**--------------APLICACIONES EN COMUNICACIONES EN RED------------**

**PRÁCTICA 3:**

Chat Multicast

**Alumno:**

Meza Vargas Brandon David

**Grupo:**

3CM16

**Profesor:**

Moreno Cervantes Axel Ernesto

**Índice**

[**Introducción** 4](#_Toc101357174)

[**Desarrollo** 5](#_Toc101357175)

[**Server** 5](#_Toc101357176)

[**SendPacket** 6](#_Toc101357177)

[**serverConnect** 6](#_Toc101357178)

[**Client** 7](#_Toc101357179)

[**Chat** 9](#_Toc101357180)

[**initComponents** 10](#_Toc101357181)

[**setAudio** 11](#_Toc101357182)

[**setPlayAudio** 13](#_Toc101357183)

[**showAvailableAudios** 14](#_Toc101357184)

[**initEmojis** 15](#_Toc101357185)

[**userLeft** 15](#_Toc101357186)

[**setCurrentUsers** 16](#_Toc101357187)

[**addChat** 17](#_Toc101357188)

[**getCurrentMsg** 18](#_Toc101357189)

[**newMessage** 18](#_Toc101357190)

[**Constants** 20](#_Toc101357191)

[**Connection** 22](#_Toc101357192)

[**selectNetworkInterface** 22](#_Toc101357193)

[**despliegaInfoNIC** 24](#_Toc101357194)

[**Pruebas de funcionamiento** 25](#_Toc101357195)

[**Conclusiones** 31](#_Toc101357196)

[**Bibliografía** 32](#_Toc101357197)

**Índice de ilustraciones**

[Ilustración 1. Server hilo. 6](#_Toc101357198)

[Ilustración 2. Método sendPacket. 6](#_Toc101357199)

[Ilustración 3. Método serverConnect. 7](#_Toc101357200)

[Ilustración 4. Hilo cliente. 9](#_Toc101357201)

[Ilustración 5. Constructor del chat 10](#_Toc101357202)

[Ilustración 6. Método initComponents. 11](#_Toc101357203)

[Ilustración 7. Método setAudio 13](#_Toc101357204)

[Ilustración 8. Método setPlayAudio 14](#_Toc101357205)

[Ilustración 9. Método showAvalableAudios 14](#_Toc101357206)

[Ilustración 10. Método initEmojis 15](#_Toc101357207)

[Ilustración 11. Método userLeft 16](#_Toc101357208)

[Ilustración 12. Método setCurrentUsers 17](#_Toc101357209)

[Ilustración 13. Método addChat 18](#_Toc101357210)

[Ilustración 14. Método getCurrentMsg 18](#_Toc101357211)

[Ilustración 15. Método newMessage 20](#_Toc101357212)

[Ilustración 16. Clase constants 22](#_Toc101357213)

[Ilustración 17. Método selectNetworkInterface 24](#_Toc101357214)

[Ilustración 18. Método despliegaInfoNIC 24](#_Toc101357215)

[Ilustración 19. Ejecución del servidor. 25](#_Toc101357216)

[Ilustración 20. Solicitud de nombre al usuario 25](#_Toc101357217)

[Ilustración 21. Solicitud de selección de interfaz de red. 26](#_Toc101357218)

[Ilustración 22. Interfaz del chat. 26](#_Toc101357219)

[Ilustración 23. Ventana de los clientes. 27](#_Toc101357220)

[Ilustración 24. Mensaje general 27](#_Toc101357221)

[Ilustración 25. Mensajes de usuarios. 28](#_Toc101357222)

[Ilustración 26. Envió de audio. 28](#_Toc101357223)

[Ilustración 27. Reproduciendo audio, 29](#_Toc101357224)

[Ilustración 28. Mensaje privado de brandon a liz 29](#_Toc101357225)

[Ilustración 29. Mensaje de liz a brandon 30](#_Toc101357226)

[Ilustración 30. Mandando emojis. 30](#_Toc101357227)

[Ilustración 31. Emojis en el chat. 30](#_Toc101357228)

# **Introducción**

Muchas empresas hacen uso de internet para ofrecer sus productos y servicios, ya que a través de este medio pueden estar en contacto directo con los clientes potenciales. Algunas de las principales ventajas de Internet como medio para realizar negocios son la cobertura a nivel global, así como la disponibilidad del servicio los 365 días, las 24 horas. del día. Sin embargo, estas ventajas se convierten también en un reto para las empresas, pues se debe proporcionar a los clientes una vía de comunicación directa para resolver las dudas de los clientes potenciales, atender sus peticiones, así como proporcionarles información adicional a la expuesta en sus portales. Existen diversos recursos que pueden ser utilizados, como los foros o el correo electrónico, pero ninguno de ellos permite la comunicación en tiempo real. El chat es un excelente recurso para este tipo de propósito, ya que además de ser una vía de comunicación en tiempo real, permite comunicar a dos o más usuarios entre sí. Típicamente los chats se han implementado haciendo uso de sockets de flujo, pero existen otras alternativas que también merece la pena probar, tales como los sockets de datagrama o los de multidifusión

Hoy en día muchos tipos de aplicaciones se cuenta con un servicio de chat, ya sea con fines de diversión o de negocio. Este tipo de servicio es muy socorrido cuando se trata de brindar una comunicación más personalizada que la brindada por medios tales como correo electrónico o foros. En el chat la comunicación puede ser fluida, es decir, en tiempo real y además se pueden interactuar dos o más personas a la vez.

Por lo anterior, en el presente documento se presenta la práctica realizada la cual consta de un chat que permite mandar mensajes a un chat general y mensajes privados, así como emojis y audios.

# **Desarrollo**

A continuación se muestra el desarrollo de la práctica con las capturas de todo el código utilizado

# **Server**

Primeramente tenemos el servidor el cual se encargará de recibir mensajes del cliente y a su vez enviar mensajes indicando que es una repuesta del servidor

El hilo se encarga de determinar el tipo de mensaje del que se trata y realizar una acción en base a eso, si el mensaje indica un inicio de chat se le mandará a los clientes la lista de usuarios conectados, si el mensaje se trata de un mensaje tal cuál este se mandará al cliente y si se indica un fin de sesión se removerá ese cliente de los chats y se enviará a los clientes conectados la nueva lista de usuarios en el chat.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ilustración 1. Server hilo.

## **SendPacket**

Este método se encarga de enviar el paquete al cliente como se ha realizado en prácticas anteriores, a continuación se presenta el código correspondiente.

Texto

Descripción generada automáticamente

Ilustración 2. Método sendPacket.

## **serverConnect**

Este método se encarga de mostrar en el servidor las interfaces de red disponibles para que se seleccione una y de esta manera unir al servidor al grupo multicast, como el servidor está en un ciclo infinito se le verifica si el servidor ha sido invocado solo una vez, si no es así se selecciona la misma red elegida la primera vez de manera automática.

Texto

Descripción generada automáticamente

Ilustración 3. Método serverConnect.

# **Client**

En el cliente tenemos un hilo que se encarga de gestionar las conexiones y mensajes de los clientes en el chat, lo que se hace primero es unir al cliente al mismo grupo multicast al que está unido el servidor.

Al inicio se le manda al servidor un mensaje de inicio de chat, pues el usuario al ejecutar el programa y al unirse adecuadamente a un grupo da inicio al chat.

Posteriormente en un ciclo infinito se pregunta por la operación que se está realizando en el cliente, si es escritura quiere decir que tenemos que escribir un mensaje, para esto obtenemos el mensaje dependiendo si es de fin de chat, mensaje general o mensaje privado entre usuarios

Si la operación que se está realizando es de lectura entonces tenemos que setear un nuevo mensaje el cual es el que se enviará.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ilustración 4. Hilo cliente.

# **Chat**

En esta parte tenemos todo lo necesario para desplegar la interfaz gráfica del usuario

Texto

Descripción generada automáticamente

Ilustración 5. Constructor del chat

## **initComponents**

Aquí empezamos por inicializar los componentes participantes en la interfaz del usuario, entre ellos la ventana principal y el botón de enviar, así como la llamada a otros métodos que inicializan otras cosas.

Texto

Descripción generada automáticamente

Ilustración 6. Método initComponents.

## **setAudio**

Aquí se crean los componentes para enviar el audio, se crea el botón correspondiente y su event listener que se encargará de enviar el audio primeramente se encarga de establecer la carpeta donde se guardará el audio y posteriormente se crea una instancia de la clase que se encarga de grabar el audio, se grabará por 5 segundos

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ilustración 7. Método setAudio

## **setPlayAudio**

Este método crea el botón para reproducir un audio, posteriormente se manda a llamar la clase que muestra los audios disponibles.

Texto

Descripción generada automáticamente

Ilustración 8. Método setPlayAudio

## **showAvailableAudios**

Este método se encarga de ir a la carpeta correspondiente para mostrarle al usuario los audios que están disponibles para que el pueda seleccionar uno y a partir de eso se crea una instancia de la clase que se encarga de reproducir el audio seleccionado.

Texto

Descripción generada automáticamente

Ilustración 9. Método showAvalableAudios

## **initEmojis**

Este método se encarga de poner los emojis disponibles en la interfaz de usuario. Cada emoji corresponde a un botón que cuando el usuario lo pulsa se le adjunta a su mensaje de texto.

Texto

Descripción generada automáticamente

Ilustración 10. Método initEmojis

## **userLeft**

Este método indica cuando un usuario se desconecta del chat, este método agrega un escuchador a la ventana al momento de que se pulsa el botón de cerrar

Cuando pasa eso se cambió el estado de conectado a false y se cambia la operación a escritura para indicar que salió del chat.

Texto

Descripción generada automáticamente

Ilustración 11. Método userLeft

## **setCurrentUsers**

Este método recorre el arreglo que contiene los usuarios conectados en tiempo real, poniéndolos todos en un botón para que el usuario al hacer clic en unide ellos puedan iniciar un chat privado. A cada botón se le asigna un escuchador que agregará al usuario clicando en una nueva pestaña del tabbed Pane

Al final se tiene que repintar la ventana para ver cuando hay actualizaciones en los usuarios conectados.

Texto

Descripción generada automáticamente

Ilustración 12. Método setCurrentUsers

## **addChat**

Aquí simplemente se agrega a las pestañas una nueva en caso de que sea un nuevo chat se agrega una nueva ventana con un título que será igual al nombre del usuario con el que se inició un chat.

Texto

Descripción generada automáticamente

Ilustración 13. Método addChat

## **getCurrentMsg**

Este método solo se encarga de devolver el mensaje actual.

Texto

Descripción generada automáticamente

Ilustración 14. Método getCurrentMsg

## **newMessage**

Este método es el más importante de la clase pues se encarga de establecer de que tipo de mensaje se trata y crear el mensaje.

En primer lugar tenemos que si es un mensaje que indica un inicio de sesión se setea la lista de usuarios conectados.

Si es un mensaje se corta la parte que indica que es un mensaje y se verifica si es un mensaje privado, de ser así se corta el mensaje para determinar el receptor y emisor. Si el usuario conectado es igual al que recibe el mensaje y el que mando el mensaje ya está en las pestañas del chat solo se cambia la pestaña actual, en caso contrario se crea la pestaña con el método antes mencionado. En caso de que el usuario conectado sea igual al que manda el mensaje se obtiene la pestaña y si no corresponde a un chat privado se obtiene la pestaña 0 que corresponde al chat general.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ilustración 15. Método newMessage

# **Constants**

En esta clase solo se guardan los mensajes y algunas configuraciones usadas en todo el programa como constantes.

Texto

Descripción generada automáticamente

Pantalla de computadora con fondo negro

Descripción generada automáticamente con confianza media

Ilustración 16. Clase constants

# **Connection**

En esta clase encontramos los métodos referentes a conexiones usadas por el servidor y cliente

## **selectNetworkInterface**

Este método selecciona las interfaces de red disponibles en el equipo.

Si el usuario es el servidor se le muestra la información de manera textual, en caso de que sea el cliente como se está ejecutando con una interfaz también se le da a escoger la interfaz de red de manera gráfica.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ilustración 17. Método selectNetworkInterface

## **despliegaInfoNIC**

Este método solo muestra la información de la NIC como su nombre, dirección y algo muy importante, si soporta multicast o no.

Texto

Descripción generada automáticamente

Ilustración 18. Método despliegaInfoNIC

## **Pruebas de funcionamiento**

Ahora pasaremos con las pruebas de la aplicación.

Primeramente debemos correr el servidor y escoger la interfaz multicast deseada para comenzar a aceptar conexiones de clientes como se muestra en la siguiente ilustración.

Tabla

Descripción generada automáticamente con confianza media

Ilustración . Ejecución del servidor.

Posteriormente ya podemos conectarnos como clientes, al ejecutar un cliente se le pedirá que ingrese su nombre y seleccione su interfaz de red pero ahora de una manera gráfica como se muestra a continuación.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración . Solicitud de nombre al usuario

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Solicitud de selección de interfaz de red.

Una vez ingresado su nombre y seleccionado su interfaz de red se abre el chat.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración . Interfaz del chat.

Se procederá a conectar a dos usuarios más.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Ventana de los clientes.

Podemos ver que a cada cliente conectado se le muestran los usuarios conectados al chat en la parte de la derecha, si manda un mensaje en la sala general se manda el mensaje a todos como se muestra a continuación.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración . Mensaje general

Y así se pueden hacer la conversación en la sala general con los miembros activos del chat.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración . Mensajes de usuarios.

Para mandar un audio basta con pulsar el botón de enviar audio, se grabará un audio de 5 segundos y se indicará en el chat que un usuario envió un audio.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Envió de audio.

Para reproducir el audio se deberá pulsar en el botón que dice reproducir audio y se mostrará una lista de audios disponibles para reproducir.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración . Reproduciendo audio,

Para entablar una conversación privada se deberá pulsar en uno de los usuarios conectados mostrados en la parte de la derecha y se abrirá una nueva pestaña.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración . Mensaje privado de brandon a liz

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración . Mensaje de liz a brandon

En la parte inferior derecha se pueden encontrar emojis que los usuarios pueden mandar.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración . Mandando emojis.

Los caracteres en el chat se reemplazarán por los emojis una vez sean mandados como se ve a continuación.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración . Emojis en el chat.

# **Conclusiones**

Para concluir, esta práctica me pareció una muy buena práctica para aplicar lo visto en clases sobre multicast, pues a partir de estos se logró realizar un chat con múltiples usuarios, cosa que se sigue viendo hoy en día con distintas aplicaciones entre las más famosas Facebook.

Durante el desarrollo de la práctica se me presentaron algunas complicaciones, entre las más destacables fue el hacer que se identifique el mensaje general o el mensaje privado, pero esto se resolvió gracias a la estructura de mensajes que nos recomendó el profesor, por medio de etiquetas se indicó si era un mensaje privado, el remitente y el receptor y cortando el mensaje y guardando esos cortes en nuevas variables se identificó si el mensaje era privado o general.

Sin duda una práctica retadora pero que al final se logró resolver de buena forma, una de las ventajas principales al usar sockets de datagrama o de multidifusión es que la información llegad e manera rápida y segura.

# **Bibliografía**

1. programmer (2020). “Sockets de datagrama”. Recuperado de : https://programmerclick.com/article/638561752/