Tarea 02: Consultar Fabricantes de tarjetas de red

Ricardo Silva Pimentel, **ricardo.silvap@alumnos.uv.cl**
Joaquín Martínez, joaquin.martinez@alumnos.uv.cl
Alonso Vera, alonso.vera@alumnos.uv.cl

1. Introducción

La identificación de fabricantes a partir de direcciones MAC es una práctica común en la administración de redes, ya que permite a los administradores conocer la procedencia de los dispositivos conectados. Esta tarea tiene como objetivo desarrollar una herramienta de línea de comandos en Python llamada **OUILookup**, que permite consultar el fabricante de una tarjeta de red a partir de su dirección MAC utilizando una API pública. La importancia de esta herramienta radica en su capacidad para simplificar la gestión de redes y proporcionar información valiosa para la identificación de dispositivos. En este informe, se presentará el problema que se busca resolver, el diseño y la implementación de la solución, así como los resultados de las pruebas realizadas y las conclusiones obtenidas.

2. Descripción del problema y diseño de la solución

La tarea consiste en desarrollar una herramienta de línea de comandos llamada **OUILookup** que permita a los usuarios consultar el fabricante de una tarjeta de red a partir de su dirección MAC. Cada dispositivo de red está identificado por una dirección MAC única, pero esta dirección no proporciona información directamente accesible sobre el fabricante del dispositivo.

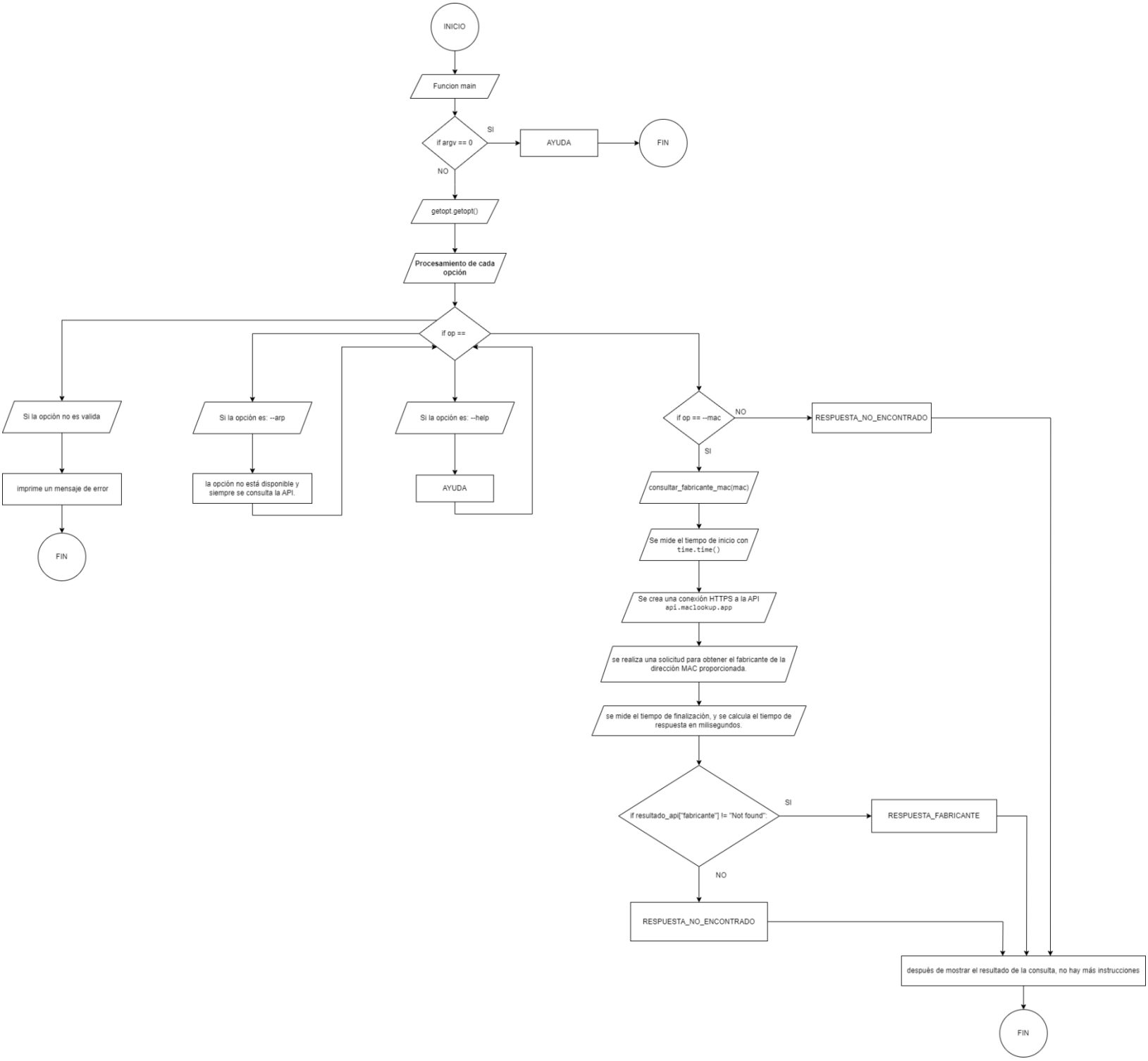
El desafío radica en implementar un codigo que permita a los usuarios ingresar una dirección MAC y recibir rápidamente información sobre el fabricante, utilizando para ello una API REST pública.

Los aspectos específicos del problema incluyen:

1. **Acceso a datos externos**: La herramienta debe interactuar con una API pública, lo que implica gestionar solicitudes HTTP y manejar posibles errores de conexión o respuestas inesperadas.
2. **Interfaz de línea de comandos**: La herramienta debe ser fácil de usar, permitiendo a los usuarios introducir direcciones MAC y recibir información sin complicaciones.
3. **Manejo de errores y respuestas**: Es fundamental que la herramienta maneje adecuadamente casos en los que la dirección MAC no se encuentra en la base de datos de la API, mostrando un mensaje claro al usuario.
4. **Eficiencia en la consulta**: La implementación debe ser lo suficientemente rápida para proporcionar respuestas en tiempo real, minimizando el tiempo de espera para el usuario.

Diseño de la solución

La solución propuesta se basa en una arquitectura sencilla que integra una interfaz de línea de comandos con una API REST pública. El diagrama de clases a continuación ilustra la estructura básica del sistema:



Estructura principal del código

El programa se estructura principalmente en dos funciones clave: una encargada de realizar la consulta a la API y otra responsable de gestionar los argumentos proporcionados por el usuario en la línea de comandos. La combinación de ambas funciones permite que el software sea flexible y útil en la práctica, facilitando al usuario tanto la consulta de fabricantes como la navegación por las opciones del programa.

Consultar_fabricante_mac(): Esta función es el núcleo del programa, ya que realiza la tarea principal: consultar a la API para obtener el fabricante de una dirección MAC. Sin esta función, el programa no cumpliría su propósito.

```python
def consultar_fabricante_mac(mac: str) -> dict:
    inicio = time.time()  # Medir el tiempo de inicio
    conn = client.HTTPSConnection("api.maclookup.app")
    conn.request("GET", "/v2/macs/" + mac)
    respuesta = conn.getresponse()
    fin = time.time()  # Tiempo de fin
    tiempo_transcurrido = round((fin - inicio) * 1000, 3)  # Calcular el tiempo en milisegundos

    if respuesta.status == 200:
        data = loads(respuesta.read())
        fabricante = data.get("company")  # Obtener el fabricante de la respuesta
        # Verificar si la clave "company" existe, si no, retornar "Not found"
        if not fabricante:
            return {"fabricante": "Not found", "tiempo(ms)": tiempo_transcurrido}
        return {"fabricante": fabricante, "tiempo(ms)": tiempo_transcurrido}
    else:
        return {"fabricante": "Not found", "tiempo(ms)": tiempo_transcurrido}
```

Figura 1. Estructura del "Consultar_fabricante_mac()".

Main(): Esta función organiza todo el flujo del programa. Maneja los argumentos de la línea de comandos y decide cuándo ejecutar la consulta o mostrar ayuda al usuario. Actúa como el controlador general del programa.

```python
def main(argv):
    # Si no se proporcionan argumentos, mostrar la ayuda
    if len(argv) == 0:
        print(AYUDA)
        sys.exit(2)

    try:
        opts, args = getopt.getopt(argv, "", ["mac=", "arp", "help"])
    except getopt.GetoptError:
        print(AYUDA)
        sys.exit(2)

    for opt, arg in opts:
        if opt == "--mac":
            # Consultar la API siempre
            resultado_api = consultar_fabricante_mac(arg)
            if resultado_api["fabricante"] != "Not found":
                print(RESPUESTA_FABRICANTE.format(mac=arg, vendor=resultado_api["fabricante"], time=resultado_api["tiempo(ms)"]))
            else:
                print(RESPUESTA_NO_ENCONTRADO.format(mac=arg, time=resultado_api["tiempo(ms)"]))
        elif opt == "--arp":
            print("La opción --arp no está disponible, siempre se consulta la API.")
        elif opt == "--help":
            # Muestra la ayuda
            print(AYUDA)
        else:
            print("Opción no válida. Use --help para ver las opciones.")
            sys.exit(2)
```

Figura 2. Estructura del "Main()".

Direcciones MAC Aleatorias

Las **direcciones MAC aleatorias** son una característica implementada en dispositivos electrónicos para mejorar la privacidad y seguridad del usuario, especialmente en redes Wi-Fi. En lugar de utilizar la dirección MAC única y fija que se encuentra en el hardware del dispositivo, muchos sistemas operativos ahora permiten que los dispositivos generen direcciones MAC temporales al conectarse a redes [2].

- **Funcionamiento**: Esta tecnología permite a los dispositivos crear una nueva dirección MAC cada vez que se conectan a una red Wi-Fi, lo que dificulta el rastreo del dispositivo en diferentes redes. Según Apple, "el uso de direcciones MAC aleatorias ayuda a proteger la privacidad del usuario al ocultar su dirección MAC real" [1].

- **Beneficios de Privacidad**: La aleatorización de las direcciones MAC previene que terceros, como administradores de red o atacantes, rastreen el comportamiento del usuario a través de redes. Esto es especialmente útil en entornos públicos donde la vigilancia es más probable [2].

- **Implementación**: La funcionalidad está disponible en la mayoría de los sistemas operativos modernos, incluyendo **iOS** y **Android**. Los usuarios pueden habilitar o deshabilitar esta opción en la configuración de red de sus dispositivos, aunque en redes empresariales, esta funcionalidad puede interferir con la autenticación basada en MAC [1][2].

- **Consideraciones**: Aunque las direcciones MAC aleatorias mejoran la privacidad, pueden causar problemas en redes que dependen de la autenticación por MAC, lo que puede requerir ajustes adicionales en la configuración de red para permitir el acceso a dispositivos que utilizan direcciones aleatorias [2][3].

3. Implementación

La implementación de OUILookup se realizó utilizando Python, debido a su simplicidad y la disponibilidad de bibliotecas útiles como requests para realizar solicitudes HTTP. A continuación, se describen las partes más importantes del código:

Instalación de Dependencias

Para ejecutar la herramienta, es necesario instalar la biblioteca requests. Esto se puede en la **Figura 1**:

```
PS C:\Users\ricka\Desktop> pip install requests
>>
Collecting requests
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl.metadata (34 kB)
Collecting idna<4,>=2.5 (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests)
  Downloading urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2024.8.30-py3-none-any.whl.metadata (2.2 kB)
Downloading requests-2.32.3-py3-none-any.whl (64 kB)
Downloading certifi-2024.8.30-py3-none-any.whl (167 kB)
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl (100 kB)
Downloading idna-3.10-py3-none-any.whl (70 kB)
Downloading urllib3-2.2.3-py3-none-any.whl (126 kB)
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2024.8.30 charset-normalizer-3.3.2 idna-3.10 requests-2.32.3 urllib3-2.2.3
```

Figura 3. Esta es una figura. El título debe estar centrado

Uso de la Herramienta

Una vez que se ha instalado la biblioteca, se puede utilizar la herramienta **OUILookup** a través de la línea de comandos. A continuación, se presentan algunos ejemplos de uso:

- Para consultar el fabricante de una dirección MAC específica:

```
PS C:\Users\ricka\Desktop> python OUILookup.py --mac 98:06:3c:92:ff:c5
MAC address : 98:06:3c:92:ff:c5
Fabricante : Samsung Electronics Co.,Ltd
Tiempo de respuesta: 613ms
```

Figura 4. La figura muestra la consulta de una MAC especifica.

- Para mostrar los fabricantes de los hosts disponibles en la tabla ARP:

```
PS C:\Users\ricka\Desktop>  python3 OUILookup.py --arp
La opción --arp no está disponible, siempre se consulta la API.
PS C:\Users\ricka\Desktop>
```

Figura 5. Ya que se consulta a la API no es necesaria la tabla ARP.

- Para mostrar la ayuda y conocer las opciones disponibles:

```
PS C:\Users\ricka\Desktop> python OUILookup.py -help
Uso: python OUILookup.py --mac <mac> | --arp | [--help]
--mac: MAC a consultar. Ejemplo: aa:bb:cc:00:00:00.
--arp: muestra los fabricantes de los hosts disponibles en la tabla ARP.
--help: muestra este mensaje.
```

Figura 6. Muestra comando para mostrar los parámetros del comando –help.

4. Pruebas

Las pruebas del código se realizaron para garantizar que todas las funcionalidades trabajaran como se esperaba. Se definieron varios casos de prueba, incluyendo:

Caso de éxito: Consultar una dirección MAC válida.

Resultado esperado: Debería retornar el nombre del fabricante.

```
PS C:\Users\ricka\Desktop> python OUILookup.py --mac 98:06:3c:92:ff:c5
MAC address : 98:06:3c:92:ff:c5
Fabricante : Samsung Electronics Co.,Ltd
Tiempo de respuesta: 623ms
```

Figura 7. Muestra el resultado esperado para una MAC valida.

Caso de no encontrado: Consultar una dirección MAC que no exista en la BD.

Resultado esperado: Debería indicar que el fabricante no fue encontrado.
Prueba 2:

```
PS C:\Users\ricka\Desktop> python OUILookup.py --mac 98:06:3f:92:ff:c5
MAC address : 98:06:3f:92:ff:c5
Fabricante : Not found
Tiempo de respuesta: 604ms
```

Figura 8. Muestra el resultado esperado para una MAC invalida.

También se realizaron pruebas para los macs que serán revisados en la tarea:

```
PS C:\Users\ricka\Desktop> python3 OUILookup.py --mac 98:06:3c:92:ff:c5
MAC address: 98:06:3c:92:ff:c5
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de respuesta: 570.376ms

PS C:\Users\ricka\Desktop> python3 OUILookup.py --mac 9c:a5:13
MAC address: 9c:a5:13
Fabricante: Samsung Electronics Co.,Ltd
Tiempo de respuesta: 476.629ms

PS C:\Users\ricka\Desktop>  python3 OUILookup.py --mac 48-E7-DA
MAC address: 48-E7-DA
Fabricante: AzureWave Technology Inc.
Tiempo de respuesta: 638.686ms
```

Figura 9. Respuesta de los 3 MAC a evaluar.

5. Discusión y conclusiones

Los resultados obtenidos demostraron que **OUILookup** es capaz de consultar correctamente la información del fabricante a partir de direcciones MAC válidas y manejar adecuadamente los errores cuando las direcciones no se encuentran. Se aprendió la importancia de implementar un manejo de errores efectivo para mejorar la experiencia del usuario. Sin embargo, una limitación es que la herramienta depende de la disponibilidad y precisión de la API externa utilizada.

6. Referencias

[1] https://support.apple.com/es-cl/102509
[2] https://www.redeszone.net/noticias/wifi/direccion-mac-aleatoria-wifi/
[3] https://telefonicatech.com/blog/mac-aleaotorias-y-privacidad-parte2
[4] https://maclookup.app/api-v2/documentation