Laboratorio 1: Redes de Computadores

Profesor: Jorge Díaz
Ayudantes: Juan Cucurella, Nicolás Rodríguez

Agosto 2025

1 Objetivos del laboratorio:

- Aprender a utilizar sockets UDP y TCP
- Familiarizarse con el protocolo HTTP
- Entender como se comportan las arquitecturas cliente-servidor
- Entender de manera práctica el comportamiento de la capa de aplicación

2 Introducción

Un socket corresponde a la interfaz existente entre las capas de aplicación y transporte. Este permite establecer una conexión entre aplicaciones para que se pueda llevar a cabo el intercambio de mensajes entre estas. Dichas aplicaciones pueden estar ejecutándose tanto en una misma máquina como en dos diferentes, dando lugar a la estructura cliente-servidor.

Los sockets pueden operar sobre diferentes protocolos de transporte, siendo los más utilizados TCP (Transmission Control Protocol) y UDP (User Datagram Protocol). Mientras que TCP garantiza una comunicación confiable y ordenada mediante el establecimiento de una conexión, UDP permite una transmisión más rápida, pero sin garantizar la entrega de los datos.

Gracias a su flexibilidad, los sockets son la base de múltiples aplicaciones de red, como navegadores web, sistemas de mensajería, servicios de streaming y videojuegos en línea, permitiendo una comunicación eficiente y escalable entre procesos y dispositivos en una red local o en Internet.

3 Laboratorio 1

3.1 Enunciado

Un juego popular dentro de grupos de personas es *la frase interminable* o *teléfono roto*, este consiste en armar una frase, que idealmente sea coherente, donde cada jugador agrega una nueva palabra a las mencionadas con anterioridad y pasa el mensaje al siguiente jugador. Existen muchas variaciones de este juego para hacerlo más difícil y que sea más desafiante.
Es por esto que en su búsqueda implacable por estar al día con las tendencias, una empresa anónima le pide a usted y a su grupo de trabajo recrear este juego a través del uso de nuevas tecnologías y protocolos de red.

3.2 Explicación general

En este laboratorio, deberán programar clientes capaces de conectarse entre sí con sockets TCP, UDP y un servidor HTTP simple para introducir los conceptos respecto a la capa de aplicación, realizando una interacción entre cada uno de los componentes.

Como esquema general, deben conectar los clientes y servidores de tal forma que se replique el comportamiento del juego de *la frase interminable*. Inicialmente, se enviará un mensaje a un servidor TCP. Dicho servidor deberá tomar el mensaje y agregar información a través de la entrada estándar. Posteriormente, lo enviará vía UDP a un segundo servidor, el cual se encargará de repetir el proceso, fabricar y enviar una solicitud HTTP al servidor web con el mensaje nuevo. El servidor HTTP deberá reconstruir el mensaje hasta dicho punto, agregar nueva información y generar un mensaje a través de TCP para el servidor original, repitiendo el proceso indefinidamente hasta contar con un mensaje lo suficientemente largo y almacenarlo en un archivo de texto. Todo esto seguirá una interacción similar al siguiente diagrama:

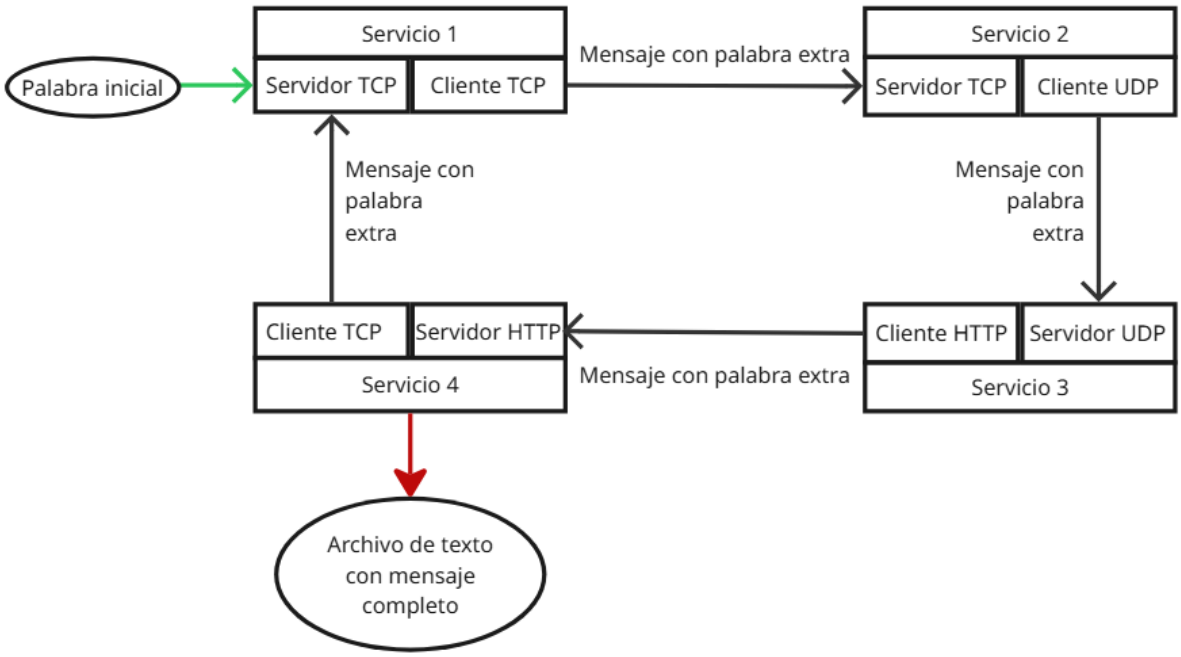

Figure 1: Ejemplo de la interacción solicitada

Nota: Cuentan con la libertad de entregar 3 clientes en archivos por separado, 2 archivos con el cliente TCP-UDP y el cliente HTTP por separado o un solo archivo que cuente con los 3 clientes implementados.

3.3 Pre-laboratorio

Antes de comenzar a trabajar en su entrega de laboratorio, es necesario realizar las siguientes tareas:

- Leer atentamente y considerar las reglas del presente laboratorio (*Consulte sección "Reglas de Laboratorio"*)
- Instalar Python 3.10 o superior en su sistema
- Instalar las librerías clave para trabajar el laboratorio (*Consulte sección "Librerías permitidas"*)

3.4 Experiencia a realizar

Tomando en cuenta la explicación general otorgada previamente, su implementación de los componentes debe cumplir con los siguientes pasos:

1. Al iniciar la interacción, el servicio 1 debe solicitar (a través de entrada estándar) un largo mínimo del mensaje final y una palabra con la cual iniciar la interacción.
2. Una vez ingresados dichos datos, el servicio debe componer un string con el formato "**[Timestamp]-[LargoMínimo]-[LargoActual]-[Mensaje]**" y enviarlo al servicio 2 a través de conexión TCP.
3. El servicio 2 debe recibir el string y solicitar una nueva palabra por entrada estándar, una vez realizado, debe agregar la palabra nueva al mensaje recibido y actualizar el largo actual del mensaje en el string.
4. Una vez preparado el nuevo string, se envía al servicio 3 a través de UDP.
5. El servicio 3 recibe el string actualizado y repite el proceso, agregando una nueva palabra y actualizando el largo del mensaje. Una vez actualizada la información, el servicio 3 debe enviar una solicitud HTTP al servicio 4 con el string actualizado en el cuerpo de la solicitud.
   Nota: el servicio 3 debe enviar la solicitud HTTP utilizando una conexión TCP. No se admite el uso de la librería ***requests***
6. El servicio 4 recibe el string nuevamente actualizado a través de un servidor HTTP simple, de la cual extrae el string y separa sus partes fundamentales.
   Nota: Tienen la libertad para implementar el servidor HTTP utilizando ***http.server*** *para facilitar el trabajo en esta sección.*
7. En base a la información extraída del mensaje, el servidor configura el largo mínimo del mensaje y compara si el largo actual supera al mínimo.
   Si el largo actual supera al mínimo, el servicio guarda el mensaje con timestamp en un archivo de texto y comienza la finalización de la interacción.
   En caso de que el largo no supere al mínimo, el servicio 4 solicita una nueva palabra al usuario a través de consola y la concatena al mensaje, de la misma forma que el resto de servicios. Si esto ocurre, se envía el mensaje al servicio 1 a través de TCP y se repite la cadena ya explicada.
8. Para el término de la interacción, debe implementar un orden explícito para la propagación de las señales de finalización.
9. El servicio 4 debe propagar una señal de finalización al siguiente servicio en la cadena para, posteriormente, cerrar las conexiones activas y terminar la ejecución.
10. Los servicios siguientes deben repetir la acción, propagar la señal de finalización al siguiente servicio en el orden establecido por su grupo, cerrar conexiones activas y terminar el proceso.
11. El último servicio en la cadena debe recibir la señal de finalización y terminar su ejecución directamente.

3.4.1 Requerimientos

En base al procedimiento enumerado anteriormente, se detallan a continuación los requerimientos mínimos para los componentes a implementar, con el propósito de lograr la interacción comentada:

- Todos los clientes deben ser capaces de abrir conexiones hacia su servidor correspondiente de manera correcta.
  - El cliente del servicio 1 debe funcionar en base al protocolo TCP y conectarse correctamente al servicio 2 usando dicho protocolo.
  - El cliente del servicio 2 debe funcionar en base al protocolo UDP y conectarse correctamente al servicio 3 usando dicho protocolo.
  - El cliente del servicio 3 debe funcionar en base al protocolo TCP para realizar solicitudes HTTP, enviando dichas solicitudes de manera correcta al servicio 4.
  - El cliente del servicio 4 debe funcionar en base al protocolo TCP y conectarse correctamente al servicio 1 usando dicho protocolo.
- Todos los servicios que requieran entradas de datos deben hacerlo a través de entrada estándar.
- Todos los mensajes enviados hacia los servicios deben ser en formato UTF-8 para mantener compatibilidad, cumpliendo con la lógica detallada en secciones anteriores.
- Los clientes deben ser capaces de recibir, leer e imprimir las respuestas de los servicios a los cuales se conectan.
- Los clientes deben enviar la información a los servicios de la manera más limpia posible, es decir, debe asegurarse que los mensajes que envíe al servidor no contengan espacios de más, caracteres no esperados, entre otros.
- Los servidores deben ser capaces de recibir conexiones entrantes, además de interpretar y tratar los mensajes recibidos según la lógica especificada.
- Todas las conexiones inactivas deben ser cerradas oportunamente, es decir, se debe evitar reabrir conexiones ya existentes.
  Nota: Para evitar esto, pueden cerrar las conexiones luego de una interacción y abrirlas solo en caso de necesidad, o mantener una lógica para reutilizar conexiones.
- En caso de recibir una señal o mensaje de finalización, si el servicio no es el último en la cadena de término, se debe enviar una señal de finalización a otro servicio siguiente, posteriormente, para todos los casos, se deben cerrar todas las conexiones y finalizar la ejecución del programa.
- Todos los mensajes deben seguir el formato especificado "**[Timestamp]-[LargoMínimo]-[LargoActual]-[Mensaje]**", incluidas las señales de finalización, en cuyo caso el formato corresponderá a "**[Timestamp]-[MensajeParaFinalizar]**"
- El cliente HTTP solicitado para el servicio 3 debe realizar las solicitudes utilizando un socket TCP.
- El servidor HTTP debe ser capaz de extraer el mensaje desde el cuerpo de la solicitud.
  Nota: puede establecer el formato del cuerpo de la solicitud a libertad, es decir, texto plano, JSON o estructuras similares, no obstante, si desea utilizar alguna librería no listada en el presente enunciado debe notificarlo, con el objetivo de autorizar o denegar su uso.
- El servicio 4, en alguna parte de su lógica (servidor o cliente), debe ser capaz de analizar si el mensaje cumple con el largo mínimo configurado en la interacción, si el mensaje alcanza el largo mínimo, debe ser capaz de iniciar la cadena de finalización de servicios, si el largo no es alcanzado, debe ser capaz de enviar mensajes al servicio 1 para continuar y realizar otro ciclo de la cadena.
  Nota: Existen múltiples formas de llevar esto a cabo. Una de ellas consiste en enviar un mensaje en cadena, donde un servidor envía la solicitud de finalización al siguiente servidor. Luego, este

último repite el proceso hacia el siguiente salto antes de finalizar su ejecución. No obstante, si desea experimentar y realizar este proceso de forma más creativa, es libre de implementar lo que necesite, siempre y cuando utilice únicamente los recursos y librerías permitidas.

Nota: Puede reciclar código entre los componentes que debe implementar si lo necesita, siempre y cuando se cumpla con los requerimientos presentados

4 Librerías permitidas

Se detallan las librerías permitidas para la presente experiencia de laboratorio, en caso de requerir el uso de otra librería que no esté presente en este listado (para cualquier tipo de funcionalidad que desee implementar), debe dejar una consulta al respecto en Aula para analizar y autorizar su uso.

- **Asyncio**
- **Socket**
- **re** (expresiones regulares, para patrones de texto)
- **os**
- **http.server** (para el servidor HTTP solicitado)
- **Datetime** (para generar timestamps)
- **JSON** (Solo en caso de que lo necesite para interpretar información en el servidor HTTP)

5 Reglas de entrega

- El laboratorio se realiza en tríos seleccionados en Aula.
- La fecha de entrega es el dia **22 de Agosto** a las 23:59 hrs.
- 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 y solicitados en el presente enunciado para la correcta ejecución de la entrega.
- 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 15 ptos.
- Cualquier sospecha de copia será notificada debidamente al 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.
- Cualquier sospecha de uso de herramientas de inteligencia artificial se tendrá en consideración para la evaluación de su entrega, lo que podría conllevar descuentos posteriores.
- Errores de formato involucran una penalización de 15 puntos a la entrega.
- Errores en el nombre de archivo de la entrega involucran una penalización de 15 puntos a la entrega.
- Errores en los roles USM de alguno de los estudiantes del grupo conllevan un descuento de 5 puntos a la entrega.

- Faltas graves de ortografía en el informe de Laboratorio implican penalización de 5 a 15 puntos a la entrega dependiendo de la cantidad de faltas encontradas.
- No especificar fuentes de información o bibliografía conlleva una penalización de 10 puntos a la entrega.
- En ciertos casos de sospecha de actividad maliciosa, copia, uso de IA u otras conductas, es posible que se cite al grupo a una sesión de defensa, en cuyo caso los ayudantes de laboratorio se contactarán vía correo con los grupos correspondientes. En caso de ser citado a defensa y no asistir, se aplicarán las medidas correspondientes y el laboratorio será evaluado con nota 0.