

Informe

Protocolos de Comunicación

Trabajo Especial

Grupo 1

Integrantes

- **Lucas Casagrande 55302**
- **Nicolás Castaño 53384**
- **Magdalena Vega 55206**

Índice

ÍNDICE	2
DESCRIPCIÓN DETALLADA DE LOS PROTOCOLOS Y APLICACIONES DESARROLLADAS	3
Servidor	3
Conexión Inicial	3
Funcionalidades del proxy	3
Silenciado	3
Multiplexado	4
Leet	4
Métricas	4
Funcionamiento Interno	4
Parseo de las stanzas	4
Protocolo:	4
PROBLEMAS ENCONTRADOS DURANTE EL DISEÑO Y LA IMPLEMENTACIÓN	9
LIMITACIONES DE LA APLICACIÓN	10
POSIBLES EXTENSIONES	10
CONCLUSIONES	10
EJEMPLOS DE PRUEBA	10
GUÍA DE INSTALACIÓN DETALLADA Y PRECISA.	12
INSTRUCCIONES PARA LA CONFIGURACIÓN	12
EJEMPLOS DE CONFIGURACIÓN Y MONITOREO	12
DOCUMENTO DE DISEÑO DEL PROYECTO	14

Descripción detallada de los protocolos y aplicaciones desarrolladas

Servidor

Conexión Inicial

Para la conexión inicial el Proxy tiene que comportarse como un servidor, debido a que es necesario primero conseguir el JID para saber si es necesario hacer la multiplexación.

El funcionamiento del Proxy en la conexión inicial es la siguiente:

- El Proxy escucha en el puerto 42069 (es necesario que el cliente se conecte a este puerto)
- Una vez que un cliente se conecta a este puerto se empieza a escuchar para que el cliente comience con la conexión
- Una vez que el usuario envió la stanza de stream se obtiene de esta el dominio al que se quiere conectar
- Se le pide al cliente que se autentique en formato plano
- Una vez que el cliente se autentica se obtiene el usuario de dicha autenticación y se determina si es necesario multiplexarlo, se conecta al servidor que corresponda de forma no bloqueante, haciendo uso de OP_CONNECT (se hablara más adelante de como configurarlo).
- Cuando la conexión con el servidor está finalizada se intenta conectar a dicho servidor con la cuenta del usuario o con la multiplexada si así lo requiriera
- Se le envía al cliente la respuesta de haberse conectado (success o failure(en caso de failure se cierra la conexión)) y el proxy comienza a funcionar como proxy.

Funcionalidades del proxy

El proxy tiene la posibilidad de manejar servidores que se encuentren en diferentes direcciones. Para configurar esto es necesario hacerlo desde el administrador utilizando el comando host. En caso de no tener el host al que se quiere conectar dentro de la lista, se intentara conectar al nombre del dominio.

Silenciado

Se determinó como silenciado de un usuario a la imposibilidad de recibir mensajes por parte del usuario como de mandarlos. Esto es, un usuario que esta silenciado no puede comunicarse de ninguna forma con otros usuarios. Para que esto funcione es necesario que el usuario se encuentre conectado al proxy, en caso de silenciar a un usuario no conectado no se realiza ninguna acción. Adicionalmente se decidió que cuando alguien se intenta comunicar con un usuario que se encuentra silenciado se simula que el usuario silenciado le envió un mensaje que dice “No estoy disponible”. Más allá de que XMPP tiene soporte para este tipo de error, pensamos que en caso de usuarios no expertos le resulta mejor un mensaje como el que enviamos que un error en la recepción del mensaje.

Multiplexado

Para el multiplexado se decidió que para poder realizar una multiplicación correcta el usuario y la contraseña tienen que ser los mismos en ambos servidores, por lo tanto, no sería posible, por ejemplo, multiplexar test@muffin.com como test2@muffin2.com.

Leet

Cuando un usuario es configurado para que este "l33teado" lo que genera es que todos los mensajes salientes de dicho usuario son convertidos a l33t. Para que esto se lleve a cabo es necesario que dicho usuario esté conectado al proxy

Métricas

Se tienen las siguientes métricas (que son volátiles):

- Cantidad de bytes leídos por el proxy
- Cantidad de accesos (la cantidad de clientes que se conectaron)
- La cantidad de mensajes bloqueados
- La cantidad de mensajes "l33teados"

Para poder acceder a dichos mensajes es necesario conectarse al servidor por el puerto del administrador.

Funcionamiento Interno

Nuestro Proxy es concurrente, para lograr esto se utiliza un selector que escucha en todas las conexiones disponibles, esto permite que los Read y Write no sean bloqueantes. Dentro de las Keys de cada conexión se guarda un objeto (ProxyConnection) que tiene la información de la comunicación (Nombre del usuario, Buffers, etc.). También el ProxyConnection es el que se encarga de hacer el Read y Write en los channels.

Parseo de las stanzas

El parseo de los mensajes se implementó completamente por nosotros. Una vez que se lee del buffer este es convertido a un CharBuffer, que luego es usado para hacer las validaciones correspondientes. Debido a que el parser está implementado por nosotros, solo realiza las operaciones necesarias, ya sea modificar el body de un mensaje o conseguir el "to" de una stanza, entre otras.

Una de las tareas de las que se encarga el parser es evaluar si la stanza esta bien formada. Existen 3 posibilidades, que este bien formada, que este incompleta o que tenga error. En caso de que la stanza esta incompleta no es enviada al destino correspondiente, sino que se la guarda y se queda esperando que se envíe el resto. En el caso en el que la stanza está bien formada o que tenga error esta es pasada enviada a donde corresponda. La única excepción a esto es cuando se recibe una stanza incompleta, pero el tamaño de dicha stanza es igual al tamaño del buffer, por lo cual no se podría almacenar más información en el buffer. En este caso lo que se hace es enviar la stanza aunque este incompleta (después de evaluarla para ver si es necesario hacerle alguna modificación), es por esto que en mensajes extremadamente largos (más grandes que 64KB (el tamaño de los buffers)) el funcionamiento del proxy no se garantiza que sea correcto.

Protocolo:

Nuestro Protocolo es de texto plano. El ABFN es el siguiente:

S	= (Query / Response) ENTER DOT ENTER
Query	= C SP String SP String / D SP String / E / "SEE" SP F
C	= "LOGIN" / "REGISTER" / "MULTIPLEX" / "HOST"
D	= "UNMULTIPLEX" / "SILENCE" / "UNSILENCE" / "L33T" / "UNL33T"
E	= "ACCESS" / "BYTES" / "EXIT" / "BLOCKED" / "CONVERTED"
F	= "MULTIPLEX" / "L33T" / "SILENCE" / "HOST"
String	= *VCHAR
Response	= "RESULT" SP Status SP Message
Message	= String / List
List	= ENTER String /
Status	= "OK" / "ERROR"
ENTER	= %x0A
DOT	= %0x2E

Como se puede ver en el ABFN el protocolo soporta múltiples líneas, y utiliza la cadena ENTER DOT ENTER para determinar el final del mensaje. El Protocolo utiliza encoding UTF-8 para permitir enviar usuarios de XMPP que se encuentran encodados de esta forma. El protocolo acepta los comandos tanto en mayúscula como minúscula. Para poder comunicarse con el Proxy mediante este protocolo solo es necesario establecer una conexión con el puerto específico para eso(42070).

- Si se recibe un comando que no es conocido por el protocolo devuelve RESULT ERROR Command does not exist
- Si se encuentra con un error que no puede resolver devuelve RETURN ERROR Unknown error
- Todos los mensajes del protocolo terminan en \n.\n
- En cualquier caso, si el mensaje está mal formado devuelve RETURN ERROR Invalid parameters
- En cualquier caso, menos login, si el usuario no está ingresado devuelve RETURN ERROR Not logged in

A continuación, se detallan los comandos soportados por el protocolo y que el usuario puede utilizar por consola. Además del formato del comando, se especifican parámetros y mensajes de validez o error.

LOGIN

-El comando es de la forma LOGIN user password

-Recibe dos parámetros separados con un espacio, el primero siendo el nombre de usuario y el segundo es la contraseña para poder ingresar como usuario.

-Si la contraseña o el usuario no son correctos, devuelve RESULT ERROR Username or password is not correct

-Si el usuario existe con esa contraseña devuelve RESULT OK Logged in

-Si el usuario ya está loggeado devuelve RESULT ERROR Already logged in

-Si no se mandan parámetros o se manda uno solo devuelve RESULT ERROR Invalid parameters

REGISTER

-El comando es de la forma: REGISTER user password

-Recibe dos parámetros separados con un espacio, el primero siendo el nombre de usuario y el Segundo es la contraseña para poder registrar a otro usuario

-Si no hubo problemas devuelve RESULT OK registered

-Si ya existe un usuario registrado con ese nombre devuelve RESULT ERROR already registered

-Si no se mandan parámetros o se manda solo uno devuelve RESULT ERROR Invalid parameters

MULTIPLEX

-El comando es de la forma MULTIPLEX user server

-Recibe dos parámetros separados con un espacio, el primero siendo el usuario, y el segundo al servidor origen diferente.

-Si el usuario no está multiplexado y se recibe el servidor, devuelve RETURN OK user multiplexed to server

-Si ya existe una multiplexación para ese usuario, se pisa el otro valor y devuelve RETURN OK user has been changed to multiplex to server

-Si no se mandan parámetros o se manda uno solo devuelve RESULT ERROR Invalid parameters

UNMULTIPLEX

-El comando es de la forma UNMULTIPLEX user

-Recibe un parámetro que es el usuario a desmultiplexar

- Si no está ingresado devuelve RESULT ERROR Not logged in \n.\n
- Si el usuario no esta multiplexado devuelve RESULT ERROR Cannot unmultiplex someone who is not multiplexed
- Si lo pudo desmultiplexar devuelve RETURN OK user is multiplexed
- Si no se mandan parámetros devuelve RETURN ERROR Invalid parameters

SILENCE

- El comando es de la forma SILENCE user
- Recibe un parámetro que es el usuario a silenciar
- Si no se especifican parámetros, devuelve RETURN ERROR Invalid parameters
- Si el usuario ya está silenciado devuelve RETURN ERROR user already silenced
- Si se pudo silenciar al usuario devuelve RETURN OK user silenced

UNSILENCE

- El comando es de la forma UNSILENCE user
- Recibe un parámetro que es el usuario a des silenciar
- Si no se mandan parámetros devuelve RETURN ERROR Invalid parameters
- Si el usuario no está silenciado devuelve RETURN ERROR user is not silenced
- Si se pudo silenciar al usuario devuelve RETURN OK user unsilenced

L33T

- El comando es de la forma L33T user
- Recibe un parámetro que es el usuario a aplicar la transformación
- Si no se mandan parámetros devuelve RETURN ERROR Invalid parameters
- Si el usuario ya tiene aplicada la transformación devuelve RETURN ERROR user already l33t
- Si pudo aplicar la transformación, devuelve RETURN OK user is l33t

UNL33T

- El comando de la forma UNL33T user
- Recibe un parámetro que es el usuario a revertir la transformación de los mensajes
- Si no se mandan parámetros devuelve RETURN ERROR Invalid parameters
- Si el usuario no tiene aplicada la transformación devuelve RETURN ERROR Cannot unl33t someone who is not l33t
- Si pudo aplicar la transformación devuelve RETURN OK user is unl33t

HOST

- El comando es de la forma HOST server host
- Recibe dos parámetros que el primero es el servidor y el segundo es a que host debería ir
- Si se envía un parámetro o no se mandan parámetros devuelve RETURN ERROR Invalid parameters
- Si el server no tiene devuelve RETURN OK Host server has been added
- Si ya existe una host para ese server, se pisa el otro valor y devuelve RETURN OK Host server has been changed

ACCESS

- El comando es de la forma ACCESS
- Si recibe algún parámetro devuelve RETURN ERROR Invalid parameters
- En caso de que el parámetro ingresado sea el correcto, devuelve RETURN OK número, donde el número es la cantidad de accesos que hubo

BYTES

- El comando es de la forma BYTES
- Si recibe algún parámetro devuelve RETURN ERROR Invalid parameters
- En caso de no recibir parametros, devuelve RETURN OK número, donde el número es el número de bytes que se transfirieron

BLOCKED

- El comando es de la forma BLOCKED

-Si recibe algún parámetro devuelve RETURN ERROR Invalid parameters

-En caso de no recibir parámetros, devuelve RETURN OK número, donde el número es el número de mensajes que fueron bloqueados

EXIT

-El comando es de la forma EXIT

-Si recibe algún parámetro devuelve RETURN ERROR Invalid parameters

-Si no hay usuario registrado devuelve RETURN ERROR Not logged in

-Si hay usuario registrado devuelve RETURN OK Logged Out

CONVERTED

-Es de la forma CONVERTED

-Si recibe algún parámetro devuelve RETURN ERROR Invalid parameters

-En caso de no recibir parámetros, devuelve RETURN OK número, donde el número es el número de mensajes que fueron convertidos

SEE

-El comando es de la forma SEE parámetro

-Si no se reciben parámetros o el parámetro no es igual a MULTIPLEX, SILENCE, L33T o HOST devuelve RETURN ERROR Invalid Parameters

-Si el parámetro es igual a MULTIPLEX o HOST devuelve RETURN OK\n y después imprime una línea por cada usuario multiplexado o que tiene host, donde la línea es de la form: param1 param2\n

-Si el parámetro es igual a SILENCE o L33T devuelve RETURN OK\n y después imprime una línea por cada usuario silenciado o transformado donde la línea es de la forma: user\n

Problemas encontrados durante el diseño y la implementación

-Hay algunos emoticones que cuando se genera la conversión a l33t dejan de verse de forma correcta, debido a que algunos caracteres que lo componen se ven modificados.

-Si se manda un mensaje con estilo no se lo convierte a l33t.

-Si el mensaje que recibe es mas grande que el tamaño del buffer no se garantiza un correcto funcionamiento del proxy.

-Se encontró un problema al hacer el logger donde el logger tira un warning al comienzo de la ejecución.

Limitaciones de la aplicación

-Emojis

Como se mencionó anteriormente, ciertos emojis no pueden mostrarse con claridad debido a la conversión de mensaje que realiza el proxy. Esto sucede cuando el usuario a quien se le está mandando el emoji tiene la opción de poder transformar los mensajes.

-Se parsea cierta parte de los mensajes

Para mensajes que son muy largos, lo que ocurre es que se muestra el mensaje completo, pero no convertido en su totalidad. Es decir, el mensaje tiene sus caracteres convertidos en el formato l33t hasta el tamaño a saber. Una vez superado este tamaño, el mensaje no se encuentra convertido.

-La configuración de los puertos es única y no se puede cambiarse escucha a los clientes en un puerto específico y, el monitoreo y administración del mismo es única también.

Posibles extensiones

-Poder agregar otro tipo de transformaciones para los mensajes

-Poder bloquear a un usuario. Es decir, un usuario bloquea a otro, en donde el usuario bloqueado no le puede mandar mensajes al que lo bloquea, pero sí de la otra forma

-Agregar más métricas para poder observar otros estados de la comunicación

-Emojis sin problema

-Silenciar un grupo

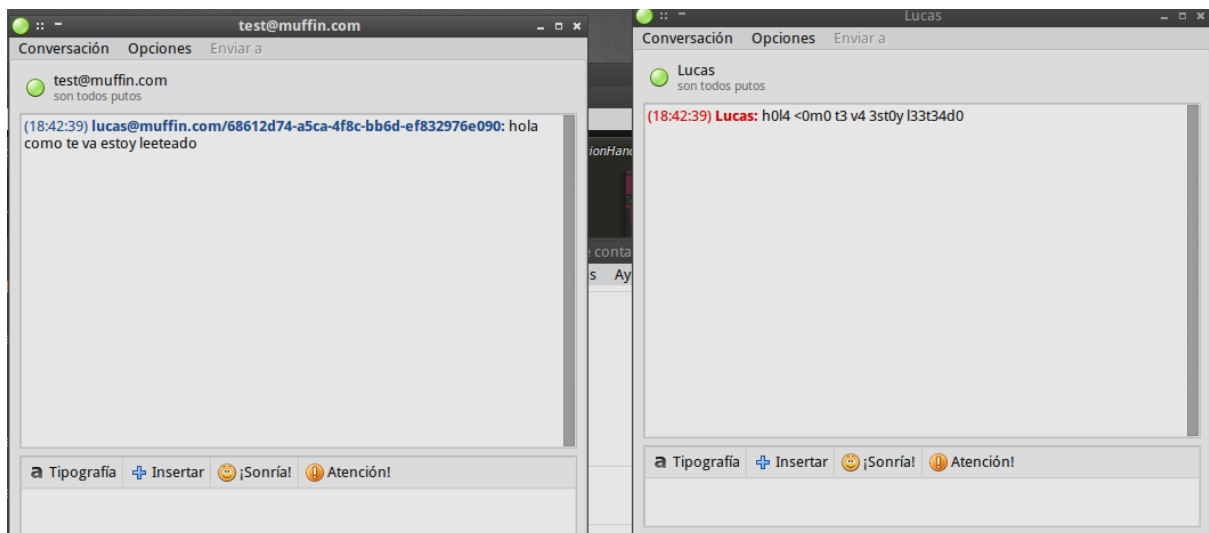
-Que se pueda convertir el mensaje completo

Conclusiones

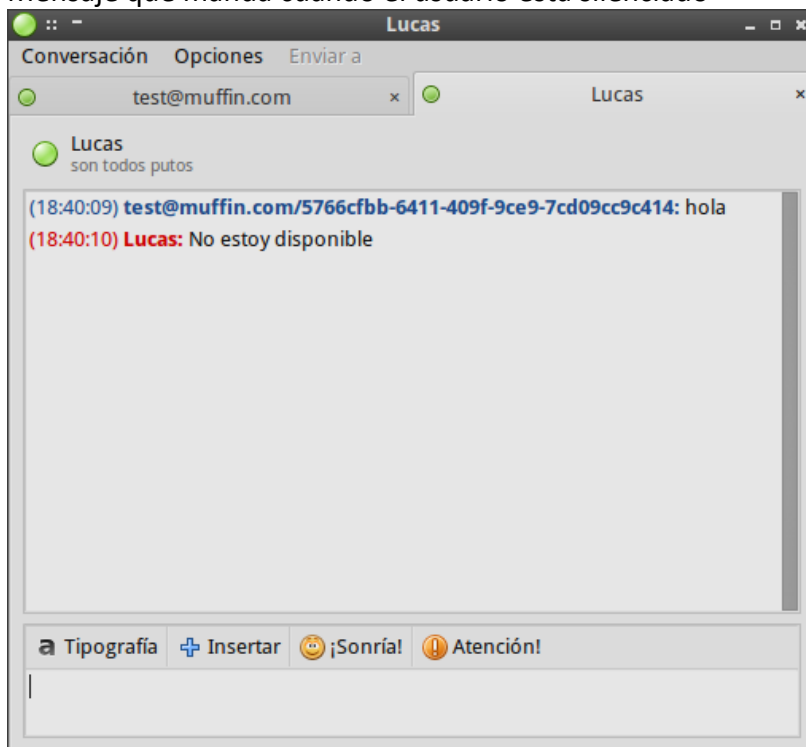
La realización de este trabajo práctico permitió a los integrantes del grupo interiorizarse sobre el protocolo XMPP, así como también entender como configurar un servidor que haga uso de este protocolo, y posteriormente utilizar clientes de chat basados en dicho protocolo. También es importante decir que este trabajo práctico ayudó para poder entender como pensar un protocolo creado por nosotros. Y poder entender como documentarlo bien y pensar como poder mostrar todos los comandos que nosotros queríamos mostrar y los errores que estos traen.

Ejemplos de prueba

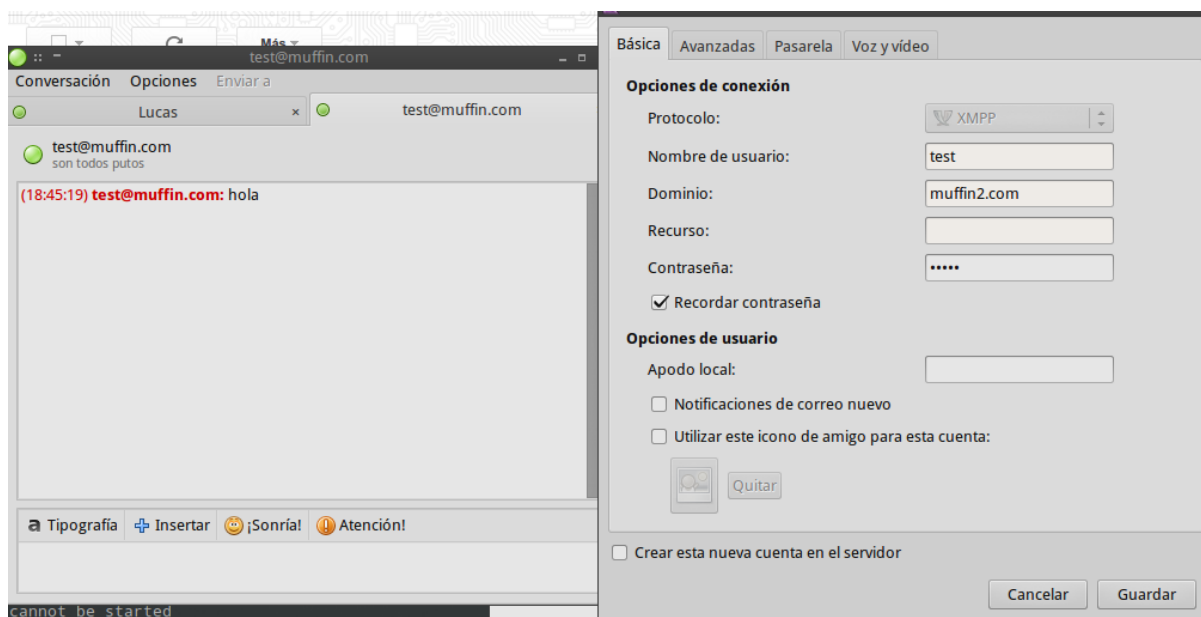
L33T



Mensaje que manda cuando el usuario esta silenciado



Si el usuario esta multiplexado



Guía de instalación detallada y precisa.

Para instalarlo es necesario seguir los siguientes pasos:

- Descargar el código fuente del repositorio de BitBucket
- Ejecutar en la consola `mvn package` en el directorio raíz del proyecto
- Ir a la carpeta `target`
- Ejecutar `java -jar XMPPProxy-1.0-SNAPSHOT.jar`

Instrucciones para la configuración

El servidor escucha a los clientes en el puerto 42069, y utiliza el puerto 42070 para la administración y monitoreo del mismo. Por lo tanto, para configurar el proxy solo es necesario conectarse al puerto 42070 (ya sea mediante netcat o alguna otra aplicación)

Ejemplos de configuración y monitoreo

En la siguiente imagen se muestra varios datos que se le pueden pedir al servidor para monitorearlo, como también la respuesta de este.

```

RESULT ERROR Already logged in
.
bytes
.
RESULT OK 36404050
.
access
.
RESULT OK 25154
.
see multiplex
.
RESULT OK
test@muffin2.com test@muffin.com
nicolas@example.com nicolas@muffin.com
.
see l33t
.
RESULT OK
nicolas@muffin.com
lucas@muffin.com
nico3@example.com
nico9@example.com
nico4@example.com
nico8@example.com
ncastano@example.com
nico5@example.com
nico7@example.com
nico@example.com
nico2@example.com
nico6@example.com
.

```

También hay comandos para configurar el proxy

```

lucas@Muffin: ~
lucas@Muffin:~$ nc localhost 42070
localhost [127.0.0.1] 42070 (?) : Connection refused
lucas@Muffin:~$ nc localhost 42070
login muffin muffin
.
RESULT OK Logged in
.
multiplex lucas@muffin.com lucas@muffin2.com
.
RESULT OK lucas@muffin.com multiplexed to lucas@muffin2.com
.
l33t lucas@muffin.com
.
RESULT ERROR lucas@muffin.com already l33t
.
unl33t lucas@muffin.com
.
RESULT OK lucas@muffin.com is unl33t
.
silence lucas@muffin.com
.
RESULT OK lucas@muffin.com silenced
.

```

Documento de diseño del proyecto

