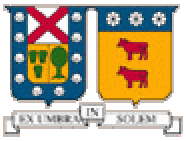
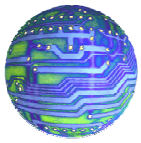


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA



Tarea Nº 3

Programación de Sistemas

Segundo Semestre 2003

"Pesaje de Paginas WEB"

(Documentación)

Rodrigo Pinto A 9821028-8
Christian Lalanne A 9821082-2

Objetivos:

El objetivo de la presente tarea es crear una aplicación capaz de determinar el tamaño de una pagina web completa pedida por el usuario al programa.

Estructura de la solución ideada:

La solución a esta tarea se compone de dos partes; un programa cliente (**alimentador**) encargado de conectarse a un programa servidor, corriendo en una dirección ip y puerto especificado como argumento y solicitar la pagina web deseada por el usuario (entregada también como argumento); y un programa servidor (**ppw**), encargado de solicitar la pagina web pedida por el programa cliente (realizando todas las conexiones requeridas) a un servidor intermediador proxy ubicado en una máquina del ATMLAB de la red de electrónica en la ip y puerto también entregados como argumento al programa.

A continuación se describe con mas detalle ambos programas:

1.- Programa Cliente (alimentador):

Sintaxis: alimentador <IP_servidor de pesaje> <Puerto_servidor de pesaje> <URL>

IP_servidor de pesaje: dirección ip de la maquina en la que esta corriendo el programa servidor (**ppw**), para la tarea la dirección corresponde a 172.16.0.4 que es la dirección de aragorn.elo.utfsm.cl, (pero podría ser la dirección de otra maquina corriendo este programa).

Puerto_servidor de pesaje: numero de puerto en el cual esta corriendo el programa servidor (**ppw**), este puerto queda determinado al correr el programa servidor explicado mas adelante, y debe ser un puerto no privilegiado (numero mayor a 1024).

URL: Dirección de la pagina web que se desea pesar, esta no debe comenzar con http://, ya que de otra manera, el programa funcionaria, pero los directorios creados por el programa no serian borrados.

El código del programa cliente fue hecho en bash, para simplificar esta parte de la solución, primero se setea la variable de entorno http_proxy, usando el comando export y dándole los valores correspondientes a la dirección ip de la maquina en la cual esta el programa servidor (**ppw**) y el puerto en el que esta corriendo, para luego ocupar el comando wget –p que es el que finalmente solicita la pagina que se desea pesar y la función que entabla todas las conexiones requeridas para bajar exitosamente la pagina, finalmente con el comando rm –rf se borra la carpeta creada que contiene la pagina web pedida; para un mejor entendimiento del programa; en la Fig.1 se muestra un diagrama de bloques con el ciclo de este.

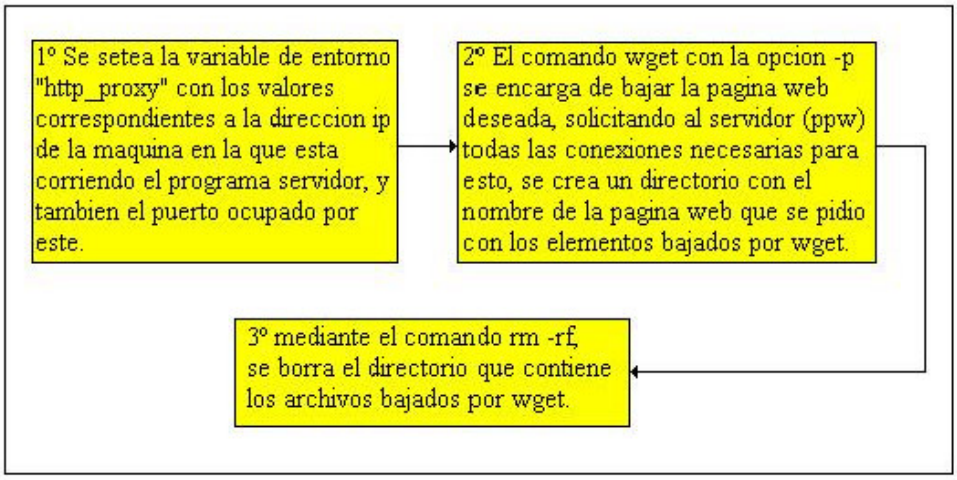


Fig.1 Diagrama de bloques programa cliente (**alimentador**)

2.- Programa servidor (ppw):

Sintaxis: ppw <puerto_local> <IP_proxy> <puerto_proxy>

Puerto_local: numero del puerto elegido por el usuario para correr el programa servidor; debe ser un puerto no privilegiado (numero mayor 1024), este mismo numero de puerto debe ser especificado en el programa cliente (**alimentador**) para que este sepa en que puerto esta funcionando el servidor (**ppw**).

IP_proxy: dirección IP de la maquina servidor intermediario (proxy) al que se conectara el programa servidor (**ppw**) solicitándole la pagina web pedida por el cliente (**alimentador**); en nuestro caso la dirección es 200.1.28.8 (dirección de la maquina proxy del ATMLAB de la red de electrónica).

Puerto_proxy: numero del puerto en el cual esta corriendo el servidor proxy (y al cual deberá conectarse el programa servidor) de la maquina a la que se conecta el programa servidor (**ppw**) para solicitar las paginas deseadas por el cliente (**alimentador**); en nuestro caso este numero es 3128, que es el numero de puerto en el cual funciona el servidor proxy de la maquina del ATMLAB de la red de electrónica).

El código del programa **ppw** fue escrito en lenguaje C , **ppw** lo primero que hace es obtener en una estructura hostent los datos del proxy, luego con estos datos se llenan los campos de la estructura sockaddr_in llamada proxy, luego se llena la estructura sockaddr_in llamada server que contiene, por ejemplo, el puerto en que va a funcionar el servidor ppw; luego se crea el socket S que va a permitir aceptar conexiones desde el alimentador. Con la función bind se asocia la estructura server nombrada anteriormente (puerto y dirección ip) con el socket S que es por donde se van a aceptar las conexiones con el alimentador.

Una vez definidos los conjuntos de descriptores readfds y readfdsCopy lo primero que se hace es agregar el socket S antes descrito al conjunto de descriptores readfdsCopy y además agregar el descriptor 0 (que representa la entrada estándar ó teclado) también al readfds copy , para después dentro del ciclo for infinito, copiar todos los descriptores de readfdsCopy al conjunto de descriptores readfdsCopy el cual será monitoreado con la función select.

Una vez que se ha usado la función select para monitorear el conjunto de descriptores readfdsCopy y el programa sabe que descriptores son los que tienen actividad, lo primero que hace el programa es chequear si el socket S tiene actividad, si la tiene es porque el alimentador solicita crear una nueva conexión; luego de aceptarla con la función accept y agregar el socket retornado por esta función al arreglo de socket socklist[] (en donde se van a guardar todos los socket que genera el accept, es decir, todos los socket con los cuales se transfieren datos del alimentador al **ppw** y viceversa), se agrega este socket generado por el accept al conjunto de descriptores readfdsCopy para que en el próximo ciclo del for infinito, este también sea monitoreado para ver si tiene actividad.

Inmediatamente después de hacer todo lo mencionado en el párrafo anterior, se crea el socket SL para conectarse al proxy y se guarda este socket SL (en realidad el descriptor del socket) en el arreglo proxylist[], (donde se van a guardar todos los socket con los cuales se van a transferir y recibir datos con el proxy) y se agrega este socket al conjunto de descriptores readfdsCopy (para que en el próximo ciclo del for este descriptor se monitoreado por el select).

Luego, se utiliza la función connect para establecer la conexión con el proxy , y se revisa si el descriptor 0 (ó entrada estándar) que fue monitoreado ya con la función select, tiene actividad; si es así es porque en el alimentador el usuario ha presionado ENTER, lo que le indica al ppw que debe mostrar por pantalla las estadísticas pedidas en las especificaciones de la tarea (el número de conexiones establecidas desde le inicio, el número de conexiones establecidas desde el último ENTER, el número de bytes transmitido por el proxy desde el inicio (peso de la pagina) y el número de bytes transmitidos por el proxy desde el último ENTER).

Finalmente se revisa si hay actividad en los sockets del arreglo socklist y después proxylist; si la hay, sus requerimientos son atendidos por la función manejo; que recibe bytes desde el alimentador y los envía al proxy o recibe bytes desde el proxy y los envía al alimentador dependiendo en que orden se le entreguen los parámetros. Para un mejor

entendimiento de la tarea se presenta a continuación un diagrama de bloques explicando el ciclo realizado por el programa servidor (**ppw**).

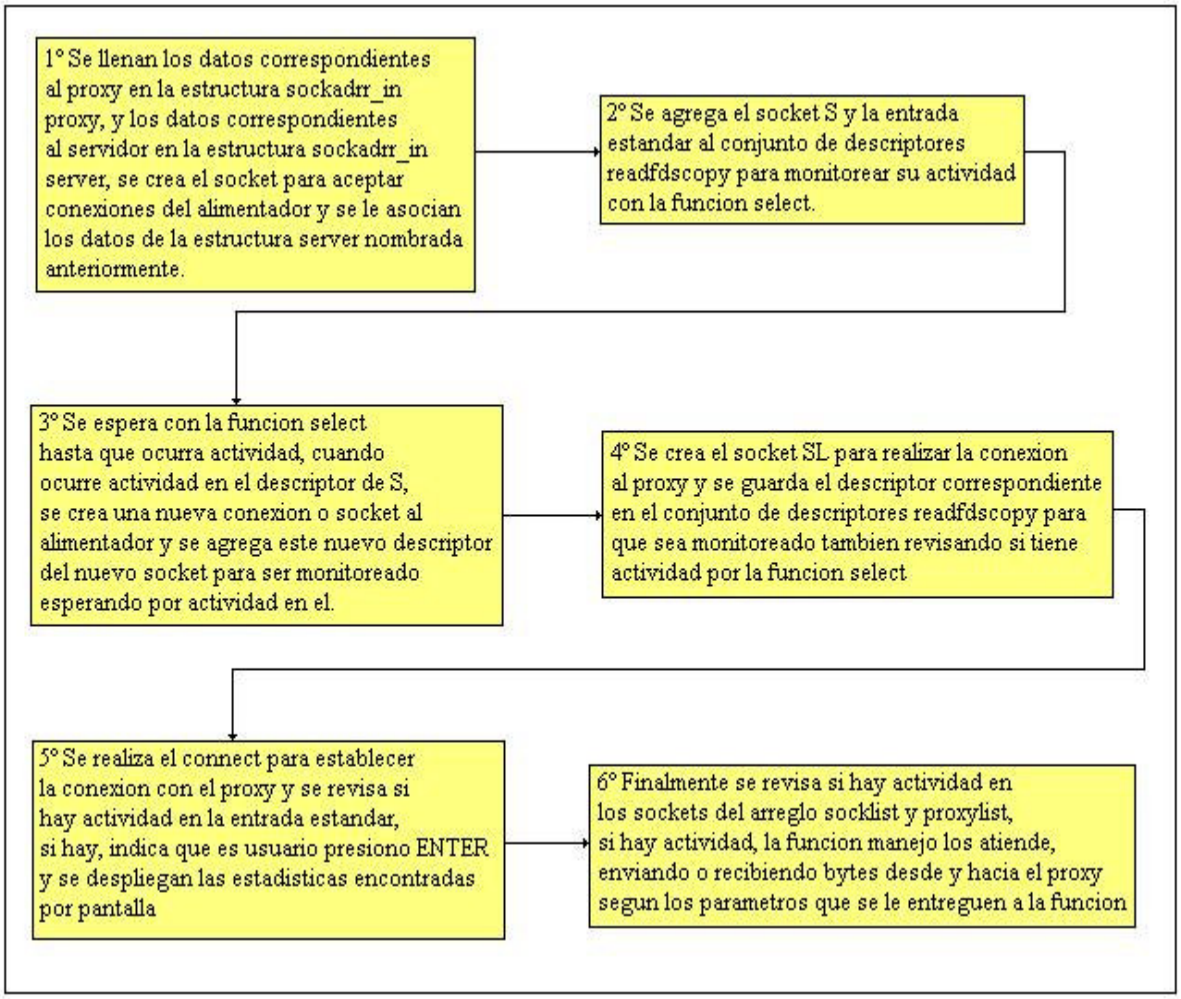


Fig. 2 Diagrama de bloques servidor **ppw**.

Fig. 3 Ejemplo de ejecución del programa pesando las paginas especificadas en la tarea.

Problemas:

El único problema encontrado en la ejecución de nuestra solución, es que si el alimentador crea una carpeta (siempre se crea una carpeta donde baja los archivos de la pagina solicitada, generalmente con el mismo nombre de la url solicitada), de distinto nombre a la url solicitada, esta no será borrada, el origen de este problema es el comando ocupado para eliminar la carpeta creada por el alimentador, este comando (rm –rf $3) fue la mejor solución encontrada, también se pensó el borrar todas las carpetas del directorio donde se ejecuta el alimentador, pero dicha solución fue descartada por presentar problemas obvios. La mayor cantidad de tiempo ocupada fue en establecer conexiones entre el alimentador y el ppw, y el ppw y el proxy.
Como ultima observación, debemos decir que no es necesario reiniciar el servidor **ppw** después de pesar una pagina, ya que este servidor entrega las conexiones realizadas desde el ultimo ENTER y los bytes transmitidos por el proxy (“Peso de la pagina WEB”) también desde el ultimo ENTER; solo es necesario que la primera pagina baje completamente, para empezar a “pesar” la otra y mirar los campos mencionados anteriormente, como se muestra en la Fig. 3.