Universidad Simón Bolívar
Redes de Computadores 1
Juan Garcia 05-38207
Jessica Perez 05-38724
Enero – Marzo 2009

# ***Manual de Uso - Proyecto 2***

**Nombre de las funciones:**

'c_tiempo' : aplicación Cliente (TCP o UDP)

's_tiempo' : aplicación Servidor (TCP o UDP)

**Descripción:**

'c_tiempo' y 's_tiempo' son usados para sincronizar los relojes de una maquina cliente, con el de una maquina servidor. Para ello usa el Algoritmo de Marzullo, el cual consiste en que el cliente pregunta varias veces la hora al servidor y encuentra un promedio entre los intervalos para hallar con mayor exactitud la el tiempo deseado.

'c_tiempo' se aplica en el cliente y es estrictamente necesario especificar el puerto por el cual se conectará al servidor, la dirección del servidor, el número de veces que el cliente preguntará al servidor y finalmente el protocolo a usar, éste puede ser UDP o TCP. Opcionalmente se le puede pedir que sólo muestre la hora, sin llegar a modificarla. Adicionalmente, la aplicación cliente permite realizar concurrentemente peticiones a varios servidores, dispuestos en un archivo de texto plano, en una misma corrida del programa.

's_tiempo' se corre en el servidor, para su funcionamiento es necesario especificarle el puerto de entrada y el protocolo de conexión, siendo éste UDP o TCP. Adicionalmente, esta aplicación presenta la característica de concurrencia, la que permite al servidor atender varias peticiones de clientes a la vez. También ahora el servidor posee una medida de calidad, que afecta los intervalos que Marzullo utilizara en su calculo. De manera que, dependiendo de la calidad, los intervalos que se obtengan de un servidor dado, aparecerán en la lista final de intervalos, tantas veces repetidos, como sea el numero de calidad. Esto, para que al calcular la hora final, aquellos servidores con alto nivel de calidad, tengan relevancia en el calculo, sobre aquellos con baja calidad.

**Sintaxis del Cliente 'c_tiempo':**

Para ejecutar el programa cliente, se debe ingresar en la linea de comandos, la siguiente instrucción:

***'c_tiempo -f <archivo_texto> -p <puerto> -v <número de preguntas> {-t|-u} [-n]'***

donde:

***-f <nombre_archivo>*** Indica el archivo de texto en donde se encuentran listados, las direcciones IP o los nombres de los servidores a los cuales se quieren hacer peticiones.

***-p <puerto>*** Indica el puerto a través del cual se debe contactar a la aplicación servidor.

***-v <número de preguntas>*** Indica el número de veces que el cliente preguntará la hora al servidor.

***-t*** Indica que se desea utilizar el protocolo TCP. La aplicación servidor debe estar ejecutándose empleando el mismo protocolo.

***-u*** Indica que se desea utilizar el protocolo UDP. La aplicación servidor debe estar ejecutándose empleando el mismo protocolo.

***-n*** Indica que la hora obtenida debe mostrarse por la salida estándar, en lugar de tratar de establecerla como hora del sistema.

**Sintaxis del Servidor 's_tiempo' :**

*Para ejecutar el programa servidor, debe colocarse en la linea de comandos, la siguiente instrucción:*

***'s_tiempo -p <puerto> -c <calidad> {-t|-u}'***

*donde:*

***-p <puerto>*** Indica el puerto a través del cual se aceptarán peticiones de la aplicación cliente.

***-c <calidad>*** Indica la calidad del servidor que vamos a ejecutar.

***-t*** Indica que se desea utilizar el protocolo TCP. Se contestarán peticiones de la aplicación cliente utilizando TCP.

***-u*** Indica que se desea utilizar el protocolo UDP. Se contestarán peticiones de la aplicación cliente utilizando TCP.

"**NOTA**": En ambos casos la opción '-t' y '-u' son mutuamente excluyentes, y la opción '-n' es opcional. Además el orden de entrada de los parámetros no afecta el resultado de la función.

**Detalles de Implementación:**

- Cliente – Servidor TCP:

Para la implementación concurrente de la aplicación usando protocolo TCP, se modificó el modelo original para incorporar la posibilidad de realizar consultas simultaneas a múltiples servidores y que cada servidor pueda contestar múltiples preguntas al mismo tiempo.

Tanto en el cliente como en el servidor TCP se hizo uso de procesos para agregar la concurrencia a las aplicaciones. El Cliente lee desde un archivo una lista de servidores, por cada servidor crea un proceso que se encarga de hacer las respectivas peticiones y aplicar la calidad del servidor correspondiente a cada intervalo. El servidor por cada conexión crea un proceso que se encarga de atenderla. Se decidió implementar esta aplicación con procesos por las características del protocolo TCP, el cual necesita de una conexión establecida para realizar sus peticiones y hacer sus respectivas comprobaciones, por ello se crea un proceso por cada conexión al servidor, de esa manera se trata cada conexión por separado y de igual forma el servidor atiende las peticiones por conexión, cada proceso por separado se encarga de responder a las peticiones enviadas por el cliente.

- Cliente – Servidor UDP:

Para la implementación concurrente de la aplicación usando el protocolo UDP, se partió del modelo original, que permitía realizar varias peticiones desde un cliente único a un solo servidor. En esta implementación se usaron las herramientas que ofrece la librería de Sockets de C para comunicaciones con protocolo UDP.

La aplicación de concurrencia al sistema vino dada gracias a Hilos de ejecución (Threads). En el programa servidor, cada vez que llega una petición de un cliente, se crea un hilo de ejecución para que la maneje. De esta manera siempre que llegue una petición al servidor, la misma podrá ser atendida de inmediato. El porque de la implementaron con hilos, es debido a las características del protocolo UDP. Dicho protocolo maneja una cola de peticiones no establecidas por una conexión, a diferencia de TCP. Acepta todas las peticiones de una misma manera, sin importar de cual cliente provenga la misma.

Entonces nuestra implementación del servidor crea un hilo de ejecución para cada petición, esto permite responder cada petición paralelamente, y satisfacer a todos los clientes.

Para la implementación del cliente también se usaron hilos de ejecución. De esta manera se garantiza igual uso de los recursos del sistema en el servidor y el cliente. Además, dado que UDP no es orientado a conexión, y muchos paquetes se pierden en la trasmisión, requiriendo retransmisión, la implementación con hilos ahorra espacio en memoria (debido al uso de memoria compartida de los hilos), cada vez que se debe transmitir una petición, a diferencia de los procesos, que duplica el código y el espacio de memoria, cada vez que se hace fork().

## Diagramas de Secuencia TCP:

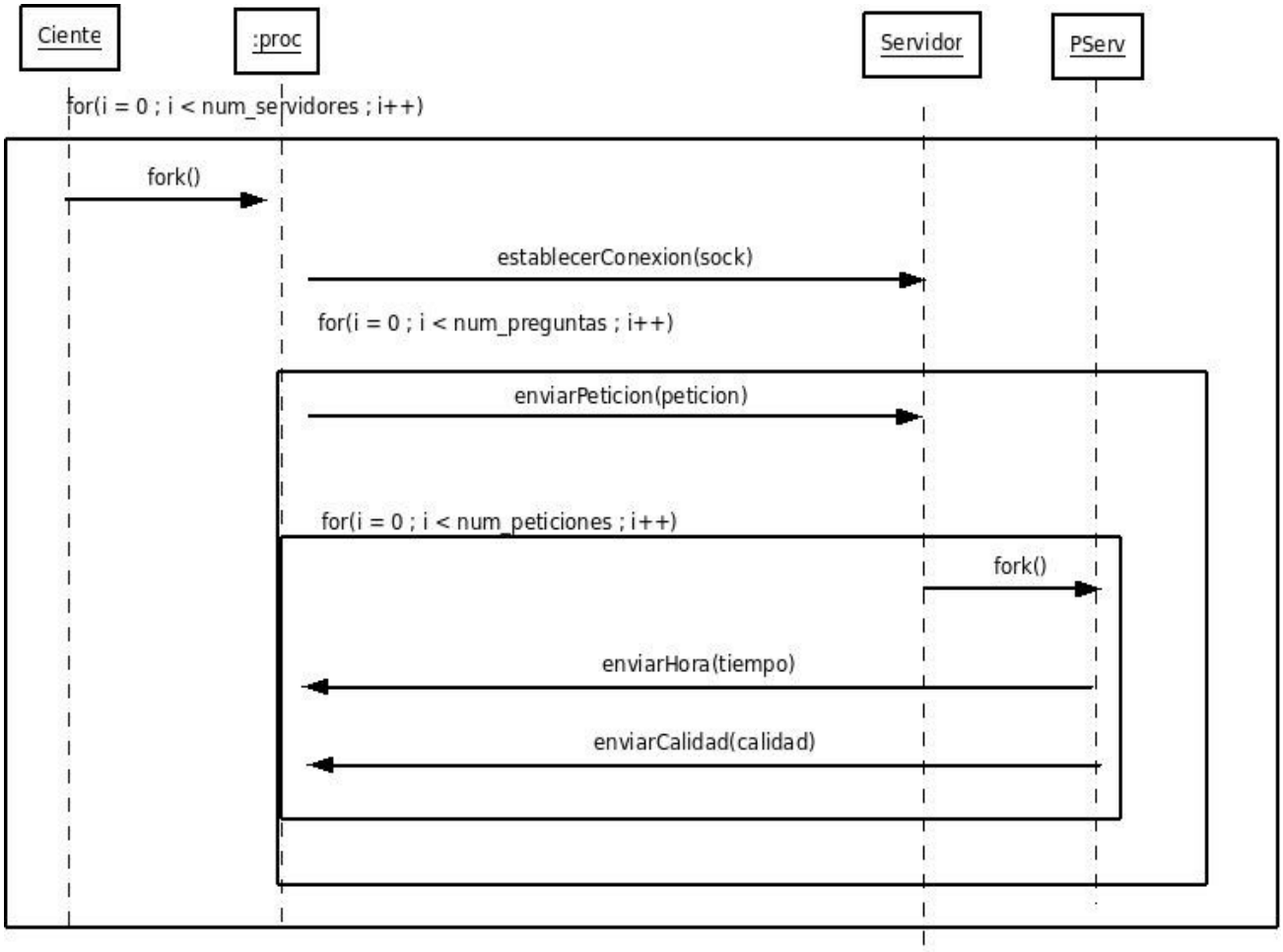

**Diagrama de secuencia UDP:**