Laboratorio 1: Redes de Computadores

Profesor: Jorge Díaz M.
Ayudantes: Iñaki Oyarzun M. & Javiera Cárdenas

Marzo - Abril 2023

1 Objetivos del laboratorio

- Aprender a utilizar sockets UDP y TCP en `Python`.
- Conocer y aplicar la estructura cliente-servidor.
- 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, `Python` junto a otros programas (como Golang) 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, `Golang` 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).

3 Laboratorio 1

3.1 Enunciado

Es el año 2XXX, el complejo de investigación Black Mesa, ubicado en Nuevo México, ha causado un desastre que significaría el fin de la humanidad como se conocía, la llamada *"Cascada de Resonancia"*. Producida en el Sector C de pruebas, la cual investigaba la creación de portales para viajes a larga distancia.
Un error científico provocó una reacción en cadena de aparición de portales hacia otras dimensiones, como XEN, habitadas por alienígenas monstruosos y entidades altamente nocivas para el planeta y entre ellas, igualmente a la peor entidad conocida en el multiverso entero.

Los Combine, un imperio interdimensional disperso entre una cantidad infinita de universos y sistemas solares. Cuyo único objetivo es dominar el multiverso, junto con controlar y esclavizar a todas las especies que existen.
La humanidad ante esta gran amenaza, decidió unir fuerzas contra este monumental mal común, sin embargo, todo fue en vano, en tan solo 7 horas logró sucumbir el planeta completo. Pasando a ser anexada la tierra como parte de los Combine y los humanos esclavizados.

Sin embargo, aún existe una luz de esperanza, sobrevivientes al gran desastre del laboratorio de Black Mesa crean una gran fuerza de resistencia que busca como objetivo luchar desde las sombras por la tan anhelada libertad de los humanos. Liderada y enardecida por su actualmente desaparecido líder de iniciales G.F.
Actualmente la resistencia sobrevive del espionaje. Uno de sus soldados, Barney Calhoun se ha logrado infiltrar dentro de las líneas de los Combine como parte de la guardia de protección Civil para averiguar los planes ocultos y poder actuar desde las sombras. Barney aprovechando sus conocimientos tecnológicos ha dejado a disposición un terminal escondido dentro de los servidores de la Citadela, lo cual permitirá el traspaso de las fotografías de los documentos o planes de los Combine.
Ustedes como grandes programadores de renombre previo a la caída de la humanidad son contactados por la resistencia para diseñar un sistema que permita acceder y obtener los datos desde el servidor creado por Barney.

De ustedes dependerá la supervivencia de la humanidad...

3.2 Explicación

El servidor configurado desde la citadela se conforma de acuerdo al siguiente informe de Barney:

```
(...)  De acuerdo al tiempo que pude obtener durante las limpiezas de sectores, pude
configurar un servidor base que pueda enviar fotos al azar de los documentos de los
combine.  Este servidor trabaja de acuerdo a comandos recibidos del exterior, para
que así pueda lograr camuflarse con las señales de los intercomunicadores de los
soldados(...)

No lo olviden, la IP del servidor es:  jdiaz.inf.santiago.usm.cl

Deben enviar primero un mensaje por medio de un protocolo inseguro para solicitar
los datos de una imagen, en el siguiente formato:'GET NEW IMG DATA'

A cualquiera de los siguientes puertos:  [50006, 50007, 50008, 50009, 50010]

El servidor contestará con un texto que contenga los datos de la imagen, de la siguiente
forma:  'ID:X W:X H:X P1TCP:X P2UDP:X PV:X'o
'ID:X W:X H:X P1TCP:X P2UDP:X P3UDP:X PV:X'

Donde:
- ID es el identificador de la imagen
- W y H son el ancho y alto de la imagen para que puedan calcular el buffer PNG (recuerden
que el buffer en bytes de un png es la multiplicación de las dimensiones por los
3 canales)
- P1TCP es el puerto al que deberán consultar la primera parte de la imagen mediante
una conexión segura
- P2UDP y P3UDP son los puertos al que deberán consultar la segunda y tercera parte
de la imagen mediante una conexión insegura.  Puede enviar al azar una división en
2 o 3 partes.
- Y PV que sera un puerto seguro para verificar los datos entregados de la imagen
para que puedan confirmar antes de escribir la foto

Con esa información solo tendrán que conectarse de forma segura a la misma IP siguiendo
el orden de los puertos para poder solicitar las partes la imagen con el siguiente
comando:  'GET Y/Y IMG ID:XX' (donde Y/Y corresponde a la parte de la imagen que
se pide, por ejemplo 1/3 o 2/2, XX es el numero del id, en caso de ser un dígito
es solo X).

Luego de eso, el servidor enviará la parte imagen en bytes (recuerden el buffer calculado
para evitar la pérdida de datos y si falla vuelvan a intentar hasta que funcione).(...)

Mucha suerte en todo compañeros...
```

De lo anterior se identifican 2 pasos para la obtención de una imagen

3.2.1 Paso 1: Obtención de los datos de la imagen

Deberán enviar un comando al servidor por medio de una conexión UDP de acuerdo a los datos entregados que son:

- **IP: jdiaz.inf.santiago.usm.cl**
- **Puertos (cualquiera): [50006, 50007, 50008, 50009, 50010]**

El comando es: `GET NEW IMG DATA` (**Sin espacios de más y solo en mayúsculas**)

Luego el servidor puede contestar con uno de estos mensajes:

`ID:X W:X H:X P1TCP:X P2UDP:X PV:X` o `ID:X W:X H:X P1TCP:X P2UDP:X P3UDP:X PV:X`
Donde cada X serán los valores del:

- **ID** de imagen
- **W**idth de la imagen
- **H**eight de la imagen
- **P1TCP** el primer puerto para pedir la primera parte de la foto
- **P2UDP** el segundo puerto para pedir la segunda parte de la foto
- **P3UDP** el tercer puerto para pedir la tercera parte de la foto
- **PV** es el puerto para verificar los bytes de cada parte de la foto una vez obtenidos

Nota: Recuerde que los valores no llevan ceros antepuestos, es decir, para el valor 1 no es 001 o 01

3.2.2 Paso 2: Solicitud de la imagen

Este proceso debe realizar la obtención de la imagen y reconstrucción de la misma a partir de los bytes recibidos. El proceso a realizar es el siguiente:

- Establecer una conexión **TCP** con el servidor de acuerdo al puerto **P1TCP** indicado del mensaje anterior.
- Calcular el buffer requerido de acuerdo a lo indicado por el primer mensaje obtenido
- Enviar el mensaje de solicitud de la imagen: `GET Y/Y IMG ID:X` (Con X el numero del ID entregado del mensaje anterior e Y los valores que serán dependiendo de la cantidad de partes que tiene la foto, por ejemplo 1/2 o 1/3)
- De acuerdo al buffer calculado, recibir los bytes de la respuesta del servidor para la primera parte
- Establecer una conexión **UDP** con el servidor de acuerdo al puerto **P2UDP** indicado del mensaje anterior.
- Enviar el mensaje de solicitud de la imagen: `GET Y/Y IMG ID:X` (Con X el numero del ID entregado del mensaje anterior e Y los valores que serán dependiendo de la cantidad de partes que tiene la foto, por ejemplo 2/2 o 2/3)

- De acuerdo al buffer calculado, recibir los bytes de la respuesta del servidor para la segunda parte
  En caso de haber recibido una tercera parte, repetir el paso anterior pero con el valor 3/3

- Con los bytes ya obtenidos, se debe realizar una conexión **TCP** al puerto **PV** obtenido y enviar los bytes de las 3 o 2 partes de la imagen juntas para comparar con la foto original. Si el servidor responde `'200: SUCCESS'` querrá decir que los bytes obtenidos de las 3 partes en conjunto conforman la imagen correctamente. Y en caso de obtener un error, deberá repetir el proceso completo hasta que obtenga la respuesta de confirmación.

- Con los bytes obtenidos y verificados en conjunto, reconstruir la imagen con el nombre: `<ID imagen>.png`

Nota: Se recomienda revisar las condiciones para tener en cuenta los textos que deben ser mostrados por pantalla durante el desarrollo de ambos pasos

3.3 Topología del Sistema

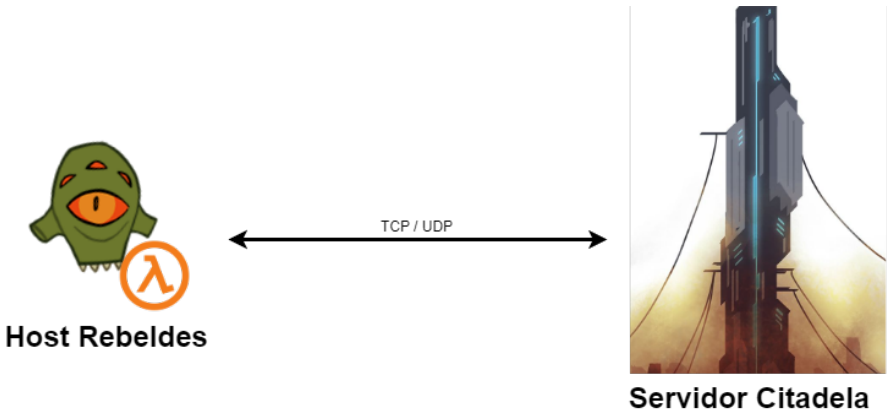

Figure 1: Topología de lo que se busca implementar: Siendo una estructura clásica de comunicación entre un cliente y su servidor)

3.4 Diagrama

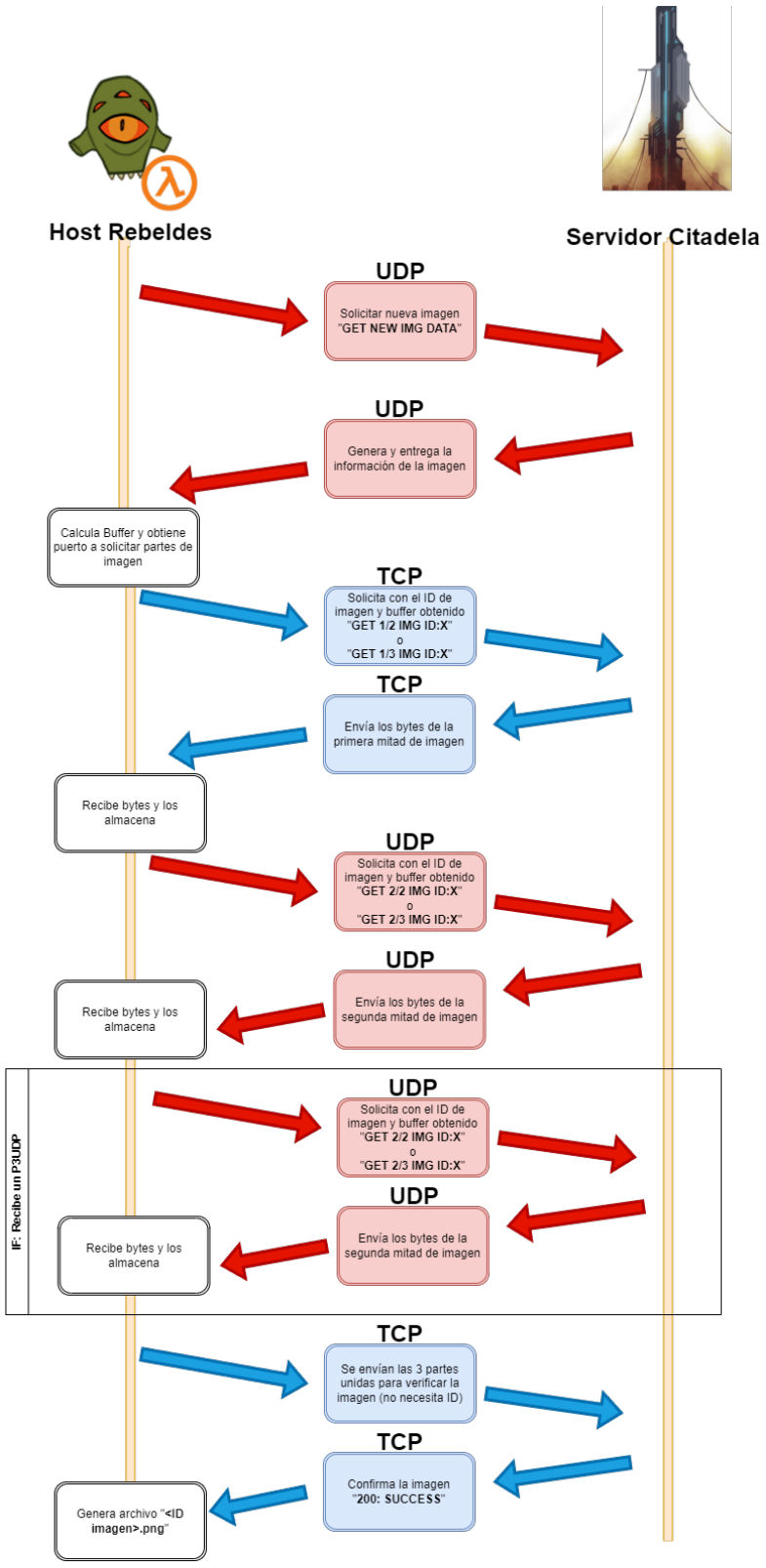

Figure 2: Diagrama del proceso del Laboratorio

3.5 Condiciones:

Para el desarrollo de este laboratorio, se hace importante el uso de prints por pantalla para dar conocimiento de lo que se encuentra ocurriendo dentro de la interfaz en todo momento, es por ello que, se plantean a continuación los prints mínimos que **debe** tener el programa.

- Cada mensaje a enviar y respuesta recibida debe ser vista por pantalla, por ejemplo:
  - `Mensaje enviado a <IP>:<Puerto> por <TCP/UDP>:  <Mensaje>`
  - `Mensaje recibido de <IP>:<Puerto> por <TCP/UDP>:  <Mensaje>`

  Nota: Para el caso de los bytes, si prefieren pueden indicar solamente que se recibieron los bytes para evitar el print muy largo
- Durante el inicio y término de la escritura de la imagen se debe indicar cuando se realiza, por ejemplo:
  - `Iniciando escritura de imagen`
  - `Escritura de imagen finalizada`
- Para cada nuevo intento deberá separarse con un print entre cada iteración para lograr ser diferenciada. Por ejemplo:
  - `** Nuevo Intento **`
  - `** FIN **`

Por ende, durante el intercambio de mensajes con el servidor, debe notarse cuando se envía y se recibe un mensaje, al igual que señalar cuando un proceso (como el de verificación de los bytes) fue exitoso o si hubo un error.

4 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.
- Se debe hacer uso de los sistemas de LDS para poder probar el código y ejecutarlo, se indicará en la ayudantía las instrucciones requeridas para aquello y posibles programas recomendados para el trabajo.
- La fecha de entrega es el día **14 de Abril de 2023 hasta las 23:59**
- El código debe correr en Python 3.7+ en Ubuntu 22.04. Solo está permitido hacer uso de la librería `socket`.
- 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. Con el código bien identado, comentado, sin warnings ni errores.
- 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 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.

5 Consultas:

Para hacer las consultas, recomendamos hacerlas por medio del foro del ramo en Aula. De esta forma los demás grupos pueden beneficiarse en base a la pregunta. **Se responderán consultas hasta 48 hrs. antes de la fecha y hora de entrega.**