TelePong
Proyecto N1
Telemática
2023-2

Versión: 1.0
Última actualización: Sept 4 de 2023

1 Introducción

El propósito de este proyecto es el desarrollo de habilidades para la programación en red, aplicaciones concurrentes, así como el diseño e implementación de protocolos.

En el contexto de este curso, un protocolo se entiende como el conjunto de normas, reglas o parámetros que rigen el intercambio coordinado de mensajes entre dos entidades paritarias. Los protocolos son fundamentales para la comunicación entre las aplicaciones que se ejecutan o corren en una red de datos.

Para lograr los objetivos propuestos, se va a emplear la API de Sockets Berkeley para escribir una aplicación servidor que reciba peticiones desde una aplicación cliente.

Por favor, tenga presente que este es un proyecto de asignatura bastante demandante en tiempo. Entre más pronto lo empiece mejor. Igualmente, siéntase libre de consultar con su profesor las diferentes dudas o preguntas que tenga del mismo.

2 Antecedentes

2.1 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 ex 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 e una máquina escribe los datos que dese transmitir en el socket y éstos datos los puede leer otra aplicación que se ejecuta en otra máquina.

2.2 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, usted debe decidir qué tipo de socket va a emplear y justificar a la luz de los requerimientos de su aplicación, así como del diseño de su protocolo, porque escoge uno, el otro o ambos y en que escenarios.

3 Escenario

3.1 Contexto

Usted ha sido contratado por una empresa que se dedica al desarrollo de juegos en línea. Específicamente, usted hace parte del equipo que se encarga de desarrollar toda la infraestructura para que los usuarios puedan acceder a los diferentes “Networked Games” (NGs) que la compañía desarrolla. Este tipo de juegos involucra uno o más jugadores que se conectan por Internet.

Este año, la compañía está celebrando su quinto aniversario y por este motivo quiere celebrar con el lanzamiento de un par de juegos “retro”, a usted le toca implementar el juego, Pong.

Pong es un juego clásico de computadoras, cuya descripción la puede observar en (https://en.wikipedia.org/wiki/Pong).

Para efectos de su proyecto, usted hace parte del equipo de desarrollo encargado de toda la parte de comunicaciones para que dos o más usuarios puedan jugar Pong. Es así como se ha definido que la aplicación responde a un estilo arquitectónico cliente/servidor. De esta forma, los usuarios descargan una aplicación cliente la cual se conecta a un servidor.

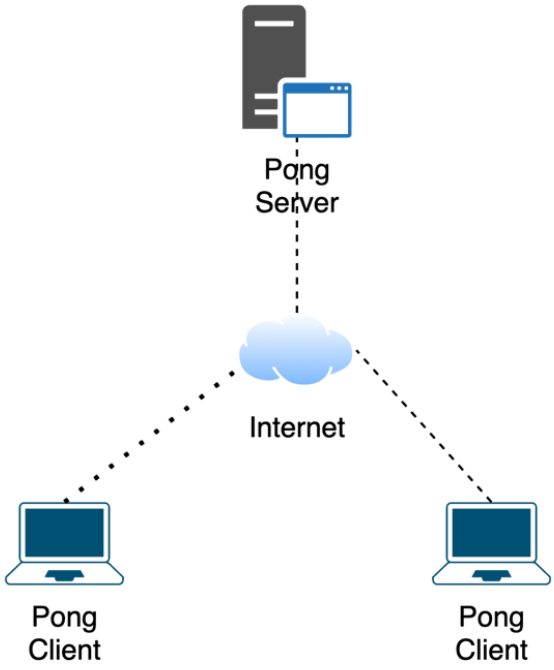

Figura 1. Vista de alto nivel para el juego Pong.

3.2 Requerimientos

3.2.1 Desarrollar la aplicación Cliente y la aplicación Servidor.

El juego Pong se debe diseñar respondiendo a una arquitectura cliente/servidor, donde se diferencian claramente las dos entidades: cliente y servidor.

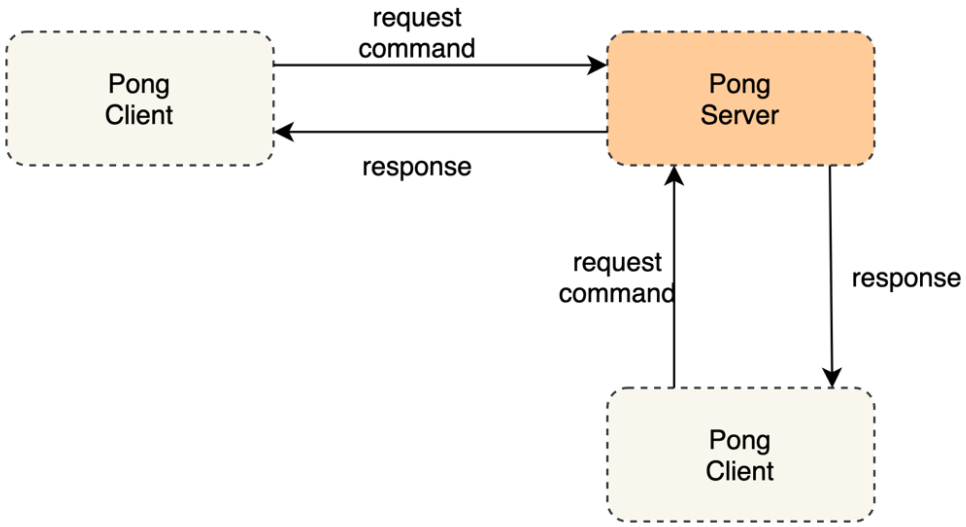

Figura 2. Arquitectura Cliente/Servidor para el juego en línea Pong

En términos generales para este tipo de arquitecturas de juegos en línea, el cliente recibe una entrada (input) del usuario y procede a enviar un comando al servidor. Por otro lado, el servidor que es el encargado de mantener el estado del juego y comunicarse con los clientes; ejecuta la orden, actualiza el estado, y envía el mensaje de notificación a los otros clientes.

De esta forma el cliente se debe conectar y registrar (crear su perfil) a un servidor para poder jugar. El proceso de registro es simple: me conecto al servidor enviado un mensaje con nickname, dirección de correo electrónico y cualquier otro dato relevante que usted considere.

Tenga en cuenta que, como se mencionó anteriormente, en esta arquitectura el servidor es el responsable de mantener el "estado" del juego. Es así como el servidor recibe la información de los clientes conectados y que se encuentran en un juego, procesan la información y actualizan el estado. Una vez esto se logra, se envía la información a todos los clientes.

Lo anterior implica que debe mantener la información de las "n" parejas de clientes que están jugando entre sí, validad los movimientos, resultados, etc, relacionados con el juego de cada pareja de clientes. De igual forma, debe registrar todos los eventos, movimientos, etc, de un cliente e informar al otro. Todo lo relacionado con eventos, los debe imprimir por consola, así como enviarlos a un archivo de log.

3.2.2 Diseño, especificación e implementación de protocolo:

1. 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 (PongClient) y el servidor para el juego (PongServer).

   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 se ejecuta sobre la siguiente jerarquía de protocolos, tal como se ilustra en la figura XXX, donde "MyAppGameProtocol" representa el protocolo que usted diseña y especifica en la capa de aplicación.

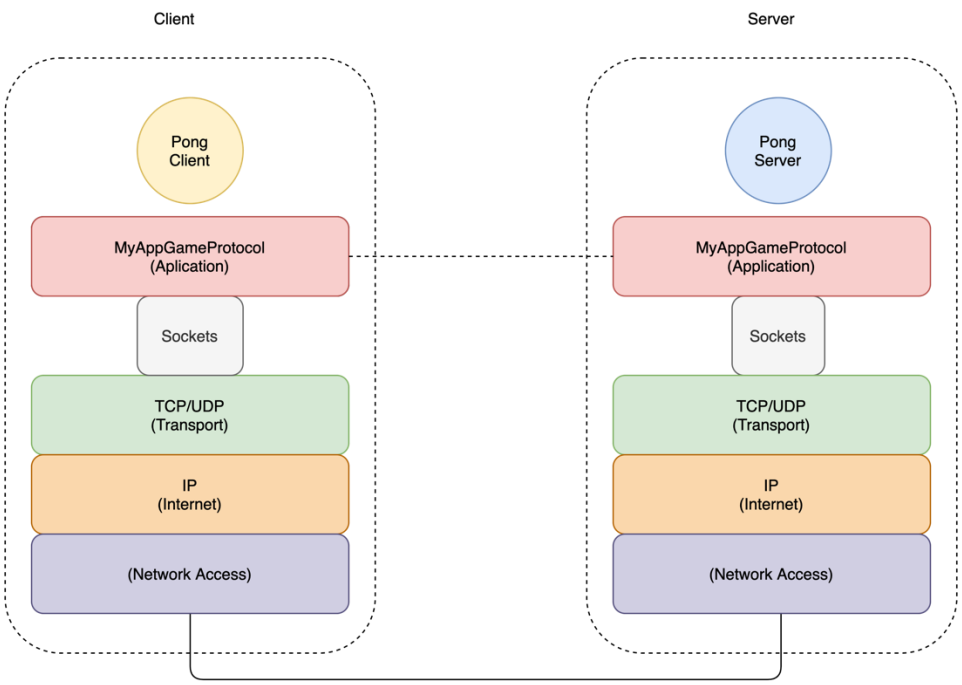

Figura 3. Jerarquía de Protocolos

2. Implementación del Protocolo: Se requiere que usted implemente su protocolo tanto en la estación cliente como servidor. Para la implementación de este protocolo se espera que usted lo codifique acorde al diseño y especificación planteando en el literal 1 de esta subsección.

3.2.3 Su servidor debe ser capaz de soportar múltiples clientes.

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.

4 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 (PongClient) puede ser escrita en el lenguaje de preferencia del grupo que soporte la API sockets tal como: Python 3.X, Java, C++, C#, etc.
- La aplicación servidor (PongServer) **SOLO** debe ser implementada en lenguaje C.

    - **Nota**: No se recibe ningún proyecto, en su totalidad, con la aplicación servidor desarrollada en un lenguaje diferente a C.
- No se permite utilizar ninguna clase existente o personalizada de sockets. Se debe utilizar la API de Berkeley.
- Su servidor debe implementar el concepto de “logger”. Esto con el fin de poder visualizar por la terminal todas las peticiones entrantes , así como la respuesta que se envía a cada cliente.
- Su servidor de juego se debe ejecutar de la siguiente manera:
    - $./server <PORT> <Log File>, donde:
        - PORT es el puerto de la máquina en la cual está corriendo su servidor de juego.
        - LogFile es el archivo de log que se va a generar con toda la información concerniente a peticiones, errores, info, etc.
- La aplicación PongServer se debe desplegar y ejecutar en un servidor en nube. Para esto utilice la cuenta de AWS academy.
- Para documentar el diseño y especificación de "MyAppGameProtocol”, por favor apóyese de herramientas como UML o cualquier otra con el fin de ilustrar las funcionalidades, así como el funcionamiento de este.

5 Entrega

- **Equipo de trabajo:** El proyecto debe ser desarrollado en grupos de 3 personas como máximo. **NO** debe ser desarrollado de manera individual.
- **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 repo 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
    - Conclusiones
    - Referencias
- **Fecha de entrega:** 3 de octubre, las sustentaciones por definir en esa semana.
- **Mecanismo de entrega:** Por el buzón de Interactiva virtual se debe realizar la entrega (a más tardar el 3 de octubre). 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.

6 Referencias

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