



Instituto Tecnológico
de Buenos Aires

Trabajo Práctico Especial

INFORME

Protocolos de Comunicación

Primer Cuatrimestre 2023

Grupo 2

Integrantes:

Juan Cruz Bafico - Legajo 62070
jbaxico@itba.edu.ar

Franco Bosetti - Legajo 61654
fbosetti@itba.edu.ar

Gonzalo Alfredo Martone - 62141
gmartone@itba.edu.ar

Santiago Ballerini - Legajo 61746
sballerini@itba.edu.ar

Índice

<u>Introducción</u>	3
<u>Protocolos Utilizados</u>	3
<u>POP3 (RFC-1939)</u>	3
<u>TCP (RFC-793)</u>	3
<u>Simple POP3 Server Configuration and Monitoring Protocol</u>	3
<u>Simple POP3 Server Configuration and Monitoring Protocol</u>	4
<u>Introducción</u>	4
<u>Especificaciones del protocolo de monitoreo</u>	4
<u>Comandos:</u>	4
<u>Respuestas:</u>	6
<u>Token de autenticación:</u>	6
<u>Estructuras de importancia:</u>	6
<u>Utilización del servidor de monitoreo:</u>	7
<u>Aplicaciones desarrolladas</u>	8
<u>Main</u>	8
<u>Servidor POP3</u>	8
<u>Rendimiento del servidor: Buffer Size</u>	9
<u>Rendimiento del servidor: Performance</u>	10
<u>Servidor de monitoreo</u>	10
<u>Problemas encontrados</u>	11
<u>Limitaciones</u>	12
<u>Posibles extensiones</u>	12
<u>Conclusiones</u>	13
<u>Ejemplos de prueba</u>	14
<u>Instalación</u>	16
<u>Ejemplos de configuración y monitoreo</u>	17
<u>Arquitectura de la aplicación</u>	21

Introducción

El objetivo del trabajo es implementar un servidor para el protocolo POP3 (Post Office Protocol versión 3)[RFC1939] que pueda ser usado por Mail User Agents tales como Mozilla Thunderbird, Microsoft Outlook y Evolution para la recepción de correos electrónicos.

Protocolos Utilizados

POP3 (RFC-1939)

Como principal objetivo de este proyecto es llevar a cabo un servidor hecho para el protocolo POP3, se tuvo que tener en cuenta el mismo. Desde los comandos básicos disponibles hasta los estados de conexión “TRANSACTION”, “UPDATE”, entre otros, fueron tenidos en cuenta para su correcta implementación.

TCP (RFC-793)

En el RFC de POP3 está definido que este utiliza el protocolo de transporte TCP. Escuchando sus conexiones en el puerto 110 y crea una conexión entre él y el servidor una vez que un cliente se quiera conectar. Con respecto a los objetivos del proyecto, se tuvo que llevar a cabo un manejo para procesar y efectuar los comandos deseados de forma no bloqueante.

Simple POP3 Server Configuration and Monitoring Protocol

Protocolo propio desarrollado e implementado para ofrecer un servicio de administración del servidor POP3. Su especificación se encuentra en la siguiente sección y su RFC correspondiente dentro del proyecto.

Simple POP3 Server Configuration and Monitoring Protocol

Introducción

La siguiente sección describe nuestro protocolo de monitoreo del servidor POP3 para que el cliente pueda interactuar correctamente con él.

El mismo es un protocolo no orientado a conexión y utiliza TCP como protocolo de transporte.

El protocolo se basa en el intercambio de comandos y respuestas entre un cliente y un servidor. El cliente envía un comando al servidor y este responde con un header de respuesta y, en caso de ser necesario, un body. Una vez que el cliente recibe la respuesta se cierra la conexión.

Es importante aclarar que el servidor requiere de un token de autenticación para poder correr cualquier comando. Dentro de las especificaciones del protocolo, se explicará con mayor detalle el funcionamiento del protocolo.

Especificaciones del protocolo de monitoreo

Comandos:

Los comandos consisten en la address en donde se correrá el servidor, un token de autenticación seguido de un espacio y una palabra clave "case insensitive" que identifica el comando. Luego de la palabra clave se pueden enviar argumentos separados por espacios. El comando finaliza con un par CR LF.

Todo el contenido del comando debe consistir de caracteres ASCII imprimibles. Tanto el token de autenticación como los argumentos pueden contener hasta 40 caracteres. Estos comandos están especificados en el RFC del protocolo.

En cuanto al cliente que desarrollamos, los comandos disponibles para el cliente son los siguientes:

- ADD_USER
- DELETE_USER
- CHANGE_PASSWORD
- CHANGE_USERNAME
- LIST
- METRICS
- LOGS
- SET_MAX_USERS
- SET_MAX_CONNS

A continuación se encuentra una tabla donde se indican la utilidad de cada uno de ellos y si requieren argumentos.

Op Code	Nombre del comando	Funcionalidad	Argumentos
0	ADD_USER	Agrega un usuario al servidor a partir de un nombre y una contraseña provista por el cliente	2 argumentos: username pass (es importante que estén separados por espacio)
1	DELETE_USER	Elimina un usuario del servidor	1 argumento: username
2	SET_MAX_USERS	Configura la cantidad máxima de usuarios que puede tener el servidor	1 argumento: num
3	SET_MAX_CONNS	Configura la cantidad máxima de conexiones que puede tener el servidor	1 argumento: num
4	LIST	Lista los usuarios del servidor	No usa argumentos
5	METRICS	Obtiene métricas del servidor, estas son: historial de conexiones, conexiones actuales y cantidad de bytes transferidos.	No usa argumentos
6	LOGS	Obtiene los accesos de todos los usuarios durante el tiempo de ejecución del servidor	No usa argumentos
7	CHANGE_USERNAME	Actualiza el nombre de un usuario con el nuevo que le fue provisto por el cliente	2 argumentos: username new_name (es importante que estén separados por un espacio)
8	CHANGE_PASSWORD	Actualiza la contraseña de un usuario con la nueva que le fue provista por el cliente	2 argumentos: username new_pass (es importante que estén separados por un espacio)

Para obtener más información detallada acerca de cada comando, como por ejemplo ejemplos de respuesta, esta se encuentra disponible dentro de nuestro “*RFC_Monitor.txt*” dentro de la carpeta de “*docs*” de nuestro proyecto.

Respuestas:

Las respuestas consisten en un header de respuesta indicando si el comando fue ejecutado correctamente o no. El header puede tomar los siguientes valores y es terminado con un par CR LF:

- OK: El comando fue ejecutado correctamente.
- ERR: El comando no fue ejecutado correctamente.

En caso de que el comando haya sido ejecutado correctamente, y este requiera de un body, este se envía luego del header de respuesta. El body consiste en una serie de líneas de texto terminadas en un par CR LF. El body termina una vez que se recibe una línea que contenga un solo punto “.” y un par CR LF.

Token de autenticación:

En cuanto al token de autenticación, este debe ser acordado previamente entre el cliente y el servidor. El protocolo no define de qué manera se realiza el acuerdo del token de autenticación. Como mencionado anteriormente, el token de autenticación que nosotros hemos configurado para nuestro servidor es “*password_secreta*”.

Estructuras de importancia:

Dentro de todas las estructuras que utiliza el servidor de monitoreo destacaremos las siguientes:

```
typedef struct configCDT{
    unsigned max_users;
    unsigned max_conns;
}configCDT;
```

configCDT es la estructura que se utiliza para configurar los parámetros relacionados al servidor, como por ejemplo, la cantidad máxima de conexiones que permite el mismo.

```
typedef struct metricsCDT{
    unsigned historic_conns;
    unsigned current_conns;
    unsigned bytes_transf;
}metricsCDT;
```

metricsCDT es la estructura que se utiliza para guardar la información que se considera necesaria para realizar un monitoreo de forma correcta. Entre los datos que se guardan se encuentran el historial de conexiones, las conexiones actuales y la cantidad de bytes transferidos.

```
typedef struct logsCDT{
    char * username;
    time_t date_hour;
    struct logsCDT * next;
}logsCDT;
```

Por último, dentro del servidor se encuentra una lista de los logs de conexión. logsCDT contiene el nombre de usuario del cliente que se conectó y cuando lo hizo.

Utilización del servidor de monitoreo:

Para poder utilizar el servidor hay que hacer lo siguiente:

- Primero hay que tener el servidor POP3 corriendo, se puede correrlo escribiendo en la consola “./pop3_server” ya habiendo instalado el proyecto.
- Luego, el servidor ya está disponible para interactuar con él. Como mencionado anteriormente, para poder usar el servidor de monitoreo debe correr la siguiente línea de comando:

```
“./pop3_monitor $ADDR $TOKEN $COMMAND $ARGS”
```

Donde \$ADDR es la address en la cual se va a hacer la conexión con el servidor, \$TOKEN es password_secreta, \$COMMAND es alguno de los comandos disponibles del monitor y \$ARGS son los argumentos que se necesitan para el respectivo comando, separados por espacio.

Aplicaciones desarrolladas

Main

El main es el que se encarga de realizar la configuración inicial de la conexión entre el cliente y el servidor de monitoreo del servidor POP3, a partir de los argumentos recibidos, y ponerlo en funcionamiento, iniciando todas las librerías que permiten el correcto funcionamiento del servidor.

Estas librerías incluyen al selector, al manejador de usuarios y a la librería de configuración y monitoreo. El manejador de usuarios al iniciarse, carga en memoria el contenido del archivo “users.txt” localizado en la carpeta “/server/data” en donde se encuentran los usuarios con sus contraseñas.

Una vez terminada la configuración inicial, el servidor comienza a escuchar en los puertos 8888 y 8889 en busca de las conexiones de los clientes. A medida que se van conectando los clientes, los sockets de su conexión se van agregando al selector de sockets. De esta manera, podemos ofrecer una solución no bloqueante para nuestro servidor.

Servidor POP3

El servidor pop3 de nuestra aplicación maneja una estructura “client_t” donde se guardan todos los datos relevantes para la sesión de un cliente pop3. Esto incluye el fd de su socket, sus buffers de salida y entrada, un parser (que parsea los comandos del cliente), una máquina de estados (que maneja el estado de la sesión) y un manejador de su directorio de emails. Esta estructura es creada al aceptarse una conexión de pop3 e irá mutando a medida que se intercambie información entre el cliente y el servidor. La definición de la estructura es la siguiente:

```
typedef struct client {
    state_machine_t state_machine;
    parser_t parser;

    int client_sd;

    size_t response_index;
    char *response;

    char *user;

    bool closed;
    bool authenticated;
    bool response_is_allocated;

    struct buffer buffer_in;
    uint8_t buffer_in_data[BUFFSIZE];

    struct buffer buffer_out;
    uint8_t buffer_out_data[BUFFSIZE];

    // For retrieving messages
    FILE *message_file;
    bool writing_from_file;

    message_manager_t message_manager;
} client_t;
```

La autenticación del usuario se realiza mediante la librería de manejo de usuarios iniciada al levantarse el servidor. Una vez autenticado el usuario se le instancia un manejador de mails al que se le realizarán todas las consultas asociadas a los comandos de pop3, el directorio de mails del usuario se encuentra en la carpeta “server/data/maildrops/<username>”. Una vez terminada la sesión de pop3, esta estructura es liberada. En cuanto al comando de lectura de archivos, se le asigna al “client_t” un puntero a stream de archivo que contiene el output de un proceso hijo que transforma el mensaje de mail. Por el momento la transformación del mensaje de mail realiza el “byte-stuffing”. Una vez que la sesión de pop3 entra al estado UPDATE, el manejador de mails elimina todos los mails marcados con el comando DELE.

Rendimiento del servidor: Buffer Size

Por otro lado, quisimos probar distintos tamaños de *buffer* para ver cómo varían los tiempos del comando RETR dependiendo cada uno. Ya que este número tiende a ser relativamente arbitrario (dentro de lo razonable y las aproximaciones en las cuentas que se realicen), decidimos intentar con distintos tamaños posibles para probar cual sería más eficiente. A continuación, se muestran los resultados de esto.

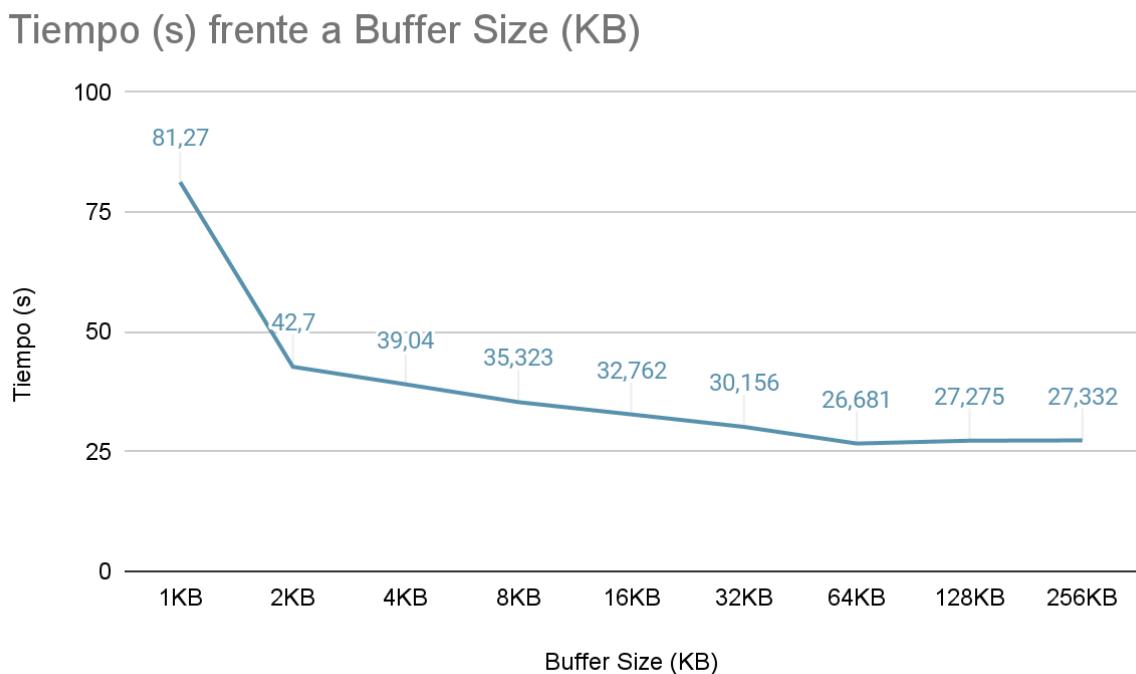


Figura 1: Gráfico de los tiempos del comando RETR para un mail de 1,7GB

En base a los resultados de esto, podemos ver que, por ejemplo, con tan solo duplicar el tamaño del buffer de 1KB a 2KB tenemos una mejoría de velocidad del 48%. De todo esto, podemos concluir que el tamaño de buffer más performante será

el de 64KB ya que al procesar un archivo de tamaño considerable fue el más performante. Además, se puede ver que a medida que se sigue incrementando el tamaño del buffer, el tiempo comienza a incrementar nuevamente. Este será el que utilizaremos en nuestro proyecto.

Rendimiento del servidor: Performance

Una de las cosas que pensamos que era importante de revisar es ¿qué pasaría si varios clientes al mismo tiempo intentan obtener mails muy pesados? ¿Afectaría esto el rendimiento del servidor?

Para probarlo creamos un archivo de 1,7GBs y lo ingresamos en la casilla de varios de los usuarios del servidor. A partir de eso empezamos a hacer pedidos de este mail mediante el comando RETR desde 1 a 4 clientes, de esta forma poniendo a prueba el servidor y mostrando el rendimiento del mismo.

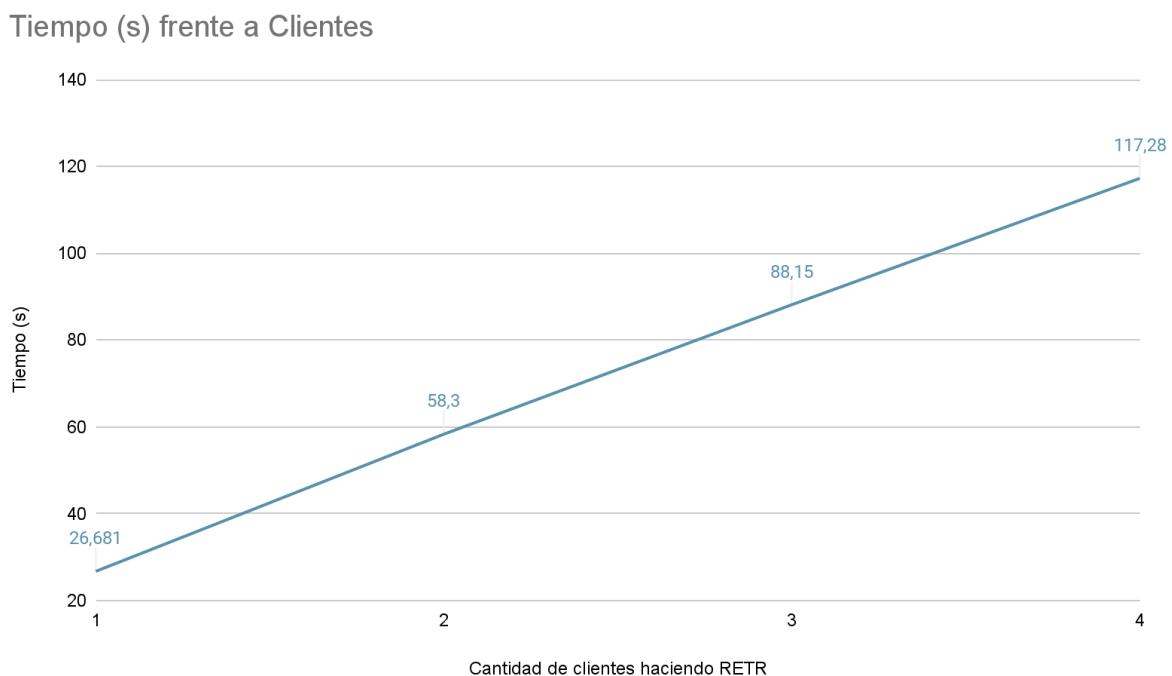


Figura 2: Gráfico de los tiempos de comandos RETR simultáneos para un mail de 1,7GB

Como podemos ver el rendimiento se va degradando linealmente, lo cual era esperado por lo que podemos concluir que tiene un uso eficaz de los recursos disponibles.

Servidor de monitoreo

El servidor de monitoreo, al igual que el servidor de pop3 maneja una estructura “monitor_client_t” donde se guardan todos los datos relevantes para la

sesión de un cliente de monitoreo. Esta estructura es de mayor sencillez que la de servidor pop3 puesto que no precisa de un parser ni de una máquina de estados ni de un manejador de mails. Simplemente contiene los buffers de entrada y salida del usuario, su socket y dos buffers donde se guarda la request del usuario y la response del servidor. La declaración de la estructura es la siguiente.

```
typedef struct monitor_client {
    int client_sd;

    size_t response_index;
    char *response;

    uint8_t request[MAX_REQUEST_LENGTH];
    size_t request_index;
    bool finished_request;

    bool closed;
    bool response_is_allocated;

    struct buffer buffer_in;
    uint8_t buffer_in_data[BUFFSIZE];

    struct buffer buffer_out;
    uint8_t buffer_out_data[BUFFSIZE];
} monitor_client_t;
```

El servidor de monitoreo realizará llamadas a funciones de la librería de monitoreo para responder a los comandos del cliente administrador. La librería de monitoreo, a su vez, realizará llamados a la librería de manejo de usuarios para los comandos relacionados con el manejo de usuarios. Esta estructura es liberada una vez que se cierra la conexión entre el cliente y el servidor de monitoreo.

Problemas encontrados

Algunos de los problemas encontrados durante el desarrollo del TPE fueron:

- Durante el comienzo del proyecto cuando se intentó de utilizar los parches provistos por la cátedra, al intentar adaptarlos correctamente para nuestro servidor hubo mucho tiempo donde se dificultó hacer funcionar estas adaptaciones
- Hubo mucho tiempo que nos dedicamos a pensar cómo debería ser la implementación de nuestro protocolo de monitoreo, discutiendo si debería ser orientado a conexión o no y sobre qué protocolo de transporte debería trabajar. Al final, como mencionamos anteriormente, se decidió implementarlo de tal forma que no sea orientado a conexión y que funcione sobre TCP.

Decidimos definir el protocolo de esta manera debido a que no encontramos la necesidad de mantener una sesión abierta entre el cliente y el servidor. En cuanto a la autenticación de un usuario que debe poder acceder al servicio se puede resolver con un token de autenticación definido previamente.

Limitaciones

- Una de las limitaciones de nuestro proyecto es que al correr el comando RETR de POP3, este bloquea temporalmente el read. Esto ocurre porque leemos directamente del stream del pipe del proceso hijo de transformación de mensajes. Manejar la disponibilidad de lectura del pipe con select sería una implementación más eficiente.
- Actualmente no es posible configurar ni la ubicación del directorio de mails de usuarios ni la ubicación del archivo donde estos son persistidos.
- Otra funcionalidad no configurable es el comando corrido por el proceso de transformación de mensajes puesto que este debe ser editado en el código fuente.
- Las métricas del servidor son volátiles puesto que no son persistidas en ningún archivo, pese a que la consigna no exige cumplir con este requerimiento creemos que una implementación persistente de las métricas y los logs sería de mayor utilidad.

Posibles extensiones

- Permitir cambios en *Runtime* sobre más parámetros relacionados al servidor, como por ejemplo la ubicación del archivo donde persistirán los usuarios, la ubicación de los directorios de mails e incluso el comando del proceso hijo de transformación de mensajes. Una alternativa menos ambiciosa podría ser permitir establecer estos parámetros como argumentos del programa main del servidor.
- Guardar métricas individuales para cada usuario, como por ejemplo la cantidad de veces que se conectó, cuando se dió de alta, entre otras. A su vez, también se podría guardar la cantidad de mails leídos y borrados.
- Realizar una implementación no bloqueante de la lectura de la salida del proceso de transformación de mensajes puesto que este es uno de los grandes problemas de performance que contiene nuestro proyecto.

Conclusiones

Para concluir, luego de haber realizado el TPE entre los integrantes del grupo llegamos a la conclusión de que a lo largo del desarrollo del proyecto es de vital importancia tener un buen manejo de los temas que fueron enseñados durante la cursada. De esta forma se puede avanzar a un paso más rápido en el desarrollo del mismo y no quedarse sobre el mismo problema por un largo tiempo. Se lograron abordar las problemáticas y decisiones dadas por la cátedra y discutidas en las clases de consulta de una forma desarrollada y colaborativa. Además realizar este proyecto nos dio la posibilidad de comprender de mejor manera las decisiones que se tienen que tener en cuenta a la hora de desarrollar tu propio protocolo.

El TPE estuvo lleno de dificultades, problemas y obstáculos para lograr los objetivos del mismo pero sentimos que hemos logrado crear un proyecto con el cual nos sentimos satisfechos.

Ejemplos de prueba

Para inicializar el servidor, se corre ./pop3_server:

```
ubuntu@primary:/Users/francobosetti/Documents/ITBA/3er Año/2do Cuatrimestre/Protocolos de Comunicacion/TP-Protos$ ./pop3_server
DEBUG: src/user-manager.c:616, Loaded user jbafico
DEBUG: src/user-manager.c:618, User count: 1
DEBUG: src/user-manager.c:616, Loaded user fbosetti
DEBUG: src/user-manager.c:618, User count: 2
DEBUG: src/user-manager.c:616, Loaded user sballerini
DEBUG: src/user-manager.c:618, User count: 3
DEBUG: src/user-manager.c:616, Loaded user gmartone
DEBUG: src/user-manager.c:618, User count: 4
INFO: src/server.c:178, === [SERVER STARTED] ===
INFO: src/server.c:179, Listening on port 8888
INFO: src/server.c:180, Max queued connections is 10
INFO: src/server.c:181, Attending a maximum of 500 clients
```

Para conectarse al servidor POP3 utilizando Netcat:

```
ubuntu@primary:/Users/francobosetti$ nc -C 192.168.64.2 8888
+OK POP3 SERVER READY
```

Para autenticarse dentro del servidor se utilizan los comando USER y PASS:

```
+OK POP3 SERVER READY
USER fbosetti
+OK
PASS 1234
+OK AUTHORIZED
```

En caso de que el usuario no exista, aparece el siguiente error:

```
+OK POP3 SERVER READY
USER francoo
+OK
PASS 1
-ERR NO SUCH USER
```

Por otro lado, en caso de que la contraseña sea inválida aparece el siguiente error:

```
USER fbosetti
+OK
PASS 3333
-ERR INVALID PASSWORD
```

Comando CAPA (en estado GREETING)

```
CAPA  
+OK  
CAPA  
USER  
PIPELINING
```

Comando LIST cuando no hay mails disponibles:

```
LIST  
+OK NO MESSAGES IN MAILDROP
```

Comando LIST con mails disponibles:

```
list  
+OK SCAN LISTING FOLLOWS  
1 3397  
2 3403
```

Comando LIST de un mail existente:

```
list 1  
+OK 1 3397
```

Comando LIST pasándole como argumento el número de un mail inexistente:

```
LIST 67  
-ERR NO SUCH MESSAGE
```

Comando RETR pasándole como argumento un número de mail existente:

```
retr 1  
+OK HERE COMES THE MESSAGE  
Cras cursus mattis est, in gravida est vulputate id. Nam quis finibus libero, quis gravida elit. In nec fermentum nibh, vel dictum turpis. Maecenas lorem ante, dictum eu sapien nec, vestibulum lobortis augue. Nullam convallitudin interdum néque ut sagittis. Praesent porttitor bibendum turpis, nec porttitor elit tristis que ut. Curabitur elementum ante arcu, suscipit egestas felis aliquet ac. Donec facilisis sollicitudin dolor sed fermentum. Vivamus dapibus urna lobortis iaculis venenatis. Mauris quis risus elit. Mauris at velit eget tortor venenatis euismod eu ultricies tellus. Maecenas ultrices nisi nec pretium pulvinar. Phasellus sagittis velit eu dignissim semper. Proin vel nibh justo.  
Suspendisse sit amet dui lacinia justo malesuada cursus nec et arcu. Suspendisse potenti. Cras sit amet cursus nisi. Nunc consequat velit ac fringilla porta. Fusce dignissim libero ut nisl gravida varius. Nulla sollicitudin pretium mi, et interdum libero facilisis nec. Fusce vel leo ut sem finibus finibus. Morbi commodo dapibus elit at lacinia. Donec commodo finibus massa, commodo semper urna ornare molestie. Morbi quis risus nunc. Aliquam convallis nibh tincidunt, varius quam ac, rhoncus lorem.  
Vestibulum vitae dolor ultrices, aliquet nisi ac, varius tortor. Phasellus in tempor nulla. Quisque hendrerit magna eros, a hendrerit elit rutrum vitae. Mauris maiores augue, sagittis elementum sagittis et, ultricies eget magna. Nunc auctor ex urna, vitae tincidunt justo porta sed. Curabitur non neque semper, dignissim urna ac, rhoncus turpis. Etiam scelerisque nibh a nulla vulputate condimentum. Nullam luctus lorem vel dapibus aliquet. Donec egestas nunc sed nisl ultricies, quis rutrum dolor hendrerit. Suspendisse aliquet, purus ut tristique eleifend, ligula lectus dictum nisi, sed molestie enim nisl quis elit. In facilisis posuerunt dui, a malesuada justo pretium et. Nulla cursus magna at mauris egestas, vel dapibus odio iaculis. Fusce sed felis diam. Vivamus interdum lacinia gravida. Sed in eros ac nunc rhoncus posuere sed ullamcorper metus.  
0  
In hac habitasse platea dictumst. Nullam sollicitudin, quam ac viverra rhoncus, nunc quam faucibus dolor, non commodo tellus dolor non augue. Cras eget cursus quam. Etiam a eros id arcu dapibus faucibus. Aenean pretium lectus dapibus turpis fringilla rutrum vel id nulla. Aliquam dapibus, purus a consectetur interduum, massa justo ultricies turpis, at accumsan nisl lectus eget magna. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. In ornare dolor eget posuere convallis. Aliquam venenatis nulla enim, non pharetra velit maximus volutpat. Aliquam molestie nisl nec odio consequat, eu lobortis nunc malesuada. Etiam diam leo, semper quis odio. Donec a lacus et eros dapibus sagittis et in nisi. Nam consequat erat lacus, quis dapibus justo maximus in. Nullam varius malesuada rhoncus.  
Quisque laoreet iaculis tortor, tincidunt auctor turpis blandit non. Aenean dui magna, elementum eget turpis sed, vulputate sagittis massa. Aliquam quis liber ex. Proin mollis ipsum sapien, eu vestibulum odio ornare eu. Donec a risus rhoncus, feugiat mi id, feugiat enim. Quisque tincidunt varius tincidunt. In urna tortor, luctus viverra congue vitae, sagittis eu mauris. Etiam lobortis cursus magna, vel maximus erat aliquam quis. Nulla ullamcorper metus quis nisl placerat venenatis et sed ex.
```

Comando NOOP:

```
[NOOP  
+OK
```

Comando STAT:

```
stat  
+OK 2 6800
```

Comando QUIT:

```
[QUIT  
+OK QUITING, BYE :)
```

Pipelining:

```
unicacion/TP-Protos  
$ printf "USER fbosetti\nPASS 1234\nLIST\nSTAT\nRETR 1\nsoycomandinvalido\nDELE 68\nQUIT\n" |  
netcat -C 192.168.64.2 8888  
+OK POP3 SERVER READY  
+OK  
+OK AUTHORIZED  
+OK NO MESSAGES IN MAILDROP  
+OK 0 0  
-ERR INVALID MESSAGE NUMBER  
-ERR UNKNOWN COMMAND  
-ERR NO SUCH MESSAGE  
+OK QUITING, BYE ::)
```

Instalación

Para poder instalar el proyecto, usted debe pararse sobre la carpeta raíz del proyecto y ejecutar el comando “*make all*”. Esto generará los ejecutables correspondientes, estos son “*pop3_server*” y “*pop3_monitor*”.

```
jbafileco@DESKTOP-QOLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023 1Q/Protos/TP-Protos$ make all
```

Ahora basta con correr el servidor POP3 y con este en ejecución correr el comando deseado con el monitor si así se desea o conectarse al servidor POP3 directamente para utilizarlo.

Ejemplos de configuración y monitoreo

A continuación veremos ejemplos de la utilización de los comandos del servidor de monitoreo explicados anteriormente. Recordar que los comandos se corren de la siguiente manera:

```
“./pop3_monitor $ADDR $TOKEN $COMMAND $ARGS”
```

Comando ADD_USER

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -ADD_USER juansito password  
OK
```

Agrega un usuario al servidor

Comando DELETE_USER

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -DELETE_USER juansito  
OK
```

Elimina un usuario del servidor

Si se le pasa un usuario que no existe dentro del servidor esta es la salida:

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -DELETE_USER juansito  
ERR
```

Comando CHANGE_PASSWORD

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -CHANGE_PASSWORD juansito password_nueva  
OK
```

Actualiza la contraseña de un usuario con la nueva que le fue provista por el cliente

Si se le pasa un usuario que no existe dentro del servidor esta es la salida:

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -CHANGE_PASSWORD flores password_nueva  
ERR
```

Comando CHANGE_USERNAME

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -CHANGE_USERNAME juansito juanBafico  
OK
```

Actualiza el nombre de un usuario con el nuevo que le fue provisto por el cliente

Si se le pasa un usuario que no existe dentro del servidor esta es la salida:

```
jbafico@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023_1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_s  
ecreta -CHANGE_USERNAME juansito juanBafico  
ERR
```

Comando LIST

```
jbafileco@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023 1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_secreta -LIST
OK
juanBafico
jbafileco
fbosetti
sballerini
gmartone
.
```

Lista los usuarios del servidor

Comando METRICS

```
Protocolos de Comunicacion/TP-Protos$ ./pop3_monitor localhost password_secreta -METRICS
OK
1
9
593
```

Comando LOGS

```
Protocolos de Comunicacion/TP-Protos$ ./pop3_monitor localhost password_secreta -LOGS
OK
fbosetti 2023-06-24 11:49:02
gmartone 2023-06-24 11:52:46
fbosetti 2023-06-24 11:53:11
jbafileco 2023-06-24 11:53:25
jbafileco 2023-06-24 11:53:41
jbafileco 2023-06-24 11:53:52
sballerini 2023-06-24 11:54:04
fbosetti 2023-06-24 11:54:11
fbosetti 2023-06-24 11:55:16
```

Obtiene los accesos de todos los usuarios durante el tiempo de ejecución del servidor

Comando SET_MAX_USERS

```
Protocolos de Comunicacion/TP-Protos$ ./pop3_monitor localhost password_secreta -SET_MAX_USERS 5
OK
```

Configura la cantidad máxima de usuarios que puede tener el servidor

Comando SET_MAX_CONNS

```
Protocolos de Comunicacion/TP-Protos$ ./pop3_monitor localhost password_secreta -SET_MAX_CONNS 1
OK
```

Configura la cantidad máxima de conexiones que puede tener el servidor

Caso en el que le pasen un comando incorrecto:

```
jbafileco@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023 1Q/Protos/TP-Protos$ ./pop3_monitor localhost password_secreta -COMANDOINEXISTENTE
Invalid command or arguments
```

Monitoreo de iniciación correcta del servidor de pop3:

```
jbafileo@DESKTOP-QQLTQKQ:/mnt/c/Users/juanc/Documents/ITBA/2023 1Q/Protos/TP-Protos$ ./pop3_server
DEBUG: src/user-manager.c:616, Loaded user juanBafico
DEBUG: src/user-manager.c:618, User count: 1
DEBUG: src/user-manager.c:616, Loaded user jbafileo
DEBUG: src/user-manager.c:618, User count: 2
DEBUG: src/user-manager.c:616, Loaded user fbosetti
DEBUG: src/user-manager.c:618, User count: 3
DEBUG: src/user-manager.c:616, Loaded user sballerini
DEBUG: src/user-manager.c:618, User count: 4
DEBUG: src/user-manager.c:616, Loaded user gmartone
DEBUG: src/user-manager.c:618, User count: 5
INFO: src/server.c:178, === [SERVER STARTED] ===
INFO: src/server.c:179, Listening on port 8888
INFO: src/server.c:180, Max queued connections is 10
INFO: src/server.c:181, Attending a maximum of 500 clients
```

Ahora pasaremos a mostrar ejemplos de monitoreo dentro del servidor de POP3, principalmente mediante logs que lo muestran.

Monitoreo de conexión del monitor al servidor pop3:

```
INFO: src/monitor-server.c:59, monitor client connection with sd:4 accepted
DEBUG: src/monitor-server.c:88, monitor reading on sd:4
DEBUG: src/monitor-server.c:108, recv 512 bytes from monitor sd:4
DEBUG: src/monitor-server.c:121, read p from monitor sd:4
DEBUG: src/monitor-server.c:121, read a from monitor sd:4

DEBUG: src/monitor-parser.c:11, parsing client request
DEBUG: src/monitor-parser.c:38, token: LISTUSERS
DEBUG: src/monitor-commands.c:158, Handling listusers command
DEBUG: src/monitor-server.c:144, monitor writing on sd:4
INFO: src/monitor-server.c:161, sent 60 bytes to monitor sd:4
INFO: src/monitor-server.c:73, monitor client connection with sd:4 disconnected
```

Podemos ver bien como el monitor se conecta al servidor, le envía el mensaje del cliente ya parseado por el monitor, luego el servidor de pop3 lo parsea por su cuenta y resuelve el comando pedido, le envía la respuesta al monitor para luego cerrar la conexión con el mismo

Monitoreo de conexión de un cliente al servidor pop3:

```
DEBUG: src/monitor.c:146, Current connections: 1
DEBUG: src/monitor.c:147, Historic connections: 1
INFO: src/pop3.c:148, client connection with sd:4 accepted
INFO: src/states/greeting.c:37, sent 23 bytes to sd:4
```

Podemos ver como el servidor entró en el estado de greeting luego de recibir al cliente.

Monitoreo de comando del cliente en el estado de authorization al servidor de POP3:

```
DEBUG: src/states/states-common.c:54, state:authorization reading on sd:4
DEBUG: src/states/states-common.c:73, recv 6 bytes on state:authorization from sd:4
DEBUG: src/states/states-common.c:99, state:authorization writing on sd:4
INFO: src/states/states-common.c:117, sent 32 bytes on state:authorization to sd:4
```

Monitoreo de sesion iniciada por el cliente en el servidor de POP3:

```
DEBUG: src/monitor.c:146, Current connections: 1
DEBUG: src/monitor.c:147, Historic connections: 1
INFO: src/pop3.c:148, client connection with sd:4 accepted
INFO: src/states/greeting.c:37, sent 23 bytes to sd:4
DEBUG: src/states/states-common.c:54, state:authorization reading on sd:4
DEBUG: src/states/states-common.c:73, recv 13 bytes on state:authorization from sd:4
DEBUG: src/states/states-common.c:54, state:authorization reading on sd:4
DEBUG: src/states/states-common.c:73, recv 2 bytes on state:authorization from sd:4
DEBUG: src/states/states-common.c:99, state:authorization writing on sd:4
INFO: src/states/states-common.c:117, sent 5 bytes on state:authorization to sd:4
DEBUG: src/states/states-common.c:54, state:authorization reading on sd:4
DEBUG: src/states/states-common.c:73, recv 9 bytes on state:authorization from sd:4
DEBUG: src/states/states-common.c:54, state:authorization reading on sd:4
DEBUG: src/states/states-common.c:73, recv 2 bytes on state:authorization from sd:4
DEBUG: src/monitor.c:128, New log: fbosetti
DEBUG: src/states/states-common.c:99, state:transaction writing on sd:4
INFO: src/states/states-common.c:117, sent 16 bytes on state:transaction to sd:4
```

Como podemos ver, el cliente se encuentra en el estado de authorization y luego de iniciar sesion, donde se puede ver que inicio sesion en “*New log: fbosetti*”, vemos como el cliente entra al estado de transaction.

Monitoreo de comando del cliente en el estado de transaction al servidor de POP3:

```
DEBUG: src/states/states-common.c:54, state:transaction reading on sd:4
DEBUG: src/states/states-common.c:73, recv 6 bytes on state:transaction from sd:4
DEBUG: src/states/states-common.c:54, state:transaction reading on sd:4
DEBUG: src/states/states-common.c:73, recv 2 bytes on state:transaction from sd:4
DEBUG: src/states/states-common.c:99, state:transaction writing on sd:4
INFO: src/states/states-common.c:117, sent 29 bytes on state:transaction to sd:4
```

Monitoreo de la desconexión del cliente del servidor POP3 usando el comando QUIT:

```
DEBUG: src/states/states-common.c:73, recv 4 bytes on state:transaction from sd:4
DEBUG: src/states/states-common.c:54, state:transaction reading on sd:4
DEBUG: src/states/states-common.c:73, recv 2 bytes on state:transaction from sd:4
INFO: src/states/update.c:65, sent 21 bytes to sd:4
INFO: src/pop3.c:214, closing client with sd:4
DEBUG: src/monitor.c:152, Current connections: 0
INFO: src/pop3.c:161, client with sd:4 disconnected
INFO: src/pop3.c:214, closing client with sd:4
DEBUG: src/monitor.c:152, Current connections: 0
```

Arquitectura de la aplicación

El siguiente gráfico muestra la arquitectura de nuestro servidor. Los clientes pop3 se conectan al puerto 8888 donde tendrán su sesión de pop3. En cambio, los usuarios administradores se conectan al puerto 8889 donde podrán configurar el servidor de monitoreo mediante nuestro protocolo. También se puede ver como los servidores interactúan con las librerías.

