Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2018-2019

Práctica 2. Introducción al desarrollo de algoritmos de Net2Plan

(1 sesión)

Autor:

Pablo Pavón Mariño
Juan Pedro Muñoz Gea
María Victoria Bueno Delgado
Ángel Antonio Pintado Sedano
Juan Carlos Sánchez Aarnoutse

1 Objetivos

Los objetivos de esta práctica son:

1. Introducir el concepto de algoritmos de Net2Plan, y como ejecutarlos en Net2Plan .
2. Introducir el desarrollo de nuevos algoritmos offline de diseño para Net2Plan .

2 Duración

Esta práctica está diseñada para una sesión de dos horas

3 Evaluación

Esta práctica ha sido diseñada para guiar al estudiante en su aprendizaje en Net2Plan. Las anotaciones que los estudiantes hagan en este documento son para su uso cuando repasen la asigantura y no es necesario que se entregue al profesor para su evaluación.

4 Documentación

Los recursos necesarios para este trabajo de laboratorio son:

- La herramienta Net2Plan y su documentación (ver `http://www.net2plan.com/`).
- Las instrucciones de esta práctica.

5 Trabajo previo antes de venir al laboratorio

- Leer el capítulo 5 del manual del usuario.
- Leer la información de Javadoc:
  1. Interfaz IAlgorithm
  2. Clases NetPlan , Node , Link , Demand , Route

Todos ellos se encuentran en el paquete com.net2plan.interfaces.networkDesign

6 Configurar Eclipse para desarrollar algoritmos Net2Plan

El estudiante puede utilizar cualquier IDE (Integrated Developer Enviroment) para la programación en Java en casa. Sin embargo en los PCs del laboratorio se ha instalado Eclipse, y en esta sección guiará brevemente al estudiante sobre cómo configurar Eclipse para desarrollar algoritmos Net2Plan. Los menús y opciones indicadas a continuación se corresponden con la versión de Eclipse Marte 1 versión 4.5.1 . Otras versiones de Eclipse pueden tener nombres de menús y opciones un poco diferentes.

Los pasos a seguir son los siguientes:

1. Abrir Eclipse.
2. Crear un proyecto Java Eclipse: Crear un nuevo proyecto Java en el menú File->New->Java project . Como entorno de ejecución utilice Java 7 o superior. Anote la ubicación del proyecto ya que es donde se ubicarán los ficheros class y ficheros Java: utilizaremos esta información más adelante
3. Añadir las bibliotecas Net2Plan al proyecto: Para ser capaz de escribir algoritmos Net2Plan y construirlos (compilarlos), el proyecto Eclipse creado necesita saber dónde se encuentran las librerías de Net2Plan. Esto se puede hacer de diferentes maneras . Por ejemplo , haga clic en el nombre del proyecto, y a continuación, utilizar la opción Build path.... -> Configure Build Path. Luego, en la pestaña Libraries haga clic en el botón Add External JARs..... Navegar por las carpetas e ir a la carpeta lib de su instalación de Net2Plan(*) . A continuación, añadir a la ruta todos los archivos .jar.

   (*) Si instalamos Net2Plan de tal forma que el fichero Net2Plan.jar se encuentra en Net2Plan-0.5.0, entonces la carpeta lib se encuentra en Net2Plan-0.5.0/lib
4. Instalar los solvers: En futuras sesiones de laboratorio (no en esta), será necesario tener acceso a los solvers GLPK e IPOPT . Desafortunadamente , todavía no se puede ejecutar el solver IPOPT en Linux , y necesitaremos utilizar Windows en el laboratorio . Para instalar estos solvers simplemente:
   - Compruebe si la máquina virtual de Java que está utilizando en su computadora es de 32 o 64 bits . Esto se puede hacer esto con el comando java -version en una consola.
   - Descargar los ficheros dll del GLPK y IPOPT para la versión de 32 o 64 bits , la que corresponda a su máquina virtual Java, como se muestra en:

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

   - Copiarlos en cualquier directorio. Si lo coloca en la carpeta Windows / System32 , hace que sea inmediatamente accesibles a Net2Plan. Otra opción es colocarlos en la misma carpeta donde se encuentra Net2Plan.jar. Luego , en el menú File -> Options seleccionar los parámetros lpkSolverLibraryName e ipoptSolverLibraryName en consecuencia

6.1 Abrir la documentación para desarrollar

Es muy importante que los alumnos se acostumbren a programar Net2Plan en Java, haciendo una utilización inteligente de la documentación disponible. Entonces, antes de iniciar cualquier desarrollo, los estudiantes deben:

1. Abrir la guía de usuarios Net2Plan (presionando F1 en Net2Plan).
2. Abrir la versión local de Net2Plan Javadoc , que describe las librerías de Net2Plan . Para ello, desde el menú Help ->Library API Javadoc
3. Abra la versión local de estándar de Java Javadoc, que describe las librerías de Java . Está accesible desde los favoritos del navegador instalado.

   Es importante que los estudiantes se familiaricen con el uso de esta documentación durante la programación , y mantenerla siempre abierta en navegadores separados . Esta documentación será el único recurso disponible durante los exámenes de laboratorio.

7 Crear el primer algoritmo Net2Plan

En este curso, todos los algoritmos offline de Net2Plan a desarrollar consistirán en a añadir / modificar el código a una plantilla sencilla proporcionada : AlgorithmTemplate.java . Añadir este fichero al proyecto de la siguiente manera :

1. Copiarlo a la carpeta src dentro de la carpeta del proyecto Eclipse . A continuación , pulse F5 o actualice el proyecto Eclipse, para hacer que Eclipse incorpore automáticamente el fichero. Como alternativa, puede arrastrar y soltar el fichero en la carpeta src del proyecto Eclipse . Entonces , Eclipse le preguntará si desea copiar el archivo o enlazarlo.Elija copiar.
2. Abra el fichero AlgorithmTemplate.java en Eclipse . Cambiar la declaración del paquete al paquete en el que ha copiado.
3. Compila tu proyecto. No debe tener ningún error , y además producirá un fichero .class . Este fichero estará disponible en la carpeta bin , junto a la carpeta src del proyecto Eclipse.
4. Para ejecutar tu algoritmo offline en Net2Plan puede: (i) arrastrar y soltar en la pestaña Algorithm execution , o (ii) cargar el fichero .class clase en el directorio bin utilizando el botón Load.

   La plantilla del algoritmo no hace nada más que imprimir un mensaje de "OK" en la pantalla .

7.1 Modificando el algoritmo

1. .¿Cuál es la descripción del algoritmo que se muestra en Net2Plan ? Identificar este texto en el fichero Java . ¿Qué función del fichero le devuelve la descripción que se imprime?

   Cambiar la descripción del algoritmo , y compile de nuevo el archivo. Para actualizar la descripción en el GUI ,tiene que volver a cargarlo.
2. Comprueba los parámetros del algoritmo en Net2Plan . Identificar la parte del fichero Java donde se establece esta información. Añadir un nuevo parámetro llamado newIntegerParam , con un valor por defecto de 7 , y una descripción de "I am new". Escribir el código que lee en el algoritmo el valor que se le pasa a newIntegerParam, lo analiza como un entero y lo almacena como un número entero en newIntegerParam variable.
3. Modificar el algoritmo para que se imprima en el System.out , los valores actuales de los parámetros de entrada . ¿Cómo podemos ver los mensajes que los algoritmos se imprimen en System.out?

   Llamar al algoritmo con distintos valores para los parámetros de entrada . Vea cómo se muestran estos valores en la consola Java.

8 Segundo algoritmo

A partir del fichero AlgorithmTemplate.java , los estudiantes crearán un algoritmo Net2Plan más complejo, que creará una topología de enlaces y nodos, con demandas y rutas . Para ello, por favor, primero copie la plantilla AlgorithmTemplate.java al proyecto. A continuación , cambie su nombre a MyFirstAlgorithm.java .

Recordemos que en Java el nombre de archivo debe ser igual al nombre de la clase. Usted puede cambiar el nombre del archivo en Eclipse haciendo clic en el nombre, y luego pulsando F2 . Eclipse renombrará tanto el archivo como el nombre de la clase, constructores , etc. dentro del archivo Java.

El alumno debe completar los siguientes pasos:

1. El algoritmo debe tener uno y solo un parámetro de entrada único llamado linkCapacity, con un valor por defecto de 1.0, y la siguiente descripción: "Esta es la capacidad para de todos los enlaces"

2. El algoritmo debe eliminar primero todos los nodos existentes en el diseño de entrada. Utilice la función removeAllNodes en el objeto NetPlan (consulte el Javadoc !!!).

   Utilice la función addNode en el objeto NetPlan (consulte el Javadoc !!!).

3. El algoritmo debe crear tres nodos, colocados en las posiciones (0,0), (10,0), (5,5). Utilizar los nombres nodoA, nodoB y nodoC . Los nodos no tendrán atributos.

4. El algoritmo debe conectar los nodos en un triángulo, con dos enlaces en direcciones opuestas (esto significa un total de seis enlaces). Establecer todas las longitudes de los enlaces iguales a 10 unidades, y una velocidad de propagación de 200000 km por segundo. La capacidad de enlace se será la que se haya introducido en el parámetro de entrada linkCapacity.

   Utilice la función de AddLink del objeto NetPlan (consulte el Javadoc !!!). En este y en TODAS las funciones a desarrollar durante este curso, nunca utilizar los parámetros opcionales del tipo NetworkLayer. Sólo están disponibles en diseños de múltiples capas, y en este curso todos los diseños son de una sola capa.

   Desarrollar, compilar y ejecutar el algoritmo. Debe producir un diseño como el que se muestra en la Fig. 1

5. Imprimir en la consola de Java el índice y el identificador (Id) de todos los enlaces creados.

   Utilice los métodos GetIndex y getId de los objetos Link devueltos.

   Tenga en cuenta que todos los índices comienzan en cero y son consecutivos, mientras que los identificadores son una serie de números más arbitrarios.

6. Añadir una demanda de tráfico que parta del nodoA y termine en el Nodo B, de 5 unidades de tráfico.

   Utilice la función de addDemand del objeto NetPlan (consulte el Javadoc !!!).

   Ejecutar el algoritmo y comprobar que se crea la demanda. Comprobar que el tráfico no se cursa.

7. Establecer el tipo de tráfico de enrutamiento a SOURCE ROUTING, especificando que el tráfico será cursado utilizando el objeto Routes (en lugar de utilizar las reglas de reenvío).

   Utilice la función de setRoutingType del objeto NetPlan (consulte el Javadoc !!!).

8. Añadir una ruta al diseño, asociada a la demanda que se acaba de crear. La ruta debe cursar 2 unidades de tráfico, que ocupan 2 unidades de capacidad en cada enlace atravesado. El camino

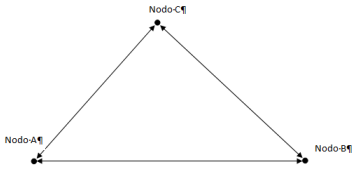

Figure 1: `Network`

se compone de solamente el enlace directo entre los 2 nodos. Utilice la función de addRoute del objeto NetPlan (consulte el Javadoc !!!).

Ejecutar el algoritmo y comprobar que se crea la ruta . Tenga en cuenta que, dado que la demanda ofrece 5 unidades , y sólo cursan 2 unidades , aún hay un bloqueo del 60%

9. Añadir una segunda ruta al diseño, asociada a la misma demanda . La ruta debería cursar 3 unidades de tráfico, ocupando 3 unidades de capacidad en cada enlace atravesados. El camino se compone de 2 enlaces nodo A -> nodo B -> nodo C. Ejecutar el algoritmo y comprobar que se crea la ruta. Comprobar que la demanda se cursa completamente (0% de bloqueo).

9 Ahora por tu cuenta

Ejercicio 1: Modificar el algoritmo anterior para imprimir en la consola de Java todos los nodos de la red , con su índice y su nombre . Para ello, utilice dos formas:

- Utilizando el método de getNodes en el objeto NetPlan en un bucle for como:

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

- Tomando los nodos a partir de los índices , en un bucle for como:

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

Ejercicio 2: Crear un algoritmo que recibe un diseño de entrada , eliminar todos los enlaces en el mismo , y añade una enlace entre cada par de nodos . Todos los enlaces tendrán la misma capacidad , propuesta por el parámetro de entrada del algoritmo linkCapacity

Para la eliminación de los enlaces , utilice la función de removeAllLinks del objeto NetPlan (Consulte el Javadoc !!!) .

Ejercicio 3:Crear un algoritmo que recibe un diseño de entrada, y el índice de un nodo como un parámetro de entrada (parámetro hubIndex , con valor por defecto 0) .A continuación , eliminará todos los enlaces del diseño de entrada excepto aquellos que empiezan o terminan en el nodo hubIndex .

Para la eliminación de los enlaces , utilice la función remove en el objeto Link (ver el Javadoc !!!)

Ejercicio 4: Crear un algoritmo que recibe un diseño de entrada, elimina todos los nodos, y luego crea una serie de nodos dados por el parámetro de entrada numNodes (por defecto 10) . Las coordenadas X e Y de las posiciones de cada nodo se eligen aleatoriamente utilizando una distribución uniforme entre 0 y 10.

Consulte la clase java.util.Random de la versión de Java (consulte el Javadoc !!!).

10 Trabajo en en casa tras la práctica en el laboratorio

Se anima al alumno/a a completar toda los ejercicios que no ha podido completar durante la sesión de laboratorio.