Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Modelo de Redes

“Software para el cálculo de Latencia de Redes”

Pedraza Celón Ian Yael

06/02/2020

La latencia es el tiempo de retraso que ocurre en una comunicación. Es usualmente medida en milisegundos desde el momento en el que es enviada una señal por el emisor al momento al que es recibida por el receptor.

La latencia está relacionada con el tiempo que toma un bit en viajar de un extremo de un medio al otro.

Depende principalmente de tres factores:

- **Tiempo de propagación** del bit por el medio, que depende del tiempo de propagación de la corriente o luz por el medio, además de la distancia recorrida.

- **Máxima cantidad de datos** que pueden ser transmitidos por la red sin segmentarse, Al tiempo de transmisión.

- Tiempos de espera para difundir un paquete a través de un conmutador, a demás del trafico de la red. A este tiempo se le denomina **tiempo de cola**.

Entonces el cálculo de la latencia esta dado básicamente por la siguiente ecuación:

$$Latencia = Tiempo\ de\ propagación + Tiempo\ de\ transmisión + Tiempo\ de\ cola$$

Donde el tiempo de propagación esta dado por:

$$Tiempo\ de\ propagación = \frac{Distancia\ a\ recorrer}{Velocidad\ de\ la\ luz}$$

Mientras que el tiempo de transmisión se define como:

$$Tiempo\ de\ transmisión = \frac{Tamaño\ del\ paquete}{Tasa\ de\ transferencia\ teorica}$$

Entonces finalmente tenemos la siguiente ecuación

$$Latencia = \frac{Distancia\ a\ recorrer}{Velocidad\ de\ la\ luz} + \frac{Tamaño\ del\ paquete}{Tasa\ de\ transferencia\ teorica} + Tiempo\ de\ cola$$

Con esta ecuación podemos calcular la latencia a través de un medio punto a punto, pero considerando que una red real nunca esta conectada así, y es necesario por pasar por varios dispositivos de una red mas grande, entonces tenemos que hacer la sumatoria de todas esas latencias entre dispositivos para obtener una latencia final.

Teniendo entonces:

$$Latencia = \sum_{i=1}^{n} \frac{Distancia\ a\ recorrer_i}{Velocidad\ de\ la\ luz_i} + \frac{Tamaño\ del\ paquete_i}{Tasa\ de\ transferencia\ teorica_i} + Tiempo\ de\ cola_i$$

El tiempo de cola del ultimo dispositivo será cero, debido a que el paquete de datos es para este dispositivo y no tiene que procesar nada más.

Siempre se buscará el camino con menor latencia posible, porque nos brinda una mejor tasa de transferencia.

Una tasa de transferencia alta es deseable cuando se transmiten grandes cantidades de datos.

Una latencia baja es importante cuando se transmiten bajos volúmenes de datos.

Desarrollo del software

El software fue desarrollado en java y consiste en cargar un archivo con los datos de la red, y el programa se encarga de buscar la ruta más rápida para él envió de información a través de la red.

El programa calcula todas las posibles rutas, obtiene la latencia de cada camino, y escoge el que menor latencia tenga para enviar los paquetes de datos; una vez hecho esto, calcula el tiempo que tardaría en enviarse un paquete de un tamaño dado.

A continuación, se explica a detalle el funcionamiento del software.

1. **Archivo.**

   El archivo para cargar los datos debe contener la siguiente información con la siguiente estructura.

Tabla 1: Estructura del archivo

Numero de nodos, Numero de enlaces
Lista de nodos, cada nodo debe contener la siguiente información: • Numero de nodo • Tiempo de cola (en segundos)
Lista de enlace cada enlace debe contener la siguiente información: • Nodo de origen • Nodo de destino • Velocidad (en Mbps) • Distancia (en metros) • Tamaño de los datos de usuario del paquete (en bytes) • Tamaño de los datos de control del paquete (en bytes)
Tamaño del paquete a enviar (en gigabytes)
Nodo origen, Nodo destino

En ejemplo del archivo puede ser el siguiente:

Tabla 2: Ejemplo del archivo de datos

6, 8
0, 0.0 1, 0.00023 2, 0.00069 3, 0.00040 4, 0.0098 5, 0.0
0, 1, 100, 85, 300, 1500 1, 2, 300, 68, 350, 1400 1, 4, 150, 91, 400, 900 2, 3, 1000, 57, 300, 1500 3, 4, 300, 71, 200, 1000 4, 3, 300, 71, 200, 1000 4, 5, 100, 70, 500, 550 3, 5, 100, 80, 100, 1600
2.8
0, 5

El archivo de prueba integrado por defecto es: **test.txt**

2. Arquitectura

La arquitectura de la aplicación está basada en un modelo Model View View Model (MVVM), separando por un lado la capa de la vista, la capa del procesamiento de datos y la capa de manejo de datos, modularizando cada parte del software y haciéndola trabajar en conjunto.

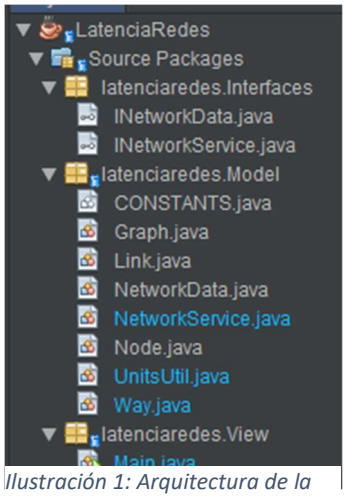

Ilustración 1: Arquitectura de la aplicación

3. Diagrama de Clases

El diagrama de clase de la aplicación se muestra a continuación. En el siguiente punto se explicará a detalle el funcionamiento de cada clase a detalle y como contribuye al funcionamiento del software.

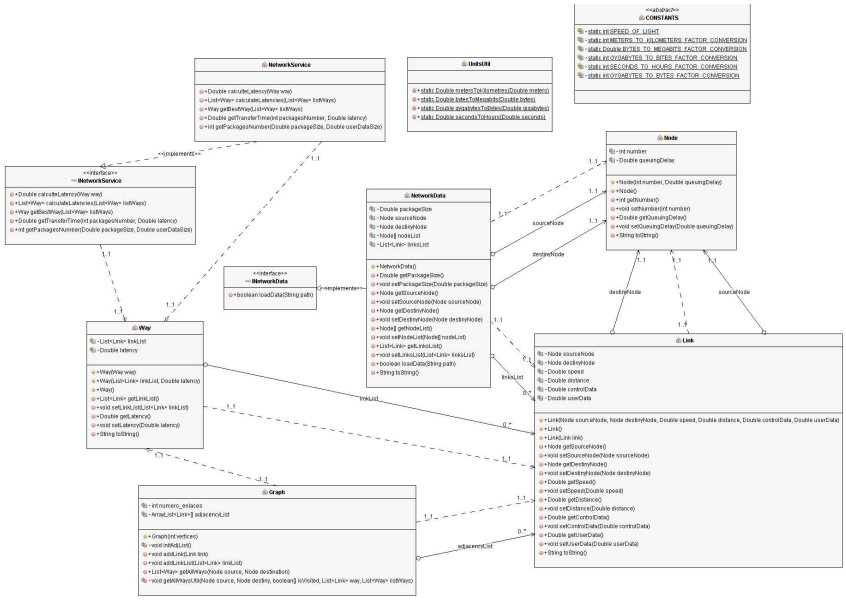

Ilustración 2: Diagrama de clases UML de la aplicación

4. Funcionamiento.

Clases Auxiliares:

Tenemos tres clases auxiliares, que nos van a ayudar a procesar algunos datos.

La clase CONSTANTS, nos provee de los factores de conversión, para convertit de ciertas unidades. Mientras que la clase UnitsUtil nos proporciona los métodos para hacer las conversiones.

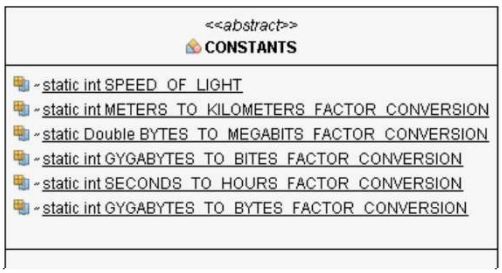

Ilustración 4: Clase CONSTANTS

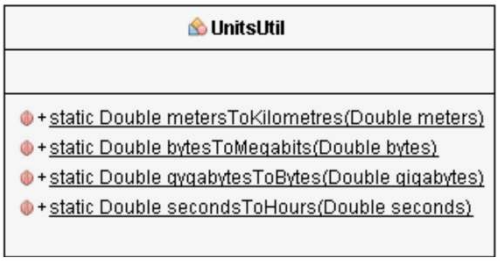

Ilustración 3: Clase UnitUtils

Clases de Datos:

Se cuenta con tres clases de datos: *Node*, *Link* y *Way*. En donde se modelan los nodos y enlaces respectivamente. En la clase *Way* tenemos lo que es una lista de enlaces que forman un *camino* y una latencia asociada con ese camino.

Por otro lado, en la clase *Node*, cuenta con el número de nodo y su tiempo de cola. Y finalmente la clase *Link*, tiene un nodo de origen, nodo de destino, velocidad, distancia, tamaño de los datos de usuario y tamaño de los datos de control.

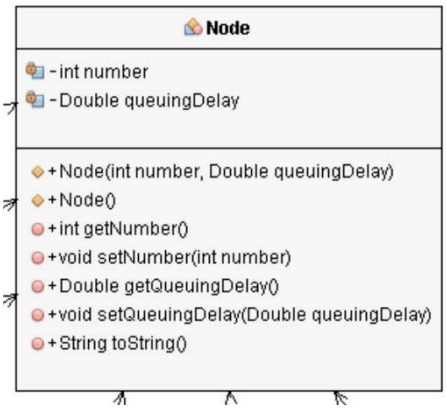

Ilustración 6: Clase Node

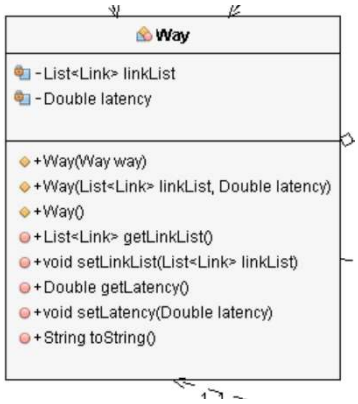

Ilustración 5:Clase Way

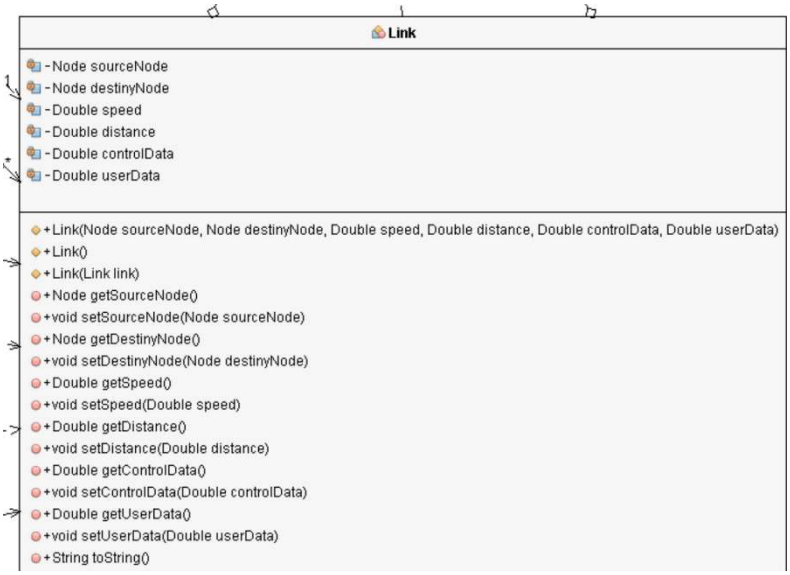

Ilustración 7: Clase Link

Carga de Datos:

Para la cargar de datos tenemos una clase especial *NetworkData*, que nos permite leer el archivo introducido, así como para almacenar esos datos y hacer uso de ellos en conjunción de otras clases.

NetworkData implementa la interface *INetworkData* que nos da el método de carga de datos *loadData(path: String)*, que recibe como parámetro la ruta del archivo a cargar.

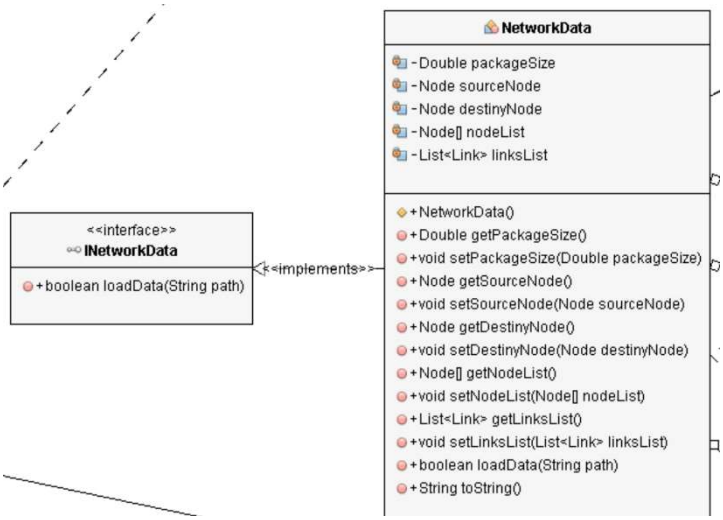

Ilustración 8: Clase NetworkData e Interface INetworkData

Manejo del Grafo

El manejo del grafo asociado a la red lo lleva acabo clase *Graph,* que se encarga de generar el grafo a partir de una lista de enlaces, así como de buscar todos los caminos posibles de un punto a otro, este método nos devuelve una lista de caminos *(Way)* que modelamos anteriormente.

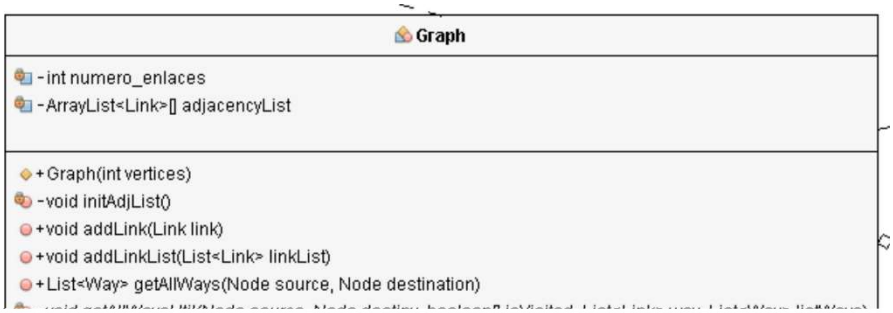

Ilustración 9: Clase Graph

Cálculo de Datos

Los cálculos necesarios para el software están a cargo de la clase *NetworkService* en conjunto con su interface *INetworkSerive*. Esta clase consta básicamente de cinco métodos.

calculateLatency(way: Way) : Double: Recibe un camino como parámetro y como su nombre lo indica calcula la latencia de un camino.

calculateLatencies(listWays: List<Way>) : List<Way>: A partir de una lista de caminos calcula la latencia de cada uno de ellos y devuelve la misma lista de caminos pero ya con sus respectivas latencias asociadas.

getBestWay(listWays: List<Way>) : Way: Devuelve el mejor camino posible a partir de una lista de caminos con sus latencias calculadas.

getTransferTime(packagesNumber: Int, latency: Double): Double: Obtiene el tiempo de transferencia de un archivo, tomando el número de paquetes por los que dividirá y la latencia del camino por donde se enviara.

getPackagesNumber(packageSize: Double, userDataSize: Double): Int: Obtiene el número de paquetes por los cuales se dividirá el archivo a enviar por la red.

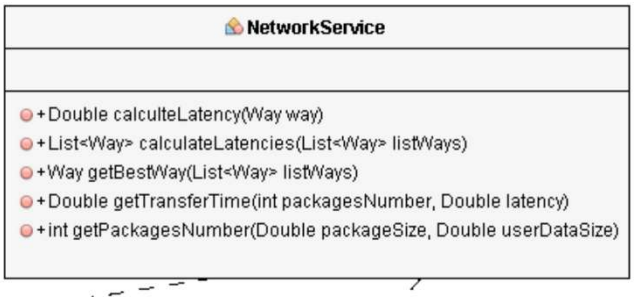

Ilustración 11: Clase NetworkService

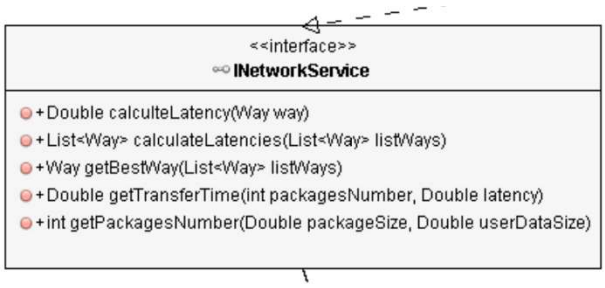

Ilustración 10: Interface INetworkService

Todas las clases anteriores es su conjunto hacen funcionar la aplicación para poder calcular la manera más rápida para poder enviar un archivo a través de una red, encontrando todos los caminos posibles y el más adecuando.

Conclusión.

La latencia de redes es un elemento sumamente importante para considerar al momento de enviar algún archivo para la red, porque de esta manera nos estamos asegurando de mandar cada paquete de datos de la manera más rápida posible, y segura, ya que entre menos latencia haya en mejores condiciones llegará la información y se generarán menos reenvíos de información, optimizando así el tiempo.

El usuario común no tiene que preocuparse por esto, debido a que lo hacen los dispositivos de manera automática, pero nosotros como ingenieros en ciencias de la computación o administradores de redes, es un factor muy importante, que hay que saber hacer.

Bibliografía.

Percy, A. (1999). Understanding latency in IP telephony. Brooktroot Technology, 34-41.

Olmos I.. (2014). Rendimiento de una red. 04/01/2020, de BUAP Sitio web: https://www.cs.buap.mx/~iolmos/redes/3_Rendimiento.pdf

S. Singhal. (2015). Print all paths from a given source to a destination. 04/02/2020, de GeeksforGeeks Sitio web: https://www.geeksforgeeks.org/find-paths-given-source-destination/