DESARROLLO DE UN SOFTWARE BASADO EN JAVA CAPAZ DE INTERPRETAR TRAZAS DE ARCHIVOS GENERADOS POR NS-2

Fredy Pachón Barbera
Especialista en Teleinformática
Ingeniero de campo Sonda Colombia
fredy.pachon@co.sonda.com
Bogotá, Colombia

German Andrés Acosta Ruge
Especialista en Teleinformática
Ingeniero de campo Teeaccess LTDA
grupo3@teleaccess.com.co
Bogotá, Colombia

Tipo: Artículo reporte de caso

Fecha de Recepción: Junio 10 de 2013
Fecha de Aceptación: Agosto 6 de 2013

DEVELOPMENT OF SOFTWARE BASED ON JAVA ABLE TO INTERPRET TRACES OF FILES GENERATED BY NS-2

ABSTRACT
During the Simulation of a data network, the goal is to evaluate, investigate and analyze its behavior in a real environment, always looking for the optimization of the resources. The analysis of this information is a key issue that helps to know the strengths and weaknesses of the simulated network. The final objective of this article is to show the development of an interpretation tool of traces generated by NS-2 based on the programming language Java and the statistical variables that were taken into account for the implementation, in order to make easier the analysis of simulated networks by using the network simulator tool. To finally achieve the construction of software, and manage to determine estimates of jitter, throughput and bytes sent by each node, starting from the traces generated by NS-2.

Key words: NS-2, simulation, software, JAVA.

RESUMEN
En la simulación de una red de datos, se trata de evaluar, investigar y/o analizar su comportamiento en un ambiente real, siempre buscando la optimización los recursos. El análisis de la información es un punto clave para conocer fortalezas y debilidades de la red simulada. El objetivo de este artículo es mostrar el desarrollo de una herramienta de interpretación de trazas generadas por NS-2 basado en el lenguaje de programación Java y las variables estadísticas que se tuvieron en cuenta para la implementación, con el fin de facilitar el análisis de redes simuladas con la herramienta network simluator. Finalmente se logra la construcción del software, logrando determinar los cálculos de jitter, throughput y bytes enviados por cada nodo, a partir de las trazas generadas por NS-2.

Palabras claves: NS-2, simulación, software, JAVA.

1. INTRODUCCIÓN

Una de las herramientas más utilizadas para el análisis e investigación en redes de datos cableadas e inalámbricas es la herramienta discreta network simulator [1], orientado a objetos escrito en C++, y con interprete Otcl. La aplicación almacena todo el resultado de sus simulaciones en archivos de texto plano, con extensión .tr . Para interpretar esta información es necesario utilizar aplicaciones de terceros, integrarlos a NS -2, y así analizar el comportamiento de la red simulada.

La herramienta más utilizada para el análisis de trazas se llama Xgraph. [2] , ofrece graficas simples y se invoca utilizando los scripts de simulación en NS-2. Otra de la herramienta usadas, tiene como nombre tracegraph [3], la cual es más robusta, capaz de generar graficas de throughput y jitter, tiene como inconveniente que requiere de librerías propias de Matlab [4] para su funcionamiento, su sitio oficial no tiene gran soporte y los links proporcionados en la página de soporte [3], en su mayoría no funcionan.

Con el ánimo de explotar en mayor cantidad la información que proporcionan las trazas, se plantea la opción de crear una aplicación basada en Java, que no requiera software propietario, que sea interactivo y que permita generar figuras y datos estadísticos de la simulación.

El desarrollo de esta aplicación busca, como primer objetivo establecer que variables graficar y que datos estadísticos es posible mostrar al usuario. Otro factor a analizar es el tiempo de procesamiento que tomara el software, al cargar los archivos, por lo que se analizarán diferentes tamaños de ficheros para determinar los límites del software.

Este artículo mostrará las diferentes fases utilizadas, en la construcción de la aplicación tomando como base la metodología del Proceso Unificado Racional (RUP), en conjunto con UML [5], compuesto por cuatro fases: inicio, elaboración, construcción y transición.

El artículo se estructura inicialmente explicando la metodología utilizada para el desarrollo del software, explicando brevemente lo realizado en de cada una de las fases de la metodología RUP, en la sección 3, se explicara el modelado de comportamiento, donde se detallan los casos de usos, la explicación de las clases utilizadas, librerías, y como se realizaron los cálculos de las variables escogidas, en la sección 4, se explica el modelo estructural del programa, los resultados obtenidos y finalmente las conclusiones del trabajo.

2. METODOLOGÍA

En el desarrollo del software, en una primera fase, de forma tentativa se determinaron las clases candidatas a utilizar en la aplicación, teniendo en cuenta el objetivo principal del programa, (leer archivo, estructura traza, interfaz gráfica y graficar). Claramente el único actor, es el usuario y se planteó un bosquejo inicial sobre la interfaz gráfica.

En análisis más detallado, en una segunda fase, denominada de elaboración, se realizó una inspección más detallada de las clases candidatas a utilizar en el software, se planteó como opción tener una figura 3D, con sus respectivas estadísticas, generando dos clases candidatas adicionales, una para la parte descriptiva y otra para estadística.

En este proceso, se estableció graficar las variables de rendimiento jitter y throughput, completando de esta forma las clases candidatas, que servirán de base para iniciar la construcción de la aplicación.

Se realizó la búsqueda, de las herramientas necesarias para construir las figuras en Java, la más utilizada Jfreechart [6], y como opción para construir figuras en 3D, se encontró la librería jmathplot [7], ambos de libre distibución.

Durante esta fase de construcción, se comenzó el desarrollo, utilizando y generando un producto inicial con la librería Jfreechart, la figura 3D se elaboró utilizando la librería de Jmathplot.

El cuadro de controles ofrecido por Jmathplot, brinda características útiles (entre las que se encuentran zoom más, menos, edición de gráfica y tabla de datos), por lo que se decidió utilizar esta librería, para construir figuras en 2D y descartar el uso de Jfreechart. Finalmente en la fase de transición se verificó el funcionamiento del programa, se elaboró la documentación y se entregó el producto final modelado de comportamiento [5].

2.1. Caso de uso

El diagrama de casos del software desarrollado se visualiza en la figura 1.

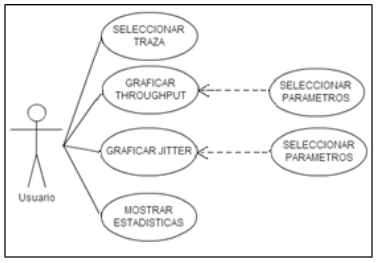

Figura 1. Diagrama de casos de uso

La interfaz gráfica al ejecutar el programa se muestra en la figura 2.

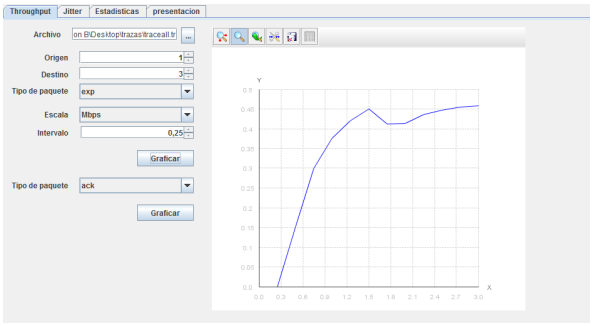

Figura 2. Interfaz gráfica

2.2. Diagramas de secuencia

Los casos de uso y sus diagramas de secuencia se detallan a continuación:

Seleccionar traza: Al iniciar la aplicación lo primero que debe hacer el usuario es la selección del archivo de traza, generado; este se puede visualizar el diagrama de secuencia en la figura 3, figura 4 y figura 5, donde cada uno muestra cómo se crean los objetos correspondientes a su tipo, throghput, jitter y estadísticas.

Graficar throughput: cargado el archivo el usuario podrá construir esta variable. La figura 3, muestra el diagrama de secuencia donde explica el proceso interno del software cuando el usuario elige el tipo de paquete, nodos origen-destino y si la gráfica lo expresará en Mbps o Kbps, de esa manera se crea el objeto gráfica y se ubica en la interfaz de usuario.

Graficar jitter: siguiendo la misma metodología para la variable anterior, la figura 4, visualiza como ejecuta la creación de objetos cuando el usuario selecciona la traza, modificando los parámetros nodo y el tipo de paquete, posteriormente se crea el objeto de tipo gráfica y se coloca en la interfaz de usuario.

Graficar estadísticas: solo cuenta la opción de graficar nodos, se crea el objeto estadísticas y el objeto de tipo gráfica y se colocan en la interfaz, aquí se mostrarán los esquemas de paquetes generados por nodos, paquetes perdidos y una gráfica 3D representando estos valores (figura 5).

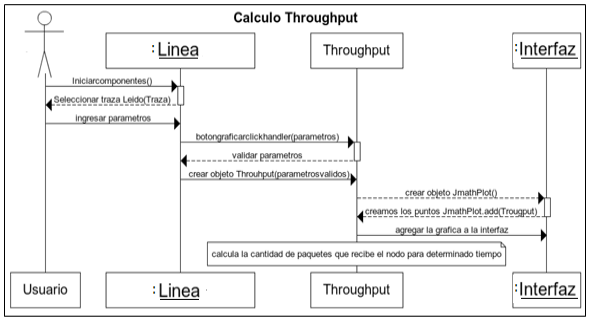

Figura 3. Diagrama de secuencia throughput

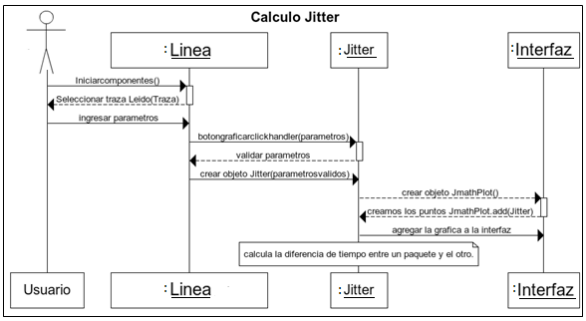

Figura 4. Diagrama de secuencia jitter

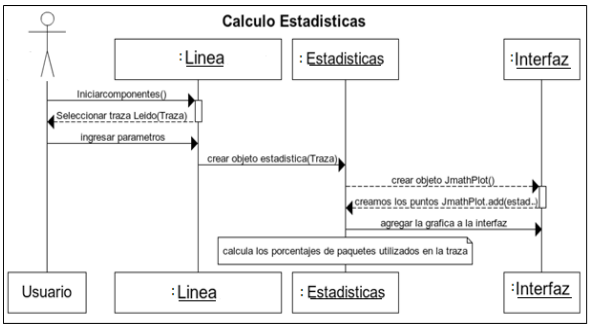

Figura 5. Diagrama de secuencia estadísticas

2.3. Estructura de la traza

Los archivos generados por NS 2 presentan la estructura mostrada en la figura 6. La captura se debe interpretar en forma de columnas, la primera describe el evento r, +-d, el cual indica si se recibió el paquete, si está ingresando, liberando o si el paquete se perdió; la segunda columna describe el tiempo del evento; la tercera y cuarta columna, origen y destino; la quinta el tipo de paquete que se está procesando (ack, tcp, cbr, etc); la sexta muestra el tamaño del paquete; la séptima indica si se está utilizando algún tipo de bandera (no utilizada en el programa), la octava corresponde al identificado de flujo, la novena y décima columna se debe interpretar como puerto destino y puerto origen, la onceava muestra la secuencia del evento y una última que describe el identificador de paquete.

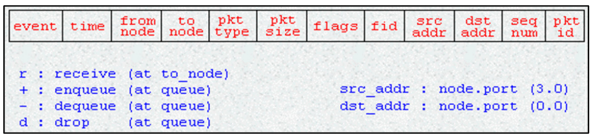

Figura 6. Estructura de las trazas [12]

Tabla 1. Muestra de una traza

r	0,114	0	2	cbr	500	-----	1	0	9	0	0
r	0,120667	0	2	cbr	500	-----	1	0	9	1	1
r	0,127333	0	2	cbr	500	-----	1	0	9	2	2
r	0,134	0	2	cbr	500	-----	1	0	9	3	3
r	0,140667	0	2	cbr	500	-----	1	0	9	4	4
r	0,147333	0	2	cbr	500	-----	1	0	9	5	5
r	0,154	0	2	cbr	500	-----	1	0	9	6	6

La muestra tomada en tabla 1 es un ejemplo de los datos que genera NS-2 cuando se toman datos de la simulación. Tomando los datos de la primera fila, se puede concluir que se está recibiendo un paquete a los 0,114s, entre el nodo 0 y 2, el tipo de datos corresponde a CBR (Constant Bit Rate), con un tamaño de 500 bytes, posee un identificador de flujo 1, y tiene asociado un puerto origen 0, y destino 9 finalmente la secuencia e identificador de paquete es 0.

Los cálculos realizados, para throughput y jitter se realizaron teniendo en cuenta, los flujos recibidos en el nodo, es decir para todos los eventos marcados con la letra r.

Teniendo claro la estructura de la traza, se logró establecer la forma de calcular el jitter y el throughput para la aplicación.

2.4. Calculo de throughput

Basados en [8] [9] [10] [11] y en los scripts hechos en NS-2 se realizó el cálculo de throughput mediante la ecuación (1).

$$Throughput = (bits/escala)/tiempo \quad (1)$$

Donde cada una de las variables tiene el siguiente significado:

Throughput: es el resultado de la cantidad de datos por unidad de tiempo, representado en la ordenada.

Bits: tamaño del paquete en bits, ns2 muestra el tamaño en bytes por lo que se realiza la conversión a bits, simplemente multiplicando por 8, y poder interpretar las gráficas en bits/s.

Escala: especifica si el resultado será medido en Kbps o Mbps.

Tiempo: tiempo en segundos

De la ecuación (2), se determinó las coordenadas en la abscisa, utilizados para graficar.

$$Tiempo = tiempo + intervalo \quad (2)$$

Donde la variable tiempo representa el momento en el que sucedió el evento y la variable intervalo representa cada cuanto se tomará la muestra.

Los cálculos hechos para esta variable, se hacen teniendo en cuenta la cantidad de paquetes que recibe el nodo destino en determinado tiempo.

2.5. Calculo del jitter

Basados en [12] [13] [14] [11], se realizó el cálculo del jitter, a partir de la diferencia de tiempo entre un paquete y el otro de la secuencia de llegada.

Tomando como ejemplo la tabla 1, el primer tiempo que aparece es 0,114s, seguido de 0,120667s y 0,127333s, lo que indicaría que la tabla de datos en la ordenada estaría dada por un primer tiempo de 0,114s, seguida de la diferencia entre el segundo paquete y el primero de 0,006667s y así sucesivamente se establecen los puntos para la ordenada, ya en la abscisa se grafica la columna secuencia, para cerciorarse de la continuidad de los paquetes.

2.6. Estadísticas

Este cálculo se realizó teniendo en cuenta la columna size (tamaño del paquete) de la traza, los eventos marcados con "d" (perdida de paquetes) y los tipos de paquetes.(cbr, exp, ack, tcp, etc). Básicamente se calculó cuantos bytes envió cada nodo por paquete y los bytes perdidos en la simulación.

3. MODELADO ESTRUCTURAL

3.1. Diagrama de clases

El diagrama de clases se visualiza en la figura 7, a continuación se describe cada una de las clases, por las que está compuesto el desarrollo implementado.

Principal: esta clase contiene el método principal "main" para inicialización de la clase interfaz, desde la cual se controla el comportamiento del programa, no tiene atributos.

Interfaz: Tiene todo el diseño gráfico del programa, y su objetivo es leer y almacenar los archivos traza en una lista, utilizando la estructura "try-catch". Este solicita y recibe la información a todas las clases del programa de acuerdo a la selección que realice el usuario. Los atributos que lo componen son de tipo label, TextArea, Spinners y TextField. Los métodos que se incluyeron son: public interfaz (), Private void initComponentes (), donde el primero de ellos es el encargado de mostrar la ventana de selección del archivo al usuario y el segundo de preparar y crear los objetos necesarios para el procesamiento de los datos.

Línea: tiene asociado el archivo traza compuesto por los métodos "getters y setters", cada uno tiene el nombre de cada una de las columnas que conforman la traza. Los atributos son de tipo "double y String" de acuerdo al dato que suministra la traza. Los métodos son del tipo "getters y setters" para el almacenamiento y solicitud de los datos obtenidos de la traza seleccionada.

Datos: es una clase auxiliar de donde se toman solo los datos requeridos para hacer el cálculo estadístico que se muestra en la interfaz gráfica 3D, para posteriormente pasar por parámetros la información a la clase Estadisticas 3D. Los atributos son de tipo "double", para manejo del nodo origen destino y tamaño del paquete, además incluye el método sumar, que es el encargado de realizar el cálculo para determinar los bytes usados en la simulación.

Estadísticas 3D: prepara los datos para la generación de las figuras 3D. Utiliza un atributo de tipo "array double", para almacenar la información que se va a imprimir, y está compuesto por el método "public Estadisticas3d" que recorre el array de tipo "double", para obtener los datos necesarios, que se utilizarán en las figuras 3D, y por medio del método" public datos" se entrega la secuencia a la interfaz.

Estadísticas: tiene como responsabilidad calcular cuántos paquetes perdidos y procesados por la simulación, valiéndose de los cálculos hechos en la clase "paquetestadistica". A partir

del método "estadísticas", se comparan los diferentes paquetes, se recorre el archivo y se compara qué tipo de paquetes tiene la traza con el listado de datos que pueden existir en NS-2 [15], los tipos de datos perdidos y bytes generados (acertados), se almacenan utilizando los métodos "AgregarA y AgregarP".

Jitter y Throughput: cada uno crea un objeto que procesa los datos para generar las secuencias que se requieren para construir la gráfica "jitter y throughtput", utilizando las ecuaciones (1) y (2). En este caso los atributos son de tipo "double y String" y cuenta con los métodos "Jitter y Throughput" que básicamente utilizan las ecuaciones (1) y (2).

Throughputpaquetes: Esta clase realiza el cálculo del rendimiento total de la simulación, a diferencia de la clase "throughput", esta no tienen en cuenta la columna "evento" de la traza, lo que hace que calcule el tráfico de datos perdidos y recibidos, de forma conjunta. El atributo es tipo "double" y El método utilizado en este caso realiza la operación de calcular el throughput total utilizando la ecuación (1), con la operación "Troughputpack" y retorna los datos a la interfaz para imprimir.

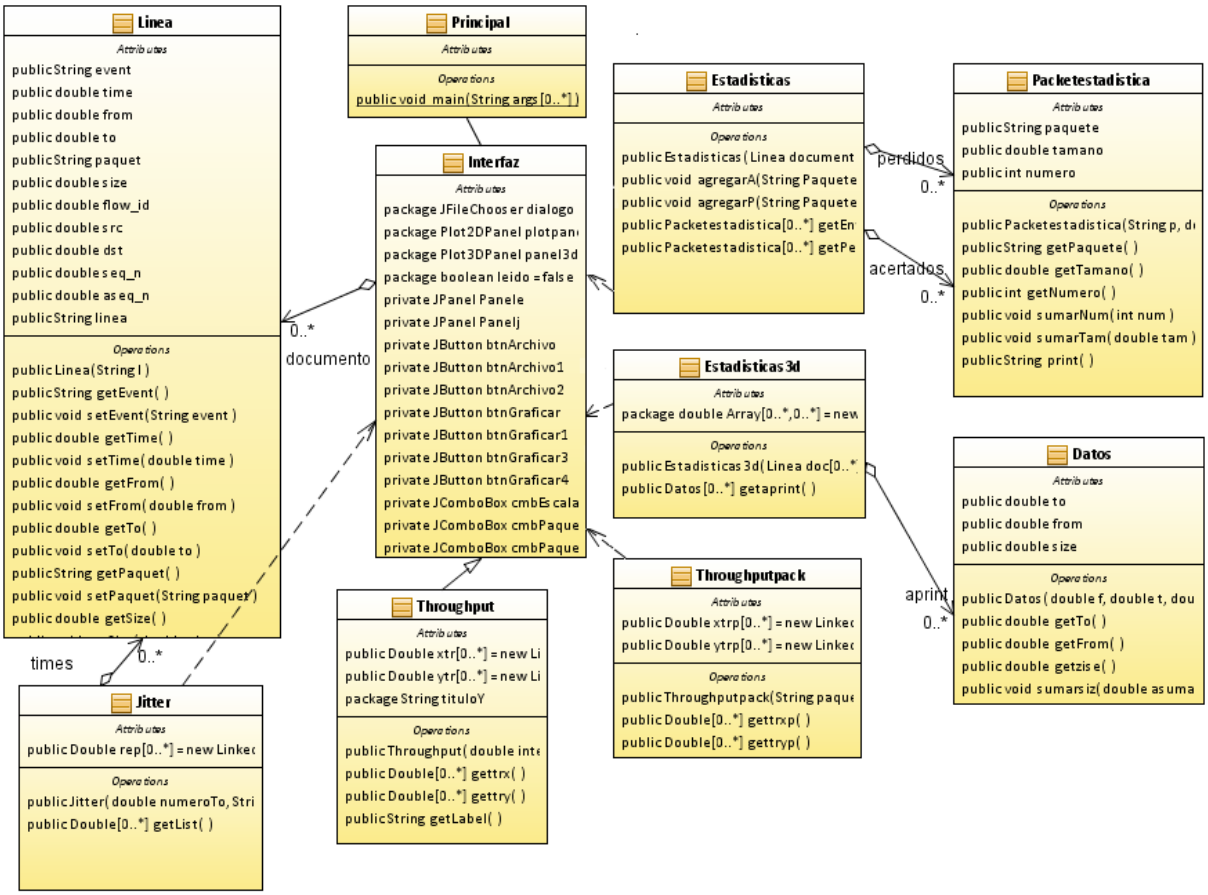

Figura 7. Diagrama de clases

3.2. Librerías utilizadas

Son del tipo jmathplot, jmatharray y jmathio [7]. Estas librerías forman un conjunto de herramientas para realizar graficas de tipo estadístico y matemático, fueron elegidas porque permiten crear figuras 2D, 3D, además poseen cuadros de control que permiten manejar, tabla de valores, permiten la personalización de la gráfica y es de libre distribución [16].

4. ANÁLISIS DE RESULTADOS

Como se mencionó previamente, este proyecto generó un software basado en Java capaz de representar las variables jitter y throughput de una forma sencilla y transparente al usuario final; la veracidad de las gráficas y la velocidad de procesamiento fueron dos factores que se tuvieron en cuenta para el producto final.

4.1. Procesamiento

Con el software NS-2 usandolo sobre la distribución de Linux Suse 12 [17], se generaron archivos tras de diferentes tamaños para validar la cantidad de registros capaz de procesar. Los resultados se muestran en la tabla 2. La prueba se realizó sobre una estructura de 3Gb de RAM, procesador Dual Core de 2.2Ghz. Los tiempos de procesamiento se fueron incrementando y generó bloqueos en la aplicación, con archivos superiores a los 10 MB.

Tabla 2. Prueba realizada trazas

Tamaño (MB)	Tiempo Procesando (minutos)
Superiores a 1	Desestabiliza la aplicación
10,7	8,3
9	7,2
8	6
7,2	3,4
6,5	2,34
6,1	1,80
5,6	1,40
5,3	1,20
3,8	1,10
3,5	1
3,3	55
3	50
2,7	48
2,5	45
2	20
1,6	18
1,3	15
1	12

4.2. Gráficas

Las figuras generadas con el software desarrollado se compararon con [12] [8] y utilizando la herramienta "tracegraph", teniendo como referencia los paquetes recibidos; se obtuvieron estructuras iguales en el 97,3 % de los casos, lo que permitió validar que las fórmulas utilizadas son correctas. La figura 8, visualiza una muestra del jitter, utilizando un script "awk", "columns" y la herramienta "gnuplot" [18]. La figura 9 muestra como solo cargando el archivo y seleccionando los parámetros deseados se obtiene la variable que se quiere estimar en el tiempo a partir del desarrollo creado, donde es claro que son muy similares.

Se generó el throughput con "tracegraph" propio de NS-2 (figuras 10 y 11) y el resultado se comparó con el software desarrolla-do. Los resultados fueron muy similares (figura 12). La figura 13, muestra un ejemplo de una gráfica 3D generado por la aplicación.

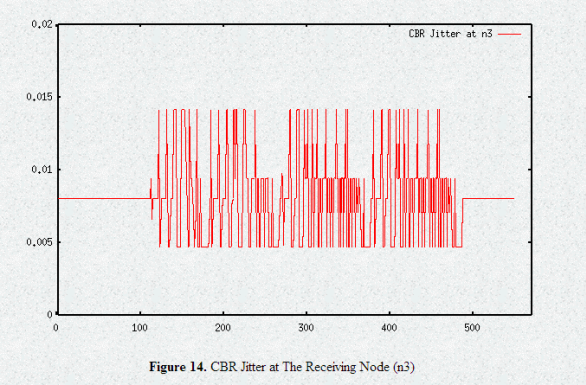

Figura 8. Jitter con awk, colums y gnuplot [12]

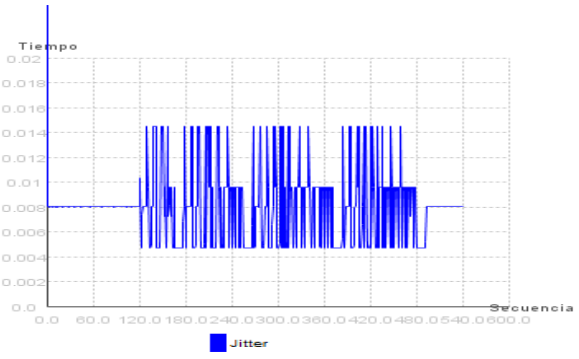

Figura 9. Jitter generado con el software desarrollado

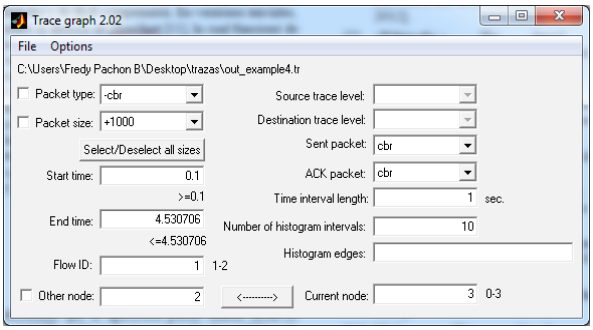

Figura 10. Cuadro opciones Tracegraph

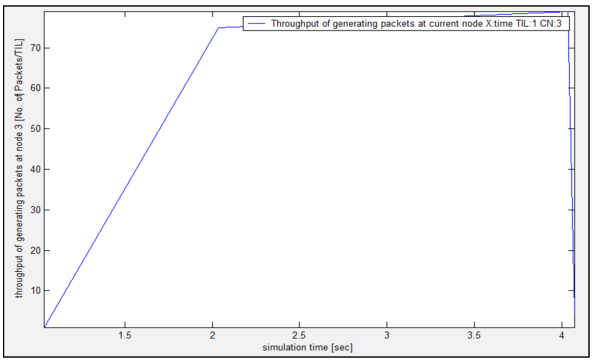

Figura 11. Throughput con tracegraph

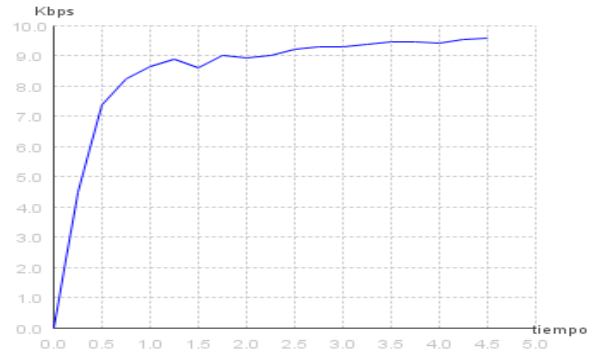

Figura 12. Throughput con el software generado

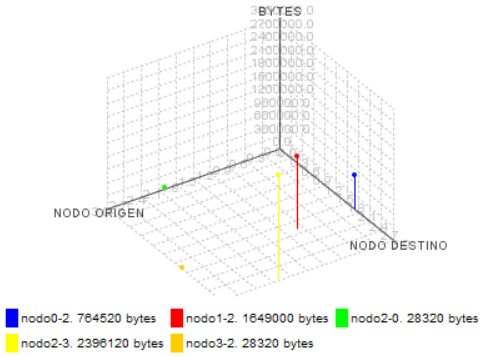

Figura 13. Total paquetes recibidos entre nodos

5. CONCLUSIONES

El software se diseñó con clases que son gratuitas y de fácil comprensión. En versiones iniciales, se utilizó la librería de "jfree-chart" [6], la cual funcionó de manera correcta, es una librería bastante robusta pero que no cuenta con la opción de figuras 3D, donde se pretendía graficar tres variables, nodo origen, destino y total byes recibidos. Con la librería de "Jmathplot" se logró la implementación de la gráfica 3D y 2D que posee propiedades interactivas.

En las gráficas de throughput generadas por "tracegraph", se observaron diferencias, en comparación con el software desarrollado, debido a la gran cantidad de parámetros que se pueden seleccionar y que generan variaciones al ser comparada con la aplicación desarrollada, pero que en términos generales son cercanas al comportamiento de la red simulada. Los parámetros seleccionables en "tracegrahp" se observan en la figura 10.

Se determina que la aplicación puede utilizar archivos hasta 10.7MB, equivalente a 220500 líneas de texto; para procesar archivos de mayor tamaño desestabilizan la propuesta desarrollada, porque en el proceso de lectura de archivos se utilizó el método "FileReader y BufferReader", los cuales leen línea a línea el archivo traza y arman la estructura del archivo en una clase diferente, para posteriormente proceder a generar la figura de acuerdo a los parámetros seleccionados por el usuario, esto causa gran cantidad de procesamiento y consumo de memoria.

Referencias Bibliográficas

[1] NS-2; The Network Simulator - ns-2. [En línea], consultado en Agosto 10 de 2012, disponible en: http://www.isi.edu/nsnam/ns/.

[2] Manual NS-2. [En línea], consultado en Abril 14 del 2012 10 de 2012, disponible en: http://www.isi.edu/nsnam/ns/doc/node5.html.

[3] AngelFire. [En línea], consultado en Septiembre 4 de 2012, disponible en: http://www.angelfire.com/al4/esorkor/.

[4] Mathwork. [En línea], consultado en Mayo 27 de 2012, disponible en: http://www.mathworks.com/products/matlab/.

[5] IBM. [En línea], consultado en 17 de Junio de 2012, disponible en: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[6] Jfreechart. [En línea], consultado en Julio 9 de 2012, disponible en: http://www.jfree.org/jfreechart/.

[7] Berilios. [En línea], consultado en Marzo 7 de 2012, disponible en: http://jmathtools.berlios.de/doku.php?id=downloads.

[8] Udin Harun. [En línea], consultado en 25 de Mayo de 2012, disponible en: http://alkautsarpens.wordpress.com/2008/05/15/visualize-trace-file-data-with-gnuplot/.

[9] Wikipedia. [En línea], consultado en 11 de Septiembre de 2012, disponible en: http://es.wikipedia.org/wiki/Throughput.

[10] ISI. [En línea], consultado en Septiembre 7 de 2012, disponible en: http://mailman.isi.edu/pipermail/ns-users/2004-September/044669.html.

[11] ISI. [En línea], consultado en Julio 20 de 2012, disponible en: http://www.isi.edu/nsnam/htdig/search.html.

[12] NS by example. [En línea], consultado en Marzo 30 de 2012, disponible en: http://nile.wpi.edu/NS/.

[13] Tektronix. [En línea], consultado en 11 de Abril del 2012, disponible en: http://www.tek.com/application/jitter-measurement-and-timing-analysis.

[14] Wikipedia. [En línea], consultado en Junio 8 de 2012, disponible en: http://es.wikipedia.org/wiki/Jitter.

[15] NS-2. [En línea], consultado en Abril 19 de 2012, disponible en: http://www.isi.edu/nsnam/ns/doc/node290.html.

[16] Wikipedia. [En línea], consultado en Agosto 1 de 2012, disponible en: http://es.wikipedia.org/wiki/GNU_General_Public_License.

[17] Open Suse. [En línea], consultado en Mayo 22 de 2012, disponible en: http://es.opensuse.org/Bienvenidos_a_openSUSE.org.

[18] Gnuplot. [En línea], consultado en Abril 1 de 2012, disponible en: http://www.gnuplot.info/.