Trabajo Práctico Especial

Protocolos de Comunicación - 2C 2023

Grupo N7

Integrantes del grupo:

61413	Alasia, Gastón
61016	Della Torre, Matías
61156	Escudeiro, Patricio

1. Índice

Descripción detallada de los protocolos y aplicaciones desarrolladas

Servidor POP3

En primer lugar se desarrolló un servidor POP3 que cumple con los requerimientos especificados en el RFC 1939 así como también incluye algunas extensiones del mismo, especificadas en el RFC 2449. Se trata de un servidor no bloqueante, que soporta hasta 500 conexiones de clientes simultáneas tanto mediante IPv4 como IPv6.

En cuanto a las características del RFC, soporta todos los comandos requeridos como CAPA, LIST, STAT, RETR, RSET Y DELE. Con respecto a las extensiones, soporta autenticación con USER y PASS. También soporta pipelining y respuestas multilínea.

El servidor se ejecuta especificando el puerto donde escuchará las conexiones y a partir de ese momento pueden conectarse tanto clientes usuarios como el cliente de monitoreo. Este último es el que permite registrar usuarios y contraseñas válidas para que luego sean autenticados los clientes usuarios.

En caso de que un usuario o el servidor de monitoreo se conecten, el socket pasivo crea uno activo y se establece la conexión TCP. A partir de allí el servidor soporta los comandos enviados por el cliente. Al momento de la desconexión ya sea por un comando QUIT como por un CTRL+C , se liberan los recursos para dicho cliente.

Por defecto, buscará la carpeta de inbox en el directorio current/username/cur, por lo que esta debe estar previamente creada. Los mails dicha carpeta deben tener el formato Internet Message (RFC 5322) y de no tenerlo, el comportamiento del server es inesperado.
Para saber qué esta sucediendo en el servidor, este loguea por salida estándar información sobre cada conexión y desconexión de cliente, y cada log in y log out de usuario.

Protocolo de monitoreo

El protocolo creado es binario, TCP y no orientado a conexión. Se detalla en su RFC correspondiente, ubicado en la carpeta *documentation* del proyecto.

Cliente de monitoreo/configuración

El cliente de monitoreo/configuración se conecta y se comunica con el servidor POP3 mediante el protocolo de monitoreo descripto previamente. Al ser no

orientado a conexión, se ejecuta el binario con ciertos parámetros que realizan una acción y luego el proceso finaliza. Para realizar otra acción, se debe volver a ejecutar el binario *client*.

El cliente tiene dos funcionalidades principales: obtener métricas históricas con respecto al uso del servidor y configurar ciertos parámetros del servidor.

Las métricas que ofrece del servidor son:

- Cantidad de conexiones históricas
- Cantidad de conexiones concurrentes
- Cantidad de bytes transferidos

Las configuraciones que se le pueden hacer al servidor:

- Registrar y eliminar usuarios clientes
- Registrar y eliminar administradores
- Cambiar el Maildir

Para ver todas las opciones que presenta el cliente, se debe correr:
'./client -h '

Problemas encontrados durante el diseño y la implementación

En primer lugar, como decisión de diseño se planteó que el servidor fuera de tipo no bloqueante. Esto significó hacer uso de *pselect* con el file descriptor de cada socket. Por simplicidad, se intentó hacer uso del selector provisto por la cátedra en los “patches”. El problema fue comprender en su totalidad estos archivos para poder hacer un uso eficiente del selector. Finalmente fue comprendido y se pudo comenzar con la creación de estados.

Con respecto a los estados, hubo dudas sobre cuáles era necesario implementar como un único estado y cuáles por separado. Por ejemplo en el estado de Authentication del servidor POP3, primero se había considerado tener dos estados: uno antes de que el usuario ingrese el comando USER y uno luego. Lo mismo en Transaction: un estado para leer y otro para escribir. Esto finalmente fue cambiado por un único estado para Authentication y otro para Transaction, ya que comparían los mismos recursos y buffers y respetarían la estructura de *statbl* y sus handlers. La única excepción se dio en el caso de los estados *MAIL_READ* y *MAIL_WRITE*, ya que al estar suscribiendo al selector un fd archivo en lugar de un fd de socket resultó más práctico tener dichos estados separados. Sin embargo, se podrían unificar en uno solo pero esto excedía los tiempos de entrega del proyecto.

Con respecto al parser de comandos, en un primer momento se pensó en una implementación desde cero sin tener el cuenta el provisto por la cátedra. Luego de comprender las dificultades de hacerlo desde cero, se pudo utilizar el de la cátedra aunque no de la manera en que estaba pensado para usarse (por ejemplo usando mail los *parser_event*). Sin embargo, cumple con su función. También se plantearon varios alternativas sobre si usar una instancia del parser diferente por cada estado que parsea comandos, pero luego se decidió reutilizar el mismo ya que cumplirían exactamente la misma función y tendrían el mismo autómata subyacente.

En cuanto al diseño de las estructuras y dónde almacenar la información, se terminó decidiendo por agregar toda la información de un cliente en concreto en la estructura *pop3*. Fue motivo de debate por la cantidad de información que se debe almacenar allí, como una lista de información sobre los correos, las credenciales (que luego de Authentication ya no son utilizadas). Para solucionar esto se optó por la modularización y el uso de estructuras tanto como fue posible.

Para la carga de mails, se presentaron dos inconvenientes. En primer lugar hubo que decidir en qué momento cargarlos en memoria y de qué manera. Se había considerado la posibilidad abrir todos los correos del usuario y guardar sus fd en un array en la estructura pop3, pero luego se desistió ya que no tenía sentido abrir mails si el usuario luego no haría un RETR de dichos mails. La idea original era abrirlos al inicio del estado de Transaction y nada más. La solución tomada fue solo guardar en un array una estructura que contiene información de los mails del usuario (filename y tamaño), para luego hacer un open a demanda. Otro array, en cambio, guarda los flags de cada mail que indican si fue marcado como borrado o no. En cuanto a cuándo cargar los mails, se decidió cargarlos antes de la ejecución de cada comando de Transaction. De este modo, se recalculan siempre la cantidad de mails y la cantidad de octetos total de la casilla del usuario reduciendo errores que surgían si no se realizada de ste modo. Los flags, en cambio, son persistentes ante múltiples comandos de Transaction y solo se reinician con el comando RSET.

La lectura de mails, por otra parte, presentaba la dificultad de combinarla con el selector que ya se usaba para los fd de los sockets. Finalmente se decidió registrar los fd de archivo junto con los fd de socket en el selector, y se tuvo que cambiar alguna funcionalidad del selector con handlers específicos si se trata de fd de mail en lugar de socket.

Limitaciones de la aplicación

El proyecto presenta algunas limitaciones, entre ellas:

- Imposibilidad de modificar el tamaño del buffer a demanda: requeriría pequeños cambios en el protocolo de monitoreo y en el server ya que fueron pensados para poder escalarse. Pero no se pudo realizar por falta de tiempo.
- Necesidad de tener una carpeta con el nombre del usuario creada para que el servidor pueda leer mails desde la misma. Esto es inherente al intentar imitar el acceso a carpetas de mail de usuario de Linux (que tiene permisos para acceder a carpetas de todos los usuarios del sistema).
- Ausencia de un semáforo en la carpeta de un usuario para evitar que dos clientes que se logueen con el mismo user y password no puedan modificar simultáneamente el mismo maildir.
- La longitud del path a la carpeta Maildir debe tener alrededor de 40 caracteres.
- El nombre de usuario puede tener como máximo 4 caracteres y la contraseña 16.
- El protocolo de monitoreo debe correr obligatoriamente en el puerto 1081.

Posibles extensiones

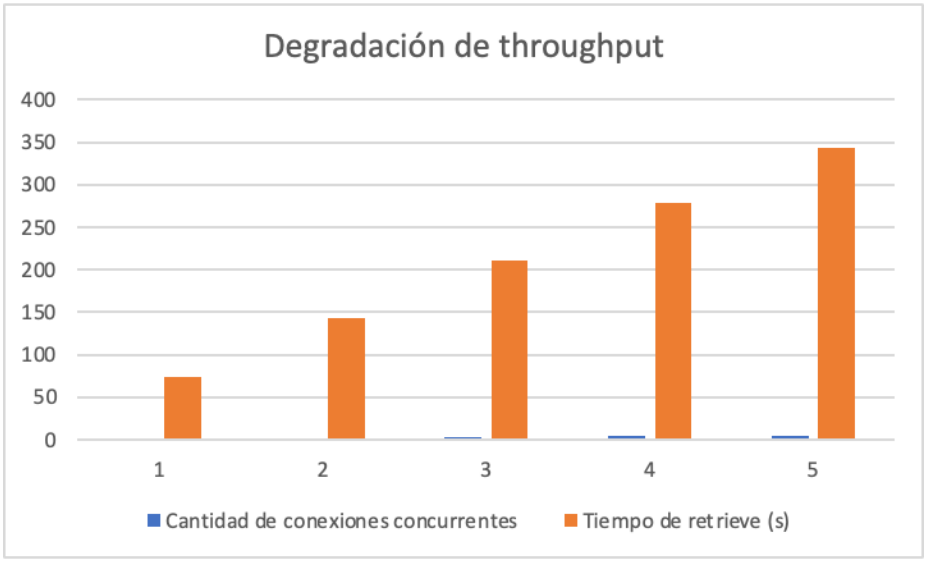
- Implementar un mecanismo de transformación de mails externo.
- Poder agregar tokens de administrador al cliente de monitoreo.
- Poder tener un logging más detallado y guardado en un archivo para que sea persistente.
- Poder setear configuraciones del server de manera persistente.
- Extender los límites de tamaño para los nombres de usuario, contraseñas y paths.

Prueba de stress

Las pruebas se realizaron de la siguiente manera: se crearon 5 usuarios en el servidor mediante el cliente a los cuales se les agregó en su Maildir el mismo archivo de 1,33GB. Se realizó luego un retrieve de dicho mail, primero desde una terminal, luego desde dos simultáneamente, luego desde tres simultáneamente, y así sucesivamente.

Los resultados obtenidos fueron los siguientes:

Cantidad de conexiones concurrentes	Tiempo de retrieve (s)
1	74
2	143
3	211
4	279



Conclusiones

En conclusión el servidor cumple con todos los requisitos funcionales y no funcionales pedidos en el enunciado. Permite conexiones IPv4 e IPv6, soporta pipelining y respuestas multilínea. Es escalable con capacidad para agregar extensiones a futuro.

La creación de dicho servidor es una demostración de la aplicación y consolidación de los conocimientos adquiridos durante el desarrollo de la materia sobre protocolos de comunicación.

Ejemplos de prueba

A continuación se adjuntan algunas imágenes que muestran el correcto funcionamiento del servidor mediante el uso de diferentes comandos que soporta.

Log in:

```
matidellatorre@matipc:~/Documents/tps/TP-Protos-1C2023-G7$ nc -C -v localhost 9999
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:9999.
+OK Server ready
USER foo
+OK
PASS bar
+OK Logged in.
```

CAPA:

```
CAPA
+OK
CAPA
USER
PIPELINING
STAT
LIST
RETR
DELE
RSET
NOOP
.
```

LIST (sin argumentos)

```
LIST
+OK 23 messages (1450504453 octets)
1 26
2 26
3 26
4 26
5 26
6 26
7 26
8 26
9 26
10 1450493349
11 26
12 26
13 26
14 26
15 5
16 26
17 26
18 26
19 26
20 26
21 10579
22 26
23 26
.
```

LIST (mail en específico)

```
LIST 21
+OK 21 10579
```

STAT (sin argumentos)

```
STAT
+OK 23 1450504453
```

STAT (mail en específico)

```
STAT 10
+OK 23 1450504453
```

DELE

```
DELE 23
+OK message deleted
```

RETR

```
RETR 21
+OK
Delivered-To: mdella@itba.edu.ar
Received: by 2002:a05:6a20:8c30:b0:11a:4138:daec with SMTP id j48csp2618401pzh;
        Sun, 18 Jun 2023 13:49:01 -0700 (PDT)
X-Received: by 2002:a05:600c:3653:b0:3f7:5e07:ea54 with SMTP id y19-20020a05600c36530
b003f75e07ea54mr7031022wmq.13.1687121341661;
        Sun, 18 Jun 2023 13:49:01 -0700 (PDT)
ARC-Seal: i=1; a=rsa-sha256; t=1687121341; cv=none;
        d=google.com; s=arc-20160816;
        b=JjNpB/PoUhAm1MBB1BiQQ2kY+1y+BypftQ5HW/o428DGgNpywbZfal4r2HhPKGJuSo
         grO6R1j5ECTIUhxeG+lkRRjQLE0HqAH9iXTE/jFSR3ukeAZjWymI8CX5u98rYKFAc4w4
         QQgf2sHMHX2Wr5Utb3HEO0QvpVJjXymJZNcflwtTXXhprYU6UK1i0Pye39dfHl9F+zk0
         bErQicVSKyejJd7zSwSIghYA3vlcRKaBqJkMBYJ1zo2hpDJXfw7sVy7RyuC0zkm98Ca1
         i5pynWOt/K/q3NMlABscJf7OR8/njaojjpQs1aaTvROwJyiVqMc8JPRw0JpEPuuTxPYC
         8qIw==
ARC-Message-Signature: i=1; a=rsa-sha256; c=relaxed/relaxed; d=google.com; s=arc-2016
816;
        h=cc:to:subject:message-id:date:from:in-reply-to:references
         :mime-version:dkim-signature;
        bh=r5hz7dGnHv6Ya5kJxrA+IPvdiD/a5IMCY6DfI+5kGSI=;
        b=CvNelISGVHwUS67vHNrnAdx/DNcdiYQNhrItxGrk0MvIaPnlzcK8YSs54ADbDc0xxS
         fIvd342QeKZUqzGKAmETNUCk7NUOwp2Z0yF6+fk+dswGsIlWW5gH8J2CCGnjGQopywHT
         wwm1+Rh6PO6sOvj1CuznU6m9pCcl9K4oOoqGkLhEdzEXxATb0/wFsezIrQ8ys3CZkTOT
         bnTZxApr1qJCM5y4eh9wMU7TmLcjkN7jJtObhP9zmbYvAaFtpKm6TlHj8WW1R0uGPTUI
         cQB4ccomW/Yejj1stru7e6nuW0WGpHecevMxQM1V3BeA1lTJe0RMLv6uH49cs8AbW0fr
         39dA==
ARC-Authentication-Results: i=1; mx.google.com;
       dkim=pass header.i=@gmail.com header.s=20221208 header.b=Y0ZrTbnc;
       spf=pass (google.com: domain of agustin.golmar@gmail.com designates 209.85.220
41 as permitted sender) smtp.mailfrom=agustin.golmar@gmail.com;
       dmarc=pass (p=NONE sp=QUARANTINE dis=NONE) header.from=gmail.com
Return-Path: <agustin.golmar@gmail.com>
Received: from mail-sor-f41.google.com (mail-sor-f41.google.com. [209.85.220.41])
        by mx.google.com with SMTPS id l14-20020a7bc44e000000b003f8d8195d6csor1330192
```

QUIT

```
QUIT
+OK
```

Lectura de mails mediante curl

```
matidellatorre@matipc:~/Documents/tps/TP-Protos-1C2023-G7$ curl -v pop3://foo:bar@loca
lhost:9999/5
*   Trying 127.0.0.1:9999...
* Connected to localhost (127.0.0.1) port 9999 (#0)
< +OK Server ready
> CAPA
< +OK
< CAPA
< USER
< PIPELINING
< .
> USER foo
< +OK
> PASS bar
< +OK Logged in.
> RETR 5
< +OK
Hola este es un mail

.
```

Guía de instalación

Para compilar el proyecto, simplemente se debe hacer un 'make all' en la carpeta raíz del mismo. Se generarán dos binarios: *pop3* y *client*. Estos corresponden al servidor POP3 y al cliente de configuración/monitoreo, respectivamente.

Instrucciones para la configuración

El primer paso consiste en crear el directorio Maildir donde se tendrán los usuarios para los cuales vamos a querer autenticarnos y leer mails, cada uno con su carpeta /cur, en donde se ubicarán los mails destinados a este usuario. Se detalla en un ejemplo el formato de este directorio a continuación:

```
./Maildir
        |_ /foo
        |       |_ /cur
        |               |_mail1
        |               |_mail2
        |_ /user2
                |_ /cur
                        |_mail1
                        |_mail2
```

Para utilizar el cliente de monitoreo y configurar el servidor pop3 se debe utilizar la siguiente sintaxis:

"Usage: ./client [OPTIONS]... TOKEN [DESTINATION] [PORT]"

- Como se mencionó en limitaciones el cliente solo puede correr en el puerto 1081*

Donde podemos elegir dentro de las siguientes opciones de comandos:

Comando	Argumento	Descripcion
-h	-	imprime los comandos del programa y termina.
-c	-	imprime la cantidad de conexiones concurrentes del server.
-C	-	imprime la cantidad de conexiones históricas del server.
-b	-	imprime la cantidad de bytes transferidos del server.
-u	user:pass	agrega un usuario al servidor pop3 con el nombre y contraseña indicados.
-U	user:token	agrega un usuario administrador con el nombre y token indicados.
-d	user	borra el usuario del servidor pop3 con el nombre indicado.
-D	user	borra el usuario administrador con el nombre indicado.
-m	path al directorio en cuestion (*1)	cambia el maildir del servidor al pedido.
-v	-	imprime la versión del programa y termina.

(*1) Considerar que path al nuevo directorio no puede ser muy extenso como se mencionó en limitaciones. Este path debe dirigirse a un directorio con el formato previamente especificado.

Ejemplos de configuración y monitoreo

Los siguientes ejemplos corresponden al uso de la aplicación cliente:

Help

```
matidellatorre@matipc:~/Documents/tps/TP-Protos-1C2023-G7$ ./client -h
Usage: ./client [OPTIONS]... TOKEN [DESTINATION] [PORT]
Options:
-h                  imprime los comandos del programa y termina.
-c                  imprime la cantidad de conexiones concurrentes del server.
-C                  imprime la cantidad de conexiones históricas del server.
-b                  imprime la cantidad de bytes transferidos del server.
-u <user:pass>      agrega un usuario al servidor pop3 con el nombre y contraseña indi
cados.
-U <user:token>     agrega un usuario administrador con el nombre y token indicados.
-d <user>           borra el usuario del servidor pop3 con el nombre indicado.
-D <user>           borra el usuario administrador con el nombre indicado.
-m                  cambia el maildir del servidor al pedido.
-v                  imprime la versión del programa y termina.
```

Agregar usuario

```
matidellatorre@matipc:~/Documents/tps/TP-Protos-1C2023-G7$ ./client -u mati:pass 12345
6789abcdef1
The new user: 'mati' is now added in the server
```

Obtener una métrica de usuario

```
matidellatorre@matipc:~/Documents/tps/TP-Protos-1C2023-G7$ ./client -C 123456789abcdef
1
The amount of historic connections is: 1
```

Documento de diseño del proyecto

Se encuentra en la carpeta *documentation* del proyecto.