REDES DE COMUNICACIONES II

PRÁCTICA 2
Cliente IRC

Jorge Parrilla Llamas
(jorge.parrilla@estudiante.uam.es)
Javier de Marco Tomás
(javier.marco@estudiante.uam.es)

1. INTRODUCCIÓN

La práctica consiste en la implementación de un cliente IRC programado bajo el lenguaje C usando la librería gráfica GTK. Para el desarrollo del cliente se ha seguido la implementación del RFC 2812 y 2813 según se solicitaba como requisito técnico del enunciado de la práctica.

Además, se ha seguido la implementación proporcionada por la librería eps-redes2.

Una vez desarrollada la práctica hemos realizado una serie de pruebas basándonos en una conexión al servidor de Metis mediante el cuál hemos sido capaces de probar todos los comandos implementados en la práctica.

El resultado de la prueba ha sido satisfactorio y todos los comandos funcionan como debería, así como la transferencia de ficheros de un cliente a otro.

2. DISEÑO

La práctica se ha diseñado de una forma parecia a como funcionaba el servidor, utilizando en este caso dos arrays de punteros a función mediante los cuales somos capaces de repartir los distintos comandos que recibimos o enviamos a la función destinada para ello.

Distinguimos dos grandes estructuras:

- **Entradas:** Las denominamos así y se refieren a los mensajes que recibimos desde el servidor. Cada vez que recibimos un mensaje del servidor lo preprocesamos para saber exactamente a qué función corresponde y tras ello lo enviamos a la función encarga de repartir cada comando a su función correspondiente mediante un array de punteros a dichas funciones.

- **Salidas:** Con ellas nos referimos a los comandos que escriben los usuarios en el cliente desarrollado para tal efecto. Gestionamos dichas salidas de la misma forma que las entradas, primero preprocesamos la salida para saber qué tipo de comando es y tras ello lo enviamos a su función de salida mediante un array de punteros a función.

Listado de comandos implementados (cliente a servidor)

```
functions[UNICK]      = &server_out_command_nick;
functions[UJOIN]      = &server_out_command_join;
functions[UNAMES]     = &server_out_command_names;
functions[ULIST]      = &server_out_command_list;
functions[UPART]      = &server_out_command_part;
functions[UMODE]      = &server_out_command_mode;
functions[UKICK]      = &server_out_command_kick;
functions[PRIVMSG]    = &server_out_command_privmsg;
functions[UWHOIS]     = &server_out_command_whois;
functions[UINVITE]    = &server_out_command_invite;
functions[UTOPIC]     = &server_out_command_topic;
functions[UME]        = &server_out_command_me;
functions[UMSG]       = &server_out_command_msg;
functions[UNOTICE]    = &server_out_command_notice;
functions[UIGNORE]    = &server_out_command_ignore;
functions[UWHO]       = &server_out_command_who;
functions[UWHOWAS]    = &server_out_command_whowas;
functions[UMOTD]      = &server_out_command_motd;
functions[UAWAY]      = &server_out_command_away;
functions[UPING]      = &server_out_command_ping;
```

Listado de comandos implementados (servidor a cliente)

```
functions[[NICK]              = &server_in_command_nick;
functions[[JOIN]              = &server_in_command_join;
functions[[PART]              = &server_in_command_part;
functions[[MODE]              = &server_in_command_mode;
functions[[TOPIC]             = &server_in_command_topic;
functions[[KICK]              = &server_in_command_kick;
functions[[PRIVMSG]           = &server_in_command_privmsg;
functions[[PING]              = &server_in_command_ping;
functions[[PONG]              = &server_in_command_pong;
```

Listado de respuestas del servidor implementadas

```
functions[[RPL_WELCOME]       = &server_in_command_rpl_welcome;
functions[[RPL_CREATED]       = &server_in_command_rpl_created;
functions[[RPL_YOURHOST]      = &server_in_command_rpl_yourhost;
functions[[RPL_LUSERCLIENT]   = &server_in_command_rpl_luserclient;
functions[[RPL_LUSERME]       = &server_in_command_rpl_luserme;
functions[[RPL_MOTDSTART]     = &server_in_command_rpl_motdstart;
functions[[RPL_MOTD]          = &server_in_command_rpl_motd;
functions[[RPL_ENDOFMOTD]     = &server_in_command_rpl_endofmotd;
functions[[RPL_WHOREPLY]      = &server_in_command_rpl_whoreply;
functions[[RPL_AWAY]          = &server_in_command_rpl_away;
functions[[RPL_NOWAWAY]       = &server_in_command_rpl_nowaway;
functions[[RPL_TOPIC]         = &server_in_command_rpl_topic;
functions[[RPL_NOTOPIC]       = &server_in_command_rpl_notopic;
functions[[RPL_YOUREOPER]     = &server_in_command_rpl_youroper;
functions[[RPL_LUSEROP]       = &server_in_command_rpl_luserop;
functions[[RPL_LUSERCHANNELS]= &server_in_command_rpl_luserchannels;
functions[[RPL_YOURESERVICE] = &server_in_command_rpl_youreservice;
functions[[RPL_MYINFO]        = &server_in_command_rpl_myinfo;
functions[[RPL_ENDOFWHO]      = &server_in_command_rpl_endofwho;
functions[[RPL_ENDOFWHOIS]    = &server_in_command_rpl_endofwhois;
functions[[RPL_INFO]          = &server_in_command_rpl_info;
functions[[RPL_WHOISUSER]     = &server_in_command_rpl_whoisuser;
functions[[RPL_WHOISCHANNELS]= &server_in_command_rpl_whoischannels;
functions[[RPL_WHOISOPERATOR]= &server_in_command_rpl_whoisoperator;
functions[[RPL_WHOISSERVER]   = &server_in_command_rpl_whoisserver;
functions[[RPL_WHOISIDLE]     = &server_in_command_rpl_whoisidle;
functions[[RPL_CHANNELMODEIS]= &server_in_command_rpl_channelmodeis;
functions[[RPL_ENDOFNAMES]    = &server_in_command_rpl_endofnames;
functions[[RPL_LIST]          = &server_in_command_rpl_list;
functions[[RPL_LISTEND]       = &server_in_command_rpl_listend;
functions[[RPL_NAMREPLY]      = &server_in_command_rpl_namreply;
```

Listado de errores del servidor implementados

```
functions[[ERR_CANNOTSENDTOCHAN]       = &server_in_command_err_cannotsendtochan;
functions[[ERR_ALREADYREGISTRED]       = &server_in_command_err_alreadyregistred;
functions[[ERR_NONICKNAMEGIVEN]        = &server_in_command_err_nonicknamegiven;
functions[[ERR_ERRONEUSNICKNAME]       = &server_in_command_err_erroneusnickname;
functions[[ERR_NICKNAMEINUSE]          = &server_in_command_err_nicknameinuse;
functions[[ERR_NICKCOLLISION]          = &server_in_command_err_nickcollision;
functions[[ERR_UNAVAILRESOURCE]        = &server_in_command_err_unavailresource;
functions[[ERR_RESTRICTED]             = &server_in_command_err_restricted;
functions[[ERR_PASSWDMISMATCH]         = &server_in_command_err_passwdmismatch;
functions[[ERR_BANNEDFROMCHAN]         = &server_in_command_err_bannedfromchan;
functions[[ERR_CHANNELISFULL]          = &server_in_command_err_channelisfull;
functions[[ERR_CHANOPRIVSNEEDED]       = &server_in_command_err_chanoprivsneeded;
functions[[ERR_INVITEONLYCHAN]         = &server_in_command_err_inviteonlychan;
functions[[ERR_NOCHANMODES]            = &server_in_command_err_nochanmodes;
functions[[ERR_NOSUCHCHANNEL]          = &server_in_command_err_nosuchchannel;
functions[[ERR_UNKNOWNMODE]            = &server_in_command_err_unknownmode;
functions[[ERR_NOMOTD]                 = &server_in_command_err_nomotd;
functions[[ERR_NOSUCHNICK]             = &server_in_command_err_nosuchnick;
```

Procolo PING-PONG

Hemos imitado el funcionamiento de un cliente real de IRC mediante el uso de este protocolo. El funcionamiento del PING-PONG es básico y necesario para el buen desarrollo de una sesión de chat mediante IRC, para ello el cliente realiza una comprobación de PING cada 30 segundos.

Mediante un hilo del cliente enviamos cada 30 segundos un PING al servidor, el cliente espera recibir PONG en un periodo de tiempo determinado.

En caso de que el cliente no reciba el PONG de parte del servidor se procede a desconectar la sesión actual ya que se interpreta que el servidor ha dejado de estar disponible.

Procolo de llegadas

Es importante destacar que hemos tenido especial consideración con las llegadas del servidor al cliente, para ello hemos diseñado un hilo específico y exclusivo para la recepción de dichas llegadas a través del socket.

Los mensajes son recibidos y preprocesados, una vez que el mensaje se considera íntegro es enviado a la función de

preprocesado de las entradas del servidor (detallado más arriba).

Envío de ficheros

Este punto de uno de los más importantes para el cliente ya que nos permite la transferencia de ficheros entre los distintos clientes del canal IRC. Funciona de forma sencilla pero intensa, es decir, se requiere el uso de un socket entre los dos clientes de forma directa ya que como se menciona en el enunciado de la práctica se debe implementar de esta manera para evitar costes excesivos en el servidor.

Por ello, cuando un cliente inicia la transferencia de un fichero a otro cliente se crea un socket único para la petición de transferencia del fichero.

Supongamos que un usuario A desea enviar un fichero a un usuario B, podemos resumir el protocol de transferencia de ficheros de la siguiente forma:

(Continúa en la siguiente página).

1) El usuario A envía la petición de transferencia al usuario B.

2) El usuario B recibe la petición de transferencia del fichero y responde con una confirmación o rechazo a dicha transferencia.

3) El usuario A recibe la confirmación/rechazo de la petición de transferecia.

4) En caso de que se haya confirmado, el usuario A crea un nuevo socket con el usuario B únicamente destinado a la transferencia del fichero. En caso de rechazo, se cierra el socket generado previamente para la confirmación y se da por finalizada la transferencia.

5) El usuario B recibe los datos del fichero del usuario A.

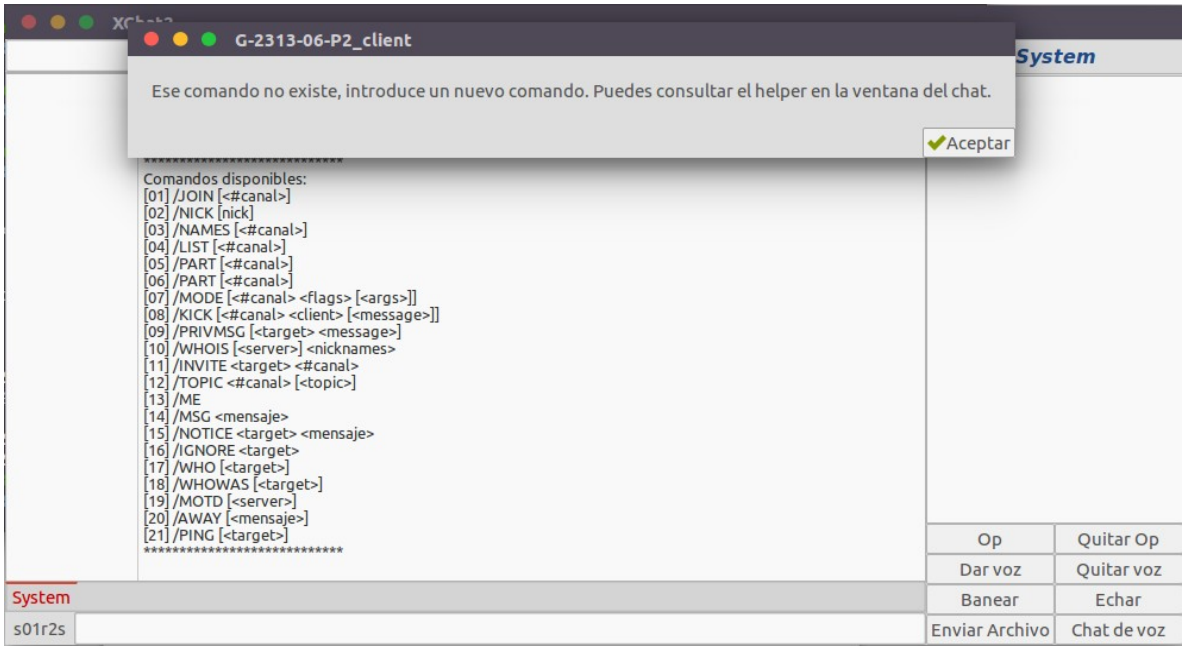
Mensaje de ayuda

En caso de que el usuario introduzca un comando incorrecto en la ventana del chat se le mostrará de forma automática un menú contextual de ayuda en el cuál se le mostrará un listado de los comandos existentes.

Es importante controlar los comandos ya que en caso de que sea un usuario nuevo podría no saber cómo funciona IRC, por ello hemos creído que era necesario incorporar dicha

herramienta a pesar de que no se pide en el enunciado de la práctica.

Se muestra un ejemplo a continuación:



(Más información en la web de documentación detallada)

G-2313-06-P3/doc/html/index.html

3. CONCLUSIONES TÉCNICAS

Ha sido bastante complicado lidiar con los problemas de los sockets en la transferencia de ficheros ya que es complicado que todo funcione como debería y que cada usuario espere la respuesta que debe recibir para ejecutar el siguiente paso.

Es importante destacar el uso varios arrays de punteros a función.

Nota: ***Es posible que el cliente genere violaciones de segmento aleatorias (no es problema de nuestra codificación), dichas violaciones se deben a bugs en la librería facilitada para el desarrollo de la práctica, así como de GTK.***

4. CONCLUSIONES PERSONALES

Ha sido una práctica más larga de lo que aparentaba pero realmente nos ha gustado ya que hemos conseguido mejorar nuestra forma de trabajar con los sockets y el envío de ficheros.