INICIACIÓN A LA INVESTIGACIÓN
REVISTA ELECTRÓNICA
UNIVERSIDAD DE JAÉN

*Ini Inv*, e5: a9 (2013)

# Diseño de aplicaciones cliente/servidor para el aprendizaje de las tecnologías de comunicación

Amelia Zafra, Eva Gibaja, María Luque, Sebastián Ventura

*Departamento de Informática y Análisis Numérico. Universidad de Córdoba.*
*Campus Universitario de Rabanales s/n, 14071, Córdoba.*
azafra@uco.es

**Resumen**

Las redes de comunicación se han convertido en una materia indispensable para los alumnos del grado de Ingeniería Informática, su rápido avance y su introducción en todos lo ámbitos de nuestras vidas han contribuido a su importancia. Sin embargo, su enseñanza es compleja debido a la gran diversidad de tecnologías, modelos y herramientas implicadas. En este contexto, es interesante empezar a abordarlo con un alto nivel de abstracción para posteriormente llegar a un nivel más específico. Una tecnología básica que permite profundizar en conceptos específicos como son los conceptos de modularización, servicios, protocolos e interfaces, serían los sockets. Desde esta perspectiva, se presentan ejemplos prácticos que permiten que el alumno trabaje con esta tecnología y desarrolle aplicaciones cliente/servidor en red que son accesibles desde cualquier localización.

## 1. INTRODUCCIÓN

Las redes de comunicación han cambiado nuestras vidas, sus usos tanto en el ámbito educativo, como en el profesional, como en el hogar han afectado a nuestra forma de comunicarnos. Vivimos en una era conectada, que nos permite estar permanentemente conectados e informados de los últimos acontecimientos. En este contexto, el estudio de estos sistemas para un estudiante del grado de Ingeniería Informática, se ha convertido en un aspecto clave, considerando competencias esenciales que debe adquirir cuando finalice sus estudios.

En la actualidad, el conjunto de protocolos de comunicación de computadoras de mayor difusión mundial, lo constituye TCP/IP. Sobre él funcionan la mayoría de las aplicaciones de red, como servidores web, correo electrónico, sistemas de transferencia de archivos, entre otros. La arquitectura TCP/IP está formada por cuatro capas perfectamente distinguibles: capa de acceso a la red, capa de red, capa de transporte y capa de aplicación. Es fundamental que el alumno aprenda el funcionamiento de estas capas, conociendo los protocolos, servicios e interfaces que presenta.

En este trabajo se presenta un recurso para abordar la tarea de que los alumnos diseñen su propia aplicación en red utilizando algunas de las interfaces disponibles de programación que existen para tal fin. Concretamente, el recurso se centra en la propuesta de aplicaciones de red usando el modelo cliente/servidor mediante el uso de la interfaz sockets de UNIX. Con el diseño de este tipo de aplicaciones los alumnos fortalecen conceptos como la transparencia entre las capas, las definiciones de interfaces, servicios y protocolos, la importancia de una

especificación detallada que permita que los protocolos diseñados puedan ser implementados sin ambigüedades. Concretamente, con el diseño de esta práctica se pretende:

- Explicar las nociones de programación con sockets. Los sockets son la base de las comunicaciones entre ordenadores y suponen el punto de enlace entre nuestra aplicación e Internet. Por lo tanto es imprescindible conocer los cimientos de las comunicaciones entre ordenadores para poder hacer uso de ellos.

- Obtener una visión clara de las características y posibilidades de la comunicación entre procesos remotos basada en el modelo cliente/servidor, utilizando los servicios del nivel de transporte de la arquitectura TCP/IP, mediante la programación de aplicaciones clientes/servidoras basadas en sockets.

En este artículo se dará una introducción a los sockets y se plantearan dos propuestas para realizar aplicaciones de red describiendo sus fundamentos y planificación.

## 2. LOS SOCKETS COMO TECNOLOGÍA DE COMUNICACIÓN

Los mecanismos tradicionales de comunicación entre procesos en UNIX (semáforos, colas de mensajes, memoria compartida y “pipes”) permiten que procesos independientes cooperen, se excluyan mutuamente de secciones criticas e intercambien datos, conceptos que los estudiantes estudian en la asignatura de Sistemas Operativos. A pesar de su utilidad para comunicar procesos que residen en la misma máquina, su extensión a un entorno en red es complejo e introduce una sintaxis diferente a la que se utiliza en otras operaciones habituales como, por ejemplo, el manejo de ficheros. La utilización de sockets permite superar estas limitaciones proporcionando una forma homogénea de enfocar la comunicación entre procesos, tanto en una maquina aislada como en un entorno de red.

De este modo los sockets [1] se sitúan como una tecnología de red altamente extendida y que suele presentarse muy integrada con el sistema operativo, aportando la base para tecnologías de nivel superior. En el entorno docente, las prácticas realizadas con sockets están muy generalizadas ya que establecen los fundamentos de la comunicación y permiten el entendimiento de sus procesos más básicos.

En el dominio internet (TCP/IP) los sockets permiten acceder a los servicios del nivel de transporte tanto a través de conexiones (TCP) como en modo sin conexión (UDP). La forma habitual de implantar una aplicación basada en sockets es utilizar el modelo cliente-servidor. La comunicación es asimétrica, con un cliente que solicita servicios y un servidor que los proporciona bajo demanda. Ambos procesos (cliente y servidor) se ejecutan de forma asíncrona, siendo los instantes de comunicación los puntos de sincronización en la evolución de ambos.

## 3. CONTEXTO DE LA ASIGNATURA

Los objetivos generales de la asignatura de Redes consisten en proporcionar conocimiento concreto al tiempo que una visión global e integradora sobre los protocolos, configuraciones y estrategias que dan soporte a las aplicaciones que deben ejecutarse sobre redes de computadores. Se trata de una asignatura común de tercer curso del grado de Ingeniería Informática y que consta de 6 créditos (3 créditos teóricos y 3 prácticos).

Algunas de las competencias principales que se considera en la asignatura y que quedarían cubiertos con esta práctica sería:

- Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes.
- Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería.
- Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
- Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet y diseñar e implementar aplicaciones basadas en ellas.

## 4. DESCRIPCIÓN DEL RECURSO PROPUESTO

Para que el alumno conozca las principales tecnologías y paradigmas del diseño de aplicaciones en red usando el modelo de capas, se propone resolver la aplicación mediante una arquitectura cliente-servidor que será implementada con sockets utilizando un servicio concurrente y un protocolo de transporte basado en conexión. De este modo se trabaja a nivel de capa de aplicación siguiendo el modelo TCP/IP [2].

Concretamente, se plantean dos propuestas, una de ellas para el diseño del juego de la ruleta en red y otra el diseño de un sistema IRC (Internet Relay Chat).

Según estas propuestas un cliente dispondrá de una aplicación de usuario mediante la que podrá hacer uso de la aplicación concreta que se desarrolle y la aplicación servidora se localizar´a en una maquina servidora a la que tendrán acceso los diferentes clientes. En las siguientes secciones estudiaremos la definición de dichas prácticas, así como los fundamentos, objetivos y planificación en las que se basan.

### 4.1. FUNDAMENTOS DE LA PRÁCTICA

La programación de aplicaciones sobre TCP/IP se basa en el llamado modelo cliente-servidor. Básicamente, la idea consiste en que al indicar un intercambio de información, una de las partes debe iniciar el dialogo (cliente) mientras que la otra debe estar indefinidamente preparada a recibir peticiones de establecimiento de dicho dialogo (servidor).

Cada vez que un usuario cliente desee entablar un dialogo, primero deberá contactar con el servidor, mandar una petición y posteriormente esperar la respuesta.

Los servidores pueden clasificarse atendiendo a si están diseñados para admitir múltiples conexiones simultaneas (servidores concurrentes), por oposición a los servidores que admiten una sola conexión por aplicación ejecutada (llamados iterativos).

Otro concepto importante es la determinación del tipo de interacción entre el cliente y el servidor, que puede ser orientada a conexión o no orientada a conexión. Estas corresponden, respectivamente, con los dos protocolos característicos de la capa de transporte: TCP y UDP. TCP (orientado a conexión)

garantiza toda la fiabilidad requerida para que la transmisión esté libre de errores: verifica que todos los datos se reciben, automáticamente retransmite aquellos que no fueron recibidos, garantiza que no hay errores de transmisión y además, numera los datos para garantizar que se reciben en el mismo orden con el que fueron transmitidos. Igualmente, elimina los datos que por algún motivo aparecen repetidos, realiza un control de flujo para evitar que el emisor envíe más rápido de lo que el receptor puede consumir y, finalmente, informa a las aplicaciones (tanto cliente como al servidor) si los niveles inferiores de red no pueden entablar la conexión. Por el contrario, UDP (no orientada a conexión) no introduce ningún procedimiento que garantice la seguridad de los datos transmitidos, siendo en este caso responsabilidad de las aplicaciones la realización de los procedimientos necesarios para subsanar cualquier tipo de error.

La práctica propone la utilización de sockets TCP con un servidor concurrente y para su realización se requieren conocimientos básicos sobre sockets y de su interfaz con el protocolo TCP, y dependiendo del lenguaje de programación elegido, también serán necesarios conocimientos adicionales sobre gestión y comunicación entre procesos, hebras, manejadores de señales y temporizadores. En concreto, nuestro enfoque propone el uso de librerías de sockets bajo el sistema operativo Linux [3] mediante C estándar o C++. Esto permite al alumno adquirir los conocimientos necesarios para La comprensión de la tecnología básica.

En todos los casos se aportará al alumno unos documentos de especificación donde se describirá detalladamente los protocolos de aplicación que se implementaran. Con ello se logrará que el alumno:

- Tome conciencia del nivel de detalle que debe tener la especificación determinando en cada caso el tipo y orden de los datos a transmitir.
- Se familiarice con el desarrollo de aplicaciones complejas, donde ceñirse a la especificación planteada es vital para un correcto trabajo. De hecho el alumno será consciente de que un cliente/servidor desarrollado por ´el deberá poder ser sustituido por el de otro compañero, sin que esto afecte al funcionamiento global de la aplicación.
- Consolide sus conocimientos sobre conceptos relacionados con otras asignaturas previas, con temas de concurrencia utilizados en sistemas operativos, de programación utilizados en las asignaturas relativas a programación y por supuesto los conceptos estudiados en Redes, de manera que sirva de hilo conductor sobre el que contemplar el resto de conocimientos asociados a la asignatura.

### 4.2. OBJETIVOS

El objeto de esta práctica se puede resumir en:

- Conocer las funciones básicas para trabajar con sockets y el procedimiento a seguir para conectar dos procesos mediante un socket.
- Comprender el funcionamiento de un servicio orientado a conexión y confiable del envío de paquetes entre una aplicación cliente y otra servidora utilizando sockets.
- Ilustrar los conceptos vistos en las clases de teoría sobre el nivel de control de transporte en Internet: TCP.
- Comprender las características o aspectos claves de un protocolo:
    - Sintaxis: formato de los paquetes
    - Semántica: definiciones de cada uno de los tipos de paquetes

### 4.3. PLANIFICACIÓN

Para el caso de la asignatura de Redes se dispone de un total de cuatro sesiones de clases prácticas de dos horas de duración cada una.

### 4.4. PROPUESTA 1. JUEGO DE LA RULETA

Diseño e implementación del juego de la ruleta permitiendo jugar de manera individual o bien en grupos. La finalidad del juego es averiguar la frase o palabra secreta que se muestra en cada partida.

La comunicación entre los clientes del juego de la ruleta se realizará bajo el protocolo de transporte TCP y será un servicio concurrente. La práctica que se propone consiste en la realización de una aplicación cliente/servidor que implemente el juego de la ruleta mediante el cual, los usuarios del juego (los clientes) se conectan al servicio (el servidor) y pueden jugar una partida, permitiendo o bien jugar una partida de forma individual o bien competir en el juego con otros usuarios. Todas las partidas tanto individuales como colectivas versarán sobre la categoría de refranes.

**1) Juego Individual:** La primera variante del juego nos permite jugar una partida al juego de la ruleta de forma individual. Nos conectaremos al servidor y nos permitirá iniciar una partida.

El procedimiento que se seguirá será el siguiente:

• Un cliente se conecta al servicio y si la conexión ha sido correcta el sistema devuelve +0k. Usuario conectado.

• Para poder acceder a los servicios es necesario identificarse mediante el envío del usuario y clave para que el sistema lo valide . Los mensajes que deben indicarse son: USUARIO usuario para indicar el usuario, tras el cual el servidor enviará +Ok. Usuario correcto o -Err. Usuario incorrecto. En caso de ser correcto el siguiente mensaje que se espera recibir es PASSWORD password, donde el servidor responderá con el mensaje de +Ok. Usuario validado o -Err. Error en la validación.

• Un usuario nuevo podrá registrarse mediante el mensaje REGISTRO -u usuario -p password. Se llevará un control para evitar colisiones con los nombre de usuarios ya existentes.

• Una vez conectado y validado, el cliente podrá llevar a cabo una partida individual en el juego indicando un mensaje de PARTIDA-INDIVIDUAL . Recibido este mensaje en el servidor, ´este se encargará de mandarle una frase con la que iniciar el juego indicando mediante el símbolo "-" las posiciones que deben ser rellenadas con consonante o vocal y el símbolo espacio en blanco " " para indicar el espacio en blanco entre las palabras que componen la frase. No se tendrán en cuenta en la frase los signos de puntuación como elemento a adivinar por el jugador. En este punto son admitidos tres tipos de mensajes que se puede enviar al servidor

  o CONSONANTE letra, donde letra indica una consonante que se piensa que puede estar en la frase.

  o VOCAL letra, donde letra indica una vocal que se piensa que puede estar en la frase.

  o RESOLVER frase, donde frase representará una cadena que contiene el refrán que queremos resolver.

El contenido de la cadena debe ser independiente de si se trata de mayúsculas o minúsculas.

• Cada vez que un cliente mande un mensaje de CONSONANTE/VOCAL al servidor, éste deberá devolverle +Ok. Existe la consonante seguido por la frase con la información que se lleva de momento si la letra está contenida en la frase a resolver o +Ok. No existe la consonante seguido por la frase con la información que se lleva de momento en caso de que la letra no exista en la palabra a averiguar. En caso de error en la recepción del mensaje, debido a que se trate de una consonante o vocal inválida, se procederá con un mensaje -Err. Mensaje no válido.

• Se llevará la cuenta del número de intentos que ha realizado el usuario para resolver la frase, para que finalmente al terminar la partida se le asigne una puntuación a cada usuario. La puntuación será fija y tendrá establecido los siguientes valores, si se ha resuelto en menos de 5 intentos tendrá una valoración de 150 puntos, si lo consigue entre 5 y 8 intentos se le asignará 100 puntos, entre 9 y 11 será 70 puntos, entre 12 y 15 será 50 puntos y todo lo que sea más de 15 intentos tendrá una puntuación de 0 puntos.

• Simplemente se podrá enviar un mensaje RESOLVER, en caso de acertarse o fallarse se termina la partida y el usuario simplemente podrá optar por salir del sistema o realizar otra partida:

o Si se acierta la palabra se le mandará al cliente un mensaje +Ok. Enhorabuena, mostrándole una estadística de las frases acertadas y falladas y su puntuación desde que se registró por primera vez.

o Si se falla la palabra, se le mandará al cliente un mensaje +Ok. Le deseamos mejor suerte la próxima vez, mostrándole una estadística de las frases acertadas y falladas y su puntuación desde que se registró por primera vez.

o Un mensaje de un cliente en la que se envía la petición SALIR, hará que el servidor le dé de baja del sistema. Este mensaje puede ser enviado en cualquier momento, incluso en una partida empezada.

• Cualquier mensaje que no use uno de los especificadores detallados, generará un mensaje de -Err por parte del servidor.

**2) Juego Colectivo:** La segunda variante del juego nos permite jugar con varias personas, concretamente el juego estará formado por grupo de tres personas.

El procedimiento que se seguirá será el siguiente:

• Un cliente se conecta al servicio y si la conexión ha sido correcta el sistema devuelve +0k. Usuario conectado.

• Para poder acceder a los servicios es necesario identificarse mediante el envío del usuario y clave para que el sistema lo valide . Los mensajes que deben indicarse son: USUARIO usuario para indicar el usuario, tras el cual el servidor enviará +Ok. Usuario correcto o -Err. Usuario incorrecto. En caso de ser correcto el siguiente mensaje que se espera recibir es PASSWORD password, donde el servidor responderá con el mensaje de +Ok. Usuario validado o -Err. Error en la validación.

• Un usuario nuevo podrá registrarse mediante el mensaje REGISTRO -u usuario -p password. Se llevará un control para evitar colisiones con los nombre de usuarios ya existentes.

• Una vez conectado y validado, el cliente podrá llevar a cabo una partida en el juego indicando un mensaje de PARTIDA-GRUPO. Recibido este mensaje en el servidor, este se encargará de comprobar las personas que tiene pendiente para comenzar una partida:

- Si con esta petición, ya se forma un grupo de tres personas. Se le deja un mensaje a los tres usuarios indicando que va a comenzar la partida +Ok. Empieza la partida y seguido se indica la frase con la que iniciar la partida. Así, como información indicando de quién es el turno.

- Si todavía falta alguna persona para iniciar la partida, mandará un mensaje al nuevo usuario, especificando que tiene su petición y que está a la espera de la conexión de otros jugadores. El usuario en este caso deberá quedarse a la espera de recibir un mensaje para jugar o también se le permitirá salir de la aplicación.

• La representación de la frase indicará todas las posiciones que representan una consonante o vocal mediante el símbolo "-" y el espacio en blanco " ", se utilizará para representar la separación entre las palabras de la frase. No se tendrán en cuenta en la frase los signos de puntuación como elemento a adivinar por el jugador. El orden de los usuarios en el juego será el mismo orden en el que se ha realizado la conexión (aunque puede implementarse otro criterio si lo veis más conveniente).

• Cada usuario podrá ir mandando mensajes de CONSONANTE/VOCAL letra, mientras la frase contenga la letra que va especificando. Esta información se va mostrando a cada uno de los usuarios, al igual que los mensajes que envía el usuario que está jugando en cada momento. Cuando falle, se pasará a otro usuario, así hasta que un usuario escoja la opción de RESOLVER la frase. En este punto son admitidos tres tipos de mensajes que se puede enviar al servidor:

- CONSONANTE letra, donde letra indica una consonante que se piensa que puede estar en la frase. La puntuación que se obtiene, sería 50 puntos por cada vez que la consonante aparezca en la frase a resolver. Si en la frase desconocida no aparece la consonante, se pasará el turno al siguiente jugador.

- VOCAL letra, para poder mandar este mensaje necesitas tener al menos 50 puntos, que se restarán por cada vocal que solicites, con independencia del número de veces que aparezca.

- RESOLVER frase, donde frase representará una cadena que contiene el refrán que queremos resolver.

• Se debe tener cuidado en el servidor cuando lleguen peticiones de usuarios cuando no sea su turno de jugar. En caso de que envíen un mensaje cuando no es su turno, deberá responder que todavía no es turno y deben esperar. La especificación del mensaje que se enviaría sería: -Err. Debe esperar su turno.

• En caso de que un usuario envía la petición RESOLVER y ésta no sea la frase correcta, la partida termina, el usuario que ha fallado queda automáticamente con una puntuación de cero y ganaría el usuario que tuviese hasta ese momento una mayor puntuación. Se enviará un mensaje a cada uno de los jugadores, para indicar puntuaciones y ganador.

• Para salir del servicio se usará el mensaje SALIR, de este modo el servidor lo eliminará de clientes conectados y el juego continuará entre los restantes participantes. Este hecho se le debe de comunicar al resto de usuarios. Cuando quede un único jugador podrá terminar el panel o salir del sistema.

• Cualquier mensaje que no use uno de los especificadores detallados, generará un mensaje de -Err por parte del servidor.

Algunas restricciones a tener en cuenta son:

• La comunicación será mediante consola.

• El cliente deberá aceptar como argumento una dirección IP que será la dirección del servidor al que se conectará.

• El protocolo deberá permitir mandar mensajes de tamaño arbitrario. Teniendo como tamaño máximo de envío/recepción de una cadena es 200 caracteres.

• El servidor aceptará servicios en el puerto 2050.

• El servidor debe permitir la conexión de varios clientes simultáneamente. Se utilizará la función select() para permitir esta posibilidad.

• El número máximo de clientes conectados será de 50 usuarios.

• Todos los mensajes devueltos por el servidor que se necesiten contemplar, seguirán el criterio de ir precedidos por +Ok. Texto informativo, si la petición se ha podido aceptar sin problemas y -Err. Texto informativo, si ha habido algún tipo de error.

• Las frases que se van a resolver no tendrán en cuenta los signos de puntuación. En caso de que la frase cuente con alguno de ellos, aparecerá directamente como elemento visible desde el principio en la frase que se le muestra al usuario.

**3) Resumen de paquetes a utilizar:** Considerando la práctica completa, vamos a considerar los siguientes tipos de mensajes con el siguiente formato cada uno:

• USUARIO usuario: mensaje para introducir el usuario que desea.

• PASSWORD contraseña: mensaje para introducir la contraseña asociada al usuario.

• REGISTRO -u usuario -p password: mensaje mediante el cual el usuario solicita registrarse para acceder al juego de la ruleta que escucha en el puerto TCP 2050.

• PARTIDA-INDIVIDUAL: mensaje para iniciar un juego individual.

• PARTIDA-GRUPO: mensaje para iniciar una partida en grupo.

• CONSONANTE letra, donde letra indica una consonante que se piensa que puede estar en la frase.

• VOCAL letra, para poder mandar este mensaje necesitas tener al menos 50 puntos, que se restarán por cada vocal que solicites, con independencia del número de veces que aparezca.

• RESOLVER frase, donde frase representará una cadena que contiene el refrán que queremos resolver.

• SALIR: mensaje para solicitar salir del juego.

• Cualquier otra tipo de mensaje que se envía al servidor, no será reconocida por el protocolo como un mensaje válido y generará su correspondiente -Err. por parte del servidor.

### 4.5. PROPUESTA 2. SERVIDOR DE IRC

Diseño e implementación de una versión simplificada de un servidor de IRC en modo consola. Internet Relay Chat (IRC) es un protocolo de aplicación que permite la conversación entre grupos de usuarios en Internet en tiempo real (síncrono) a través de Internet, teniendo las posibilidades de distintos grupos de usuarios, conversaciones privadas, existencia de moderadores en los grupos, etc.

El IRC se definió originalmente en el RFC 1459, pudiendo encontrarse las actuales especificaciones en los RFC 2810, 2811, 2812 y 2813.

La comunicación entre los clientes del chat se realizará bajo el protocolo de transporte TCP. La práctica que se propone consiste en la realización de una aplicación cliente/servidor que implemente un servicio de chat simplificado mediante el cual, los usuarios del chat (los clientes) se conectan al servicio de chat (el servidor) y pueden mandar y recibir mensajes de forma instantánea entre los distintos usuarios. Se plantean dos versiones de las prácticas una con mayor complejidad que la otra, para que el estudiante pueda elegir hasta que nivel quiere llegar.

**1) La versión reducida:** implica menos programación y el procedimiento que se seguirá será el siguiente

• Un cliente se conecta al servicio y si la conexión ha sido correcta el sistema devuelve +0k. Usuario Conectado.

• Un cliente podrá participar en el chat indicando un mensaje de UNIR usuario, siendo usuario el identificador que desea usar en el chat. Recibido este mensaje en el servidor, éste se encargará de enviar al nuevo usuario los usuarios que ya están conectados en ese momento (de este modo el usuario que se conecte tendrá conocimiento de quienes forman parte de la comunicación) y al resto de usuarios les informará del nuevo usuario que se ha conectado.

• Cada vez que un cliente mande un mensaje al servidor, ´este deberá reenviarlo a su vez a todos los demás clientes conectados, sin incluir al cliente del que recibió el mensaje.

• Un mensaje de un cliente en la que se envía la petición SALIR, hará que el servidor le dé de baja y ya no le mande más mensajes, además le comunicará al resto de usuarios que dicho cliente ha abandonado la conversación.

Algunas de las restricciones a tener en cuenta son:

• La comunicación será mediante consola.

• Solamente se permite una única conversación, todos los usuarios que se conecten al chat participan en la misma conversación.

• El cliente deberá aceptar como argumento una dirección IP que será la dirección del servidor al que se conectará.

• El protocolo deberá permitir mandar mensajes de tamaño arbitrario. Teniendo como tamaño máximo envío una cadena de longitud 250 caracteres.

• El servidor aceptará servicios en el puerto 2050.

• El servidor debe permitir la conexión de varios clientes simultáneamente. Se utilizará la función select() para permitir esta posibilidad.

• El número máximo de clientes conectados en el chat será de 50 usuarios.

• Todos los mensajes que se envíen al servidor recibirán una respuesta indicando al menos +Ok. Texto informativo o que ha habido algún error -Err. Texto informativo.

**2) Segunda versión:** la segunda parte de la práctica consistirá en introducir la posibilidad de mantener varias salas de chats operativas, pudiendo decidir cada usuario registrado en qué sala desea participar, así como la creación y eliminación de salas. El procedimiento que se seguirá será el siguiente:

• Un cliente se conecta al servicio y si la conexión ha sido correcta el sistema devuelve +0k. Usuario conectado.

• Para poder acceder a los servicios es necesario identificarse mediante el envío del usuario y clave para que el sistema lo valide . Los mensajes que deben indicarse son: USUARIO usuario para indicar el usuario, tras el cual el servidor enviará +Ok. Usuario correcto o -Err. Usuario incorrecto. En caso de ser correcto el siguiente mensaje que se espera recibir es PASSWORD password, donde el servidor responderá con el mensaje de +Ok. Usuario validado o -Err. Error en la validación.

• Un usuario nuevo podrá registrarse mediante el mensaje REGISTRO -u usuario -p password -d Nombre y Apellidos -c Ciudad. Se llevará un control para evitar colisiones con los nombre de usuarios.

• Una vez que el proceso de validación haya sido correcto, el servidor lo registra como cliente del chat y le envía los canales disponibles para comunicarse. El usuario indicará el canal de chat en la que desea participar mediante un mensaje UNIR nombreCanal. Si el usuario desea abrir un nuevo canal podrá realizarlo mediante el mensaje AÑADIR nombreCanal -d breveDescripcion a partir de este momento el servidor también ofrecerá este nuevo canal a los nuevos usuarios que se conecten y también se lo comunicará a los usuarios conectados en otros canales mediante un mensaje. El servidor también controlará que no haya más de un canal con el mismo identificador.

• Para abandonar un canal se usará el mensaje DEJAR canal, de este modo el usuario ya no puede mandar mensajes ni recibirlos con respecto a ese canal, pero podrá conectarse a otros canales. Si el usuario que abandona un canal era el último usuario que tenía el canal, este canal se eliminará de los canales disponibles.

• Para salir del servicio se usará el mensaje SALIR, de este modo el servidor lo eliminará de clientes conectados.

• Cualquier mensaje que no use uno de los especificadores detallados, se considerará un mensaje para compartir con los usuarios del canal. Además de las restricciones contempladas en la primera versión, algunas restricciones a tener en cuenta en esta nueva versión que se debe implementar es:

  o Siempre existirá un canal por defecto, el cual estará en funcionamiento mientras esté el servidor activo tenga o no clientes. Un máximo de 15 canales podrán estar activos a la vez, en caso de que un cliente desee crear un nuevo canal habiendo 15 canales activos no se permitirá esta acción y tendrá que esperar.

  o Todos los mensajes mandados al servidor con respecto a la conexión y validación o la creación de canales recibirán una respuesta indicando que ha sido correcto +Ok. Texto informativo o que ha habido algún error -Err. Texto informativo.

En esta ampliación del chat, el servidor también llevará un control de la IP, la hora de conexión y de desconexión de cada usuario. Esta información se almacenará en ficheros de texto plano, generando un fichero por día, cuyo nombre será Día-Mes-Año (para obtener esta información el servidor hará uso a su vez del servidor de daytime de la máquina gongora.uco.es).

No es necesario establecer una exclusión en la comunicación del fichero ya que la concurrencia es simulada y por tanto no van a acceder dos procesos a la vez al fichero.

**3) Resumen de paquetes a utilizar:** considerando la práctica completa, vamos a considerar los siguientes tipos de mensajes con el siguiente formato cada uno:

- USUARIO usuario: mensaje para introducir el usuario que desea conectarse (se considera en la segunda parte de la práctica).
- PASSWORD contraseña: mensaje para introducir la contraseña asociada al usuario (se considera en la segunda parte de la práctica).
- REGISTRO -u usuario -p password -d Nombre y Apellidos -c Ciudad: mensaje mediante el cual el usuario solicita registrarse para acceder al servicio de chat que escucha en el puerto TCP 2050 (se considera en la segunda parte de la práctica).
- UNIR usuario: mensaje para solicitar permiso para introducirse al canal de comunicación del chat (se considera en la primera parte de la práctica).
- UNIR nombreCanal: mensaje para solicitar permiso para introducirse a un canal concreto (se considera en la segunda parte de la práctica).
- AN˜ ADIR nombreCanal -d breveDescripcion: mensaje para añadir un nuevo canal al chat (se considera en la segunda parte de la práctica).
- DEJAR canal: mensaje para solicitar salir de un canal (se considera en la segunda parte de la práctica).
- SALIR: mensaje para solicitar salir del chat (se considera en la ambas partes de la práctica).
- Cualquier otra línea que se escriba será reconocida por el protocolo como un mensaje a compartir con el resto de usuarios del canal de comunicación en el chat.

## 5. CONCLUSIONES

El diseño de una aplicación en red como la que se plantea en este trabajo ha sido utilizada en la asignatura troncal de Redes de tercer curso del grado de Ingeniería Informática en la Universidad de Córdoba en los últimos años.

La práctica propuesta permite la experimentación con tecnologías básicas, como son los sockets. Nuestra experiencia nos permite asegurar que los alumnos que han participado en las prácticas han conseguido comprender bien el concepto de diseño de aplicaciones en red y el estudio de los protocolos a utilizar. Además, afianza los conceptos relativos a interfaz, servicio y protocolo y adquiere suficientes conocimientos para ser capaz de diseñar y desarrollar aplicaciones que pueden utilizarse en Internet siguiendo el modelo cliente/servidor.

De este modo, podemos concluir que la experiencia está resultando muy positiva, lográndose los objetivos establecidos y percibiendo un alto grado de interés y motivación por parte de los estudiantes.

## AGRADECIMIENTOS

Los autores agradecen a la Junta de Andalucía y al Ministerio de Tecnología y Ciencia la financiación mediante los proyectos P08-TIC-3720 and TIN-2011-22408, y los fondos FEDER.

## BIBLIOGRAFÍA

[1] S. Walton, Programación de Socket Linux, Pearson Alhambra, 2001.

[2] A. S. Tanenbaum, Redes De Computadoras, Pearson, 2003.

[3] J. Gómez López, N. Padilla Soriano, and J. A. Gil Martínez-Abarca, Administración de sistemas operativos Windows y Linux. Un enfoque práctico, Ra-Ma, Librería y Editorial Microinformática, 2006.