Universidad Politécnica de Cartagena

TEORÍA DE REDES DE TELECOMUNICACIONES

Grado en Ingeniería Telemática
Grado en Ingeniería en Sistemas de Telecomunicación

Curso 2018-2019

Práctica #8. Congestion control

(1 sesión)

Autor:

Pablo Pavón Mariño

1 Objetivo

El objetivo de esta práctica es:

1. Crear algoritmos de Net2Plan que resuelvan las formulaciones del problema de control de congestión usando la librería *Java Optimization Modeler* (JOM).
2. Adquirir experiencia con las diferentes formas de implementar los problemas de optimización en JOM, beneficiándose de las capacidades de la representación vectorial.

2 Duración

La práctica está diseñada para 1 sesión de 2 horas.

3 Evaluación

La práctica se ha diseñado para guiar a los estudiantes en el proceso de aprendizaje de Net2Plan. Las anotaciones que los estudiantes hagan en el documento son para su propio uso en el estudio de la asignatura (examen parcial), y no tienen que entregar nada al profesor para su evaluación.

4 Documentación

En esta práctica servirá de apoyo para su realización:

- Documentación de la librería JOM (ver `http://www.net2plan.com/jom`).
- Herramienta Net2Plan y documentación (ver `http://www.net2plan.com/`).
- Instrucciones del guión.

5 Trabajo previo

Antes de asistir a la sesión de prácticas se recomienda:

- Leer la sección 3.8 y 6.2 de [1], y leer los apuntes de control de congestión.
- Repasar la documentación de JOM en `http://www.net2plan.com/jom`, en particular, como se implementan las restricciones en forma vectorial.

6 Estimación del rendimiento de las fuentes TCP Reno

Sea $\mathcal{G}(\mathcal{N}, \mathcal{E})$ una red, con un conjunto de nodos $\mathcal{N}$ y un conjunto de enlaces $\mathcal{E}$ dado. Las capacidades de los enlaces $(u_e, e \in \mathcal{E})$ son conocidas. El tráfico ofrecido está compuesto por un conjunto de demandas

unicast $\mathcal{D}$. Para cada demanda $d \in \mathcal{D}$, la secuencia de enlaces que atraviesa el camino de la demanda p_d es conocido, pero la cantidad de tráfico ofrecido h_d no se conoce.

Se asume que cada demanda representa un flujo unidireccional de conexiones TCP elefante (que siempre se quiere transmitir tráfico), utilizando una versión TCP-Reno. Como se ha explicado en teoría, el modelo NUM (*Network Utility Maxmimization*) se puede utilizar para estimar el tráfico medio inyectado (en equilibrio macroscópico) para cada conexión. En particular, las tasas medias de conexión TCP se obtienen de la solución óptima encontrada con la siguiente formulación:

- Parámetros de entrada (constantes conocidas):
  - $\mathcal{N}$: Conjunto de nodos.
  - $\mathcal{E}$: Conjunto de enlaces.
  - $u_e, e \in \mathcal{E}$: Capacidad del enlace e.
  - $\mathcal{D}$: Conjunto de demandas unicast ofrecidas (conexiones TCP, consideradas unidireccionales).
  - $p_d, d \in \mathcal{D}$: Secuencia de enlaces atravesados de una conexión TCP d. Denotamos $\mathcal{P}_e$ al conjunto de conexiones TCP que atraviesa un enlace e.
  - $RTT_d, d \in \mathcal{D}$: Round-trip-time de los paquetes en la conexión d. Se asume que en una conexión, el RTT se considera solo el tiempo de propagación de atravesar los enlaces (multiplicado por 2, para tener en cuenta el camino de vuelta).
- Variables de decisión:
  - $h_d, d \in \mathcal{D}$: Tráfico medio inyectado por cada conexión TCP d.
- Formulación:

$$\max \quad -\sum_d \frac{3}{2RTT_d^2 h_d}, \quad \text{subject to:} \tag{1a}$$

$$\sum_{d \in \mathcal{P}_e} h_d \leq u_e, \quad \forall e \in \mathcal{E} \tag{1b}$$

$$h_d \geq 0, \quad \forall d \in \mathcal{D} \tag{1c}$$

La función objetivo (1a) representa el modelo NUM, que intenta maximizar la suma de las utilizaciones de cada conexión TCP. La función de utilidad $U_d(h_d)$ de una conexción TCP Reno d, de acuerdo con el modelo visto en teoría, es:

$$U_d(h_d) = -\frac{3}{2RTT_d^2 h_d}$$

Las restricciones (1b) son las estándar de la capacidad de un enlace, y significan que, para cada enlace, el tráfico que lleva el enlace debe ser menor o igual a su capacidad (esto es, el enlace no se satura). Finalmente (1c) prohibe inyectar tráfico negativo.

7 Algoritmo Net2Plan

El alumno deberá desarrollar un algoritmo de Net2Plan que resuelva el problema (1) realizando los siguientes pasos:

1. Copiar la plantilla `AlgorithmTemplate.java` disponible en el Aula Virtual y renombrarla como `TCPReno.java`.

2. Establezca el tipo de routing del objeto `NetPlan` como source routing.

3. Eliminar el tráfico cursado y borrar todas las rutas de la red.

4. Para cada demanda unicast, crear una ruta que lleve 0 unidades de tráfico, eligiendo para ello la ruta más corta entre el nodo origen y destino de la demanda, medida en número de saltos(p.ej. utilizando el método `getShortestPath` de `GraphUtils`).

5. Crear un objeto tipo `OptimizationProblem` (p.ej. `op`).

6. Añadir las variables de decisión del problema, con el nombre `h_d`: una variable por cada (*ruta*. La coordenada i-th corresponde al índice i del objeto `Route` (que está asociado a una demanda específica). El valor mínimo de la variable es 0 y el máximo `Double.MAX_VALUE`.

7. Establecer la función objetivo del problema. Para ello puede utilizar método de la clase `NetPlan`:

   `netPlan.getVectorRoutePropagationDelayInMiliseconds`

   para obtener el vector `DoubleMatrix1D` con el retardo de propagación (medido en ms) para cada camino, que puede convertirse en un array standar `double[]` utilizando el método `toArray`. Recalcar que el round-trip time (RTT) de una conexión es el doble del retardo de propagación [1]. Se recomienda crear un vector con los coeficientes: $z_d = -\frac{3}{2RTT_d^2}$, y añadirlo como parámetro de entrada del problema, y hacer que la función objetivo sea maximizar:

   `sum (z_d ./ h_d).`

   where `./` es la división elemento a elemento de dos arrays del mismo tamaño.

8. Utilice un bucle `for` con tantas iteraciones como enlaces, para añadir las restricciones de capacidad de los enlaces(1)b. Para añadir la restricción de un enlace `e`:

   - Establezca los parámetros de entrada de JOM:
     - `P_e` con los índices de las rutas atravesando el enlace `e`. Para ello, utilice el método *getTraversingRoutes* del objeto nodo, para obtener los enlaces de salida, y el método *NetPlan.getIndexes* para convertir la colección de enlaces a sus índices.
     - `u_e` con la capacidad del enlace. .
   - Establezca la restricción utilizando la función `sum`, sobre `h_d`, pero restringiendo la suma a los elementos en `P_e`.

9. Llame al solver para encontrar la solución numérica. Como el problema no es lineal, debe usar el solver `ipopt`.

10. Obtenga los datos obtenidos en la solución primal.

11. Guardelos en el objeto `netPlan`. Para ello, utilice un blucle `for` iterando sobre las rutas del diseño. Para cada ruta: (i) establezca el tráfico cursado de la ruta, y la capacidad del enlace ocupado de acuerdo a la solución óptima, (ii) establezca el tráfico ofreido de la demanda asociada a la ruta, para que sea igual al tráfico cursado.

[1] Tenga en cuenta que multiplicar la función objetivo por un valor positivo constante no cambia el óptimo del problema. Por tanto, los factores constantes deben quitarse de la función objetivo. El resultado no cambiará si el RTT se mide en segundos o en cualquier otra unidad

7.1 Chequeo del algoritmo

Carga la red `example7nodesWithTraffic.n2p`. El algoritmo debería producir una solución con un total de 600.36 unidades de tráfico total ofrecido (cursado), y todos los enlaces con una utilización del 100%.

8 Variaciones del problema

Quiz 1. Modifica el problema en (1) tal que ahora el coste de la función objetivo sea la conocida función de justicia α-fair:

$$\sum_d \frac{h_d^{1-\alpha}}{1-\alpha}$$

para $\alpha \geq 0, \alpha \neq 1$, y

$$\sum_d \log h_d$$

cuando $\alpha = 1$. Implementa el algoritmo de Net2Plan que tiene como parámetro de entrada `alpha` (por defecto a 1), y resuelve la formulación NUM cuyo factor α sea igual al valor introducido en el parámetro de entrada `alpha` del algoritmo. Para comprobar la solución, carga la red `example7nodesWithTraffic.n2p`. El throughput para $\alpha = 2$ es 514.64, y para $\alpha = 1$ es 549.26.

Quiz 2. Utilice el algoritmo que acaba de desarrollar para rellenar la siguiente tabla, en la que se muestra como el throughput de la red varia cuando se aplican distintos valores de justicia α para control de congestión.

La tabla se refiere a la red NSFNET del fichero `NSFNet_N14_E42.n2p`, con capacidades en los enlaces de $u_e = 500$, y una demanda para cada par de nodos[2]. Se incluyen los resultados para $\alpha = 2$.

Topología NSFNET

α	**Jain fairness factor**	**Throughput** $\sum_d h_d$
0		
0.5		
1		
2	0.48	12311
5		

Column *Jain fairness factor* debe mostrar el Jain factor J (ver p.ej. "Jain's fairness measure" en `http://en.wikipedia.org/wiki/Fairness_measure`). el factor J es una medida de justicia en la distribución de los recursos. En nuestro caso, J se denota como:

$$J = \frac{(\sum_d h_d)^2}{|\mathcal{D}| \times \sum_d h_d^2}$$

donde $|\mathcal{D}|$ es el número de demandas. Jain factor será máximo ($J = 1$) cuando todas las demandas tengan exactamente la misma tasa ("maximum fairness"), y tomará el valor mínimo posible ($J = 1/|\mathcal{D}|$) cuando una demanda reciba tráfico y el resto no reciba nada ("minimum fairness").

[2] *Note*: Para valores elevados de α (p.ej. $\alpha \geq 4$), las soluciones pueden diferir en distintas ejecuciones debido a inestabilidades numéricas.

El estudiante debería incluir en el código del algoritmo que ejecute y pinte con `System.out` el índice J para la solución obtenida.

- ¿Hay demandas con tasa asignada 0 en el caso de $\alpha = 0$? ¿Podría una red como Internet ser util si el control de congestión fuera diseñado para maximizar el throughput?
- ¿Cuál es la tendencia del throughput y la justicia al incrementar α?

9 Forma matricual de las restricciones del problema (opcional)

9.1 Restricciones de capacidad de los enlaces

Las restricciones de capacidad de los enlaces en (1b) toman la forma:

$$\sum_{d:e\in \sqrt{}^{d}} h_d \leq u_e, \forall e \in \mathcal{E}$$

Todas las restricciones $|\mathcal{E}|$ se pueden representar con una única desigualdad vectorial, como sigue:

$$A_{ep} \times h'_d \leq u'_e \tag{2}$$

donde:

- A_{ep} es la matriz de asignación enlace-a-ruta. Es una matriz de $|\mathcal{E}| \times |\mathcal{P}|$ con una fila por cada enlace, una columna por cada ruta (en este caso una por demanda). La coordenada (e, p) es el numero de veces que el camino p atraviesa el enlace e.
  - La matriz de asignación enlace-a-ruta se puede obtener como una matriz dispersa (spare matrix) en Net2Plan utilizando el método de la clase `NetPlan`:

    `getMatrixLink2RouteAssignment`
- h_d es un vector fila $1 \times |\mathcal{D}|$, con las variables de decisión (el tráfico se cursa en la ruta asociado a la demanda d).
- u_e es un vector fila $1 \times |\mathcal{E}|$ con la capacidad de cada enlace.
  - El vector de las capacidades de los enlacee se puede obtener en Net2Plan utilizando el método de la clase `NetPlan`:

    `getVectorLinkCapacity`

Quiz 3. Reescriba las restricciones de capacidad de los enlaces utilizando la forma matricial. Recuerde que el operador de JOM `*` implementa la multiplicación de matrices estándar.

10 Trabajo en casa despúes de realizar la práctica

Al estudiante se le anima a completar todos los *Quizs* que no haya podido terminar durante la sesión de prácticas.

Bibliography

[1] *P. Pavón Mariño, "Optimization of computer networks. Modeling and algorithms. A hands-on approach", Wiley 2016.*