



**UNIVERSIDAD DE GUADALAJARA**

---

**DIVISIÓN DE TECNOLOGIAS PARA LA  
INTEGRACION CIBER-HUMANA**

**DTO. DE CIENCIAS COMPUTACIONALES**

**Actividad: 06, 25A.**

**“Análisis de algoritmos”**

**Optimización de Transferencia de Archivos en  
una VPN con Algoritmos Voraces**

Alumnos:

**NAVARRETE MARTINEZ DAVID ALEJANDRO - Designer/dev.**

**PAEZ MEDRANO KEVIN RANDU - Project Manager/dev.**

**HARO DELGADO IVÁN ALEJANDRO - Developer/Tester.**

Profesor:

**Prof. JORGE ERNESTO LOPEZ ARCE DELGADO**

Guadalajara, Jal.

Marzo, 2025

## Descripción:

### Parte 1. Configuración de la VPN:

Se utilizó la aplicación de hamachi para poder administrar el servidor con los clientes, en este caso VPN\_team es el servidor con el host KRPM, los demás usuarios se unieron como clientes. A cada uno se le asigna una IP estática desde el momento en que se une al servidor, esto lo hace automáticamente el servidor.



Para verificar la conexión exitosa con el servidor podemos realizar la misma latencia manualmente desde nuestro símbolo de sistema con los ips dados como se muestra a continuación: KRPM a LeonovoDavid.

```
C:\Users\Randu>ping 25.59.199.229

Haciendo ping a 25.59.199.229 con 32 bytes de datos:
Respuesta desde 25.59.199.229: bytes=32 tiempo=74ms TTL=128
Respuesta desde 25.59.199.229: bytes=32 tiempo=19ms TTL=128
Respuesta desde 25.59.199.229: bytes=32 tiempo=11ms TTL=128
Respuesta desde 25.59.199.229: bytes=32 tiempo=36ms TTL=128

Estadísticas de ping para 25.59.199.229:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
        (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 11ms, Máximo = 74ms, Media = 35ms
```

## Parte 2. Medición de Métricas de Red

Para medir la latencia entre nodos creamos un script en donde mide automáticamente la capacidad que se tiene entre cada nodo, como se muestra a continuación:

```
def ping_latency(ip):
    result = subprocess.run(["ping", ip, "-n", "4"], capture_output=True, text=True, encoding='cp1252')
    output = result.stdout

    # Buscar "Average", "Promedio" o "Media"
    match = re.search(r"(Average|Promedio|Media)[^\d]*(\d+)\s*ms", output)
    return int(match.group(2)) if match else None
```

En este espacio llama a cada uno de los usuarios para realizar la latencia entre cada uno.

```
& C:/Users/Randu/AppData/Local/P
Latencias.py"
Latencia KRPM → DAVID: 12 ms
Latencia KRPM → IVAN: 13 ms
Latencia DAVID → KRPM: 0 ms
Latencia DAVID → IVAN: 15 ms
Latencia IVAN → KRPM: 0 ms
Latencia IVAN → DAVID: 36 ms
```

Para sacar el ancho de banda usamos la aplicación llamada iperf3 en donde se transmite por medio del listener con el servidor. Aquí una muestra de como es que fuimos sacando cada ancho de banda entre los diferentes nodos.

```
C:\Users\Randu\Escritorio\iperf3.17.1_64\iperf3.17.1_64>iperf3.exe -c 25.59.199.229
Connecting to host 25.59.199.229, port 5201
[ 5] local 25.59.202.144 port 55501 connected to 25.59.199.229 port 5201
[ ID] Interval           Transfer     Bitrate
[ 5]  0.00-1.01      sec    768 KBytes    6.23 Mbits/sec
[ 5]  1.01-2.01      sec    1.00 MBytes    8.39 Mbits/sec
[ 5]  2.01-3.01      sec    896 KBytes    7.36 Mbits/sec
[ 5]  3.01-4.00      sec    1.00 MBytes    8.40 Mbits/sec
[ 5]  4.00-5.00      sec    1.38 MBytes   11.5 Mbits/sec
[ 5]  5.00-6.00      sec    1.00 MBytes    8.40 Mbits/sec
[ 5]  6.00-7.00      sec    1.00 MBytes    8.38 Mbits/sec
[ 5]  7.00-8.00      sec    896 KBytes    7.34 Mbits/sec
[ 5]  8.00-9.01      sec    896 KBytes    7.31 Mbits/sec
[ 5]  9.01-10.01     sec    1.00 MBytes    8.36 Mbits/sec
- - - - -
[ ID] Interval           Transfer     Bitrate
[ 5]  0.00-10.01     sec    9.75 MBytes    8.17 Mbits/sec
[ 5]  0.00-10.02     sec    9.62 MBytes    8.06 Mbits/sec
sender
receiver
```

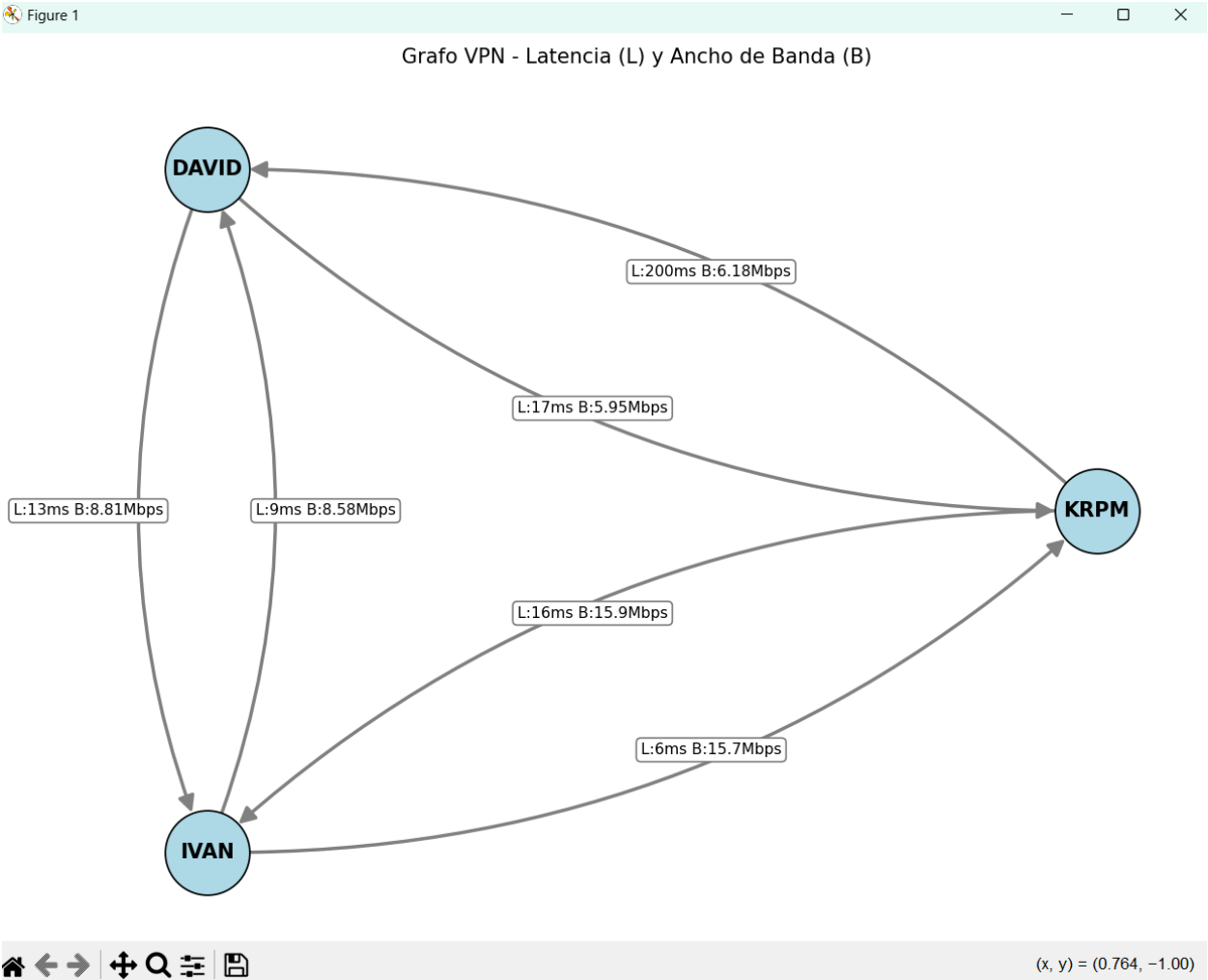
Finalmente, así es como logramos sacar la latencia y el ancho da banda para cada nodo conectado en diferentes direcciones.

Tabla con las métricas obtenidas:

Tabla de Métricas:

Origen	Destino	Latencia	Ancho de Banda
-----			
KRPM	DAVID	200 ms	6.18 Mbps
KRPM	IVAN	16 ms	15.9 Mbps
DAVID	KRPM	17 ms	5.95 Mbps
DAVID	IVAN	13 ms	8.81 Mbps
IVAN	KRPM	6 ms	15.7 Mbps
IVAN	DAVID	9 ms	8.58 Mbps

Grafo visualizado:



### Parte 3. Implementación de Dijkstra ("File Transfer Optimizer") (con GUI)

#### Protocolo Utilizado: TCP (Transmission Control Protocol)

El sistema implementado utiliza el **protocolo TCP**, a través de **sockets en Python**, para la transferencia de archivos entre nodos en una red VPN. Las razones para usar TCP incluyen:

- **Confiabilidad:** TCP asegura la entrega ordenada y completa de los datos.
- **Control de flujo y congestión:** Permite una transmisión adaptativa según las condiciones de la red.
- **Orientación a conexión:** Se establece una conexión persistente entre emisor y receptor.

En el código, se emplea la función `socket(AF_INET, SOCK_STREAM)`, donde `SOCK_STREAM` indica el uso de TCP. El archivo se envía primero enviando su nombre, seguido de los datos binarios en fragmentos de 4096 bytes.

#### Explicación del Algoritmo

El sistema implementado se basa en el uso de algoritmos de optimización sobre grafos para mejorar la eficiencia en la transferencia de archivos dentro de una red VPN. Específicamente, se utiliza el algoritmo de Dijkstra para determinar la ruta más óptima entre el nodo emisor y el nodo receptor. Esta optimización puede realizarse en función de dos criterios: menor latencia o mayor ancho de banda. En el primer caso, el algoritmo considera como peso de las aristas la latencia medida en milisegundos entre los nodos, con el objetivo de minimizar el tiempo de respuesta. En el segundo caso, el peso de cada arista se calcula como el inverso del ancho de banda disponible ( $C=1/BC = 1/BC=1/B$ ), lo que permite que Dijkstra minimice este valor y, por ende, maximice el ancho de banda utilizado. La ruta óptima generada por este algoritmo se utiliza posteriormente para transferir el archivo y comparar el tiempo requerido con respecto a una ruta directa no optimizada. Adicionalmente, se implementa el algoritmo de Kruskal para construir un Árbol de Expansión Mínima (MST) que conecta todos los nodos de la red utilizando el mayor ancho de banda posible entre ellos. Esta topología alternativa permite evaluar la eficiencia global de la red y sugiere una posible mejora estructural en la disposición de enlaces. Ambos algoritmos se integran en una interfaz gráfica (GUI) desarrollada en Tkinter, lo que facilita la selección de archivos, elección de nodos y visualización de rutas y métricas de rendimiento de forma intuitiva.

#### Algoritmo Dijkstra

```
203
204     # Calcular ruta óptima
205     ruta_optima = nx.dijkstra_path(G, nodo_local, destino, weight=peso)
206     ruta_ips_optima = [ips[n] for n in ruta_optima[1:]]
207
208     # === Ruta directa (latencia directa sin Dijkstra) ===
209     latencia_directa = G[nodo_local][destino]['latency']
210
211     # Calcular ruta directa
212     ruta_directa = [destino]
213     ruta_ips_directa = [ips[destino]]
214
215     # Medir tiempos
216     tiempo_optimo = 0
217     for ip in ruta_ips_optima:
218         _, t = medir_transferencia(ip, archivo)
219         tiempo_optimo += t
220
221     tiempo_directo = 0
222     for ip in ruta_ips_directa:
223         _, t = medir_transferencia(ip, archivo)
224         tiempo_directo += t
```

## Proceso:

### 1. Para latencia:

- Considera la latencia como peso directo
- Encuentra el camino con menor suma de latencias

### 2. Para ancho de banda:

- Transforma el ancho de banda a su inverso ( $1/bw$ )
- Encuentra el camino con menor suma de inversos (equivalente a mayor ancho de banda mínimo)

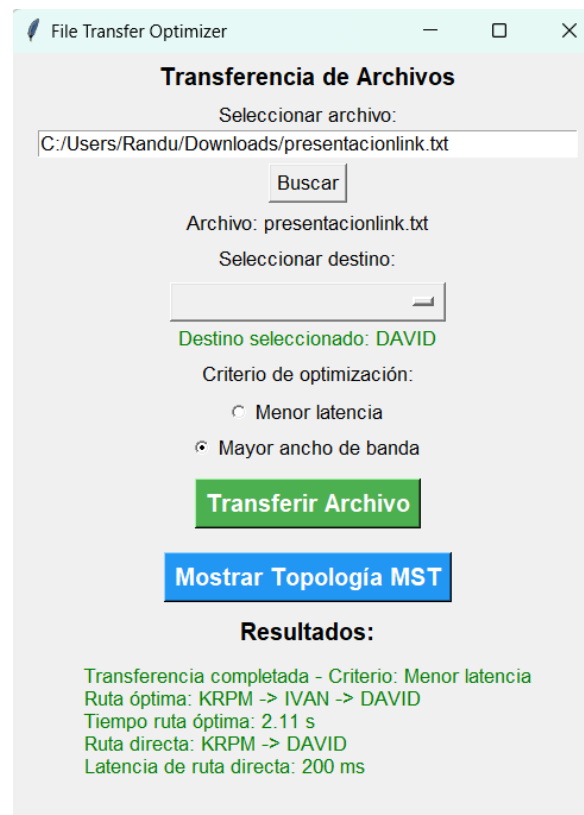


The screenshot shows a window titled "File Transfer Optimizer". Inside, the main heading is "Transferencia de Archivos". Below it, there is a text input field labeled "Seleccionar archivo:" and a "Buscar" button. Further down is another text input field labeled "Seleccionar destino:". Below that, it says "Destino: Ninguno seleccionado" in blue text. Then, there is a section for "Criterio de optimización:" with two radio buttons: "Menor latencia" (unselected) and "Mayor ancho de banda" (selected). Below these are two buttons: a green "Transferir Archivo" button and a blue "Mostrar Topología MST" button. At the bottom of the window, the word "Resultados:" is displayed.



```
C:\Users\Randu\OneDrive\CUCEI\Análisis Algoritmos\prueba>python server.py
Servidor escuchando en 25.23.33.99:5001
Conexión desde ('25.59.202.144', 55862)
Archivo 'presentacionlink.txt' recibido correctamente.
```

## Tiempo de transferencia



The screenshot shows a web application titled "File Transfer Optimizer" with a window icon and a close button. The main heading is "Transferencia de Archivos". Below it, there is a text input field for "Seleccionar archivo:" containing the path "C:/Users/Randu/Downloads/presentacionlink.txt" and a "Buscar" button. The "Archivo:" field displays "presentacionlink.txt". Below that is a "Seleccionar destino:" dropdown menu. The selected destination is "DAVID", shown in green text as "Destino seleccionado: DAVID". There are two radio buttons for the "Criterio de optimización:": "Menor latencia" (selected) and "Mayor ancho de banda". A green "Transferir Archivo" button is below the radio buttons. A blue "Mostrar Topología MST" button is at the bottom. The "Resultados:" section shows green text: "Transferencia completada - Criterio: Menor latencia", "Ruta óptima: KRPM -> IVAN -> DAVID", "Tiempo ruta óptima: 2.11 s", "Ruta directa: KRPM -> DAVID", and "Latencia de ruta directa: 200 ms".

El tiempo de transferencia es un parámetro fundamental que indica cuánto demora en transmitirse un archivo desde un origen hasta un destino a través de la red. Este valor depende principalmente de tres factores: el tamaño del archivo (a mayor tamaño, más tiempo requerido), el ancho de banda disponible (que determina la capacidad máxima de transmisión de datos por segundo) y la latencia de la red (retrasos en la comunicación entre nodos). En el sistema implementado, el tiempo de transferencia se calcula considerando tanto la velocidad de transmisión (basada en el ancho de banda mínimo de la ruta) como los retrasos acumulados de la latencia en cada salto. La optimización de este tiempo es crucial, ya que permite seleccionar la ruta más eficiente, reduciendo significativamente la espera para el usuario final, especialmente cuando se transfieren archivos grandes.

Adicionalmente, se considera el tiempo redondo (RTT), que incluye no solo el envío del archivo sino también la confirmación de recepción, dando una medida más completa del desempeño real de la transferencia.

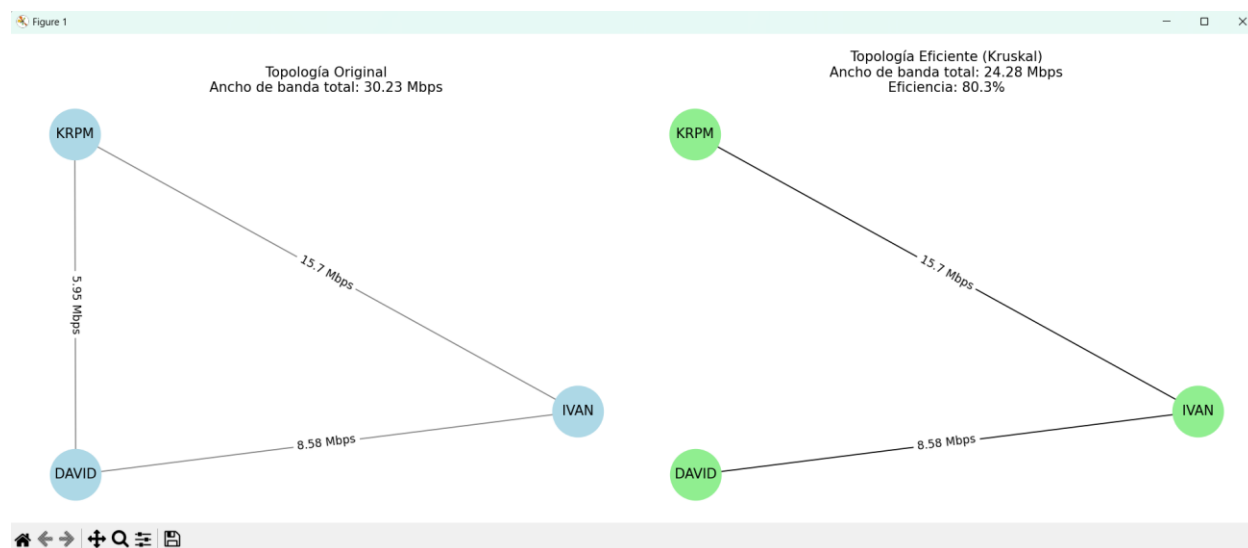
## parte 4. Implementación de Kruskal ("Topología Eficiente")

### Explicación del MST (Árbol de Expansión Mínima)

Dentro del sistema propuesto, se utiliza el algoritmo de Kruskal para generar un Árbol de Expansión Mínima (MST, por sus siglas en inglés) sobre el grafo que representa la topología de la red VPN, donde los nodos corresponden a los equipos conectados y las aristas representan los enlaces de comunicación entre ellos. El objetivo del MST es encontrar una estructura de conexión que una todos los nodos de la red utilizando las conexiones de mayor ancho de banda posible, lo cual se logra minimizando el costo asociado al inverso del ancho de banda ( $C=1/BC = 1/BC=1/B$ ) para cada enlace. Esta transformación permite aplicar Kruskal de manera efectiva, ya que este algoritmo busca minimizar el peso total del grafo resultante.

En el caso específico donde el nodo KRPM desea enviar un archivo a DAVID, se presenta un escenario en el que la ruta directa tiene una latencia considerablemente alta (200 ms). Gracias a la estructura del grafo y la información recopilada, el sistema detecta que una ruta alternativa a través de IVAN ofrece una latencia mucho menor (por ejemplo,  $KRPM \rightarrow IVAN = 6 \text{ ms}$ ,  $IVAN \rightarrow DAVID = 8 \text{ ms}$ ), resultando en una ruta más eficiente en términos de tiempo de transferencia. Esta observación demuestra la importancia de contar con una topología optimizada, donde se aprovechen las mejores conexiones disponibles.

El MST calculado permite visualizar una versión simplificada pero altamente eficiente de la red, eliminando enlaces redundantes y preservando solo aquellos que maximizan el rendimiento global. Esto no solo reduce el consumo de recursos, sino que también puede orientar futuras decisiones de diseño de red, mantenimiento y priorización de rutas en tiempo real. El MST proporciona una representación clara de cómo deberían distribuirse idealmente las conexiones en la red para asegurar una alta eficiencia, siendo especialmente útil en redes con múltiples nodos y variaciones significativas de rendimiento entre enlaces.





## **Conclusión grupal:**

El uso de las ips y vpn es algo “nuevo” que nos adentramos en descubrir la funcionalidad de cada paso en este proyecto, fue más tiempo la investigación en profundizar la funcionalidad de cada parte, este código es una aplicación para transferir archivos entre diferentes dispositivos en una red, optimizando la transferencia según ciertos criterios como la latencia o el ancho de banda.

En este proyecto nos dimos cuenta de que usar algoritmos como Dijkstra y Kruskal realmente marca la diferencia cuando se trata de optimizar una red para transferir archivos de manera eficiente. Por un lado, el algoritmo de Dijkstra nos ayudó a encontrar la mejor ruta para enviar archivos de un nodo a otro con la menor latencia posible, lo cual es súper útil cuando queremos que los archivos lleguen rápido, sin importar tanto el ancho de banda.

Por otro lado, Kruskal nos permitió ver cómo se podría organizar la red de forma más inteligente si lo que más nos importa es aprovechar al máximo las conexiones con mayor ancho de banda. Así, se forma una especie de “mapa ideal” de la red, quitando las conexiones que no aportan mucho y dejando solo las más fuertes.

Como ventaja, estos algoritmos nos permiten tomar decisiones más inteligentes, dependiendo de si queremos velocidad o estabilidad en la conexión. Además, ayudan a visualizar y entender mejor cómo está funcionando la red.

La desventaja es que todo esto requiere medir bien los datos (latencia, ancho de banda, etc.) y que, si alguna conexión cambia o falla, hay que volver a recalcular. También, en redes más grandes, el procesamiento puede volverse más pesado.