Laboratorio 1: Redes de computadores

Profesor: Viktor Tapia
Ayudantes de Tarea: Javiera Cárdenas

Agosto 2023

1 Objetivos del laboratorio

- Aprender a utilizar sockets UDP y TCP en Python y Go.
- Conocer y aplicar la estructura cliente-servidor.
- Aprender a realizar el análisis del tráfico de red a partir de la herramienta Wireshark.
- Familiarizarse con protocolos altamente usados en Internet.

2 Introducción

Un socket corresponde a la interfaz existente entre las capas de aplicación y transporte, estos permiten establecer una conexión entre aplicaciones, para que, de esta forma, pueda ser llevado a cabo el intercambio de mensajes entre las mismas. Estas aplicaciones pueden estar ejecutándose tanto en una misma maquina o en dos, en donde esta interacción permite dar lugar a la estructura cliente-servidor. Es en este sentido donde, tanto Python como Go entregan la posibilidad de facilitar el trabajo de establecer una conexión a partir del uso de sockets (de dominio Unix en este caso). Por un lado, Python posee una libreria llamada **socket** y por el otro, Go con su libreria **net**. Como fue adelantado previamente, este laboratorio pretende evaluar una estructura clásica en el ámbito de redes: la **cliente-servidor**, donde la carga de trabajo se encuentra distribuida entre los servidores (proveedores del servicio) y los clientes (consumidores del servicio provisto). Ademas de aquello, se agrega la utilización de la herramienta Wireshark. Este software es el encargado de realizar el análisis del trafico de red en tiempo real, teniendo como agregado, la posibilidad de facilitar la identificación de los protocolos antes mencionados **TCP** y **UDP**.

3 Tarea

3.1 Enunciado

El Conecta 4 es un juego clásico de la niñez de la gran mayoría, es un juego por turnos en donde se solicita que se conecten 4 fichas iguales en una línea recta continua que puede ser vertical, horizontal o diagonal en cualquier sentido, el primero en conectar 4 gana, Es en este contexto donde, para una primera parte, se les encomendará llevar a cabo la creación de este juego, en un contexto de Jugador versus Computadora, para la facilidad de todo se les recomienda tratar el tablero como un 6x6 y las columnas del tablero como pilas que se van llenando hacia arriba. Para llevar a cabo esta tarea, se les recomienda revisar el diagrama que se encuentra más abajo.
Como fue mencionado, deberán construir una arquitectura cliente-servidor para este juego, el cual, constará de tres nodos:

- Cliente
- Servidor Intermediario
- Servidor Conecta4

3.2 Cliente

Este proceso cumple el rol de jugador. El cliente debe satisfacer las siguientes tareas:

- Establecer una conexión **TCP** con el servidor intermediario.
- No debe conectarse directamente con el Servidor Conecta4.
- Debe mostrar por consola los resultados de cada uno de los turnos y luego el resultado final, indicando si ganó la máquina, el jugador o hubo un empate, para, por ultimo, solicitar una nueva partida o directamente terminar todo.
- El código de este programa debe estar escrito en Python.

3.2.1 Estructura de los resultados

- - - - - - - - Bienvenido al Juego - - - - - - - -
- Seleccione una opción
1-Jugar
2-Salir
>>1
respuesta de disponibilidad: OK
- - - - - - - - Comienza el Juego - - - - - - - -
El juego debe ser capaz de mostrar el tablero mostrando visualmente las posiciones disponibles y las ocupadas tal como se ejemplificaría en el siguiente ejemplo

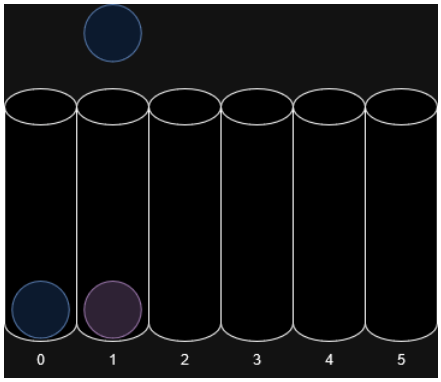

Figure 2: Ejemplo Visualización de las pilas

3.3 Servidor - Intermediario

Este nodo cumple el rol de comunicar el Cliente con el Servidor Conecta4. Para esto, se deben satisfacer las siguientes tareas:

- Mantener una conexión **TCP** con el Cliente.
- Conectarse, cuando sea requerido, con el Servidor Conecta4 mediante una conexión UDP.
- Responder al Cliente con el mensaje que recibe del Servidor Conecta4.
- Este debe procesar el turno revisando posibles ganadores y enviar el resultado junto con la jugada al Cliente.
- Alertar al Servidor Conecta4 del término del juego para que, este pueda terminar su ejecución
- Debe terminar su ejecución cuando el Cliente le indique el termino del juego (No sin antes igualmente notificar al Servidor Conecta4).
- El código de este programa debe estar en Python.
- Informar sobre el intercambio de mensajes entre los demás nodos.

3.4 Servidor Conecta4

Este nodo cumple el rol de BOT en la lógica del Jugador versus Computadora. Por lo tanto, este nodo juega contra el Cliente ejecutando jugadas aleatorias hasta que se le indique el final del juego. Para esto, se deben satisfacer las siguientes tareas

- Abrir una conexión **UDP** para comunicarse con el Servidor Intermediario.
- Abrir otra conexión **UDP** en un puerto aleatorio (entre 8000 y 65.535) cada vez que se pida una jugada.
- Enviar mensajes al Servidor Intermediario y también recibir mensajes del mismo.
- Debe terminar su ejecución cuando se lo indique el Servidor Intermediario.
- El código de este programa debe estar escrito en Go.
- Informar de intercambios de mensajes dentro de su consola, junto con la apertura y cierre de puertos.

3.5 Topología del sistema

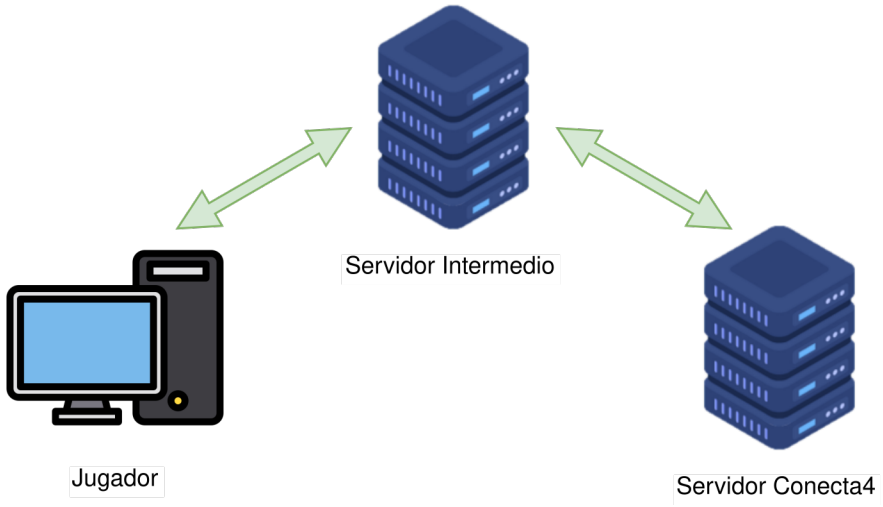

Figure 1: Topología de lo que se busca implementar: Esta puede ser hallada comúnmente en configuraciones en las que se tiene un Proxy en la ubicación del servidor-Intermediario, o igualmente un balanceador de carga.

3.6 Diagrama

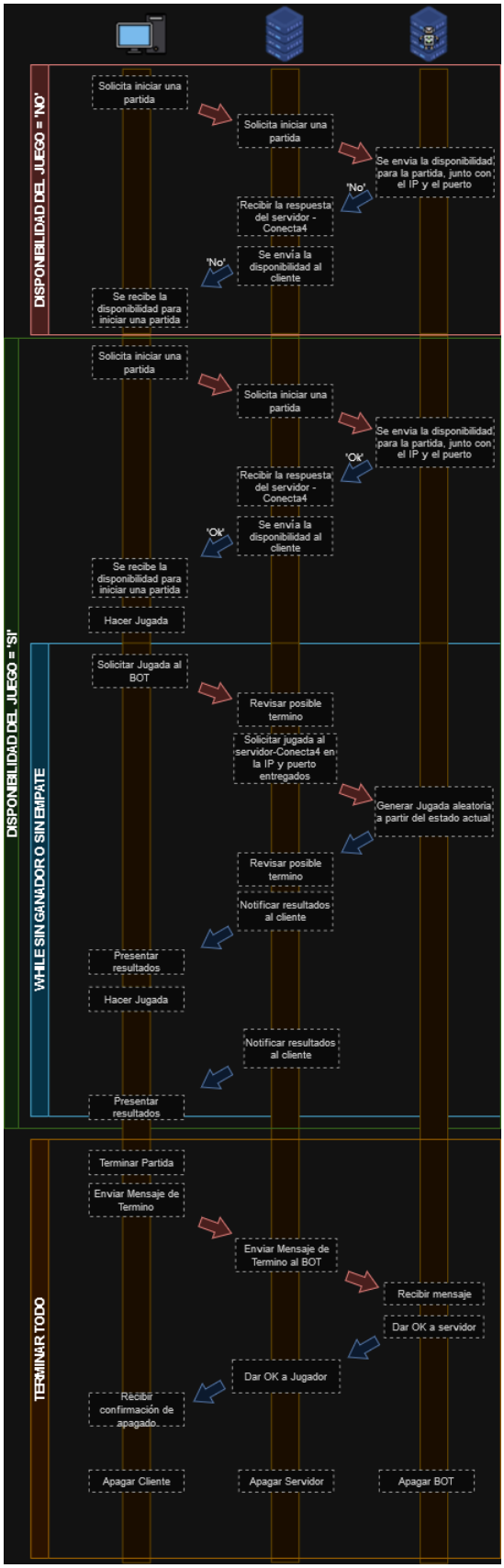

Figure 2: Diagrama del proceso del Laboratorio

4 Análisis de tráfico

Empleando la herramienta antes mencionada (Wireshark debería analizar junto con su compañero los paquetes asociados a la aplicación desarrollada para luego responder las siguientes preguntas en un archivo pdf adjunto a la entrega en un zip.

1. Si se analiza el número de los mensajes enviados dentro de la aplicación. ¿Cuántos son los que logra detectar Wireshark?. Y comparando en base al código, ¿Es la misma cantidad?, si no lo es, ¿A qué se debería?

2. ¿Cuál es el protocolo que se debiese ver a la hora de revisar el intercambio de mensajes en Wireshark? ¿Y cuáles encontró?

3. ¿El contenido de los mensajes dentro de Wireshark son legibles?, ¿Por Qué si? o ¿Por Qué no?

5 Reglas de entrega

- La tarea se realiza de forma individual.
- La fecha de entrega es el día **07 de Octubre hasta las 23:55**.
- El código debe correr en Python 3.7. Solo está permitido hacer uso de la librería socket. Para las conexiones del código en Go, se debe utilizar la librería net.
- La entrega debe realizarse a través de Classroom, en un archivo comprimido .zip, indicando el número de Laboratorio y su nombre en el siguiente formato: L1-Nombre-Apellido.zip, Ejemplo: L1-Javiera-Cárdenas.zip.
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución de la entrega. Teniendo al menos un archivo para el Cliente, Servidor Intermediario y Servidor Conecta4. Con el código bien identado, comentado, sin warnings ni errores. Además del archivo .pdf con las respuestas respectivas a las preguntas.
- Las preguntas deben ser hechas por el foro de dudas y/o en la sesión de consultas.
- Debe entregar un **README** con sus nombres y las instrucciones necesarias para ejecutar correctamente el laboratorio (ADVERTENCIA: Si no se entrega dicha información, se colocaría una nota mínima a la entrega y posteriormente se tendría que coordinar una sesión de apelación.)
- Cada día o fracción de atraso se les descontara 20 puntos a la nota máxima posible, hasta un máximo de 1 día completo de atraso, pasado este plazo los laboratorios serán evaluados con nota 0.
- Cualquier sospecha de copia será notificada debidamente a su profesor y evaluada con nota mínima. **Siendo tomado en cuenta también cualquier copia directa de algún sitio web o foro.**