Protocol Chat

Proyecto N1

Telemática

26 de Septiembre de 2024

1. Introducción

En el contexto actual de la tecnología de la información, la comunicación en tiempo real entre usuarios es fundamental. Este proyecto tiene como objetivo desarrollar una aplicación de chat utilizando el modelo cliente/servidor, donde los estudiantes implementarán un protocolo de capa de aplicación que permita la interacción entre múltiples clientes a través de un servidor central. La aplicación de chat debe ser capaz de gestionar la comunicación en tiempo real, permitiendo que los usuarios envíen y reciban mensajes de forma eficiente y efectiva. De esta manera, se busca desarrollar las competencias necesarias en el diseño y desarrollo de aplicaciones concurrentes en red. Para lograr esto, se empleará la API Sockets con el fin de escribir un protocolo de comunicaciones que permita a una aplicación cliente comunicarse con una aplicación servidor.

2. Objetivos

- **Desarrollar una aplicación de chat cliente/servidor**: Los estudiantes implementarán tanto el lado del cliente como el del servidor, utilizando un protocolo de comunicación diseñado específicamente para este proyecto.
- **Implementar un protocolo de capa de aplicación**: Se espera que los estudiantes diseñen y desarrollen un protocolo que permita la transmisión de mensajes de texto entre el cliente y el servidor. Este protocolo debe ser robusto y fácil de extender para futuras funcionalidades.
- **Utilizar la API de sockets**: La comunicación entre el cliente y el servidor se llevará a cabo utilizando sockets, permitiendo el envío y recepción de datos de forma efectiva.
- **Soportar múltiples conexiones concurrentes**: El servidor debe ser capaz de manejar múltiples conexiones de clientes al mismo tiempo, permitiendo que diferentes pares de clientes se comuniquen simultáneamente sin interferencias.

3. Marco Teórico

3.1. Sockets

Tradicionalmente, a los desarrolladores de aplicaciones, las APIs de desarrollo de los diferentes lenguajes de programación, les realizan abstracciones que les ocultan los detalles de implementación de muchos aspectos, y el proceso de transmisión de datos no es la excepción.

Un socket es una abstracción a través de la cual las aplicaciones pueden enviar y recibir datos. Al respecto, el socket se puede entender como el mecanismo que utilizan las aplicaciones para conectarse a la red, específicamente con la arquitectura de red o stack de protocolos y de esta forma comunicarse con otras aplicaciones que también se encuentran "conectadas" a la red. De esta forma, la aplicación que se ejecuta en una máquina escribe los datos que desea transmitir en el socket y estos datos los puede leer otra aplicación que se ejecuta en otra máquina.

3.2. Tipos de Sockets

Existen diferentes tipos de sockets. Para este proyecto vamos a utilizar la API Sockets Berkeley, específicamente los sockets tipo Stream (SOCK STREAM) o Datagram (SOCK DGRAM). En el marco de este proyecto, acorde al protocolo que está implementando usted debe decidir qué tipo de socket va a emplear y justificar a la luz de los requerimientos de su aplicación.

4. Requerimientos del Proyecto

4.1. Diseño de la Aplicación

- **Arquitectura**: La aplicación se basará en un modelo cliente/servidor. El servidor será responsable de gestionar las conexiones de los clientes y el intercambio de mensajes, mientras que los clientes se encargarán de la interfaz de usuario y la comunicación con el servidor.

- **Chat Protocol**: Se requiere que usted diseñe y especifique un protocolo de comunicaciones para la capa de aplicación que permita el envió de la información entre la aplicación cliente (ChatClient) y el servidor de comunicaciones (ChatServer).

  Tenga en cuenta que el líder técnico del equipo de desarrollo, indica que el protocolo debe ser codificado tipo texto. Es importante resaltar que usted debe describir la especificación del servicio, el vocabulario de mensajes, las reglas de procedimiento y todo lo que usted considere necesario para el buen funcionamiento del protocolo.

  De esta forma, se espera su protocolo se integre a la arquitectura de red TCP/IP a través de la API de sockets. Es así como su aplicación cliente o servidor consume los servicios de "MyChatProtocol" el cual representa el protocolo que usted diseña y especifica en la capa de aplicación.

- **Protocolos de transporte**: Los estudiantes deben seleccionar por implementar su protocolo sobre UDP o TCP, considerando las ventajas y desventajas de cada uno en el contexto de la aplicación de chat.

4.2. Funcionalidades del Servidor

- **Conexión de Clientes**: El servidor debe permitir que múltiples clientes se conecten al mismo tiempo.
- **Recepción y Envío de Mensajes**: El servidor debe recibir mensajes de los clientes y retransmitirlos a los destinatarios correspondientes.
- **Concurrencia**: Las aplicaciones en red son concurrentes. En este sentido, el servidor para el juego que se está implementando no es la excepción. De esta forma, se requiere que su servidor se capaz de soportar de manera concurrente múltiples parejas de usuarios jugando de manera simultánea. Para efectos de esta práctica, se permite el uso de hilos para soportar la concurrencia del servidor.
- **Manejo de Errores**: Se deben implementar mecanismos de manejo de errores para asegurar que las conexiones se gestionen de forma robusta y eficiente.

4.3. Funcionalidades del Cliente

- **Interfaz de Usuario**: Los estudiantes deben crear una interfaz de usuario básica modo texto (consola) que permita a los usuarios enviar y recibir mensajes.
- **Conexión al Servidor**: El cliente debe ser capaz de conectarse al servidor y autenticarse adecuadamente.
- **Envío de Mensajes**: Los usuarios deben poder enviar mensajes a otros usuarios de manera sencilla y rápida.

5. Detalles de Implementación y Uso de la Aplicación

A continuación, se detallan algunos aspectos que se deben considerar para la realización de su proyecto.

- La aplicación cliente puede ser escrita en el lenguaje de preferencia del grupo que soporte la API sockets, tal como: Python 3.X, Java, C, C++, C#, Go, etc.
- Se debe utilizar la API de Berkeley.
- La aplicación servidor se debe desplegar y ejecutar en un servidor en la nube. Para esto, utilice la cuenta de AWS Academy.
- Para documentar el diseño e implementación de su solución, por favor utilice herramientas como UML o cualquier otra con el fin de ilustrar las funcionalidades, así como el funcionamiento de este.

6. Aspectos Para Considerar para el Desarrollo

- **Equipo de trabajo**: El proyecto debe ser desarrollado en grupos de 3 personas como máximo. NO debe ser desarrollado de manera individual.
- **Cronograma**: Defina un plan de desarrollo, hitos, victorias tempranas, hasta la finalización del proyecto.
  - Tenga presente que tiene dos semanas calendario para desarrollar el proyecto. Se espera que el 60 % del tiempo lo invierta en el análisis, diseño, implementación, pruebas y documentación para la aplicación servidor. El otro 40 % restante, para la aplicación cliente.
- **Punto de Chequeo 1**:
  - Octubre 6: Ya para esta fecha, usted debería tener la aplicación Servidor totalmente implementada.
- **Tiempo**: Debe gestionar muy bien su tiempo para poder alcanzar el objetivo final del proyecto. Empiece a trabajar apenas le publiquen su enunciado.
- **Consultas/dudas**: Si tiene alguna duda, por favor, acuda a su profesor y/o coordinador de la asignatura lo más pronto posible.
- **Cliente o Servidor**: Se sugiere que empiece por desarrollar la aplicación servidor y luego continúe con la aplicación cliente.

7. Entrega

- **Repositorio**: Trabaje con Git. Cree un repositorio privado para su proyecto. Recuerde que usted no puede compartir el código de su trabajo con nadie.
- **Documentación**: La documentación se debe incluir en el repositorio en un archivo README.md. En este archivo se requiere que usted incluya los detalles de implementación donde como mínimo se esperan las siguientes secciones:
  - Introducción.
  - Desarrollo.
  - Aspectos Logrados y No logrados.
  - Conclusiones.
  - Referencias.
- **Video**: Entregue y sustente al profesor mediante un video creado por el grupo, donde explique el proceso de diseño, desarrollo y ejecución (no más de 20 minutos). Posteriormente y, en caso de ser necesario, se le citará a una sustentación presencial. Se debe ser claro en el video el funcionamiento de la solución, tanto para el cliente como para el servidor.

- **Fecha de entrega Final**: Martes 15 de Octubre de 2024.
- **Mecanismo de entrega**: Por el buzón de Interactiva virtual se debe realizar la entrega. La entrega debe incluir un enlace a un repositorio en GitHub. A partir de esta fecha y hora, no se puede realizar ningún commit al repositorio.

8. Evaluación

- **Funcionalidad**: La aplicación cumple con los requisitos establecidos y funciona como se espera.
- **Calidad del Código**: El código es limpio, bien estructurado y documentado.
- **Innovación y Creatividad**: Se valorará la implementación de características adicionales que mejoren la funcionalidad del chat.
- **Presentación del Informe**: La claridad y profundidad del informe técnico.
- Se realizará un proceso de sustentación para la verificación de la práctica acorde a lo entregado por el buzón y con lo explicado inicialmente en su video.

9. Referencias

- https://beej.us/guide/bgnet/
- https://beej.us/guide/bgc/
- https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/