# UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS


Javier Alberto Galindo Parra

Reporte de script para la medición de latencia y ancho de banda

Jorge Ernesto Lopez Arce Delgado

11 de mayo de 2025

# Reporte

## Introducción

El objetivo de este script fue crear una herramienta que me permitiera medir la latencia y el ancho de banda entre varios nodos de una red y visualizar gráficamente esta información para poder interpretarla fácilmente.

Primero decidí que las IPs de los nodos se iban a guardar en un archivo de texto llamado ips.txt, con una IP por línea. Así sería fácil modificar los nodos sin cambiar el código.

Usé la librería argparse para permitir que el usuario del script pudiera configurar:

- El archivo con las IPs (--nodos)
- El puerto donde escucha iperf3 (--puerto-iperf)
- Cuántos paquetes usar para medir el ping (--conteo-ping)
- La duración de la prueba de iperf (--duracion-iperf)

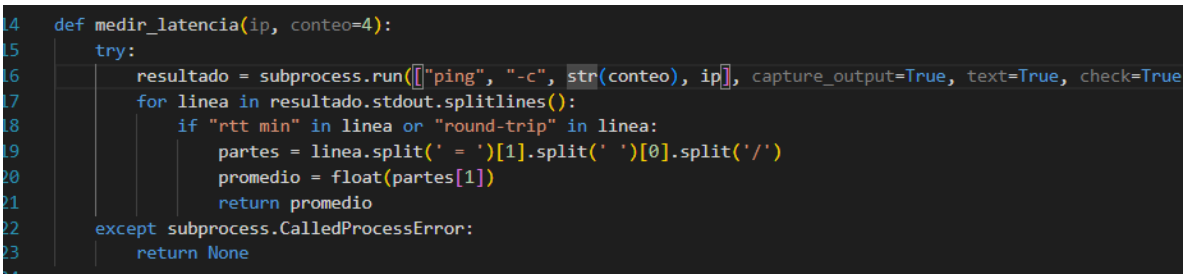
Esto lo agregué en la función main().

Hice varias funciones para que fuera mas fácil de leer el main.

**def medir_latencia(ip, conteo=4):**

Esta función se encarga de medir la latencia (ping) hacia una IP.

- Uso el comando ping desde Python usando subprocess.run.
- Le paso el número de paquetes con -c y la IP.
- Busco la línea que contiene “rtt min” o “round-trip”, y ahí obtengo el promedio de latencia.
- Si algo falla (por ejemplo, si la IP no responde), regreso None.



```
14  def medir_latencia(ip, conteo=4):
15      try:
16          resultado = subprocess.run(["ping", "-c", str(conteo), ip], capture_output=True, text=True, check=True
17          for linea in resultado.stdout.splitlines():
18              if "rtt min" in linea or "round-trip" in linea:
19                  partes = linea.split(' = ')[1].split(' ')[0].split('/')
20                  promedio = float(partes[1])
21                  return promedio
22      except subprocess.CalledProcessError:
23          return None
```

**def medir_ancho_banda(ip, puerto=5201, duracion=10):**

Aquí hago algo parecido pero con iperf3:

- Uso subprocess.run para ejecutar iperf3 como cliente contra la IP que quiero probar.
- Especifico el puerto y la duración de la prueba.
- Busco en la salida de texto la línea que contiene “sender” y “Mbits/sec” para obtener el ancho de banda.
- También regreso None si no hay respuesta o falla la conexión.

```
def medir_ancho_banda(ip, puerto=5201, duracion=10):
    try:
        resultado = subprocess.run([
            "iperf3", "-c", ip, "-p", str(puerto), "-t", str(duracion), "-f", "m"
        ], capture_output=True, text=True, check=True)
        for linea in resultado.stdout.splitlines()[::-1]:
            if "sender" in linea and "Mbits/sec" in linea:
                campos = linea.split()
                ancho = float(campos[-3])
                return ancho
    except subprocess.CalledProcessError:
        return None
```

**def cargar_nodos(ruta_archivo):**

Esta función solo abre el archivo .txt con las IPs y devuelve una lista con las IPs, quitando los espacios o líneas vacías.

En la función main(), por cada nodo de la lista, lo comparo contra todos los demás (excepto consigo mismo) y ejecuto ambas pruebas:

- Latencia (medir_latencia)
- Ancho de banda (medir_ancho_banda)

```
def cargar_nodos(ruta_archivo):
    with open(ruta_archivo) as f:
        return [linea.strip() for linea in f if linea.strip()]

```

Guardo cada resultado como una tupla (origen, destino, latencia, ancho_banda) en una lista metricas.

## Exportación de Resultados

Uso csv.writer para guardar todos los datos en un archivo llamado metricas.csv, con columnas:

- Origen
- Destino
- Latencia (ms)
- Ancho de banda (Mbps)

## Visualización Gráfica

Decidí usar networkx para crear dos grafos dirigidos:

**Grafo de Latencia**

- Cada nodo se conecta a otro con una arista cuyo peso es la latencia en milisegundos.
- El grafo se dibuja en forma circular y se guarda como imagen grafo_latencia.png.

**Grafo de Ancho de Banda**

- Aquí el peso de cada arista es el inverso del ancho de banda (1 / Mbps). Esto es útil si algún día quiero aplicar algoritmos como Dijkstra, donde menor peso significa mejor ruta.
- También lo guardo como imagen grafo_ancho_banda.png.

Ambos grafos se dibujan con matplotlib usando nx.draw y nx.draw_networkx_edge_labels.

## Resultados

- Un archivo metricas.csv con todos los valores medidos.
- Dos imágenes que permiten visualizar la latencia y el ancho de banda entre todos los nodos.