Protocolo del equipo 1 modificado con base a inspiración del equipo 2.

Equipo 1:
- Nathalie Alfaro Quesada, B90221
- Jesús Alonso Porras Arguedas C26007
- Jordan Esteban Barquero Araya C30965
- Rodrigo Mendoza Quesada C04813

Equipo 2:
- Jorge Salas
- Gabriel Bermudez
- Alejandro
- Jordan Obando

Protocolo de comunicación formal

Propósito

- Objetivo: Estandarizar el diálogo Cliente ↔ Tenedor ↔ Servidor de Figuras en la simulación para listar y consultar figuras de tipo ASCII.
- Ámbito: Simulación sin sockets, de momento (IPC local); no define seguridad ni autenticación.
- Inspiración: Sintaxis HTTP (líneas y encabezados), pero con nombres propios y reglas simplificadas.

Transporte (IPC) y codificación

- Transporte: Cualquier IPC local: Pipes, FIFOs, colas POSIX (mq_*), o memoria compartida + semáforos.
- Codificación: UTF-8 en todo el mensaje.
- Conexión lógica: Persistente hasta que una parte envíe 'Connection: close' o termine el proceso/hilo. Se propone que tanto los servidores como los tenedores envíen un aviso mediante UDP a los demás dispositivos de la misma red. Dicho aviso debe indicar el nodo en que se encuentra activo, especificando si actúa como servidor o como tenedor, además de proporcionar la dirección IP y el puerto de comunicación correspondiente. De esta forma, los demás equipos pueden conocer qué nodos están disponibles y listos para establecer conexiones a través de TCP.

Conexión

Antes de iniciar la comunicación, los actores que participan en esta deben estar conscientes de la presencia de los demás. Para lograr esto, se propone que los componentes establezcan una conexión al ser encendidos por medio de un acuerdo.

Al establecer esta conexión basta con que un componente le comunique a los demás que se encuentra activo para que reciba de respuesta quiénes más lo están y cuál es

su rol dentro de la comunicación. Una vez establecida la conexión, pueden comunicarse utilizando el protocolo.

Hay 2 casos para este modelo de comunicación. Caso uno, el componente se enciende, realiza un broadcast con: dirección IP, puerto y tipo de componente (tenedor, servidor o cliente); como es el único, no recibe respuesta y queda a la espera.

Caso dos, cuando ya hay otro componente: en este caso cuando el segundo componente se enciende, recibe respuesta y puede establecer de una vez la conexión. Cada uno debe recibir el reconocimiento del otro que recibió la información de la conexión, de lo contrario debe esperar a que otro componente inicie la conexión.

El flujo del caso dos entre tenedor y servidor se vería así:

> [Servidor: espera] broadcast para solicitar conexión (no recibe nada, queda en espera)
> [Tenedor: espera] broadcast para solicitar conexión -> [Servidor]
> [Servidor: iniciado] envía al tenedor la respuesta con su información -> [Tenedor]
> [Tenedor: conectado] envía ACK al servidor de que tiene todo para la conexión -> [Servidor: conectado]

Formato del mensaje

Se planea que los mensajes tengan una estructura sencilla de dónde comienza y termina el mensaje para saber que solo hay que tomar lo que está en estas etiquetas creadas, estas etiquetas estarán en mayúscula y el inicio se denota con la palabra BEGIN seguido por un slash y luego continua el cuerpo o la información relevante, luego sigue otro slash con la palabra END que denota que termina ese comunicado, un ejemplo de la estructura principal es:

BEGIN/data/END

Delimitación de encabezados:

- BEGIN/
- /END

Las instrucciones principales y definidas por el momento presentan este tipo de estructura, cuando se ejecuta el programa, se enciende el tenedor y el servidor, por lo cual estos indican la dirección IP de la computadora donde están funcionando y el puerto de comunicación, además si se desconectan, también emiten el mensaje:

Servidor:

- BEGIN/ON/SERVIDOR/{ip}/{puerto}/END
- BEGIN/OFF/SERVIDOR/{ip}/{puerto}/END

Tenedor:

- BEGIN/ON/TENEDOR/{ip}/{puerto}/END

- BEGIN/OFF/TENEDOR/{ip}/{puerto}/END

Para estandarizar las peticiones UDP, se definen los siguientes puertos para el mensaje de broadcast:
- Tenedor: 4321
- Servidor: 1234

Estos puertos se deben usar para recibir mensajes de conexión y desconexión.

En caso de que el cliente solicite una figura que no aparece en la lista de figuras, es porque no está contenida en el sistema de archivos, por lo cual se le desplegará un mensaje de error con un código:

- 404 - Not found - La figura no existe en el sistema de archivos

En caso de que algunos de los tres protagonistas (Cliente, Tenedor o Servidor) pierda la conexión por diversas razones, estos emitirán un código más un mensaje que indican el incidente:

- 100 - Servidor sin conexión
- 101 - Tenedor sin conexión
- 102 - Cliente sin conexión

Comandos

Mostrar la lista de figuras con el comando LIST y si hace esta acción con éxito, despliega la lista de figuras junto con el código de 200 OK:

LIST /figures
Respuesta:
PPI/1.0 RES 1 200 OK
ballena
pez
tortuga

El “200 OK” indica que la acción se puede realizar con éxito.

Para solicitar una figura en específico se usa el comando GET más el nombre de la figura y si se realiza la acción con éxito, se despliega la figura completa:

GET /nombreDeLaFigura

:)

Si desea agregar una figura que no está contenida en el sistema de archivos, esta acción se podrá realizar con el comando ADD más el nombre del archivo de texto que

contiene la figura, si la figura fue añadida correctamente, seguido se muestra un código de éxito en la acción “200 OK”:

ADD /figuraNueva.txt
200 OK

Administración de recursos

El uso eficiente de memoria y almacenamiento es esencial para garantizar el rendimiento del sistema y la disponibilidad de los recursos, por lo cual nos centramos en que el servidor como el tenedor mantienen listas de figuras en estructuras dinámicas en memoria (como “vector” o “queue”) permitiendo su crecimiento o eliminación de manera eficiente conforme se actualizan los recursos disponibles, además se emplean mecanismos de liberación de memoria como destructores luego de desconexiones, de modo que no se acumulen referencias a servidores inexistentes.

Cambios realizados entre protocolos

El protocolo del equipo 2 se modifica con el protocolo del equipo 1, en la sección de **Propósito** del equipo 2 se mantiene al igual que la sección de **Transporte (IPC) y codificación** aunque en “Conexión lógica” se detalla más con base al protocolo del equipo 1.

En la sección **Conexión** se añade lo comentado en clase entre ambos equipos.

En la sección **Formato del mensaje** del equipo 2, estos se ven poco intuitivos, por lo cual entre los únicos dos compañeros del equipo 1 presentes se decide cambiar a una forma más sencilla utilizando unas etiquetas de inicio y fin, más otros detalles que se mencionan en el protocolo del equipo 1.

Además el equipo 1 presenta trabajar con puertos específicos.

Para el código de error (404) del equipo 2, se le amplia el mensaje que especifica el error como propuesta del equipo 1. También se agregan 3 tipos de código de error propuestos por el equipo 1.

En la sección **Comandos** del equipo 2 se mantiene el comando de LIST, pero el equipo 1 decide agregar el utilizar el GET para obtener figuras específicas y una posibilidad de ADD para que el cliente pueda agregar una figura que no existe.

Por último el equipo 1 añade unas indicaciones sobre **Administración de recursos.**