Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería
Licenciatura en Informática
Algorítmica y programación II

Laboratorio: Diseño y Análisis de una Red de Computadoras

Alumnos: Bonansea Camaño, Mariano Nicolas.
Rivero, Lucia Jazmín.
Cátedra: Renato Mazzanti, Gustavo Daniel Samec, Marcos zarate
Fecha: 18/06/2024

Introducción:

En el presente trabajo se plantea el diseño y análisis de una red de computadoras que incluye tanto PCs como routers. El mismo se desarrolló mediante el uso de diversas estructuras de datos y Tipos Abstractos de Datos (TADs) así como también interfaces graficas de usuario. El objetivo del sistema desarrollado es el de garantizar la conectividad y optimizar la comunicación entre todas las computadoras en la red.

Planteo del Problema:

Se desea desarrollar una red de computadoras que incluye tanto PCs como routers en la cual se garantice la conectividad y se optimice la comunicación entre todas las computadoras en la red. Además, se le debe permitir al usuario verificar la conectividad de los distintos dispositivos y conexiones de la red, encontrar la ruta de direcciones IP de los paquetes de datos entre dos dispositivos y visualizar todas las conexiones utilizadas en las transmisiones de datos entre los equipos.

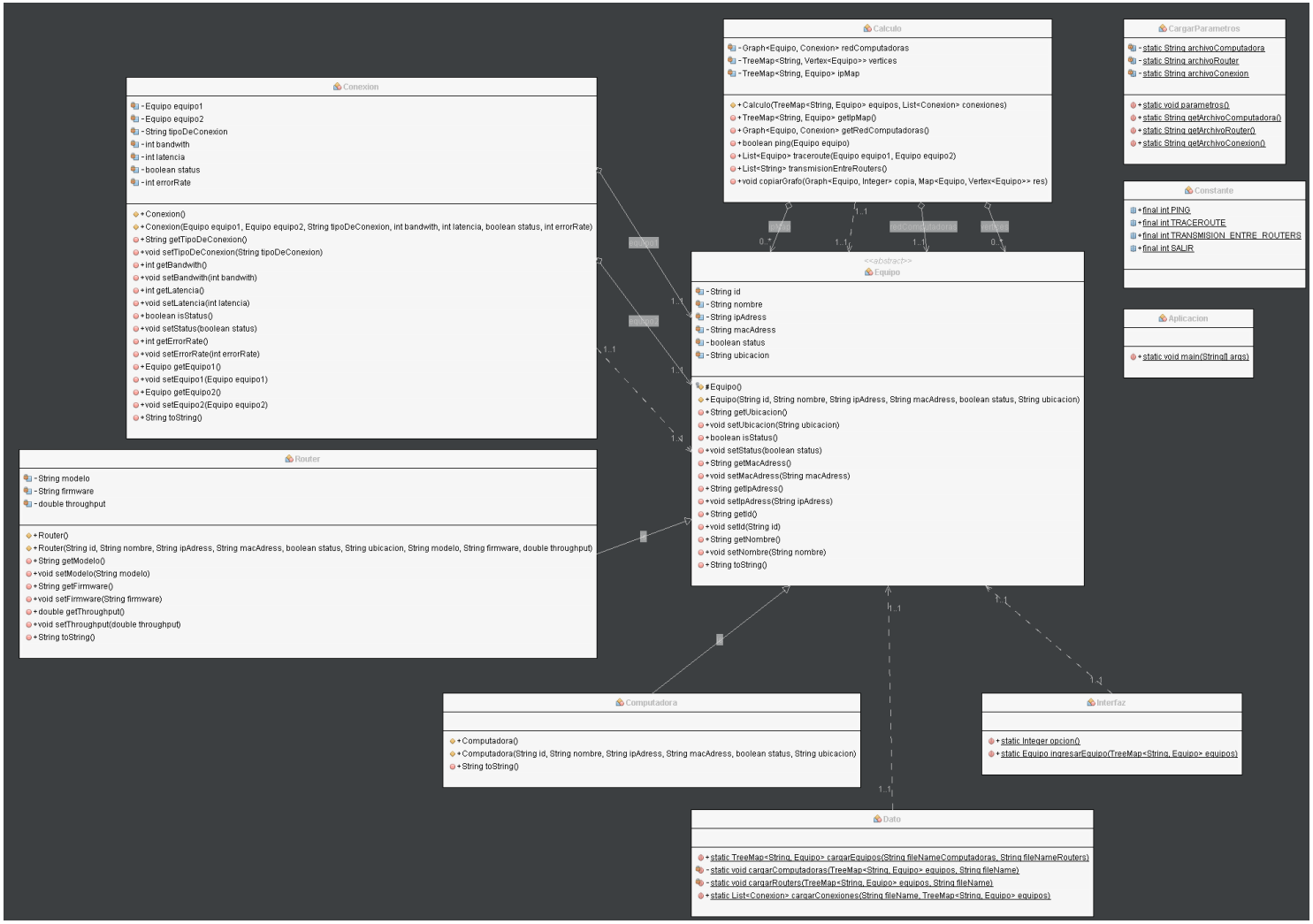
Análisis de las estructuras seleccionadas:

Para el desarrollo de la red de computadoras se utilizó el TAD Grafo, ya que el mismo consta de vértices (nodos) y aristas (conexiones) y es una representación ideal para resolver este problema por los siguientes motivos:

- Representación de la Red: En una red de computadoras, tanto los PCs como los routers pueden representarse como nodos en un grafo, y las conexiones entre ellos pueden representarse como aristas. Esto permite modelar la red de una manera intuitiva y visual.
- Conectividad: Los grafos permiten verificar la conectividad entre los nodos. Por ejemplo, se pueden utilizar algoritmos como el de búsqueda en profundidad (DFS) o búsqueda en amplitud (BFS) para verificar si existe un camino entre dos nodos específicos.
- Optimización de la Comunicación: Los grafos son útiles para optimizar la comunicación en la red. Por ejemplo, pueden utilizarse algoritmos como el de Dijkstra para encontrar el camino más corto entre dos nodos, lo que puede ayudar a minimizar la latencia en la red.
- Ruta de Direcciones IP: Al asignar una dirección IP a cada nodo, es posible utilizar el grafo para rastrear la ruta que un paquete de datos toma a través de la red.
- Visualización de Conexiones: Los grafos proporcionan una forma visual de representar todas las conexiones en la red, lo que puede ser útil para la depuración y el mantenimiento.
- Fácil escalabilidad de la red: A medida que se agregan más dispositivos, la red puede adaptarse sin comprometer el rendimiento ni la conectividad gracias al uso de grafos.

Además, también se utilizaron otros TADs como Mapas y Listas de forma auxiliar complementadas con los grafos para poder realizar diversos algoritmos necesarios para el funcionamiento general del programa como por ejemplo la carga de datos al grafo a partir de archivos de texto.

Diagrama de Clases (UML):



Manual de funcionamiento:

- ## Ingreso de datos:

Los datos a cargar de los distintos routers, computadoras y conexiones son leídos cada uno desde su respectivo archivo de texto asociado, para seleccionar los archivos que corresponderán a cada elemento debe modificarse la ruta del archivo del parámetro deseado en el archivo “config.properties”. Cabe aclarar que los datos son cargados al grafo únicamente a través de este medio, y el mismo no tiene en consideración validaciones en cuanto a que una computadora solo pueda conectarse a un solo router.

Ejemplo archivo config.properties:

config.properties

```
# Archivo de Computadoras
computadora=computadora.txt
# Archivo de Routers
router=router.txt
# Archivo de Conexiones
conexion=conexion.txt
```

Formato de routers en archivo:

id;nombre;ipAdress;macAdress;status;ubicacion;modelo;firmware;throughput;

Ejemplo archivo routers:

router.txt

```
100;routerOficina;192.168.1.1;00:0a:95:9d:68:16;true;OficinaCentral;Cisco RV340;1.0.03.15;900;
200;routerSucursalA;192.168.2.1;00:0a:95:9d:68:17;true;SucursalA;Netgeat Nighthawk R700;V1.0.9.88_10.2.88;1300;
300;routerSucursalB;192.168.3.1;00:0a:95:9d:68:18;true;SucursalB;Tp-Link Archer C7;315.3 build 180305 Rel.51282n;1750;
400;routerSucursalC;1952.168.4.1;00:0a:95:9d:68:19;true;SucursalC;Asus RT-AC68U;3.0.0.4.384_21045;1900;
```

Formato de computadoras en archivo:

id;nombre;ipAdress;macAdress;status;ubicacion;

Ejemplo archivo computadoras:

computadora.txt

```
001;PCPrincipal;192.168.1.100;00:0a:95:9d:68:20;true;Oficina Central;
002;Laptop-01;192.168.1.101;11:22:33:44:55:66;true;Sala de reuniones;
003;PCSecundaria;192.168.2.100;00:0a:95:9d:68:21;true;Sucursal A;
004;Estación-01;192.168.2.101;12:34:56:78:90:AB;true;Área de desarrollo;
005;PCTerciaria;192.168.3.100;00:0a:95:9d:68:22;true;Sucursal B;
006;Portátil-02;192.168.3.101;99:88:77:66:55:44;true;Sala de conferencias;
007;PCCuarto;192.168.4.100;00:0a:95:9d:68:23;true;Sucursal C;
008;PC-02;192.168.4.101;A1:B2:C3:D4:E5:F6;true;Área de soporte;
```

Formato de conexiones en archivo:

idNodoOrigen;idNodoDestino;tipoDeConexion;bandwith;latencia;status;errorRate;

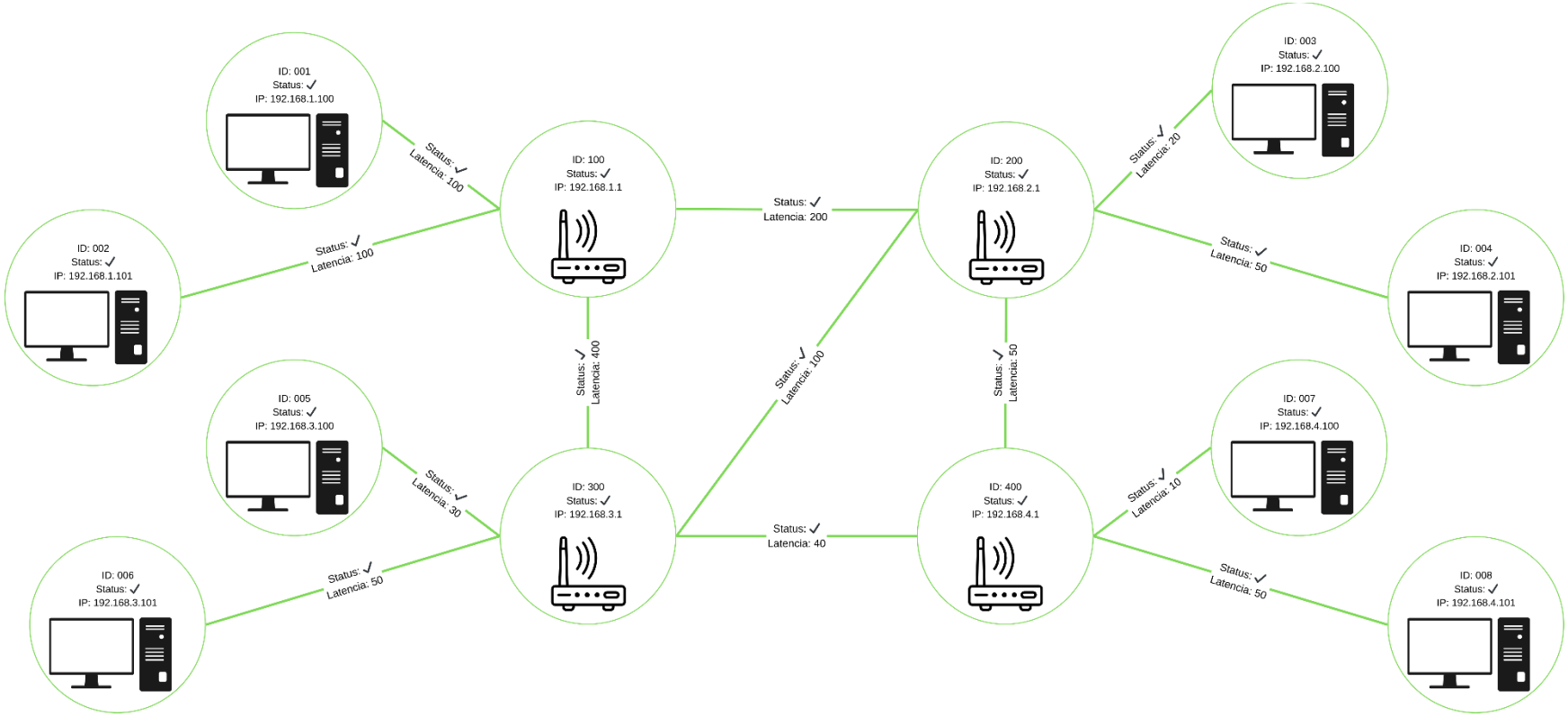
Ejemplo conexiones:

conexion.txt

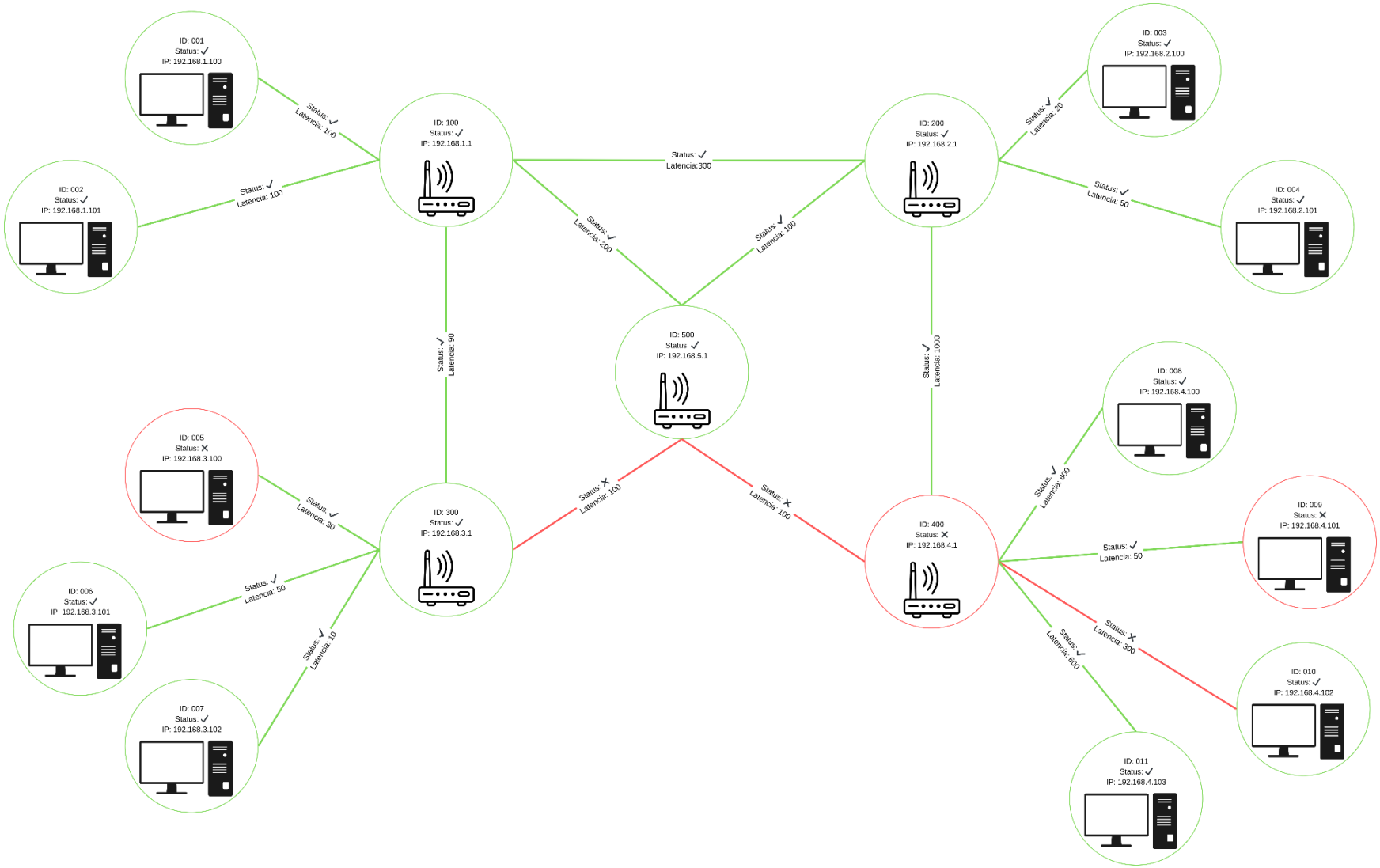
```
001;100;FibraOptica;100;100;true;0;
002;100;FibraOptica;100;100;true;1;
003;200;ADSL;20;20;true;5;
004;200;CableCoaxial;50;50;true;3;
005;300;4GLTE;30;30;true;2;
006;300;Wi-Fi;50;50;true;5;
007;400;DSL;10;10;true;2;
008;400;CableCoaxial;50;50;true;4;
100;200;FibraSimétrica;200;200;true;2;
200;300;Satélite;100;100;true;1;
200;400;CableDeRed;50;50;true;1;
300;100;Satélite;400;400;true;1;
300;400;CableDeRed;40;40;true;1;
```

- **<u>Redes de computadoras cargadas en archivos de ejemplo</u>**:

Red de computadoras 1 (archivos: computadora.txt, router.txt, conexión.txt):



Red de computadoras 2 (archivos: computadora2.txt, router2.txt, conexión2.txt):



Bonansea Camaño, Mariano Nicolas - Rivero, Lucia Jazmín.

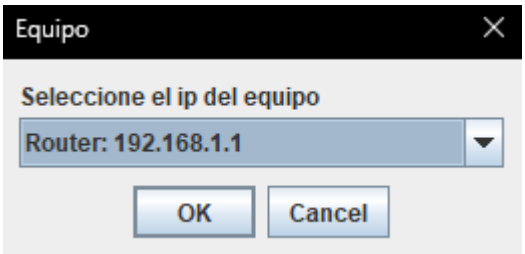
- **Pruebas**:

1- Definir los archivos de texto de routers, computadoras y conexiones que se desean utilizar en el archivo “config.properties”.

2- Ejecutar la clase “Aplicación.java”.

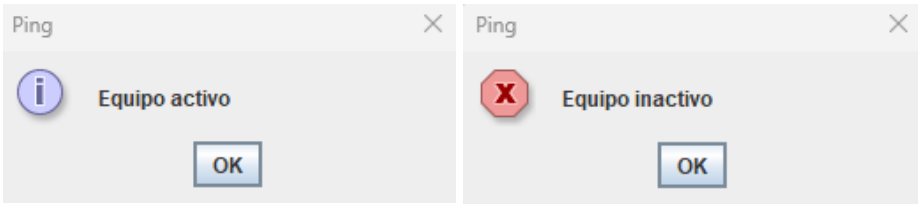
3- Se presentará un menú de opciones a través del cual se pueden realizar las distintas consultas sobre la red de Computadoras creada:

4- Al seleccionar una opción se presentarán una o dos selecciones de equipo/s para realizar la operación seleccionada. En el caso del Ping solo se requiere seleccionar el equipo del que se desea conocer su estatus, y en el caso de traceroute se deberá seleccionar el equipo de origen y el equipo de destino. La única excepción a esta selección es la opción de Transmisión entre routers que no necesita el ingreso de ningún equipo, ya que con solo seleccionarla se visualizara todas las conexiones posibles entre routers.

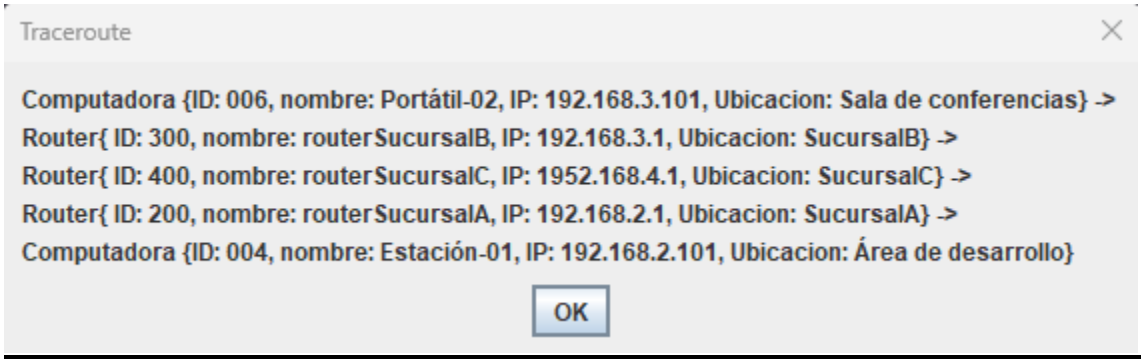


- **Resultados de salida**:

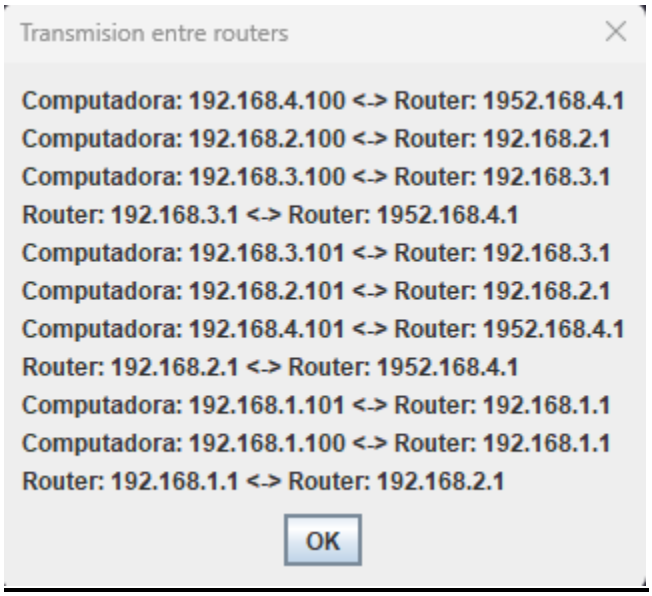
Ping:



Traceroute:



Transmisión entre routers:



- **Notas sobre funcionamiento del programa**:

El peso de los arcos del grafo de copia para realizar las distintas operaciones se ve determinado por la “latencia” de las conexiones especificadas en el archivo de texto de conexiones.

Se opto por la selección de este atributo de conexión para el peso del arco, ya que según la definición de latencia como “el tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino”, esto implica que cuanto menor es la latencia, más rápida es la percepción de la conexión, ya que los datos llegan más rápido. Esto se corresponde directamente con el funcionamiento de los algoritmos necesarios para las operaciones planteadas, de la clase “GraphAlgorithms”, ya que los mismos operan utilizando los arcos de menor peso (a menor latencia mayor velocidad de conexión) y no los de mayor peso.

Si bien desde la catedra se planteo el uso de el ancho de banda como peso de los arcos, este no fue utilizado de esta manera, ya que según la definición de ancho de banda como “la cantidad de datos que pueden ser transmitidos en una conexión de red en un tiempo determinado”, esto implica que a mayor ancho de banda, mayor es la capacidad de la conexión para transferir datos rápidamente, lo que generalmente resulta en una conexión más rápida. Esto no se corresponde con el funcionamiento de los algoritmos de la clase “GraphAlgorithms”, ya que operan de manera opuesta a la definición (a mayor ancho de banda mayor velocidad de conexión). Tomando como ejemplo el caso de traceroute utilizando shortestpath() este retornaría el camino de menor ancho de banda entre dos equipos, por lo que no seria el mas rápido sino el más lento.

Por otro lado, también se planteo el uso de un peso de arco con la fórmula 1/ancho de banda (el cual si se corresponde con el funcionamiento de los algoritmos de la clase “GraphAlgorithms”), pero el mismo tampoco fue empleado debido a que el parámetro necesario de peso de arco para los métodos de GraphAlgorithms es un entero y no un double (Graph<E,Integer>) lo que significa que al realizar la división el resultado se redondeara, dando como resultado en la mayoría de los casos peso 0. Este resultado no demuestra correctamente el funcionamiento de los algoritmos, ya que la mayoría de los arcos tendrían el mismo peso y por lo tanto existirían muchos caminos iguales y no uno más rápido que otros que sea distinguible del resto.

Errores detectados:

- No se realiza ningún tipo de validación a la hora de cargar los archivos en lo que respecta a que elementos pueden estar conectados con otros, es decir por ejemplo que una computadora solo pueda estar conectada a un solo router, sino que las mismas dependen de las especificadas por el usuario en el archivo de texto de conexiones.

Posibles mejoras y extensiones:

Si bien en el presente trabajo no se desarrolló el incremento 3 opcional debido a la complejidad de implementación del algoritmo, así como también la necesidad de refactorización de gran parte del código base, una de las posibles mejoras y extensiones al programa seria la implementación de este a través del algoritmo de Ford-Fulkerson, la cual se detalla a continuación:

1- Realizar una copia del grafo original cargados desde el archivo a un grafo dirigido con el formato Graph<Equipo,Integer>.
   Es un requisito para aplicar el algoritmo de Ford-Fulkerson, ya que el concepto de flujo máximo se basa en la dirección del flujo desde una fuente a un sumidero, y las capacidades de las aristas son direccionales.

2- Aplicar el algoritmo de Ford-Fulkerson a la copia del grafo:
   Inicialización:
   Inicializar el flujo máximo de todos los arcos a 0.
   Búsqueda de caminos aumentantes:
   - Mientras exista un camino aumentante desde la fuente hasta el sumidero en el grafo residual:
     1. Encontrar el camino aumentante:
        - Utiliza una búsqueda en anchura (BFS) para encontrar un camino desde la fuente hasta el sumidero.
        - Si se encuentra un camino, almacenarlo.
     2. Calcular la capacidad residual mínima:
        - Recorrer el camino aumentante encontrado y determina la capacidad residual mínima entre las aristas de este camino.
     3. Aumentar el flujo máximo:
        - Sumar la capacidad residual mínima encontrada al flujo máximo actual.
     4. Actualizar las capacidades residuales:
        - Para cada arista en el camino aumentante, reducir su capacidad por la capacidad residual mínima encontrada.

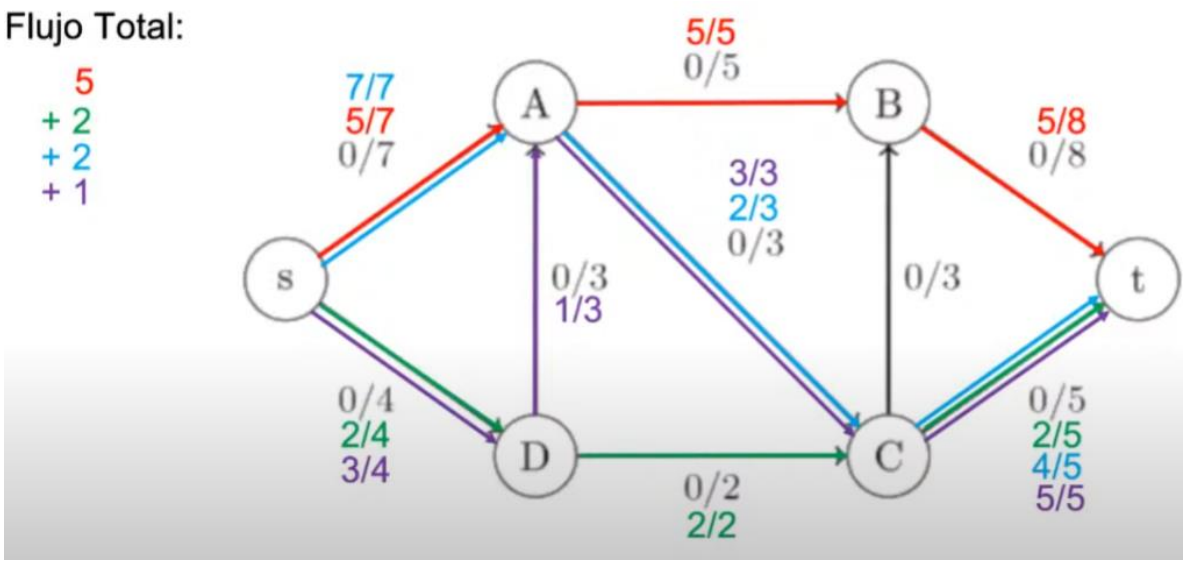
- Aumentar la capacidad de la arista inversa por la misma cantidad (esto es para permitir flujos de retroceso en futuras iteraciones).

Finalización:

- Una vez que no se puedan encontrar más caminos aumentantes, el flujo máximo encontrado es el flujo máximo total en la red.

3- Retornar el resultado entero del flujo máximo total

Ejemplo grafico de la implementación de la solución:



Por otro lado, otras de las posibles mejoras al algoritmo serian el desarrollo de algún tipo de validación a la hora de cargar los datos del grafo desde el archivo, así como también la implementación de una UI que permita al usuario la opción de crear su propia red de computadoras en tiempo de ejecución (en vez de generarla a través de un archivo) y que luego le permita guardarla en formato de archivos de texto para su posterior uso.

Conclusiones:

El uso del TAD Grafo en el desarrollo de esta red de computadoras demostró ser una decisión acertada, proporcionando una base sólida para la conectividad y comunicación eficiente. Los algoritmos de Dijkstra y de Árbol de Expansión Mínima jugaron roles cruciales en la optimización y visualización de la red, asegurando que los datos se transmitieran de manera efectiva y que la estructura de la red fuera clara y manejable.

A través de la estructura de grafo, se aseguró que todos los dispositivos estuvieran conectados de manera eficiente, permitiendo una comunicación fluida y sin interrupciones.

Implementando el algoritmo de Dijkstra, se permitió al usuario encontrar la ruta óptima de direcciones IP para los paquetes de datos entre dos dispositivos cualesquiera en la red. Esto garantizó que los datos siempre tomaran el camino más corto y eficiente, mejorando significativamente el rendimiento de la red.

Se proporcionaron herramientas para que el usuario pudiera verificar la conectividad entre los diferentes dispositivos y conexiones de la red, asegurando que todos los nodos estuvieran correctamente interconectados y operativos.

Y finalmente, utilizando el Árbol de Expansión Mínima del grafo, se permitió la visualización de todas las conexiones más rápidas utilizadas en las transmisiones de datos entre todos los equipos. Esto permitió una mejor comprensión de la topología de la red y ayudó en la identificación de las rutas críticas para la comunicación.