Práctica de Sockets.

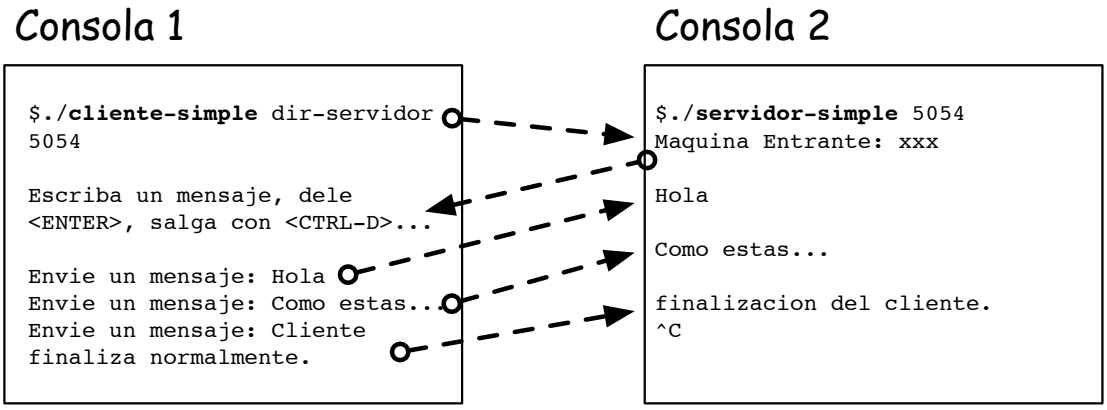
Andrés Arcia-Moret

El siguiente trabajo práctico tiene como finalidad comprender los alcances de la comunicación confiable y no confiable a través de la Interfaz de Sockets. Los arreglos del cliente y del servidor se harán en Lenguaje C.

Recuerde utilizar el laminario de la sesión de Sockets para ayudarse en la completación de los programas.

1. Conexión simple cliente – servidor en modo confiable (usando TCP).

   Dentro de la carpeta "practica-sockets" copie en un directorio aparte los archivos **cliente-simple.c** y **servidor-simple.c**. Estos archivos han de ser completados según las laminas vistas en la parte teórica. Una vez que los programas hayan sido completados, usted debería estar en capacidad de poder ejecutar la siguiente interacción entre el cliente y el servidor.



   Recuerde que el servidor debe ser lanzado primero con un número de puerto (del tipo usuario) primero que el cliente.

   Una vez que tenga listos los programas, pruebe el cliente y responda a las siguientes preguntas:

   a. ¿Qué mensaje arroja el cliente cuando se le suministra un puerto invalido?
   b. ¿Qué mensaje arroja el cliente cuando se le suministra un nombre de maquina o un IP invalido?

2. Conexión con un servidor TCP para soportar múltiples clientes simultáneos (servidor concurrente).

   Ahora copie en otro directorio el archivo **servidor-simple.c** y transformelo para que soporte visitas concurrentes. Para ello, el servidor tiene que crear un nuevo proceso por cada solicitud recibida.

   Existe la posibilidad de ver en el "servidor de escucha" a los clientes que desean conectarse a través de un único puerto. Para ello, la estructura sockaddr_in tiene un campo llamado "port" que contiene el número del puerto del cliente. Recuerde que el número de puerto debe ser tratado por la función **int ntohs(int)**. ¿Podría explicar por qué?

   Luego de haberlo transformado, responda a las siguientes preguntas:

   a. En los ejemplos completados, se ha hecho arbitrariamente la selección de colocar la dirección del socket como INADDR_ANY. Obser lo que sucedería si esta dirección se cambia por una dirección específica (por ejemplo, 192.168.1.240).
   b. Con el comando **netstat –atn** en Linux, indique el número del puerto que se le ha asignado al cliente una vez que éste ha sido conectado al servidor. Escriba enteramente la línea que le corresponde al cliente.
   c. Con la ayuda de Wireshark observe lo que sucede a nivel TCP (use la palabra TCP en la linea de filtrado) cuando la función **connect()** es llamada desde el servidor. Busque en las láminas o en Google, ¿Cómo se denomina a la fase inicial de intercambios de mensajes de TCP?

3. Comunicación en modo no confiable - datagrama (UDP).

   3.1 Emisor y receptor simple.

   En esta práctica usted tomará los archivos **cliente-no-conf.c** y **servidor-no-conf.c** complete los ejercicios especificados dentro de los programas. EL objetivo es que no exista una "conexión" como en el ejercicio anterior. Esto implicaría que el cliente (emisor) pueda enviar a cualquier receptor y que un servidor (receptor) pueda recibir de cualquier emisor.

   Para realizar este programa es necesario conocer el tamaño del mensaje que se requiere enviar. Este mensaje será enviado a partir de un buffer …

   Luego de haberlo transformado, responda a las siguientes preguntas:

   a. Explique el comportamiento si se envía un mensaje hacia una máquina inexistente y, en una segunda ocasión, hacia un puerto inexistente. ¿Qué tipo de comportamiento (mensajes) obtiene?
   b. Constate el tamaño máximo del mensaje UDP que puede enviarse. Para ello, haga distintas pruebas de tamaño de mensaje entre 65520 y

65535 (este último representa el tamaño más grande “permitido”). Una vez determinado el tamaño exacto, observe con Wireshark cuantos datagramas son enviados con un mensaje tan grande.

c. Ejercicio para la casa: Existe una opción para aumentar el tamaño del buffer del socket: SO_RCVBUF. ¿Cómo se puede utilizar esta opción para poder enviar mensajes de mayor tamaño? ¿Cuál es el valor por omisión en un kernel Linux actual?

3.2 El modo broadcast (envío a toda la subred):

A través de la función setsockopt y de la dirección IP correcta (en modo broadcast, visible a través de **ifconfig**) se puede enviar un mismo mensaje sin especificar el destino.

Modifique el programa cliente para poder enviar un mensaje a todas las maquinas de la sala. Recuerde que debe especificar el puerto destino.