchar los comandos y las respuestas del servidor, para, mediante un array de funciones, realizar en el cliente las operaciones correspondientes a cada uno de esos comandos o respuestas.

La idea detrás de este planteamiento es que el hilo principal maneje la interfaz y “atender” a las interacciones con el usuario, enviando mensajes al servidor según el usuario realice operaciones.
Por otra parte, el hilo secundario escucha las respuestas del servidor y es el encargado de efectuar verdaderamente los cambios en la interfaz.
De esta manera, cualquier operación que realice el usuario no surte efecto hasta que el servidor confirme dicha acción, logrando así que todos los clientes tengan una información real, actualizada y que concuerde con la del servidor. Esto es posible gracias a que las conexiones de esta parte son sobre TCP.

Además de esto, hay otro hilo paralelo que envía periódicamente comandos WHO con los canales al servidor, para mantener la información de los mismos actualizada, tal y como hace el cliente IRC xchat. De esta forma, añadimos cierta robustez extra a la información de nuestro cliente.

En cuanto a las funcionalidades extra de envío de ficheros y de audio, las relegamos en otros hilos para no bloquear la interfaz de ninguna forma.

Los módulos que hemos empleado se dividen en:
xchat2
en este módulo se encuentra toda la funcionalidad de los callbacks de la interfaz, junto con algunas funciones extra de manejo de hilos (por ejemplo, el hilo que envía comandos who cada cierto tiempo). Es también el punto de entrada (main) de nuestro cliente.
userCommands
en este módulo se encuentran las funciones que manejan los comandos que el usuario escribe en el cuadro de texto.
basicComandsFromServer
en este módulo se encuentran las funciones que reaccionan a los comandos principales que puede enviar el servidor. Estos son aquellos que en el mensaje vienen representados no como un número sino como el nombre del comando en si. Por ejemplo, JOIN, NICK , etc.
repliesFromServer
En este módulo se encuentran las funciones que reaccionan al resto de comandos del servidor, es decir, aquellos que son representados mediante un código numérico. No hay demasiadas implementadas porque la mayoría consisten solamente en ser impresas en el registro plano. Sin embargo, de añadir funcionalidades a raiz de dichas respuestas, irían en este módulo
userTools
En este módulo se encuentran algunas funciones extra que nos facilitan ciertas operaciones a lo largo del resto del código del cliente.
Audio
Este módulo está diseñado única y exclusivamente para enviar, recibir y reproducir audio. En concreto, en la parte de recibir y reproducir audio se optó por implementar un buffer de escritura y lectura que, gracias a un desfase en un hilo adicional que se encarga de reproducir, mejoraría la experiencia eliminando cortes en la reproducción.

Cabe a destacar que, en la parte de audio y de envio de ficheros, dado que las ips que metis nos proporciona son dinámicas, no nos es posible (en principio) encontrar la manera de enviar audio y/o ficheros a otros clientes. Sin embargo, para simplemente probar estas características, lo hemos hecho contra localhost (y de hecho así está en el código del cliente).

Funcionalidad IRC: a grandes rasgos, qué funciones del protocolo se han implementado

Las funciones del protocolo que se han implementado han principalmente la emisión de comandos al servidor y la recepción de las respuestas del mismo.
Para cada respuesta del servidor, se ha implementado una función que actualiza la interfaz de acuerdo a los cambios que sucedieran en el servidor.
Así que, como tal, no hemos implementado funciones del protocolo más allá de la comunicación con el servidor, dado que la mayor parte de la “lógica” del protocolo tiene lugar en el servidor.

Las funciones que hemos implementado tienen que ver con la interfaz que se le presenta al usuario y cómo le permite interactuar con el resto de usuarios y con el propio servidor mediante el protocolo IRC.

Además de esto, hemos implementado ciertas funcionalidades externas al protocolo como lo son el envío de audio y de ficheros.

Conclusiones técnicas: temas concretos de la asignatura que se han aprendido al realizar la práctica

Realizando esa práctica hemos aprendido la estructura general de cómo se programa una aplicación cliente de un protocolo, así como reforzado nuestros conocimientos sobre el protocolo IRC en concreto.
Además, hemos aprendido cómo codificar un envío de ficheros mediante conexión TCP y un envío de audio, con un protocolo RTP mediante sockets UDP. En concreto, hemos podido comprobar la gran diferencia que hay entre usar un buffer en el envío de audio y no usarlo.

Además de todo esto, hemos reforzado nuestros conocimientos previos sobre hilos, manejo de ficheros y control de sockets (lo cual incluye, por ejemplo, obtener en c la ip a través de un hostname y viceversa).

Conclusiones personales: a qué se ha dedicado más esfuerzo, comentarios generales

La implementación de los callbacks y de los comandos de usuario ha sido lo menos complicado de la práctica.
Sin embargo, la implementación de las funciones que reaccionaban a las respuestas del servidor han sido en general más complejas, debido en muchas ocasiones a pequeñas sutilezas del protocolo, Además, manejar una interfaz gráfica desde c, a pesar de contar con las librerías de metis, en parte ha complicado este último aspecto.
Sin embargo, lo más complicado del cliente en nuestra opinión ha sido el envío de ficheros y en especial el envío de audio, por contener en general mayor complejidad con respecto a las conexiones y, en el caso concreto del envío de audio, tener que implementar un buffer circular de escritura y lectura simultanea para eliminar cortes en la comunicación debido a pérdida de paquetes o a simple retardo en la recepción de paquetes.