Informe

Protocolos de Comunicación

Trabajo Especial

**Integrantes**

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

Índice

[Índice 2](#_Toc466969997)

[Descripción detallada de los protocolos y aplicaciones desarrolladas 3](#_Toc466969998)

[Problemas encontrados durante el diseño y la implementación 3](#_Toc466969999)

[Limitaciones de la aplicación 4](#_Toc466970000)

[Posibles extensiones 4](#_Toc466970001)

[Conclusiones 4](#_Toc466970002)

[Ejemplos de prueba 4](#_Toc466970003)

[Guía de instalación detallada y precisa. No es necesario desarrollar un programa instalador. 4](#_Toc466970004)

[Instrucciones para la configuración. 4](#_Toc466970005)

[Ejemplos de configuración y monitoreo. 4](#_Toc466970006)

[Documento de diseño del proyecto (que ayuden a entender la arquitectura de la aplicación) 4](#_Toc466970007)

# Descripción detallada de los protocolos y aplicaciones desarrolladas

Servidor:

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 proxy conection es el que se encarga de hacer el Read y Write en los channels.

El parseo de los mensajes se implemento 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 esta 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.

Protocolo:

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

S = (Query / Response) ENTER . 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"

Como se puede ver en el ABFN el protocolo soporta múltiples líneas, y utiliza la cadena ENTER . ENTER para determinar el final del mensaje. El Protocolo utiliza encoding UTF-8 para permitir enviar usuarios de XMPP que se encuentran encodeados 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 especifico para eso(42070). Es necesario hacer un login para poder empezar a interactuar.

# 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.

# Limitaciones de la aplicación

# Posibles extensiones

# Conclusiones

# Ejemplos de prueba

# Guía de instalación detallada y precisa. No es necesario desarrollar un programa instalador

Para instalarlo en necesario seguir los siguientes pasos:

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

# Instrucciones para la configuración

El servidor esucha a los clientes en el puerto 42069, y utliza el puerto 42070 para la administacion y monitoreo del mismo.

# Ejemplos de configuración y monitoreo

# Documento de diseño del proyecto (que ayuden a entender la arquitectura de la aplicación)