72.07 - Protocolos de Comunicación - TPE: Implementación de un Servidor POP3

Alumno	Legajo	Mail
Juan Adolfo Rosauer Herrmann	61240	jurosauer@itba.edu.ar
Ian James Arnott	61267	iarnott@itba.edu.ar
Lautaro Nicolas Gazzaneo	61484	lgazzaneo@itba.edu.ar

Índice:

Descripción detallada de los protocolos y aplicaciones desarrolladas

Nuestro grupo encaró primero el desarrollo del servidor POP3
con la comunicación TCP en IPV4, sabiendo que una vez confirmado el correcto funcionamiento de esta, se podría pasar a realizar de manera un poco análoga el desarrollo de TCP en IPV6 y de UDP.

Primero desarrollamos el “accept_connection_handler” para probar si podíamos generar una conexión al socket que creamos.

Luego, dividimos el problema en partes y tratamos de encarar cada una de ellas, una a la vez.

Primero, en el archivo “pop_utils” en el cual se desarrollaron funciones para parsear ciertos caracteres de especial importancia, como, el espacio, el return, el end, y también uno para el default(cualquier otro carácter).

Segundo, en el archivo “pop_stm_handlers” desarrollamos las funciones de la state machine y definimos el “read_command” y el “write_command” que leen el comando lo parsean y ejecutan el comando correspondiente, en el caso del read, y mandan la correcta respuesta al cliente dependiendo del comando utilizado en el caso del write.
También, se definen los comandos posibles de los dos estados en los que puede estar el usuario AUTHENTICATION o TRANSITION.

Tercero, en pop3.h se definen ciertas estructuras para ayudar al desarrollo de las funciones como: “connection”, esta estructura contiene los buffer de comandos y server tanto como, la stm, su último estado, el comando, el estado/data del usuario, el parser de esa conexión y el socket usado por la conexión.
“command_buff”, esta estructura contiene el file descriptor de la conexión, flags(si hubo error, si finalizo, si ok,...), una string para el comando y otra para los argumentos de este y sus respectivos índices, una string para el buffer del mail y un buffer para el “retiro” del mail.
“user_state”, contiene flags de request y autentication, el username, el username index y el inbox del usuario.

“inbox state”, contiene la mail_dir, el bite size, la dimensión, los mails, el fd del mail a “retirar” y el índice.
“mail”, contiene el path, el tamaño y una flag para ver si tiene que ser eliminado el mail.

En el pop3.c, se definen la tabla de estados de la stm, la tabla del parser y el client handler. También se describen y desarrollan las funciones de los comandos tanto las funciones de ejecución como las de write.

Para el lado del cliente, se desarrollaron 4 funciones simples derivadas de las desarrolladas anteriormente como, un client handler para aceptar la conexión, una función “parser” para analizar lo que recibimos del cliente, una función para cambiar la password de un usuario, y por último, una función para eliminar a un usuario.

Problemas encontrados durante el Diseño y la Implementación

El principal problema encontrado durante el desarrollo y la implementación fue que para desarrollar correctamente las funciones, tuvimos que hacer un seguimiento extensivo de las librerías proporcionadas por la cátedra para ver qué funciones de las librerías deberíamos usar y cómo funcionan y se relacionan entre ellas.

En el desarrollo de las state machine, el parser y las funciones de los comandos se tuvo este problema que la mayoría del tiempo fue dedicado a entender y razonar el material provisto por la cátedra y como este se relaciona con las funciones y elementos a desarrollar, como por ejemplo, la parte que más costó entender para nosotros fue la state machine, como los estados de ejecución y las transiciones entre los estados y las funciones que se ejecutan cuando esto pase, fue de una complejidad mayor que todo lo anteriormente visto, especialmente cuando se hacía debugging.

Testeo

Cuenta de testing

Incluimos un usuario *“facha”* dentro de los archivos del proyecto para poder realizar pruebas.

Conexiones concurrentes

Para poder testear que las capacidades de nuestro servidor POP3 son adecuadas. Realizamos un test de conexiones concurrentes. El test consta de dos partes, por primer lado un script Python *client_test.py* que utiliza una librería para conexión a servidores POP3 para ejecutar los comandos adecuados. Este script realiza la conexión y autenticación de nuestro usuario de prueba. Luego mediante ***STAT*** obtiene la cantidad de mails en la casilla, y procede en un ciclo infinito (con intervalos de 2 segundos) a obtener los mails mediante ***RETR***.
A este script le complementamos un script de shell *run_multiple_tests.sh* que se encarga de correrlo en forma concurrente una $1 cantidad de veces. Para cumplir con lo pedido por la cátedra, nosotros obtuvimos buenos resultados con 512 conexiones.
Hay que notar que en ese test de 512 conexiones, el servidor POP3 utilizó 3GB de RAM para su funcionamiento, una restricción a tener en cuenta para replicar el test.

Byte Stuffing

Para poder certificar que la implementación es acorde al RFC, y que los mails están libres de Byte Stuffing es decir que no hay una corrupción o pérdida de datos en nuestros mails. Simplemente descargamos algún mail utilizando *curl* y lo comparamos utilizando *diff* con nuestra copia original.
El comando *curl* se ve de esta manera:
curl pop3://localhost:1110/1 -u facha:pass > download
En nuestro testeo, la implementación funciona de forma esperada.

Limitaciones de la Aplicación

Las limitaciones de la aplicación son las siguientes como:

- Los cambios que realice un usuario solo se verán impactados en la siguiente conexión.

- El username tiene un máximo de caracteres posibles.

- Si un usuario es eliminado, mientras está conectado, este podría seguir con su conexión hasta que salga de esta. Cuando vuelva a intentar conectarse verá que su usuario fue eliminado.

Posibles Extensiones

Se consideraron las siguiente posibles extensiones para una versión 2.0 (“electric boogaloo”) del protocolo y desarrollo de servidor:

- Optimizar y aumentar la cantidad de usuarios que pueden estar presentes a la vez en el servidor.

- Mejorar la eficiencia en el searching y guardado de usuarios y contraseñas.

- Mejorar la seguridad del servidor, implementar un cifrado de usuario y contraseñas y también una mejora en cómo se guardan los mails en el servidor, así posibles hackers, no puedan acceder fácilmente a la información y mails de usuarios.

- Aumentar los buffers de escritura y lectura de los servidores, usuarios, etc.

Conclusiones

El trabajo práctico resultó ser un gran desafío para el grupo, teniendo que implementar lo visto en la cursada y analizar detalladamente lo dado por la cátedra para la resolución del proyecto.

El proyecto supuso el desarrollo de funciones y aplicaciones que nos llevaron a indagar más allá de lo visto y entendido durante el año. Analizando los archivos dados y viendo lo que servía más, como funcionaba y cómo utilizarlo correctamente, nos llevamos más conocimiento del esperado en este proyecto.

Para finalizar, este proyecto trajo consigo una cantidad de obstáculos para pasar. En el grupo estamos contentos con el producto final y su funcionalidad, consiguiendo así una experiencia para llevar adelante en siguientes materias.

Guia de instalacion y ejecucion

Posicionado en el directorio principal del proyecto. Se debe primero ejecutar el comando *make clean* para asegurar que no hayan archivos objetos que interfieran con la nueva instalación. Luego ejecutar el comando *make* para correr la instalación automática del proyecto utilizando *Makefile*.
Una vez compilado el proyecto, habrá un archivo ejecutable *main* en el directorio principal. En caso de no tener permisos de ejecución sobre el mismo puede ejecutar *chmod +x main* para otorgarlos.
Para comenzar la ejecución del servidor simplemente corra el archivo main con su configuración deseada.
./main <puerto-pop3> <puerto-cliente> -u <usuario:contraseña>
El servidor escuchará mediante IPv4 y IPv6 TCP en *puerto-pop3* para poder realizar las conexiones. Por otro lado, las conexiones para gestión correrán en UDP IPv4 y IPv6 en *puerto-cliente* .
El comando *-u* permite agregar pares *usuario:contraseña* que podrán ser utilizados por conexiones entrantes. Para concatenar diversos usuarios se debe especificar cada vez.
Es importante que en el directorio de mail (por defecto en la carpeta mail de la raíz del proyecto) se encuentre la estructura correcta. Donde para cada usuario existe un subdirectorio que se llame exactamente como su nombre de usuario que contenga la totalidad de sus mails.