Práctica 5 - Programación: Aplicación a problemas de encaminamiento

1. Introducción

El objetivo de esta práctica es implementar uno de los algoritmos de encaminamiento vistos en clase.

Vamos a suponer la siguiente topología:

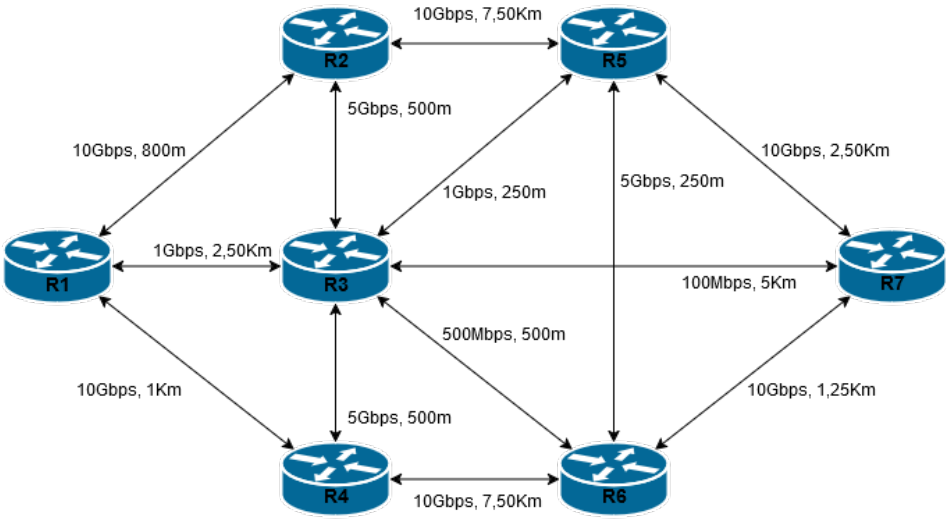

Figura 1 - Topología

En ella tenemos indicados 7 routers (conmutadores de paquetes capa 3) y los enlaces entre ellos. Para cada enlace, se indica su distancia y velocidad (las mismas en ambos sentidos). Utilizaremos estos valores para calcular el coste asociado a cada enlace mediante la siguiente expresión, indicando la velocidad en Mb/s y el retardo de propagación en nanosegundos (puede suponer una velocidad de propagación de 220000 Km/s):

$$C = \frac{T}{BW}$$

Fórmula 1 - Cálculo del coste

Los metadatos de los enlaces deberán leerse desde un fichero de texto (se puede descargar un fichero de ejemplo desde los materiales de la asignatura). En este fichero de texto encontraremos tantas líneas como enlaces en la topología. Cada línea está compuesta de 4 columnas separadas por una tabulación, donde se indican los dos routers que se conectan mediante el enlace en cuestión (1ª y 2ª columna), la velocidad del enlace en Gb/s (3ª columna) y la distancia en metros (4ª columna). Todos los enlaces son bidireccionales y con las mismas características en ambos sentidos.

2. Lectura del fichero

En este apartado se pide leer el fichero de enlaces y almacenar estos datos en una matriz cuadrada donde cada dimensión se corresponda con uno de los routers y el valor sea el coste asociado del enlace.

Asignaremos `R1` al índice `0`, `R2` al índice `1` y así sucesivamente. Cuando un par de routers no tengan enlace entre ellos, se pide almacenar como coste el valor infinito. Cuando ambos routers sean el mismo, se almacenará el valor 0. Puede asumir que el número de routers será siempre el mismo e igual a 7.

Para leer datos de un fichero de texto, puede utilizar las clases FileReader[1] y BufferedReader[2]. El programa deberá leer la ruta al fichero de enlaces como primer argumento. Puede implementar el programa a partir del siguiente ejemplo:

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.util.Arrays;


public class BestRoute {
    public static void main(String[] args) throws Exception {
        float[][] links = readLinks(args[0]);
        for (float[] link: links) {
            System.out.println(Arrays.toString(link));
        }
    }

    public static float[][] readLinks(String filename) throws Exception {
        float inf = Float.POSITIVE_INFINITY;
        float[][] links = {
            {0  , inf, inf, inf, inf, inf, inf},
            {inf, 0  , inf, inf, inf, inf, inf},
            {inf, inf, 0  , inf, inf, inf, inf},
            {inf, inf, inf, 0  , inf, inf, inf},
            {inf, inf, inf, inf, 0  , inf, inf},
            {inf, inf, inf, inf, inf, 0  , inf},
            {inf, inf, inf, inf, inf, inf, 0  },
        };

        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line;
        while ((line = br.readLine()) != null) {
        }

        return links;
    }
}
```

[1] https://docs.oracle.com/javase/8/docs/api/java/io/FileReader.html
[2] https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html

Si utiliza este ejemplo, una vez complete el procesado del fichero de texto deberá obtener la siguiente salida:

```
$ java BestRoute enlaces.txt
[0.0, 0.36363634, 11.363636, 0.45454547, Infinity, Infinity, Infinity]
[0.36363634, 0.0, 0.45454547, Infinity, 3.409091, Infinity, Infinity]
[11.363636, 0.45454547, 0.0, 0.45454547, 1.1363636, 2.2727273, 45.454544]
[0.45454547, Infinity, 0.45454547, 0.0, Infinity, 3.409091, Infinity]
[Infinity, 3.409091, 1.1363636, Infinity, 0.0, 0.22727273, 1.1363636]
[Infinity, Infinity, 2.2727273, 3.409091, 0.22727273, 0.0, 0.5681818]
[Infinity, Infinity, 45.454544, Infinity, 1.1363636, 0.5681818, 0.0]
```

Punto de control 1 (30%): Muestre al profesor el programa indicado.

3. Cálculo de la mejor ruta

A partir del programa anterior, extiéndalo para que calcule la mejor ruta entre dos routers indicados como segundo y tercer argumento. En este caso el programa deberá imprimir por pantalla la ruta seleccionada como una lista de los routers por los que se pasa para ir de origen a destino (ambos incluidos).

Para el cálculo de esta ruta deberá utilizar el algoritmo de Bellman-Ford o el de Dijkstra, según su preferencia.

Para que puedan probar si el funcionamiento es el adecuado, la salida deberá ser, por ejemplo:

```
$ java BestRoute enlaces.txt R1 R7
R1 R2 R3 R5 R6 R7
```

Punto de control 2 (50%): Muestre al profesor el programa indicado.

Ayuda: Si su programa está funcionando de una forma no esperada puede utilizar el fichero `ejemplo_clase.txt`, que describe la topología del ejemplo visto en las transparencias de las clases de teoría. Con este ejemplo debería poder comparar el progreso del programa en cada iteración. Tenga en cuenta que deberá ajustar su programa para funcionar con un número diferente de equipos.

4. Cálculo alternativo

Repita el funcionamiento del programa anterior, pero esta vez utilizando el algoritmo que no utilizó en la anterior implementación.

Punto de control 3 (20%): Muestre al profesor el programa indicado.