- Protocolos de Comunicación -

Trabajo Práctico Especial

Proxy Server POP3

- Entrega -

AROLFO Franco

MEHDI Tomas

MOZZINO Jorge

Índice

Introducción

Se nos propuso la implementación de un proxy POP3, con ciertos requerimientos especiales.

Para el desarrollo del TP, consideramos que los siguientes RFC nos serán útiles:

- ✓ RFC 1939: Especificación del protocolo POP3
- ✓ RFC 2449: Especificación de un mecanismo de extensión de POP3
- ✓ RFC 2045, RFC 822: Especificación del formato MIME de emails.
- ✓ RFC 1321: Especificación del hash MD5.

Detalles de implementación

El proxy se implementó de la siguiente manera.

Se utilizaron sockets no bloqueantes para todas las conexiones (conexiones de un cliente, conexiones salientes al servidor, conexiones entrantes de administradores). Hemos decidido que todo corra en el mismo thread para evitar problemas de sincronización (y se supone que las conexiones de administradores deberían ser poco costosas, por lo que no influiría mucho en el throughput).

Una vez que se recibe una conexión de un cliente todos los mensajes que se le envíen al servidor se interceptarán para ver si corresponde alguna acción sobre él.

Inicialización de la aplicación

El proxy es requerido con una serie de especificaciones. Algunas de ellas tienen parámetros que podrían ser cargados al inicio de la aplicación. Por ejemplo, el server default del proxy, el puerto al que escucha el servicio de monitoreo remoto, los administradores habilitados para modificar el proxy(ver más adelante), restricciones particulares por usuario o IPs bloqueadas.

Para ellos utilizaremos un documento XML cargado con los datos necesarios, el cual se parsea utilizando jaxb y al inicio de la aplicación y se cargan las variables necesarias con dichos datos.

Se utiliza una interfaz AppConf la cual contiene los métodos necesarios para inicializar usuarios, con sus respectivas restricciones y preferencias, administradores, transformaciones, ip's restringidas y el default server.

La restricción de "from" funciona de las siguientes maneras:

- ✓ Si se inicializa con un email y username es lo mismo que inicializar solo con email, ya que se validará unicamente por macheo exacto del email.
- ✓ Se se inicializa solo el username se valida por que el mail contenga el "user".

NOTA: El XML no es modificado al invocarse el monitoreo remoto.

Sigue un ejemplo de un XML de configuración

Arquitectura del proxy

Como se comentó antes, se utilizaron sockets no bloqueantes para comunicarse con el servidor y el cliente.

Para mantener los estados y variables de la condición, a cada key (instancia de la clase SelectionKey) se le adjuntó una instancia de la clase Session. Recordar que en una conexión virtual entre en servidor y el cliente existen en realidad 2 canales (uno del proxy al servidor, y otro del cliente al proxy). Por este motivo, había 2 instancias de Session por conexión Cliente-Servidor.

En la clase Session se guardan variables como:

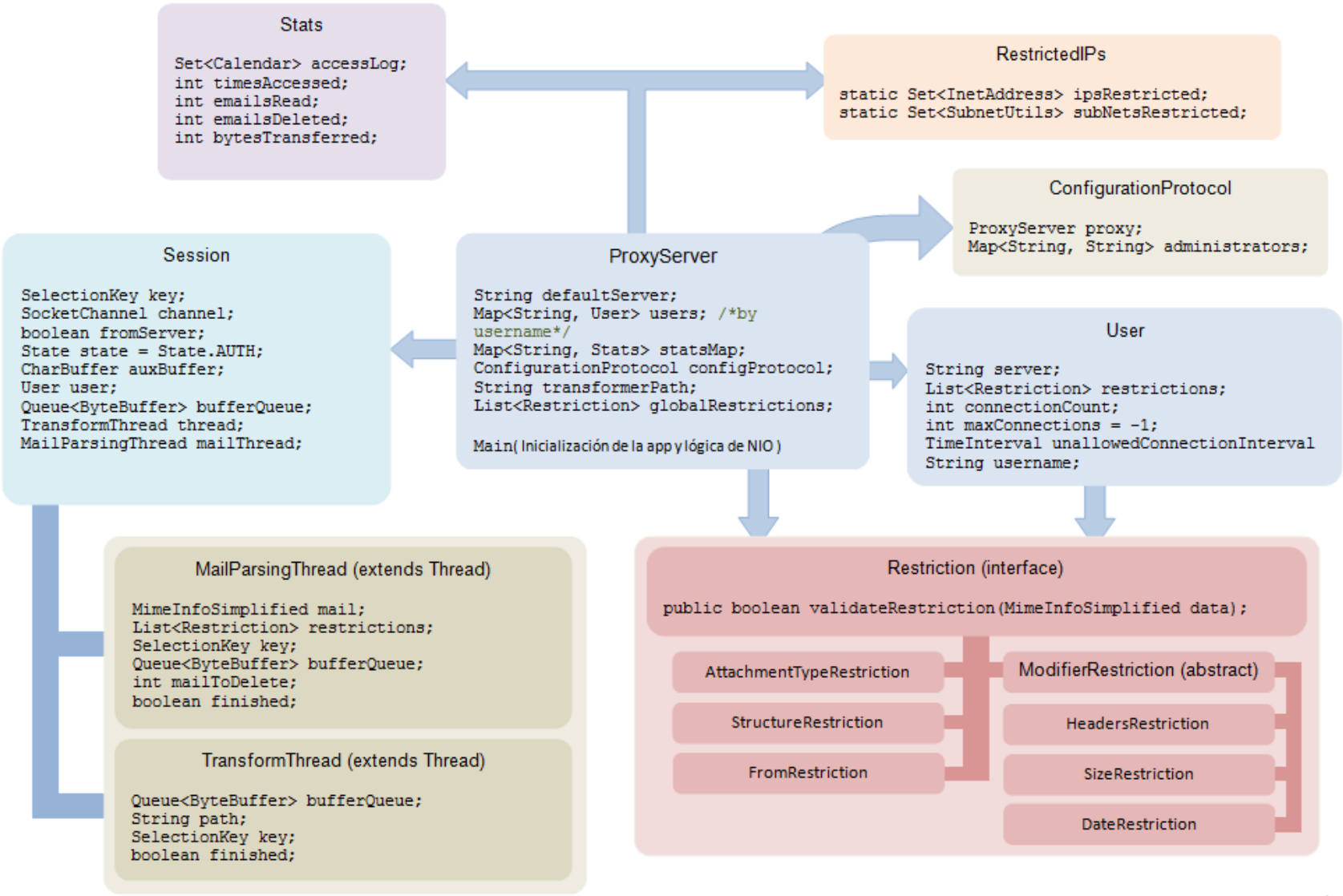
- ✓ La llave de selección simétrica (es decir, si la sesión está del lado cliente, se guarda la llave de la conexión con el servidor y viceversa).
- ✓ El canal simétrico.
- ✓ Un boolean que indica si la sesión está del lado del cliente o del servidor.
- ✓ El estado de la sesión (incluye los estados de POP3 y algunos estados 'custom'.
- ✓ Un buffer auxiliar para procesar el comando.
- ✓ El usuario que corresponde a la sesión.
- ✓ Una cola de ByteBuffers en la que se va encolando lo que se quiere escribir.
- ✓ Referencias a los threads de transformación y para parsear el mail (creados y usados al transformar el mail y al verificar si se puede borrar un mail, respectivamente).

Una vez que el cliente se conecta al proxy, el proxy responderá con -ERR a cualquier mensaje que no sea USER o QUIT. Si se recibe QUIT se cerrará la conexión. Si se recibe USER, se verificará si existe una configuración para el usuario y se creará una conexión con el servidor correspondiente. Si se había recibido un USER anteriormente se cerrará la conexión con el servidor anterior.

Luego del USER, se espera el comando PASS y, una vez que se recibe, se forwardea al servidor y se espera la respuesta. Si se recibe un -ERR se forwardea al cliente. Si se recibe un +OK, se verificará que el usuario no esté restringido para conectarse (por ejemplo, puede haber excedido la cantidad máxima de conexiones o puede no poder conectarse a esa hora).

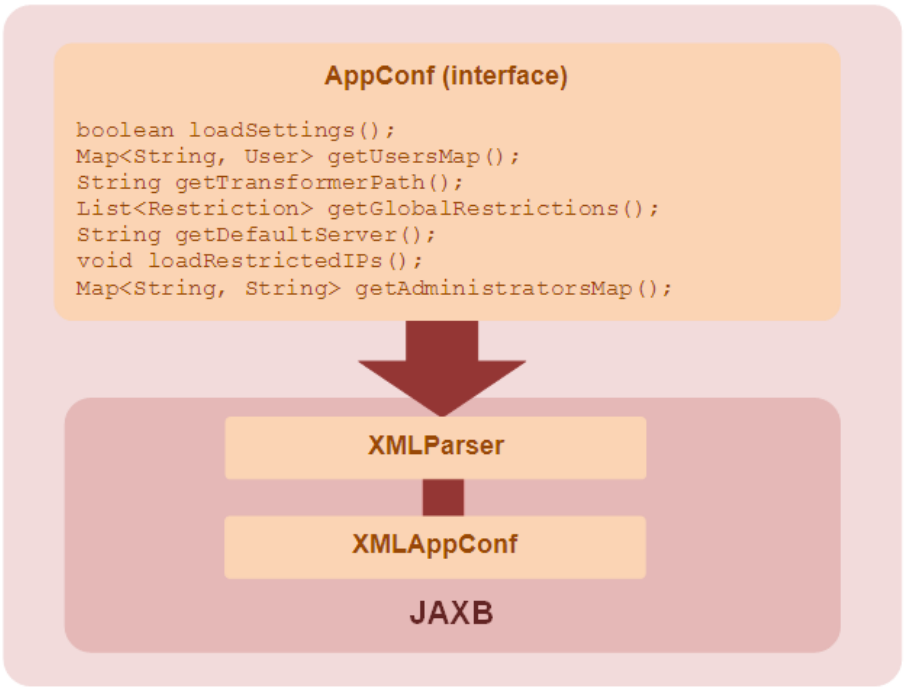
Se decidió validar las restricciones una vez confirmado el PASS ya que, así como el RFC de POP3 sugiere responder +OK para todos los USER para que no se pueda obtener información sobre los usuarios existentes, se necesitaría ser el propio usuario para poder ver cómo está restringido y así un usuario cualquiera no puede tener acceso a las restricciones del resto de los usuarios.

Podemos analizar mejor la arquitectura de la aplicación por medio del siguiente diagrama:

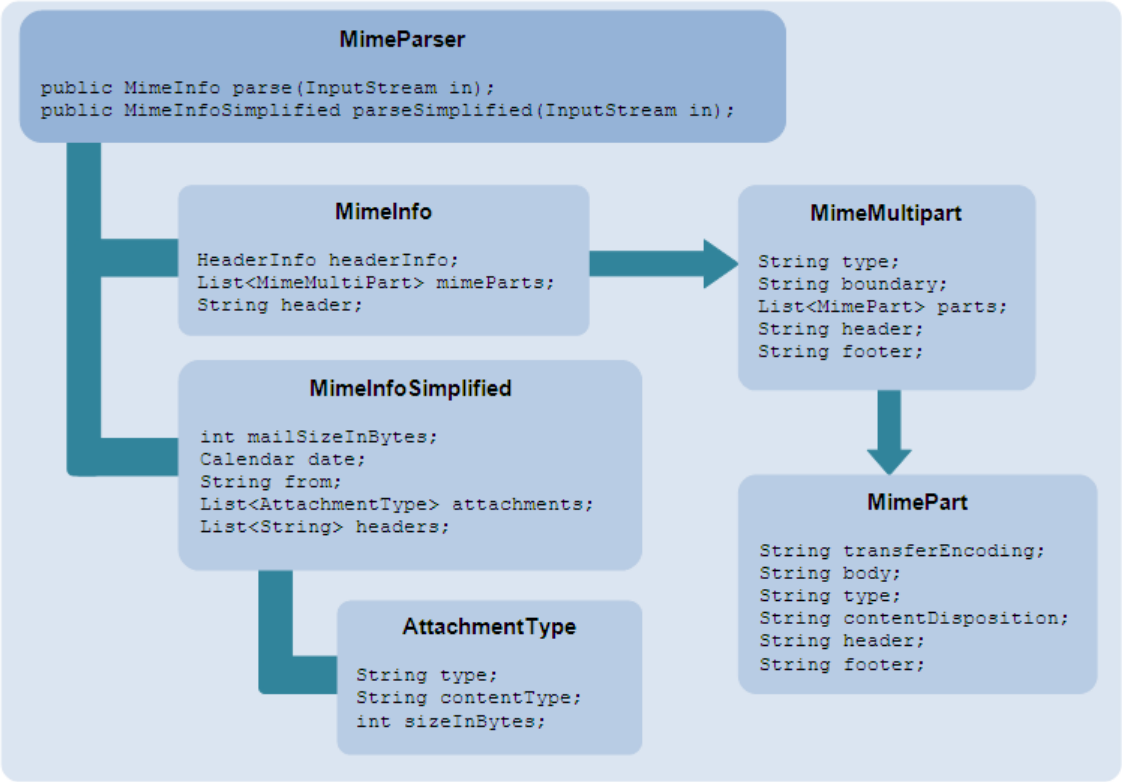

Stats
Set<Calendar> accessLog;
int timesAccessed;
int emailsRead;
int emailsDeleted;
int bytesTransferred;
RestrictedIPs
static Set<InetAddress> ipsRestricted;
static Set<SubnetUtils> subNetsRestricted;
ConfigurationProtocol
ProxyServer proxy;
Map<String, String> administrators;
Session
SelectionKey key;
SocketChannel channel;
boolean fromServer;
State state = State.AUTH;
CharBuffer auxBuffer;
User user;
Queue<ByteBuffer> bufferQueue;
TransformThread thread;
MailParsingThread mailThread;
ProxyServer
String defaultServer;
Map<String, User> users; /*by username*/
Map<String, Stats> statsMap;
ConfigurationProtocol configProtocol;
String transformerPath;
List<Restriction> globalRestrictions;
Main(Inicialización de la app y lógica de NIO)
User
String server;
List<Restriction> restrictions;
int connectionCount;
int maxConnections = -1;
TimeInterval unallowedConnectionInterval
String username;
MailParsingThread (extends Thread)
MimeInfoSimplified mail;
List<Restriction> restrictions;
SelectionKey key;
Queue<ByteBuffer> bufferQueue;
int mailToDelete;
boolean finished;
TransformThread (extends Thread)
Queue<ByteBuffer> bufferQueue;
String path;
SelectionKey key;
boolean finished;
Restriction (interface)
public boolean validateRestriction(MimeInfoSimplified data);
AttachmentTypeRestriction
StructureRestriction
FromRestriction
ModifierRestriction (abstract)
HeadersRestriction
SizeRestriction
DateRestriction

Adicionalmente, estas clases hacen uso de las siguientes clases:

- Para la inicialización de la aplicación, se invoca a una implementación de la interfaz AppConf:



- Para el parseo y procesamiento de mails:



Restricciones especificadas

Para las restricciones por usuario, esto es, cantidad de veces que se puede loguear por día, horarios en los que no puede hacerlo, validaciones acerca de si se puede eliminar un mail o no, o algún server POP3 especifico, se modelo una clase User, que contiene al identificador de cada usuario (su nickname) y clases especificas para el manejo de las restricciones antedichas.

Así, contaremos con un Mapa de username a User, y sabremos los datos referidos a dicho usuario.

En cuanto al manejo de IPs bloqueadas, contaremos con una clase específica para este requerimiento. Dicha clase tendrá las IPs que no tienen acceso, y se deberá poder cargarlas tanto como IPs individuales, como por mascara de subred. Así, solo se necesita consultar a esta clase si alguna IP está bloqueada o no. En particular, la cuestión de la lectura de una máscara de subred será solucionado mediante el uso de librerías openSource aceptadas por la cátedra.

Restricciones de borrado

Cuando se recibe un DELE el proxy lo interceptará y verificará si existe alguna restricción para borrar mensajes. Si existe, primero se hará un RETR del mensaje a borrar, si RETR devuelve +OK, se pasa todo el mensaje a un thread que parsea el mail y luego verifica las restricciones. Si pasan las restricciones, se procede a borrar el mail, si no, se responde -ERR.

Para modelar las restricciones de borrado, se creó una interfaz Restriction con el método validate y en la clase User se guarda una lista de esas restricciones. También se guarda una lista en el proxy para las restricciones globales.

Transformaciones

Para aplicar las transformaciones a los emails, se aplica una lógica similar a la de las restricciones de borrado. Cuando se pide un RETR, se intercepta, se envía un retr al server y se lanza un thread de transformación, que ejecutará la transformación asignada (si no existe, el mail se forwardea directamente al cliente).

A medida que se va recibiendo el mail, se va mandando al thread lo que se recibió. A medida que el programa lee de entrada estándar, el thread le va pasando lo que tiene en la cola. Obviamente, depende del programa transformador, la manera en la que el mail se envía al cliente: si el programa envía a la salida estándar a medida que recibe, entonces el usuario irá recibiendo las partes. Si el programa buferea todo el mail y luego hace el output, el usuario experimentará un ligero lag y recibirá todo el mail.

Estadísticas

Para mantener las estadísticas, se mantiene un mapa de nombre de usuario a una clase Stats, que guarda los siguientes campos:

- Bytes transferidos
- Cantidad de accesos
- Cantidad de mails leídos
- Cantidad de mails borrados
- Fechas de accesos.

Cuando corresponde, se modifican los valores correspondientes.

Entonces, cuando se piden las estadísticas de un usuario, basta con acceder a la instancia de Stats guardada para el usuario.

Si se piden las estadísticas generales, se recorre el mapa y recolecta lo general.

Configuración y Monitoreo Remoto: Protocolo de configuración remota

En cuanto al protocolo para configurar y obtener estadísticas del proxy, se pensó un protocolo muy similar al POP3. Habrá 2 estados. AUTHENTICATION y TRANSACTION.

El servidor escuchará en el puerto 51914.

Los comandos son case-insensitive.

Es orientado a texto de tipo request-response.

Las respuestas multi-línea estarán terminadas con CRLF.CRLF. El único comando con una respuesta multi-línea es STAT.

El protocolo soporta los siguientes mensajes

- AUTH username password
    - Parámetros
        - Username: El nombre de usuario del administrador
        - Password: La contraseña del administrador
    - Descripción
        - Inicia la sesión del administrador. Sólo se puede llamar en el estado AUTHENTICATION.
- SERVER username [server]
    - Parámetros
        - Username: el usuario al cual cambiarle el server destino
        - Server: opcional. El server del cual se pedirán los mails.
    - Descripción:
        - Si se especifica el server se pedirán los mails de username al nuevo server especificado. Si no se especifica, se le asignará el server default.
        - Username puede ser *, en cuyo caso, se aplicará la configuración a todos los usuarios.
- DSERVER server
    - Parámetros
        - Server: el nuevo server default
    - Descripción
        - El server default al cual se pedirán los mails será server.
- RESTRICT username type arguments
    - Parámetros
        - Username: el usuario al cual agregarle la restricción. Si se escribe “*” se entiende que se quiere restringir a todos los usuarios.
        - Type: el tipo de restricción.

        - Arguments: argumentos para la restricción. Varían según el tipo.
    - Descripción
        - Agrega al usuario la restricción del tipo especificado.
        - Los tipos de restricción son:
            - TIME para restringir la hora (los argumentos serían from y to). El usuario no se podrá conectar en ese rango horario.
            - LOGIN para restringir la cantidad de accesos por día (un sólo argumento, el número de logins).
            - DELETE para restringir el borrado de mensajes. Recibe 2 argumentos el tipo de restricción y la condición. Las restricciones son del tipo:
            - DATE restringe por fecha. (el parámetro es relativo, en días). Se podrán borrar únicamente los mails recibidos en los últimos x días, donde x son los días especificados.
            - FROM restringe de quién se pueden borrar mensajes (debe tener dos parámetros, un nombre de usuario y un boolean). El boolean indica si el campo from debe ser idéntico al encontrado en el mail, o una parte de él.
            - HEADER restringe por cabeceras.
            - CTYPE restringe el content type
            - SIZE restringe por tamaño.
            - STRUCT restringe por cómo es el mensaje (los parámetros pueden ser: no attachements o attachments).

- RESTRICTIP ip [mask]
    - Parámetros
        - Ip: La ip a ser bloqueada. Puede ser una ip numérica o un hostname a ser resuelto.
        - Mask: Opcional. Si se especifica, se entiende que se quiere bloquear toda una subred.
    - Descripción:
        - Bloquea las conexiones de la ip especificada.
- STAT [username]

- Parámetros
  - Username: Opcional. Si se especifica, indica que se quieren obtener las estadísticas del usuario.
- Descripción
  - Devuelve las estadísticas del servidor.

- CLOSE
  - No recibe parámetros
  - Descripción
    - Cierra la sesión del administrador

Se pueden pasar más parámetros de los necesarios, pero éstos no serán tenidos en cuenta por el servidor. La única excepción es el mensaje RESTRICT username DELETE type condition. Donde todo lo que viene luego del parámetro type (el tercer parámetro de RESTRICT) será considerado parte de la condición.

Potenciales problemas

Al analizar los RFC, el mayor inconveniente detectado es el de los servicios ofrecidos por un servidor. Dado que es muy probable que el cliente hable con el servidor default hasta que se sepa quién es el usuario que se quiere conectar, es posible que al usuario se le haya asignado un servidor distinto del default. Este servidor podría ofrecer distintos servicios que el servidor default, por lo tanto, respondería distinto al comando CAPA. Si este servidor no ofrece un servicio que sí ofrece el default, es probable que, cuando el cliente pida el servicio (ya que no sabe que se le cambió de servidor en el medio) el servidor no lo pueda responder, entrando a un estado de inconsistencia, ya que supuestamente lo ofrece, pero al pedírselo, no lo tiene.

Una posible solución a este problema, es que cuando se intercepta el comando CAPA, se responde siempre con una lista default, que indica lo mínimo que ofrecen todos los servidores.

Otro problema, es que hay clientes que envían los mensajes encriptados. Para usar este proxy, va a ser necesario deshabilitar el cifrado de paquetes.

Transformaciones de Mensajes

Para las transformaciones, nos valdremos del path de cada programa a ejecutar, y la librería provista por java para su ejecución.

Los programas serán cargados en el inicio de la aplicación, por medio del XML, o por uso del monitoreo remoto.

Lectura de mails

En cuanto al parseo de mails, se implementaron las siguientes clases: MimeParser, MimeInfo, MimeMultipart, MimePart, HeaderInfo. El mime parser tiene un método "parse()" que recibe un InputStream y hace streaming del mail obteniendo la información y almacenando en las distintas clases mencionadas anteriormente. La información del header mime la inserta en HeaderInfo, la cual es asignada a una variable de MimeInfo. Esta misma, además de tener una variable headerInfo, tiene una lista de objetos MultiPart. Cada multipart tiene su tipo y una lista de MimePart. El mimepart posee un body, donde se ubica todo aquello que no sea un encabezado del mime.

Problemas encontrados:

Al comienzo de la implementación del parser no se consideró que un mail pudiese tener varios mimeMultipart, lo que genero un fallo al parsear un mail estándar. Al notarlo se mejoro la implementación, haciendo una llamada recursiva en los momentos que aparecía el encabezado "ContenType: multipart/*".

Luego, el otro inconveniente es que hay una gran cantidad de types, por lo tanto no se analizaron todos los casos. En dichos casos, el contenido de esos encabezados termina en el body del MimePart correspondiente.

Al utilizar un InputStream se complico el reconocimiento del fin del mail, pero se soluciono determinando que terminan con un ctrl+d. Para hacerlo el toread que pardea al terminar de leer con un .\n envía la señal

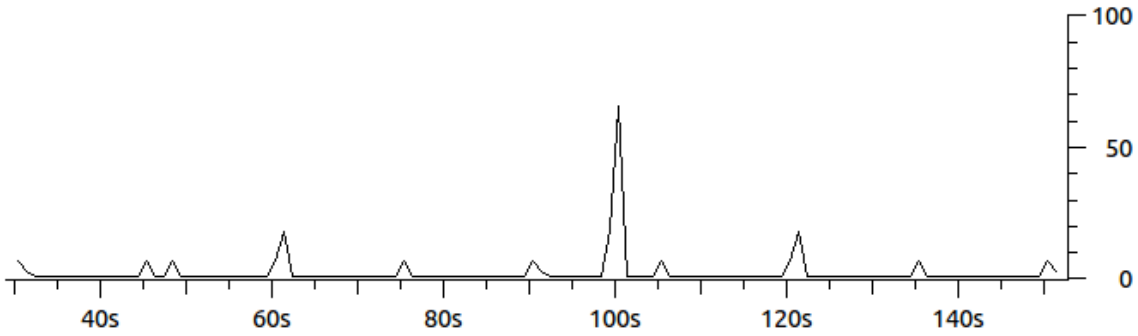
Limitaciones

Se especuló en un primer momento con que se podía implementar el APOP. Sin embargo, luego de leer el RFC correspondiente a MD5 y un poco de investigación, se concluyó que esto no era posible. Una posibilidad es darle a los usuarios la opción de que guarden sus contraseñas en el proxy, pero se descartó por las consecuencias a la seguridad que hacer esto tiene.

Si bien se usa nio, en todas las request que requiren de un RETR se lanza un toread aparte. Entonces, una de las ventajas de nio, que es que no se usa mucha memoria, ya que no hay toread, se ve oscurecida por la necesidad de threads para otra operación.

Testeo de throughput

Se hizo un retr de gran tamaño, y se analizó con el Wireshark el throughput de dicho request.



Se intentó llevar a cabo una prueba de carga del proxy, pero por limitaciones propias y del netcat (se envían todos los comandos juntos y nuestro proxy pierde el resto de los comandos), no se pudo automatizar la secuencia de comandos. Por este motivo, debimos realizar una prueba más pequeña: abriendo varios MUAs a la vez (netcat y thunderbird), se hacían pedidos al server. Cualitativamente, el proxy respondió bien, no se notó diferencia de performance (excepto cuando se aplicaban transformaciones). Consideramos que en una mayor escala el proxy respondería bien.

Ejemplos de testeo

Se probaron los siguientes escenarios:

- Concurrencia: Se probó con multiples MUAs a la vez. Previamente habíamos testeados con varias consolas abiertas y valiéndonos del programa netcat.
- Restricciones:
  Se setearon distintas restricciones y se enviaron mails para verificar su funcionamiento:
  - From: Se enviaron mails desde el email banneado y desde un email sin bannear a un mail con restricciones.
  - Size: Se enviaron mails con archivos adjuntos tales que, unos superaban el size permitido en la restricción y otros no.

  Y se sigio de manera análoga con las otras restricciones.

- Transformaciones:
  Se enviaron mails con imágenes en los adjuntos para el rotado de imágenes, o texto plano en el body para el l33t. En cuanto a la de anonimación, basto con enviar cualquier mail.

Conclusiones

Se apreció la importancia de los RFC, ya que al ser tan claros y no dar lugar a interpretación generan convenciones muy útiles. Logrando así un proxy server bastante completo y con una gran capacidad de usuarios ya que se utilizaron sockets no bloqueantes.