Trabajo Práctico Especial: ProxyPOP3

Protocolos de Comunicación

Grupo 1

Clara Guzzeti

José Martin Torreguitar

Ignacio Vidaurreta

Juan Bensadon

Índice

Protocolos y Aplicaciones desarrollados

POP3 Proxy Server

El proxy server de POP3 consiste en un nexo entre el servidor origen y el cliente. El mismo provee de funcionalidades adicionales para que el cliente acceda a sus correos electrónicos incluyendo soporte de pipelining indistinta a la capacidad del servidor origen y transformación de mensajes.

La concurrencia en el proxy se maneja a través del uso de sockets no bloqueantes en modo multiplexado. Se explota la potencia de los selectores, una abstracción del uso del select para manejar las condiciones de subscripción de lectura y escritura en los buffers.

La resolución de nombres permite uso de hilos para independizar a los handlers de los problemas de concurrencia

El manejo de pipelining para servidores que soportan o no dicha capacidad se gestionó con el uso de una cola de requests dentro de la estructura de pop3 que es manipulada por la máquina de estados del proxy.

POP3YE Server Management Protocol

Además del servidor POP3 en sí, se desarrolló un servidor que corre en otro puerto, el cual servirá para que el usuario tenga la posibilidad de iniciar sesión y luego obtener métricas sobre el funcionamiento del servidor proxy, o bien consultar y determinar cuál será el binario de transformación utilizado por el proxy. Para la comunicación con dicho servidor, se desarrolló un protocolo llamado POP3YE que detallaremos en la siguiente sección.
Para la comunicación entre el cliente y el proxy se trató de imitar un poco la forma del protocolo visto en clase **socksv5**. La comunicación entre ambos es de un conjunto de bytes

Estructura del request del cliente al proxy:

Nombre	CMD	NARGS	ARGLEN 1	ARG 1	○ ○ ○	ARGLEN N	ARG N
# Bytes	1	1	1	VAR		1	VAR

Donde el primer byte es el comando, que al día de la entrega puede ser:

- **0x00**: Pedir nombre de usuario
- **0x01**: Pedir contraseña
- **0x02**: Pedir cuántas conexiones concurrentes hay
- **0x03**: Setear la transformación
- **0x04**: Pedir cuál es la transformación actual que se ejecuta
- **0x05**: Cantidad de bytes transferidos por el proxy.
- **0x06**: Salir

Como la longitud de los argumentos es variable, es importante antes del argumento especificar la longitud de este por cuestiones de parseo.

Para la respuesta del proxy al cliente, este responde con un código de retorno para ver si el pedido fue realizado con éxito (1 byte) la longitud de un cuerpo de respuesta (en caso de que este exista) (1 byte) y el cuerpo de respuesta en sí (variable).

Proxy Server Management Client

El protocolo creado para la administración del servidor proxy es un protocolo SCTP que utiliza procesamiento binario para enviar e interpretar mensajes. Para facilitar al usuario la comunicación con el servidor de administración, se implementó un cliente que interactúa con él.

El cliente presenta al usuario con un menú que le indicará los comandos que puede ejecutar. Permite al usuario iniciar sesión y, una vez hecho esto, consultar las métricas del servidor, consultar el filtro actual y determinar un nuevo filtro a usar, todo de manera interactiva.

Internamente, crea la conexión SCTP con el servidor de administración, emite el menú que se presenta al usuario por salida estándar, y genera, acorde a los comandos ingresados por el usuario, los mensajes correspondientes en el formato del protocolo POP3.

Filtro StripMime

El filtro StripMime se trata de una aplicación que recibirá por entrada estándar un e-mail con estructura consistente con la convención MIME (RFC 1341) y, basándose en los valores de las variables de entorno FILTER_MEDIAS y FILTER_MSG, realizará un procesamiento del e-mail, para emitir el resultado por salida estándar.

Dicho procesamiento tiene como objetivo filtrar ciertas secciones del e-mail. Las secciones a filtrar estarán determinadas por el valor de su header Content-type, que se comparará contra el valor de FILTER_MEDIAS, y cuyo contenido se reemplazará por el mensaje contenido en la variable FILTER_MSG.

Problemas Encontrados

El desarrollo de todos los componentes del trabajo fue arduo. Si bien habíamos tenido experiencia con el desarrollo de bajo nivel, jamás la habíamos tenido empleando herramientas complejas como el selector o los parsers empleados por stripmime.

Asimismo, encontramos varios conflictos a la hora de integrar las distintas partes del proyecto. Las transformaciones representaron un problema al fallar exclusivamente al utilizar la aplicación de Stripmime, pero no al emplear otras aplicaciones como *cat*. Se resolvió mediante el cierre y el desregistro de los File Descriptors correspondientes.

Otro de los desafíos encontrados tiene que ver con el manejo de el CRLF como nueva línea. Tuvimos un error de diseño al comienzo del proceso de desarrollo, donde diseñamos la solución solamente considerando al '\n' como indicador de nueva línea. Esto condujo una mala estimación de tiempos y a tener que repensar el parseo de los comandos junto con el pipelining.

El procesamiento de e-mails cuya longitud superara el tamaño de los buffers resultó en un cambio en el diseño en la máquina de estados. Originalmente, se leía primero la respuesta del servidor, luego se filtraba, y luego se escribía el resultado en el cliente, y de ser el e-mail más largo que el tamaño de buffer, se repetían estas tres operaciones tantas veces como fuera necesario. Eventualmente, se resolvió primero leer el e-mail y enviar sus partes al filtro, y luego obtener las partes del mail del filtro y escribirlas al cliente. De esta manera, el proceso externo solo se inicializa una vez.

Debido a cambios constantes en el código por observar que ciertas técnicas utilizadas no estaban siendo útiles o no estaban funcionando muy bien, nuestro TP cuenta con antipatrones como *spaghetti code* y código legacy ya no usado. Se necesita un refactor para hacerlo más comprensible.

Limitaciones de la Aplicación

Si bien hemos testeado el servidor y permite hasta aproximadamente 250 conexiones concurrentes, pasado ese número de usuarios comienza a tener problemas e incluso llega a cortarse su ejecución.

El filtro empleado sólo permite filtrar secciones basadas en su Content-Type, y no hace ajustes del mensaje de filtrado basados en el encoding de las secciones. Se podría agregar la detección de encoding al filtro, e incluso desarrollar un filtro más soficticado que utilice otras técnicas más avanzadas.

Posibles Extensiones

El uso de memoria y la performance del proyecto se podría optimizar mediante una revisión más detallada de todo el código. Asimismo, se podría mejorar el estilo para evitar código repetido, mejorar nombres de ciertas variables y funciones y demás. Esto facilitaría el mantenimiento y extensión de las aplicaciones desarrolladas.

En cuanto a las métricas, el servidor de administrador permite acceder a la información de cuántas conexiones concurrentes hay en el proxy, el número de conexiones totales, y el número de bytes transferidos. La cantidad de métricas obtenidas podría mejorarse para brindar más información útil al usuario.

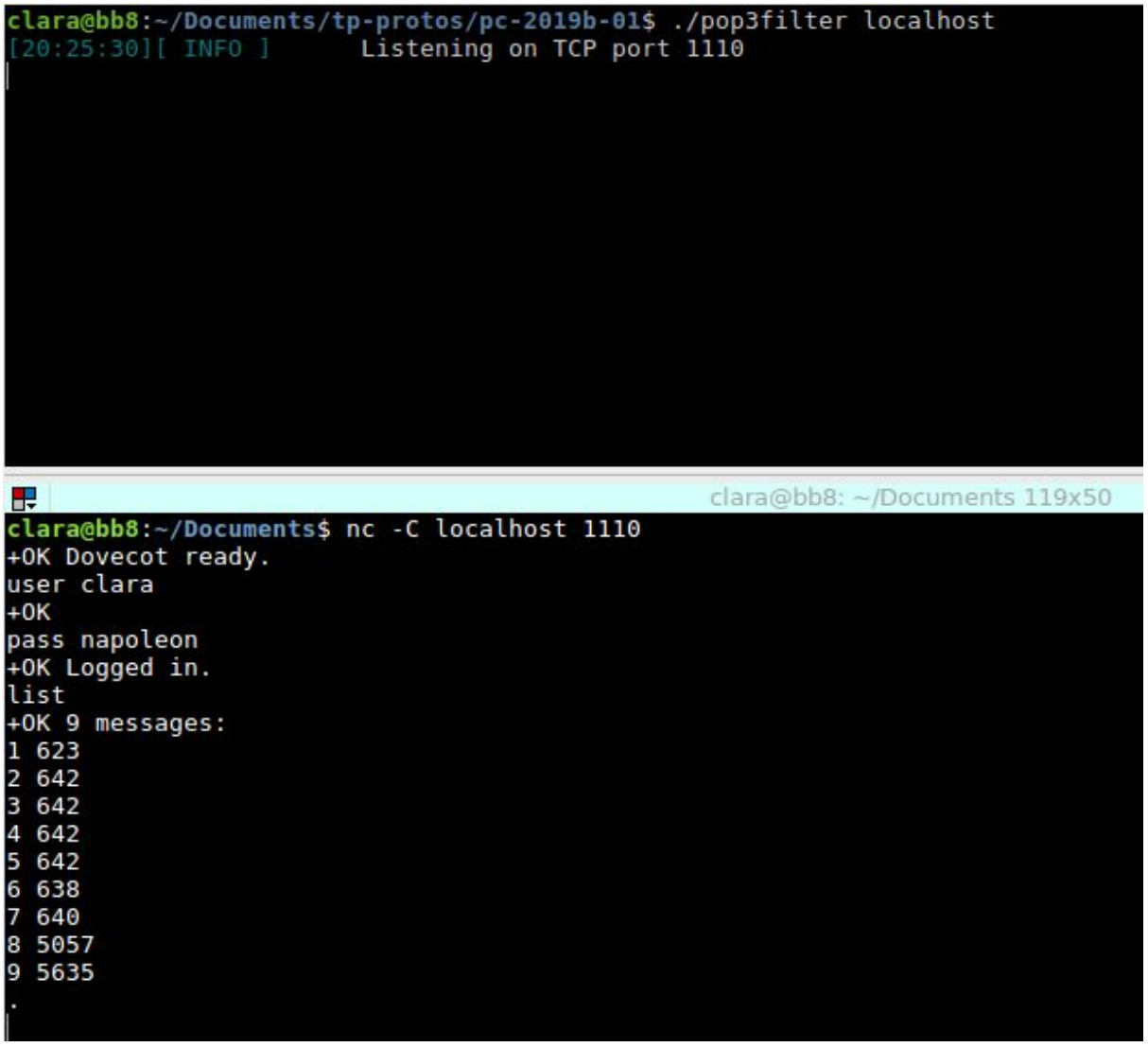
Además, por la forma modular en la que se escribió el código del cliente, este admite el agregado de nuevos comandos de una forma muy simple, por lo que podrían exetenderse de una forma bastante simple nuevos comportamientos para ser agregados.

Conclusiones

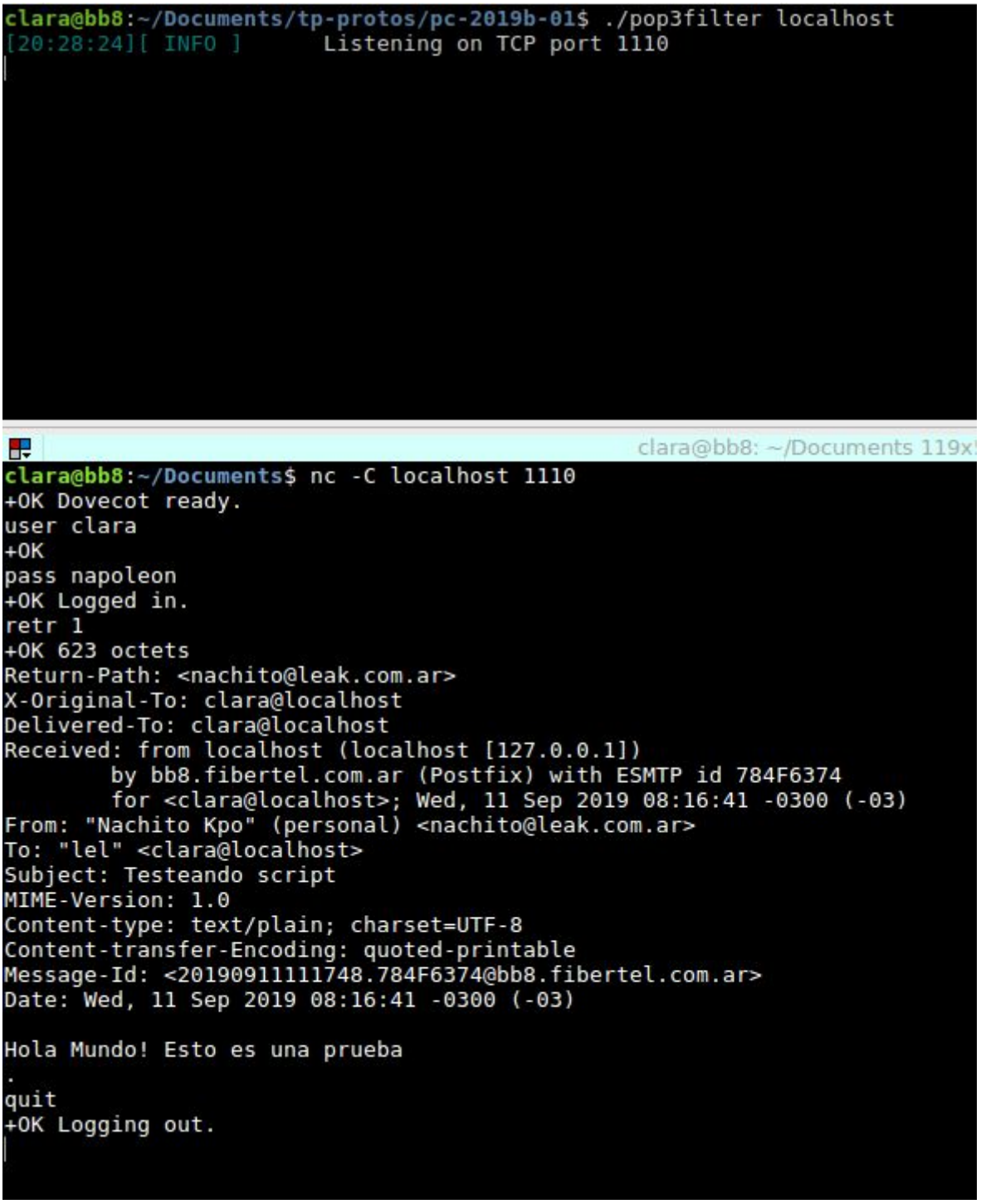
Para concluir cabe notar que la implementación del trabajo práctico nos dio un entendimiento más global del uso de sockets, protocolos y diseño de código. Al mismo tiempo, el desarrollo en bajo nivel nos ayudó a entrenar la capacidad de abstracción y a entender el potencial de las herramientas que provee.

La abstracción que nos proveen las aplicaciones y programas de alto nivel que utilizamos para interactuar con el internet esconden muy bien la complejidad que hay debajo, y este trabajo nos permitió experimentar de manera directa la razón del diseño de los protocolos, y la complejidad en muchos casos de su implementación.

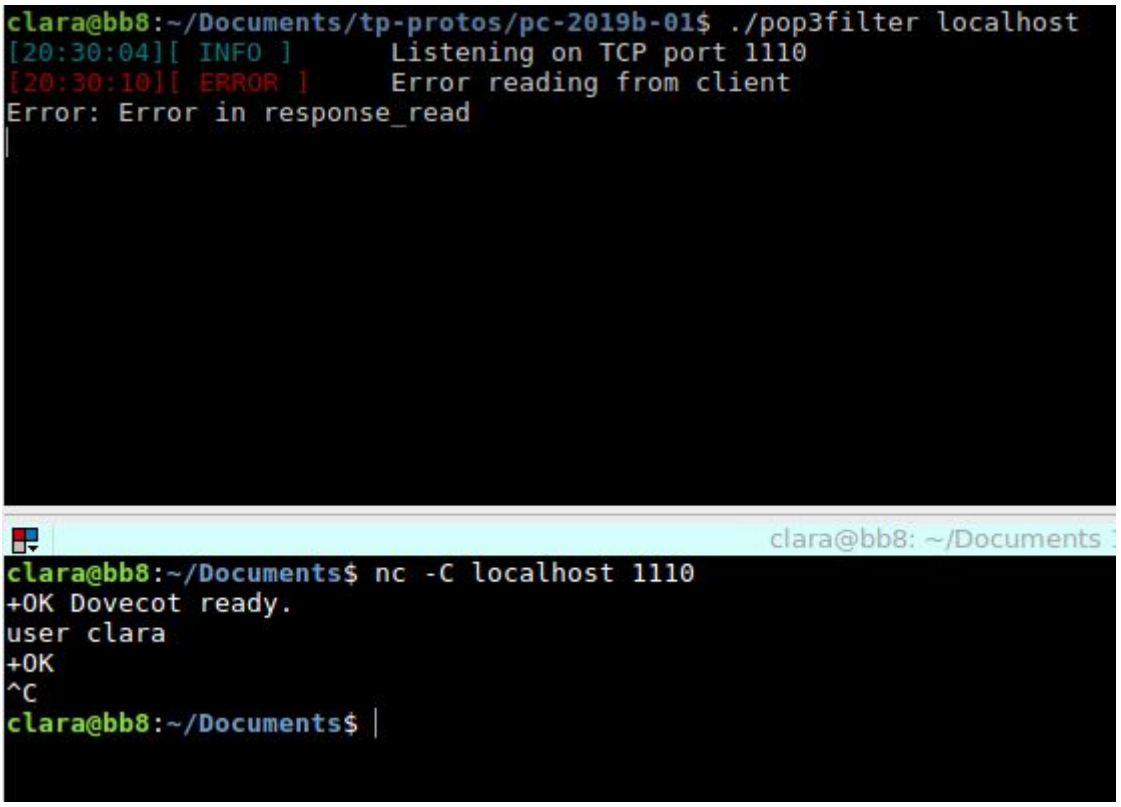
Ejemplos de prueba



Logeo y list


clara@bb8:~/Documents/tp-protos/pc-2019b-01$ ./pop3filter localhost
[20:28:24][INFO]   Listening on TCP port 1110
clara@bb8: ~/Documents 119x
clara@bb8:~/Documents$ nc -C localhost 1110
+OK Dovecot ready.
user clara
+OK
pass napoleon
+OK Logged in.
retr 1
+OK 623 octets
Return-Path: <nachito@leak.com.ar>
X-Original-To: clara@localhost
Delivered-To: clara@localhost
Received: from localhost (localhost [127.0.0.1])
by bb8.fibertel.com.ar (Postfix) with ESMTP id 784F6374
for <clara@localhost>; Wed, 11 Sep 2019 08:16:41 -0300 (-03)
From: "Nachito Kpo" (personal) <nachito@leak.com.ar>
To: "lel" <clara@localhost>
Subject: Testeando script
MIME-Version: 1.0
Content-type: text/plain; charset=UTF-8
Content-transfer-Encoding: quoted-printable
Message-Id: <20190911111748.784F6374@bb8.fibertel.com.ar>
Date: Wed, 11 Sep 2019 08:16:41 -0300 (-03)
Hola Mundo! Esto es una prueba
.
quit
+OK Logging out.

Uso de retr



Desconexión de un cliente

Guía de Instalación

En el README de la aplicación está todo correctamente explicado. De todos modos, un breve resumen:

Correr el comando **make** desde el root del directorio para compilar tanto el proxy como el admin y el filtro (Dentro de la directiva all del root corre los makefiles de los hijos haciendo *$(MAKE) -C [folder]*.

Para correr el proxy basta con desde donde se hizo el make, el siguiente comando:

./pop3filter [args]

Para correr el cliente de administración es necesario tener el proxy corriendo, moverse a la carpeta *admin* y correr el siguiente comando:

./pop3ctl [args]

Para correr el stripmime, moverse a la carpeta *mr-mime* y correr el siguiente comando:

./stripmime [mail]

Dato importante: Las instrucciones de *admin* y de *mr-mime* asumen que anteriormente se corrió el makefile en la carpeta padre. En caso de no hacerlo es necesario correr el makefile de dichas carpetas para poder obtener acceso al ejecutable.

Dato interesante: El Makefile de *mr-mime* una vez generado el ejecutable lo copia a la carpeta padre. Esto es para que esté en el sope del proxy sin necesidad de, cuando se setea dicho programa como la transformación, se tenga que especificar que está en la carpeta *mr-mime*.

Configuración

La configuración del proxy se encuentra en el archivo **config.h** (dentro de la carpeta include) y para mantener consistencia de configuración se utiliza una estructura global *struct config* options*.

Este cuenta con los siguientes campos:

- **local_port**: Puerto en el que van a escuchar los distintos clientes (Default: 1110)
- **origin_port**: Puerto con el que se va a conectar al servidor de origin (default: 110 (dovecot))
- **management_port**: Puerto para conectarse con el cliente de management (Default: 9090)
- **replacement_message**: Mensaje con el que se reemplaza la transformación filtrada
- **version**: Eye candy, marca la versión del programa que está corriendo.
- **proxy_address:** Dirección en la que servirá el proxy (la estructura **struct storage**)
- **managemenent_address**: Dirección donde sirve el servicio de management (la estructura **struct storage**[1])
- **origin_server**: Servidor de origen pop3 (se guarda como string)
- **cmd**: Guarda el proceso que se va a correr al aplicar la transformación
- **media_types**: Establece qué media type se va a filtrar

[1] Se guarda en la estructura struct storage porque esta estructura es lo suficientemente grande para poder almacenar direcciones IPv4 e IPv6