Tarea 2 - Redes de computadores: “OUILookup”

Renato Herrera Leonardini, [renato.herrera@alumnos.uv.cl](mailto:renato.herrera@alumnos.uv.cl)

Felipe Castro Olivares, [felipe.castroo@alumnos.uv.cl](mailto:felipe.castroo@alumnos.uv.cl)

Gabriel Gonzalez León, [gabriel.gonzalezl@alumnos.uv.cl](mailto:gabriel.gonzalezl@alumnos.uv.cl)

# Introducción

De forma general, se solicita la implementación una herramienta basada en línea de comandos en Python que permita consultar el fabricante de una tarjeta de red a través de su dirección MAC, esto se deberá realizar mediante una API REST de consulta, la cual estará disponible en <https://maclookup.app>.

De forma particular el software deberá procesar los parámetros mediante **getopt** siguiendo el paradigma de programación funcional, adicionalmente, el programada deberá contar con 3 argumentos posibles:

1. --help (Encargada de mostrar un mensaje de ayuda de ejecución mediante consola);
2. --mac [MAC] (A partir de una dirección MAC muestra su fabricante);
3. --arp (Muestra las direcciones MACS pertenecientes a la tabla ARP del equipo y sus respectivos fabricantes);

Específicamente para el punto 2 se debe incluir el tiempo de demora de dicha solicitud y de forma particular la solución presentada será capaz de funcionar tanto en Windows como en Linux. Es importante añadir que, en caso de que el fabricante no se encuentre en la base de datos, sea un broadcast o una MAC estática, el algoritmo entregara un mensaje de “Not found”.

Finalmente, también se requiere agregar un apartado donde se explique el funcionamiento de las MACS aleatorias y para qué sirven.

# Descripción del problema y diseño de la solución.

La problemática fundamental será la creación de la función encargada de hacer las consultas de fabricantes, dado que una vez creado dicho modulo, este mismo podrá hacer gran parte del trabajo al momento de consultar los fabricantes de las MACS pertenecientes a la tabla ARP del equipo. Las problemáticas secundarias serán la lectura mediante CLI (Interfaz de línea de comandos) y la detección del sistema operativo en el cual se ejecuta el software.

Nuestro diseño de solución se apoyará en el siguiente modelo:

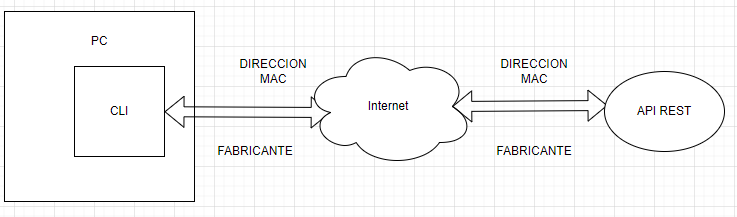


Figura 1.

Para poder llevar a cabo dicho modelo, se requerirá de una función principal capaz realizar las consultas, dado que para implementar la funcionalidad –arp, de igual manera se deberá realizar una solicitud al servicio web.

# Implementación

Primero presentamos el diagrama de flujo asociado a la solución propuesta.

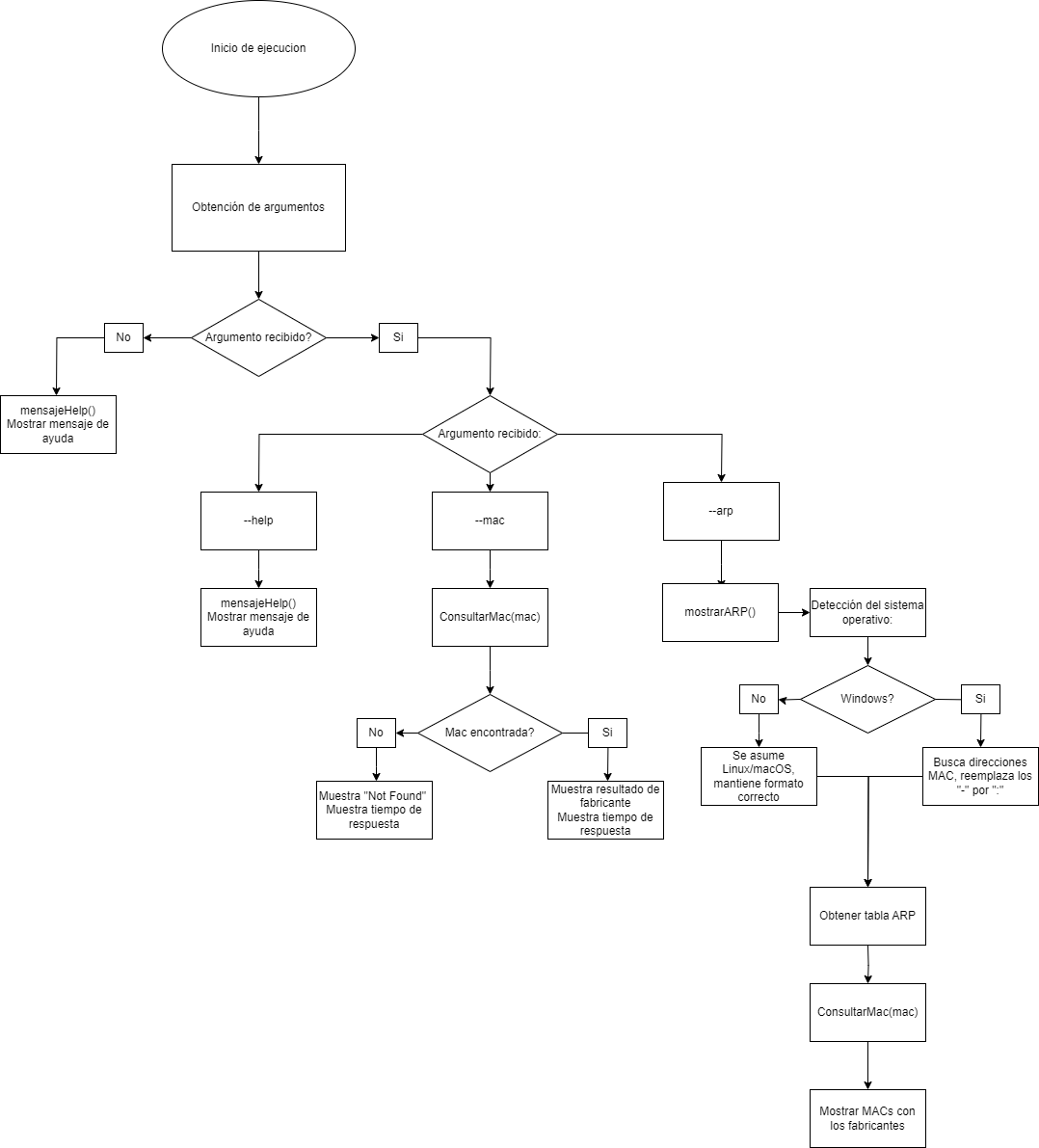


Figura 2.

A continuación, presentamos las secciones más relevantes del código.

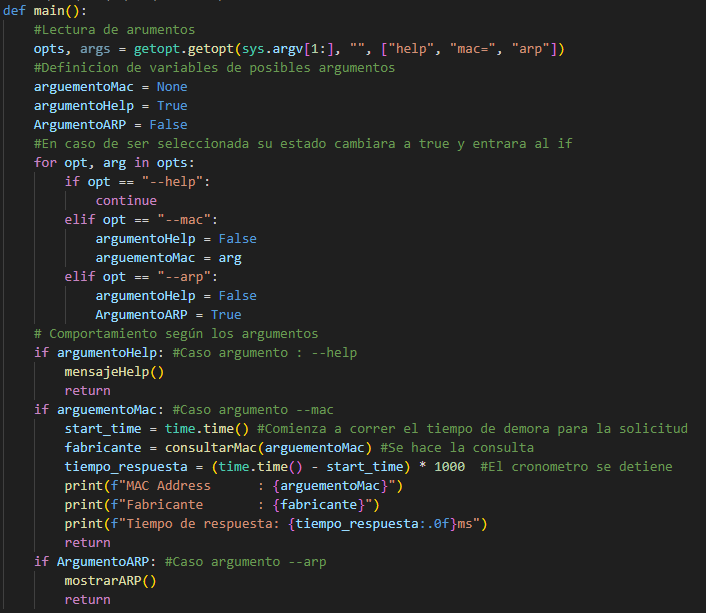


Figura 3.

La **Figura 1** corresponde al main(), aquí es donde se hace la lectura de argumentos, en caso de que uno de los 3 argumentos fuera ingresados se activa su funcionalidad especifica , en caso de no seleccionar ningún argumento, se activa automáticamente el comando –help.

Importante añadir que en el caso –mac, una vez seleccionada la opción comienza a correr el tiempo y este se detiene una vez es obtenida (O no) el fabricante.

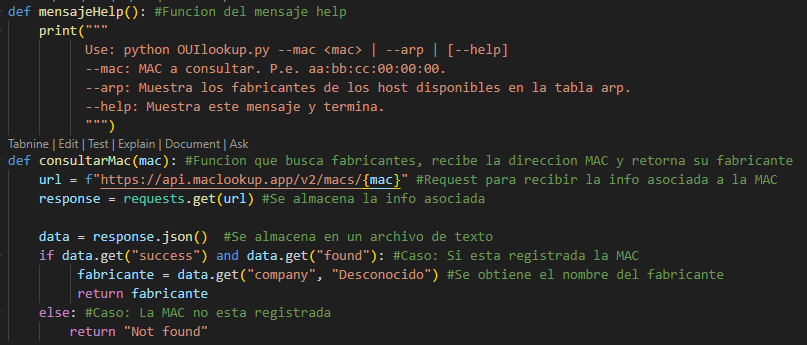


Figura 4.

En la **Figura 2** se aprecia la función mensajeHelp(), encargada de enseñar el mensaje de ayuda para el correcto uso del código. También se encuentra la función consultarMAC(), la cual recibe como parámetro una mac y realiza un request al servicio web para retornar el fabricante, en caso de no haya el fabricante retorna un string “Not found”.

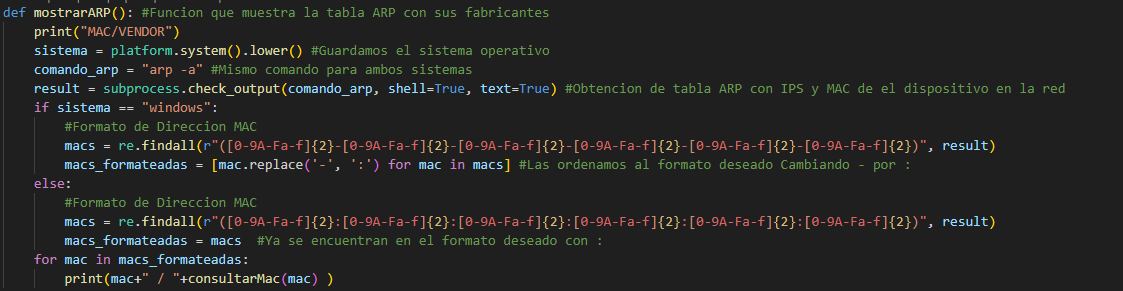


Figura 5.

En la **Figura 3** se aprecia la función mostrarARP() encargada de mostrar las MACS de la tabla ARP correspondiente al equipo en el que se ejecuta el software, con sus respectivos fabricantes. Primero se determina el sistema operativo del equipo y se obtiene la tabla ARP en caso de ser Windows o Linux se hace una búsqueda y orden diferente para mostrar lo deseado, finalmente se recorren dichas MACS y se determina el fabricante con la función consultarMac(), para ser imprimidas por consola.

Finalmente, el principal desafío que tuvimos fue realizar exitosamente el request a la API REST dado que inicialmente no sabíamos donde buscar el hipervínculo de envió, esto fue solucionado leyendo la documentación para desarrolladores que facilita el servicio web. Otro de los desafíos fue seguir el paradigma funcional de programación, dado que inicialmente el calculo del tiempo se encontraba al interior de la funcion consultarMAC(), haciendo engorroso el imprimir todas las MACS de la tabla ARP con sus fabricantes, para solucionar esto se decidió por hacer dicho calculo temporal fuera de la función a cambio de ganar reutilización del módulo.

# Pruebas

Las pruebas se realizaron con las MACS de ejemplo entregadas en la descripción de la tarea 2, todas estas fueron realizadas en un equipo Windows, adicionalmente para la prueba del comando –arp también se utilizo un equipo con el mismo sistema operativo.

A continuación, se presenta evidencia visual del funcionamiento del software.

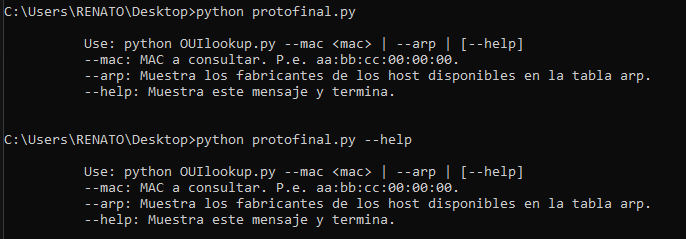


Figura 6.

En la **Figura 4** se aprecian ambos casos donde el comando –help se ejecuta, cuando se ejecuta sin ningún parámetro o cuando específicamente se llama al comando –help.

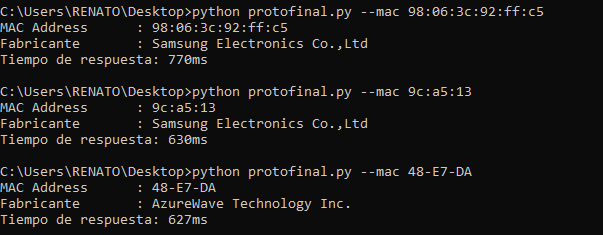


Figura 7.

En la **Figura 5** se evidencias 3 ejemplos donde el fabricante existe y es encontrado.

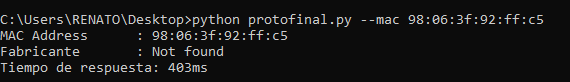


Figura 8.

En la **Figura 6** se aprecia un caso donde el fabricante no se haya en la API REST.

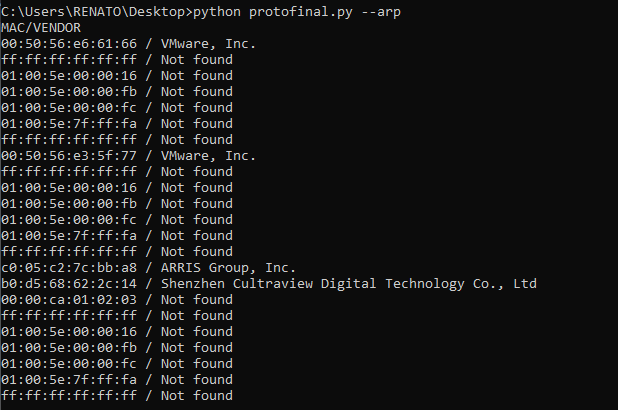


Figura 9.

En la **Figura 10** se deja en evidencia el funcionamiento del comando –arp.

Es mediante estas pruebas y los resultados presentas en las figuras que se logro asegurar el correcto funcionamiento del código. Es importante añadir que a pesar de que las pruebas fueron realizadas en Windows, el algoritmo es capaz de funcionar correctamente en Linux.

# Macs aleatorias

A continuación, se hará explicara que son las macs aleatorias, que son, para que sirven y cómo funcionan.

## ¿Qué son las Mac Aleatorias?

Las MAC aleatorias tienen la función ocultar la MAC real de los usuarios asignando, de manera aleatoria, una dirección MAC. De esta manera oculta la dirección MAC original al conectar un dispositivo a una red WiFi. Esta MAC será aleatoria y única por cada red WiFi con la cual tengamos una conexión como pueden ser las redes públicas (Mall, Universidad) y la red del hogar. Una vez generada, esta será igual a la creada en la primera conexión con dicha red WiFi. Ya que se guarda esta nueva MAC en los ajustes del dispositivo, esto permite que la seguridad del dispositivo sea mayor, ya que evita los rastreos en base a la MAC del dispositivo en base a la red.

## ¿Y cómo funciona una MAC Aleatoria?

Una MAC Real tiene 12 bytes o 48 bits de longitud, y se suele representar como 00:11:22:AA:BB:CC. La función de aleatorización MAC aleatoriza la dirección estableciendo el bit administrado localmente en 1 y el bit de unidifusión en 0. Los otros 46 bits son aleatorios. Además, tenemos otros pequeños ajustes como:

Durante la exploración de redes, el dispositivo envía solicitudes utilizando la dirección MAC aleatoria, **dificultando para terceros el rastreo del dispositivo**.

**Frecuencia de cambio:** Las MAC aleatorias cambian regularmente, ya sea cada vez que el dispositivo busca redes o en intervalos de tiempo definidos, dificultando el seguimiento continuo del dispositivo.

**Privacidad mejorada:** Al generar nuevas direcciones MAC en cada conexión, se impide que las redes puedan identificar de manera consistente un dispositivo, protegiendo así la privacidad del usuario. Además, el usuario puede configurar si desea utilizar la dirección MAC real o una aleatoria.

**Limitaciones:** En entornos corporativos o redes que requieren autenticación basada en la MAC, el uso de direcciones aleatorias puede interferir con la seguridad y la gestión de accesos.

Además, tenemos 2 tipos de Aleatorización para estas MAC siendo estos:  
**Aleatorización Persistente:** Este tipo se utiliza de forma predeterminada cuando la aleatorización MAC está habilitada. Se genera una dirección MAC aleatoria basada en los parámetros del perfil de red, como el **SSID (Service Set Identifier)** y el tipo de seguridad y se mantiene constante hasta que se realice un restablecimiento de fábrica. Además, esto permite al dispositivo recordar la conexión permitiendo así evitar el inicio de sesión y habilitar controles parentales.

**Aleatorización No Persistente:** Introducida en Android 12, este tipo se utiliza en situaciones específicas. La dirección MAC se vuelve a aleatorizar al inicio de cada conexión o se utiliza la dirección aleatoria existente, Se **Re-aleatoriza cuando la pasen más de 4 horas desde la última desconexión** y la concesión de DHCP expire o si la dirección se generó hace más de 24 horas, y está aleatorización solo ocurre al inicio de las nuevas conexiones.

# Discusión y conclusiones

El script desarrollado permite la consulta eficiente del fabricante asociado a una dirección MAC, junto con la visualización de los fabricantes de los dispositivos listados en la tabla ARP, facilitando la identificación de dispositivos de manera precisa y eficiente.

Además, el código presentado implementa la capacidad de ejecutar en diferentes sistemas operativos, detectando si este es Windows o Linux/macOS y así modificando el formato de las direcciones MAC si tiene la necesidad. Esto, junto con la capacidad de medir el tiempo de respuesta de las consultas, brinda la opción de realizar un análisis de rendimiento.

Durante el desarrollo del script, se adquirió un mayor entendimiento sobre como interactuar con APIs externas, además de la capacidad de hacer un código flexible en términos de compatibilidad y portabilidad del software a otros sistemas operativos.

Por otro lado, la integración de la tabla ARP resultó un desafío interesante, tanto en la detección, como el manejo de las direcciones MAC diferentes según el sistema operativo. Este fue resuelto mediante pruebas de listas con macs predeterminadas, con el objetivo de analizar el comportamiento del módulo consultarMac() ante el ingreso de varias direcciones.

Otro desafío fue el lograr hacer solicitudes a un API REST para obtener la información sobre los fabricantes de direcciones MAC, lo que finalmente se resolvió leyendo la documentación para desarrolladores presente en el servicio web.

Como posibles mejoras finales, se analiza el implementar más funcionalidades como tablas de dirección IP en el software.

Finalmente, a pesar de estas dificultades, el proceso permitió la comprensión sobre el manejo e interacción con servicios externos, además, el código desarrollado cumple con los objetivos planteados, entregando una herramienta funcional que además nos ofrece nueva experiencia que será fundamental para el diseño de otras aplicaciones más complejas en un futuro.

# Referencias

[1] Tarlogic Security. Uso de direcciones MAC aleatorias. (BSAM-DI-06, s. f.-b). https://www.tarlogic.com/bsam/es/controles/mac-aleatoria-bluetooth/

[2] Implementación de la aleatorización de MAC. (s. f.). Android Open Source Project. https://source.android.com/docs/core/connect/wifi-mac-randomization?hl=es

[3] Privacidad de Wi‐Fi. (s. f.). Apple Support. https://support.apple.com/es-us/guide/security/secb9cb3140c/web