Protocolos de Comunicación

Trabajo Práctico Especial

Proxy POP3

Grupo 4

Federico Tedin, Javier Fraire, Ignacio Rivera, Teresa Di Tada

1. Índice

1. Índice 2
2. Descripción de lo desarrollado 3
    2.1. Servidor Proxy 3
        2.1.1. Diseño general 3
        2.1.2. Transformación de mensajes 5
        2.1.3. Multiplexador de cuentas 5
        2.1.4. Reporte de fallos 6
        2.1.5. Monitoreo y configuración remota 6
        2.1.6. Registro de accesos 6
    2.2. Protocolo RCP 7
    2.3. Conexiones POP3 y RCP (Diagramas) 8
3. Problemas encontrados 9
4. Limitaciones 10
5. Posibles extensiones 11
6. Conclusiones 12
7. Ejemplos de testeo 13
8. Guía de instalación 15
9. Instrucciones para la configuración 16
10. Ejemplo de configuración y monitoreo 17

2. Descripción de lo desarrollado

Para el trabajo práctico, se pidió desarrollar un servidor proxy que use el protocolo POP3 para actuar como mediador entre un Mail User Agent (ejemplos: Mozilla, Thunderbird, Microsoft Outlook, Evolution, fetchmail) y un servidor POP3 (ejemplo: Dovecot). Se creó entonces un servidor proxy en Java que cumple con los requerimientos pedidos por la cátedra, y un protocolo (llamado RCP) para poder controlar la configuración del mismo remotamente. A continuación se detalla el diseño y el funcionamiento de ambas implementaciones.

2.1. Servidor Proxy

La tarea principal del servidor proxy para el protocolo POP3 es la de mediar las conexiones entre un cliente y un servidor POP3. Éste se encarga de procesar los comandos enviados por el cliente y evalúa si es necesario que dicho comando sea enviado al servidor POP3; de ser así el servidor proxy se encarga de enviarle la respuesta del servidor POP3 al cliente en cuestión.

Además de esto el servidor proxy cuenta con una serie de características que serán detalladas a continuación.

2.1.1. Diseño general

El servidor proxy utiliza un patrón de diseño similar al del patrón Reactor. El núcleo del servidor utiliza la clase java.nio.Selector de la librería NIO de Java. Cuando se inicia el servidor, se crean dos canales principales del tipo ServerSocketChannel: uno para escuchar conexiones entrantes POP3, y otro para conexiones entrantes RCP. Los dos canales se registran en el Selector, y se crean dos instancias de clases que implementan la interfaz TCPProtocol: POP3SocketHandler, y RCPSocketHandler. El servidor utiliza un mapa para decidir a cual de los dos *handlers* se asignan los canales que están listos para ser operados. Cuando se acepta una conexión nueva, se crea un objeto que actúa como estado de la conexión, que siempre se envía como “attachment”.

Cuando se conecta un cliente POP3, se comienza a leer lo que el usuario envía al servidor. Los contenidos del usuario son separados, si es posible, en líneas. Se intenta interpretar cada línea como un comando POP3, y se decide que hacer dependiendo de ciertos factores.

El servidor, inicialmente, no crea ninguna conexión a ningun servidor POP3. El servidor a conectarse depende de el usuario que el cliente especifique (de acuerdo al

enunciado del trabajo práctico). Por lo tanto, cuando el cliente envía comandos antes de especificar el usuario, el servidor debe manejar la respuesta localmente. Si ya se conoce el usuario, es posible conectarse al servidor POP3, y comenzar a transferir mensajes del cliente al servidor y viceversa. Aunque este es el funcionamiento general del proxy, existen algunas excepciones.

De acuerdo al RFC 2449, el cliente puede utilizar el comando CAPA en las etapas AUTHORIZATION y TRANSACTION. Cuando el cliente se conecta al proxy, se encuentra en estado AUTHORIZATION. Existe entonces un problema si el usuario ejecuta CAPA antes de especificar USER: el proxy no sabe qué lista de capacidades responder, ya que no se encuentra conectado a ningun servidor POP3. Para resolver éste problema, se optó por siempre responder al comando CAPA localmente, con una lista de capacidades predefinidas (configurables via XML). Para el correcto funcionamiento del proxy, ésta lista debe contener al comando USER, ya que es el comando que se utiliza para decidir a qué servidor POP3 se debe conectar. Se asume también que todos los servidores al que el proxy intenta conectarse tienen la capacidad USER. De no ser así, se le enviará al servidor el comando, y el valor retornado siempre será de error.

El tamaño de los buffers de lectura y escritura del cliente y del servidor POP3 tienen tamaño variable, configurable via XML. Con el propósito de poder operar con tamaños de buffer muy reducidos (por ejemplo, 1 byte), se optó con crear un buffer intermedio, auxiliar, de escritura. Cuando se le escribe al cliente o al servidor, los datos primero se escriben a este buffer, y luego se transfieren al de escritura en el momento adecuado. El buffer auxiliar tiene un tamaño predefinido (4096 bytes), pero puede ser modificado modificando un valor en el código fuente del proyecto. Notar que al modificar el tamaño de los buffers originales (no auxiliares) via XML, se observa un cambio en la velocidad de transferencia del servidor. Es decir, la inclusión de los buffers auxiliares no implica que la velocidad del servidor deja de depender del tamaño de los buffers originales.

Aunque el cliente puede mandar varios comandos a la vez, el proxy envía un comando a la vez al servidor POP3, lee la respuesta, y la encola para ser escrita al cliente. Cuando no hay más datos para escribir al cliente, se lo interpreta como una respuesta finalizada. Con el fin de simplificar el código y aumentar la velocidad del proxy, no se realizan muchas pruebas con los contenidos devueltos por el servidor. Solo se comprueba si es necesario aplicar transformaciones al contenido del “Subject” de un mail (como se menciona en el enunciado).

En el caso de un error de lectura o escritura de un canal de cliente o servidor, se cierran los canales necesarios (cliente y servidor). En el caso de un error en los canales que aceptan conexiones, se cierran todas las conexiones existentes y se finaliza la ejecución.

2.1.2. Transformación de mensajes

El servidor proxy del protocolo POP3 transforma los caracteres del campo “Subject” de los mails por otros caracteres especificados por la cátedra. Estas transformaciones se realizan a partir de reglas que pueden ser modificados por el administrador mediante el uso del protocolo de comunicación RCP. Las transformaciones predeterminadas en el servidor POP3 son:

- a por 4 (cuarto)
- e por 3 (tres)
- i por 1 (uno)
- o por (cero)
- c por < (menor)

De la misma manera que se pueden agregar o modificar las transformaciones de caracteres, el administrador puede habilitar o deshabilitar estas transformaciones mediante el uso del protocolo RCP que será explicado más adelante. Las transformaciones y el estado de las transformaciones se encuentran guardadas en un archivo XML de configuración que puede ser alterado mediante el uso del protocolo RCP, o con un editor de texto.

2.1.3. Multiplexador de cuentas

El servidor proxy POP3 le permite a los administradores asignar distintas cuentas de usuario a distintos servidores POP3.

Para agregar usuarios a este sistema, se puede utilizar el protocolo RCP detallado más adelante. Mientras que el servidor se encuentra funcionando, cuenta con un mapa donde las claves son los nombres de usuarios guardados y los valores son los “hostname” de los servidores POP3 asociados a estas cuentas de usuario. Cuando el servidor no esta funcionando los pares ordenados (nombres de usuarios, “hostname”) se guardan en un archivo de configuración XML para que el multiplexado de cuentas sea consistente durante todas las ejecuciones del servidor proxy POP3.

De la misma manera que las transformaciones de texto pueden ser deshabilitadas y habilitadas, el multiplexado de cuentas cuenta con la misma característica. Para esto el administrador del proxy debe utilizar el protocolo RCP para cambiar el estado de la multiplexación de cuentas, o cambiar el valor en el archivo XML.

Se cuenta con un “hostname” predeterminado para todos los usuarios que no tengan uno asociado particularmente. El valor predeterminado es “localhost”, pero también puede

ser modificado mediante el uso del protocolo RCP, o el archivo de configuración. Se puede ver como configurar este servidor predeterminado en la explicación del protocolo RCP más adelante.

2.1.4. Reporte de fallos

En el caso de que el cliente efectúe un comando POP3 inválido, el servidor proxy envía el comando al servidor y le responde al cliente exactamente lo que el servidor le responde. De esta manera se le comunica al User-Agent cualquier error que pueda haber ocurrido en el envío del mensaje. Existe una excepción a lo dicho previamente, cuando el cliente envía un comando que exceda la longitud máxima definida por el protocolo POP3 (255 caracteres), el servidor proxy se encarga de enviar una respuesta local de error sin tener que comunicarse con el servidor POP3. De no estar conectado a un servidor, se devuelve un error si el comando no pudo ser interpretado correctamente.

Para los comandos RCP, el servidor proxy se encarga de descifrar si el comando es válido o no, mediante el uso de una máquina de estado. En el caso de que el comando sea inválido el servidor proxy le responde al cliente con un mensaje de error. Esto se puede ver en la definición del protocolo más adelante.

2.1.5. Monitoreo y configuración remota

El servidor proxy cuenta con un protocolo que se encarga de la configuración y el monitoreo del mismo. Este protocolo es denominado RCP (Remote Control Protocol), que le da al administrador la oportunidad de agregar o eliminar usuarios dentro del mapeo del multiplexado de memoria, diferentes transformaciones de caracteres, entre otras cosas. También permite al administrador obtener métricas y habilitar o deshabilitar el multiplexado de cuentas y las transformaciones del “subject” de los emails. Más información sobre el protocolo RCP será provista más adelante.

2.1.6. Registro de accesos

El servidor proxy del protocolo POP3 cuenta también con un sistema de registro de los usuarios que utilizaron sus servicios. Este logueo se efectúa tanto en consola como en un archivo. Para la implementación de esta funcionalidad se utilizó la librería Open Source “log4j2” que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o “logs”.

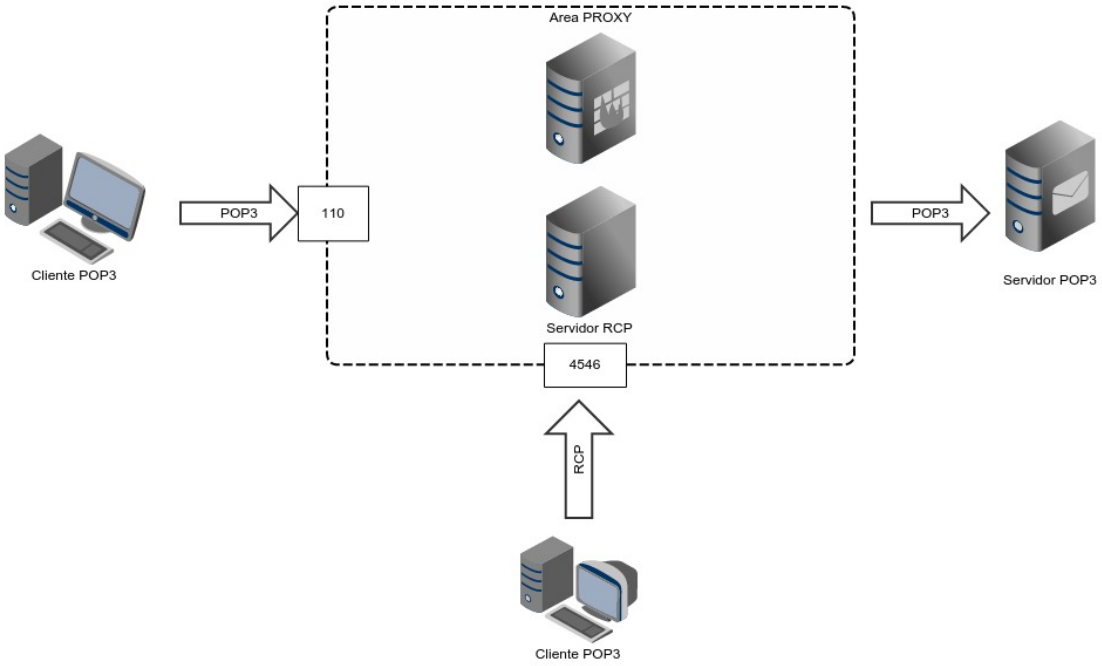
Los archivos de registro de accesos se encontrarán en la carpeta “logs” que se crea automáticamente en el root path.

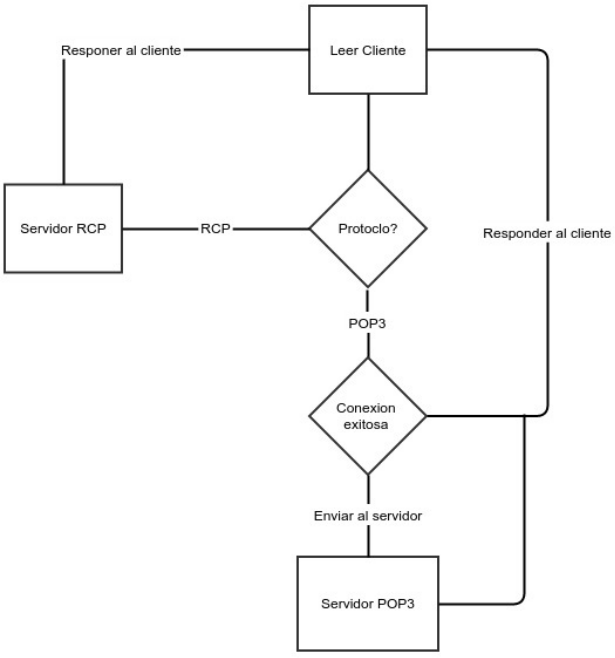
2.2. Protocolo RCP

Para monitoreo y la configuración en tiempo de ejecución del servidor proxy se implementó un protocolo denominado RCP (Remote Control Protocol). Este protocolo es similar a el protocolo POP3. Es un protocolo “request/response” que funciona a través de TCP. El protocolo RCP cuenta con un manual de uso con todos los comandos e instrucciones de como usarlo, que se encuentra en el archivo conteniendo el RFC (especificación) del mismo. Para más información mirar archivo adjunto que contiene el RFC.

El protocolo es muy similar a POP3. Se tomó esta decisión ya que el parseo es parecido y por su simplicidad de uso.

2.3. Conexiones POP3 y RCP (Diagramas)





3. Problemas encontrados

Durante el desarrollo del proxy, se encontraron varios problemas de diseño e implementación.

El primer problema fue mencionado en la descripción general del proxy. Cuando un cliente se conecta, y todavía no especificó su usuario, puede enviar el comando CAPA. Como el proxy no esta conectado a un servidor POP3, debe responder con una serie de capacidades que considere mínima para el correcto funcionamiento del mismo. Ya que el proxy armado utiliza USER como método de registrado, se requiere que los servidores a los que el proxy se conecta también tengan la capacidad USER. En cualquier momento que el usuario pida las capacidades del proxy, el comando será interceptado, y se devolverá la lista definida por el proxy (que se encuentra en un archivo XML).

Otro problema encontrado fue mantener el estado del proxy en cuanto a una conexión cliente-servidor específica. Por ejemplo, si un cliente se identificó y acaba de mandar un comando válido, el proxy debe mandar el comando al servidor POP3. Las tareas pendientes que el proxy debe realizar componen, en parte, al estado de la conexión. Este estado se guardo en forma de “flags” de bits, en dos variables del objeto estado (una para el cliente y otra para el servidor). Si es necesario escribir al servidor e ignorar al cliente, se activa el “flag” “WRITE” del servidor, y se desactiva el “flag” “READ” del cliente. Luego, basándose en estos dos campos, se registran los canales apropiados.

También se encontraron problemas durante la implementación de la transformación de “Subject” de los mensajes. Se decidió que si el mensaje que esta siendo transmitido no cumple con el formato especificado en el RFC 2822, el servidor no puede garantizar que las transformaciones serán aplicadas correctamente, o que sean aplicadas en primer lugar.

4. Limitaciones

Como se mencionó anteriormente, el proxy está limitado al comando USER como método de ingreso para el usuario. El proxy no cuenta con la capacidad de manejar datos encriptados, con lo cual la transferencia de datos protegidos no es posible con la implementación actual.

Otra limitación del proxy es la cantidad de usuarios que pueden estar conectados de forma simultáneas. Utilizando Apache JMeter, se logró estimar que la cantidad máxima de usuarios conectados al mismo tiempo es de alrededor de 5000.

5. Posibles extensiones

Una extensión que podría ser implementada es la de el patrón “Reactor”, utilizando también un conjunto de threads (“thread pool”). El problema de la implementación actual es que, mientras se maneja alguna operación de algún canal, se pueden haber habilitado varios otros canales para operaciones en ese periodo de tiempo. Esto sucede porque todas las operaciones sobre los canales ocurren sobre un mismo thread. Para solucionar este problema, se podria utilizar un número constante, predefinido de threads a los cuales se les puede delegar tareas. Entonces, cuando el Selector devuelve un canal listo para ser operado, se le delega esa tarea a un thread que no esté ocupado. De no haber ningún thread libre, se bloquea hasta que se libere uno. La ventaja de este sistema es que se pueden manejar varios canales a la vez, sin llegar a crear un número de threads elevado que termine consumiendo una alta cantidad de recursos.

Otra extensión posible podría ser la capacidad de PIPELINING, la cual permite al cliente enviar varios comandos simultáneamente, y donde se espera que el servidor responda a cada uno de estos comandos enviados en serie, de forma continua.

6. Conclusiones

Se logró crear un proxy, utilizando la librería NIO de Java, que actúe como mediador entre varios clientes POP3 y varios servidores POP3. Se logró manejar un número elevado de conexiones al evitar el uso de hilos de ejecución, y utilizando en cambio operaciones no bloqueantes. El uso de este tipo de operaciones dificultó el flujo de manejo de los canales, pero igualmente se logró que el cliente y el servidor puedan comunicarse de manera transparente.

7. Ejemplos de testeo

Para probar las transformaciones se utilizaron mails bien y mal formados a la hora de probar el correcto funcionamiento de las mismas. También se utilizaron mails de gran tamaño para ver que el proxy funcionaba de manera correcta y no mostraba ningún tipo de mal funcionamiento.

Una de las pruebas consistió en transferir mails de gran tamaño, modificando el tamaño de los buffers de lectura y escritura de los canales POP3. Utilizando un buffer de un tamaño de 4096 bytes, la velocidad de transferencia fue de aproximadamente 6.2 MB/s. Por el otro lado, cuando se utilizó un buffer de 256 bytes, la velocidad de transferencia fue de aproximadamente 2.1 MB/s. Estos datos demuestran que la velocidad de transferencia del proxy depende del tamaño de los buffers para los canales. Depende del administrador encontrar un balance entre velocidad de transferencia, y cantidad de memoria ocupada por el proxy.

Como se mencionó anteriormente, se utilizó Apache JMeter para medir la cantidad posible de conexiones al proxy (stress testing). Se concluyó que el mismo puede manejar alrededor de 5000 (5322 para ser exacto) conexiones simultáneas. No se pudo probar con más conexiones debido a las limitaciones de los sistemas operativos que limitan la cantidad de “file descriptors” abiertos. Los resultados fueron los siguientes:

Cantidad de conexiones	**“Throughput”**
100	4 MB/s
500	1.6MB/s
900	1.56 MB/s
2000	1.5MB/s (con máximo de 1.93MB/s)
2500	1.46MB/s
3000	1.4MB/s
3500	1.52 MB/s

Hay que tener en cuenta que muchas veces el cuello de botella era el servidor POP3 utilizado (Dovecot). En la mayoría de los casos, el servidor proxy respondía inmediatamente, pero cuando se ejecutaban comandos que tenían que ir al servidor POP3, el servidor respondía con demoras importantes(desde segundos hasta minutos), a veces cerrando la conexión. Por este motivo no se pudieron obtener métricas para más de 3500 conexiones, ya

que el servidor POP3 dejaba de responder. También hay que tener en cuenta, que las pruebas deberían ejecutarse miles de veces para que los resultados sean más confiables.

8. Guía de instalación

El proyecto utiliza maven. Por lo tanto para correrlo ejecutarlo como mvn install. Esto instalará todas las dependendicas y compilará el proyecto. Asegurarse de tener todos los permisos necesarios ya que el programa creara la carpeta logs y la carpeta la carpeta config.

Para ejecutar el programa se debe correr en una terminal con el comando java -jar. Deben estar presentes los archivos de configuración l33tTransformation.xml, stats.xml, config.xml y users.xml para que el servidor pueda iniciar. Estos archivos se generaran automáticamente cuando empiece el servidor. En caso de no hacerlo, deberían estar presentes en la carpeta config en el directorio de trabajo actual.

9. Instrucciones para la configuración

El proxy se configura a través de tres archivos XML.

El archivo principal de configuración es **config.xml,** donde se coloca la configuración general del proxy. Las variables que pueden ser modificadas son las siguientes:

- pop3Port: puerto en el que el proxy escucha conexiones entrantes POP3.
- rcpPort: puerto en el que el proxy escucha conexiones entrantes RCP.
- greeting: mensaje enviado al cliente junto a “+OK” al conectarse.
- pop3Host: dirección a la que el proxy abre sus canales para conexiones entrantes.
- defaultPOP3Server: dirección a la que el proxy conecta a los clientes POP3 por *default*.
- pop3BufferSize: tamaño de los buffers de lectura y escritura para los canales de cliente y servidor de conexiones POP3. Debe ser mayor que 0.
- capaList: lista de capacidades del proxy, enviadas cuando se responde al comando CAPA. Debe contener “USER”.
- password: contraseña para acceder remotamente al servidor vía RCP.
- multiplexingEnabled: permite especificar un servidor POP3 distinto por nombre de usuario.
- l33tEnabled: habilita la transformación de caracteres en el campo “Subject”.

El archivo de configuración del multiplexado es **users.xml**, en donde se debe colocar de forma clave-valor el servidor POP3 al que se quiere que un usuario se conecte a través del proxy.

El archivo **l33tTransformations.xml** permite configurar cuales caracteres del campo “Subject” son transformados a cuales. Se debe especificar cada transformación en la forma de clave-valor, de forma similar al archivo mencionado anteriormente.

10. Ejemplo de configuración y monitoreo

A continuación se incluyen algunos ejemplos de configuración, utilizados durante el desarrollo del proyecto:

config.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<params>
	<pop3Port>4545</pop3Port>
	<rcpPort>4546</rcpPort>
	<greeting>Proxy Server ready.</greeting>
	<pop3Host>localhost</pop3Host>
	<rcpHost>localhost</rcpHost>
	<defaultPOP3Server>localhost</defaultPOP3Server>
	<pop3BufferSize>4096</pop3BufferSize>
	<capaList>USER</capaList>
	<password>password</password>
	<multiplexingEnabled>true</multiplexingEnabled>
	<l33tEnabled>true</l33tEnabled>
</params>
```

users.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
	<userMap>
		<entry>
			<key>mike@aol.com</key>
			<value>localhost</value>
		</entry>
		<entry>
			<key>federicotedin@gmail.com</key>
			<value>192.168.1.4</value>
		</entry>
	</userMap>
</users>
```

l33tTransformations.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<l33tTransformations>
      <transformations>
            <entry>
                  <key>e</key>
                  <value>3</value>
            </entry>
            <entry>
                  <key>r</key>
                  <value>8</value>
            </entry>
            <entry>
                  <key>c</key>
                  <value>&lt;</value>
            </entry>
      </transformations>
</l33tTransformations>
```

A continuación se incluye una parte del archivo de monitoreo generado por la librería log4j2:

```
2014-11-11 14:12:14 INFO Client: /127.0.0.1:51169 connected.
2014-11-11 14:12:14 INFO 25 bytes written to client /127.0.0.1:51169.
2014-11-11 14:12:17 INFO 19 bytes read from client /127.0.0.1:51169.
2014-11-11 14:12:17 INFO Client: /127.0.0.1:51169 executed command 'USER'
with parameters 'mike@aol.com '.
2014-11-11 14:12:17 INFO Server: localhost/127.0.0.1:110 connected.
2014-11-11 14:12:17 INFO 20 bytes read from server localhost/127.0.0.1:110.
2014-11-11 14:12:17 INFO 19 bytes written to server
localhost/127.0.0.1:110.
2014-11-11 14:12:17 INFO 5 bytes read from server localhost/127.0.0.1:110.
2014-11-11 14:12:17 INFO 5 bytes written to client /127.0.0.1:51169.
2014-11-11 14:12:19 INFO 15 bytes read from client /127.0.0.1:51169.
2014-11-11 14:12:19 INFO Client: /127.0.0.1:51169 executed command 'PASS'
with parameters 'password '.
```

Como se puede observar, se llevan a cabo varios registros de eventos que suceden dentro del proxy. La primera línea incluida muestra que se registra cuando un cliente se conecta al proxy. La segunda y tercera línea muestra que también se registra la transferencia de información, en términos de bytes, entre servidor y cliente. Por último, también se registra cuando un cliente ejecuta un comando.