Armador de figuritas con piezas plásticas (Lego®)

Gabriel González Flores†, Sebastián Rodríguez Tencio†,
Juan Aguilar Torres†

Contributing authors: gabriel.gonzalezflores@ucr.ac.cr;
sebastian.rodrigueztencio@ucr.ac.cr; juan.aguilartorres@ucr.ac.cr;
†Estos autores contribuyeron igualmente a este trabajo.

Abstract

En este trabajo se implementa un programa que permite conectar a un cliente con un servidor de piezas, el cual le brinda la información necesaria para poder armar la figura. Y de igual manera, se trabaja en un posible protocolo para las siguientes etapas.

Keywords: lego, armador, piezas, protocolo

1 Introducción

En el ámbito de la construcción de figuras de Lego, es fundamental tener acceso a un inventario de piezas actualizado y confiable. Con este fin, en el presente trabajo, se ha diseñado un sistema de comunicación entre clientes y servidores que permitirá obtener un listado de piezas disponibles para construir una figura sencilla de Lego.

El sistema se estructura en dos etapas, en la primera los clientes interactuarán directamente con el servidor de piezas para obtener información sobre las figuras disponibles y las piezas necesarias para armarlas. En la segunda etapa, los clientes se comunicarán con servidores intermedios mediante el protocolo HTTP en un puerto TCP determinado para obtener la información.

El servidor de piezas es el encargado de proporcionar el inventario de piezas disponibles y las figuras que se pueden construir con ellas.

El objetivo de este sistema es permitir a los clientes obtener información detallada sobre las piezas necesarias para construir una figura específica, incluyendo los nombres de las piezas tal y como se identifican en el servidor. En caso de que no sea posible completar la figura, el servidor notificará al cliente con un error.

La primera entrega de esta evaluación consiste en la presentación de diversos elementos por parte de cada equipo de trabajo. En primer lugar, se realiza la entrega del cliente, es decir, la implementación del código que permitirá a los usuarios solicitar las piezas necesarias para armar una figura de Lego. Asimismo, se selecciona una figura sencilla y proporcionar el detalle de las piezas necesarias para su construcción. De esta manera, se podrán realizar pruebas y validar que el sistema está funcionando correctamente.

Una vez agregado el cliente, se continua con la segunda entrega, el servidor de piezas, para esto se implementa un servidor capaz de atender las solicitudes de los clientes por piezas para posteriormente armar figuras y devolver el listado de las piezas para armar la figura, estas solicitudes utilizan el protocolo HTTP.

Como tercera entrega se implementa un cliente en NachOS que solicita a un servidor de piezas una figura, por lo tanto se implementan syscalls relacionados a los sockets para realizar dicha tarea.

Además, se incluyen casos de prueba que permitan verificar el correcto funcionamiento del sistema en diferentes escenarios. Estos casos de prueba se publicarán en el repositorio y cubrirán diferentes situaciones que puedan presentarse.

En la última etapa de este proyecto, se implementa un componente fundamental conocido como el "servidor intermedio". Este servidor desempeña un papel crucial al actuar como un puente de comunicación entre el cliente y el servidor de piezas. Su principal objetivo es facilitar las conexiones TCP y UDP necesarias para establecer la comunicación bidireccional entre ambas partes.

El servidor intermedio se configura para recibir y transmitir datos tanto en el protocolo TCP como en el protocolo UDP. Esto permite que el cliente y el servidor de piezas se comuniquen de manera eficiente y confiable a través de diferentes canales de comunicación. Para lograr lo anterior se hace uso de un protocolo en común el cuál se especifíca en el siguiente enlace: Protocolo

(Adicionalmente, en un documento también se valorará el trabajo en equipo e individual, y se realizará una evaluación del desempeño de cada miembro del equipo.)

2 Objetivo general

Establecer una infraestructura de comunicación eficiente y confiable entre el cliente y el servidor de piezas, permitiendo la transferencia de datos y mensajes a través de conexiones TCP y UDP.

3 Objetivo específicos

Entre los objetivos específicos se plantean:

1. Diseñar un sistema de solicitud de figuras a través de una plataforma que garantice la seguridad de la transmisión de datos mediante el uso del protocolo HTTPS.

2. Desarrollar un programa capaz de devolver al usuario las piezas nesesarias para armar la figura solicitada por este.

4 Metodología

Se utilizó el lenguaje C++ en el editor de código fuente Visual Studio Code con la extensión de Live Share para lograr trabajar todos los autores del trabajo al mismo tiempo.

Como parte de los Sockets, es decir la petición que realiza un cliente a los servidores de piezas, se programaron con acceso seguro SSL en IPv4 con el fin de aumentar la seguridad entre la comunicación.

Por otra parte, con respecto al cliente NachOS, se programaron los system calls necesarios para llevar a cabo una conexión con un servidos, así como recibir y escribir diferentes datos.

Como parte del servidor de piezas, también se programaron con acceso seguro SSL en IPv4 con el fin de aumentar la seguridad entre la comunicación de client-server, además para la conexión client-server con un cliente NachOS se utiliza un server sin acceso seguro SSL.

Para la última etapa, se genera una solicitud en la cual se conecta el cliente, que solicita una figura específica, con el servidor intermedio que creamos en esta etapa, este después de recibir la solicitud y haberse conectado previamente a un servidor de piezas, solicita al respectivo servidor la pieza solicitada y por último envía la figura al cliente.

5 Resultados

Mediante el uso de sockets SSL se logró establecer apropiadamente la conexión estable al servidor de piezas. El cliente utiliza exitosamente las solicitudes HTTPS para traer, no sólo la lista de todas las figuras disponibles (que se le mostrarán al usuario), si no también traer las piezas requeridas para construir la figura que el usuario especificó en la línea de comandos, y mostrar en pantalla la cantidad de piezas necesarias para armar dicha figura.

Por otra parte, el servidor inicia con los encabezados necesarios para la programación de sockets, la manipulación de cadenas, el manejo de archivos y el subprocesamiento. Una directiva del preprocesador define el número de puerto que se utilizará para el servidor. Se define una función llamada "task" que acepta un puntero a un objeto Socket como argumento y es responsable de leer una cadena desde el socket, abrir y leer un archivo HTML almacenado en el servidor, en nuestro caso nos referimos a "Dalmata.html", enviar el contenido del archivo de al cliente y cerrar el socket. La función principal define un puntero a un subproceso y dos punteros a objetos Socket. Se crea un objeto Socket llamado "s1" con el argumento 's' para indicar que se utilizará como socket del servidor. Se llama al método Bind en el objeto de socket "s1" para enlazarlo al número de puerto especificado. Se llama al método Listen en el objeto de socket "s1" para establecer la cola de espera en 5 conexiones. Se llama al método SSLInitServer, cabe recalcar que estamos utilizando conexiones SSL para definir conexiones seguras, utilizamos un archivo ".pem" para esto. Se inicia un bucle infinito que espera conexiones de clientes utilizando el método Accept del objeto de socket "s1", para cada conexión de cliente, se crea un nuevo objeto Socket llamado "cliente" utilizando el método Accept del objeto de socket "s1". Se llama al método

SSLCreate en el objeto de socket "cliente" para crear la conexión SSL. Se crea un nuevo subproceso utilizando el constructor std::thread, pasando la función "task" y el objeto de socket "cliente" como argumentos. La función principal termina, dejando que los subprocesos creados manejan las conexiones de los clientes.

El protocolo propuesto es el propuesto por el equipo de trabajo, el cuál se detalla de una mejor manera en la sección 5.2 y posteriormente el protocolo grupal, que es el protocolo que es el protocolo seleccionado y que se utilizará en la sección 5.3.

Para la última etapa, se logró la comunicación cliente, servidor intermedio y servidor de piezas de manera local, sin embargo, en las pruebas realizadas en el laboratorio surgieron complicaciones para recibir los broadcast UDP, lo que complicó de manera significativa el avance en el proyecto, de esto se habla de una mejor manera en la sección 5.4.

5.1 Cliente NachOS

En cuanto al cliente de NachOS, se creó un servidor que acepta conexiones sin SSL, el cuál fue utilizado para conectarse con el cliente de NachOS, para implementar el cliente de NachOS se utilizaron las excepciones de sistema operativo para ejecutar este programa. Inicia mostrando con Write en el output que está iniciado y esperando el nombre de la figura. El programa entra en un bucle en el que lee un carácter a la vez del archivo de entrada estándar (ConsoleInput) y lo almacena en el arreglo buffer. El bucle continúa hasta que se lee un carácter de nueva línea. Esto se realiza con ayuda de la excepción Read con el fin de obtener el nombre de la figura.

A continuación, se llama la función Socket con los parámetros AF INET NachOS y SOCK STREAM NachOS. Esta función crea un socket para establecer una conexión de red. Luego, se llama a la función Connect para establecer una conexión en el socket recién creado. Se utiliza el valor de id del socket, la dirección IP "127.0.0.1" y el número de puerto 3141.

Después de establecer la conexión, se llama a la función Write para enviar el contenido del arreglo buffer al socket (Request de la figura), se llama a la función Read para recibir datos del socket y almacenarlos.

Finalmente, se llama a la función Write para escribir el contenido del arreglo recibido en la salida estándar (ConsoleOutput) y mostrar lo pedido por el cliente, en este caso las piezas así como su cantidad.

En resumen, el programa se encarga de establecer una conexión de red a un servidor. Envía un mensaje al servidor y espera la respuesta, que luego se muestra en la salida estándar, todo esto, utilizando las excepciones del sistema operativo NachOS.

5.2 Protocolo Propuesto

Este sistema de servidores está diseñado para trabajar en conjunto y proporcionar al cliente las piezas necesarias para armar una figura. Para lograr esto, cada servidor de piezas tiene un "struct" que incluye su dirección IP y un arreglo con las figuras que contiene. Cuando un servidor de piezas es iniciado, enviá un mensaje de "broadcast" a todos los servidores intermedios. Este mensaje incluye el "struct" del servidor de piezas, el cual contiene información importante para que los servidores intermedios

puedan localizar las piezas necesarias. Al recibir el mensaje de broadcast, cada servidor intermedio almacena la información en un mapa con la estructura "(figura - ip, ip, ...)". De esta manera, los servidores intermedios tienen acceso a las direcciones IP correspondientes para obtener las piezas necesarias de cada figura. Una vez que el servidor intermedio ha almacenado la información del servidor de piezas en el mapa, se mantiene en espera de conexiones de clientes. Cuando un cliente realiza una petición para obtener un conjunto de piezas para armar una figura, el servidor intermedio utiliza el mapa previamente creado para acceder a las direcciones IP correspondientes y obtener las piezas necesarias. Además, se ha previsto la situación en la que un servidor intermedio es iniciado después de algún servidor de piezas. En ese caso, el servidor intermedio enviará un mensaje de "broadcast" a todos los servidores de piezas para solicitar su información. Si algún servidor de piezas está en línea, responderá al mensaje con su "struct" correspondiente. Esta medida asegura que el servidor intermedio tenga acceso a la información necesaria para poder realizar su función en el sistema. De esta manera, se garan- tiza que todos los servidores, tanto de piezas como intermedios, puedan colaborar y proporcionar al cliente las piezas necesarias para armar las figuras. 3El protocolo utilizado para las conexiones entre servidores intermedios y de piezas es UDP, con los servidores intermedios escuchando en el puerto 4444 y los servidores de piezas en el puerto 5555. Por otro lado, la conexión del cliente con el servidor intermedio se realiza mediante el protocolo SSL IPV4, lo que garantiza una comunicación segura y confiable. En caso de que el servidor intermedio no logre obtener todas las piezas necesarias para armar la figura solicitada por el cliente, éste recibirá un mensaje de error. Esto significa que los servidores intermedios han sido incapaces de localizar alguna de las piezas necesarias para la figura solicitada, es decir llegó al final de las IPs disponibles en el mapa. En resumen, este sistema de servidores trabaja en conjunto para proporcionar al cliente las piezas necesarias para armar una figura. Los servidores de piezas envián información importante a través de mensajes de broadcast, que son almacenados por los servidores intermedios en un mapa. Luego, los servidores intermedios utilizan este mapa para obtener las piezas necesarias de las direcciones IP correspondientes. Todo el proceso se lleva a cabo mediante protocolos de comunicación seguros y confiables.

5.3 Protocolo Grupal

Este sistema de servidores está diseñado para trabajar en conjunto y proporcionar al cliente las piezas necesarias para armar una figura. Para lograr esto, cada servidor tiene que seguir un protocolo, el cuál se define a continuación. Para la realización de este protocolo se realizan varios tipos de mensajes los cuáles son: lego discover, lego ack, lego request, lego response, lego release.

LEGO DISCOVER: Es el mensaje que un servidor emite cuando es levantado por primera vez. Este permite a los servidores intermedios o de piezas identificar a este servidor para hacer solicitudes. Su modo de difusión es "broadcast". En el caso de que sea un servidor de piezas el que se levanta primero, el servidor intermedio es el que envía este mensaje y se identifica como un servidor intermedio por medio de una "I". Cuando el servidor de piezas recibe este mensaje crea un nuevo hilo de ejecución para atender a ese servidor. En el caso de que el servidor intermedio sea el que se levanta

primero, el servidor de piezas es el que debe enviar este mensaje y se identifica por medio de una "P". Si este fuera el caso, está seguido por nombres de figuras, pero si fuera un servidor intermedio, después del tipo de servidor no existe más información. Una vez el servidor intermedio recibe este mensaje, guarda la dirección del servidor de piezas para futuras solicitudes.

LEGO ACK: Es el mensaje que un servidor envía luego de identificar un LEGO DISCOVER. Sirve para indicarle al otro servidor que se quiere establecer una conexión con él. Su modo de difusión es "unicast". Los servidores de piezas que recibieron un LEGO DISCOVER envían este mensaje para indicarle a un servidor intermedio que su conexión fue aceptada. Una vez el servidor intermedio recibe el mensaje, guarda la dirección del servidor de piezas que devolvió este tipo de mensaje. De ser así, el servidor de piezas se identifica por medio de una "P" y el mensaje está seguido de los nombres de las figuras disponibles en ese servidor. Los servidores intermedios que recibieron un LEGO DISCOVER envían este mensaje para notificar al servidor de piezas que ya su dirección fue guardada para la comunicación con él. Este tipo de mensaje se identifica por medio de una "I" y no está seguido por nombres de figuras. Una vez el servidor de piezas recibe este mensaje debe crear un nuevo hilo de ejecución para atender futuras solicitudes.

LEGO REQUEST: Es el mensaje que envía un servidor intermedio para realizar una petición de una figura específica. Este debe ser recibido por un servidor de piezas. Su modo de difusión es "unicast".

LEGO RESPONSE: Es el mensaje que envía a un servidor de piezas con la información de la figura solicitada por el servidor intermedio. Este debe ser recibido por un servidor intermedio. Su modo de difusión es "unicast".

LEGO RELEASE: Es el mensaje que se envía cuando un servidor finaliza de dar su servicio, ya sea de piezas o intermedio. Puede ser recibido por cualquier tipo de servidor. Su modo de difusión es "broadcast", pues se debe avisar a todos los servidores que el servidor ya no estará en funcionamiento. Si fuera un servidor intermedio el que recibe este mensaje, debe eliminar al servidor de piezas de la estructura de datos para hacer consultas de figuras de lego. Y si fuera un servidor de piezas el que recibe este mensaje, debe eliminar al servidor intermedio de los clientes a los que tiene que atender. La siguiente figura explica el flujo de los tipos de mensajes entre servidores descritos anteriormente. Nótese que los mensajes LEGO DISCOVER, LEGO ACK y LEGO RELEASE, apuntan a los servidores intermedios y a los de piezas. Esto se debe a que son bidireccionales. Es decir, cualquiera de los dos puede enviar o recibir este tipo de mensaje. Para los demás -LEGO REQUEST y LEGO RESPONSE el mensaje solo tiene una posible dirección de envío.

5.4 Servidor Intermedio

En la última etapa del proyecto, se introduce la implementación del servidor intermedio, que juega un papel fundamental en la comunicación entre el cliente y el servidor de piezas. Para lograr esto, se requiere realizar ciertos cambios tanto en el cliente como en el servidor de piezas, con el fin de adaptarse al protocolo propuesto y asegurar el correcto funcionamiento de los tres componentes.

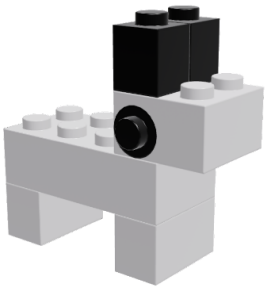
En primer lugar, es necesario ajustar el cliente para que pueda establecer una conexión con el servidor intermedio en lugar de conectarse directamente al servidor de piezas. Esto implica modificar las configuraciones de red y los parámetros de conexión en el cliente, de manera que se direccionen hacia el servidor intermedio en lugar de dirigirse directamente al servidor de piezas. Además, se deben implementar las interacciones y protocolos de comunicación específicos para trabajar con el servidor intermedio.

Por otro lado, en el servidor de piezas, se requiere realizar modificaciones para permitir la comunicación con el servidor intermedio. Esto implica ajustar las configuraciones de red y los mecanismos de escucha del servidor de piezas, de manera que pueda recibir solicitudes y enviar respuestas al servidor intermedio en el formato adecuado establecido por el protocolo.

En conjunto, estos cambios en el cliente y en el servidor de piezas permiten establecer una comunicación eficiente y segura con el servidor intermedio. El servidor intermedio, a su vez, actúa como un enlace entre el cliente y el servidor de piezas, encargándose de enrutar las solicitudes del cliente al servidor de piezas correspondiente y enviar las respuestas del servidor de piezas de regreso al cliente.

En esta etapa surgieron dos problemas principales. En primer lugar, no se pudo implementar correctamente el "lego release", lo cual resulta en un manejo deficiente de los recursos y puede generar problemas adicionales. Por otro lado, durante las pruebas en el laboratorio se constató una conexión adecuada entre las máquinas y los switches, y se logró enviar el broadcast de una máquina a otra, esto se verificó utilizando Wireshark. Sin embargo, tanto el servidor intermedio como el servidor de piezas no lograron recibir este broadcast, lo que plantea un desafío adicional en el funcionamiento del sistema.

5.5 Figura



La figura propuesta es la de un dalmata, para la realización de dicha figura solo se requiere de 6 piezas como se detalla acontinuación.

Dalmata:
2 brick 1x2 white
1 brick 2x4 white

1 brick 2x2 white
2 brick 1x1 black
Total de piezas: 6

6 Conclusión

En conclusión, la implementación del servidor intermedio en la última etapa del proyecto ha sido fundamental para lograr la comunicación efectiva entre el cliente y el servidor de piezas. A través de ajustes y modificaciones en el cliente y en el servidor de piezas, se ha logrado adaptarlos al protocolo propuesto y garantizar su correcto funcionamiento en conjunto con el servidor intermedio.

El servidor intermedio actúa como un puente entre el cliente y el servidor de piezas, permitiendo enrutar las solicitudes del cliente hacia el servidor de piezas correspondiente y enviar las respuestas del servidor de piezas de regreso al cliente. Además, se encarga de mantener un "mapa" o "router" de las figuras proporcionadas por los servidores de piezas conectados, lo que le permite manejar de manera eficiente las solicitudes de los clientes y entregar las piezas solicitadas.

En conclusión, la última etapa del proyecto presentó desafíos significativos en la implementación del servidor intermedio. A pesar de los esfuerzos realizados, no fue posible llevar a cabo la implementación del "lego release" como se mencionó anteriormente. Además, durante las pruebas en el laboratorio se detectó una dificultad en la recepción del broadcast por parte del servidor intermedio y el servidor de piezas, a pesar de contar con una conexión estable entre las máquinas y switches. Estos problemas requieren una mayor investigación y soluciones para garantizar un correcto funcionamiento del sistema en futuras etapas o proyectos similares.

7 Referencias

Silberchatz, P. Galvin y G. Gadge, "Operating Systems Concepts", 9na edición, John Wiley and Sons, 2013.