Informe TPE2

Protocolos de Comunicación

Integrantes:

- Julián Francisco **Arce**, 60509
- Roberto José **Catalán**, 59174
- Paula Andrea **Domingues**, 60148
- Gian Luca **Pecile**, 59235

Índice

Protocolos y aplicaciones desarrolladas

Se desarrolló en primer lugar un servidor proxy para el protocolo POP3 (Post Office Protocol versión 3) [RFC1957] el cual puede ser usado por diversos MUAs (Mail User Agents), tales como Dovecot o Mozilla Thunderbird, se le implementó a su vez un sistema de pipelining ([RFC2449] sección 6.6) para dicho server y posee mecanismos que permiten transformar los correos electrónicos utilizando aplicaciones externas. Además, se hace uso de nuestro propio protocolo, basado en UDP como protocolo de transporte para los usuarios administradores los cuales deben ser autenticados.

Pop3-Proxy

El servidor proxy de protocolo pop3 es un servidor concurrente con entrada salida multiplexada no bloqueante. Durante el desarrollo de dicho servidor se hizo uso de código fuente aportado por la cátedra.

En principio se crean dos sockets pasivos para escuchar conexiones entrantes para tanto IPv4 como IPv6 y se reserva un socket más -que no es pasivo ya que envía y recibe datagramas constantemente- para nuestro protocolo UDP en el proceso. Una vez configurados los mismos, se los registra en el selector junto con los handlers para manejar conexiones entrantes, escrituras, lecturas y para el envió (al igual que recepción) de datagramas UDP.

El servidor recibe opciones como parámetros por línea de comando respetando el estándar POSIX. Se hizo uso de la función *getopt()*, que se encuentra dentro de las opciones soportadas son las definidas en el documento *pop3filter.8* aportado por la cátedra.

Por último, en cuánto a la eficiencia y uso de memoria se armó un *pool* para permitir el mejor manejo de alocación de memoria de manera eficiente al hacer uso de varias estructuras complejas que almacenan información, parsers, etc.

Nuestro protocolo

Se tomó como protocolo de transporte UDP debido a la previa experiencia en el TP1, usando los conocimientos obtenidos resulta el más simple de tratar y por ende no está orientado a bytes, sino a datagramas ya que el hecho de que la información necesaria para la comunicación efectiva con el protocolo cabe dentro de un datagrama. Debido a que hace uso de UDP para transporte, no está orientado a conexión ya que hace más simple el manejo de usuarios.

Se hace uso de autorización para ingresar, por ende se requiere que el usuario administrador ingrese una contraseña válida de seis caracteres previo a cualquier comando que quiera utilizar. La misma puede ser editada mediante el uso de *SET_AUTH* pero su valor predeterminado es 000000.

Para el manejo de las responses obtenidas se encuentra basado en POP3 que se recibe como respuesta lo siguiente:

Estado	**Descripción**
+OK	Respuesta esperada.
-ERR	Razón de error.

Como se mencionó anteriormente, todos los comandos requieren de la inserción de la contraseña de administrador previo a su ejecución.

Comandos **GET**

```
000000 GET_BUFF_SIZE
```

- Tamaño del buffer a ser usado.

```
000000 GET_STATS
```

- Cantidad de conexiones históricas.
- Cantidad de conexiones concurrentes.
- Cantidad de bytes transferidos.
- Tiene el formato:

```
+OK    %d      %d      %d
```

De ser exitoso al ser ejecutado el comando con la representación numérica de lo mencionado. Se observan primero las conexiones históricas, luego las conexiones concurrentes y por último la cantidad de bytes transferidos.

Comandos **SET**:

```
000000 SET_AUTH NEWPAS
```

- Cambiar la contraseña del administrador ingresando la anterior (en este caso 000000) y luego del comando la nueva contraseña (en este caso NEWPAS). La misma debe constar si o sí de seis caracteres.

Comando **HELP**:

```
000000 HELP
```

- Lista todos los comandos que tiene a disposición el admin e información de contacto.

Problemas encontrados

En primera instancia, el manejo de la máquina de estados aportada por la cátedra ya que posee una complejidad mayor que lo usado previamente.

El manejo de hilos para situaciones no bloqueantes. En particular, sucedió que a la hora de crear un hilo que recibía como parámetro data una key, nos encontrabamos pasando mal dicho parámetro ya que se perdía al finalizar la función creadora del hilo, al consultar en clase se vio que, gracias a la cátedra, se podía solucionar creando una key en el heap y pasando la nueva key alocada en el heap al nuevo hilo.
Cuando un cliente se desconecta sucedía que no se manejaba correctamente el estado de la conexión resultando en una acumulación de estados no deseados y si bien el servidor funciona correctamente y atendiendo pedidos, no se libera de la manera correcta el cliente desconectado.
Al haberse conectado al proxy con un cliente MUA, si se manda un comando como retrieve y no se inició sesión, la respuesta es unilínea lo cual generaba una falta de respuesta del proxy que resultaba en una salida de ejecución.

Limitaciones

La principal limitación encontrada fue la falta de conocimiento sobre los temas complejos a tratar y el mal manejo del tiempo a lo largo del desarrollo del trabajo práctico.
En el caso de CAPA para averiguar si el origin soporta pipelining: Se tomó la decisión de no administrar el caso en el que el buffer se llene mientras va leyendo CAPA enviado por el origin. Teniendo en cuenta que el comando capa podría decirse que su longitud es conocida y el tamaño de nuestro buffer es mayor se decidió tomar el riesgo de no administrar el llenado del buffer en esta parte para no realizar código más complejo del necesario a la hora de obtener está información.
En cuanto a las funcionalidades de transformación, las mismas no se encuentran disponibles ya que fue implementada una versión inicial que no llegó a tener la funcionalidad necesaria para ser agregada a la versión final del proyecto.
No se logró brindar la funcionalidad de conexión vía UDP con sockets de IPv6 aunque funciona correctamente con IPv4.

Posibles extensiones

En cuánto al admin se podría implementar la encriptación de los datos sensibles como credenciales al ser ingresadas ya que pueden ser fácilmente robadas en la implementación actual, mejorando la seguridad general del protocolo. Además, se agregaría el uso de status codes a nivel protocolo para el mejor manejo de errores, estos a su vez podrían ser incluidos en el comando *getstats* que se pensaba agregar estadística de cantidad de fallas/errores pero resultó como posible extensión del proyecto. Otro comando similar al previamente mencionado que se podría agregar involucraría una lista de las conexiones del momento con IP, tiempo de inicio, cantidad de comandos ejecutados, etc.

La implementación correcta de la funcionalidad del transformer es una extensión a ser agregada.

Mejoras a la calidad de código a través del uso de herramientas scan build de clang o incluso mecanismos de testeo más robustos que se encuentren basados en Test Driven Development (TDD) con testeos unitarios de diversos componentes del sistema.

Conclusiones

En síntesis, a lo largo de la realización del trabajo práctico especial se hizo uso de conceptos de materias anteriores como sistemas operativos o arquitectura de las computadoras que fueron de utilidad y a la vez se logró poner en práctica el conocimiento adquirido a lo largo de las clases teóricas. Dicha puesta en práctica no fue sin sus dificultades y fue facilitada en parte gracias al desarrollo del primer trabajo práctico pero en general se puede decir que los conceptos más importantes desarrollados a lo largo de esta instancia de trabajo fueron el uso y manejo de protocolos como pop3 o tcp al igual que la implementación de uno propio en base a las necesidades del TP y el manejo más avanzado de sockets al igual que herramientas de debugging para solucionar problemáticas.

Instalación

Para poder correr el proyecto se hace uso de makefiles a modo de linkeditar y compilar el contenido. En la carpeta principal al realizar el comando:

```
make
```

Luego se obtienen los archivos ejecutables que se deben correr en el mismo directorio. En este caso el archivo *pop3filer* el cual debe ser ejecutado para correr el servidor y el archivo *pop3ctl*, el cual debe ser ejecutado con el fin de correr el cliente que hace uso de nuestro protocolo UDP. A continuación se detalla sobre la configuración y ejecución de los mismos que además está presente en el archivo *README.md* encontrado en este mismo directorio.

Configuración

Se debe correr el ejecutable generado previamente por el comando make y por defecto, como se puede ver en detalle en la parte de diseño del proyecto, los MUA por defecto toman el puerto 1110, nuestro protocolo el 9090 y la conexión pop3 al origin server en el puerto 1010. Los comandos para cambiar los mismos son los siguientes:

Comando	**Descripción**
-e	Especifica el archivo donde se redirecciona stderr de las ejecuciones de los filtros. Por defecto el archivo es /dev/null.
-h	Imprime la ayuda y termina.
-l	Establece la dirección donde servirá el proxy. Por defecto escucha en todas las interfaces.
-L	Establece la dirección donde servirá el servicio de management. Por defecto escucha únicamente en loopback.
-o	Puerto donde se encuentra el servidor de management. Por defecto el valor es

	9090.
-p	Puerto TCP donde escuchará por conexiones entrantes POP3. Por defecto el valor es 1110.
-P	Puerto TCP donde se encuentra el servidor POP3 en el servidor origen. Por defecto el valor es 110.
-t	Utilizado para las transformaciones externas. Compatible con system(3). La sección FILTROS describe como es la interacción entre pop3filter y el comando filtro. Por defecto no se aplica ninguna transformación.
-v	Imprime información sobre la versión versión y termina.

Ejemplos de configuración, prueba y monitoreo

Prueba de MUA

La siguiente imagen muestra la configuración del mail user agent dovecot habiendo corrido en *localhost* el servidor proxy y conectandose luego al puerto 1110 via TCP desde una segunda terminal.

```
Soy el fd->5                                  juarce@juarce-VirtualBox:~/Descargas$ nc localhost 1110
Nunca entramos al capa_finished               +OK Dovecot (Ubuntu) ready.
Entramos al capa_read                         CAPA
Soy el fd->5                                  +OK
Nunca entramos al capa_finished               CAPA
Entramos al capa_read                         TOP
Soy el fd->5                                  UIDL
Nunca entramos al capa_finished               RESP-CODES
Entramos al capa_read                         PIPELINING
Soy el fd->5                                  AUTH-RESP-CODE
Nunca entramos al capa_finished               STLS
Entramos al capa_read                         USER
Soy el fd->5                                  SASL PLAIN
Nunca entramos al capa_finished               .
Entramos al capa_read                         USER juarce
Soy el fd->5                                  +OK
Nunca entramos al capa_finished               PASS [redacted]
Entramos al capa_read                         +OK Logged in.
Soy el fd->5                                  LIST
Nunca entramos al capa_finished               +OK 2 messages:
Entramos al capa_read                         1 716
Soy el fd->5                                  2 715
Nunca entramos al capa_finished               .
Entramos al capa_read                         LIST 1
Soy el fd->5                                  +OK 1 716
Nunca entramos al capa_finished               RETR 1
Entramos al capa_read                         +OK 716 octets
Soy el fd->5                                  Return-Path: <juarce@locahost>
Nunca entramos al capa_finished               X-Original-To: juarce@localhost
Entramos al capa_read                         Delivered-To: juarce@localhost
Soy el fd->5                                  Received: from [127.0.0.1] (localhost [127.0.0.1])
Nunca entramos al capa_finished                       by juarce-VirtualBox (Postfix) with ESMTP id 32116103911
Entramos al capa_read                                 for <juarce@localhost>; Mon, 22 Nov 2021 10:09:50 -0300 (-03)
Soy el fd->5                                  To: juarce@localhost
Nunca entramos al capa_finished               From: Julian Arce <juarce@locahost>
Entramos al capa_read                         Subject: asdasdasd
Soy el fd->5                                  Message-ID: <73ddc1b1-26e2-e917-95e1-71b315613f6a@locahost>
Nunca entramos al capa_finished               Date: Mon, 22 Nov 2021 10:09:49 -0300
Entramos al capa_read                         User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101
Soy el fd->5                                   Thunderbird/78.11.0
Nunca entramos al capa_finished               MIME-Version: 1.0
Entramos al capa_read                         Content-Type: text/plain; charset=windows-1252; format=flowed
Soy el fd->5                                  Content-Transfer-Encoding: 7bit
Nunca entramos al capa_finished               Content-Language: en-US
Entramos al capa_read
Soy el fd->5                                  qwdqwdqwdqwdqwdqdqwdqdqwdqwd
Pipelining encontrado
Probando                                      .
```

Imagen 1: MUA en funcionamiento con prints de debugging.

Configuración y prueba de admin

Las siguientes imágenes muestran la configuración del admin y el monitoreo de los datos en base al uso del proxy.

```
pau@mango:~/Documents/2021_2Q/Protos/Protos-TP2E$ ./main -P 1111 localhost
Argument of option '-P' is 1111.
No errors on input
UDP Listener on port 9090
Listening on TCP port 1110
Listening on TCP port 1110
Waiting for incoming connection...
```

Imagen 2: Luego de compilar el proyecto, se ejecuta en el puerto 1111 debido a que el 1110 se encuentra ocupado en este caso.

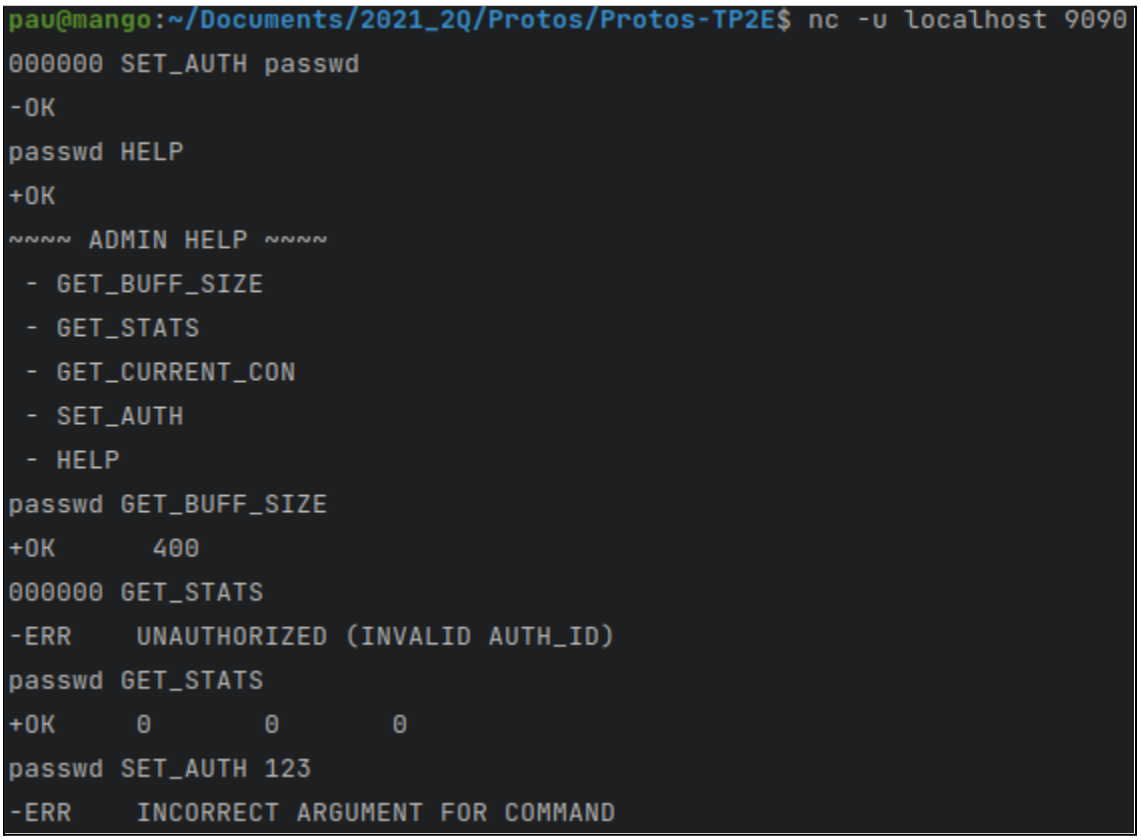

Imagen 3: El usuario administrador procede a realizar la autenticación ingresando su contraseña, posteriormente cambiando la misma y luego realiza un pedido de las estadísticas de monitoreo.

Documento de diseño del proyecto

A continuación se puede ver un diseño que refleja el flujo del sistema y los puertos, al igual que protocolos que se usan al correr el ejecutable generado sin realizar ningún cambio.

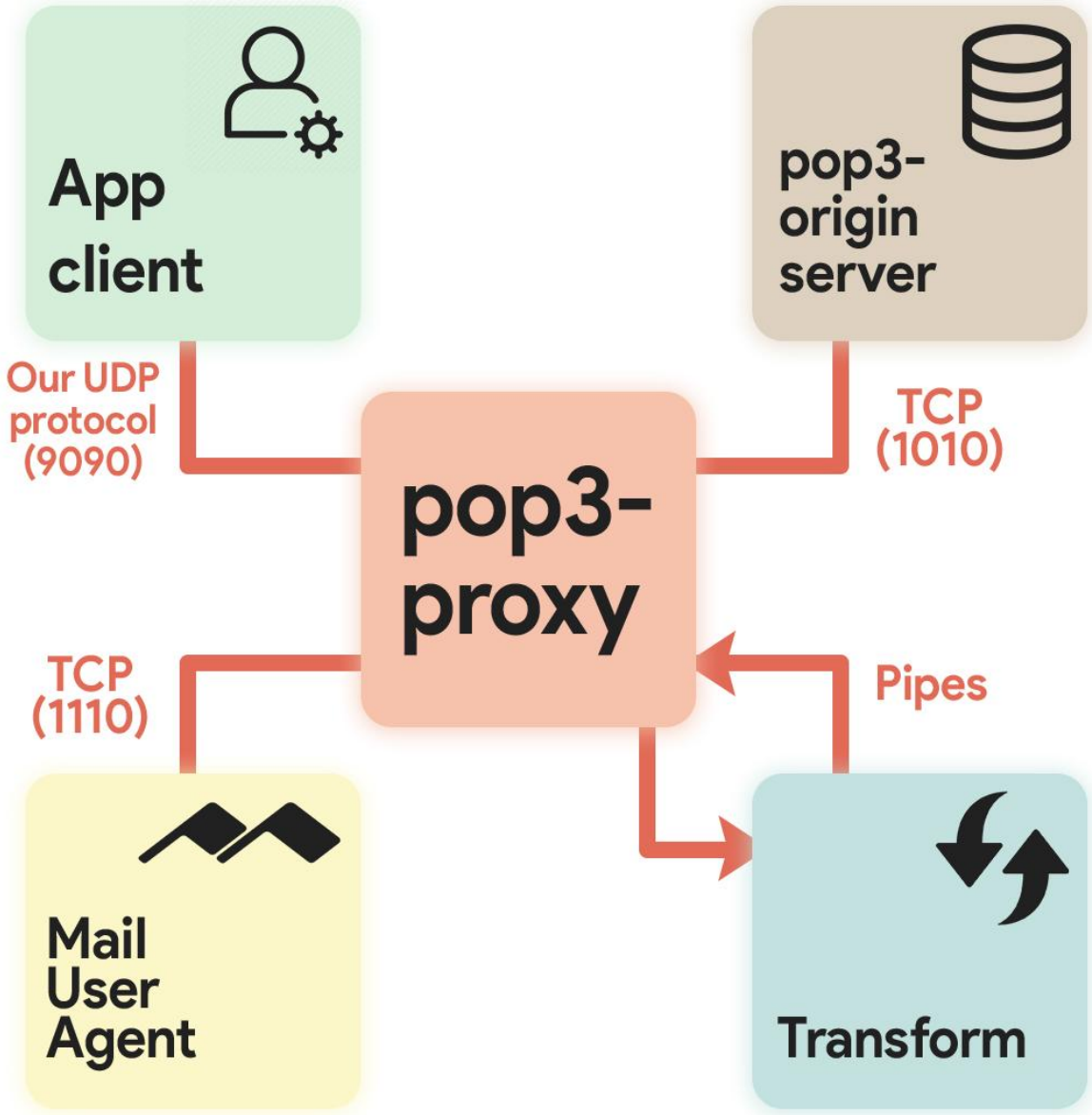

Imagen 4: Documento a modo de diseño del flujo del proyecto.