72.07 - Implementación de un servidor POP3 - Informe

Alumno	Legajo	Mail
Alejo Flores Lucey	62622	afloreslucey@itba.edu.ar
Andrés Carro Wetzel	61655	acarro@itba.edu.ar
Nehuén Gabriel Llanos	62511	nllanos@itba.edu.ar

Tabla de Contenidos

- Tabla de Contenidos
- **Descripción detallada de los protocolos y aplicaciones desarrolladas**
- **Problemas encontrados durante el diseño y la implementación**
- **Pruebas de integridad del servidor**
  - Correcto funcionamiento del *Byte Stuffing*
  - Correcto funcionamiento al trabajar con un archivo de gran tamaño
  - Correcto funcionamiento al recibir una gran cantidad de conexiones simultaneas
- **Limitaciones de la aplicación**
  - Posibles extensiones
- **Conclusiones**
- **Guía de instalación**
  - Compilación y creación de los ejecutables
  - Utilización del directorio de mails
  - Ejecución de las aplicaciones servidor y cliente
- **Ejemplos de configuración y monitoreo**
  - Comandos que no requieren token de autenticación
    - Mensaje de ayuda
    - Versión del protocolo diseñado
  - Comandos que requieren token de autenticación
    - Cambiar el directorio
    - Agregar un usuario al servidor

Cambiar la contraseña de un usuario
Eliminar un usuario
Listar los usuarios
Estadísticas
Cambiar el máximo número de mails

Descripción detallada de los protocolos y aplicaciones desarrolladas

Inicialmente, se diseñó y desarrolló un protocolo de aplicación que está pensado para trabajar sobre el protocolo de transporte UDP. Entre sus características principales encontramos que es no orientado a conexión y no es confiable.

Además, cuenta con una serie de comandos que le permite consultar estadísticas volátiles del servidor, entre las que podemos encontrar conexiones históricas, la lista de usuarios presentes en el mismo, bytes transferidos y conexiones concurrentes. Asimismo, ofrece la posibilidad de cambiar la contraseña de un usuario, agregar un usuario, remover un usuario, cambiar el directorio de mail y cambiar la cantidad máxima de mails que puede tener un usuario en el servidor. Todos estos comandos se encuentran presentes en el RFC de `Turtle Protocol v1`.

Por otro lado, se desarrolló un ejecutable servidor y una aplicación cliente.

La primera de ellas está formada por dos secciones, la encargada de las conexiones del cliente y la encargada de las conexiones al servidor POP3. En la sección del cliente se realiza el *parsing* de las peticiones realizadas por la aplicación cliente a través protocolo `Turtle Protocol v1`. En la sección del servidor, se ha desarrollado todo lo necesario para que este responda como un servidor POP3, por lo que se lo puede usar como tal sin ningún problema.

Por último, la aplicación cliente ofrece una manera intuitiva de utilizar el protocolo `Turtle Protocol v1` para administrar y

monitorear el servidor, a través de la linea de comando.

Problemas encontrados durante el diseño y la implementación

Los problemas encontrados durante el diseño y la implementación fueron varios. Al iniciar el trabajo práctico es dónde aparecieron la mayoría de las dificultades, principalmente porque había que entender correctamente el flujo de los llamados de *handlers* y cómo funcionan cada uno de los estados.

El parser también generó sus complicaciones puesto que había que tener en cuenta muchos casos límite y también saber que evento devolver dependiendo del carácter que recibamos. Debimos ser rigurosos con los contenidos de los RFCs relacionados y seguirlos a rajatabla.

Otra de los problemas encontrados durante el diseño surgió cuando se cortaba la conexión entre el servidor y el usuario de manera abrupta, mediante CTRL+C, lo que generaba que se caiga en un estado inconsistente. Hubo que analizar los diferentes estados de la máquina para determinar cuando realizar la baja del *file descriptor* en el *selector* y su correspondiente cierre.

Por último, la creación del protocolo tuvo su complejidad dado que cuando se realizó la revisión del mismo con la cátedra, se nos presentaron una serie de errores que no habíamos siquiera analizado. Entre ellos podemos encontrar a la falta de escalabilidad del mismo, la escasez de especificación en su respectiva definición y la ausencia de comandos en el protocolo.

Todos estos problemas pudieron ser sorteados y se logró una implementación que consideramos correcta.

Pruebas de integridad del servidor

Correcto funcionamiento del *Byte Stuffing*

Se probó el funcionamiento de esta funcionalidad obligatoria descripta en el RFC 1939. Para comprobar su correctitud, se le pidió al servidor que retorne el siguiente mail:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut id ligula leo. Nunc finibus eu augue nec efficitur. Vestibulum porta ac arcu in consectetur. Vivamus justo neque, consequat sed mauris eu, congue fermentum turpis. Nam ut accumsan ipsum. Mauris suscipit tempor mauris, nec congue neque consectetur quis. Fusce lectus massa, mattis ac nibh in, feugiat dapibus eros. Duis vitae eleifend nibh. Aenean id odio auctor, semper arcu et, fermentum libero.

Vestibulum a congue metus. Praesent mattis ex non imperdiet tempor. Ut vitae turpis quis quam dignissim dignissim ut ac sem. Phasellus finibus sodales nisl, at rutrum justo. Donec eu ligula risus. Pellentesque aliquet nisi arcu, sed pulvinar libero commodo id. Mauris mollis, odio in efficitur dictum, dui lacus congue neque, vitae consectetur purus leo sit amet erat.

Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Cras eget consectetur libero, ut sollicitudin sapien. Nunc vel feugiat libero, non aliquam ipsum. Nunc tempus vel odio non elementum. Phasellus a nisi ornare, lacinia urna eget, aliquam ex. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

.Aquí se debe hacer Byte Suffing

.

Arriba tambien se debería haber hecho Byte Stuffing

Se corrió el siguiente comando para verificar el funcionamiento:

```
$$ cat prueba1 | ncat -C localhost 61655 > prueba1_output
```

donde `prueba1` contiene los comandos para retornar el mail en cuestión.

Analicemos la salida del comando, presente en el archivo `prueba1_output`:

```
+OK POP3 server ready
+OK Valid mailbox
+OK Logged in and mailbox ready
+OK message follows
Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Ut id ligula leo. Nunc finibus eu augue nec
efficitur. Vestibulum porta ac arcu in consectetur.
Vivamus justo neque, consequat sed mauris eu, congue
fermentum turpis. Nam ut accumsan ipsum. Mauris
suscipit tempor mauris, nec congue neque consectetur
quis. Fusce lectus massa, mattis ac nibh in, feugiat
dapibus eros. Duis vitae eleifend nibh. Aenean id odio
auctor, semper arcu et, fermentum libero.


Vestibulum a congue metus. Praesent mattis ex non
imperdiet tempor. Ut vitae turpis quis quam dignissim
dignissim ut ac sem. Phasellus finibus sodales nisl, at
rutrum justo. Donec eu ligula risus. Pellentesque
aliquet nisi arcu, sed pulvinar libero commodo id.
Mauris mollis, odio in efficitur dictum, dui lacus
congue neque, vitae consectetur purus leo sit amet
erat.


Orci varius natoque penatibus et magnis dis parturient
montes, nascetur ridiculus mus. Cras eget consectetur
libero, ut sollicitudin sapien. Nunc vel feugiat
libero, non aliquam ipsum. Nunc tempus vel odio non
elementum. Phasellus a nisi ornare, lacinia urna eget,
aliquam ex. Vestibulum ante ipsum primis in faucibus
orci luctus et ultrices posuere cubilia curae; Class
aptent taciti sociosqu ad litora torquent per conubia
nostra, per inceptos himenaeos.


..Aquí se debe hacer Byte Suffing
```

```
..

Arriba tambien se debería haber hecho Byte Stuffing

.
+OK Have a nice day
```

Lo que se puede ver marcado en **negrita** es el contenido del archivo anteriormente mencionado, sumado al octeto de terminación y la respuesta del servidor. Como se puede observar, las líneas que comienzan con el octeto de terminación fueron prefijadas con otro octeto de terminación, como lo indica el RFC.

Así se concluye que el *Byte Stuffing* funciona correctamente.

Correcto funcionamiento al trabajar con un archivo de gran tamaño

Se utilizó el siguiente comando para crear un archivo de 1 GB, que luego será utilizado para hacer los testeos correspondientes de esta sección.

```
$$ dd if=/dev/urandom of=archivo2 bs=4096 count=250000
$$ base64 archivo2 > archivo2_base64
```

Se corrió el siguiente comando para verificar el funcionamiento:

```
$$ curl pop3://alejo:pp@localhost:61655/2 > prueba2_output
```

Luego, se corrió el comando `diff` para verificar que el contenido del archivo `prueba2_output` y el del archivo `archivo2_base64` (el archivo que se encuentra en el *mail directory*) sea el mismo.

```
$$ diff -u prueba2_output mail/alejo/cur/archivo2_base64
--- prueba2_output  2023-06-23 16:15:50.711116091 +0000
+++ mail/alejo/cur/archivo2_base64  2023-06-23 16:14:29.082231903 +0000
@@ -17964911,4 +17964911,3 @@
 pZ8YtATfuv0eK/M+gd2TTuS3GtDfDgehV+dP6DD5GFdum0CUfXLAI36NsQgCHx7BCoxJLqDG0EzG
 ONMwAV8Z1DX68ONhydUBwzCRObwyvZPX4YbheklPziCvHyX5kRp5VyQ+FdSdV6pbqBDqFyOSC5VW
 esp8i8J17vcMIIO1u1iAYw==
-
```

Se está comparando el archivo `prueba2_output` con el archivo ubicado en `mail/alejo/cur/archivo2_base64`.

Luego, se muestran las diferencias entre los archivos proporcionados. Se puede observar que hay solo una diferencia:

```
@@ -17964911,4 +17964911,3 @@
 pZ8YtATfuv0eK/M+gd2TTuS3GtDfDgehV+dP6DD5GFdum0CUfXLAI36NsQgCHx7BCoxJLqDG0EzG
 ONMwAV8Z1DX68ONhydUBwzCRObwyvZPX4YbheklPziCvHyX5kRp5VyQ+FdSdV6pbqBDqFyOSC5VW
 esp8i8J17vcMIIO1u1iAYw==
-
```

Como indica el comando, el primer archivo (`prueba2_output`), a partir de la línea 17964911 y abarcando 4 líneas, hay contenido que difiere del segundo archivo (`mail/alejo/cur/archivo2_base64`), que comienza en la línea 17964911 y abarca 3 líneas. Se puede ver que hay una línea vacía presente en el primer archivo pero no en el segundo archivo.

Esto, según lo comentado en clase y lo investigado posteriormente, se debe a como se generan los archivos en Linux utilizando la redirección de los comandos.

Es así como se concluye que el servidor funciona correctamente y no altera los contenidos de los mails

Correcto funcionamiento al recibir una gran cantidad de conexiones simultaneas

Se utilizó el siguiente archivo Bash para abrir 502 conexiones al servidor, para verificar que el servidor soporta esa cantidad de conexiones concurrentes.

```
#!/bin/bash

for ((i=1; i<=502; i++)); do
  printf "CAPA\n" $i | ncat -C localhost 61655
done
```

Se verificó que el servidor no aborta y todo funciona correctamente.

Limitaciones de la aplicación

La aplicación cuenta con algunas limitaciones que se presentarán a continuación.

La primera está relacionada con el hecho de que hay un máximo de diez usuarios en el servidor, es decir, solamente pueden existir diez subdirectorios del *mail directory* que almacenen correos de usuarios. Esto desemboca en que, si bien el servidor admite más de 500 conexiones concurrentes, sólo 10 de esas conexiones puedan avanzar al estado `TRANSACTION` de POP3.

Otra limitación que podemos encontrar, que se vincula con el usuario, es que existe un tamaño máximo para el nombre del mismo, 255 caracteres.

También podemos encontrar la limitación de que un usuario que esta *logueado* en el servidor no puede ser eliminado.

Por último, se tiene la limitación de que todos aquellos cambios que un usuario realiza tendrán impacto en la siguiente conexión y no en la actual.

Posibles extensiones

Las posibles extensiones que se consideraron para futuras versiones de `Turtle Protocol v1` y la implementación del servidor son:

- Métricas individuales para cada usuario (ejemplo: cuantos bytes tiene el directorio de un usuario).
- Aumentar la cantidad máxima de usuarios en el servidor.

- Utilizar un hash-map para el guardado de los usuarios en el servidor. Esto mejoraría la eficiencia al realizar las operaciones de búsqueda de un usuario en particular.
- Modificar y ver el tamaño del buffer de lectura/escritura sobre el servidor.

Conclusiones

Este trabajo práctico resulto ser un desafío extraordinario que puso a prueba el entendimiento de los conocimientos que se fueron obteniendo durante la cursada y la consolidación de los mismos. Al tener que realizar una implementación de un protocolo existente y el diseño y desarrollo de un protocolo propio, se pudieron abordar las problemáticas y decisiones discutidas en clase en mayor profundidad. Este enfoque puramente práctico no se tuvo hasta la realización de este trabajo práctico especial.

El proceso de desarrollo de las aplicaciones nos llevaron a ir más allá de la lectura del RFC e implementación del servidor, sino que nos permitió comprender en detalle todos los fundamentos y elementos que se deben considerar a la hora de diseñar un protocolo (escalabilidad, viabilidad, grado de utilidad de los comandos, entre otros).

Para terminar, este trabajo práctico especial presento una serie de obstáculos importantes, sin embargo, el equipo está satisfecho con el resultado obtenido y el mayor entendimiento de los contenidos de la materia.

Guía de instalación

Compilación y creación de los ejecutables

Para la creación de los ejecutables necesarios del proyecto basta con posicionarse en la carpeta raíz del mismo y ejecutar el comando `make all`. De esta manera se crearán dos ejecutables, `turtle-client` y `turtle-pop3` en la carpeta `./bin`.

```
$$ make all
```

Utilización del directorio de mails

Para utilizar el servidor POP3, se requiere de un directorio que contenga todos los correos de los usuarios del sistema. Este directorio debe contener como hijos otros directorios con los nombres de usuario. Dentro de cada uno de esos directorios debe existir una carpeta llamada `cur`, que contendrá los correos, cada uno en un archivo separado.

El esquema siguiente muestra cómo debe quedar todo, suponiendo que el directorio del mail se llama `mail` y el servidor tiene tres usuarios (arnold, rocky, ronnie):

```
.
└── mail
    ├── arnold
    │   └── cur
    │       ├── mail1
    │       ├── mail2
    │       └── mail3
    ├── rocky
    │   └── cur
    │       ├── mail1
    │       └── mail2
    └── rocky
        └── cur
            ├── mail1
            ├── mail2
            ├── mail3
            ├── mail4
            ├── mail5
            └── mail6
```

Ejecución de las aplicaciones servidor y cliente

Una vez creado los ejecutables y los directorios, procedemos a ejecutar el servidor. Nos posicionamos en la carpeta `bin` que se encuentra en la raíz del proyecto.

Iniciando por el servidor corremos lo siguiente para poder obtener información de los diferentes argumentos que les

podemos pasar:

```
$$ ./turtle-pop3 -h
```

En base a la información que se presenta allí corremos el servidor con los siguientes parámetros:

- en el puerto 61655.
- registrando un usuario de username `arnold` y contraseña `sch`.
- con el token `arnold` (debe ser un token alfanumérico de 6 bytes que solo acepta los caracteres ASCII que verifiquen la siguiente expresión regular: `a-zA-Z0-9`)
- el directorio donde se encuentra la carpeta de mails es `../../mail`. Este path debe ser relativo a la ubicación del ejecutable.

```
$$ ./turtle-pop3 -p 61655 -u arnold:sch -t arnold -d ../../mail
```

El servidor quedará esperando conexiones TCP en el puerto especificado. Quien se conecte podrá utilizar el servidor POP3.

La lista completa de parámetros que acepta el servidor es el siguiente:

- `--help` ó `-h`: Imprime un mensaje de ayuda.
- `--directory <maildir>` ó `-d <maildir>`: Especificar el path del directorio donde se encontrarán todos los usuarios con sus mails.
- `--pop3-server-port <pop3 server port>` ó `-p <pop3 server port>`: Puerto entrante para conexiones al servidor POP3.
- `--config-server-port <configuration server port>` ó `-P <configuration server port>`: Puerto entrante para conexiones de configuración.
- `--user <user>:<password>` ó `-u <user>:<password>`: Usuario y contraseña de usuario que puede usar el servidor POP3. Hasta 10.

- `--token <token>` ó `-t <token>`: Token de autenticación para el cliente.
- `--version` ó `-v` Imprime información sobre la versión.

Por otro lado, se puede ejecutar la aplicación cliente, que permite hacer modificaciones y obtener información del servidor POP3. Nos posicionamos en la carpeta `bin` dentro de la raíz del proyecto y corremos el siguiente comando:

```
$$ ./turtle-client -h
```

Con este último comando podremos observar todos los argumentos que le podemos proveer a la aplicación cliente. Una vez que vemos las opciones procedemos a correr un comando para poder obtener las estadísticas del servidor POP3.

```
$$ ./turtle-client -p 62622 -t arnold -s
```

La lista completa de parámetros que acepta el cliente es el siguiente:

- `--help` ó `-h`: Imprime un mensaje de ayuda.
- `--token <token>` ó `-t <token>`: Token de autenticación para el cliente.
- `--port <server port>` ó `-P <server port>`: Puerto para conexiones al servidor POP3 a administrar.
- `--directory <maildir>` ó `-d <maildir>`: Path del directorio donde se encontrarán todos los usuarios con sus mails.
- `--add-user <user>:<password>` ó `-u <user>:<password>`: Usuario y contraseña de usuario que puede usar el servidor POP3. Hasta 10.
- `--change-password <user>:<password>` ó `-c <user>:<password>`: Cambiar contraseña para el usuario especificado.
- `--remove-user <user>` ó `-r <user>`: Eliminar usuario del servidor POP3.

- `--list-users` ó `-l`: Listar los usuarios del servidor POP3.
- `--statistics` ó `-s`: Obtener las estadísticas del servidor POP3.
- `--max-mails <number>` ó `-m <number>`: Cambiar el máximo número de mails.
- `--version` ó `-v`: Imprime información sobre la versión.

⚠️ El token de autenticación en el servidor y el cliente **DEBE** ser el mismo si se quiere que el cliente pueda modificar o obtener información del servidor

ℹ️ El puerto por defecto del servidor POP3, que puede ser cambiado con el argumento `-p`, es 61655

ℹ️ El puerto por defecto para que el cliente y el servidor se comuniquen, que puede ser cambiado con el argumento `-P` en ambos ejecutables, es 62622

Por último, si quiere ver los logs, debe ir a la carpeta `bin` presente en la carpeta raíz del proyecto y ahí existirá un archivo denominado `turtle-pop3.log`.

Ejemplos de configuración y monitoreo

Comandos que no requieren token de autenticación

Los siguientes comandos no necesitan un token para correrse.

Mensaje de ayuda

Para poder acceder a todas las opciones del cliente debemos ingresar lo siguiente en la terminal:

```
$$ ./turtle-client -h
```

Versión del protocolo diseñado

Para poder ver la versión del Turtle Protocol debemos ingresar:

```
$$ ./turtle-client -v
```

Y como respuesta obtenemos lo siguiente:

```
Turtle Client - Turtle Protocol v1
ITBA - Protocolos de Comunicación 20231Q -- Grupo 4
```

Comandos que requieren token de autenticación

Los siguientes comandos necesitan un token para correrse. El mismo debe ser exactamente igual que el presente al correr el servidor. En todos los casos siguientes suponemos que corrimos el servidor de la siguiente manera.

```
$$ ./turtle-pop3 -p 61655 -u arnold:sch -t arnold -d ../../mail
```

Cambiar el directorio

```
$$ ./turtle-client -t arnold -d ../../../mail
```

Y como respuesta obtenemos lo siguiente:

```
Change Mail Directory: All good
No content
```

Agregar un usuario al servidor

```
$$ ./turtle-client -t arnold -u stevecroker:1rfc69
```

Y como respuesta obtenemos lo siguiente:

```
Add User: All good
No content
```

Cambiar la contraseña de un usuario

```
$$ ./turtle-client -t arnold -c stevecroker:420rfc
```

Y como respuesta obtenemos lo siguiente:

```
Change Password: All good
No content
```

Eliminar un usuario

```
$$ ./turtle-client -t arnold -r ronnie
```

Y como respuesta obtenemos lo siguiente:

```
Remove User: All good
No content
```

Listar los usuarios

```
$$ ./turtle-client -t arnold -l
```

Y como respuesta obtenemos lo siguiente:

```
List Users: All good
arnold ronnie stevecroker
```

Estadísticas

```
$$ ./turtle-client -t arnold -s
```

Y como respuesta obtenemos lo siguiente:

```
Get Server Statistics: All good
HC:0 CC:3 TB:0
```

- HC (Historical Connections): conexiones históricas desde que se puso a correr el servidor.
- CC (Concurrent Connections): conexiones concurrentes.
- TB (Transferred Byes): bytes transferidos.

Cambiar el máximo número de mails

```
$$ ./turtle-client -t arnold -m 4
```

Y como respuesta obtenemos lo siguiente:

```
Change Maximum Mails: All good
No content
```