Protocolos de Comunicación

Proxy POP3

Grupo: 1

Participantes:

- Antonielli, Julián
- Biagini, Martín
- Filipic, Joaquín
- Olivera Fedi, Ramiro

Fecha: 14/11/17

Índice

1. Descripción detallada de los protocolos y aplicaciones desarrolladas

Separamos esta primera sección en 3 partes: el proxy en sí (pop3filter), el programa de transformación externa (stripmime) y el de configuración (pop3ctl).

a) pop3filter: el proxy funciona en base a ***pselect()*** (tiene un máximo de conexiones que soporta, 1024) que se encuentra dentro de un ciclo ***while*** constante, manejando las conexiones concurrentemente. Inicialmente el programa setea todas las configuraciones necesarias y recibe los argumentos debidos (utilizados para las configuraciones de puertos, interfaces, etc.), para luego iterar sobre mencionado ciclo.

Dentro del ciclo, se resetean los sets de ***file descriptors*** (correspondientes a los ***sockets***) tanto de lectura como de escritura (los utilizados por el ***pselect***) y se añaden sólo aquellos que deben tenerse en cuenta, cuyos valores se van actualizando al manejarse las operaciones de entrada y salida. A continuación, el ***pselect()*** entra en estado de bloqueo, para desbloquearse frente a algún cambio en los ***file descriptors***. El desbloqueo puede deberse a una nueva conexión o a un requerimiento de escritura o de lectura en un ***socket***. Vale mencionar que se utilizan 2 ***sockets*** pasivos (uno para las conexiones de clientes POP3 y otro para las configuraciones) que derivan las conexiones entrantes a ***sockets*** activos.

En cuanto a las conexiones de clientes POP3, se abre un hilo para manejarlas, ya que la dirección del servidor de origen puede estar en formato IPv4, IPv6, o un nombre de host, para lo cual es necesario realizar una resolución de nombres, que es una tarea bloqueante. Al finalizar la tarea, se envía una señal al hilo principal y finaliza el hilo hijo.

Si el desbloqueo del ***pselect()*** se debió a un pedido de lectura o escritura, se llama a una función (***handleIOOperations()***) que dependiendo del pedido deriva el manejo a distintas funciones puntuales.

Para tratar el funcionamiento básico de envío de datos entre el cliente y el servidor, se hace uso de dos buffers. El cliente le escribe al servidor mediante un buffer, y el servidor le responde de la misma manera, teniendo en cuenta la funcionalidad principal de este proxy: la transformación de mails. Para esto se crea un proceso hijo al cual se le deriva el trabajo de parsear las respuestas y modificar los mails dependiendo de las especificaciones con las que esté configurado el proxy. Para empezar, se crean *pipes* para comunicar ambos procesos (uno en el que escribe el padre y lee el hijo y el otro al revés, con los datos ya transformados). El servidor deja los mails en un archivo que es levantado por el proceso hijo, parseado, y transformado si necesario, dejándolo en otro archivo, cuyo contenido es derivado por el segundo *pipe* al otro buffer para que lo pueda leer el cliente.

Se hace uso además de un log de registro de estado del proxy (al iniciar, al recibir una conexión, al desconectarse un cliente, etc.) y otro de errores.

b) <u>stripmime:</u> Se desarrolló como un programa externo la funcionalidad de parsear los mails. Se toma de un archivo, cuyo path se pasa mediante pipes, el email dejado ahí por el proxy (Que lo obtiene a partir de un RETR). Se elimina la primer línea (Mensaje de estado de POP3) y se parsea el mail utilizando una máquina de estados. Se buscan los headers Content-Type y se utiliza la configuración del proxy para filtrar determinados MIME types y reemplazarlos por un mensaje predefinido. Se combinan stacks y buffers para mantener en el estado el MIME type, soportando así tanto tipos complejos como *multipart,* como tipos simples. Todas las alteraciones se guardan en el mismo archivo leído. En caso de leerse un mail malformado se elimina la respuesta y se deja en el archivo el mensaje *-ERR Connection error or malformed mail.* Finalmente cuando el programa termina, el proxy vuelve a levantar el mensaje retornando al cliente el mensaje procesado.

c) <u>pop3ctl:</u> Se hizo un programa de configuración que recibe argumentos de similar manera a como lo hace el proxy, para poder configurarlo en tiempo real (si se cambian los puertos o las interfaces donde se escucha, se crean nuevos sockets). La forma de comunicarse es a través de un protocolo simple de mensajes en texto plano. Se pueden realizar las mismas configuraciones que al iniciar el proxy, siguiendo los lineamientos del archivo provisto "*pop3filter.8*", sumando además activación, desactivación y obtención de las siguientes métricas: conexiones concurrentes, histórico de conexiones y cantidad de bytes transferidos por los servidores.

A continuación una tabla con los comandos del protocolo y las posibles respuestas:

OS *originServer*	-OK- / -ERROR-
EF *errorFile*	-OK- / -ERROR-
PA *POP3Address*	-OK- / -ERROR-
MA *managementAddress*	-OK- / -ERROR-
RM *replacementMessage*	-OK- / -ERROR-
CT *censurableTypes*	-OK- / -ERROR-
MP *managementPort*	-OK- / -ERROR-
PP *POP3Port*	-OK- / -ERROR-
OP *originPort*	-OK- / -ERROR-
CD *command*	-OK- / -ERROR-

VN	*version*
SCC	-OK- / -ERROR-
RCC	-OK- / -ERROR-
GCC	-Concurrent connections- *n* / -Concurrent connections metrics are off-
SHA	-OK- / -ERROR-
RHA	-OK- / -ERROR-
GHA	-Historical accesses- *n* / -Historical acceses metrics are off-
STB	-OK- / -ERROR-
RTB	-OK- / -ERROR-
GTB	-Transfered Bytes- *n* / -Transfered Bytes metrics are off-

Al pedir alguna métrica, deben estar activadas para recibir su valor, entonces se dejan registradas en un log.

2. Problemas encontrados durante el diseño y la implementación

En esta sección se incluyen las dificultades principales encontradas a lo largo del desarrollo de la aplicación, y se comentan algunas soluciones (en cuanto a limitaciones y posibles mejoras, se lo deja para las siguientes secciones).

- Al recibir una nueva conexión, fue complicado manejar la correcta sincronización entre ambos hilos, principalmente porque el hilo hijo tiene que avisar de alguna manera al padre al realizar la conexión (se usa una señal). El hilo se abre siempre que se reciba una nueva conexión, lo cual no es estrictamente necesario (se debería hacer sólo al tener que resolver un nombre, no si se recibe una dirección IPv4 o IPv6). Además, se debe tener cuidado con el acceso a datos compartidos, por lo que se emplean semáforos para controlar dicho acceso.
- La sincronización y el manejo de concurrencia del proxy para mantener muchos clientes y atenderlos simultáneamente fue un problema relevante. Se debieron tener en cuenta muchos casos de tiempos (fijarse en qué momento alguien puede leer o escribir), analizar los comandos de los clientes y respuestas de servidores, y demás.
- El uso de *pipes* en sí no fue de complicación, pero sí lo fue determinar cuándo enviar las respuestas de los servidores a los procesos hijos a través de ellos y cuándo enviarlos directo a los clientes.
- Manejar los buffers también presentó inconvenientes, ya que con el uso de *pipelining* hay que poder almacenar comandos, determinar cuándo enviarlos, etc.
- Al recibir un mail, no nos resultó sencillo poder analizarlo a medida que se iba dejando en algún archivo, por lo que determinamos guardarlo entero primero y después analizarlo y operar sobre él.

3. Limitaciones de la aplicación

- Si bien se tiene en cuenta que se puede recibir la dirección del servidor origen en formato IPv4, IPv6 o el nombre asociado, lo cual era requerido, lo mismo no aplica para las direcciones de las interfaces en donde escucha el proxy, que deben ser en formato IPv4.
- El protocolo de configuración es demasiado sencillo y no se lo utiliza de manera tal de poder aprovechar el hecho de que la conexión se realiza por SCTP (uso de encabezados para determinar longitudes de mensaje, por ejemplo).
- Al setear inicialmente o a través del programa de configuración los tipos censurables o el comando a ejecutar, y luego se deseara pasar un mail directo sin filtrarlo, se debe reiniciar el proxy (ya que se chequea por NULL, y al setearse, el mail va ser filtrado bajo tales reglas).
- Por simplificación, se decidió utilizar automáticamente *pipelining* con el servidor, si éste lo soporta, una vez enviado el comando CAPA y analizando su respuesta para ver si efectivamente esta función es soportada (si no se deseara por algún motivo, no hay que enviar el comando).

4. Posibles extensiones

- Con respecto a lo mencionado de dejar los mails en un archivo y despúes parsearlo, no sería una extensión pero sí una mejora de performance realizar el parseo de datos de alguna manera simultánea a medida que se deja en el archivo
- Se puede pensar en más métricas para brindarle al usuario más información acerca del uso del proxy, como la cantidad de comandos o bytes totales enviados por un cliente, o pedir por datos de un cliente en específico en vez de datos absolutos, entre otras.
- Se podría guardar el estado de cada cliente (si ya se autenticó o no, por ejemplo) para evitar parseos innecesarios de comandos (no chequear por RETR, por ejemplo, sabiendo que al no estar loggeado se devolverá siempre error).
- El transformador de mensajes actualmente soporta solo reemplazar todos los mensajes de una lista de tipos, por un mensaje en texto plano. Sería interesante agregar la funcionalidad de poder reemplazar los mensajes a filtrar por mensajes de cualquier tipo (por ejemplo, cambiar todos las imagenes por un JSON con metadata de ellas).

5. Conclusiones

Si bien no fueron nada sencillos ni el diseño ni la implementación del proxy, fue un trabajo sumamente interesante, ya que es una aplicación útil, que puede emplearse perfectamente en una situación cotidiana para realizar filtros a pedido sobre mails con posibles intenciones malignas.

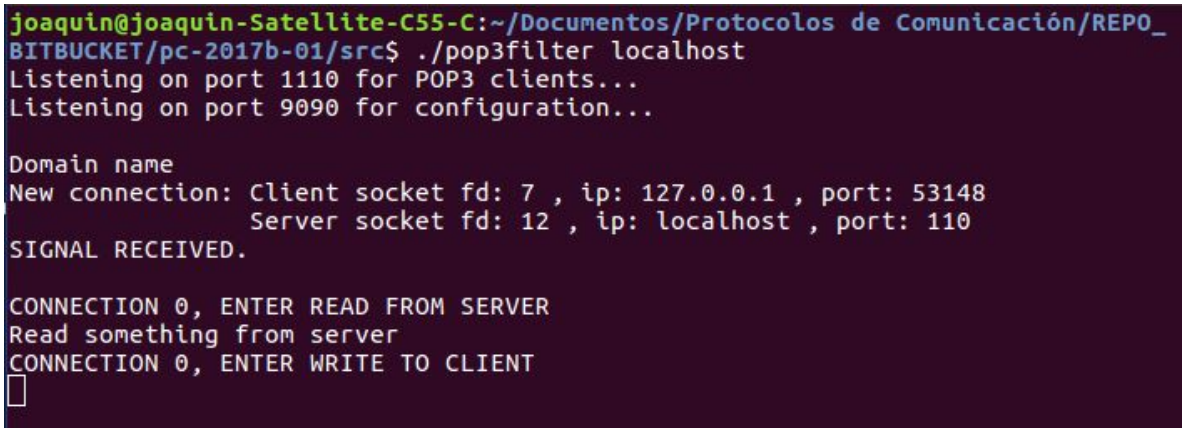
Además, si bien se cumplen las funcionalidades requeridas, es provechoso saber que se pueden ampliar esas funcionalidades o agregar nuevas dependiendo de las necesidades, la aplicación no queda estancada ahí.

La implementación en C fue determinante para entender algunos conceptos de más bajo nivel que no se habrían podido entender si se hubiera programado en JAVA, por ejemplo (creación y uso de sockets, llamadas al sistema, manejo eficiente de memoria, etc.).

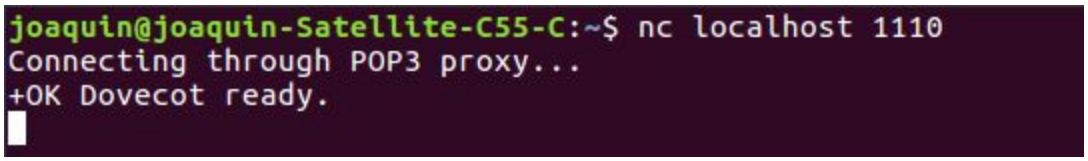
6. Ejemplos de prueba

Se muestran algunos ejemplos de uso sencillos (proxy y clientes).

Comando para correr pop3filter con configuraciones default y con servidor origen en localhost:



Comando para conectarse como un cliente pop3 al proxy que corre en las condiciones anteriores:



Comando para correr pop3filter escuchando en el puerto 1212 para clientes pop3 y en el puerto 9191 para configuraciones, y con servidor origen en 127.0.0.1:

```
joaquin@joaquin-Satellite-C55-C:~/Documentos/Protocolos de Comunicación/REPO_
BITBUCKET/pc-2017b-01/src$ ./pop3filter -p1212 -o9191 127.0.0.1
Listening on port 1212 for POP3 clients...
Listening on port 9191 for configuration...

IPv4
New connection: Client socket fd: 7 , ip: 127.0.0.1 , port: 60864
                Server socket fd: 9 , ip: 127.0.0.1 , port: 110
SIGNAL RECEIVED.

CONNECTION 0, ENTER READ FROM SERVER
Read something from server
CONNECTION 0, ENTER WRITE TO CLIENT
CONNECTION 0, ENTER READ FROM CLIENT
Read something from client
CONNECTION 0, ENTER WRITE TO SERVER
New user: joa
CONNECTION 0, ENTER READ FROM SERVER
Read something from server
CONNECTION 0, ENTER WRITE TO CLIENT
CONNECTION 0, ENTER READ FROM CLIENT
Read something from client
```

Comando para conectarse como un cliente pop3 al proxy que corre en las condiciones anteriores, y algunos comandos pop3 para loggearse:

```
joaquin@joaquin-Satellite-C55-C:~$ nc localhost 1110
joaquin@joaquin-Satellite-C55-C:~$ nc localhost 1111
joaquin@joaquin-Satellite-C55-C:~$ nc localhost 1212
Connecting through POP3 proxy...
+OK Dovecot ready.
user joa
+OK
pass psw
+OK Logged in.
list 1
+OK 1 3288
```

Ejemplo de uso de **mime_parser** (por su cuenta, sin el proxy):

Desde la carpeta **src**, correr el siguiente comando para compilar el parser y settear algunas variables de ambiente necesarias:

export CLIENT_NUM=100 && export FILTER_MEDIAS="images/png" && export FILTER_MSG="Parte reemplazada." && export POP3FILTER_VERSION=0.0 && export POP3_USERNAME=foo && export POP3_SERVER=bar && make

Luego, correr el siguiente comando para ejecutar el parser sobre el archivo de prueba **ii_images.mbox**, y ver la diferencia entre el mail original y el transformado (se borraran las imagenes de tipo **image/png**):

cp ../mime/mensajes/ii_images.mbox ./retr_mail_0100 && unix2dos ./retr_mail_0100 && FILTER_MEDIAS="image/png" ./stripmime; diff retr_mail_0100 resp_mail_0100

7. Guía de instalación detallada y precisa.

Se debe clonar el repositorio, acceder a la carpeta **src** y correr el **makefile**, puede hacer todo esto con el siguiente comando:

```
git clone https://bitbucket.org/itba/pc-2017b-01.git && cd pc-2017b-01/src && make
```

Esto compila todo lo necesario y genera los ejecutables:

- **pop3filter**: es el proxy, se lo llama con los argumentos opcionales y el servidor de origen al final de la siguiente manera:
  ./pop3filter [opciones] servidor-origen
  Y se lo deja corriendo a la espera de conexiones.

- **stripmime**: es usado internamente por el proxy cuando se deba filtrar un mail.

- **pop3ctl**: con el proxy corriendo, se lo ejecuta cuando se desee configurar algo indicando en los argumentos la dirección y puerto del proxy seguido de las opciones a configurar y/o las métricas
  ./pop3ctl dirección-proxy puerto-proxy [opciones]

8. Instrucciones para la configuración.

Si se corre el proxy **pop3filter** sin configuraciones iniciales (con el comando **./pop3filter servidor-origen**) se utilizan las configuraciones por defecto detalladas en ***pop3filter.8***.

Se pueden anexar los siguientes argumentos antes de la dirección del servidor de origen:

-h: Ayuda

-v: Versión

-e 'archivo-de-error' : Archivo donde se redirecciona la salida de errores.

-l dirección-pop3 : Interfaz donde se escucha a los clientes pop3.

-L dirección-de-management : Interfaz donde se escucha para configuración.

-m 'mensaje-de-reemplazo' : Texto de reemplazo en los mails.

-M 'media-types-censurables' : Tipos a censurar en los mails.

-o puerto-de-management : Puerto donde escucha para configuración

-p puerto-local : Puerto donde escucha para clientes pop3

-P puerto-origen : Puerto del servidor origen

-t 'cmd' : Comando a correr por consola en vez de ***stripmime*** siempre y cuando no se indiquen media-types censurables.

Al programa de configuración se los puede llamar con los mismos argumentos más los siguientes, luego de los obligatorios dirección y puerto del proxy:

-s origin-server : Dirección del servidor de origen (nombre, IPv4 ó IPv6)

-1: Devuelve la cantidad de conexiones concurrentes.

-2: Devuelve la cantidad de accesos históricos**.**

-3: Devuelve la cantidad de bytes transferidos por los servidores.

-4: Enciende la métrica de conexiones concurrentes.

-5: Enciende la métrica de accesos históricos**.**

-6: Enciende la métrica de bytes transferidos por los servidores.

-7: Apaga la métrica de conexiones concurrentes.

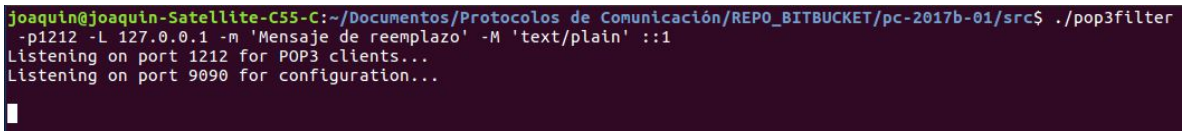
-8: Apaga la métrica de accesos históricos**.**

-9: Apaga la métrica de bytes transferidos por los servidores.

9. Ejemplos de configuración y monitoreo.

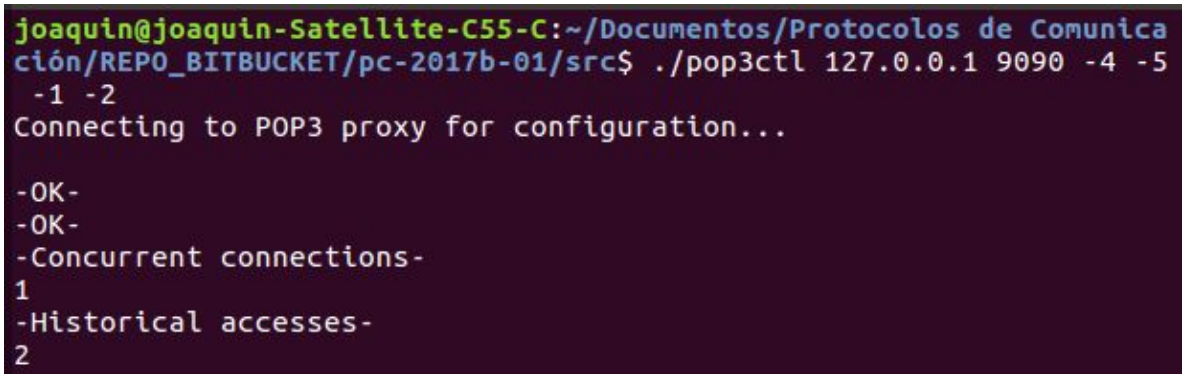
Se muestra un ejemplo de configuración directo en **pop3filter** y otro por medio del archivo de configuración **pop3ctl**. Notar que se tiene corriendo en esta prueba un servidor Dovecot en localhost, a donde se conecta el cliente a través del proxy.

Comando para correr **pop3filter** en el puerto 1212 para clientes pop3, puerto 9090 por defecto para administración, cambiado el mensaje de reemplazo y seteados los tipos MIME para censurar:



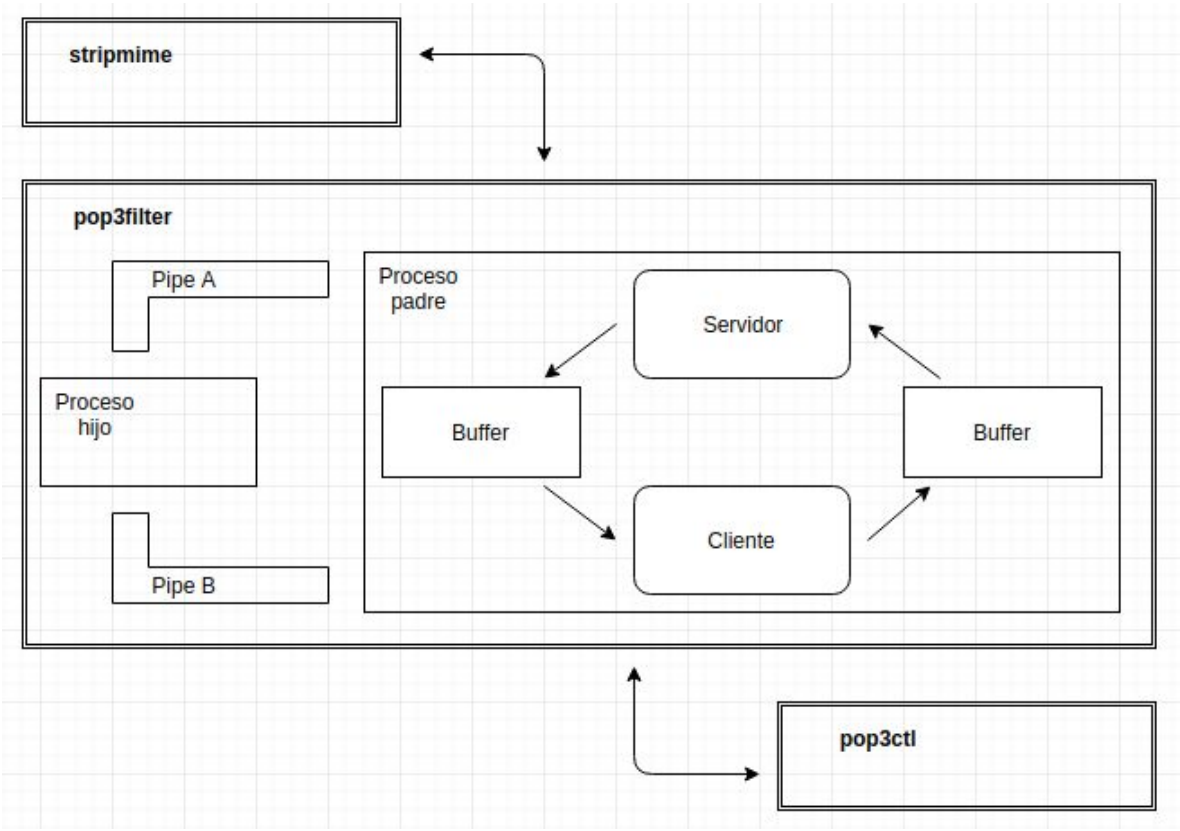
./pop3filter -p1212 -L 127.0.0.1 -m ‘Mensaje de reemplazo’ -M ‘text/plain’ ::1

Comando para correr el programa de configuración **pop3ctl**, activar las métricas de conexiones concurrentes y accesos históricos, y luego obtenerlas:



./pop3ctl 127.0.0.1 9090 -4 -5 -1 -2

10. Documento de diseño del proyecto



En **pop3filter**, el cliente le escribe al servidor. El servidor le responde directamente o por medio del proceso hijo. El Proceso hijo llama a **stripmime**, que transforma los mails. Luego se devuelve el mail transformado al cliente. A su vez, se pueden hacer configuraciones por medio de **pop3ctl**.