

Laboratorio 1: Redes de Computadores

Profesores: Erika Rosas & Jorge Díaz

Ayudantes: Iñaki Oyarzun M. & Sebastián Martínez C.

Abril 2022

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

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 máquina o en dos, en dónde 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 librería **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).

Además de aquello, se agrega la utilización de la herramienta **Wireshark**. Este software es el encargado de realizar el análisis del tráfico 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 tres en línea, o mejor conocido como *gato*, el juego que alguna vez todos hemos visto en algún punto ya sea de la etapa escolar o universitaria, por lo que, se asume que tienen conocimiento de sus reglas. 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 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 Gato

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 Gato
- 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 último, 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-----
. 0| 1 | 2
0 |  | 
---+---+---
1 |  | 
---+---+---
2 |  | 
Ingrese su jugada (x,y):
>> 0,0
=====
. 0| 1 | 2
0 o|  | 
---+---+---
1 x|  | 
---+---+---
2 |  | 
Ingrese su jugada (x,y):
>>
```

3.3 Servidor - Intermediario

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

- Mantener una conexión **TCP** con el Cliente.

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

3.4 Servidor Gato

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

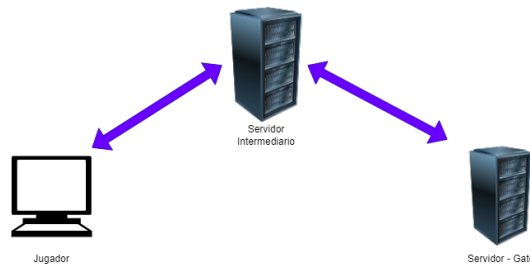


Figure 1: Topología de lo que se busca implementar: Esta puede ser hallada comúnmente en configuraciones en las que se tiene un Proxy en la ubicación del servidor-Intermediario, o igualmente un balanceador de carga por ejemplo (Se profundizará más adelante en Sistemas Distribuidos)

3.6 Diagrama

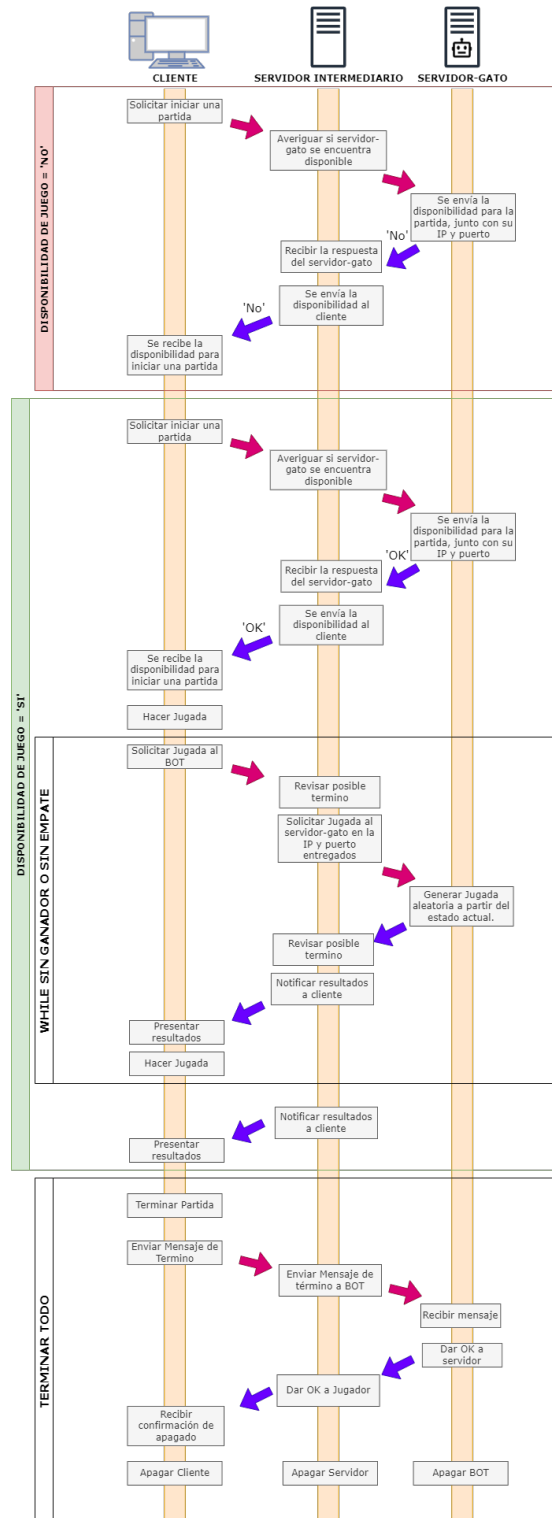


Figure 2: Diagrama del proceso del Laboratorio

4 Análisis de tráfico

Empleando la herramienta antes mencionada (**Wireshark** deberá 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 sí? o ¿Por Qué no?

5 Reglas de entrega

- La tarea se realiza en grupos de 2 personas. Estos debiesen estar inscritos dentro de la pestaña de **laboratorio** dentro de AULA.
- La fecha de entrega es el día **11 de Abril de 2022 hasta las 23:59**
- 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 Aula, en un archivo comprimido **.zip**, indicando el número de Laboratorio y grupo en el siguiente formato: **L1-Grupo[Nº Grupo].zip**, Ejemplo: **L1-Grupo01.zip**.
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución de la entrega. Teniendo al menos un archivo para el Cliente, Servidor Intemediario y Servidor Gato. Con el código bien indentado, comentado, sin warnings ni errores. Además del archivo **.pdf** con las respuestas respectivas a las preguntas.
- Debe entregar un **README** con nombre y rol de cada integrante del grupo, además de las instrucciones necesarias para ejecutar correctamente el laboratorio (**ADVERTENCIA:** Si no se entrega dicha información, se colocará un cero a la entrega y posteriormente se tendrá que coordinar una sesión de apelación.)
- Cada hora de retraso penalizará el laboratorio, descontando 10 pts.
- Cualquier sospecha de copia será notificada debidamente a su profesor y evaluada con nota 0. **Siendo tomado en cuenta también cualquier copia directa de algún sitio web o foro**. Se tendrá un software a mano para realizar dichas comparaciones.