Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2015-2016

Lab work #2. Introduction to Net2Plan algorithms development

(1 session)

Author:

Pablo Pavón Mariño

1 Objectives

The goals of this lab work are:

1. Introduce the concept of Net2Plan algorithm, and how to run them in Net2Plan.
2. Introduce the development of new offline design algorithms for Net2Plan.

2 Duration

This lab work is designed for one session of two hours.

3 Evaluation

This lab work has been designed to guide the students in their learning of Net2Plan. The annotations the students make in this document are for their use when studying the course, and do not have to be delivered to the teacher for evaluation.

4 Documentation

The resources needed for this lab work are:

- Net2Plan tool and their documentation (see `http://www.net2plan.com/`).
- Instructions in this wording.

5 Previous work before coming to the lab

- Read Chapter 5 of the users manual.
- Read the Javadoc information of:
  - Interface `IAlgorithm`
  - Classes `NetPlan`, `Node`, `Link`, `Demand`, `Route`.

  All of them are in the package `com.net2plan.interfaces.networkDesign`.

6 Setup Eclipse for developing Net2Plan algorithms

The student can use any IDE (Integrated Developer Environment) for programming in Java at home. However, the computers in the lab have Eclipse installed, and this section will briefly guide the student in how to setup Eclipse to develop Net2Plan algorithms with it. These explanations correspond to Eclipse Mars 1 Release 4.5.1. Other Eclipse versions may have slightly different menu and option names.

The steps to follow are:

1. Open Eclipse.
2. *Create a Eclipse Java project*: Create a new Java project in menu *File* → *New* → *Java project*. As an execution environment, use Java 7 or above. Write down the *location* of the project, since it is where the Java and class files are placed: we will need this information later.
3. *Add the Net2Plan libraries to the project*: To be able to write Net2Plan algorithms and build them (compile them), the created Eclipse project needs to know where the Net2Plan libraries are. This can be done in different forms. For instance, right-click in the project name, and then use the option *Build path...* → *Configure Build Path*. Then, in the tab *Libraries*, click the button *Add External JARs...*. Navigate through the folders and go to *lib* folder in your Net2Plan installation[1]. Then, add to the build path *all* the `.jar` files there.
4. *Install the solvers*: In future lab sessions (not in this), you will need access to the GLPK and IPOPT solvers. Unfortunately, we still could not make IPOPT run in Linux, and we will need to use Windows in the lab. To install these solvers just:
   - Check if the Java virtual machine you are using in your computer is of 32 or 64 bits. You can do this with the command `java -version` in a console.
   - Download the GLPK and IPOPT `dll` files for the 32 or 64 bits version, the one of your Java virtual machine, as shown in:

     `http://net2plan.com/jom/installation.php`

   - Copy them in any directory. Placing it in *Windows/System32* folder, makes it immediately accessible to Net2Plan. Other option is placing it in the same folder where `Net2Plan.jar` is. Then, in the menu *File* → *Options*, set the *glpkSolverLibraryName* and *ipoptSolverLibraryName* parameters accordingly.

6.1 Open the documentation for developing

It is of the greatest importance that students get used to program Net2Plan in Java, making a smart use of the documentation available. Then, before starting any development, the students should:

1. Open the Net2Plan users guide (pressing `F1` from Net2Plan).
2. Open the local version of Net2Plan Javadoc, describing the Net2Plan libraries. Do this from the menu *Help* → *Library API Javadoc*.
3. Open the local version of standard Java Javadoc, describing the Java lirbaries. It is accessible from the installed browser *bookmarks*.

It is important that the students get familiar with using this documentation while programming, and keep them always open in separate browsers. This documentation will be the only available resource during the lab exams.

7 Create your first Net2Plan algorithm

In this course, all the offline Net2Plan algorithms developed will start adding/modifying code to a simple template provided: `AlgorithmTemplate.java`.

Add this file to the project as follows:

[1] If we install Net2Plan so that the `Net2Plan.jar` file is in *Net2Plan-0.4.0*, then the *lib* folder is in *Net2Plan-0.4.0/lib*

1. Copy it to the `src` folder inside your Eclipse project folder. Then, press `F5` or update the Eclipse project, to make Eclipse automatically incorporate the file to it. Alternatively, you can drag-and-drop the file to the `src` folder of the Eclipse project. Then, Eclipse will ask whether you want to *copy* the file or *link* it. Choose *copy*.
2. Open the `AlgorithmTemplate.java` file in Eclipse. Change the package declaration to the package where you have copied it.
3. Build your project. It should have no errors, and produce a `.class` file. This file is available in the `bin` folder, next to `src` folder of the Eclipse project.
4. To run your offline algorithm in Net2Plan you can: (i) drag-and-drop it to the *Algorithm execution* tab, or (ii) load the class file in the `bin` directory using the *Load* button.

The template algorithm does nothing but printing an "Ok" message in the screen.

7.1 Modify the algorithm

1. What is the algorithm description shown in Net2Plan? Identify this text in the Java file. What function of the file returns the description to print?

   Change the algorithm description, and build again the file. To update the description in the GUI, you have to reload it.
2. Look at the algorithm parameters in Net2Plan. Identify the part of the Java file where this information is set. Add a new parameter called *newIntegerParam*, with a default value of "7", and a description "I am new". Write the code that reads in the algorithm the value of *newIntegerParam* passed, parses it as an integer and stores it in the integer variable *newIntegerParam*.
3. Modify the algorithm so that it prints in the `System.out`, the current values of the parameters passed. How can we see the messages that the algorithms print in `System.out`?

   Call the algorithm with different values for the input parameters. See how these values are shown in the Java console.

8 Second algorithm

Starting from the file `AlgorithmTemplate.java`, the students will create a more complex Net2Plan algorithm, which now creates a topology of links and nodes, with demands and routes.

For this, please first copy the `AlgorithmTemplate.java` template to the project. Then, rename it to `MyFirstAlgorithm.java`.

Recall that in Java the file name should be equal to the class name. You can rename the file in Eclipse by clicking in its name, and then pressing `F2`*. Eclipse will rename both the file and the class name, constructors etc. inside the Java file.*

The student should complete the following steps:

- The algorithm must have one and only input parameter called *linkCapacity*, with a default value of 1.0, and a description "This is the capacity to place in all the links".
- The algorithm should first remove all the existing nodes in the input design.

  Use the `removeAllNodes` function in `NetPlan` object (see the Javadoc!!!).

- The algorithm should create three nodes, placed at the positions (0,0), (10,0), (5,5). Use the node names *nodeA*, *nodeB* and *nodeC*. The nodes will have no attributes.

  Use the `addNode` function in `NetPlan` object (see the Javadoc!!!).

- The algorithm should connect the nodes in a triangle, with two links in opposite directions (this means a total of six links). Set all the link lengths equal to 10 units, and a propagation speed of 200000 km per second. The link capacity is made equal to the value of the input parameter *linkCapacity*.

  Use the `addLink` function in `NetPlan` object (see the Javadoc!!!). In this and in ALL the functions in this course, never use the optional parameters of the type *NetworkLayer*. They are applicable only in multilayer designs, and in this course all the designs are single-layer.

  Develop, compile and run the algorithm. It should produce a design as the one shown in Fig. 1.

- Print in the Java console the index and identifier (*Id*) of all the created links.

  Use the `getIndex` and `getId` methods of the returned *Link* objects.

  Note that the indexes all start in zero and are consecutive, while the identifiers are more arbitrary serial numbers.

- Add one traffic demand starting in *nodeA* and ending in *nodeB*, with 5 traffic units.

  Use the `addDemand` function in `NetPlan` object (see the Javadoc!!!).

  Run the algorithm and check that the demand is created. Note that the traffic is not carried.

- Set the traffic routing type to SOURCE ROUTING, specifying that the traffic will be carried using *Route* objects (instead of forwarding rules).

  Use the `setRoutingType` function in `NetPlan` object (see the Javadoc!!!).

- Add a route to the design, associated to the demand that was just created. The route should carry 2 units of traffic, occupying 2 units of capacity in each link traversed. The path is composed of just the direct link between the nodes.

  Use the `addRoute` function in `NetPlan` object (see the Javadoc!!!).

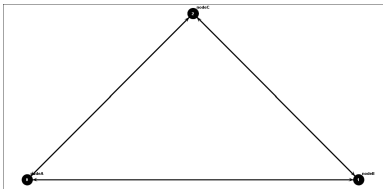

Figure 1: Network.

Run the algorithm and check that the route is created. Note that since the demand offers 5 units, and only 2 are carried yet, there is still a blocking of 60%.

- Add a second route to the design, associated to the same demand. The route should carry 3 units of traffic, occupying 3 units of capacity in each link traversed. The path is composed of the two links $nodeA \rightarrow nodeC \rightarrow nodeB$. Run the algorithm and check that the route is created. Note that the demand is now fully carried (0% blocking).

9 Now on your own

Quiz 1. Modify the previous algorithm to print in the Java console all the network nodes, with its index and its name. For this, use two forms:

- Using the *getNodes* method in `NetPlan` object in a for loop like:

```
1  for (Node n : netPlan.getNodes())
```

- Taking the nodes from the indexes, in a for loop like:

```
1  for (int indexN = 0; indexN < netPlan.getNumberOfNodes() ; indexN
       ++)
2  {
3    Node n = netPlan.getNode(indexN);
4    ...
5  }
```

Quiz 2. Create an algorithm that receives an input design, removes all the links in it, and adds one link between each node pair. All the links will have the same capacity, given by the input parameter of the algorithm *linkCapacity*.

> For removing the links, use the `removeAllLinks` function in `NetPlan` object (see the Javadoc!!!).

Quiz 3. Create an algorithm that receives an input design, and the index of a node as an input parameter (*hubIndex* parameter, with default value 0). Then, it will remove all the links in the input design, but those starting or ending in the *hubIndex* node.

> For removing the links, use the `remove` function in `Link` object (see the Javadoc!!!).

Quiz 4. Create an algorithm that receives an input design, removes all the nodes in it, and then creates a number of nodes given by the input parameter *numNodes* (defaults to 10). The X and the Y positions of each node are chosen randomly using a uniform distribution between 0 and 10.

See the *java.util.Random* class in the Java release (see the Javadoc!!!).

10 Work at home after the lab work

The student is encouraged to complete all the *Quizs* that he/she could not finish during the lab session.