Protocolos de Comunicacion
Grupo 7
Agustín Naso 60065
Gaston De Schant 60755
Natali Lilienthal 60544
Brittany Lin 60355

Trabajo Práctico Especial

Tabla de Contenidos

1. Protocolos y aplicaciones desarrolladas

Para el trabajo, decidimos diseñar e implementar dos aplicaciones y un protocolo.

1.1 Servidor proxy POP3

Para la implementación del servidor proxy del protocolo POP3, se hizo uso de los códigos fuentes provistos por la cátedra que luego fueron modificados y adaptados.

El servidor proxy POP3 maneja múltiples conexiones de manera no bloqueante. Entre los códigos provistos se encuentra una implementación de un selector que hace uso de la función pselect() para manejar los file descriptors. Para poder manejar dichos descriptors, se debe antes que nada registrarlos. Una vez registrados, sobre ellos se pueden asignar intereses tanto de lectura como de escritura o ningún interés si así se desea.

Para conocer la dirección del servidor POP3 origen, el proxy recibe dicha dirección como argumento. Dicha dirección puede ser una dirección IPv4, IPv6 o un FQDN. En el caso de un FQDN, se realiza una consulta DNS para resolver las direcciones del nombre realizada en un nuevo hilo para evitar bloquear el hilo principal. Luego, para cada conexión entrante, el proxy se intenta conectar al origen y crea o reutiliza alguna estructura POP3 que permite almacenar los recursos. La idea de reutilizar estructuras de almacenamiento de recursos permite eficientizar el uso de memoria. Para ello se implementa una pool que almacena estructuras recicladas para ser utilizadas en conexiones nuevas.

El proxy también recibe otros tipos de argumentos por línea de comando para configurar ciertas características del proxy. La función getopt() permite obtener dichos argumentos.

Por otro lado, cada conexión maneja su propia máquina de estados. Es decir, en un momento dado, cada conexión se encuentra en un estado igual o distinto de otras conexiones. Estos estados incluyen: RESOLVING, CONNECTING, HELLO, CHECK CAPABILITIES, COPY, SEND ERR MSG, DONE y ERROR. La máquina de estados permite manejar los intereses asociados al selector.

El estado RESOLVING resuelve la dirección FQDN a través de una consulta DNS en caso de ser necesario y finalmente se realiza la conexión. Una vez resuelto el RESOLVING pasa al estado CONNECTING, en el cual se asignan los intereses del cliente y del origen según el estado a seguir. En caso de que no hayan errores, el siguiente estado es el HELLO, en caso contrario puede pasar al estado SEND ERROR MSG o ERROR si el error no es rescatable. En el estado HELLO, se pone en manifiesto una de las características de un servidor POP3, un servidor "orientado a conexión" pues el origen envía un saludo inicial al cliente. Al completarse el saludo, pasa al estado CHECK CAPABILITIES que valida las capacidades del servidor origen. El cliente puede pedir las validaciones mediante el comando CAPA, en dichas validaciones siempre debe mostrar que se soporta pipelining, incluso si el servidor POP3 no lo implementa.

El estado COPY es un estado general que se encarga de la ida y vuelta de los bytes transferidos entre cliente y servidor POP3. En caso de asignar cierto filtro por línea de comando, el COPY también se encarga de transferir dichos bytes del origin al filter y luego del filter al cliente.

Finalmente, tenemos los estados de finalización: ERROR y DONE. El SEND ERROR MSG es un estado previo al ERROR para indicar al usuario que hubo cierto problema en caso de ser alguna falla inesperada.

En cuanto a los recursos, se utilizan dos estructuras de buffers. La implementación de dicha estructura fue provista por la cátedra pero modificada de manera tal que implemente un puntero de parseo. Este puntero se utiliza especialmente en los parsers para poder analizar el buffer sin consumir la información de manera innecesaria. Los dos buffers utilizados corresponden a un buffer de escritura y un buffer de lectura compartidos por el cliente, el origen y el filtro.

Los buffers permiten almacenar las comandos del cliente y las respectivas respuestas del servidor POP3. El manejo de comandos es muy particular debido a que se debe soportar pipelining incluso cuando el servidor no lo implementa. Para ello, se utiliza una cola que almacena los comandos y a medida que se desencolan dichos comandos se envían al origen para recibir su respuesta.

1.2 Protocolo implementado

El protocolo implementado para configurar el proxy se comunica a través de UDP. Al no ser orientado a conexión, cada conexión del

usuario al administrador se hace mediante un token configurado en el proxy. Luego, el administrador hace un chequeo para verificar que el token sea el correcto y procede a ejecutar la función del comando correspondiente.

Formato de mensaje a mandar por el usuario:

```
token comando [argumentos]
```

1.2.1 Comandos

stats:

Argumentos: ninguno

Comentario: permite obtener las métricas del proxy para monitorear la operación del sistema.

Ejemplo:

```
C: token stats
S: Active Connections: 5
   Total Connections: 10
   Total Bytes Transfered: 3200
```

help:

Argumentos: ninguno

Comentario: permite obtener ayuda e imprime los comandos disponibles

Ejemplo:

```
Help (try one of this commands):
    stats: print proxy's metrics
    getbuffsize: print current buffer size
    setbuffsize: set new buffer size
    gettimeout: print current timeout
    settimeout: set new timeout
    geterrorfile: print current error file
    seterrorfile: set new error file
```

getbuffsize:

Argumentos: ninguno

Comentario: permite obtener el tamaño del buffer del proxy

Ejemplo:

```
C: getbuffsize
S: Buffer size value 4000
```

setbuffsize:

Argumentos: tamaño del buffer

Comentario: permite asignar el tamaño del buffer

Ejemplo:
C: setbufsize 3200
S: New buffer size set to 3200

gettimeout:

Argumentos: ninguno

Comentario: permite obtener el timeout

Ejemplo:
C: gettimeout
S: Current timeout: 10

settimeout:

Argumentos: timeout

Comentario: permite configurar el timeout

Ejemplo:
C: settimeout 20 10
S: Timeout value:
Seconds: 20
Nano Seconds: 10

geterrorfile:

Argumentos: ninguno

Comentario: permite obtener el archivo de error

Ejemplo:
C: geterrorfile
S: Error file: /dev/null

seterrorfile:

Argumentos: archivo de error

Comentario: permite asignar el archivo de error

Ejemplo:
C: seterrorfile /dev/null
S: New error file is: /dev/null

2. Problemas durante el diseño e implementación

El primer problema encontrado surgió al intentar escuchar tanto para IPv4 como para IPv6. Luego de varios intentos de modularización, se optó por una estructura que permita almacenar tanto direcciones IPv4 como direcciones IPv6.

Por otro lado, se encontraron muchas dificultades durante el manejo del pipelining. La idea principal consistió en guardar los comandos

del pipelining en una cola e ir mandando de a uno, incluso si el servidor origen maneja pipelining. Esto permitió facilitar el parseo de la respuesta y el filtro de los mensajes. Sin embargo, esta idea también resulto difícil de implementar pues originaron muchos conflictos con los buffers ya que estos son compartidos por los tres actores: cliente, servidor y filtro.

Otra gran problema surgió en el filtrado de la respuesta. El primer paso consistió en identificar la respuesta a ser filtrada y luego des escapar los puntos agregados por el servidor POP3 para luego poder ser filtrados. Finalmente, al ser filtrados, volver a escapar los puntos. En síntesis, las dificultados surgieron al momento de parsear para saber cuando la respuesta era de interés y luego parsear para saber donde des escapar los puntos.

3. Limitaciones de la aplicación

Una posible limitación de la aplicación es la cantidad de file descriptors que se permiten registrar en el selector. Dicho selector admite 1024 file descriptors a la vez.

4. Posibles extensiones

4.1 Encriptación del token del administrador

Actualmente, el token para acceder al administrados se encuentra en la configuración del proxy. Sin embargo, seria ideal que dicho token pueda ser encriptado de manera tal que no sea accedido tan fácilmente.

4.2 Agregar implementaciones en el administrador

Una posible extensión podría ser agregar mas funcionalidades al administrador puesto que las funcionalidades ofrecidas pueden resultar limitantes.

4.3 Conexión del administrador

La conexión actual con el administrador es mediante UDP. Si bien decimos que es una conexión, no lo es realmente pues UDP tiene la característica de no ser orientado a conexión. Como vimos durante la cursada, tiene muchas limitaciones. Esto se podría mejorar implementando una conexión TCP.

5. Conclusiones

La implementación y el diseño de este TPE fue duro y difícil. Asimismo, el trabajo se tuvo que dividir entre los integrantes para acelerar el proceso. Sin embargo, al encontrarse con tantas dificultades, estas tareas se vieron mezcladas para poder solucionar entre todos los problemas.

Si bien fue un trabajo arduo, pusimos a prueba todos los conocimientos adquiridos durante la cursada. Además, el trabajo

ayudo a reforzar conceptos adquiridos en las materias previas a Protocolos de Comunicación.

El resultado final nos satisface puesto que tuvimos que afrontar muchos obstáculos. Dichos obstáculos nos frenaron muchas veces, impidiendo el avance en el trabajo pero al final logramos sobrepasar muchos de ellos.

6. Ejemplos de prueba

```
agusnaso@agustin-PC:/mnt/c/Users/agus_/Desktop/Protos/TP2/ProxyPOP3$ ./main 192.168.0.156
listening on tcp port: 1110
```

```
agusnaso@agustin-PC:/mnt/c/Users/agus_$ nc -C localhost 1110
+OK Dovecot (Ubuntu) ready.
USER brittu
+OK
PASS brlin
+OK Logged in.
LIST
+OK 3 messages:
1 483
2 492
3 507
.
RETR 3
+OK 507 octets
Return-Path: <agustin@naso.mail.com>
X-Original-To: brittu@naso.mail.com
Delivered-To: brittu@naso.mail.com
Received: from localhost (localhost [127.0.0.1])
        by anaso (Postfix) with SMTP id 88A2BE0D4D
        for <brittu@naso.mail.com>; Mon, 22 Nov 2021 22:11:27 -0300 (-03)
From: "AGUSTIN" <agustin@naso.mail.com>
To: "Shampein BRITU" <brittu@naso.mail.com>
Subject: VAMOS NATU YO CONFIO 2
Message-Id: <20211123011135.88A2BE0D4D@anaso>
Date: Mon, 22 Nov 2021 22:11:27 -0300 (-03)

britu\n.
britulin
.
```

```
agusnaso@agustin-PC:/mnt/c/Users/agus_$ printf 'USER brittu\nPASS brlin\nLIST\nRETR 1\n' | nc -C localhost 1110
+OK Dovecot (Ubuntu) ready.
+OK
+OK Logged in.
+OK 3 messages:
1 483
2 492
3 507
.
+OK 483 octets
Return-Path: <agustin@naso.mail.com>
X-Original-To: brittu@naso.mail.com
Delivered-To: brittu@naso.mail.com
Received: from localhost (localhost [127.0.0.1])
        by anaso (Postfix) with SMTP id 0A159E0F1B
        for <brittu@naso.mail.com>; Sun, 21 Nov 2021 17:56:10 -0300 (-03)
From: "Nasu" <agustin@naso.mail.com>
To: "Brittu" <brittu@naso.mail.com>
Subject: Brituuuuuuuuuu
Message-Id: <20211121205622.0A159E0F1B@anaso>
Date: Sun, 21 Nov 2021 17:56:10 -0300 (-03)

Hola britu

.
```

7. Guía de instalación

Para correr el proyecto tenemos dos ejecutables. Para compilar y linkeditar, se debe hacer un 'make all' dentro de la carpeta principal 'ProxyPOP3'.

Para la ejecución del proxy, debemos pararnos en la carpeta 'pop3filter' y correr './main -t cat 127.0.0.1'.

8. Instrucciones para la configuración

Para correr el cliente se debe correr el archivo ejecutable './client'.

9. Ejemplos de configuración y monitoreo