# Laboratorio 1
# Redes de Computadores

Profesor: Erika Rosas Olivos
Ayudantes: Ignacio Cofré
Martín Crisóstomo
Felipe Montero
Benjamín Riquelme

April 2020

## 1 Objetivos

- Familiarizarse con protocolos usados en Internet.
- Conocer la arquitectura cliente-servidor
- Aprender a utilizar sockets UDP y TCP en *Python*
- Aprender a analizar el tráfico de red usando la herramienta *Wireshark*

## 2 Introducción

Los sockets nos permiten establecer conexiones y comunicar dos aplicaciones que se ejecutan en dos máquinas, siguiendo una estructura cliente-servidor. *Python*, a través de la librearía **socket**, nos permite emplear esta estructura tanto por el lado del servidor como del cliente. La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

Por otra parte, *Wireshark* es una herramienta analizadora de protocolos, la cual permite examinar el tráfico de una red dentro de un adaptador de la máquina, permitiendo organizar y filtrar los paquetes encontrados.

## 3 Tarea

La tarea consiste en dos partes: inicialmente se deben desarrollar dos aplicaciones que cumplan los roles de cliente y servidor respectivamente en *Python*, empleando la librería *"socket"*. Posteriormente, se empleará la herramienta *Wireshark* para estudiar los paquetes enviados y recibidos por los programas desarrollados.

## 3.1 Programar

El objetivo de esta sección corresponde a crear una arquitectura cliente-servidor, en donde el cliente realiza una consulta enviando una URL, es decir, enlace de Internet, y el servidor le responde con el header de la respuesta HTTP recibida. La interacción entre el cliente y servidor se presenta en el siguiente diagrama.

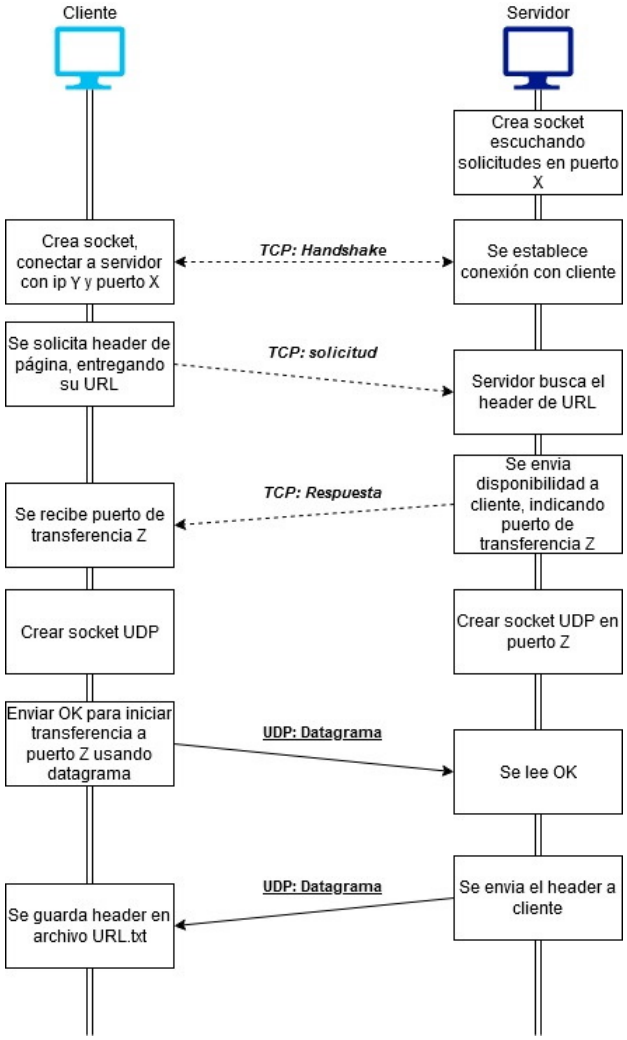


Figure 1: Interacción cliente-servidor

Es importante indicar que al establecer la conexión, el cliente puede realizar varias consultas al servidor, sin terminar la ejecución. Para terminar la interacción, el cliente debe enviar como URL la frase "terminate" al servidor, para que tanto cliente como servidor corten los elementos liberados. Toda conexión del tipo UDP debe ser usada para una interacción: al terminar la transferencia del header, se deben terminar los sockets UDP empleados.

### 3.1.1 Cliente

El cliente debe presentar las siguientes capacidades:

- Establecer una conexión TCP con el servidor

- Realizar la consulta al servidor a través de la conexión TCP, enviando una URL.
- Preparar un socket UDP que se conecte al puerto indicado en la respuesta de la consulta.
- Debe avisarle al servidor, a traves del socket UDP, que quiere empezar a recibir el mensaje HTTP.
- Poder almacenar la información recibida en un archivo *"URL.txt"*.

### 3.1.2 Servidor

El servidor, además de responder a las solicitudes de cliente, debe encontrar los mensajes HTTP, correspondientes a respuestas del mensaje "GET" del protocolo HTTP. Para ello, debe seguir los pasos establecidos por el protocolo respectivo. ***Hint**: ¿hay alguna relación entre las URL y las direcciones IP?*. Se presenta un diagrama resumiendo el comportamiento esperado entre el servidor y una página web:

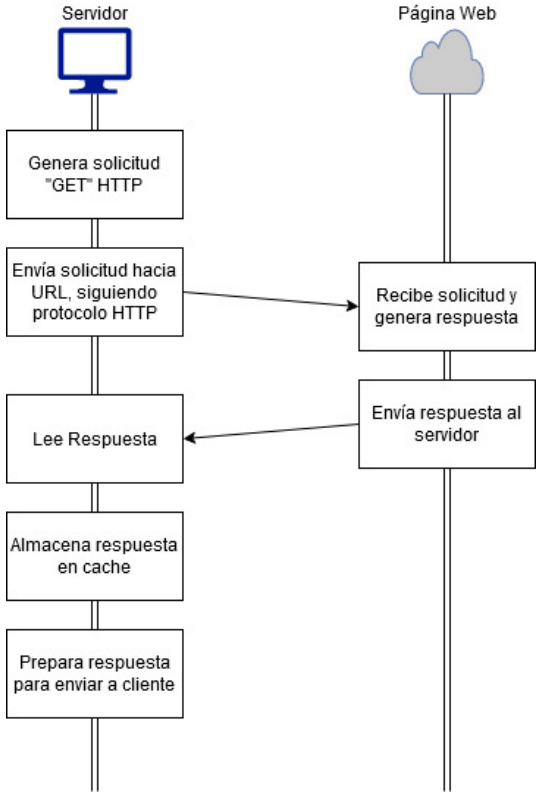


Figure 2: Interacción servidor-página web

Para reducir la cantidad de solicitudes realizadas a la red, el servidor debe implementar un cache del estilo LRU de tamaño 5. Es decir, se mantienen los headers de las respuestas HTTP de las últimas 5 URLs consultadas, y si se realiza una nueva consulta, se elimina el elemento menos reciente del cache, y se agrega la respuesta recibida al cache. En caso de consultar una URL ya almacenada en el cache, se debe actualizar su posición para mantener el orden temporal del cache, sin realizar una nueva eliminación.

El servidor debe presentar las siguientes capacidades:

- Escuchar conexiones TCP entrantes en un puerto definido.

- Procesar los mensajes enviados por un cliente
- Realizar consultas "GET" de URLs dadas.
- Mantener un cache de tamaño 5 que almacene el header de las respuestas HTTP, persistente entre ejecuciones.
- Informar al cliente del puerto para realizar transferencias UDP.
- Enviar el header HTTP solicitado por el cliente.

### 3.2 Wireshark

En esta sección, debe emplear la herramienta *Wireshark* para analizar las los paquetes relacionados a la aplicación que desarrolló y responder las siguientes preguntas:

- Referente a los mensajes realizados por las aplicaciones: ¿Qué tipos de protocolo espera ver? ¿Cuáles encontró? Justifique sus expectativas y las diferencias que encuentre.
- Las interacciones vía TCP entre el cliente y el servidor, ¿deben ocupan los mismos puertos a lo largo del tiempo? ¿Coincide con lo visto en *Wireshark*? Fundamente.
- Los contenidos de los mensajes enviados entre las aplicaciones, ¿son legibles?
- Encuentre la respuesta a la consulta HTTP recibida por el servidor, ¿el header es igual al almacenado por el cliente, o existe alguna diferencia importante? Explique.

### 3.3 Bonus (10 puntos)

Modifique el servidor para que atienda a más de un cliente al mismo tiempo. Explique diferencias importantes entre las interacciones de un cliente y otro atendidos al mismo tiempo. ***Hint**: puede resultar útil Wireshark para fundamentar esta parte.*

## 4 Consideraciones

- Consultas sobre la tarea se deben realizar en aula o enviar un correo a **martin.crisostomo@sansano.usm.cl** o **benjamin.riquelme@sansano.usm.cl**

## 5 Reglas de entrega

- La tarea se realiza en **grupos de 2 personas**
- La fecha de entrega es el día **domingo 3 de mayo de 2020.**
- El código debe correr en **Python 3.8**.
- La entrega debe realizarse a través de Aula, en un archivo comprimido **ZIP** o **TAR GZ**, y debe indicar el rol sin dígito verificador de los integrantes en el nombre. Ejemplo: **T1-Rol1-Rol2.zip**
- Debe entregar todos los archivos fuente necesarios para la correcta ejecución. Recuerde que el código debe estar identado, comentado, sin warnings y sin errores.
- Debe entregar un **README** con nombre y rol de cada integrante, además de la información necesaria para ejecutar los archivos.

- Cada día de atraso se penalizara con un descuento de **10 puntos** hasta los 3 días, Posterior a dicho tiempo se evaluara con **nota 0**.
- Copias serán evaluados con **nota 0**