Práctica 2: Sockets

Profesor: José Luis Torres Rodríguez (jluis+redes20151@ciencias.unam.mx)
Ayudante: Laura Reyes (ts.anreb@gmail.com)
Ayudante de laboratorio: Miguel Angel Piña Avelino (miguel_pinia+redes20151@ciencias.unam.mx)

Fecha de entrega: 18 de agosto de 2014 a la hora del laboratorio

Introducción

Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Práctica en laboratorio

En está práctica contarán con un ejemplo dummy de lo que es un socket. Este ejemplo es un cliente y un servidor tcp para realizar un *ECHO*.
En el laboratorio se explicará de manera un poco más detallada su funcionamiento. Este ejemplo servirá de base para realizar los siguientes ejercicios.

Ejercicios teóricos

1. Los servidores están diseñados para funcionar durante largos periodos de tiempo sin parar. Esto implica que es necesario que esten diseñados para proveer un buen servicio sin importar lo que los clientes hagan. Examine el archivo *server.c* y liste una serie de situaciones en las que un cliente puede hacer para que el servidor de un mal servicio a otros cliente. Sugiera mejoras para los problemas que encontró.

2. ¿Qué sucede si el servidor TCP nunca llama a accept()? ¿Qué sucede si un cliente TCP envía datos a través de un socket que no ha establecido conexión con la función accept() que se encuentra en el lado del servidor?

Ejercicios prácticos

1. Modifique el archivo *server.c* para enviar y recibir un sólo byte a la vez, dormir 1 segundo y enviar el siguiente byte. Compruebe que *client.c* necesita multiples recepciones de bytes para recibir correctamente la cadena enviada por el *ECHO*, a pesar de que fue enviada con un sólo send().

2. Modifique el archivo *server.c* para leer y escribir un byte a la vez y una vez hecho esto, cerrar el socket. ¿Qué sucede cuando el cliente envia una cadena de varios bytes? (Esta respuesta puede variar dependiendo el sistema operativo).

3. Modifique el archivo *server.c* para que pueda soportar múltiples clientes a la vez. Esto implica que el servidor tendrá que estarse ejecutando sin parar.

Notas sobre los programas

- Los programas anteriores deben de estar debidamente documentados (comentarios en cada función utilizada).
- Deben de tener su propio archivo *makefile* para compilarlo y ejecutarlo.
- Deben estar bien organizados. Es decir, deben de tener una carpeta para cada uno de ellos con una estructura adecuada.
- La entrega de prácticas se realizará en un archivo comprimido *tar.gz* y se envía a la dirección de correo: *miguel_pinia+redes20151@ciencias.unam.mx*