Laboratorio 1: Redes de Computadores

Profesor: Viktor Tapia
Ayudante de Laboratorio: Luciano Yevenes

22 de abril de 2025

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 an´alisis del tr´afico de red a partir de la herramienta Wireshark.
- Familiarizarse con protocolos altamente usados en Internet.

2. Introducción

Según lo visto en clases. 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 librería 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 juego de *"Adivina el Número en Red"* es una variante interactiva y distribuida del clásico juego de adivinanzas. En esta actividad, se le pide al jugador humano descubrir un número secreto (entre 1 y 100) generado por la computadora, la cual está distribuida en una arquitectura cliente-servidor con tres componentes.

El jugador tiene un máximo de 7 intentos para adivinar el número. Puede proponer valores directamente (por ejemplo, "42") y recibirá pistas tales como "El número es mayor", "El número es menor", o "¡Has acertado!" hasta que se termine el juego. El objetivo principal de este laboratorio es implementar esta lógica bajo una arquitectura distribuida, que incluirá tres nodos principales:

- Cliente
- Servidor Intermediario
- Servidor del Juego (Servidor Adivina)

3.2. Cliente

El cliente representa al jugador humano. Sus responsabilidades incluyen:

- Establecer una conexión TCP con el servidor intermediario.
- No establecer una conexión directa con el Servidor Adivina.
- Mostrar en la consola las respuestas del servidor, junto con la cantidad de intentos restantes.
- Ofrecer la opción de iniciar una nueva partida o finalizar el juego.
- El código del cliente debe estar escrito en **Python**.

Estructura esperada de la salida

```
----- Bienvenido a Adivina el Número -----
Seleccione una opción:
1 - Jugar
2 - Salir
>> 1
Conectado al servidor: OK
Intento 1 de 7: Ingrese un número:
>> 42
Respuesta: El número es menor.

Intento 2 de 7: Ingrese un número:
>> 23
Respuesta: El número es mayor.

Intento 3 de 7: Ingrese un número:
>> 30
Respuesta: ¡Has acertado!
¿Desea jugar de nuevo? (s/n)
>> n
```

3.3. Servidor Intermediario

El servidor intermediario actúa como el puente de comunicación entre el cliente y el Servidor Adivina. Sus responsabilidades incluyen:

- Mantener una conexión TCP con el cliente.
- Establecer una conexión UDP con el Servidor Adivina cuando sea necesario.
- Enviar los intentos del cliente y reenviar las respuestas del Servidor Adivina.
- Llevar la cuenta de los intentos y determinar si el juego ha terminado.

- Notificar al Servidor Adivina cuando el juego ha terminado para que este finalice su ejecución.
- Finalizar su ejecución cuando el cliente lo indique, asegurándose de notificar al Servidor Adivina.
- El código del servidor intermediario debe estar escrito en **Python**.
- Informar en consola los intercambios de mensajes entre cliente y servidor del juego.

3.4. Servidor Adivina

El Servidor Adivina es el oponente del jugador, representado por un bot que genera un número aleatorio y responde a los intentos. Sus responsabilidades incluyen:

- Abrir una conexión UDP para comunicarse con el Servidor Intermediario.
- Crear una nueva conexión UDP en un puerto aleatorio (entre 8000 y 65535) cada vez que se le solicita realizar una jugada.
- Enviar y recibir mensajes del Servidor Intermediario.
- Terminar su ejecución cuando reciba la indicación del Servidor Intermediario.
- El código del Servidor Adivina debe estar escrito en **Go**.
- Imprimir en consola los intercambios de mensajes y la apertura y cierre de puertos utilizados.

3.5. Topología del sistema

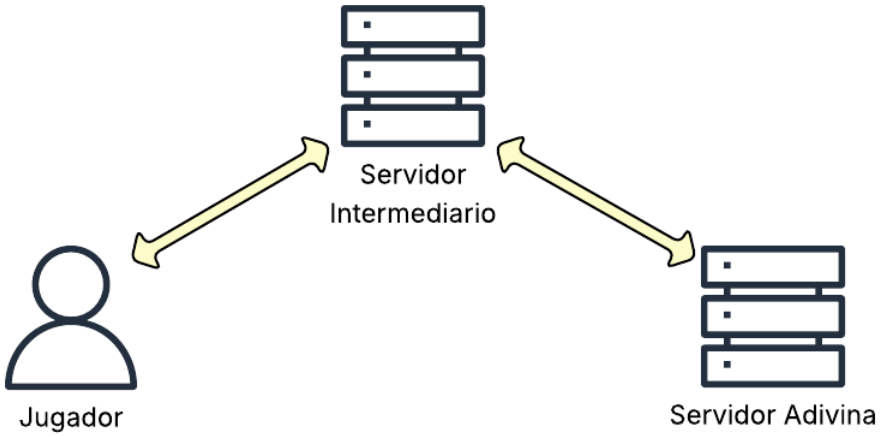

Figura 1: Topología de la arquitectura que se implementará. Esta configuración es común en sistemas que utilizan un proxy o un balanceador de carga en la posición del servidor intermediario.

3.6. Diagrama

(revisar en la página final)

4. Análisis de tráfico

Utilizando la herramienta **Wireshark**, analicen el tráfico de red generado por su aplicación y respondan las siguientes preguntas en un archivo PDF, el cual deberá ser incluido en la entrega junto con el código, empaquetado en un archivo comprimido (`.zip`):

1. ¿Cuántos mensajes logra detectar Wireshark en comparación con los enviados por la aplicación? Si hay una discrepancia, ¿a qué se debe?
2. ¿Qué protocolo(s) se observan en el intercambio de mensajes según Wireshark?
3. ¿Es legible el contenido de los mensajes capturados en Wireshark? Justifique su respuesta.

5. Consideraciones Generales

- La tarea se realiza en parejas.
- La fecha de entrega es el día **x de x de 2025 hasta las 23:59**.
- Solo está permitido hacer uso de la librería `socket` para las conexiones en Python. Para las conexiones del código en Go, se debe utilizar la librería `net`.
- La entrega debe realizarse **al correo** `buzontareasluciano@gmail.com`, en un archivo comprimido `.zip`, indicando el número de Laboratorio y su nombre en el siguiente formato: `L1-Nombre_Apellido.zip`. Ejemplo: `L1-Luciano_Yevenes.zip`.
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución de la entrega, incluyendo al menos un archivo para el Cliente, el Servidor Intermediario y el Servidor Adivina. El código debe estar bien indentado, comentado, y libre de *warnings* o errores. Además, se debe incluir un archivo **.pdf** con las respuestas respectivas a las preguntas.
- Las preguntas deben ser realizadas por **Discord**.
- Debe entregar un `README` con los nombres de los integrantes, además de las instrucciones necesarias para ejecutar correctamente el laboratorio.
- Cada hora de atraso penalizará el laboratorio, descontando **10 puntos**.
- Cualquier sospecha de copia será notificada debidamente al profesor y evaluada con nota mínima. Se tomará en cuenta también cualquier copia directa de algún sitio web o foro.

Discord de la Asignatura: https://discord.gg/592v5TBJU9

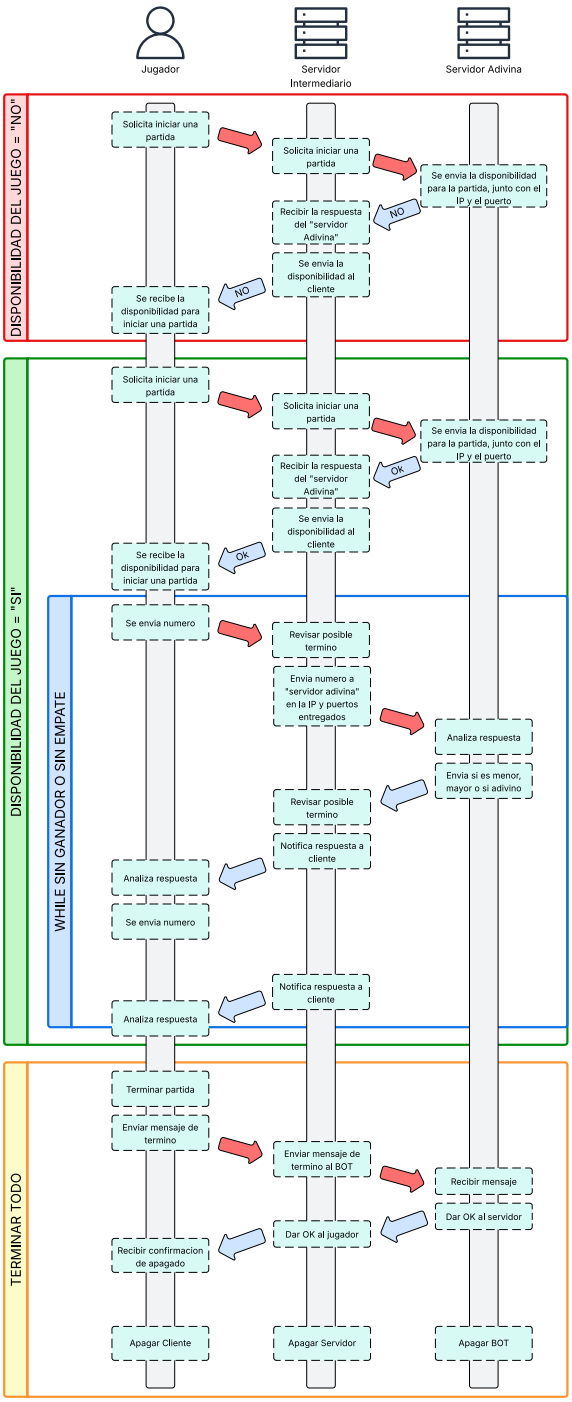

Figura 2: Topología de la arquitectura que se implementará. Esta configuración es común en sistemas que utilizan un proxy o un balanceador de carga en la posición del servidor intermediario.