Universidad Simón Bolívar
Curso: Redes II
Profesor: Wilmer Pereira
Estudiantes:
- Fernando Pérez, carné: 12-11152
- Alfredo Fanghella, carné: 12-10967

Informe de proyecto

El presente informe indica, en detalle, el diseño, implementación y la implantación del proyecto de Redes II. En éste, se explica el protocolo diseñado, junto con sus respectivos diagramas, así como los módulos y estructuras utilizadas.

Protocolo

Como primera especificación del proyecto, se pidió el diseño de una pila de protocolos que permita la comunicación entre un servidor central (SC), tres servidores secundarios (SS) y varios clientes, donde solo se atienden a los primeros 5 y el resto se incluye en una lista de espera. Se requirió, además, que este protocolo fuese capaz de tolerar fallas de comunicación, caídas de alguno de los servidores secundarios, caída del servidor central y más. En esta sección, se presentan las distintas suposiciones y decisiones que se consideraron como parte del diseño.

Servidor central

Según la especificación, se decidió que el servidor central deba cumplir con una etapa de inicialización, donde la información que posee sobre la dirección y videos que sirven los servidores centrales es vacía; una etapa de inscripción, donde los servidores secundarios se inscriben ante el primario y sincronizan la información entre ellos; y por último, la etapa de funcionamiento, donde ya el servidor central está listo para atender solicitudes tanto de los clientes como de los servidores secundarios.

Según el diseño del protocolo, en la fase de inicialización y de inscripción, si no logra establecerse una conexión con los servidores secundarios o en viceversa, se considera un error fatal. Esto se decidió así para evitar situaciones en la que los clientes deseen descargar videos que aún no se tengan sincronizados y para asegurar que cuando se realicen conexiones a este servidor, este pueda proveer respuestas adecuadas porque ya todos sus componentes están inicializados.

Con respecto a las estadísticas, tanto de videos descargados por cada cliente y la cantidad de veces que un video ha sido descargado, esto es, los comandos *videos_cliente* y

Numero_desacargas_video, se diseño la aplicación para que el servidor central apuntara las distintas estadísticas y que simplemente las envíe el cliente cuando sean requeridas.

El servidor central tiene la opción de especificar la dirección IP y el puerto a través se va a comunicar con los distintos dispositivos, tanto servidores como clientes. Se supone que esta IP y este puerto son bien conocidos para el usuario.

Además de esto, el servidor central es el que se encarga de organizar la sincronización de los videos entre todos los nodos secundarios.

Servidor secundario

Los servidores secundarios son el otro tipo de servidores que posee la aplicación. En este caso, este servidor refiere a aquel que sirve a un cliente como plataforma de descarga de videos, por partes.

Si algún servidor secundario se cae, éste notifica al cliente y al servidor central, sin dar errores fatales, para que éstos estén enterados del estado de los servidores en tiempo real.

Para los diagramas se utilizó la notación utilizada en (FTP) donde los estados iniciales se denotan con la letra **C**, los estados que esperan respuesta de otros se denotan con **W**, los estados que son exitosos con la letra **E**, los estados que son fallidos con la letra **F**, y los de sincronía con **S.** Del diagrama se omiten fallos en conectividades de la red.

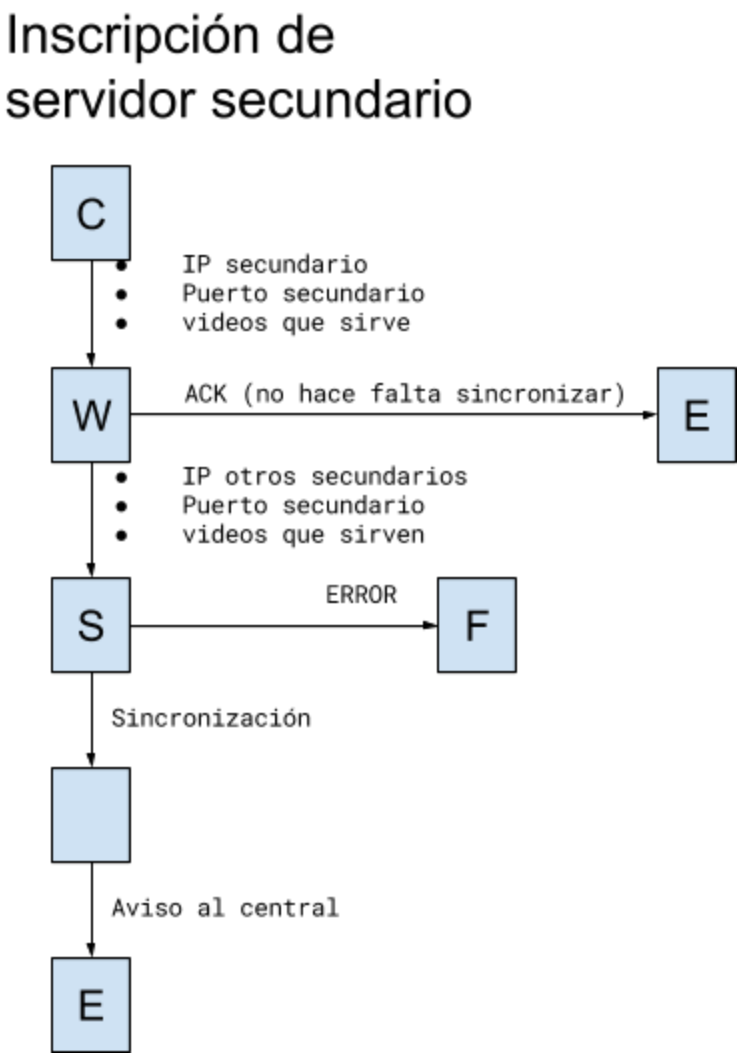

Figura 1: Muestra la máquina de estado finito que indica los pasos que sigue un servidor secundario para registrarse con el servidor central. A partir de S, muestra los pasos de la sincronía de éste con respecto a los otros.

Cliente

Inscribir un cliente

Para la inscripción de un cliente, simplemente se consideró que éste registra su nombre en la máquina local, y cada vez que envía un mensaje, tanto a un servidor secundario o al central, su nombre se envía en el mensaje para que pueda ser tomado a consideración dentro de las estadísticas.

De manera más específica, se implementó un miembro atómico que almacena el nombre del cliente en la clase Cliente que se actualiza localmente con el comando INSCRIBIR.

Listar videos

Para el cliente saber cuáles videos están aptos para su descarga, éste debe preguntarle al servidor central cuáles son. Para esto, prepara envía el comando a través de la red y espera la respuesta del servidor central

Figura 2: Muestra la máquina de estado finito que indica los pasos que sigue un servidor secundario para registrarse con el servidor central. A partir de S, muestra los pasos de la sincronía de éste con respecto a los otros.

Descargar videos

Para la descarga de un video por parte de un cliente, se obtuvo la libertad de escoger el flujo que debería seguir la aplicación. Para poder mantener al cliente *solamente* en modo cliente, es decir, que no tuviera un servidor y los inicializara,se decidió que el cliente, en primera instancia, le pregunte al servidor central cuál es la IP, puerto y videos que sirven los servidores secundarios. Cuando éste respondiera al cliente, se iban a establecer hasta tres conexiones distintas con los servidores secundarios, esto debido a que algunos de ellos pueden estar caídos y uno o más servidores deben servir las partes de los otros. Si las descargas no pueden finalizarse, se lanza un error, en cambio, si logran completarse exitosamente, se muestra un mensaje que indica que éstos se descargaron exitosamente.

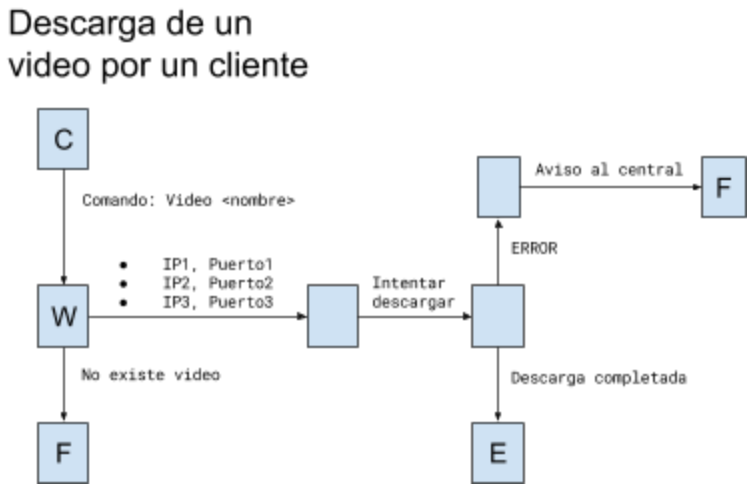

Figura 3: Muestra la máquina de estado finito que indica los pasos que sigue un servidor secundario para registrarse con el servidor central. A partir de S, muestra los pasos de la sincronía de éste con respecto a los otros.

Implementación

Arquitectura del proyecto

Para el desarrollo del proyecto, se utilizó el lenguaje Python 3y para la comunicación por red se utilizó el módulo de *sockets* así como los mensajes en formato JSON delimitados con un caracter de fín de línea *'\n'.* También se utiliza JSON para guardar los datos persistentes.

El proyecto se organizó en distintos módulos, los cuales fueron:

- **servidorbase.py:** Este archivo sirvió como clase abstracta ServidorBase para la implementación de los distintos servidores, tanto central como secundarios. Contiene métodos para iniciar un lector de comandos y un servidor TCP multihilos capaz de entender JSON concurrentes. Un método dispara el hilo con el servidor y el lector de comandos, configura un manejador de señales y pausa el hilo principal hasta que se reciba SIGINT (ctrl-C), SIGTERM o EOF (ctrl-D). Los servidores heredan de esta clase e implementan métodos para manejar los mensajes y comandos recibidos. Esta clase también implementa la persistencia de datos.
- **videocentral.py**: Este archivo se encarga de mantener la implementación del servidor central, heredando de ServidorBase e implementando métodos para el manejo de los distintos mensajes entre éste y los servidores secundarios y clientes.
- **videosecundario.py**: De manera similar al video central, provee una clase que hereda de ServidorBase e implementa distintos métodos para el manejo de los diferentes mensajes que puede recibir. Antes de pasar el control de la línea de comandos al usuario, el servidor se inscribe con el central
- **videocliente.py:** Este archivo contiene la implementación del cliente. Tiene las opciones para registrar su nombre, listar los videos y descargar uno nuevo. Maneja la línea de

comandos de forma similar a como lo hace ServidorBase, pero no ejecuta un servidor TCP. Los comandos de descarga se ejecutan en un hilo separado.

Todos los archivos proveen un menú de ayuda, accesible a través de la opción *-h* o *--help*. El resto de los parámetros tienen valores por defecto que se pueden aprovechar para simplificar la invocación de los entes del sistema (por ejemplo, si no se especifica puerto del servidor central, todos los entes toman el mismo valor por defecto para este, 50000).
Para la persistencia de los datos, se decidió mantener toda la información en memoria mientras los distintos servidores están activos, y en el caso de que éstos se caigan, el manejador de señales se encarga de escribir en disco toda la información en memoria que necesita el servidor para cuando se levante.

Los respaldos que haca cada servidor, sea central o secundario, se encuentran en los archivos *central.json*, *secundario1.json, secundario2.json, y secundario3.json,* respectivamente. Estos archivos se crean en el mismo directorio donde se ejecutan los scripts.

Los videos que puede servir cada servidor secundario están dentro de las carpetas *videos1, videos2* o *videos3*, dependiendo de si es el servidor secundario uno, dos o tres, respectivamente. Estos directorios deben estar en el mismo donde se ejecuta el script. Si no existen, el servidor secundario asume que no hay videos y los crea. Si si existen, lee automáticamente los nombres de los archivos que hay adentro.

Todos los scripts han sido probados con Python 3.4 y 3.5, y deberían funcionar con cualquier versión superior de Python 3.

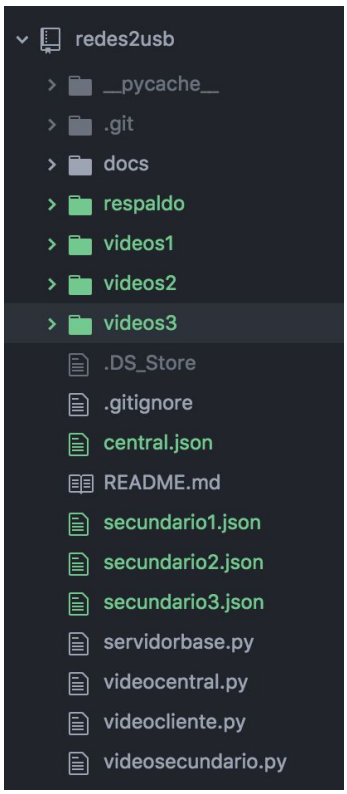

Figura 4: Una muestra de como se puede ver el directorio de donde se ejecuta el proyecto. Note que en este caso se ejecutaron todos los servidores en la misma máquina.