Tarea 3

Dante Chavez Dante.chavez@alumnos.uv.cl,

1. Introducción

Se requiere crear una herramienta basada en línea de comandos para consultar el fabricante

de una tarjeta de red dada su dirección MAC o su IP, en este informe se explicará cómo se creó esa herramienta, las funciones de esta, diagramas de flujo y documentación de Código.

2. Materiales y Métodos

Esta herramienta está basada en el sistema operativo Windows y esta desarrollada en Python, además se usará una API REST publica como base de datos, la cual está disponible en https://maclookup.app, la que consiste en obtener los fabricantes de la tarjeta de red a partir de su MAC.

3. Resultados

En esta sección se describirá la documentación del software OUILookup, cada función principal, lo que hace y su diagrama de flujo correspondiente.

En la **Figura1** se explica que significa cada figura de los diagramas de flujo.

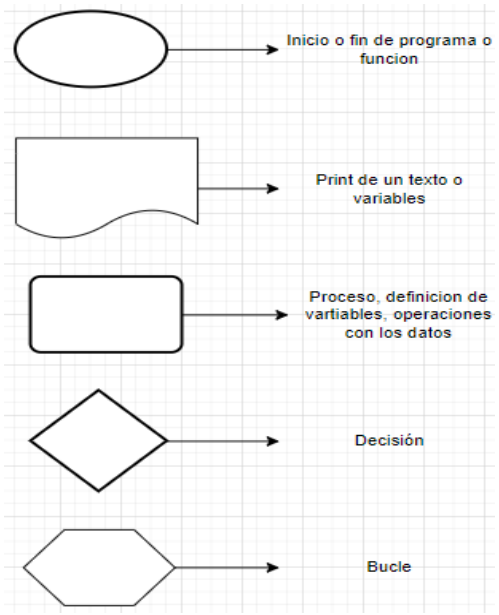

Figura 1. Explicación figura de diagrama de flujo

3.1. Función obtener datos por IP

La función para obtener datos por IP llamada en código "obtener_datos_por_ip(ip)" obtiene los datos por la IP que le llega como parámetro, importa la tabla ARP del computador a través de otra función, lee si la IP está dentro de nuestra red, la cual es 192.168.1.0/24 si la IP que le llega como parámetro está dentro de nuestra red la busca en la tabla ARP, al encontrarla obtiene su Mac usa la función "obtener_datos_por_api(mac)" para obtener información del vendedor para luego imprimirla en pantalla, además de mostrar el tiempo que se demoró el servicio REST en hacer la consulta, si la IP a buscar no está en el rango o no está en la tabla ARP se mostrara un mensaje indicando ello.

Diagrama de flujo se muestra en **Figura2**

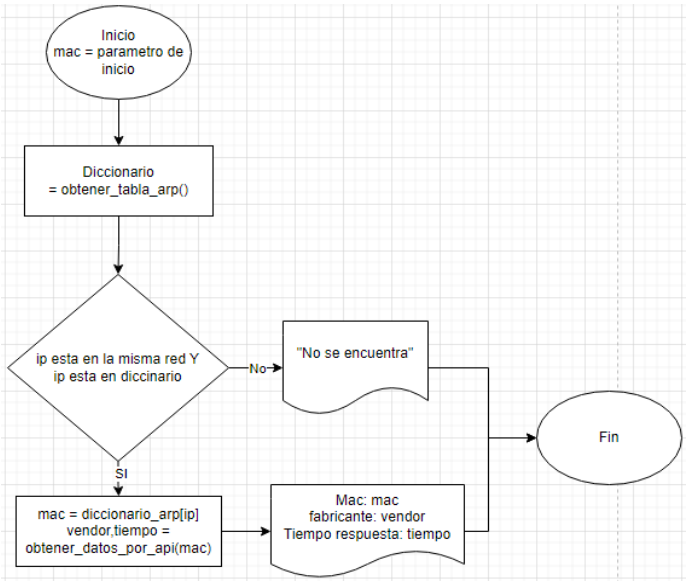

Figura 2. Diagrama de flujo función obtener datos por IP

3.2. Función obtener datos por api

La función obtener datos por Api llamada en el Código "obtener_datos_por_api(mac)" se le ingresa un parámetro el cual es la Mac, esa MAC la busca a través de la API con la librería **requests,** para luego devolver el vendedor de esa MAC, además de mostrar el tiempo que se demoró el servicio REST en hacer la consulta por medio de la librería o modulo **time** con la función time.time() la cual mide el tiempo en segundos [1], estos se pasan a milisegundos y también se retorna a donde fue llamada la función.

Diagrama de flujo se muestra en **Figura3**

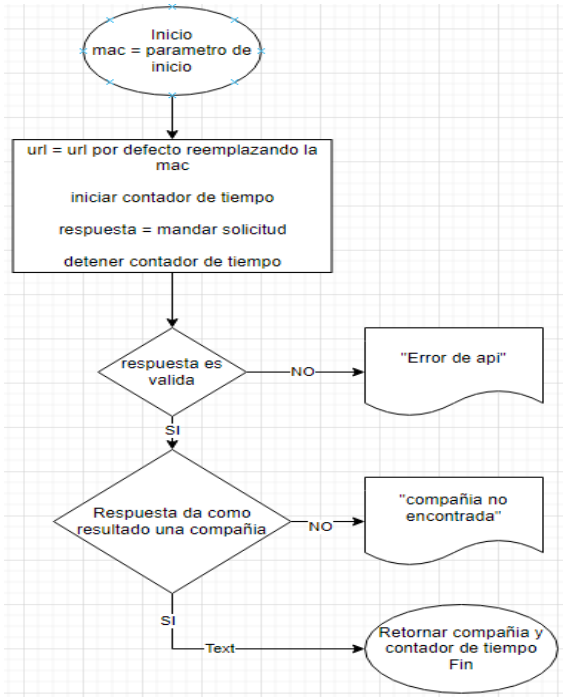

Figura 3. Diagrama de flujo función obtener datos por API

3.3. Función Obtener tabla ARP

La función para obtener la tabla ARP llamada en Código "obtener_tabla_arp" no usa parámetros, usa la librería "subprocess" para obtener la tabla ARP, ignora las 2 primeras líneas porque es texto y guarda las IP y las Mac en una librería para luego retornarla al Código que llamo esta función.

Diagrama de flujo se muestra en **Figura4**

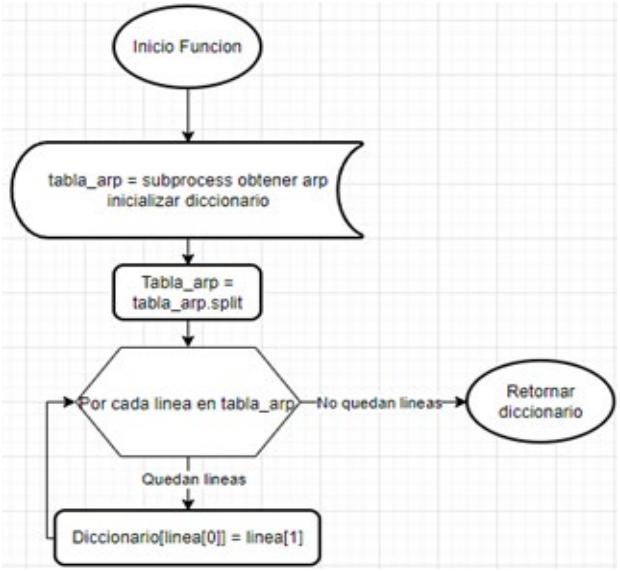

Figura4. Diagrama de flujo función obtener tabla ARP

3.4. Función Main

La función Main llamada en el Código “main(argv)” usa como parámetro los argumentos con el que se ejecutó el programa, permite elegir entre 4 opciones las cuales son: --help, --arp, --ip <argv> y --mac <argv> cada una ejecuta una de las funciones vistas anteriormente.

Diagrama de flujo se muestra en **Figura5**

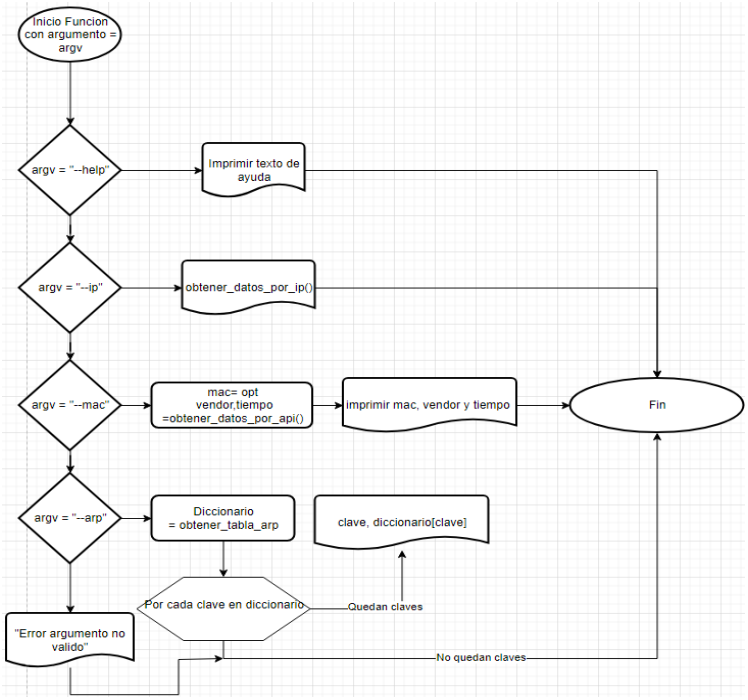

Figura 5. Diagrama de flujo función main

3.5. Inicio de programa

Al iniciar el programa se comprueba si __name__ == " main " lo que hace es permite ejecutar código cuando el archivo se ejecuta como un script, pero no cuando se importa como un módulo y después se llama a la función “main(argv)” con los argumentos con los cuales se inició el programa, además se comprueba si el usuario tiene la librería requests, si la tiene la importa y si no la instala en el ordenador del usuario.

4. Preguntas a responder

a) ¿Qué es REST? ¿Qué es una API?

REST significa en inglés “representational state transfer” lo que se traduce en “transferencia de estado representacional”, es un estilo de arquitectura de software que se centra en interfaces

uniformes, implementación independiente de componentes, escalabilidad de relaciones entre ellos para poder reducir la latencia percibida por el usuario se usa para crear aplicaciones web confiables y estas aplicaciones que obedecen el estilo de arquitectura se les llama **RESTful** sus servicios utilizan los protocolos estándar, este término también se asocia al diseño de la API, REST asociado a las API define un conjunto de funciones como get, put, delete que son del protocolo estándar que podemos usar para acceder a datos del servidor como se hizo en esta tarea, entre servidor y usuario se intercambia datos mediante HTTP. [2][3]

API significa en inglés "Application Programming Interface" lo que se traduce en "interfaz de programación de aplicaciones" estas permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos, esta comunicación se realiza entre servidor y usuario, quiero recalcar que son 2 piezas de software las que se comunican y por lo que no es una interfaz de usuario, uno como programador es el que tiene que implementar esa interfaz, un ejemplo de esto es el programa que se muestra en este informe. Otra parte destacable es que el servidor NO guarda datos del cliente entre las solicitudes y las respuestas del servidor son simples y es por esto que las APIs no muestran "como están hechas", son como funciones, uno ingresa un parámetro y la API te devuelve un valor, omitiendo el como lo hizo.[4][5]

b) ¿Cómo se relaciona el protocolo HTTP con las API REST y cuál es su función en la comunicación entre clientes y servidores?

En la pregunta **a)** se introdujo con bastante detalle sobre la relación por ser un componente fundamental de estas, continuando lo dicho en la pregunta anterior, se relacionan porque API REST usa HTTP como su protocolo de comunicación, pero no es el único. También como se explicó anteriormente API REST y HTTP utilizan protocolos estándares (explicados anteriormente), los 2 además tienen la misma característica de "Request and response" lo que significa que el servidor recibe la solicitud y enviar una respuesta, de manera simple. Por último, en las solicitudes de HTTP y API REST los servidores no guardan información sobre las solicitudes anteriores, cada solicitud debe procesarse solamente con la información que tiene el servidor y la solicitud, independiente de las anteriores solicitudes.

c) ¿Qué papel juega la dirección IP en el acceso a recursos a través de una API REST?

El papel básico es lo que se ha visto en clase, identificar dispositivos y permitir el enrutamiento de paquetes, comunicar dispositivos, etc. En resumen, se trata de comunicación fin a fin, el usuario puede pedir recursos al servidor y este los proporciona. Pero ese no es el único, otro importante es limitar cuantos recursos puede usar una IP, por ejemplo, la API usada en este trabajo tiene un límite de tarifa [5] medido con la IP que hace la solicitud, por último es la seguridad del programa y esto está relacionado también con el punto anterior, la API REST puede denegar servicios a una IP si detecta que esta es "maliciosa" o si está recibiendo solicitudes masivas como por ejemplo un ataque DoS [6], también puede ser algo simple como solo permitir a ciertas IPs conectarse, algo conocido como "whitelisting".

d) ¿Por qué es importante considerar la latencia de red y el ancho de banda? ¿Cómo afectan estos factores al rendimiento de la API?

Es importante considerar la latencia porque esta afecta la experiencia de usuario final, al igual que el ancho de banda, al hacer esta herramienta pude comprobar que la API usada en este trabajo tiene un tiempo de respuesta considerable (~750ms) esto se traduce en una experiencia lenta al usar la herramienta. También es importante considerar el lado del servidor de manera que el tiempo y capacidad de procesamiento afectan la latencia de la conexión, por el lado del ancho de banda si el cliente dispone de uno limitado se le dificultara más recibir el mensaje, por ejemplo, Google recomienda reducir el tamaño de archivo [7] comprimiendo la respuesta, también está la solución que habilito la API que se usó para esta herramienta que es poder pedir todos los datos de la MAC o solo el vendedor [8], la primera son 13 atributos que envía el servidor mientras que la segunda es solo 1, por lo que se reduce enormemente el tiempo de envío, así haciendo una mejor gestión de recursos.

e) ¿Por qué el programa desarrollado utilizando API REST es más lenta su ejecución?

Es más lento porque se le añade un intermediario y con esto más complejidad al proceso de comunicación entre cliente y servidor, lo que hace más lenta la ejecución ya que el sistema está haciendo más trabajo para completar una tarea, otro inconveniente es que por API REST tiene que pasar por la HTTP y no se puede enviar normalmente de manera local, porque en este caso simplemente se podría acceder a la base de datos.

f) ¿Cuál es la diferencia entre la dirección MAC (Media Access Control) y la dirección IP, y en qué capa de la red se utilizan cada una de ellas?

g) La MAC es un identificador único de la tarjeta red y esta es asignada de manera única por el fabricante del dispositivo por lo que este no cambia, se usa para identificar el dispositivo en la **capa 2**. En cambio, la dirección IP es un identificador que lo asigna comúnmente el ISP que tengamos contratado y este puede ir cambiando nuestra IP, se usa para identificar dispositivos en la red por lo que se usa en la **capa 3.**

h) ¿Cómo pueden las redes LAN (Local Area Networks) y WAN (Wide Area Networks) afectar la accesibilidad y la velocidad de respuesta de una API REST?

Suponiendo que la API REST se encuentra dentro de la misma red a hablar, Si es en una red LAN la latencia será mucho más baja que la WAN ya que la cantidad de nodos intermedios disminuirá drásticamente por lo que la accesibilidad y tiempo para conectar el usuario y servidor del API REST será mucho más rápido, por lado del ancho de banda también las redes LAN son más rápidas ya que es una red “cerrada” y por sufre menos variación, además de tener comúnmente más ancho de banda disponible mientas que las WAN no suelen tener el mismo porque son redes mucho más grandes y con más tráfico que las LAN. Las redes LAN tienen sus nodos comúnmente cerca uno de otros, como por ejemplo un edificio entero, mientras que las redes WAN pueden ser tener una separación enorme, como atravesar continentes enteros.

i) ¿Qué es un enrutador y cómo se utiliza para dirigir el tráfico de datos? ¿Qué relación tiene esto con el enrutamiento de solicitudes en una API REST?

De manera simple un enrutador es un dispositivo de red que opera en **capa3** y reenvía paquetes de datos entre redes informáticas, los reenvía seleccionando la ruta de manera eficiente, dirigen el tráfico de manera parecida a los switches que se vieron en clases, el enrutador posee una tabla de enrutamiento o un protocolo de enrutamiento, también tiene una toma de decisiones cundo llega un paquete la cual parecido a: actualizar tabla de enrutamiento, leer ruta de paquete, leer tabla de enrutamiento y reenviar por el puerto que este esa dirección. En cambio, la API REST tiene un enrutamiento se basa en la URI (Uniform Resource Identifier) de la solicitud, esta identifica el recurso al que se quiere acceder, luego el enrutador de la API utiliza esa información para determinar una ruta para esa solicitud, para luego reenviarla y por último acceder a la función que dará el dato solicitado. Dado lo explicado anteriormente, tanto como en enrutador y enrutador de API REST tienen la tarea de ver hacia donde se dirige la solicitud, tomar la decisión de elegir para donde dirigir esa solicitud, reenviar la solicitud para que luego esta llegue a destino.

j) **¿Cómo se asocian los puertos de red con servicios y aplicaciones específicas?**

Un puerto es un numero asignado para identificar de forma única un punto final de conexión y dirigir los datos a un servicio en específico, este puerto a nivel de software identifica un proceso especifico lo que sería una aplicación, esta asignación viene acompañada por un protocolo de transporte, pueden ser TCP o UDP. El total de puertos son números enteros sin signo de 16 bits que van desde el 0 hasta el 2^{16}-1 y se dividen en 3 categorías, también se explicara que aplicaciones específicas usan cada puerto:

- **Puertos conocidos**: van desde el 0 hasta el 1023, las aplicaciones que usan estos puertos son por ejemplo navegadores web (HTTP) que es el 80, también pueden usar la version segura que es "HTTP Secure" (HTTPS) y este se encuentra en el puerto 443. La aplicación **OUILookup** usa cualquiera de estos 2 puertos para comunicarse ya que manda solicitudes de tipo HTTP.
- **Puertos registrados**: van desde el 1024 hasta el 49151, están asignados a servicios específicos, la lista de estos servicios la maneja IANA [9].
- **Puertos dinámicos o privados**: van desde el 49152 hasta el 65535, conocidos como puertos efímeros, estos puertos es los que se usan comúnmente para aplicaciones que requieran conexión cliente-servidor.

5. Discusión y conclusiones

Creando esta herramienta se puede aprender que es bastante simple el funcionamiento de buscar el vendedor, empresa de un dispositivo por su dirección Mac, usando una API REST para automatizar el proceso, además de poder hacer lo anterior con la tabla ARP del computador para saber quién fabrico o vendió los dispositivos que tenemos alrededor de nosotros.

6. Referencias

[1] https://docs.python.org/3/library/time.html#time.timez
[2] Cesare Pautasso, REST: Advanced Research Topics and Practical Applications
[3] Roy Thomas Fielding,Architectural styles and the design of network-based software architectures
[4] Reddy, Martin (2011). API Design for C++. Elsevier Science. p. 1. ISBN 9780123850041.
[5] https://maclookup.app/api-v2/rate-limits
[6] What is a DDoS Attack? - DDoS Meaning". usa.kaspersky.com. 2021-01-13.
[7] https://cloud.google.com/healthcare-api/docs/how-tos/best-practices?hl=es-419
[8] https://maclookup.app/api-v2/documentation
[9] https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml