Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Examen 1

Modelo de redes
Ing. en ciencias de la computación
Catedrático: Dr. Iván Olmos Pineda

PRIMAVERA 2020 | SECCIÓN OO1
06 DE FEBRERO 2020

Resumen

En el presente documento se plasma las metodologías y técnicas utilizadas para la realización del programa del primer examen de la materia de Modelos de redes. Este, además contiene los pseudocódigos utilizados en la creación de dicho sistema capaz de calcular el tiempo de transferencia aproximado de un archivo entre un emisor y un receptor basado en el cálculo de latencias.

Introducción

En los últimos años la humanidad a experimentado un gran y rápido avance en el desarrollo tecnológico, extrañamente, una de las causas de este aceleramiento fue la guerra fría, las tensiones que se vivió entre Estados Unidos y Cuba, dio paso a transformar la manera en la que nos comunicamos. Dio paso a la creación de las redes de computadoras.

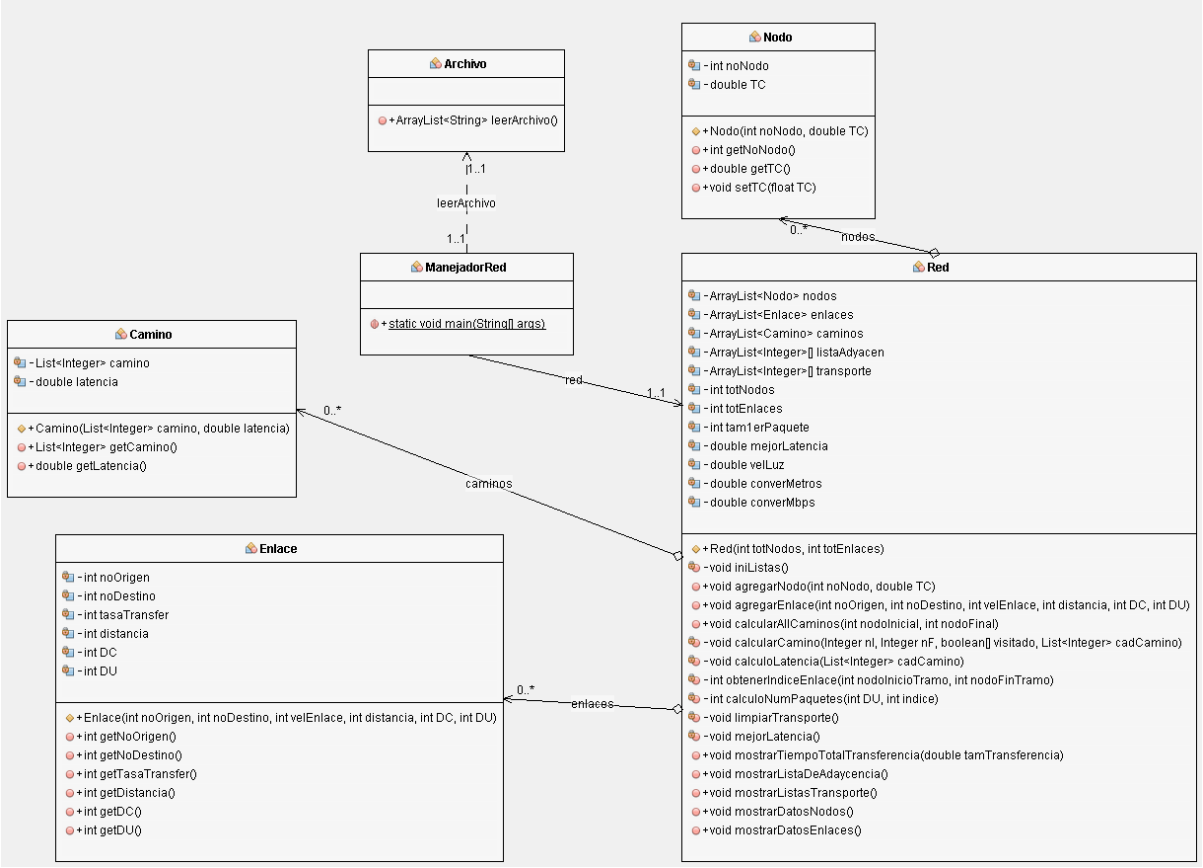
La transmisión de datos es la práctica en la cual una computadora, un emisor, puede enviar datos a otra, un receptor; pasando, quizás tal vez, a través de múltiples nodos; el paso de datos a través de estos genera un costo en cuanto a tiempo, afectando el rendimiento de la red. A esto se le llama latencia, al retaso generado por la demora en la propagación, trasmisión y tiempo de cola.

Desarrollo

El programa está desarrollado en el lenguaje de programación Java, haciendo uso de IDE Netbeans; consta de seis clases; manejadorRed, la cual inicializa el sistema, llama a la clase Archivo que lee el documento, crea una Red y manda los datos a esta.

Las otras tres clases son nuestros objetos Nodos, Enlaces y Caminos; que se implementan como arrayList para el manejo de estos durante su tratamiento en la red.

A continuación, se presenta un diagrama UML para la mejor conceptualización del sistema.



Datos de entrada

Los datos de entrada para este sistema, es el archivo .txt en el cual se encuentra la cantidad de nodos, cantidad de enlaces, el número de los nodos, así como los tiempos de cola por nodo; los datos de los enlaces entre ellos, nodo origen, nodo destino, velocidad de enlace, distancia, cantidad de datos de control, de datos de usuario. Tamaño del paquete a enviar y los números de los nodos de origen y destino.

Datos de salida

En pantalla, se muestra la cantidad de caminos posibles y se enlistan, así como sus latencias. Se proyecta la mejor latencia y finalmente el tiempo total de transferencia en segundos, minutos y horas.

Lectura del archivo

La realización de la lectura del archivo se llevo acabo en la clase Archivo, dentro del método leerArchivo(), en dicho método creando un objeto BufferedReader que nos permitía leer un archivo usando el método readLine(), se agrego todas estas líneas leídas a un ArrayList de String's y se retorno a nuestro manejadorRed; el cual instancia un objeto de tipo Red, y mediante el uso del método .split(",") se dividían en valores las cadenas para ir enviando los parámetros recibidos y así guardar los datos obtenidos del documento.

Algoritmo para cálculo de rutas

Para calcular todas las posibles rutas entre el nodo inicial y el nodo final se trabajo bajo el algoritmo de recorrido en profundidad, (DFS: Depth First Search) haciendo las modificaciones necesarias para poder cumplir con nuestra meta.
Se utilizo este algoritmo ya que se centra en visitar el vértice inicial y posteriormente ir visitando los vértices adyacentes y marcándolos hasta visitar el vértice inicial nuevamente.

Pseudocódigo algoritmo DFS:

```
DFS(vi)
{
marcar vi como visitado

para cada vk adyacente a vi
        si vk no visitado
                entonces DFS(vk)
}
```

Se utilizan dos métodos, el primero calcularAllCaminos() que recibe y permite instanciar e inicializar algunas variables a utilizar para el método principal; su función en el algoritmo es agregar al camino el nodo inicial y pasar datos al método calcularCamino(). Es importante mencionar que es en este método donde es cambiado el valor de TC de los nodos Inicial y final a 0; puesto que esta acción beneficiará al cálculo de latencias.

El segundo método calcularCamino() se utiliza para ir agregando y removiendo los nodos de nuestro camino; así como para llamar la función que nos va calculando las latencias por camino.

Pseudocódigo método calcularCamino():

```
calcularCamino(vi, vf, visitado, camino)
{
marcar vi como visitado
si vi es igual a vf
        entonces se manda el camino a calculoLatencia()
                marcar vi como no visitado

para cada vk adyacente a vi
        si vk no visitado
        entonces añadir a camino vk
                calcularCamino(vi, vf, visitado, camino)
                remover a camino vk

marcar vi como no visitado
}
```

Pseudocódigo para el cálculo de latencias

Para el calculo de latencias de los diferentes caminos se utiliza el método calculoLatencia, el cual se va ejecutando cada camino recibido del anterior método; va calculando las latencias de los costos parciales, que ahora se referirán a estos como tramos. Se ayuda de otros dos métodos, obtenerIndiceEnlace() y calculoNumPaquetes().

Pseudocódigo calculoLatencia():

```
calculoLatencia(camino)
{
para cada tramo del camino
        i = obtenerIndiceEnlace(nI,nF)

        si es la primera iteración
                entonces se almacena el valor del primer DU

        latenciaTramo = ( (enlace(i).getDistancia()*0.001/velocidadLuz) +
                        ( (enlace(i).getDU()+enlace(i).getDC() ) *
                        8x10- 6/enlace.get(i).getTasaTransfer() ) +
                        nodo.get(j).getTC() ) *
                        calculoNumPaquetes (enlace.get(i).getDU(), j)

        latenciaCamino = latenciaCamino + latenciaTramo;

limpiarTransporte()
print “Camino: “ + camino + “Latencia del camino: “ + latenciaCamino
}
```

Pseudocódigo obtenerIndiceEnlace():

```
obtenerIndiceEnlace(nodoInicioTramo,nodoFinalTramo)
{
para cada enlace existente
        si el enlace.nodoOrigen es igual a nodoInicioTramo y el enlace.Destino es
        igual a nodoFinalTramo
                entonces iEnlace es igual a ese enlace
return iEnlace
}
```

Pseudocódigo para el cálculo de paquetes

El siguiente pseudocódigo muestra como se realizó el calculo del número de paquetes, el cual es necesario e indispensable ya que nos permite realizar el calculo de la latencia por tramo de manera eficiente. Para ello se uso un arrayList que recibió el nombre de transporte, en el cual se irán almacenado las cantidades en las que sí es el caso, se subdividirá el paquete inicial. En dicho arrayList primeramente en el método calculoLatencia (fue agregado en el primer índice el primer valor de DU).

Pseudocódigo calculoNumPaquetes():

```
calculoNumPaquetes(int DU, int indice)
{
para cada elemento de cada lista de transporte
        valor = elemento j de la lista i de transporte
        si valor > DU
                entonces paquetesTotales = valor / DU
                                incompleto = valor - (DU * paquetesTotales)
                                mientras paquetesTotales no sea igual a 0
                                      agregar a transporte i + 1 paquetesTotales
                                 agregar a transporte i + 1 incompleto
                                 totalPaquetes++;
        si no
                entonces
                                agregar a transporte i + 1 valor
return totalPaquetes;
}
```

Cálculo de tiempo de transferencia

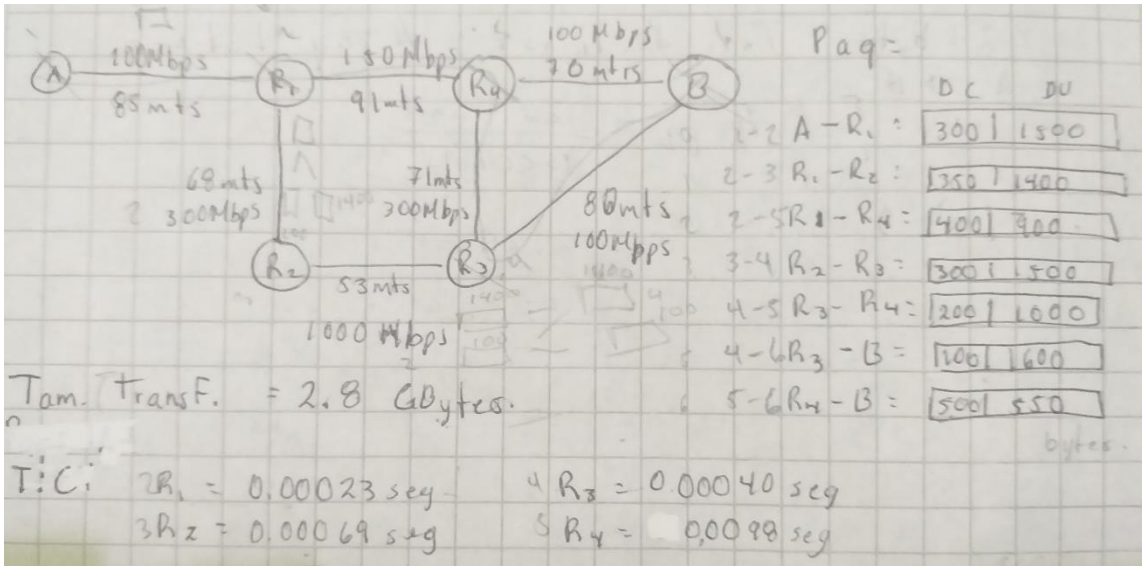
Para el calculo del tiempo de transferencia de un archivo desde el emisor al receptor se utilizó el tamaño del paquete leído en el archivo. Utilizando las siguientes ecuaciones mostradas, se calcula el numero de paquetes y el tiempo total de transferencia.

$\#paq = \frac{Tam.\ transferencia}{tam.\ Primer\ paquete}$ ecu 1.

$T.T.T = \#paq * Latencia_{A-B}$ ecu 2.

Ejemplos

El primer ejemplo por mostrar es con el archivo de texto que se generaría con el grafo visto en clase:



Datos de entrada:

```
6,7
1,0
2,0.00023
3,0.00069
4,0.00040
5,0.0098
6,0
1,2,100,85,300,1500
2,3,300,68,350,1400
2,5,150,91,400,900
3,4,1000,53,300,1500
4,5,300,71,200,1000
4,6,100,80,100,1600
5,6,100,70,500,550
2.8
1,6
```

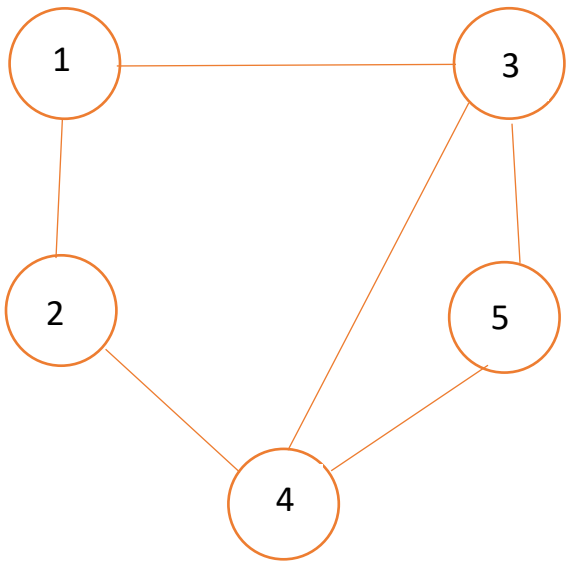
Datos de salida:

```
Camino: [1, 2, 3, 4, 5, 6] Latencia: 0.03251086666666666
Camino: [1, 2, 3, 4, 6] Latencia: 0.0029497566666666667
Camino: [1, 2, 5, 4, 6] Latencia: 0.021250563333333333
Camino: [1, 2, 5, 6] Latencia: 0.020450489999999998

Existen 4 caminos posibles

Mejor latencia: 0.0029497566666666667
Tiempo total de transferencia en segundos: 5912.253332376667
Tiempo total de transferencia en minutos: 98.5375555396111
Tiempo total de transferencia en horas: 1.6422925923268519
BUILD SUCCESSFUL (total time: 0 seconds)
```

Grafo segundo ejemplo:



Datos de entrada:

prueba: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
5,6
1,0.00011
2,0.00023
3,0.00069
4,0.00040
5,0.00098
1,2,100,85,300,1100
1,3,100,68,350,1200
3,4,150,91,400,905
3,5,1000,53,300,100
2,4,330,71,200,1700
5,4,100,80,100,1300
3.2
2,4
```

Datos de salida:

Output - TTTRedes (run)

```
run:
Camino: [2, 1, 3, 4] Latencia: 0.0011763166666666667
Camino: [2, 1, 3, 5, 4] Latencia: 0.014184229999999999
Camino: [2, 4] Latencia: 4.6297272727272726E-5

Existen 3 caminos posibles

Mejor latencia: 0.0011763166666666667
Tiempo total de transferencia en segundos: 2377.5265466333335
Tiempo total de transferencia en minutos: 39.62544244388889
Tiempo total de transferencia en horas: 0.6604240407314815
BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusión

Al finalizar la realización del sistema para el cálculo de tiempo de transferencia entre un emisor y un destinatario en una red basado en el calculo de latencias; puedo concluir que es importante conocer no solo los conceptos que existen en la comunicación de computadoras, sino además poder aplicarlos como ha sido este el caso. Conocer todos los caminos posibles que se pueden generar y sus respectivas latencias, nos ayudaran para decidir el camino más corto y así descubrir el menor tiempo de transferencia.

Bibliografía

- Gómez Vera, A. R. (2010). Modelo de red de interconexión basado en enlace. Recuperado de: https://www.recercat.cat/handle/2072/97242
- Galicia, Y. (marzo 23, 2019). Estructura de datos: Grafos, de BUAP-Blackboard Sitio web: https://buap.blackboard.com/bbcswebdav/pid-1645847-dt-content-rid-20901536_2/courses/201925_44130/ED_U4_Grafos_01%29.pdf
- Kurose, J. F., Ross, K. W., Hierro, C. M., y Pablo, Á. P. D. M., & Marrone, L. (2010). Redes de computadoras: un enfoque descendente. Addison Wesley.
- Quiñones, J. C. C. (2009). Algoritmos de planificación en redes de paquetes. Sistemas y Telemática, 7(14), 91-107. Recuperado de: https://www.icesi.edu.co/revistas/index.php/sistemas_telematica/article/view/1015
- Carrera, F. R. F. (2009). Retardo en la Transmisión de Información en la Red Internet. Eídos, (2), 3-7. Recuperado de: https://revistas.ute.edu.ec/index.php/eidos/article/view/47/44