TEC | Tecnológico de Costa Rica

Lenguajes de Programación

Proyecto I

Aplicación de mensajería

Linux

Estudiante:

Franco Vinicio Rojas Lagos

Profesor:

Bryan Tomas Hernández Sibaja

Descripción general

Este proyecto implementa un sistema de mensajería entre múltiples clientes utilizando un servidor central desarrollado en C++. El sistema permite:

- Registro automático de usuarios al conectarse.
- Envío de mensajes entre clientes.
- Almacenamiento de mensajes pendientes para usuarios desconectados.
- Descubrimiento automático del servidor mediante UDP broadcast.
- Lectura dinámica de puertos desde archivo de configuración (config.txt).

Enlace de GitHub

https://github.com/frarojas/ForkChat.git

Librerías Utilizadas y su Propósito

- `iostream` → Entrada y salida estándar.
- `fstream` → Lectura de archivos (para leer `config.txt`).
- `cstring` → Manejo de cadenas tipo C (`strcmp`, `strcpy`, etc.).
- `sstream` → Manejo de flujos de texto (`std::istringstream`).
- `map` → Estructura clave-valor para almacenar usuarios y mensajes pendientes.
- `vector` → Lista dinámica para guardar mensajes pendientes por usuario.
- `unistd.h` → Llamadas al sistema Unix (`fork`, `close`, etc.).
- `arpa/inet.h` → Funciones para manipular direcciones IP (`inet_ntoa`, `inet_pton`, etc.).
- `netinet/in.h` → Estructuras y constantes para sockets (`sockaddr_in`, etc.).
- `sys/socket.h` → Funciones de sockets (`socket`, `bind`, `send`, `recv`, etc.).
- `sys/types.h` → Tipos del sistema requeridos para sockets.
- `sys/select.h` → Mecanismo para multiplexar conexiones (`select`).
- `cstdlib` → Utilidades generales como `exit()` o `atoi()`.

Pasos de instalación del programa

Archivo config.txt

Debe contener los siguientes valores:

TCP_PORT 5000
UDP_PORT 5001

Ambos, cliente y servidor, leen automáticamente estos valores al iniciar.

Compilación

Servidor

g++ -o servidor servidor.cpp

Cliente

g++ -o cliente cliente.cpp

Manual de Usuario

1. Preparar el entorno

- Asegúrese de tener instalado g++ en su sistema.
- Cree un archivo config.txt con los puertos que usará.

Ejemplo:

TCP_PORT=5000
UDP_PORT=5001

2. Compilar los programas

g++ -o servidor servidor.cpp
g++ -o cliente cliente.cpp

3. Ejecutar el servidor

./servidor

Esto abre dos procesos:

- Uno escucha mensajes TCP.
- Otro espera solicitudes UDP para detección automática.

4. Ejecutar el cliente

./cliente

- El cliente busca automáticamente el servidor.
- Solicita el nombre del usuario.
- Permite escribir destinatario y mensaje.

5. Enviar mensajes

Cuando el cliente esté conectado:

1. Escriba el nombre del destinatario.
2. A continuación, el mensaje.
3. Formato <destinatario> <mensaje>

Ejemplo:

Luis Hola, ¿cómo estás?

6. Recibir mensajes

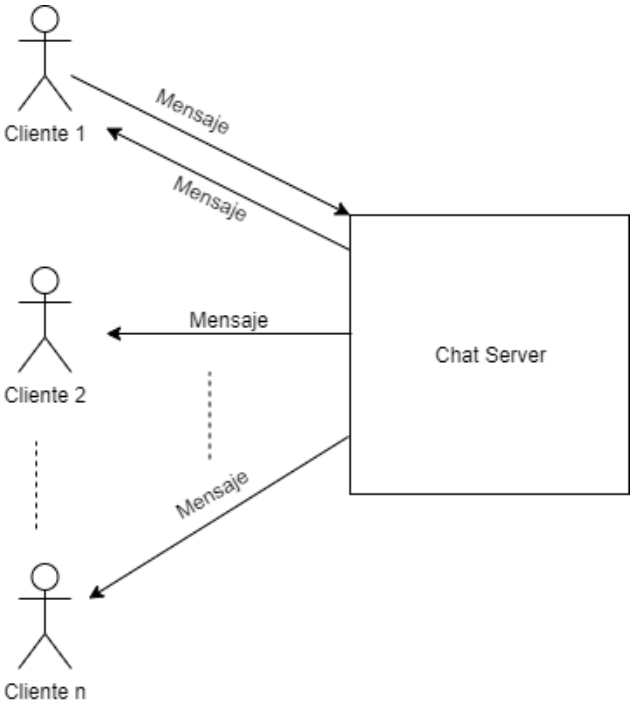
- Los mensajes se muestran en color verde.
- Si había mensajes pendientes, se muestran automáticamente al conectarse.

7. Finalizar

- Use Ctrl+C para cerrar servidor o cliente.

Arquitectura de la aplicación

Se presenta un diagrama acerca de la estructura de la aplicación:



Servidor

- `leerConfiguracion()` → Carga los puertos desde `config.txt`.
- `iniciarServidorTCP()` → Escucha conexiones TCP y maneja mensajes.
- `iniciarDescubrimientoUDP()` → Responde a mensajes de broadcast.
- `registrarUsuario()` → Almacena info de usuario conectado.
- `reenviarMensaje()` → Reenvía o almacena mensaje en espera.

Cliente

- `leerConfiguracion()` → Lee puertos desde `config.txt`.
- `discoverServerIP()` → Encuentra al servidor por UDP.
- `registrarUsuario()` → Envía nombre al servidor.
- `recibirMensajes()` → Proceso hijo que imprime mensajes.
- Proceso padre → Captura entrada del usuario para enviar mensajes.