



Instituto Tecnológico de Costa Rica

Escuela de Computación

Carrera de Ingeniería en Computación

Curso:
Principios de Sistemas Operativos (IC-6600)

Proyecto #3
Sockets

Profesor:
Armando Arce Orozco

Realizado por
Ricardo Andrey Sánchez Delgado
2014010852

Domingo 18 de junio del 2017

I Semestre 2017

Contenido

Introducción	3
Descripción del problema	4
Definición de estructuras de datos	5
Descripción detallada y explicación de los componentes principales del servidor	6
Descripción detallada y explicación de los componentes principales del cliente	7
Manejo de sockets	8
Manejo de solicitudes	8
Rutina de comparación	8
Descripción de protocolos y formatos	10
Análisis de resultados de pruebas	11
Conclusiones sobre rendimiento y correctitud	17

Introducción

Hoy en día es de gran importancia la sincronización entre las diferentes computadoras del mundo, para así poder brindar consistencia entre todos los datos y así en general formar una red donde se compartan archivos sin conflictos y se manejara todo como si fuera un solo.

Existen diferentes servicios como lo son Dropbox, OneDrive, Google Drive que ofrecen estos servicios a nivel global con ayuda de internet. Por lo que las herramientas se han vuelto muy complicadas y dependientes del internet.

En un inicio los programas eran simples, y permiten la sincronización de archivos entre diferentes computadoras a nivel local, sin grandes software, sin cosas muy complicadas.

El propósito de este proyecto es realizar un pequeño programa que se encargue de realizar la tarea anterior, por lo que se podrán añadir, modificar y eliminar archivos, adicionalmente se tratan conflictos para así mantener la integridad de los archivos.

Descripción del problema

Tercer Proyecto

Un problema que presentan los diferentes sistemas de sincronización de archivos (cómo DropBox, Live Drive y SugarSync) es que solo funcionan conectados al Internet. Desde antes existían aplicaciones sencillas que permitían sincronizar máquinas en una red sin conectarse a la Web (por ejemplo, una laptop y una computadora de escritorio).

En este proyecto se pretende desarrollar una pequeña herramienta que permita la sincronización de un directorio entre dos máquinas.

Programa sincronizador

El programa sincronizador se debe ejecutar en dos máquinas al mismo tiempo. En la primer máquina el programa recibirá como parámetro únicamente el nombre del directorio a sincronizar y quedará esperando conexiones en el puerto 8889.

En la otra máquina se debe pasar como parámetros el nombre del directorio local a sincronizar y el IP de la primer máquina.

Al iniciarse el programa deberá identificar los archivos en su directorio local, almacenando su tamaño, hora y fecha de la última actualización. Debe comparar dicha información con un listado de su corrida anterior, y determinar cuáles archivos han cambiado. Si algún archivo ha cambiado, se eliminó, o se creó, se debe enviar esa información a la otra máquina para que realice los cambios adecuados. Dentro de dichos cambios está solicitar el archivo actualizado a la otra máquina.

- Una vez que se realiza la sincronización, los programas que corren en las máquinas deben terminar. La siguiente vez que se desee sincronizar se debe volver a ejecutar el programa en ambas máquinas.
- Tome en cuenta que los relojes de las máquinas involucradas pueden estar desincronizados. Para determinar cuáles archivos son más recientes se debe realizar un cálculo de diferencias entre las horas de ambos relojes.
- Note que NO se deben escribir dos programas diferentes. Es el mismo programa que actúa como cliente o servidor dependiendo de los parámetros que se le pasen.
- Debe utilizar sockets para que se comuniquen los programas. Además, debe definir un formato y protocolo adecuado para enviar y recibir las solicitudes de archivos.
- Debe tomar en cuenta la posibilidad que un archivo haya sido modificado en ambas máquinas, provocando el caso de “copias en conflicto”. Aquí lo mejor es dejar las dos copias pero con nombres ligeramente diferentes.

Definición de estructuras de datos

Para este proyecto se utilizó principalmente una estructura de datos, esta tuvo el objetivo de almacenar los diferentes registros como lo es el nombre, cantidad de archivos, diferentes tiempos; tiempo de creación y tiempo de última modificación, así como un espacio dedicado para la validación de si un archivo ha sido eliminado o no en la parte del servidor.

```
struct table{
    char nombres[registro][256];
    long fechasServidor[registro];
    long fechasCliente[registro];
    char verificar[registro];
    int cantidadArchivos;
};
```

Descripción detallada y explicación de los componentes principales del servidor

El servidor básicamente tiene la función de ver cuales archivos se encuentran en la carpeta indicada y gestionar el tráfico que existe entre los clientes. Entonces se puede decir que los principales componentes del servidor son los siguientes:

Lector de la carpeta: Es un programa encargado de recorrer una carpeta en especifico, retornando los nombres de todos los archivos así como sus propiedades.

Persistencia: Es el elemento encargado de la lectura de los archivos y con base en sus fechas tomar decisiones relacionadas con la persistencia.

Gestor de peticiones: Parte del programa que se encarga de mandar los archivos, metadata y nombres de los archivos. Básicamente haciendo uso de sockets se mandan los archivos y se obtiene la confirmación de si estos llegaron obteniendo respuesta por parte del cliente.

Descripción detallada y explicación de los componentes principales del cliente

El cliente básicamente tiene el objetivo de conectarse con el servidor y basado en las peticiones que provengan de él, analizarlas juntos con los archivos que existen actualmente en la carpeta de la computadora para así poder tomar decisiones sobre qué tipo de operación deberá de realizar. Los componentes básicos del cliente son los siguientes:

Seleccionador de qué operación se debe de realizar: Es un conjunto de líneas de código que tienen el objetivo de seleccionar una operación, ya sea modificar, borrar o añadir un archivo. Por aparte también se encarga del manejo de conflictos en caso de que haya que renombrarlos para así tener una copia de cada uno de los archivos modificados. Todo esto basado en las peticiones que se tengan por parte del servidor. Simplemente se llama al método correspondiente y listo.

Comparador de fechas de última modificación: Este componente simplemente tiene el objetivo de verificar si una fecha es menor o mayor que otra, en otras palabras se puede limitar a una operación de verdadero o falso, se comparan las fechas y si es mejor se retorna verdadero.

Persistencia: Es el elemento encargado de tratar directamente con la persistencia, manipular la tabla de registros y guardar los archivos, todo esto haciendo uso de las funciones `fopen`, `fclose`, `fread` y `fwrite`. En realidad son operaciones muy triviales que no requieren de mucha explicación de como funcionan entonces se omiten los detalles.

Gestor de peticiones: Sección de código encargada del tratamiento de peticiones, más adelante se explicará el formato con el que se manejan las solicitudes, no obstante, de igual manera que en el cliente se basa en la modalidad solicitud y respuesta. Básicamente el tipo de peticiones que se tratan es recibir nombres, contenidos de archivos que los cuales se transfieren por bloques y finalmente peticiones que indican diferentes cosas como por ejemplo el indicar que ya terminó la finalización.

Manejo de sockets

El manejo de sockets se realizó con típicas librerías de C que realizan llamadas al sistema operativo. adicionalmente se hace uso de hilos, por lo que se puede concluir que el servidor puede atender varias llamadas al mismo tiempo. En el programa existen varios parámetros como lo es el directorio que se va a sincronizar y por parte del cliente que se debe de pasar por parámetro la IP del servidor para así poder sincronizar con el. Un parámetro que es fijo es el puerto, este es el 8889. También se lleva un registro de todos los descriptores

Manejo de solicitudes

Para el manejo de solicitudes se utiliza un buffer de tamaño $4000 * \text{sizeof}(\text{char})$, este no siempre se llena, pero es la medida que se utiliza generalmente para transportar las partes de los diferentes archivos. Se utiliza la modalidad de envío y recepción de mensajes por parte del cliente y servidor.

Los mensajes de confirmación son números vueltos string, por ejemplo un "1" se manda desde el cliente y en el servidor se realiza la recepción de este y luego se confirma si el mensaje se mandó correctamente o no.

Rutina de comparación

Este es uno de los puntos centrales del proyecto, debido a que si no se tiene una buena rutina de comparación entonces el resto de componentes aunque funcionen bien entonces el resultado final carecería de sentido. La manera en la que funciona la rutina de comparación es la siguiente:

Por medio de comparaciones, inicialmente se verifica si el archivo existe o no en la tabla de registros, si no existe entonces se procede a verificar la fecha del mismo (por parte del servidor) con la que se encuentra en la tabla de registros, en caso de que no haya prueba de que se modificó el que se encuentra de manera local en el cliente entonces simplemente este se sustituye y listo, en caso contrario de que esta comprobación no fuera exitosa entonces se debe de proceder a tratar el conflicto, que básicamente es crear otro archivo para que el usuario decida con cual de las copias es que debe de quedarse. Todo esto se puede evidenciar en el siguiente código:


```

send(sock, "1", 1, 0);
int indice = buscarArchivo(&tab, nombre);
int igual = 0; // 0 = archivo igual, 1 = nuevo archivo, 2 = modificacion o conflicto
//Tabla de registros-----
if(indice == -1){
    printf("Nuevo Archivo: %s\n", nombre);
    igual = 1;
}else{
    if(temp > tab.fechasServidor[indice]){
        if(getFechaArchivo(nombre) != tab.fechasCliente[indice]){
            strcat(nombre, "_");
            int indice2 = buscarArchivo(&tab, nombre);
            if(indice2 != -1){ // En caso de que ya se haya agregado entonces simplemente se modifica el archivo de conflicto
                eliminarArchivo(&tab, nombre);
                agregarArchivo(&tab, nombre, temp);
                tab.verificar[tab.cantidadArchivos-1] = 2;
            }else{ //Si no existe el archivo de conflicto entonces se agrega
                agregarArchivo(&tab, nombre, temp);
                tab.verificar[tab.cantidadArchivos-1] = 2;
                printf("Conflicto en: %s\n", nombre);
            }
        }else{
            eliminarArchivo(&tab, nombre);
            agregarArchivo(&tab, nombre, temp);
            tab.verificar[tab.cantidadArchivos-1] = 1; //Se reafirma que se visito el archivo, ya que al eliminarse se quita esta marca
            printf("Modificando: %s\n", nombre);
        }
        igual = 2;
    }else{ //Es igual, no se debe de modificar
        igual = 0;
    }
}
}
}

```

Descripción de protocolos y formatos

El protocolo que se utiliza para transmitir los datos se realiza por medio de la red, a través de la ip. Se utiliza la modalidad de solicitud y respuesta, puede que la respuesta a veces no sea significativamente útil pero se utiliza para indicar que las cosas andan bien, entonces se puede decir que son para actividades de verificación. Básicamente el formato y la forma que se utiliza para la transmisión de datos es el siguiente:

Metodo

- Primero se pasa el nombre del archivo que se va a pasar desde el servidor
- Se espera la confirmación por parte del cliente para verificar que todo esté bien
- Se pasa el tamaño del archivo
- Se espera la verificación
- Los archivos se pasan por bloques de $4000 * \text{sizeof}(\text{char})$
- Se espera verificación
- Se pregunta si aun quedan mas archivos por pasar, si sí entonces se repite todo desde el primer punto, sino se termina el ciclo

Formato

Para las actividades de verificación básicamente se pasan los siguientes datos:

“1” : Para indicar si una parte de un archivo se ha pasado bien o no o general para cualquier cosa que no sean las demás.

“n” : Para indicar que aún faltan archivos por pasar

“f” : Para indicar que ya no hay archivos.

Análisis de resultados de pruebas

La prueba que se va a realizar es la siguiente: Se realiza la sincronización del directorio “Carpeta” en ambas máquinas, esta carpeta en el lado del servidor cuenta con los siguientes archivos:

Archivo1:

Este es el contenido del archivo 1

Archivo2: contenido

ArchivoGrande: El cual tendrá muchas repeticiones de la palabra “word”

Especificación: El cual es la especificación en formato pdf de este proyecto

foto: La cual será una foto aleatoria.

Al realizar la corrida ejemplo con estos archivos se obtendrá la siguiente salida en consola:

Cliente

```
Se ha iniciado el programa en modo cliente
Directorio: carpeta
Direccion IP: 127.0.0.1
Socket created
Conectado
```

```
Se van a recibir los archivos
Se cargo la tabla registro
*****Tabla de Registros*****
*****
Recibiendo: carpeta/foto.jpg
Nuevo Archivo: carpeta/foto.jpg
Recibiendo: carpeta/archivo1
Nuevo Archivo: carpeta/archivo1
Recibiendo: carpeta/archivo2
Nuevo Archivo: carpeta/archivo2
Recibiendo: carpeta/Especificacion.pdf
Nuevo Archivo: carpeta/Especificacion.pdf
Recibiendo: carpeta/archivoGrande
Nuevo Archivo: carpeta/archivoGrande
Se termino de sincronizar
```

```
*****Tabla de Registros*****
carpeta/foto.jpg - 1497508189 1497508189 1
carpeta/archivo1 - 1497764010 1497764010 1
carpeta/archivo2 - 1497763986 1497763986 1
carpeta/Especificacion.pdf - 1496787110 1496787110 1
carpeta/archivoGrande - 1497763939 1497763939 1
*****
andrey@andrey-pc:~/Desktop/Proyecto 3/Cliente$ █
```

Servidor

```
Se ha iniciado el programa en modo servidor
Directorio: carpeta
Socket creado
Asociacion correcta
Esperando conexiones...
Conexion aceptada
Manejador asignado
carpeta carpeta
carpeta/foto.jpg
ruta Archivo::: carpeta/foto.jpg
Enviando: carpeta/foto.jpg
Se recibio exitosamente el nombre
Tamano archivo: 65341
Finalizo de pasar el archivo
```

```

*****
carpeta/archivo1
ruta Archivo::: carpeta/archivo1
Enviando: carpeta/archivo1
Se recibio exitosamente el nombre
Tamano archivo: 35
Finalizo de pasar el archivo

*****
carpeta/archivo2
ruta Archivo::: carpeta/archivo2
Enviando: carpeta/archivo2
Ocurrio un error al pasar el nombre
resp: 1002cont
Ocurrio un error al pasar el nombre
Tamano archivo: 10
Finalizo de pasar el archivo

*****

```

```

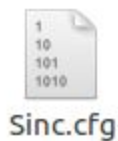
carpeta/Especificacion.pdf
ruta Archivo::: carpeta/Especificacion.pdf
Enviando: carpeta/Especificacion.pdf
Se recibio exitosamente el nombre
Tamano archivo: 89029
Finalizo de pasar el archivo

*****
carpeta/archivoGrande
ruta Archivo::: carpeta/archivoGrande
Enviando: carpeta/archivoGrande
Se recibio exitosamente el nombre
Tamano archivo: 1437697
Finalizo de pasar el archivo

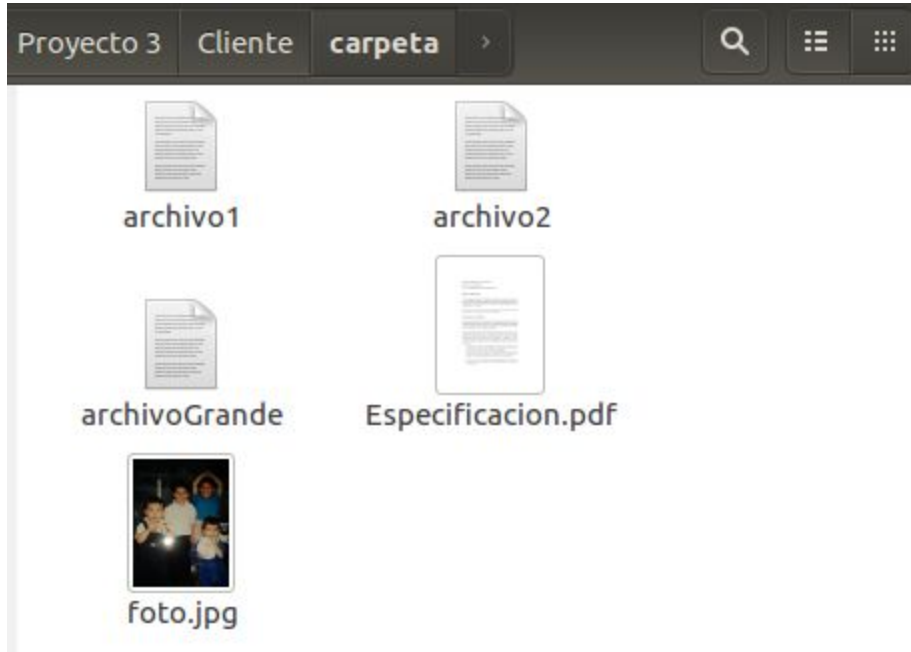
*****
Se termino de sincronizar con la conexion: 4
Cliente Desconectado

```

Al ejecutar esto se genera un archivo que contiene la información necesaria para sustituir eliminar o modificar según sean las solicitudes que se dan entre el cliente o servidor

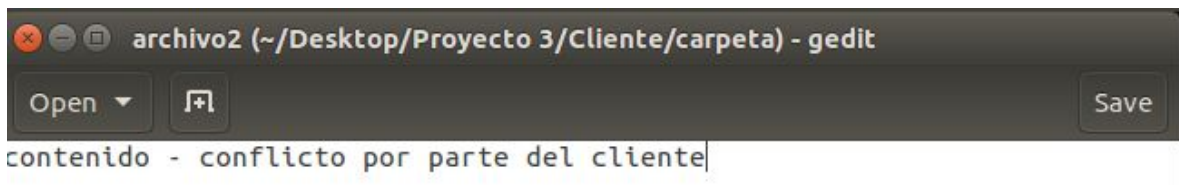


Y estos serian los archivos en el directorio



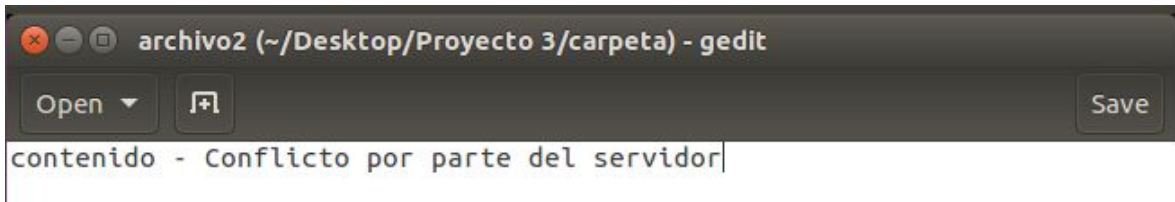
Ahora se actualizará el contenido de algunos archivos por el siguiente:

En el cliente



En el servidor





Ahora, la salida al correr de nuevo el programa con estos cambios en el directorio da la siguiente salida en la consola:

Cliente

```
Se ha iniciado el programa en modo cliente
Directorio: carpeta
Direccion IP: 127.0.0.1
Socket created
Conectado

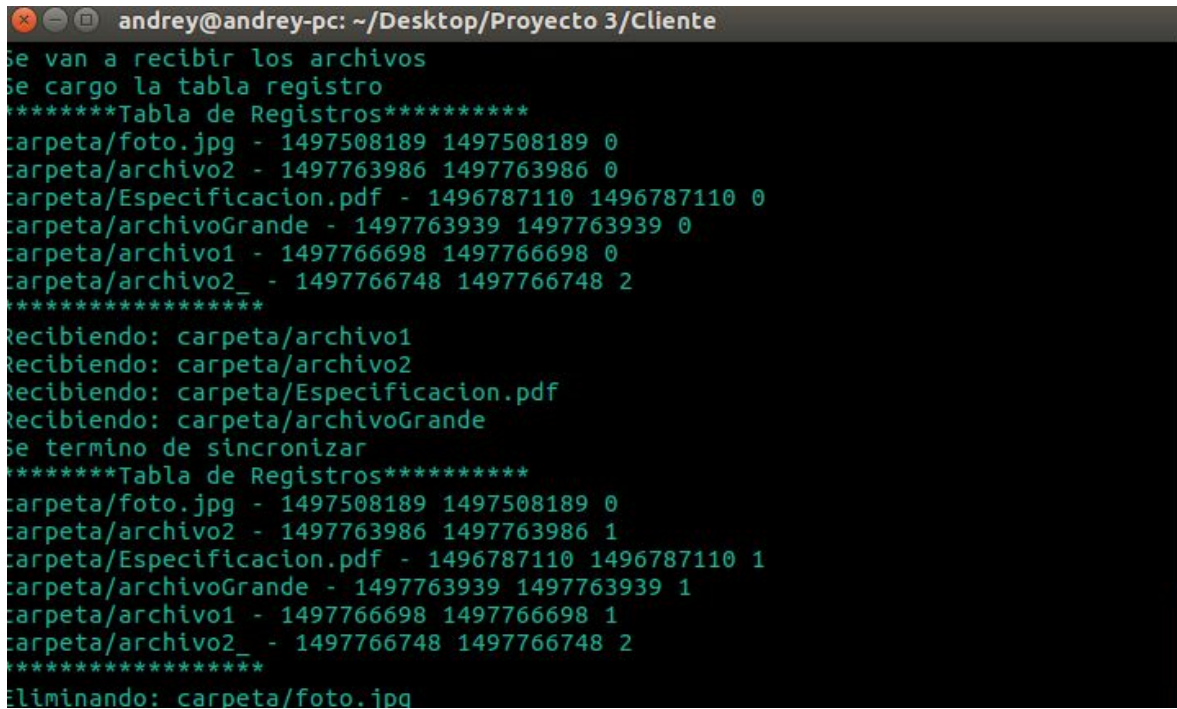
Se van a recibir los archivos
Se cargo la tabla registro
*****Tabla de Registros*****
carpeta/foto.jpg - 1497508189 1497508189 0
carpeta/archivo1 - 1497764010 1497764010 0
carpeta/archivo2 - 1497763986 1497763986 0
carpeta/Especificacion.pdf - 1496787110 1496787110 0
carpeta/archivoGrande - 1497763939 1497763939 0
*****
Recibiendo: carpeta/foto.jpg
Recibiendo: carpeta/archivo1
Modificando: carpeta/archivo1
Recibiendo: carpeta/archivo2
Conflicto en: carpeta/archivo2_
Recibiendo: carpeta/Especificacion.pdf
Recibiendo: carpeta/archivoGrande
Se termino de sincronizar
```

Se puede ver que en este caso se realizan diferentes acciones, se verifica el foto.jpg, el archivo 1 se modifica y cuando se compara archivo 2 el sistema se da cuenta de que hay una falta de integridad por parte del servidor y el cliente con respecto a la última modificación, en otras palabras se puede decir que hay un conflicto porque se realizaron cambios en ambos lados, el sistema soluciona esto agregando el símbolo “_” al nombre del archivo, para hacer referencia al archivo del servidor con el que se tiene conflicto.

Servidor

En el caso del servidor se tiene una corrida prácticamente exactamente igual a la anterior, prácticamente se tienen sólo diferencias en las corridas del lado del cliente, esto hablando en términos del registro de acciones que se plasman en la consola.

y ahora si se quiere eliminar el archivo foto.jpg entonces el estado de la consola será el siguiente:



```
andrey@andrey-pc: ~/Desktop/Proyecto 3/Cliente
Se van a recibir los archivos
Se cargo la tabla registro
*****Tabla de Registros*****
carpeta/foto.jpg - 1497508189 1497508189 0
carpeta/archivo2 - 1497763986 1497763986 0
carpeta/Especificacion.pdf - 1496787110 1496787110 0
carpeta/archivoGrande - 1497763939 1497763939 0
carpeta/archivo1 - 1497766698 1497766698 0
carpeta/archivo2_ - 1497766748 1497766748 2
*****
Recibiendo: carpeta/archivo1
Recibiendo: carpeta/archivo2
Recibiendo: carpeta/Especificacion.pdf
Recibiendo: carpeta/archivoGrande
Se termino de sincronizar
*****Tabla de Registros*****
carpeta/foto.jpg - 1497508189 1497508189 0
carpeta/archivo2 - 1497763986 1497763986 1
carpeta/Especificacion.pdf - 1496787110 1496787110 1
carpeta/archivoGrande - 1497763939 1497763939 1
carpeta/archivo1 - 1497766698 1497766698 1
carpeta/archivo2_ - 1497766748 1497766748 2
*****
Eliminando: carpeta/foto.jpg
```

Es prácticamente el mismo estado de la consola típica a excepción que al final del screenshot se puede observar donde dice: “Eliminando: carpeta/foto.jpg” lo cual obviamente indica que se eliminó el archivo nombrado.

Conclusiones sobre rendimiento y correctitud

1. El programa funciona en un 100% correctamente, por lo que se pueden pasar archivos muy grandes, vídeos, fotos, documentos debido a que la transferencia de estos se realiza por partes.
2. Más que la velocidad de las computadoras, lo que tiene más impacto en la velocidad de transferencia de archivos es la velocidad de la red.
3. El sistema está desarrollado con el objetivo de trabajar con hilos, por lo que se puede atender varias peticiones a la vez, mejorando el rendimiento siempre y cuando lo permita la red.