Universidad Nacional Autónoma de México

Facultad de Ciencias

Ciencias de la Computación

Práctica No 6

Cliente - servidor

ASIGNATURA

Redes de Computadoras 2025-1

Integrantes del equipo DIA 2.0

López Diego Gabriela 318243485

San Martín Macías Juan Daniel 318181637

Rivera Zavala Javier Alejandro 311288876

Juárez Ubaldo Juan Aurelio 421095568

Ortiz Amaya Bruno Fernando 318128676

FECHA DE ENTREGA

27 de Octubre del 2024

INTRODUCCIÓN

Los investigadores Pokémon están empezando a crear simulaciones de combates, por lo que deben visualizarlos, el inconveniente, es que no se pueden registrar muchas de las mediciones en un combate real, por lo que te piden que generes un programa que haga dicha simulación. Cabe recordar que estos profesores viven distanciados, por lo que deberán crear el programa que se conecte a un servidor donde se emule el combate.

CLIENTE - SERVIDOR

Una característica importante en redes de computadora, es el intercambio de información entre dos máquinas que componen dicha red. Aunque si bien, la información se tiende a dividir en mensajes y a su vez en paquetes, estos paquetes tienden a ser divisiones de lo que viene siendo el mensaje y una vez llegado al destino se ensamblan. Cabe aclarar, que estos pueden llegar íntegros o bien pueden perder datos en el camino (ya sea mala conexión, desconexión o algún factor adicional).
En el modelo Cliente – Servidor, se compone de un cliente y un servidor (también pueden ser n servidores y n clientes), el servidor lo que hace principalmente es procesar las respuestas del cliente y devolver normalmente algún estatus (estos normalmente no son visibles tal cual al usuario final o si lo son, vienen decorados) y en caso del cliente, solicita alguna cosa (ya sea información, o que procese información). Cabe aclarar, que el servidor puede llegar a ser una computadora ordinaria a una computadora con mayor nivel de procesamiento, esto se va a definir de acuerdo a lo que se va a desarrollar así como la cantidad de usuarios.

ACTIVIDAD

Para la actividad deberás programar alguno de los siguientes ejercicios, estos varían de a la dificultad, por lo que deberás programar entre 1 o 4 ejercicios.

- Selecciona 1
    1. Batallas pokemon (Sus pokemones fueron definidos al inicio del excel)
    2. Serpientes y escaleras
    3. Ajedrez
- Selecciona 2
    1. Adivina quien
    2. Uno (Básicamente queda raro para dos personas, pero funciona)
    3. Póker o baraja española
- Selecciona 4 ***(Nuestro equipo decidió implementar las opciones marcadas en amarillo y azul)***
    1. Gato
    2. Piedra papel o tijera
    3. 4 en línea
    4. Memorama
    5. Ahorcado (Genera una colección medianamente grande de palabras, o bien generas la cadenas pseudo aleatoriamente)

6. Lotería

Para esto usarán sockets, uno será para el cliente y el otro para el servidor, deben tener en consideración lo siguiente:

1. Podrán usar solo sockets y algunas bibliotecas adicionales (Normalmente de la paquetería Util de Java y su equivalente en otro lenguaje)
2. No pueden usar paqueterias que resuelvan el problema en general, deben crear la solución por sí mismos.
3. En caso de que se ignore lo anterior, se calificará con 0 dicho ejercicio
4. Debe venir bien documentado, así como una sección de cómo se juega (ya sea dentro del código o un txt)
5. Por la parte del Servidor, deberán programar un AI básica que realice los movimientos o jugadas, aunque no debe ser muy refinada, debe funcionar por lo menos decente.
6. Naturalmente debe ser 1 a 1 la conexión, pero si quieren agregar una conexión adicional está bien.

TEORÍA

1. Investiga en qué consisten las distintas versiones de HTTP.

   HTTP es el protocolo en el que se basa la Web. Fue inventado por Tim Berners-Lee entre los años 1989-1991. HTTP ha evolucionado, desde un protocolo destinado al intercambio de archivos en un entorno de un laboratorio semi-seguro, al actual laberinto de Internet, sirviendo ahora para el intercambio de imágenes, vídeos en alta resolución y en 3D.
   La versión inicial de HTTP, no tenía número de versión; aunque posteriormente se la denominó como 0.9 para distinguirla de las versiones siguientes. HTTP/0.9 es un protocolo extremadamente sencillo: una petición consiste simplemente en una única línea, que comienza por el único método posible GET, seguido por la dirección del recurso a pedir. No usa cabeceras HTTP, con lo cual únicamente es posible transmitir archivos HTML, y ningún otro tipo de archivos. Tampoco había información del estado ni códigos de error. Ante sus limitaciones, se crea HTTPS/1.0. Donde el concepto de cabeceras de HTTP, se presentó tanto para las peticiones como para las respuestas, permitiendo transmitir otros documentos además de HTML. Finalmente, la primera versión estandarizada de HTTP: el protocolo HTTP/1.1, se publicó en 1997, tan solo unos meses después del HTTP/1.0, la cual añadió muchas mejoras como permitir una conexión reutilizado, enrutamiento, respuesta a las peticiones, mecanismos de gestión a la cache, negociación del contenido entre servidor y cliente, alojar varios dominios en la misma IP gracias a sus cabeceras, etc.
   En 2015 por fin llegó una especificación más adecuada para la web moderna que HTTP/1.1: HTTP/2. Se centra en la multiplexación de conexiones, es decir, permite mantener varias “conversaciones” con el servidor al mismo tiempo en la misma conexión. Esto permite la descarga de múltiples archivos simultáneamente sin tener que consumir tantos recursos como en HTTP/1.1. El último borrador del estándar fue publicado en Mayo de 2021, HTTP/3 es básicamente las mejoras propuestas por

HTTP/2 sobre una nueva capa de transporte que no es TCP: QUIC. QUIC está basado en UDP, pero establece la forma de crear canales con control de flujo, cifrado y multiplexación para poder servir mejor a la web moderna.

2. Investiga algún protocolo similar a HTTP

   HTTPS, que significa Hypertext Transfer Protocol Secure, es la versión segura de HTTP. Incorpora una capa adicional de seguridad mediante el uso de certificados SSL (Secure Socket Layer) o TLS (Transport Layer Security) para cifrar las comunicaciones entre el navegador del usuario y el servidor web.
   Esto asegura que cualquier dato transferido, como información personal o detalles de tarjetas de crédito, permanezca privado y protegido contra posibles incursiones externas con malas intenciones. Además de mejorar la seguridad de una web, HTTPS también autentifica las páginas, asegurando a los usuarios que están conectados a la web correcta y no a una réplica fraudulenta

3. Aunque si bien algunos juegos son 1 a 1, suponiendo que se puede, de qué manera sería meter otra conexión adicional para que funcione con más personas.

   Para permitir que varios jugadores se involucren en una misma ronda de un mismo juego como el gato o piedra, papel o tijeras, podríamos asignar un turno a cada participante. De este modo, todos podrán jugar en la partida, y su participación se bloqueará hasta que llegue su turno. A medida que avanza la partida, los jugadores serán eliminados uno por uno, hasta que solo quede un único ganador. Durante este proceso, aunque no puedan jugar, todos los participantes tendrán la capacidad de observar el estado del juego y las actualizaciones sobre los movimientos de los demás. Además, en juegos como el gato, podríamos considerar reglas que permitan jugar en un tablero más grande y definir claramente cómo se determina el ganador en caso de empate.

4. Qué ventajas ves sobre este tipo de conexión, así como desventajas.
   Ventajas:
   - **Centralización del control.** En el modelo cliente-servidor, el servidor maneja la lógica y el control del juego, lo que nos asegura que cada cliente tenga una experiencia consistente.
   - **Seguridad y monitoreo.** Ya que el servidor actúa como punto de control único, nos permite monitorear y asegurar que los datos de los clientes y del juego se mantienen seguros.
   Desventajas:
   - **Dependencia hacia el servidor.** Ya que es nuestro punto de control único, si se desconecta o falla, los clientes no tendrán acceso al juego o servicio.
   - **Problemas de latencia.** Esto puede ser común en los videojuegos, ya que si el servidor está lejos (físicamente) del cliente, puede haber retrasos en la respuesta, lo que afecta la experiencia de usuario.

5. Suponiendo que tuviera más ejecuciones tu servidor por otros usuarios, contesta la siguientes preguntas:
    a. Si un usuario ya tiene iniciada alguna partida y otro se conecta, ¿Juega la misma partida que el usuario 1 o se crea una nueva?
    En el servidor TIC-TAC-TOE, cada cliente recibe un hilo exclusivo, (Clase ClientReplier) que maneja la lógica del juego de manera independiente, por lo que es una partida nueva.
    b. Si dos usuarios se conectan casi simultáneamente, ¿a quien atiende primero el servidor?
    El servidor procesa las conexiones en el orden en el que llegan, por lo que simplemente se atiende al primero en llegar sin ningún criterio en especial
    c. Suponiendo que n usuarios se conectan al servidor no importa en qué orden, ¿Cómo procesa las solicitudes el servidor?
    De nuevo, el servidor asigna un hilo a cada cliente en orden de llegada, y cada hilo maneja un juego de manera independiente, por lo que cada cliente puede jugar de manera paralela.
    d. Propón una mejora para que se puedan atender las solicitudes de acuerdo a lo siguiente:
        i. Siendo una programación sin alguna biblioteca adicional.
        Respuesta:
        Se podría implementar una cola (FIFO) para manejar las solicitudes. Esto ayudaría en escenarios con muchos usuarios conectados simultáneamente, de forma que el servidor procesa solicitudes en el orden en que las recibe.
        ii. Usando un framework o biblioteca adicional.
        Respuesta:
        Con la biblioteca asyncio, el servidor puede manejar múltiples clientes de forma asincrónica, mejorando la eficiencia al procesar varias conexiones sin la necesidad de crear un hilo por cada cliente. Esto es útil para escalar el servidor y optimizar el uso de recursos en el caso de muchos usuarios simultáneos.
        iii. Justifica de manera detallada ambos casos.
6. Con lo investigado en la pregunta anterior, menciona algunos servicios comerciales y otros no tan comerciales donde se emplee el cliente servidor y como lo hacen (a grandes rasgos, no en cuestión de programación)
    Servicios comerciales:
    - **Servicios de Streaming:** Utilizan modelos Cliente-Servidor para transmitir contenido Multimedia, los clientes envían solicitudes de reproducción de contenido, y el servidor responde con el contenido adecuado.
    - **Videojuegos multijugador en línea:** Utilizan el modelo Cliente-Servidor para que el servidor gestione el estado del juego y se asegura que todos los jugadores tengan una experiencia sincronizada.
    Servicios no comerciales:
    - **Servicios públicos en internet:** Si tomamos como ejemplo Wikipedia, que es una página no comercial, los usuarios (clientes) envían solicitudes al servidor para obtener

páginas de contenido.
- **Redes de juego LAN:** En juegos de red local, un dispositivo actúa como servidor (comúnmente llamado host), y permite que los demás jugadores en la misma red se conecten y jueguen juntos.

7. Escribe lo aprendido sobre esta práctica así como dificultades.
   Para el desarrollo de esta práctica, decidimos implementar 4 juegos sencillos, que nos permitiría aprender lo esencial sobre el modelo cliente - servidor. Aprendimos principalmente cómo utilizar la herramienta sockets e hilos en Python para establecer protocolos de comunicación entre un cliente (jugador/usuario) y el servidor (jugador oponente/IA básica) al enviar y recibir datos, así como definiendo la manera en la que se enviarán los mensajes correctamente y sin fallos entre ambos. Usamos hilos ya que nos permitió una correcta sincronización entre los mensajes y que además nos otorgaba la posibilidad de conectar más de un cliente. Investigamos e indagamos mucho sobre las bibliotecas socket y threading en Python, ya que sabíamos que eran herramientas importantes que nos iban a permitir un buen funcionamiento en nuestros juegos. Una de las mayores dificultades que se nos presentaron para desarrollar los juegos, fue la correcta sincronización al envío de mensajes del servidor a los clientes. En varias ocasiones, observamos el correcto funcionamiento del juego del gato pero con el servidor enviando mensajes en un orden incorrecto. También observamos que el cliente no lograba mandar de forma exitosa los mensajes de su movimiento en la partida del juego al servidor. Sobre la IA básica que se encuentra en los servidores, al ser juegos muy simples, simplemente utilizamos la biblioteca random de Python para asignar el movimiento del oponente. Nota importante, gracias a los hilos, para los juegos logramos que varios clientes puedan estar conectados y en comunicación con el servidor, permitiendo así la ejecución de varias partidas con varios usuarios.

BIBLIOGRAFÍA

- Patricio, H. (2021, diciembre 15). HTTP 1.1, HTTP/2 y HTTP/3. The Dojo MX Blog. https://blog.thedojo.mx/2021/12/15/http-1-1-http-2-y-http-3.html
- Evolución del protocolo HTTP. (s/f). MDN Web Docs. Recuperado el 23 de octubre de 2024, de https://developer.mozilla.org/es/docs/Web/HTTP/Evolution_of_HTTP
- Acibeiro, M. (2023, diciembre 27). Diferencia entre HTTP y HTTPS: Seguridad y rendimiento en la web. GoDaddy Resources - Spain; GoDaddy. https://www.godaddy.com/resources/es/seguridad/diferencia-entre-http-y-https