UNICIENCIA 24
pp. 83-89
2010

IMPLEMENTACIÓN DE UN SERVIDOR FTP UTILIZANDO EL MODELO CLIENTE/SERVIDOR MEDIANTE EL USO DE SOCKETS EN LENGUAJE C UNIX CON EL FIN DE MEJORAR LOS TIEMPOS DE RESPUESTA EN LA RED

Juan de Dios Murillo Morera
e-mail: jmurillo@una.ac.cr
Santiago Caamaño Polini
e-mail: scaamano@costarricense.cr

RESUMEN

Este trabajo pretende evaluar la latencia en la transferencia de archivos utilizando un servidor FTP con un modelo cliente-servidor empleando una computadora con el sistema operativo Fedora para ejecutar el código del modelo cliente/servidor con sockets en lenguaje C UNIX, con el fin de simular un servidor que contiene archivos con diferentes formatos y tamaños, y medir la latencia de la transmisión al subir y descargar los archivos del servidor, usando diferentes tamaños de buffer. Con los resultados del retardo en la transmisión en los diferentes escenarios y al compararlos, se observa que entre mayor sea el tamaño del buffer es menor la latencia y conforme aumenta el tamaño del archivo la latencia aumenta, sin importar el formato, ni el tamaño del buffer.

ABSTRACT

This study aims to assess the latency in transferring files using an FTP server with a client-server model, using a computer with the Fedora operating system to run the code of client / server with sockets in UNIX C language, this in order to simulate a server that contains files with different formats and sizes, and measure the latency of the transmission to upload and download files from the server, using different buffer sizes. With the results of transmission delay in different scenarios and compare them, it is observed that if buffer size is larger, the latency decreases, and if the file size increases, the latency too, regardless of format or size buffer.

Palabras clave: Servidor FTP, *socket*, modelo cliente/servidor, latencia, *buffer*.

Keywords: FTP Server, Socket, Client/Server Model, Latency, Buffer

INTRODUCCIÓN

Los usuarios de una red están ingresando constantemente a servicios de servidores tales como ficheros, correos, aplicaciones, FTP, entre otros, los cuales se quejan muchas veces del tiempo que se tarda para poder obtener el servicio Por este motivo se está constantemente buscando reducir la latencia en los envíos de paquetes de datos y el consumo de ancho de banda.

Existen dispositivos y aplicaciones que solucionan los problemas creados por la latencia en la red, esto permite que las aplicaciones incrementen el flujo de transferencia. Con este trabajo, se pretende evaluar la latencia en la transferencia de archivos, mediante ciertas mediciones y con ellas determinar cuáles factores influyen para que se dé dicho retardo y cuáles de estos pueden ser mejorados para hacer más eficaz la transmisión. Para medir la latencia en la transferencia de archivos, se usa *software* desarrollado especialmente para este fin y no se encontraron antecedentes sobre el uso del modelo Cliente/Servidor mediante sockets en lenguaje C UNIX para este tipo de pruebas.

OBJETIVO GENERAL

Evaluar la latencia en la transferencia de archivos utilizando un servidor FTP con un modelo cliente-servidor.

Recibido 10 de abril de 2009 – aceptado 15 de junio de 2009

OBJETIVOS ESPECÍFICOS

Analizar los principales pasos del modelo cliente/servidor con el fin de tener claro cuál es el funcionamiento.

Establecer una comunicación por medio de sockets en C de UNIX entre varios clientes y un servidor.

Identificar el comportamiento de la transferencia de archivos al enviar distintos tipos de archivos y medir la latencia de los mismos.

Determinar la latencia de la transferencia de archivos con diferentes tamaños de buffer.

MODELO CLIENTE/SERVIDOR

> *"La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes." (Sepulveda, 2009).*

Según la definición de la referencia anterior, es un modelo en el que el procesamiento requerido para realizar una tarea se divide en dos o más procesos cooperantes. Se denomina cliente al proceso que solicita los recursos, y servidor al proceso que responde a las solicitudes. Para que pueda existir comunicación entre el cliente y el servidor debe existir la infraestructura necesaria para los mecanismos de direccionamiento y transporte. Este modelo tiene las características de que el control está centralizado los clientes no se comunican entre sí y solo los clientes pueden tomar la iniciativa para la comunicación. Utilizando el protocolo TCP/IP, un cliente envía una solicitud al servidor. Este procesa la tarea y devuelve los resultados al cliente. Usualmente un servidor escucha solicitudes en puertos TCP conocidos por los clientes. Los clientes utilizan puertos arbitrarios para enviar solicitudes al servidor.

TRANSFERENCIA DE ARCHIVOS

La transferencia de archivos en un entorno de red implica un conjunto de reglas y procedimientos para que se "entiendan" las partes implicadas en la transferencia y se pueda realizar el envío de manera satisfactoria.

A esto se le conoce como protocolo y es la forma en que se envían datos por una red. En un modelo cliente-servidor resulta necesario que un protocolo sea el encargado de regular la transferencia del archivo desde un punto a otro.

Ello implica que se lleven a cabo varias tareas para poder enviarlo:

Identificar el archivo por transferir y abrirlo en modo lectura.

Especificar la ruta destino, crear el archivo y abrirlo en modo escritura.

Se transfiere el contenido del origen al destino.

Para asegurar el control del flujo, se deben dar mensajes de control entre ambas partes.

Se indica el final del archivo para que el receptor cierre el archivo.

SERVIDOR FTP:

> *"FTP es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor." (Morales, 2009).*

> *"Servidor FTP: Computadora que funciona como servidor para ofrecer ficheros a través del protocolo de FTP a clientes FTP o a un navegador que lo soporte." (Pereira, 2009).*

Para definir el concepto como tal, es necesario partir de la definición de servidor, el cual es una computadora especializada con capacidad de brindar servicios varios a ordenadores conectados en una red. El servidor posee el *hardware* necesario para soportar las

solicitudes de los clientes que accederán a sus servicios. FTP es uno de los tantos servicios que puede brindar el servidor, al ser uno de los más utilizados desde su creación. FTP es un protocolo para la transferencia de archivos en el modelo TCP/IP y opera en la capa de aplicación del mismo. Para la realizar la transferencia FTP, utiliza los puertos 20 para datos y 21 para control, los cuales son el punto final de una conexión lógica y el medio de comunicación entre la aplicación del cliente y el servidor.

SOCKETS

> *"Es una interfaz de entrada-salida de datos que permite la intercomunicación entre procesos ejecutándose en el mismo o en distintos sistemas, unidos mediante una red." (Pereira, 2009).*

Según el autor, un *socket* es un mecanismo mediante el cual se comunican procesos con el fin primordial de intercambiar información de forma bidireccional entre distintas máquinas.

Tipos de *Socket*

> *"Sockets Stream son los más utilizados, hacen uso del protocolo TCP. Sockets Datagram hacen uso del protocolo UDP". (Pereira, 2009).*

Según el autor, existen dos tipos de socket, uno como Stream el cual es orientado a la conexión, y el Datagrama que es no orientado a la conexión.

Función *Socket*

Para crear un *socket,* se llama a la función **"socket()"**, la cual devuelve el descriptor, utilizado para conectarse, enviar y recibir datos.

El descriptor posee la información del dominio donde se realiza la conexión, el tipo de *socket*, y el protocolo.

Función Bind

Cuando se crea el *socket* en el servidor se llama a la función "bind()", la cual se utiliza para asignarle una dirección IP y un puerto por donde escuchará las solicitudes del cliente.

Formas de almacenamiento en memoria

> *"Network Byte Order y Host By Order son los métodos que el sistema operativo utiliza para almacenar los datos en la memoria". (Pereira, 2009).*

Para enviar un dato por la red, este debe tener el formato Network Byte Order y los datos que se reciben de la red deben convertirse al formato Host Byte Order.

Comunicación por medio de *sockets* tipo *Stream*

Para establecer la comunicación entre el cliente y el servidor se deben realizar los siguientes pasos:

- Cliente y servidor crean el *socket* con la función **socket()**.
- Se nombra el *socket* en el servidor con la función **bind()**.
- El servidor entra en estado de escucha de conexiones con la función **listen()**.
- El cliente solicita la conexión por medio de la función **connect()**; dicha función quedará bloqueada hasta que el servidor acepte la conexión o bien si no hay servidor en el sitio indicado, saldrá dando un error. En esta llamada se debe facilitar la dirección IP del servidor y el número de servicio que se desea.
- El servidor la acepta la conexión con la función accept() y retorna el descriptor del socket.
- Cuando se establece la comunicación tanto el cliente como el servidor pueden enviar con la función write() y recibir con la función **read()** datos y archivos.
- Se finaliza la comunicación tanto en el cliente como en el servidor mediante la función **close()**.

*Nota: cuando se cierra la conexión, realmente solo se corta la comunicación con el cliente, porque el servidor

continúa escuchando en espera de nuevas conexiones con clientes.

LATENCIA

> *"Mide el tiempo transcurrido entre la realización de una petición y el comienzo de la visualización o ejecución de los resultados. Se mide en unidades de tiempo (segundos, milisegundos...)".* **[R3]**

> *"En redes informáticas de datos la latencia es la suma de retardos dentro de la red. El retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red".* **[R3]**

Otros factores que influyen en la latencia son:

> El tamaño de los paquetes por transmitir.
> El tamaño del buffer empleado para transmitir

MATERIALES Y METODOLOGÍA

Los materiales que se emplean para realizar pruebas de medición de latencia en la transferencia de los archivos (servidor FTP) son:

- Una computadora con el sistema operativo Fedora Core versión 8 (para la programación y ejecución de los códigos fuente).
- Terminal del sistema operativo para compilar y ejecutar el código.
- Código fuente del cliente y el servidor con *sockets* en lenguaje C UNIX.
- Carpeta con el nombre Servidor, que contiene archivos con diferentes formatos y tamaños.
- Una carpeta con el nombre Cliente1, que está vacía, que se emplea para guardar los archivos del cliente1.
- Una carpeta con el nombre Cliente2, que está vacía, que se emplea para guardar los archivos del cliente1.
- Para medir la latencia, se incluye en el código fuente del cliente y el servidor, la instrucción "system(date)", la cual retorna la fecha y hora en que se ejecuta. Se utiliza el comando al inicio y fin de la transmisión. Se toma la diferencia entre ellos como el retardo en segundos en la transferencia. Se hacen cambios en los códigos fuente del cliente y el servidor de tal manera que permita el manejo de archivos.
- Al ejecutar el servidor en la terminal del sistema, se debe especificar el puerto que este va a emplear y el tamaño del *buffer*. Cuando el servidor está en ejecución espera conexión de los clientes (para estas pruebas solo se usa un máximo de dos clientes).

Al ejecutar el cliente en la terminal del sistema, se debe especificar la dirección IP del servidor (para estas pruebas se hace "localhost"), el puerto que emplea el servidor para la conexión y el tamaño del *buffer*.

Una vez aceptada la conexión el servidor envía un archivo con el nombre de los archivos que posee al cliente, el cual debe especificar la ruta donde desea guardarlo (en la carpeta Cliente1, para el caso del primer cliente) y se observa si se hizo la transferencia y su duración.

Se le presenta al cliente la opción de descargar archivos del servidor. Si desea realizar alguna descarga debe indicarlo digitando un 1 (0 si **NO** lo desea), luego debe indicar la cantidad de archivos a descargar, para cada uno de los archivos se debe indicar la ruta de origen en el servidor y la ruta de destino donde desea guardarlo en el cliente. Se observa si se hizo la transferencia a la carpeta del cliente correspondiente y su duración.

Se le presenta al cliente la opción de subir archivos al servidor, si desea subir algún debe indicarlo digitando un 1 (0 si **NO** lo desea), luego debe indicar la cantidad de archivos a subir, para cada uno de los archivos se debe indicar la ruta de origen en el cliente y la ruta de destino en el servidor. Se observa si se

hizo la transferencia a la carpeta del servidor y su duración.

Por último, se cierra la conexión con el cliente y el servidor queda en espera de más clientes. Para realizar las pruebas de latencia en la transferencia de los archivos, se levanta el servidor, se ejecutan dos clientes los cuales van a descargar y subir varios archivos de diferente formato y tamaño para medir el tiempo que dura la transferencia utilizando un cronómetro.

Luego, se modifica el tamaño del *buffer*, y empleando los mismos archivos se realiza el procedimiento

RESULTADOS

Se realizaron varias pruebas por medio de un servidor FTP con el modelo cliente/servidor mediante *sockets* en lenguaje C de UNIX. Se levantó un servidor que contenía en una carpeta seis archivos de diferentes formatos y tamaños (ver tabla 1). También, se ejecutaron 2 clientes para que descargaran y subieran archivos al servidor.

Tabla 1. Resultados de pruebas de latencia de transferencia de archivos

Características de Archivos de Prueba		DOWN LOAD Buffer 1024	UPLOAD Buffer 1024	DOWN LOAD Buffer 512	UPLOAD Buffer 512
Formato	Tamaño (MB)	Latencia (s)	Latencia (s)	Latencia (s)	Latencia (s)
pdf	60,4	3	3	3	3
pdf	4,35	0,5	0,5	0,5	0,5
doc	20,2	2	2	1	1
doc	3,37	0,5	0,5	1	0,5
ppt	72,5	3	3	2	3
ppt	4,77	0,5	0,5	0,5	0,5
avi	801	40	41	47	47
avi	47,2	3	3	3	3

Para la primera prueba se establece un *buffer* de transferencia de 1024. El primer cliente descarga todos los archivos que hay en el servidor, mientras el segundo cliente espera a que el primero termine. Esto se debe a que la conexión de los *sockets* no está empleando hilos, entonces se maneja una cola.

El cliente muestra la hora en que inicia y finaliza la transferencia del archivo. Se calcula la diferencia entre los tiempos y se obtiene la latencia de descarga con un *buffer* de 1024 (ver tercera columna de la tabla 1).

Se copian los mismos archivos del servidor en la carpeta del cliente, para evitar variaciones en los resultados, y se suben dichos archivos al servidor. Se hace la resta entre las horas de inicio y de fin de las trasmisiones para obtener la latencia de subida de archivos con un *buffer* de 1024 (ver cuarta columna de la tabla 1).

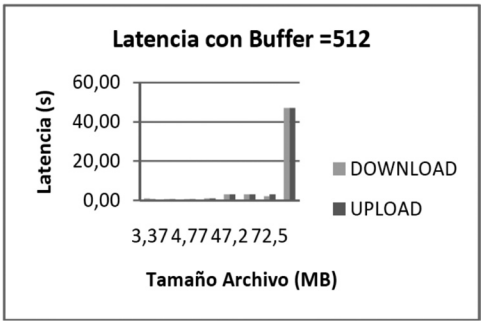
Para la segunda prueba se establece un *buffer* de transferencia de 512, y el cliente descarga los mismos archivos que en la prueba anterior desde el servidor. Se toma la hora en que inicia y finaliza la transferencia del archivo, se calcula la diferencia de tiempo y se obtiene la latencia de descarga con un buffer de 512 (ver cuarta columna de la tabla 1).

Se copian los mismos archivos del servidor en la carpeta del cliente, y se suben dichos archivos al servidor, se hace la resta entre las horas de inicio y de fin de las trasmisiones, para obtener la latencia de subida de archivos con un buffer de 512 (ver quinta columna de la tabla 1).

Cuando se intentó hacer pruebas con un *buffer* de 2048, se presentaba un error que evitaba que tanto el servidor como el cliente siguieran ejecutándose.

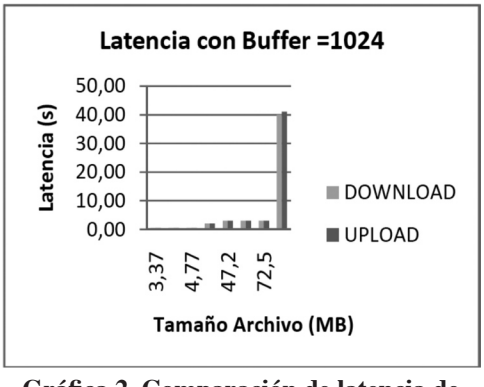
DISCUSIÓN DE RESULTADOS

Se analizan los resultados realizando varias comparaciones entre las latencias que se obtienen de las pruebas llevadas a cabo. Si comparamos la descarga y la subida de archivos con un mismo tamaño de buffer en este caso 512, como podemos observar en la gráfica 1, los resultados son casi iguales para los dos escenarios. Esto se debe a que el buffer para realizar la transferencia, tanto de subida como de bajada, es el mismo.



Gráfica 1. Comparación de latencia de descarga y subida de archivos con un *buffer* de 512

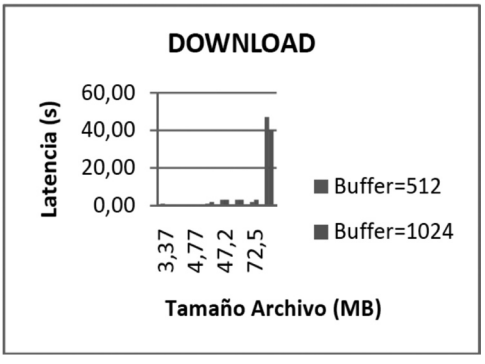
Si comparamos la descarga y la subida de archivos con un mismo tamaño de buffer, para este caso 1024 (ver gráfica 2), obtenemos el mismo resultado que en la comparación anterior, ya que sin importar el tamaño del buffer que se emplee, mientras sea el mismo para descargar y subir archivos, no provoca cambios significativos.



Gráfica 2. Comparación de latencia de descarga y subida de archivos con un *buffer* de 1024

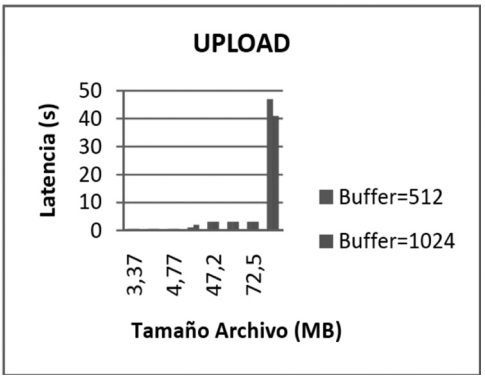
No obstante, si comparamos la descarga de archivos, con diferentes tamaños de *buffer*, para este caso 1024 y 512 (ver gráfica 3), se puede notar que la latencia es mayor con un buffer de menor tamaño.

Lo anterior se debe a que se aprovecha mejor la memoria en el bus interno de la computadora, lo cual hace que a mayor *buffer*, la velocidad de transmisión de los datos sea considerablemente mayor, en ambos sentidos. Aumentando el tamaño del *buffer*, la cantidad de información por transportar disminuye al tener menos encabezados y acuses de recibido entre el cliente y el servidor.



Gráfica 3. Comparación de latencia de descarga de archivos con diferentes tamaños de *buffer*

El análisis anterior también aplica cuando se suben archivos con diferentes tamaños de *buffer*, para este caso 1024 y 512 (ver gráfica 4).



Gráfica 4. Comparación de latencia de subida de archivos con diferentes tamaños de *buffer*

Al comparar todas las gráficas vemos un factor común entre ellas, que conforme aumenta el tamaño del archivo, la latencia aumenta, sin importar mucho el formato, porque el archivo se debe segmentar en más partes para que el buffer realice la transferencia.

Se ha podido concluir según el experimento llevado a cabo que entre mayor sea el tamaño del *buffer* es menor la latencia de

transferencia de archivos, porque hace más eficiente este proceso.

CONCLUSIONES

Cuanto mayor sea el tamaño del buffer es menor la latencia de transferencia de archivos, ya que hace más eficiente este proceso.

Conforme aumenta el tamaño del archivo la latencia aumenta, sin importar mucho el formato, ni el tamaño del *buffer*.

Con el modelo cliente/servidor implementado con *sockets* en lenguaje C de UNIX, al ejecutar varios clientes, si estos no establecen la conexión empleando hilos, mientras un cliente este cargando o descargando un archivo, el resto de clientes tendrán que esperar a que termine, porque la conexión de los *sockets* sin hilos, es manejado mediante una cola.

Para establecer una conexión exitosa mediante el modelo cliente/servidor se deben llevar a cabo una serie de pasos: crear el *socket*, nombrar el *socket* en el servidor, el servidor debe entrar en estado de escucha de conexiones, el cliente debe solicitar la conexión, el servidor debe aceptar la conexión, tanto el cliente como el servidor pueden enviar y recibir datos y archivos y por ultimo finalizar la comunicación.

RECOMENDACIONES

Se recomienda para pruebas posteriores realizar el manejo de las conexiones de los clientes al servidor mediante hilos, para que estos trabajen de manera concurrente y puedan ser atendidos en forma paralela.

Las pruebas se efectuaron de forma local, lo cual podría ser un aspecto por considerar en una mejora futura al llevar a cabo las pruebas de una forma no *localhost* sino distribuida haciendo uso de varias estaciones de trabajo con iguales condiciones de *hardware* y de *software* o bien con escenarios de diferentes condiciones de *hardware* y de *software*.

BIBLIOGRAFÍA

Morales Vázquez, J. (2009). Una Introducción a la Monitorización de Recursos en UNIX, Recuperado 19 de abril del 2009, de <http://www.moratalaz.jazztel.es/pdfs/monitorizacion.pdf>

Pereira, A. (2009). Programación de sockets en lenguaje C, Recuperado 19 de abril del 2009, de <http://www.eslinux.com/articulos/8591/programacionsockets-lenguaje-c>

Sepúlveda Ibáñez, D. (2009). Arquitectura Cliente/Servidor, Recuperado 19 de abril del 2009, de <http://www.csae.map.es/csi/silice/Global71.html>