Universidad del Valle de Guatemala
Sistemas Operativos
Docente: Sebastián Galindo

Proyecto 1 - Chat

Gustavo Adolfo Cruz Bardales - 22779
Javier Andres Chen - 22153
Miércoles, 2 de abril de 2025

Introducción

El presente informe detalla el desarrollo e implementación de un sistema de chat en tiempo real utilizando WebSockets como tecnología de comunicación principal. El proyecto fue desarrollado como parte del curso de Sistemas Operativos, con el objetivo de aplicar conceptos fundamentales de programación concurrente, protocolos de comunicación y desarrollo de aplicaciones cliente-servidor.

El sistema desarrollado permite la comunicación entre múltiples usuarios a través de una interfaz gráfica y una línea de comandos opcional, ofreciendo funcionalidades como el envío de mensajes públicos y privados, gestión de estados de usuario, visualización del historial de conversaciones y lista de usuarios conectados.

Para la implementación del proyecto se utilizaron diversas tecnologías y bibliotecas, entre las que destacan libwebsockets para la comunicación mediante WebSockets, pthreads para la implementación de concurrencia, y WXWidgets para el desarrollo de la interfaz gráfica del cliente.

\

Objetivos

Objetivo General

Desarrollar un sistema de chat cliente-servidor que permita la comunicación en tiempo real entre múltiples usuarios mediante WebSockets, implementando conceptos de programación concurrente y gestión de recursos compartidos.

Objetivos Específicos

- Diseñar e implementar un protocolo de comunicación mediante WebSockets que garantice la interoperabilidad entre clientes y servidores desarrollados por diferentes equipos.
- Aplicar técnicas de programación concurrente para manejar múltiples conexiones simultáneas en el servidor.
- Implementar mecanismos de sincronización para evitar condiciones de carrera en recursos compartidos.
- Desarrollar un cliente funcional con capacidad de envío y recepción de mensajes en tiempo real.
- Desplegar el sistema en una infraestructura cloud (AWS EC2) para facilitar la comunicación remota entre usuarios.

Arquitectura del Sistema

La arquitectura del sistema implementa un modelo cliente-servidor tradicional, donde:

- **El Servidor**: Actúa como punto central para la gestión de conexiones, enrutamiento de mensajes y almacenamiento de estado. Está programado en C++ utilizando libwebsockets para el manejo de las conexiones WebSocket y pthreads para la concurrencia.
- **El Cliente**: Se conecta al servidor mediante WebSockets, permitiendo a los usuarios enviar y recibir mensajes, cambiar estados y visualizar información del sistema. Está desarrollado en C++ con una interfaz de línea de comandos e interfaz gráfica.

El sistema utiliza un modelo de comunicación basado en mensajes binarios estructurados según el protocolo definido por la clase. Cada mensaje contiene un tipo, que determina la acción a realizar, y datos asociados a dicha acción.

Protocolo de Comunicación

El protocolo de comunicación implementado utiliza mensajes binarios con una estructura definida. Cada mensaje comienza con un byte que indica el tipo de mensaje, seguido por datos específicos según el tipo.

Tipos de Mensajes

Tipo	Descripción	Estructura
10	Registro de usuario	[10][longitud_nombre][nombre][estado]
11	Solicitar lista de usuarios	[11]
12	Obtener información de usuario	[12][longitud_nombre][nombre]
13	Cambiar estado	[13][estado]
14	Enviar mensaje público	[14][longitud_mensaje][mensaje]
15	Enviar mensaje privado	[15][longitud_destino][destino][longitud_mensaje][mensaje]
16	Solicitar historial de mensajes	[16]
50	Respuesta de error	[50][código_error]

51	Respuesta lista de usuarios	[51][longitud_datos][contador_usuarios][[longitud_nombre][nombre][estado]...]
52	Respuesta información de usuario	[52][longitud_nombre][nombre][estado]
53	Notificación de usuario registrado	[53][longitud_nombre][nombre][estado]
54	Notificación de cambio de estado	[54][longitud_nombre][nombre][estado]
55	Mensaje recibido	[55][tipo_mensaje][longitud_emisor][emisor][longitud_mensaje][mensaje]
56	Respuesta historial de mensajes	[56][contador_mensajes][[longitud_emisor][emisor][longitud_mensaje][mensaje]...]

Estados de Usuario

Código	Estado
0	DESACTIVADO
1	ACTIVO
2	OCUPADO

3	INACTIVO

Códigos de Error

Código	Descripción
1	El usuario que intentas obtener no existe
2	El estatus enviado es inválido
3	¡El mensaje está vacío!
4	El mensaje fue enviado a un usuario desconectado

Universidad del Valle de Guatemala
Sistemas Operativos
Docente: Sebastián Galindo
Primer Semestre, 2025

Investigación de Tecnologías

Libwebsockets

Libwebsockets es una biblioteca de código abierto que proporciona una implementación del protocolo WebSocket (RFC 6455) para C/C++. Esta biblioteca permite la creación de aplicaciones cliente-servidor que requieren comunicación bidireccional en tiempo real.

Libwebsockets está diseñada para ser ligera y eficiente, con un uso mínimo de memoria y procesamiento. Además, cumple con los estándares RFC 6455 para WebSockets, asegurando interoperabilidad con otros sistemas, e incluye soporte para conexiones seguras mediante OpenSSL.

Utiliza un modelo de programación basado en eventos y callbacks, lo que facilita el manejo de múltiples conexiones. Permite definir distintos protocolos de aplicación sobre WebSockets.

Pthreads

POSIX Threads (Pthreads) es una API estándar para la creación y manipulación de hilos en sistemas operativos compatibles con POSIX. Esta biblioteca permite la implementación de programación concurrente, permitiendo que múltiples tareas se ejecuten de forma simultánea dentro del mismo proceso.

Implementa el estándar IEEE POSIX 1003.1c, asegurando portabilidad entre sistemas compatibles (Mac, Linux, BSD, Microsoft con WSL). Proporciona funciones para crear, sincronizar, y terminar hilos e incluye mutex, variables de condición, semáforos y barreras para coordinar la ejecución de hilos. Permite configurar atributos como la prioridad del hilo, la política de planificación y el tamaño de la pila.

Threads de C++

La biblioteca estándar de C++ (desde C++11) incluye soporte nativo para programación concurrente a través de la biblioteca <thread>, proporcionando una abstracción de alto nivel sobre los sistemas de hilos del sistema operativo subyacente.

Está diseñada para integrarse de forma natural con el resto de la biblioteca estándar de C++. Oculta los detalles específicos de la implementación de hilos del sistema operativo. Además que proporciona mutex, variables de condición, cerrojos únicos y compartidos, y locks con tiempo de espera.

WXWidgets

WXWidgets es una biblioteca de interfaz gráfica de usuario (GUI) multiplataforma, escrita en C++ pero con enlaces para varios otros lenguajes de programación. Proporciona una API única que funciona en múltiples plataformas, incluyendo Windows, macOS, Linux/Unix, y otras. En específico, se utilizó la implementación con gtk3.

Proporciona una API consistente que funciona en diferentes sistemas operativos. Utiliza los controles nativos del sistema operativo cuando es posible, lo que resulta en aplicaciones con apariencia nativa. Además que incluye una amplia gama de widgets y componentes para construir interfaces completas.

Paralelización y Concurrencia

La implementación de concurrencia y paralelización en el sistema de chat es fundamental para su correcto funcionamiento, permitiendo manejar múltiples conexiones y operaciones simultáneas. A continuación, se detallan los aspectos más relevantes:

Implementación en el Cliente

En el cliente, la concurrencia se implementa principalmente mediante un hilo dedicado para la escucha de mensajes entrantes, en el archivo Conexion.cpp. Además de implementar pthreads para el manejo de los mensajes, tomando en cuenta mensajes como se definió en el protocolo que incluyen las llamadas y manejos de errores dentro del mismo sistema.

Implementación en el Servidor

El servidor implementa un modelo más complejo de concurrencia pues este, maneja la inicialización del servidor y el bucle principal de eventos de libwebsockets; Supervisa la inactividad de los usuarios y actualiza estados automáticamente. Los callbacks de libwebsockets manejan eventos como conexiones, mensajes y desconexiones.

Referencias

1. Libwebsockets Documentation. Recuperado de https://libwebsockets.org/
2. wxWidgets Documentation. Recuperado de https://www.wxwidgets.org/docs/
3. POSIX Threads Programming. Recuperado de https://computing.llnl.gov/tutorials/pthreads/
4. AWS EC2 User Guide. Recuperado de https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
5. WebSocket Protocol - RFC 6455. Recuperado de https://tools.ietf.org/html/rfc6455