**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE CANTABRIA**

*Proyecto Fin de Carrera*

# INTERFAZ GRÁFICA DE USUARIO PARA LA SIMULACIÓN Y ENSEÑANZA DE SISTEMAS DE COLA MEDIANTE NS-2

## (Graphical user interface for simulating and teaching queuing systems in NS-2)

**Para acceder al Titulo de**

**INGENIERO DE TELECOMUNICACIÓN**

**Autor: Rubén Rodríguez Pérez**

**Noviembre - 2012**

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

# INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

**Realizado por:** Rubén Rodríguez Pérez
**Director del PFC:** Klaus D. Hackbarth y Laura Rodríguez de Lope
**Título:** "INTERFAZ GRÁFICA DE USUARIO PARA LA SIMULACIÓN Y ENSEÑANZA DE SISTEMAS DE COLA MEDIANTE NS-2"
**Title:** "Graphical user interface for simulating and teaching queuing systems in NS-2"

**Presentado a examen el día:** 23 de Noviembre 2012

para acceder al Título de

**INGENIERO DE TELECOMUNICACIÓN**

Composición del Tribunal:
Presidente (Apellidos, Nombre): García Gutiérrez, Alberto
Secretario (Apellidos, Nombre): Hackbarth, Klaus D
Vocal (Apellidos, Nombre): Sánchez González, Luis

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

**Agradecimientos**

A mi familia y amigos

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. Introducción y objetivos

La teoría de colas es una herramienta de análisis matemático que concierne a diferentes entornos de investigación y desarrollo, tales como: el análisis y dimensionado de sistemas de telecomunicaciones [1], la investigación de operaciones en el ámbito industrial, el rendimiento de procesos en sistemas informáticos o la maestría en recursos humanos dentro de la empresa.

A través de esta herramienta el analista o ingeniero es capaz de analizar y formular modelos de sistema de servicio real denominados sistemas de cola (SdC) con el fin de alcanzar un equilibrio entre la eficacia del servicio que se desea prestar y la eficiencia del mismo (QoS o grado de servicio).

Dada su relevancia, teórica y práctica, el aprendizaje de esta materia en el ámbito docente de las diferentes maestrías anteriormente mencionadas tiene reservado un espacio importante dentro de sus planes de estudio [2]. Por ello, el desarrollo de aplicaciones orientadas a facilitar su comprensión se considera una meta deseable; máxime si se tiene en cuenta la complejidad del tema que se trata.

La obligada naturaleza aleatoria de los datos a estudiar o la dificultad de los cálculos a desarrollar son sólo algunas de las cuestiones a tener en cuenta cuando se realiza una aproximación al servicio real mediante modelado de sistemas de cola. Es por esto que el concepto de simulación adquiere un especial interés como alternativa complementaria al estudio teórico de la teoría de colas. El desarrollo de novedosas tecnologías y lenguajes de programación o la creciente mejora en capacidad del cálculo computacional convierten a la simulación en toda una herramienta con la que modelar situaciones reales de manera más o menos precisa, pero al menos suficiente.

La dificultad encontrada para localizar simuladores, con carácter gratuito, orientados a simular específicamente sistemas de cola, provee un buen espacio de trabajo, presente y futuro, donde el estudio y adaptación de un paquete de simulación orientado a simular dichos sistemas se revela interesante. Además, la experiencia con

estos simuladores se considera una muestra necesaria como referente para la validación final de resultados de un posible nuevo diseño.

De entre los diferentes lenguajes de programación o librerías existentes para conformar simulaciones, se escoge el paquete software de código abierto Network Simulator 2 (NS-2). Se trata de un potente simulador, de reconocido prestigio dentro del ámbito científico y docente, que integra un buen número de módulos dedicados a la simulación de distintos aspectos de las redes de comunicaciones. Además, su caracterización para simulación de sistemas de cola en sistemas de telecomunicaciones ya ha sido referenciado en varias publicaciones, y más específicamente, en un proyecto anterior a éste realizado en el Grupo de Ingeniería Telemática de la Universidad de Cantabria [3].

Sin embargo, el motor NS-2 posee algunas características que lo señalan como una herramienta de trabajo con una curva de aprendizaje realmente pronunciada.

NS-2 carece de una interfaz gráfica de entrada que facilite la navegación entre las diferentes opciones o parámetros para la simulación de escenarios tipo. Por tanto, es imprescindible el estudio previo e investigación a través de documentación especializada, en ocasiones poco organizada y detallada, o proyectos de diferente índole, surgidos alrededor de NS-2, como fuente de aprendizaje y capacitación con el simulador.

NS-2 obliga al conocimiento, en general, de diferentes lenguajes de programación (C++, Otcl, awk, PERL). Una característica fundamental del simulador es la necesidad de programar scripts Otcl como argumento indispensable para ejecutar una simulación. Además, el procesamiento de resultados, de cara a la obtención de resultados concretos, hace necesario el procesamiento de ficheros resultado de traza ".tr" mediante lenguajes, especialmente indicados para dichas tareas, como awk o PERL.

NS-2 no aporta una interfaz de salida capaz de automatizar la representación y organización de resultados de texto o gráfico (no obstante, el paquete sí integra un

módulo de representación gráfica denominado Xgraph cuya ejecución se debe realizar de manera explícita).

Por todo lo anterior, y con el propósito de mejorar la experiencia con el simulador, en este proyecto se lleva a cabo el desarrollo de una interfaz gráfica de usuario (o aplicación de escritorio) encaminada a facilitar, en concreto, la manipulación del motor NS-2 con aplicación a simular sistemas de cola G/G/1; sin perder por ello, la facultad de simular otros sistemas de carácter general.

La tecnología de diseño empleada para la implementación de dicha aplicación es Java. Se trata de un lenguaje de programación de innegable actualidad e implantación, y que sobresale por poseer una concepción que "favorece" su aprendizaje.

Aunque las interfaces surgidas alrededor del simulador NS-2 son escasas, debido seguramente a la complejidad de acometer dicha tarea, se puede destacar como denominador común en todas ellas el empleo de este mismo lenguaje como base de su programación. Es por ello que la experiencia con este tipo de interfaces; así como, el estudio del entrañado Java aplicado al desarrollo de la interfaz gráfica de usuario se convierte en parte central del trabajo a realizar.

Como resumen de objetivos a cumplir, con el diseño propuesto, se reseña el planteamiento de su programación, de cara a la usabilidad, en función de las tres etapas características de la simulación con NS-2: entrada de datos/configuración, procesamiento y entrega/visualización/organización de resultados. Para tal fin, y dado el carácter integrador buscado con el diseño, se llevará a cabo, también, la explotación del paquete de edición OpenOffice y el paquete de representación gráfica Gnuplot, ambos enmarcados dentro del ámbito de software libre.

Las características principales de la aplicación a diseñar son:

- Codificación de escenario para SdC G/G/1 mediante creación automática de script en lenguaje Otcl.

- Carga y modificación de scripts en lenguaje Otcl para simulación con NS-2.
- Conexión automatizada con motor de simulación NS-2.
- Procesamiento automático de trazas para SdC G/G/1 mediante lenguaje AWK[1].
- Visualización automática de resultados analíticos para SdC G/G/1.
- Visualización automática de resultados por simulación para SdC G/G/1.
- Visualización automática de resultados gráficos para SdC G/G/1.
- Apoyo a la representación gráfica de resultados mediante software Gnuplot[2].
- Apoyo a la representación y organización de los resultados con software de edición OpenOffice.
- Archivo integrado de ayuda y documentación en formato .xml.
- Portabilidad entre diferentes sistemas GNU\Linux.

Por otro lado, si se tiene en cuenta que el motor NS-2 corre exclusivamente bajo distribuciones de sistemas GNU\Linux y que la aplicación diseñada gira en torno a dicho simulador, surge la necesidad de buscar una solución tecnológica con la que construir un espacio de trabajo robusto, y fácilmente transportable a otros equipos con independencia del SO. La solución, de coste cero, se encuentra en la virtualización, y más concretamente en el software Oracle VM VirtualBox.

El objetivo es analizar y poner en práctica este tipo de tecnología, tan de moda en los departamentos TIC[3] de empresas y universidades, a fin de enriquecer el proyecto presente, y otros futuros, y aprovecharse de beneficios colaterales, resultado de su explotación, tales como el ahorro de espacio físico, en energía o la reducción de contaminación medioambiental.

[1] AWK: lenguaje de programación orientado al procesamiento de datos almacenados en ficheros de texto

[2] Gnuplot: software de libre distribución orientado a la representación gráfica

[3] TIC es el acrónimo de Tecnologías de la Información y Comunicación

Finalmente, y en relación al estudio recientemente publicado sobre el uso del software libre en el seno de las universidades españolas [4], se quiere destacar como objetivo conductor, durante la elaboración del proyecto, el compromiso por explotar este tipo de tecnologías generadoras de conocimiento.

## 1.1. Estructura de la memoria

La memoria del proyecto se ha estructurado de la siguiente manera:

El capítulo 2 transmite la importancia de la teoría de colas y su aplicación mediante modelado de sistemas de cola. También, muestra la formulación utilizada para el cálculo analítico de parámetros de rendimiento en sistemas de cola G/G/1.

El capítulo 3 está dedicado a la simulación y sus aplicaciones, dando mayor importancia a la simulación por eventos discretos. También, presenta otros simuladores de sistemas de cola usados para comparar los resultados obtenidos con NS-2.

El capítulo 4 desarrolla las características del simulador NS-2, y su definición para la simulación de sistemas de cola G/G/1.

EL capítulo 5 refleja la importancia de la interfaz gráfica de usuario y su desarrollo mediante lenguaje de programación Java. Además, presenta otras aplicaciones Java realizadas en torno al simulador NS-2.

El capítulo 6 se dedica a la aplicación de escritorio desarrollada y sus características.

El capítulo 7 refleja el estudio realizado sobre software de virtualización como solución de distribución completa y segura.

El capítulo 8 presenta las conclusiones y líneas futuras de trabajo.

# 2. Teoría de colas

## 2.1. Introducción

Cuando uno se pregunta sobre la cantidad de tiempo que a lo largo de su vida pasa esperando en diferentes tipos de cola, la respuesta puede ser realmente abrumadora y apremiante. Existen estudios en los que se cifra dicha cantidad en meses, e incluso años [5] [6]. Lo cierto es que las situaciones de espera son muy comunes en el diario de todo el mundo, se espera para realizar operaciones en el banco, para entrar en una discoteca de moda o para descargar documentos de Internet.

El modelado de estas situaciones como sistemas de cola es el ámbito de aplicación de la teoría de colas [7] [8]. En la Figura 1 se muestra un ejemplo de un sistema de cola simple. Está claro que el cliente[4] quiere esperar lo menos posible por el servicio demandado; sin embargo, el que lo presta debe valorar las diferentes inversiones a realizar para reducir ese tiempo de espera. La teoría de colas ofrece respuestas a la viabilidad o no de dichas inversiones.

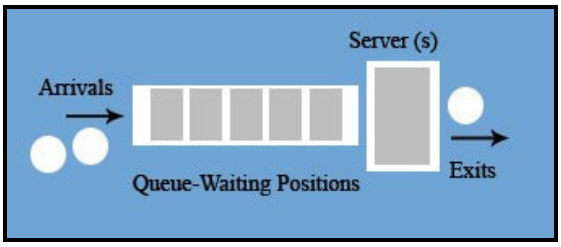


**Figura 1. Gráfico de un sistema de cola simple**

Otra de las preguntas que puede surgir es cuánto de satisfactorio ha resultado la espera en relación al servicio prestado. No se valora de igual manera el tiempo gastado en pagar la compra del día que el tiempo que se gasta en descargar una página web de internet. La exigencia en el grado de servicio es un factor importante. La

[4] cliente = entidad demandante de servicio

teoría de colas es un modelo matemático capaz de valorar y dimensionar las necesidades de una situación de espera a fin de optimizar el servicio prestado.

## 2.2. Aplicaciones

Cabe imaginar que los distintos ambientes donde está presente la teoría de colas son tantos como lo son las veces en que se necesita prestar un servicio, y la capacidad para hacerlo en un tiempo determinado es superada por la demanda, generándose así, una cola o línea de espera. Dependiendo del tipo de servicio ofrecido, el tipo de cliente que genera la demanda, de cómo se genera esa demanda o de cómo se gestiona la cola en que se espera, la teoría de colas está presente en diferentes áreas como: la industria, la economía, la salud, los transportes, la informática y por supuesto, las telecomunicaciones.

Aplicando la teoría de colas se puede realizar un estudio detallado del número medio de pacientes que llegan a un consultorio médico cada día, el tiempo medio que permanecen en el consultorio o lo que tardan de media en ser atendidos por el médico. Los resultados obtenidos modelan la realidad de trabajo del consultorio médico, su capacidad; y permiten establecer, gracias a su carácter estadístico, cuáles han de ser los recursos necesarios en el consultorio para atender de manera adecuada a los pacientes [9].

El diseño de ordenadores es otro ejemplo de cómo el estudio de las líneas de espera facilita la mejora de rendimiento en la ejecución de aplicaciones. En un sistema simple de ordenador la CPU ejecuta paquetes de instrucciones de cada uno de los procesos que se generan cuando un usuario inicia una aplicación o simplemente, escribe algo por teclado. La CPU no puede ejecutar todos los procesos al mismo tiempo y éstos, deben esperar su turno para ser atendidos. El algoritmo usado para la gestión de esta línea de espera se denomina Round Robin. Evidentemente, a mayor complejidad del diseño también tenemos mayor complejidad del sistema o sistemas

de colas a estudiar, como en el caso de sistemas distribuidos cliente-servidor o sistemas síncrono multiprocesador.

Las redes de comunicaciones, su vasta infraestructura y los servicios sobre los que ella se ofrecen ponen de relevancia, hoy más que nunca, la importancia a la hora de utilizar herramientas que ayuden en el diseño y dimensionado (teoría de colas, simuladores, mediciones). La VoIP, los servicios "in streaming" o la videoconferencia son sólo algunas de las aplicaciones en tiempo real para las que la espera es fatal. Los paquetes de datos, en que se fragmenta la información a transmitir, son ahora la entidad demandante de servicio. En su camino hacia el destino atraviesan diferentes sistemas de cola. El objetivo de la teoría de colas es caracterizar esas líneas de espera o formular otras nuevas. Encontrar un modelo matemático que se asemeje lo más posible al comportamiento real del servicio en la infraestructura de red; y así, poder establecer (prever) cuál es el número y tipo de enlaces necesarios, el ancho de banda de los mismos, el tamaño de buffer de espera de los diferentes dispositivos (rooters, multiplexores, nodos centrales) que dan servicio a los paquetes en su camino hacia el destino o las políticas de gestión de los paquetes en su espera para servicio (LIFO[5], FIFO[6], Prioridad). Con todo ello se busca que el servicio tenga la calidad deseada con el menor costo total posible.

Por último, cabe señalar, cómo muchos másteres en recursos humanos, gestión y administración e ingeniería industrial tienen dentro de su programa docente la formulación matemática sobre la que versa la teoría de colas y su carácter probabilístico como una asignatura fundamental (ejemplo: investigación de operaciones, métodos cuantitativos para los negocios) a la hora de preparar a quienes tienen que dimensionar o ajustar los recursos de una empresa (número de trabajadores

[5] LIFO es el acrónimo de Last In First Out
[6] FIFO es el acrónimo de First In First Out

en una tarea, rotaciones en el puesto de trabajo, tiempos de atención al cliente, demoras en procesos, etc.).

## 2.3. Características generales de un sistema de cola

Uno de los errores que se puede cometer a la hora de modelar la realidad con teoría de colas es no analizar convenientemente cuál es el problema o sistema de cola al que uno se enfrenta. Para que los resultados obtenidos a partir del sistema de cola formulado nos permitan evaluar o dimensionar el servicio real con fiabilidad, el sistema de cola deberá haber sido inicialmente bien definido y caracterizado. De otra manera, el problema sólo conseguirá soluciones erróneas.

Se presentan, a continuación, los elementos generales a estudiar en todo sistema de cola:

a) Población potencial: es el número total de clientes que requieren servicio en un determinado momento. Se considera que el tamaño de la población es infinito cuando el número de clientes potenciales en relación a la capacidad del sistema es muy grande; en caso contrario, se define una población finita. La población potencial puede tener tamaño igual a la unidad, un sólo cliente.

b) Proceso de llegadas de los clientes: es la distribución de probabilidad con la que los clientes llegan al sistema (proceso aleatorio). El proceso de llegadas puede ser determinístico, es decir, los clientes llegan al sistema en intervalos de tiempos fijos y conocidos, o aleatorio modelado con una función de densidad de probabilidad para el tiempo entre llegadas.

c) Disciplina de Cola: es el algoritmo utilizado para colocar a los clientes en la cola en el momento de entrar a ser servidos. Se destacan algunas:

- FIFO: el primer cliente que entra en la cola de espera es el primero en ser servido.

- LIFO: el último cliente en entrar en la cola de espera es el primero en ser servido.

- Round-Robin: método por el que todos los clientes son servidos durante un tiempo equitativo hasta que salen del sistema. Normalmente, el orden que se sigue es del primero al último, y vuelta a empezar.

- RSS[7]: los clientes en espera son seleccionados de entre la cola mediante una política aleatoria.

- Priority: los clientes son escogidos para servicio atendiendo a un prioridad estática o dinámica de unos sobre otros.

d) Capacidad del sistema: es el número de clientes que caben en la cola más en servicio al mismo tiempo. El tamaño de la cola puede ser considerado finito o infinito. La facilidad matemática para resolver problemas consiste en considerar un tamaño de cola infinito en tanto en cuanto su capacidad finita real nunca se alcanza[8].

e) Proceso de servicio de los servidores: es la distribución de probabilidad con la que los clientes son servidos (proceso aleatorio). El proceso de servicio puede ser determinístico, es decir, el tiempo que un cliente pasa en servicio es fijo y conocido, o aleatorio modelado con una función de densidad de probabilidad para el tiempo de servicio.

[7] RSS es el acrónimo de Random Service Selection
[8] El caso contrario es un sistema de espera-pérdida

f) Tipo de servicio: se pueden adoptar diferentes estrategias atendiendo al número de servidores, disposición de los servidores (serie o paralelo), etc., como se puede ver en la Figura 2.

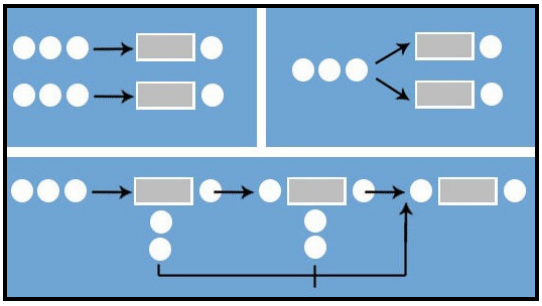

**Figura 2. Estrategias de servicio de sistemas de cola**

g) Red de colas: sistema multietapa formado por varios sistemas de cola independientes o no entre sí; en la Figura 3 se puede ver un ejemplo gráfico. Un ejemplo general es la red de internet. Las redes de colas pueden ser cerradas (la población de clientes es fija y permanece en el sistema indefinidamente) o abiertas (la población de clientes puede variar en el tiempo; los clientes entran al sistema y tras pasar por una o varias colas abandonan el sistema). Un caso particular, si se cumplen determinadas condiciones, lo constituyen las llamadas redes de Jackson abiertas o cerradas.

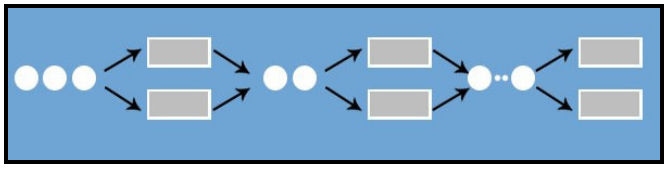

**Figura 3. Red de colas en serie**

## 2.4. Notación de un sistema de cola

La notación de Kendall es el standard usado para describir y clasificar un sistema de cola de manera clara y concisa. Tiene la siguiente expresión:

A/B/C/K/N/D ó A/B/C

- A: representa el proceso de llegadas de clientes al sistema.

- B: representa el proceso de servicio de los clientes en los servidores.

Usualmente, los símbolos A y B pueden tomar los siguientes valores:

  - G: Indica una distribución de llegadas (A) o de servicio (B) de tipo genérico. Es una distribución no especificada, pero bien definida y de la que se conocen sus parámetros media y varianza.

  - D: Indica una distribución con tiempo entre llegadas (A) o de servicio (B) constante.

  - M: Indica un proceso con tiempo entre llegadas (A) o de servicio (B) de tipo exponencial (Proceso de Poisson, Markoviano).

  - Ek: Indica una distribución de llegadas (A) o de servicio (B) de tipo Erlang-k.

  - Hk: Indica una distribución con tiempo entre llegadas (A) o de servicio (B) hiperexponencial de orden k. Se trata de una suma de pesos de k exponenciales.

  - PH: Indica una distribución de llegadas (A) o de servicio (B) tipo fase. Algunas de las distribuciones anteriores son casos especiales de este tipo de distribución (ejemplo: exponencial, Erlang[9]). Se trata de una distribución resultado de una combinación arbitraria de exponenciales.

- C: representa el número de servidores que hay en el sistema. Determina el número de clientes que pueden ser atendidos al mismo tiempo.

[9] Erlang es la medida de tráfico telefónico que debe su nombre a los trabajos de A. K. Erlang

- K: representa la capacidad del sistema. Número de clientes en cola más número de clientes en servicio. Este valor se puede omitir si se considera que el espacio en la cola para clientes es infinito.

- N: representa el tamaño de la población potencial. Si el valor de N es infinito se puede omitir de la notación.

- D: representa la disciplina de cola. Su valor se puede omitir si se trata de una disciplina de cola tipo FIFO.

## 2.5. Parámetros de rendimiento. Formulación de sistema G/G/s

En teoría de colas, el rendimiento de un sistema de cola se mide a partir de parámetros generales derivados bajo la condición de observar al sistema en cuestión en su estado estable o estacionario. Bajo esta premisa, los parámetros de rendimiento son una medida promedio del estado del sistema. El analista de colas puede entonces diseñar o bien evaluar la efectividad del servicio atendiendo a estos valores. El objetivo es lograr un diseño óptimo con el menor costo total posible, o bien identificar cuáles son las razones por las que por ejemplo: los clientes están sufriendo una espera excesiva en el servicio o los servidores están demasiado tiempo ocioso; y por tanto, evitar las pérdidas económicas que esto puede provocar en un negocio o corregir el mal funcionamiento de una aplicación.

La formulación matemática a desarrollar para obtener los parámetros de rendimiento de un sistema de cola es compleja y laboriosa; y en muchos casos, aún no

resuelta. A continuación, se presentan los parámetros de rendimiento característicos para sistemas de cola tipo G/G/s[10]:

- λ o tasa de llegadas: representa el número medio de llegadas de clientes al sistema por unidad de tiempo. También es útil la relación 1/λ o tiempo medio entre llegadas y su varianza.

- μ o tasa de servicio: representa el número medio de clientes servidos por unidad de tiempo cuando el servidor no está ocioso. También es útil la relación 1/μ o tiempo medio de servicio y su varianza.

- ρ o intensidad de tráfico: representa la tasa de tiempo que los servidores están ocupados. Se expresa según la relación:

  $\rho = \lambda / s*\mu$ siendo s el número de servidores en el sistema.

Se debe cumplir la condición necesaria de tener en el sistema de cola un valor de $\rho<1$ para evitar que la cola crezca de manera inexorable y se sature.

La intensidad de tráfico ofrecida a un sistema de cola, pese a no tener unidades, es medida en "Erlang".

- Lq: representa el número medio de clientes en la cola del sistema.

- L: representa el número medio de clientes en el sistema (cola + servidores).

- Wq: representa el tiempo medio de espera de un cliente en la cola del sistema.

[10] Gi/Gi/s nomenclatura Kendall para distribución general e independiente

- W: representa el tiempo medio de espera total para un cliente en el sistema (cola + servidores).

En la Tabla 1 se pueden ver las expresiones analíticas que permiten calcular los parámetros de rendimiento anteriores:

| Parámetros de rendimiento G/G/s | |
|---|---|
| $RR = \frac{\lambda}{\mu} = \frac{\frac{1}{T_a}}{\frac{1}{T_s}}$ <br> $\lambda \equiv Tasa\ media\ de\ llegadas$ <br> $\mu \equiv Tasa\ media\ de\ servicio$ <br> $T_a \equiv Tiempo\ medio\ de\ llegadas$ <br> $T_s \equiv Tiempo\ medio\ de\ servicio$ | (2.1) |
| $p_1 = \frac{cov_a + cov_s}{(2 * (S * \mu - \lambda))}$ <br> $cov_a = var_a * \lambda^2$ <br> $cov_s = var_s * \mu^2$ <br> $S = Número\ de\ servidores$ <br> $var_a = Varianza\ del\ tiempo\ de\ llegadas$ <br> $var_s = Varianza\ del\ tiempo\ de\ servicio$ <br> $p_2 = p_1 * \frac{RR^S}{\left(Factorial(S) * \left(\frac{1-\lambda}{S*\mu}\right)\right)}$ <br> $p_3 = \frac{p_2}{\left(Exp^{RR} * \left(Poisson(S-1, RR, 1)\right) + \frac{RR^S}{\left(Factorial(S) * \left(\frac{1-\lambda}{S*\mu}\right)\right)}\right)}$ <br> $W_q = p_3$ | (2.2) |
| $L_q = \frac{1}{T_a} * W_q$ (2.3) <br> $L = L_q + \frac{\frac{1}{T_a}}{\frac{1}{T_s}}$ (2.4) <br> $W = \frac{L}{\frac{1}{T_a}}$ (2.5) <br> $\rho = \frac{\lambda}{\mu} = \frac{\frac{1}{T_a}}{\frac{1}{T_s} * S}$ | (2.6) |
| -Fórmulas válidas para una sola fuente de tráfico | |
| -Fórmulas según aproximación Allen-Cunneen / Algoritmo en QTSplus 3.0 [46] | |

**Tabla 1. Fórmulas de parámetros de rendimiento G/G/S**

# 3. Simulación

## 3.1. Introducción

En algunas ocasiones, la realización analítica o práctica de un problema concreto no puede ser llevada a cabo por diversas razones como: la dimensión del problema (demasiados datos y variables a tener en cuenta), un desarrollo complejo (alta probabilidad de equívoco), el grado de peligrosidad (procesos industriales), razones éticas (biomedicina), los datos tienen un carácter aleatorio o simplemente, la falta de recursos (no se dispone del espacio y material técnico necesario). La simulación, si bien no es la respuesta definitiva al problema anterior, es una herramienta añadida y fiable con la que obtener una solución próxima a la realidad.

Se trata de una técnica numérica multidisciplinar ampliamente avalada por la comunidad científica, y que se extiende cada vez más sobre otros ámbitos como la empresa o la enseñanza. La razón de su expansión se encuentra en el progresivo crecimiento de capacidad de las distintas herramientas de cálculo computacional, el desarrollo y aplicación de novedosas tecnologías y lenguajes de programación y el bajo coste.

Al simular se experimenta durante un periodo de tiempo sobre una situación cualquiera del mundo real cuya estructura ha sido definida electrónicamente en un computador mediante relaciones matemáticas y lógicas. Actualmente, se pueden encontrar simuladores de todo tipo y enmarcados en diferentes áreas como: simuladores de vuelo, simuladores de procesos industriales, simuladores meteorológicos, simuladores de redes, etc. El objetivo del experimento es ayudar a diseñar nuevos sistemas, evaluar y corregir errores en los ya implantados, favorecer decisiones ante problemas concretos o transmitir conocimientos sobre diferentes temas.

## 3.2. Aplicaciones

Los términos simulación o simulador se han hecho más o menos conocidos a la inmensa mayoría gracias, en gran parte, a la información que nos ha ido llegando a través de noticias en los telediarios o documentales acerca de los logros de la investigación científica en diferentes áreas [10]; y en los cuales, dicha técnica ha tenido un papel importante. Un ejemplo relevante es el simulador del Big Bang creado en el CERN[11] con el que se quieren descifrar los enigmas del universo y responder a preguntas filosóficas sobre la existencia y composición de la materia y antimateria [11].

En el ámbito empresarial el simulador no ha tardado en hacerse un hueco dado su bajo coste en relación a las múltiples ventajas que ofrece [12]. Mediante la simulación el empresario puede por ejemplo: tomar decisiones acerca de la evolución financiera, medir el impacto a priori que puede tener la sustitución de un elemento del sistema por otro, facilitar la etapa de diseño de un proceso, entrenar a los trabajadores en una tarea determinada o poner el simulador al servicio del cliente a fin de que sea éste quien a través de una interfaz simple e intuitiva se haga una idea aproximada de lo que la entidad le puede ofrecer, por ejemplo con un simulador de crédito hipotecario.

En las universidades, la simulación juega dos papeles importantes. Por un lado, en los departamentos de I+D+I se trabaja conjuntamente con organismos públicos y empresas privadas en la construcción de simuladores. Un ejemplo claro de ello lo constituye la incorporación de 13 grupos de investigación de la Universidad de Cantabria en el clúster SICC[12] y el Simulador SGEX, con el objetivo común de innovar en materia de energía renovable marina [13]. Y por otro, el simulador se

[11] CERN es el acrónimo de European Organization for Nuclear Research
[12] SICC es el acrónimo de Sea of Innovation Cantabria Cluster

convierte en una herramienta más dentro de la docencia con la que el profesor puede capacitar más fácilmente al alumno en el aprendizaje de alguna materia. Mediante la simulación el alumno refuerza los conceptos teóricos, los pone en práctica y adquiere una visión más amplia de la problemática estudiada.

A continuación, se mencionan algunos de los simuladores que forman parte del programa docente en la Universidad de Cantabria:

- Comnet III: se trata de un simulador de redes de comunicaciones (configuración de topologías de red, protocolos de comunicación, retardos, etc.) [14].

- Aspen: se trata de un simulador de procesos químicos (operaciones básicas de fluidos, calor y transferencia de materia).

- PSPICE: simulación de circuitos electrónicos analógicos y digitales.

- MIPSIT: se trata de un simulador de microprocesadores (código ensamblador, segmentación).

- Fortran para la simulación electromagnética: se utiliza el lenguaje de programación Fortran para construir programas que simulen diferentes comportamientos de la señal electromagnética.

- CESAR: Centro de Entrenamiento por Simulación en Anestesia, Reanimación y Cuidados Críticos. El aula de Simulación Clínica del departamento de enfermería donde se utiliza ha recibido recientemente el premio UESCE en la modalidad de mejor unidad docente 2010.

- SIESTA: simula el comportamiento de sólidos y moléculas de forma realista a una escala atómica resolviendo las ecuaciones fundamentales que rigen el comportamiento de los materiales. Se trata de un software

simulador reconocido internacionalmente y referenciado en multitud de estudios y artículos de física. La autoría es mayoritariamente española.

Por último, se citan algunas de las aplicaciones que la simulación ofrece específicamente en el ámbito de trabajo y estudio de las telecomunicaciones:

- Práctica en el diseño de diferentes topologías de red standard (wireless, cableada, bluetooth). Configuración de dispositivos. Evaluación de diferentes escenarios. Estudio y aprendizaje de protocolos y algoritmos de comunicación.

- Pruebas de carga y funcionamiento en condiciones normales y en situaciones no deseables de una red ya implantada. Rendimiento. Evaluación de impacto por cambios en la red.

- Desarrollo de nuevos algoritmos y sistemas de comunicación. Programación e integración de los módulos desarrollados en el simulador. Pruebas y evaluación de resultados con el simulador.

- Comparativa de la simulación realizada con resultados teóricos o medidas y experiencias reales.

- Reproducción y estudio de fenómenos interesantes de la red.

- Análisis y escalabilidad de macro-redes.

## 3.3. Caracterización. Tipos de simulación

En el ámbito de la simulación, el sistema real se representa a través de modelos físicos o matemáticos que imitan con el mayor detalle posible, o al menos con el detalle necesario, el comportamiento real del sistema. Si se deja a un lado simuladores basados en modelos físicos, más costosos, como por ejemplo los tanques

simuladores de oleaje o las cabinas de simulación para pilotos; usualmente, los modelos de simulación tienen carácter matemático y lógico.

Como ya se ha dicho anteriormente, el modelo matemático debe ser fiel al sistema que intenta imitar y por tanto, tener claro cuáles ha de ser los elementos que no deben quedar fuera del modelo. Para ello, el sistema se definirá en el modelo a través de un conjunto de entidades de particular importancia que actúan o interactúan para lograr el objetivo que se estudia. Igualmente, será necesario definir un conjunto de variables que representen cuál es el estado del sistema.

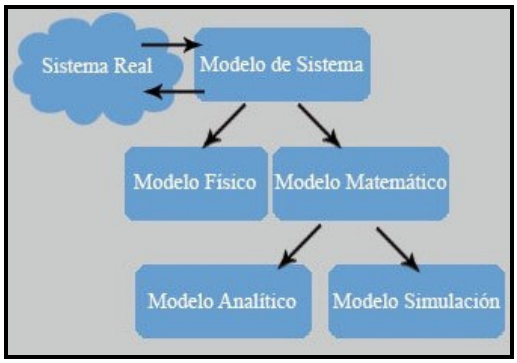


**Figura 4. Clasificación modelos de sistema real**

Como se ve en la Figura 4 dependiendo del tipo de modelo resultante, es decir, su mayor o menor complejidad, el modelo matemático podrá ser clasificado de puramente analítico si la solución buscada es exacta y abarcable; o bien de simulación si la complejidad del modelo así lo exige. Un ejemplo de esto último lo encontramos en la teoría de colas ya comentada y los modelos de sistemas de cola. Como ya se ha comentado, la solución analítica en muchos de los modelos no es alcanzable o bien costosa manualmente y por tanto, se hace necesario el uso del simulador mediante computación.

En este punto, hay que hacer una nueva división a fin de distinguir entre los diferentes modelos de simulación como se ve en la Figura 5:

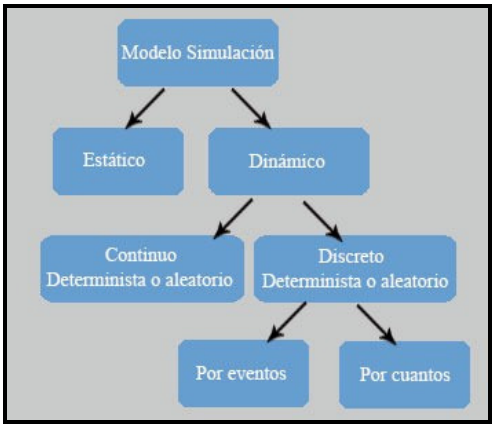


**Figura 5. Clasificación modelos de simulación**

- Modelo estático: representa el sistema para un tiempo dado o bien un sistema en el que la variable tiempo carece de importancia. Un ejemplo es la simulación Montecarlo.

- Modelo dinámico: la variable tiempo juega un papel importante en el sistema a modelar. En este caso, las variables definidas del modelo varían con el tiempo y muestran el estado del sistema en diferentes instantes. Por ejemplo: la simulación del tiempo de vida de un circuito electrónico o los sistemas de cola.

- Modelo determinista: se considera que un modelo es determinista cuando en su diseño no existen elementos de carácter aleatorio. Es decir, las salidas o resultados del modelo son conocidos y únicos una vez se han introducido las entradas necesarias. Un ejemplo es la resolución de ecuaciones diferenciales.

- Modelo probabilístico: en este caso, alguno de los elementos de entrada del sistema modelado tienen carácter aleatorio. Un ejemplo lo constituye la simulación de sistemas de cola con procesos de llegada y salida aleatorios.

- Modelo continuo: cuando las variables que definen el estado del sistema cambian de manera no interrumpida en el tiempo. Un ejemplo puede ser la simulación del caudal de un río o un embalse.

- Modelo discreto: cuando el estado del sistema modelado varía en instantes separados de tiempo. Se dice que el modelo discreto es "por eventos" cuando el sistema actualiza su estado al producirse un suceso o evento; en este caso, el intervalo de tiempo es aleatorio según ocurrencia de próximo evento o suceso (asíncrono). Por el contrario, si el intervalo de tiempo considerado para actualizar el estado del sistema es pequeño y fijo, se dice que el modelo discreto es "por cuantos" (síncrono). Los sistemas de cola son un ejemplo de modelo de simulación discreto "por eventos".

### 3.3.1. Simulación de eventos discretos

Como ya se ha visto en el apartado anterior, el modelo de simulación utilizado para representar sistemas de cola se caracteriza por ser dinámico, discreto y aleatorio. Este tipo de simulación es comúnmente conocida como "Simulación de Eventos Discretos (SED)" [15]. A continuación, se presentan los elementos comunes en todo software simulador de eventos discretos:

- Entidades: son objetos dinámicos de particular importancia dentro del modelo de simulación (paquetes de información o clientes, servidores y colas). A lo largo de la simulación cambiarán de estado debido a su propia actividad o como consecuencia de su actividad con otras entidades. Las entidades pueden ser temporales o permanentes en el sistema dependiendo de su tiempo de vida a lo largo de la simulación. Se agrupan en clases, y por tanto, las entidades pertenecientes a una misma clase tendrán una serie de características comunes o atributos.

- Atributos: son los encargados de diferenciar y caracterizar a una entidad (ejemplo: entidad: paquete de información / atributos: tipo de paquete, tamaño del paquete). El valor del atributo puede estar predefinido en algunos casos, o bien, ser definido y asignado en la simulación.

- Eventos o sucesos: son todas las ocurrencias que a lo largo de la simulación van cambiando el estado del sistema (llegada de clientes al sistema, saturación de la cola de espera, clientes servidos). Los eventos pueden ser seguros si son predecibles, o bien, condicionados cuando su ocurrencia depende del cumplimiento de una serie de condiciones. El tiempo de simulación avanza entre y conforme a la aparición de estos eventos; nada cambia en el sistema entre eventos. La buena y eficiente programación del manejador de eventos es primordial en el software simulador; se encargará de que la secuencia de eventos se produzca de la misma forma que ocurriría en el sistema real.

- Actividades: secuencia de ocurrencias referentes a una entidad. Por lo general, una entidad comenzará una actividad con un evento condicional y la terminará con un evento seguro (ejemplo: espera en cola).

- Variables: almacenan toda la información referente al modelo simulado. Son las encargadas de definir el estado del sistema con cada evento o suceso, al ser actualizadas. Tipos de variables:

  - Variables de entrada: son variables a definir por el usuario final del simulador antes de la simulación. Los valores que pueden tomar son introducidos o seleccionados por el usuario a través de una interfaz de entrada. Dichos valores son necesarios en la definición del modelo que se quiere simular (semilla de aleatoriedad para la simulación, tipo de distribución de llegada o salida, tasa de llegada de clientes, tasa de servicio, número de servidores, etc.).

- Variables de estado: describen el estado del sistema durante la simulación (entidades en cola o espera de recurso, entidades bloqueadas, valores, eventos ocurridos).

- Contadores estadísticos: almacenan información sobre variables que mantienen una función objetivo (número de clientes en el sistema en un instante dado, tamaño de la cola). Se utilizan para grabar información relevante que ocurre durante la simulación, y cuyo análisis final facilita la obtención de resultados y estadísticas necesarios para medir el rendimiento de la simulación realizada (tiempo medio de espera de un cliente en el sistema o en la cola). Muchas de ellas están ya definidas en el software simulador, y otras pueden ser creadas por el usuario experimentado.

- Reloj de simulación: variable global que representa el tiempo simulado. Permite fijar el inicio y final de la simulación, así como los momentos exactos en los que se quiere recopilar información determinada o desencadenar algún evento. El tiempo de simulación irá avanzando de un evento a otro hasta que no haya más eventos a ejecutar, el tiempo de ejecución del próximo evento supere la duración fijada para la simulación o se cumpla alguna condición que finalice la simulación. Una de las mayores ventajas que nos ofrece la simulación es la aceleración del tiempo real de ejecución de manera que el equivalente a minutos del reloj real puede ser simulado en sólo segundos.

- Lista de eventos: como ya se ha comentado, el mecanismo de avance del tiempo es por eventos. Esta variable establece la dinámica de la simulación guardando registro del tiempo en que ocurrirán eventos futuros dentro del modelo simulado.

- Recursos: son un tipo especial de clase de entidad usados por otras entidades en el desarrollo de sus actividades (ejemplo: servidor). Tienen una determinada capacidad que puede variar durante la simulación.

- Colas: las colas tienen lugar cuando las entidades deben esperar para acceder a un recurso determinado que está ya ocupado o al límite de su capacidad o bien, cuando deben esperar por el cumplimiento de alguna condición.

- Generador de números aleatorios: se utiliza para generar valores independientes de variables aleatorias. Permite definir diferentes distribuciones de probabilidad indispensables para generar con aleatoriedad procesos de llegada de clientes y procesos de servicio (exponenciales, uniformes, weibul, gamma, etc.). La programación o elección de un buen generador de números aleatorios dentro del simulador es determinante en la valoración de resultados a fin de lograr simulaciones independientes o variables aleatorias incorreladas.

- Lenguaje de programación, programa principal y rutinas: el software de simulación puede estar basado en lenguajes propios de simulación (GPSS[13], SIMAN[14] o SLAM[15]) que facilitan la programación específica del modelo a simular gracias a que incorporan determinados módulos, sentencias, librerías o funciones (generador de números aleatorios) reduciendo tiempo y esfuerzo; o bien, mediante lenguajes de alto nivel y de propósito general como Pascal, Fortran o C ++. Finalmente, el simulador debe ser un programa robusto y eficiente en su ejecución

[13] GPSS es el acrónimo de General Purpose Simulation System
[14] SIMAN es el acrónimo de Simulation Analysis
[15] SLAM es el acrónimo de Simulation Language for Alternative Modeling

encargado de aglutinar diversas rutinas propias de este tipo de simuladores como: rutinas de eventos encargadas de simular dicha ocurrencia y generar otras nuevas, rutinas de trazado encargadas de recopilar datos durante la simulación o rutinas de inicialización y finalización de la simulación.

## 3.4. Simuladores de sistemas de cola (software)

En este apartado se presentan simuladores relacionados con la teoría de colas con carácter de software libre. Algunos de ellos ayudarán a la comparativa de resultados obtenidos con NS-2 en la simulación de sistemas de cola G/G/1. Los simuladores estudiados son: AQUAS, Queue 2.0, WinQSB 2.0 y QTSPlus 3.0. Ver anexo I.

# 4. Simulación con Network Simulator (NS-2)

## 4.1. Introducción. Caracterización y análisis

Network Simulator 2 se ha convertido con el paso de los años en un standard para la simulación de redes de comunicaciones [16]. Se trata de un potente software de simulación de reconocido prestigio en el ámbito científico y docente [17].

El paquete NS-2 contiene, a grandes rasgos, diferentes modelos para la simulación como: aplicaciones http, Telnet, ftp o CBR; protocolos de transporte TCP[16] (Reno, Tahoe, Vegas, etc.) o UDP[17]; routing cableado unicast/multicast; modelos de cola como DropTail, RED, CBQ, FQ, SFQ o DRR; capa MAC; medio físico cableado o wireless, etc.

Dentro del mundo universitario, el simulador NS-2 es toda una herramienta para la difusión del conocimiento en el área de las telecomunicaciones. Es aquí donde ha experimentado un mayor impulso y crecimiento debido a varias razones que se exponen a continuación:

- El software al completo se distribuye bajo licencia GPL[18]; por lo que es libre y gratuito su uso, modificación y posterior divulgación. La institución tiene en NS-2 la posibilidad de incorporar tanto a nivel docente mediante laboratorios y clases prácticas como a nivel investigación a través de proyectos, nuevo material de apoyo y desarrollo sin costo alguno.

- Es software de código abierto. Esta característica hace que muchas investigaciones realizadas en el seno de la universidad encuentren en el simulador una plataforma bastante completa en la que poder

[16] TCP es el acrónimo de Transmission Control Protocol
[17] UDP es el acrónimo de User Datagram Protocol
[18] GPL es el acrónimo de General Public License

implementar específicamente el objeto de estudio para su posterior testeo; el investigador puede concentrar su tiempo y esfuerzo en programar o adecuar el simulador según su conveniencia pero haciendo uso de otros muchos elementos que a pesar de ser necesarios son igualmente secundarios y ya contenidos en el software. De igual modo, alumnos de estudios técnicos de diferente índole (informática, telemática, redes) tienen la posibilidad de experimentar de una manera más práctica las diferentes materias de la carrera y lanzarse a crear y desarrollar sus propias ideas dentro del marco de trabajo que el simulador les ofrece.

- La programación del software NS-2 (C++[19]) posee una naturaleza modular donde cada uno de los módulos aglutina y modela características standard de las comunicaciones como pueden ser las referidas a protocolos de comunicación (TCP y sus variantes, UDP), tipos de colas (FIFO, LIFO, RED queue) o topologías de red (cableado, wireless). Dicha naturaleza facilita la incorporación de nuevos módulos creados por diferentes desarrolladores y la mejora de las capacidades del simulador. En torno a estos nuevos proyectos se genera documentación y experiencia específica que es compartida, y posteriormente, aprovechada por otros desarrolladores para mejorar en su trabajo con el simulador.

- El simulador cuenta con abundante documentación [18]. Los usuarios inexpertos pueden iniciarse a través de pequeños manuales donde se exponen las funciones más generales acompañadas de ejemplos prácticos de ejecución (ejemplo: “NS for Beginners”). Para aquellos con necesidades más avanzadas existen manuales donde se explica con más detalle los módulos C++ que lo forman (ejemplo: “The NS

[19] C++: lenguaje de programación C orientado a objetos

manual: formerly notes and documentation") [19] [20]. Otra de las ventajas en este aspecto, derivada de su creciente uso, es la presencia en internet de foros de debate, blogs no oficiales y páginas donde encontrar diferentes ejemplos y proyectos personales que aumentan considerablemente el conocimiento sobre el simulador [21].

- Revisiones, contribuciones y mejoras periódicas. Actualmente, el desarrollo y mantenimiento del simulador es llevado a cabo por varias instituciones a través de diferentes grupos de investigación permanentes como SAMAN[20] y CONSER[21], y contribuciones importantes de otros grupos como ACIRI[22] o Sun Microsystems. La última versión estable de NS-2, y sobre la que se ha trabajado la mayor parte del tiempo en este proyecto, es la 2.34 liberada en junio de 2008 [22].

Pese a las ventajas anteriormente comentadas, se ha de destacar la curva de aprendizaje pronunciada que posee el simulador. El uso avanzado de la herramienta (modificación de módulos existentes o desarrollo de otros nuevos) obliga al conocimiento de diferentes lenguajes de programación como C++ y Otcl[23], o AWK[24] y Perl[25]; y a un estudio detallado de la arquitectura del simulador. Aunque la documentación, como ya se ha dicho, es una ayuda; no está organizada y en muchas ocasiones, se echa en falta una visión más detallada. Por ello, el tiempo y esfuerzo de trabajo con el simulador es elevado, y se ha de combinar con el estudio y aprendizaje adquirido de diferentes proyectos desarrollados en universidades o escuelas técnicas, y

[20] SAMAN es el acrónimo de Simulation Augmented by Measurem and Analysis for Networks

[21] CONSER es el acrónimo de Collaborative Simulation for Education and Research

[22] ACIRI es el acrónimo de International Computer Science Institute

[23] Otcl es un lenguaje de programación TCL orientado a objetos

[24] AWK es un lenguaje de programación para procesar datos (Alfred Aho, Peter Weinberger y Brian Kernighan)

[25] PERL es un lenguaje de programación que toma características de otros lenguajes como C, Sh, AWK, sed, Lisp y otros muchos (Practical Extraction and Report Language)

artículos alrededor de todo el mundo[26].

En el caso de un uso natural del simulador, y concretamente dentro del ámbito docente, el alumno debe superar una serie de inconvenientes como son:

- Falta de interfaz gráfica de entrada. El alumno no puede conocer de manera inmediata las opciones de simulación y por tanto, ha de realizar un estudio previo de las diferentes funcionalidades modeladas (protocolos soportados, tipos de red simulables, etc.) y sus particularidades (parámetros de configuración disponibles). De esta manera, se hace una idea de lo que puede o no puede simular, así como de las condiciones bajo las que realiza la simulación de cara al análisis de resultados.

- Generación de scripts Otcl. Una característica fundamental del simulador es la necesidad de programar scripts Otcl como argumento indispensable para ejecutar una simulación. Dicho programa es el contenedor de todo el escenario de simulación (aplicaciones usadas, protocolos de transporte, topología de red, ficheros de resultados, etc.). El alumno debe adquirir previamente un conocimiento básico del lenguaje tcl[28], y de su definición y aplicación en el entorno de NS-2.

- Falta de interfaz gráfica de salida. El simulador, por defecto, entrega diferentes resultados en forma de ficheros que contienen la información acaecida durante la simulación (ejemplo: ficheros de traza ".tr", ficheros para motor de animación gráfica NAM[29] ".nam" o

[26] Se señalan, a continuación, algunos ejemplos: "Creating and Testing a PoissonProcess Traffic Generator for NS2", "Diseño de un entorno para el estudio de los parámetros de funcionamiento del protocolo TCP", "Interfaz gráfica para generación de escenarios de redes bluetooth compatibles con el simulador NS2", "Towards Comparable Network Simulations", "Simulación y comparativa de mecanismos de conmutación en redes ópticas"

[28] TCL es el acrónimo de Tool Command Language

[29] NAM es el acrónimo de Network Animator

ficheros de monitoreo de colas ".tr"). Por otro lado, el usuario final del simulador tiene también la facilidad de poder programar sus propios procesos de recolección de datos, volcando la información generada en ficheros. Posteriormente, y dada la falta de una interfaz gráfica nativa que lo haga, es necesario realizar un procesamiento de dichos ficheros a fin de obtener, con toda esa información, los datos concretos que fueron objetivo de la simulación en su inicio (ejemplo: throughput del enlace, delay de los clientes, nº medio de clientes en el sistema, nº medio de clientes en la cola, etc.). Esta tarea puede ser llevada a cabo, bien a través de una programación previamente planificada y eficiente en lenguaje C++, mediante programación tcl en el script Otcl de configuración inicial, o bien, de la forma más extendida, programando el procesamiento mediante lenguajes, especialmente recomendables en este aspecto, como AWK o PERL.

Ante la problemática anterior, y para suavizar la curva de aprendizaje, han surgido proyectos orientados a facilitar la usabilidad del simulador mediante diferentes interfaces gráficas de usuario. El mayor inconveniente en este tipo de desarrollos, como cabe imaginar, se deriva de la complejidad para contemplar en una misma interfaz de entrada/salida todos los posibles escenarios de simulación disponibles en NS-2, así como, la concreción en la obtención de resultados. Se nombran a continuación algunas de ellas: NsG2 (interfaz de entrada), NsWorkbench (interfaz de entrada), Nans (interfaz de salida).

El paquete – software de simulación NS-2 se distribuye únicamente para su ejecución en distribuciones de sistemas operativos GNU/Linux[30] (Fedora, Ubuntu, Open Suse, etc.). Se trata de sistemas operativos con carácter de software libre y

[30] GNU/Linux es un sistema operativo combinación de GNU (sistema operativo completo tipo Unix de software libre. Se usa habitualmente con un núcleo denominado Linux) y Linux (núcleo de sistema operativo libre tipo Unix. Se distribuye junto con un paquete de software dando lugar a la distribución Linux)

gratuito que benefician el desarrollo de conocimiento. La instalación del simulador puede resultar en ocasiones complicada dependiendo de la distribución utilizada, y debido a la existencia de alguna dependencia tales como librerías o versión de compilador. Ver anexo II.

Por otro lado, también es posible la ejecución del simulador en sistemas operativos Windows a través de una aplicación denominada Cygwin (software gratuito) [23]. Cygwin es, en pocas palabras, un emulador de Linux. Permite portar software que se ejecuta con API[31] POSIX (Unix) a Windows, usando para ello llamadas a la API Win32 nativa de Windows. La interacción entre las dos API, base de la funcionalidad Cygwin, se realiza gracias a la biblioteca "cygwin.dll". No es una opción recomendable para un trabajo continuado con el simulador.

Actualmente, y dado el éxito que NS-2 cosecha, se está desarrollando paralelamente un proyecto llamado NS-3[32]. Se trata de un nuevo concepto de simulador nacido a partir de NS-2 (y otros, como YANS y GTNetS), con el que no mantiene compatibilidad. En este caso, NS-3 se caracteriza por estar implementado completamente en C++, y contener un mayor y detallado número de funcionalidades modeladas (ejemplo: Ipv4 en detalle, y algo de Ipv6) [24]. Sin duda alguna, NS-3 se está esforzando por mejorar la claridad de su diseño, su escalabilidad, su flexibilidad y la integración de los avances del mundo real. Sin embargo, y como ya se hizo notar en la introducción, este proyecto surge para dar continuidad a otro anterior en el que el simulador NS-2 es su eje central [3]. Por ello, y porque al comienzo del trabajo el simulador NS-3 no tenía todavía la madurez que rápidamente va ganando, NS-2 fue la elección como motor de simulación.

[31] API es el acrónimo de Application Programming Interface
[32] NS-3 es el acrónimo de Network Simulator 3

## 4.2. Arquitectura básica del simulador NS-2

Network Simulator 2 es un simulador de eventos discretos basado en dos lenguajes diferentes de programación. Por un lado, el lenguaje C++ se usa para construir lo que se considera el núcleo del simulador, y los módulos de simulación. Por otro lado, el lenguaje Otcl es el lenguaje utilizado como interfaz. Con estos dos lenguajes, el simulador NS-2, separa la implementación detallada de protocolos y algoritmos característicos de las redes de comunicaciones de la parte de creación de escenarios de simulación.

La idoneidad de uno y otro lenguaje atiende a las propias características del lenguaje utilizado en cada caso. El lenguaje C++, aunque lento en los cambios, es robusto y eficiente en tiempo de ejecución para el procesado y manejo de los datos, y la programación en detalle de los distintos protocolos o arquitecturas de red soportados. El Lenguaje Otcl, aunque lento en tiempo de ejecución, ofrece rapidez en el cambio de los diferentes parámetros configurables de un escenario de simulación, y en la generación de dicho escenario; gracias a Otcl, el usuario tiene la posibilidad de investigar y simular diferentes escenarios en un tiempo reducido.

En resumen, el simulador NS-2 presenta dos jerarquías de clases estrechamente relacionadas: la jerarquía compilada escrita en C++, y la jerarquía interpretada de Otcl. El linkado entre las clases de ambas jerarquías se realiza a través de la interface TclCl. Todo este contenido software puede ser estudiado a través del sistema de archivos que conforma el paquete NS-2.

En la Figura 6 se puede observar el esquema funcional del simulador.

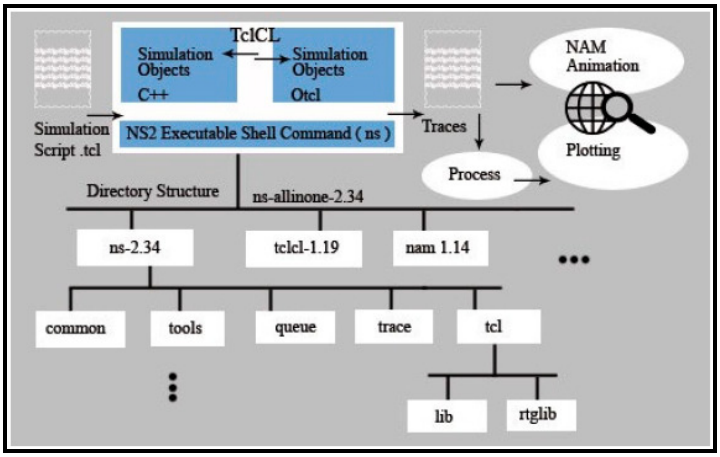


**Figura 6. Esquema funcional del simulador NS-2**

## 4.3. Simulación de sistemas de cola G/G/1 con NS-2

En este apartado se detallan todos los aspectos que se han de tener en cuenta para el uso de NS-2 como simulador de sistemas de cola G/G/1.

En NS-2, el servidor queda definido por un enlace del tipo n (0) ↔ n (1) (simplex o duplex) con tres atributos característicos: ancho de banda en bps, retardo de propagación en sg y tipo de disciplina de cola. En la Figura 7 se muestra un ejemplo gráfico del escenario de simulación.

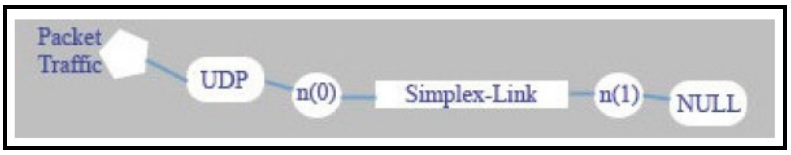


**Figura 7 . Escenario de simulación SdC G/G/1 en NS-2**

En la Figura 8, se puede ver el esquema donde se representan los componentes del enlace en la clase Otcl Link de NS-2:

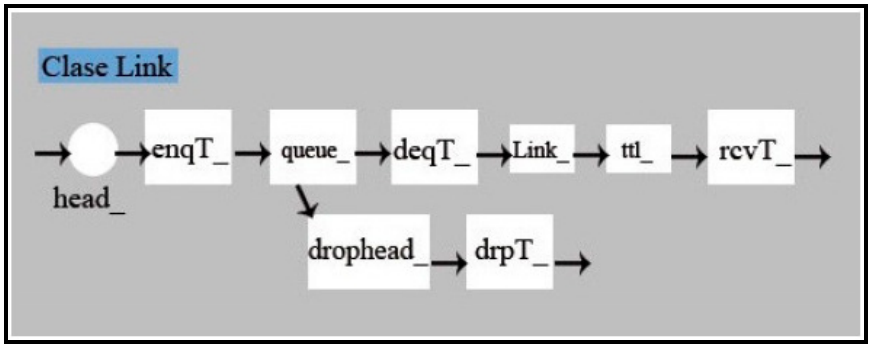


**Figura 8. Clase Otcl Link de enlace NS-2**

Cuando se realiza la simulación, se guarda registro en la traza resultado de los siguientes eventos:

- Paquete entra en la cola: enqT_ (+)
- Paquete sale de la cola y entra a servicio: deqT_ (-)
- Paquete sale del sistema: recvT_ (r)
- Paquete cae de la cola por saturación del enlace: drpT_ (d)

Posteriormente, la traza deberá ser procesada a través de AWK para obtener los parámetros de rendimiento característicos del sistema simulado.

### Programación tcl básica

El usuario debe adquirir, previamente, un conocimiento básico en el uso del lenguaje tcl. Ver anexo III.

### Parámetros de configuración Otcl en NS-2 para simular G/G/1

- Crear una instancia de simulación u objeto simulador

```
set ns [New Simulator]
```

Con este comando se crea el scheduler o planificador de la simulación. Como se verá a continuación, el objeto simulador creado "$ns" tiene asociadas diferentes funciones encargadas de configurar y planificar la simulación.

- Aleatoriedad para la simulación:

```
…
set rep 3
global defaultRNG;
$defaultRNG seed 1130
set arrivalRNG [new RNG]
set serviceRNG [new RNG]
for {set r 1} {$r < $rep} {incr r} {
$arrivalRNG next-substream;
$serviceRNG next-substream;
}
…
```

Dotamos de aleatoriedad a la simulación definiendo la semilla con la que se alimentará el generador de números aleatorios utilizado por NS-2. En el caso de sistemas de cola tipo G/G/1 se debe preservar la aleatoriedad del proceso de llegada de paquetes y del proceso de servicio de paquetes.

- Crear ficheros de traza:

```
…
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
…
```

Se crea un fichero de traza llamado “out.tr” donde NS-2 escribirá por defecto la información acaecida durante la simulación. El fichero “out.nam” almacenará la información escrita por NS-2 durante la simulación, necesaria para poder alimentar el motor de animación gráfico nativo de NS-2 llamado NAM. Nam permite visualizar la topología de red creada y la animación de los paquetes. Además, facilita en su interfaz el uso de diferentes herramientas de inspección. En ambos casos, el comando “trace-all” y “namtrace-all”, respectivamente, permite grabar los datos con formato general.

A continuación, en la Tabla 2 y Tabla 3, se puede ver la definición de campos que son grabados en cada caso:

| Evento | Tiempo | Nodo origen | Nodo Destino | Tipo Paquete | Tamaño Paquete | Banderas | Id Flujo | Origen | Destino | Número Secuencia | Id Paquete |
|---|---|---|---|---|---|---|---|---|---|---|---|
| + | 0.0001 | 0 | 1 | udp | 1424 | ------- | 0 | 0.0 | 1.0 | 0 | 0 |
| - | 0.0001 | 0 | 1 | udp | 1424 | ------- | 0 | 0.0 | 1.0 | 0 | 0 |
| + | 0.015826 | 0 | 1 | udp | 1732 | ------- | 0 | 0.0 | 1.0 | 1 | 1 |
| + | 0.07192 | 0 | 1 | udp | 161 | ------- | 0 | 0.0 | 1.0 | 2 | 2 |
| + | 0.079761 | 0 | 1 | udp | 91 | ------- | 0 | 0.0 | 1.0 | 3 | 3 |

**Tabla 2. NS-2/Captura de traza "out.tr"**

| Event | Time | Source | Destiny | Packet Type | Packet Size | Color | Packet id | Flow id | Port | Src | Dst | S. Number |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + | -t 0.0001 | -s 0 | -d 1 | -p udp | -e 1424 | -c 0 | -i 0 | -a 0 | -x | {0.0 | 1.0 | 0 |
| - | -t 0.0001 | -s 0 | -d 1 | -p udp | -e 1424 | -c 0 | -i 0 | -a 0 | -x | {0.0 | 1.0 | 0 |
| h | -t 0.0001 | -s 0 | -d 1 | -p udp | -e 1424 | -c 0 | -i 0 | -a 0 | -x | {0.0 | 1.0 | -1 |
| + | -t 0.01582573 | -s 0 | -d 1 | -p udp | -e 1732 | -c 0 | -i 1 | -a 0 | -x | {0.0 | 1.0 | 1 |
| + | -t 0.07191961 | -s 0 | -d 1 | -p udp | -e 161 | -c 0 | -i 2 | -a 0 | -x | {0.0 | 1.0 | 2 |

**Tabla 3. NS-2/Captura de traza "out.nam"**

- Crear un nodo de red:

```
...
set n(0) [$ns node] #nodo emisor
set n(1) [$ns node] #nodo receptor
…
```

- Crear un enlace de red:

```
...
set capacity 100000.0
set qsize 100000000
$ns duplex-link $n(0) $n(1) $capacity 0ms DropTail
$ns queue-limit $n(0) $n(1) $qsize
…
```

Se crea un enlace duplex entre los nodos "n(0)" y "n(1)" con una capacidad de 100kbps, un retardo de propagación de 0msg, una política de cola tipo FIFO (o Droptail) y un tamaño de cola límite especificado por el valor de "$qsize". Hay que aclarar que en NS-2 el servidor de paquetes es el propio enlace; de ahí que cuando se habla de cola de servidor se hace referencia a la cola del enlace.

- Crear un agente de transporte en nodo emisor:

```
…
set PacketSize 100000
set src0 [new Agent/UDP]
$src0 set packetSize_ $PacketSize
$ns attach-agent $n(0) $src0
...
```

Se crea un agente de transporte de paquetes tipo UDP llamado "$src0". Se especifica el tamaño de datagrama UDP fijando el valor de la variable "$packetSize_". Finalmente, se asigna el agente "$src0" al nodo emisor de paquetes "n(0)" mediante comando "attach-agent".

- Crear un agente sumidero en nodo receptor:

```
...
set sink [new Agent/Null]
$ns attach-agent $n(1) $sink
…
```

Se crea un agente de recepción de paquetes tipo "Null"; eso significa que una vez los paquetes son recibidos, el agente "$sink" se encarga de descartarlos. Se asigna el agente "$sink" al nodo receptor de paquetes "n(1)" mediante comando "attach-agent".

- Crear relación de conexión entre nodo emisor y nodo receptor:

```
…
$ns connect $src0 $sink
…
```

El comando "connect" no debe ser entendido como una conexión de nivel de transporte. Se trata de un comando "$ns" que posibilita la conexión lógica entre dos objetos "cualesquiera" de la red; en este caso, se encarga de conectar dos objetos – agente del nivel de transporte ("$src0" y "$sink").

- Crear un monitor de cola:

```
…
$ns monitor-queue $n(0) $n(1) [open qm(1).tr w] 0.1
[$ns link $n(0) $n(1)] queue-sample-timeout;
…
```

Se especifican los nodos de la red ("n(0)" y "n(1)") entre los que se quiere realizar el monitoreo de cola. Se puede apreciar que la cola de servidor a analizar es la cola del enlace dispuesto entre los dos nodos. Se crea un fichero "qm(1).tr", con formato de traza, donde se almacenará la información correspondiente al estado de la

cola, con un paso de 0.1 sg, hasta final de simulación. La Tabla 4 muestra los campos que se graban en el fichero creado.

| Tiempo | Origen | Destino | Tamaño Cola Bytes | Tamaño Cola Paquetes | Paquetes en Cola | Paquetes a Servicio | Paquetes Caídos | Bytes en Cola | Bytes a Servicio | Bytes Caídos |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.10000000000000001 | 0 | 1 | 1534.2977613200003 | 1.3597419800000003 | 5 | 1 | 0 | 3775 | 1424 | 0 |
| 0.20000000000000001 | 0 | 1 | 2132.4656225900003 | 4.8530437300000004 | 9 | 2 | 0 | 6973 | 3156 | 0 |
| 0.30000000000000004 | 0 | 1 | 4351.6206531280004 | 8.1358207276000005 | 14 | 5 | 0 | 9205 | 3775 | 0 |
| 0.40000000000000002 | 0 | 1 | 4356.6830371335991 | 9.5193810876000011 | 16 | 6 | 0 | 9377 | 5010 | 0 |
| 0.5 | 0 | 1 | 4310.649824873798 | 9.2779234251999956 | 20 | 9 | 0 | 11728 | 6973 | 0 |

**Tabla 4. NS-2/Captura de traza "qm(1).tr"**

- Definir las distribuciones de probabilidad de tiempo entre llegadas de paquetes y tiempo de servicio:

```
…
set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1.0/30]
$InterArrivalTime use-rng $arrivalRNG
set medium_arrival [expr double(1.0/30)]
set variance_arrival  [expr double(1.0/(30*30))]
…
```

Se especifica una distribución de tiempo entre llegadas de paquetes de tipo "Exponencial". Se fijan los parámetros que definen dicha distribución; en el caso de la distribución exponencial se define el valor medio del tiempo entre llegadas "1/lambda" mediante el parámetro "avg". Se aplica semilla de aleatoriedad mediante comando "use-rng".

Las variables "medium_arrival" y "variance_arrival" almacenan respectivamente, la media y la varianza de la distribución de llegadas definida. Estos valores serán utilizados para obtener los resultados teóricos del SdC simulado.

```
…
set PSize [new RandomVariable/Exponential]
$PSize set avg_ [expr $capacity/(8*33)]
$PSize use-rng $serviceRNG
set medium_service [expr double(1.0/33)]
set variance_service [expr double(1.0/(33*33))]
…
```

Se especifica una distribución de tiempo de servicio de tipo "Exponencial". Se fijan los valores de los parámetros característicos de dicha distribución; en el caso de

la distribución exponencial se define el valor medio de tiempo de servicio. Se aplica semilla de aleatoriedad mediante comando "use-rng". Se calcula y almacena el valor de la media y la varianza de la distribución de servicio definida.

En la Tabla 5 se muestran las distribuciones consideradas, y sus parámetros característicos:

| Distribuciones de Probabilidad ( Llegada / Servicio ) | |
|---|---|
| **Distribuciones Aleatorias** | **Parámetros** |
| Exponencial ( /Exponential ) | avg_ |
| Uniforme ( /Uniform ) | min_ \| max_ |
| Constante ( /Constant ) | val_ |
| Normal ( /Normal ) | avg_ \| std_ |
| Erlang ( /Erlang ) | lambda_ \| k_ |
| Gamma ( /Gamma ) | alpha_ \| beta_ |

**Tabla 5. NS-2/Distribuciones de llegada y servicio**

- Crear proceso para el envío de paquetes:

```
proc send packet {} {
global ns src0 InterArrivalTime PSize
set time [$ns now]
set nextbytes [expr round([$PSize value])]
$ns at [expr $time + [$InterArrivalTime value]] "sendpacket"
set bytes $nextbytes
$src0 send [expr $bytes]
}
```

Se define el proceso encargado de simular el SdC conforme a las distribuciones de llegada y servicio especificadas.

- Crear proceso para finalización de simulación:

```
...
set execnam 1
...
proc finish {} {
global ns tf nf execnam medium_arrival medium_service variance_arrival variance_service
...
$ns flush-trace
close $tf
close $nf
if { $execnam == 1 } { exec nam out.nam & }
```

```
exit 0
}
...
```

Dentro del proceso denominado "finish" se programan todas las tareas que se tienen que realizar una vez haya finalizado la simulación. La variable "execnam" sirve para activar o no la ejecución del motor gráfico NAM.

- Comandos de control de la simulación:

```
…
$ns at 0.000 "sendpacket"
$ns at $duration "finish"
$ns run
…
```

La simulación del escenario de SdC tipo G/G/1 configurado comienza en 0sg. Con estos comandos se especifica el momento de la simulación en que se inicia el proceso de envío de paquetes, y el momento en que debe finalizar la simulación y por tanto, iniciarse el proceso "finish". El comando "run" inicia la simulación del script programado.

### Utilización de nuevos archivos de traza

Los nuevos archivos de traza referenciados en este apartado suponen una facilidad para el procesado y cálculo final de los resultados. Se trata de generar cuatro trazas (o ficheros de texto con extensión ".tr") personalizadas con parte de la información acaecida en la simulación. En la Tabla 6 se muestran los campos que almacenan cada una de las trazas:

| arrivals_queue.tr | | |
|---|---|---|
| Id paquete | Instante tiempo paquete entra en cola | Tamaño del paquete |
| **entry_queue.tr** | | |
| Id paquete | Instante tiempo paquete pasa a servicio | Tamaño del paquete |
| **out_queue.tr** | | |
| Id paquete | Instante tiempo paquete sale del sistema | Tamaño del paquete |
| **length_queue.tr** | | |
| Id paquete | Instante tiempo paquete permanece en cola | Tamaño del paquete |

**Tabla 6. NS-2/Nuevos archivos de traza**

Esta facilidad ha sido desarrollada por un alumno de la universidad de Cantabria, en un proyecto anterior a éste [3]. Siguiendo las recomendaciones del autor ha sido integrada satisfactoriamente en la versión de software simulador NS 2.34 utilizada en este proyecto.

### AWK: procesamiento de ficheros

En este proyecto, se utilizan dos scripts AWK. El script "analitic_results.awk[33]" para el cálculo de parámetros de rendimiento teóricos (SdC G/G/1), y el script "simulation_results.awk[34]" para el cálculo de parámetros de rendimiento simulados (SdC G/G/1). Ver anexos IV y V.

AWK es el lenguaje de programación utilizado para procesar los ficheros de traza obtenidos de la simulación. El intérprete de scripts o programas AWK (awk) forma parte como herramienta standard de los sistemas operativos GNU/Linux [25] [26].

Se trata de un lenguaje óptimo para el procesado de ficheros de texto en los que los datos están distribuidos en forma de tabla (awk ve el fichero a procesar como un conjunto de filas o registros y columnas o campos). La programación en AWK es "similar", en este aspecto, a la programación en lenguaje C. AWK permite realizar operaciones aritméticas, lógicas, trabajar con variables y estructuras de control como:

[33] Script AWK según algoritmo expuesto en Tabla 1
[34] Script AWK adaptado de proyecto "Modelamiento de sistemas de cola G/G/1 en NS" [3]

if, for y while, crear funciones, etc.

La estructura general de un programa AWK es como sigue:

```
BEGIN { acción }
/patrón/ {acción}
END {acción}
```

En BEGIN se especifican las acciones a realizar al comienzo de la ejecución antes de que los datos sean leídos. A continuación, dentro de /patrón/ se programan las tareas que se deben ejecutar por cada fila o registro del fichero de datos a procesar. Finalmente, dentro de END se ejecutan las órdenes programadas una vez los datos han sido procesados.

La forma general de ejecutar un programa AWK, mediante consola, es como sigue:

```
awk [-v var1=valor …] [-f programa.awk] Fichero
```

donde: "awk" es el intérprete de programas escritos en lenguaje AWK, "-v" es una opción para dar valor a variables contenidas en el "programa.awk", "-f" es la opción para pasar el programa escrito en AWK y "Fichero" es un archivo de texto que contiene los datos a procesar.

### Generador de números aleatorios (RNG)

La generación de números aleatorios en NS-2, necesaria para acercar la simulación al comportamiento real del sistema, se realiza a través del llamado Random Number Generator (RNG). Utiliza un generador de números pseudo-aleatorios propuesto por L'Ecuyer llamado "MRG32k3a".

La idea fundamental consiste en alimentar el RNG mediante la variable Otcl correspondiente llamada "seed", dentro del script de simulación. De esta manera, según el valor otorgado a "seed" se toma posición en el stream correspondiente de números pseudo-aleatorios, y dentro de éste, RNG toma de manera secuencial números

aleatorios para la simulación.

Por defecto, NS-2 inicializa la simulación mediante una variable llamada "DefaultRNG" con "seed" igual a 1. De esta forma determinista, las consecutivas simulaciones, si no hay cambio en la alimentación, obtienen resultados iguales (opción apropiada para control de errores o análisis de comportamientos complejos de la red). Sin embargo, para asegurar aleatoriedad es necesario alimentar manualmente el RNG según lo visto más arriba.

Se recalca la necesidad de alimentar cada variable aleatoria, necesaria en la simulación, con independencia de la otra; en otras palabras, que se tomen números aleatorios de diferentes streams para conformar dichas variables. En este caso, se utilizan los comandos Otcl: "[new RNG]" y "use- rng"; esto asegura que el stream seleccionado por NS-2 para cada variable aleatoria definida va a tomar valores aleatorios de diferentes streams.

En el caso particular de SdC G/G/1 se deben crear dos variables aleatorias independientes por escenario de simulación (número de fuentes=1 y número de servidores=1). La variable aleatoria que define el proceso de tiempo entre llegadas, y la variable aleatoria que define el tamaño de los paquetes servidos.

Este aspecto del simulador ha sido objeto de varios artículos científicos en los que se exponen los errores cometidos por el uso incorrecto del generador o las posibles mejoras del mismo [27] [28] [29] [30].

# 5. Interfaces gráficas de usuario y Java

## 5.1. Introducción. Importancia de la GUI. Interfaces para NS-2

En el capítulo anterior se ha hecho referencia a una serie de inconvenientes que dificultan la rápida comprensión y manejo del simulador NS-2. Entre todos ellos se ha hecho especial hincapié en la falta de un entorno gráfico que posibilite una comunicación fluida entre el usuario y la herramienta de simulación. El paquete de simulación es complejo y amplio en contenido; y el uso y manipulación del mismo lleva asociado cargas adicionales relacionadas con la generación de escenarios de simulación mediante "scripts", práctica en diferentes lenguajes de programación o procesamiento de resultados para su posterior análisis y representación gráfica.

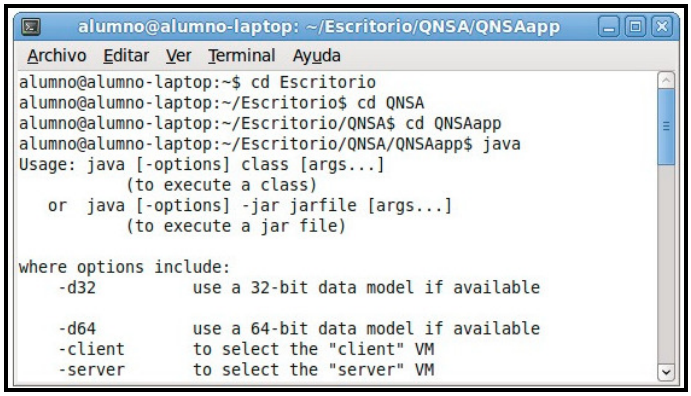


**Figura 9. Captura de consola de sistema GNU\Linux**

En este sentido, la interfaz de trabajo con el simulador queda supeditada al propio entorno gráfico del S.O.[35] GNU\Linux en el que se ejecuta el paquete de simulación. Para cualquier desarrollador o estudiante técnico interesado en el funcionamiento del simulador, dicho entorno se revela más que satisfactorio dada la necesidad de conocer la herramienta en profundidad, y las habilidades personales presupuestas para la explotación y mantenimiento del S.O. anfitrión y las herramientas

[35] S.O. es el acrónimo de Sistema Operativo

de que dispone. Este perfil de usuario no tiene problemas para hacer uso de la consola de sistema, como se puede ver en el ejemplo de la Figura 9, y ejecutar las órdenes necesarias para correr las simulaciones, automatizar procesos, posicionarse en un directorio de trabajo determinado o llamar a otros programas que puedan ser necesarios para la consecución de objetivos. Sin embargo, si lo que se pretende es hacer un uso docente de la herramienta NS-2 para la transmisión de conocimientos en una materia determinada; el alumno o usuario final de la herramienta no necesita conocer los entresijos del simulador, sino sólo aquello que le posibilite y facilite una óptima simulación.

La interfaz gráfica de usuario (o GUI[36]) es un programa informático concebido para hacer más amigable la comunicación entre el usuario final y el software a ejecutar. La GUI se presenta como un conjunto de objetos gráficos de fácil manejo, a través de los cuales se expone la información y acciones disponibles. La GUI hace posible que el usuario final manipule de manera directa el software a ejecutar, y se abstraiga de todo aquello que le debe ser ajeno. Actualmente, con el boom del diseño gráfico y la necesaria usabilidad de las aplicaciones y SS.OO.[37], el concepto de interfaz gráfica de usuario ha cobrado especial relevancia; de tal manera, que se considera un elemento indispensable e inseparable de cualquier software. El catálogo de ejemplos a este respecto es innumerable: desde las aplicaciones para el teléfono móvil, puntos telemáticos de información y plataformas web hasta los muy demandados SS.OO. MAC.

En el anexo VI, se expone una muestra del trabajo realizado por otros desarrolladores, sobre todo pertenecientes al ámbito universitario, encaminado a mejorar la experiencia del usuario final con el paquete de simulación NS-2 mediante la construcción de una interfaz gráfica de usuario [31].

La importancia de estos proyectos surgidos en torno al simulador, al margen de

[36] GUI es el acrónimo de Graphical User Interface
[37] SS.OO. es el acrónimo de Sistemas Operativos

su naturaleza directamente relacionada con los objetivos de este proyecto, reside en la utilización en todos ellos del lenguaje de programación Java como tecnología de diseño.

Entre ellos se destacan: NS Workbench, NSG2 y SimBT como interfaces de entrada, y Nans como interfaz de salida.

La escasez en número de esta clase de aplicaciones pone de relevancia por un lado, la necesidad de dotar al simulador con software de este tipo a fin de mejorar la experiencia del usuario final, y por otro, la dificultad en la consecución de una interfaz gráfica que contemple todas las etapas de la simulación con NS-2: entrada de datos/configuración, post-procesamiento y entrega/visualización de resultados.

En este capítulo se presenta Java como la tecnología de desarrollo utilizada para implementar una interfaz gráfica de usuario que posibilite la manipulación del software NS-2 como simulador de sistemas de cola G/G/1 [32].

## 5.2. Java. Características. Justificación de uso

Java es un lenguaje de programación orientado a objetos (OO) desarrollado por Sun Microsystems. Posee una sintaxis similar a la utilizada en C++, pero con una curva de aprendizaje más rápida [33] [34]. Actualmente, la mayor parte de tecnologías en torno a Java de Sun están liberalizadas bajo licencia GNU GPL / Java Community Process.

Java implica considerar objetos y clases. El objeto es una entidad encargada de aunar en un mismo paquete, datos (estados del objeto) y métodos (comportamientos del objeto). Así, una de las mayores promesas de Java es lograr objetos lo más genéricos posible; facilitar su capacidad de movilidad entre proyectos, es decir, que un objeto creado en un proyecto denominado “A” pueda ser aprovechado directamente en un proyecto “B”, reduciendo de forma drástica el tiempo de desarrollo. Una clase es una especie de plantilla para crear objetos. Cada vez que en un programa se instancia un

objeto se debe especificar la clase a la que pertenece para que el compilador entienda las características de ese objeto. La clase presentará el estado de un objeto mediante atributos, y el comportamiento del objeto mediante métodos a través de los cuales se pueden modificar los valores de atributos.

Se trata de un lenguaje que ha experimentado una considerable expansión e implantación en la última década gracias a una serie de características que lo han convertido en una solución multiplataforma en el diseño de interfaces gráficas de usuario y/o aplicaciones. A continuación, se explican estas características:

- Simplicidad: la "sencillez" del lenguaje radica en su naturaleza orientado a objetos, y en la eliminación de algunos elementos relacionados con el mantenimiento y origen de errores durante el desarrollo, presentes en otros lenguajes OO como C++. Un ejemplo de esto se encuentra en el llamado "recolector de basura" de Java.

- Portabilidad: es la bandera insignia de Java. Cumple con el axioma "escribe una vez y ejecuta en cualquier sitio". Para lograrlo, Java genera su propio lenguaje máquina denominado Bytecode.

- Solidez: se fundamenta en las propias características de programación en Java, y en su resistencia ante posibles errores de programación.

- Código Abierto: como ya se ha mencionado, una gran parte de la tecnología Java (código, API´s y librerías) se encuentra actualmente liberalizada; favoreciendo así, el desarrollo libre y gratuito en el mundo empresarial, universitario y particular.

También, es necesario nombrar otras razones "particulares" que hacen especialmente atractiva la idea de codificar la interfaz de usuario con lenguaje Java.

- La documentación oficial existente en torno a este lenguaje es elevada. Esto ofrece un valor necesario a la hora de afrontar la programación.

- Se trata de un lenguaje muy extendido en la práctica. La experiencia de

otros desarrolladores, fácilmente accesible a través de foros y páginas personales, es una fuente de información importante a la hora de superar problemas en la programación.

- Es un lenguaje de programación que goza de gran presencia en el ambiente empresarial, y su estudio y aprendizaje supone una motivación añadida en el proyecto.

- El resultado final es una interfaz portable fácilmente entre diferentes distribuciones GNU/Linux utilizadas actualmente en la universidad.

- Java dota de escalabilidad al proyecto final. De esta manera, la interfaz puede ser mejorada o ampliada en un futuro sin grandes problemas.

Por último, se explican los requerimientos de sistema para el desarrollo y ejecución de una interfaz de escritorio codificada en Java. La tecnología Java se divide en tres grupos atendiendo a la plataforma destino donde se ejecuta:

- Java ME o J2ME: para su aplicación en terminales o dispositivos de pequeño tamaño como móviles, tablets o PDA´s. Ejemplo: desarrollo de juegos o pequeñas aplicaciones.

- Java EE o J2EE: para su ejecución en entornos de red y web. Ejemplo: desarrollo de applet y servlet.

- Y Java SE o J2SE: para equipos de trabajo como ordenadores de sobremesa o portátiles.

Java SE es la API a utilizar para el desarrollo de la interfaz gráfica de usuario o aplicación de escritorio objetivo en este proyecto. También conocida como J2SE Development Kit o JDK[39] (7.0 es la versión más reciente), consta de una serie de

[39] JDK es el acrónimo de Java Development Kit

programas y librerías (ejemplo: compilador "javac", depurador "jdb", intérprete "java", etc.) que permiten la codificación de este tipo de aplicaciones en Java.

La ejecución de la interfaz de escritorio necesitará de la instalación del Java Runtime Environment o JRE[40], contenido dentro del paquete JDK o también, instalable (ver anexo VII) de manera separada. Ésta es la máquina virtual de Java encargada de interpretar el programa. Actualmente, se distribuye por defecto con todo tipo de sistemas operativos (Windows, Linux, MAC).

Tanto el JDK como el JRE son facilitados de manera gratuita por Sun Microsystems [35].

## 5.3. IDE[41]. Entorno de desarrollo

La implementación de una aplicación de escritorio en Java lleva asociado el trabajo con una serie de herramientas que permiten la generación del código fuente de la misma, depuración de errores, compilación y finalmente, su ejecución. Como ya se ha comentado en el apartado anterior, todas estas herramientas son facilitadas de manera gratuita por Sun Microsystems con el paquete JDK.

El entorno de trabajo por defecto con este tipo de herramientas a través de consola y editor de textos se antoja pobre y lento para el desarrollo de medianos y grandes proyectos. La alternativa a ésto se encuentra en hacer uso de un IDE o entorno de desarrollo integrado.

El IDE es una aplicación en la que se integran las herramientas comentadas anteriormente, y otras muchas que facilitan enormemente la labor del programador. Su función es la de crear un óptimo ambiente de trabajo para la programación.

[40] JRE es el acrónimo de Java Runtime Environment
[41] IDE es el acrónimo de Integrated Development Environment

Actualmente, se pueden encontrar todo tipo de IDE's, bien centrados en las labores de programación para un lenguaje en concreto, o bien, facilitando el trabajo con varios lenguajes de programación.

En lo que se refiere al trabajo con la tecnología Java que ocupa este proyecto se destacan fundamentalmente dos entornos de desarrollo integrado: Eclipse y Netbeans [36]. Ambos, constituyen en sí mismos ejemplos de proyectos multiplataforma realizados con tecnología Java en régimen de código abierto. Las diferencias entre ambos no son realmente destacables, y suponen en cualquier caso, una buena opción para el desarrollo en lenguaje Java. En el anexo VIII se documenta un breve resumen del IDE Netbeans utilizado en este proyecto.

.

# 6. Queuing Network Simulator Application

## 6.1. Características generales de la aplicación

Este capítulo expone el escritorio QNSA[42] desarrollado, y su aplicación a SdC G/G/1. Se trata de una interfaz gráfica de usuario codificada en Java, cuya meta fundamental es crear un entorno amigable para la simulación de sistemas de cola G/G/1 con NS-2.

**Figura 10. Iconografía lanzador QNSA**

A continuación, se muestra un resumen de las principales características de la aplicación:

- Desarrollo de la interfaz gráfica en lenguaje de programación Java.
- Configuración de escenario de simulación para SdC G/G/1.
- Codificación de escenario para SdC G/G/1 mediante creación automática de script en lenguaje Otcl.
- Carga y modificación de scripts en lenguaje Otcl para simulación con NS-2.
- Conexión automatizada con motor de simulación NS-2.
- Procesamiento automático de trazas para SdC G/G/1 mediante lenguaje AWK.
- Visualización automática de resultados analíticos para SdC G/G/1.

[42] QNSA es el acrónimo de Queuing Network Simulator Application

- Visualización automática de resultados por simulación para SdC G/G/1.
- Visualización automática de resultados gráficos para SdC G/G/1.
- Apoyo a la representación gráfica de resultados mediante software Gnuplot.
- Apoyo a la representación y organización de los resultados con software de edición OpenOffice.
- Archivo integrado de ayuda y documentación en formato .xml.

La operabilidad de la interfaz está planificada y desarrollada en base a tres etapas características de la simulación con NS-2:

### 1. Entrada de datos

La aplicación QNSA, a través de la pestaña denominada "Input", facilita la programación del escenario de simulación para sistemas de cola G/G/1. El usuario puede introducir o seleccionar los parámetros característicos para configurar dicho escenario con la ayuda de diferentes soluciones gráficas como botones, campos de texto, combos, etc. La aplicación, automáticamente, se encargará de traducir las distintas acciones generadas por el usuario en las instrucciones de lenguaje Otcl correspondientes. El resultado es el script Otcl requerido por NS-2 como argumento indispensable para ejecutar la simulación.

También, desde esta pestaña, el usuario puede arrancar el motor NS-2 para simular el script Otcl creado. Adicionalmente, se facilita desde la interfaz la carga y modificación de scripts Otcl de manera general.

### 2. Procesamiento de trazas

Como ya se ha expuesto en el capítulo dedicado al paquete NS-2, una vez concluida la simulación, los resultados obtenidos tienen formato de traza. Estos ficheros de traza contienen toda la información acaecida durante la simulación; por tanto, el conocimiento de su estructura es indispensable para entender los eventos

ocurridos durante la simulación, y realizar un análisis primario del caso simulado. Por otro lado, es obligado realizar un procesamiento particular de las trazas cuando lo que se busca con la simulación es obtener un resultado más concreto y organizado.

La aplicación QNSA, de manera transparente al usuario, realiza el procesado automático de trazas con el fin de obtener las medidas de rendimiento del SdC G/G/1 simulado. Igualmente, la aplicación facilita el valor teórico esperado de dichas medidas.

En la pestaña denominada "Process", el usuario puede ejecutar mediante botones los dos procesos anteriores. Internamente, dicho cálculo se solucionará mediante sendos programas escritos en lenguaje AWK e interpretados por el software de sistema awk.

También, dentro de ésta pestaña, el usuario tiene acceso a los distintos resultados teóricos y simulados. Igualmente, tiene la posibilidad de visionar automáticamente resultados gráficos.

### 3. Representación - organización de resultados

QNSA, a través de la pestaña "Output", integra facilidades para el tratamiento de los resultados como: visionado, almacenaje, creación de gráficas o impresión que favorecen, de manera fluida y sencilla, su gestión.

De manera adicional, desde esta pestaña se habilita la posibilidad tanto de crear documentos propios (memorias) como la lectura de otros (documentación, guiones) que puedan ser de ayuda para desarrollar y entender las simulaciones. La aplicación, para tal fin, explota el software de edición OpenOffice, y el software de representación gráfica Gnuplot.

Otra de las características importantes a destacar sobre esta aplicación es que se asienta en el aprovechamiento de diferentes tecnologías con carácter de software libre cuya integración favorece la libre creación y conocimiento.

En la Tabla 7 que sigue se expone un resumen del software involucrado en el desarrollo y explotación de la aplicación QNSA:

| | Java | NS2 | AWK | OpenOffice | Gnuplot | Ubuntu |
|---|---|---|---|---|---|---|
| **Versión** | -JDK 1.6.0<br>-JRE 1.6.0 | -NS 2.34<br>-gcc 4.3 | mawk(-) | 3.3.0<br>SDK | 4.4.0 | 9.04 |
| | Gratuito | Gratuito | Gratuito | Gratuito | Gratuito | Gratuito |
| **Licencia** | GNU GPL | GNU GPL | GNU GPL | GNU GPL | Código abierto gnuplot license | Código abierto |
| **Uso** | -Programación de aplicación<br>-Entorno de ejecución | -Herramienta de Simulación<br>-Compilador | Procesamiento de Ficheros .txt | Organización de Resultados | Representación Gráfica | S.O. virtualizado con Oracle VM VirtalBox |

**Tabla 7. Software utilizado en el desarrollo y explotación de "QNSA"**

La interfaz, como se verá a continuación, quiere ser un entorno de trabajo sencillo con la que el usuario inexperto pueda acercarse por primera vez al simulador NS-2 a través del estudio de SdC G/G/1, sin perder noción alguna de sus características básicas de funcionamiento (generación de scripts Otcl, generación/visionado de ficheros de traza, procesamiento de trazas mediante programación AWK, representación de resultados), y facilitar su manipulación.

## 6.2. Detalle de implementación

Java pone a disposición del desarrollador bibliotecas de clases a través de las cuales se puede acceder a los diferentes componentes (botones, combos...) y eventos que conforman el esqueleto de una interfaz gráfica de usuario. Dos de las bibliotecas más conocidas son "awt" y "swing".

La historia de Java presenta a "awt" como el origen de este tipo de bibliotecas con las que se realizaban las primeras interfaces gráficas de usuario; "swing", constituye la continuación, complemento y mejora de la biblioteca "awt".

Awt utiliza código nativo de la plataforma en su desarrollo. Esto se traduce en una aplicación más rápida en tiempo de ejecución; sin embargo, genera dependencia

funcional de la plataforma para la que se construye la aplicación, y por tanto, reduce la portabilidad del proyecto. Se localiza en el paquete "java.awt".

Swing está escrito en Java. La ejecución de la aplicación es más lenta, sin embargo, no existe, en líneas generales, dependencia funcional de los componentes con la plataforma destino. Se localiza en el paquete "javax.swing".

Actualmente, "swing" es la biblioteca más extendida como elección mayoritaria en la implementación de interfaces gráficas de usuario porque es, al fin y al cabo, una evolución de la biblioteca "awt" donde, además de encontrar correspondencia con los componentes "awt", se localizan otras ventajas y mejoras posteriores.

A continuación, se muestra un resumen de la jerarquía de componentes "swing" utilizada en QNSA:

1) Contenedor de nivel superior "jFrame" u objeto ventana principal que ofrece visibilidad para toda la interfaz gráfica de usuario en cualquier parte del escritorio.

1.1) Barra de menú "jMenuBar". Dentro de ésta se localizan 3 componentes de menú "jMenu"; y a su vez, cada uno de éstos integra items de menú "jMenuItem".

1.1.1) File ["New" "Exit"]

1.1.2) Settings ["Directories QNSA"]

1.1.3) Help ["About QNSA" "Help QNSA"]

1.2) Un panel principal "jPanel". Contenedor para 3 pestañas "jTabbedPane". Cada una de las pestañas da acceso a su vez a un contenedor "jPanel".

1.2.1.1) Inputpanel ("jPanel")

1.2.1.2) Processpanel ("jPanel")

1.2.1.3) Outputpanel ("jPanel")

Los componentes integrados en cada uno de estos tres paneles pueden ser identificados en las capturas de la aplicación QNSA que se ven en la Figura 11, Figura 12 y Figura 13:


**Figura 11. QNSA/Captura de interfaz de entrada "Input"**

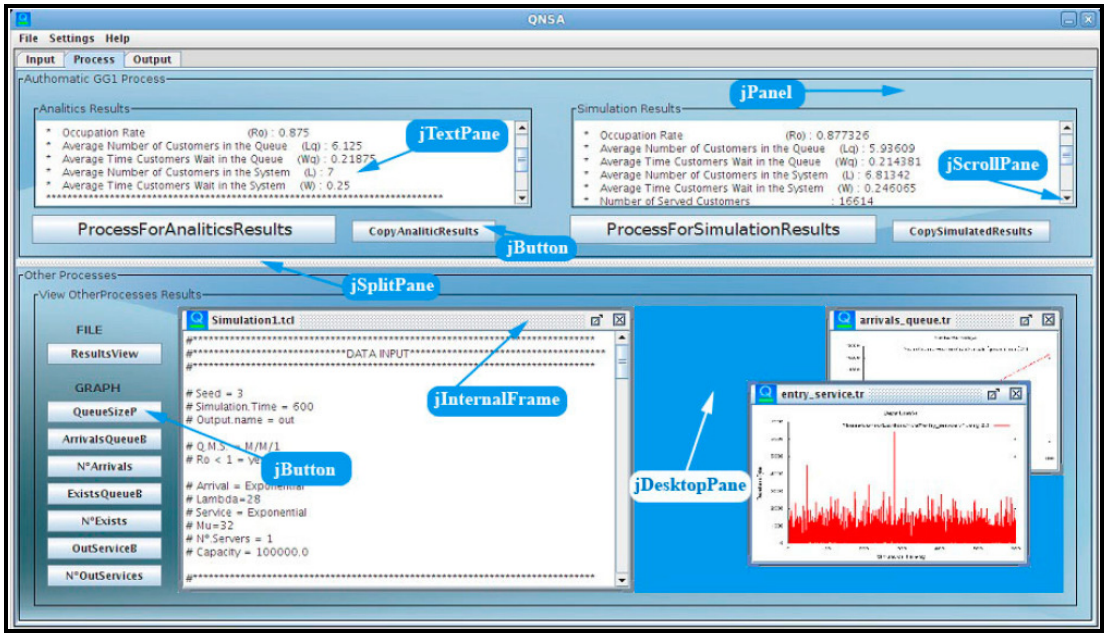


**Figura 12. QNSA/Captura de interfaz de proceso "Process"**

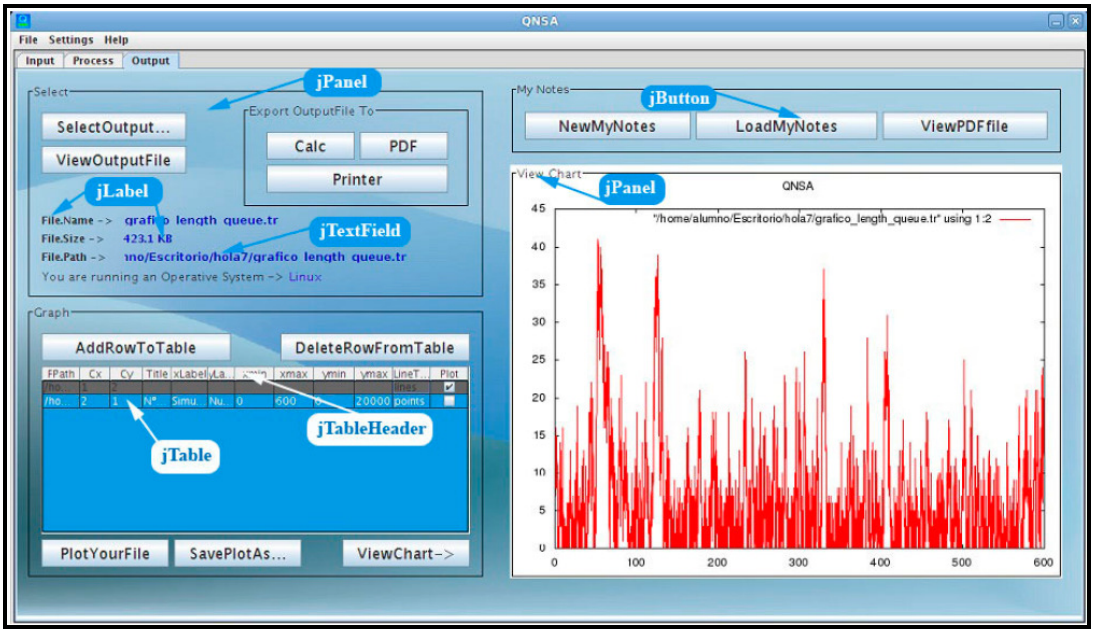


**Figura 13. QNSA/Captura de interfaz de salida "Output"**

### 6.2.1. Software

El paquete de interfaz QNSA está formado por un ejecutable Java denominado "guiGG1.jar" más un conjunto de librerías contenido dentro de la carpeta "lib". El archivo "guiGG1.jar" contiene el programa Java compilado en forma de archivos ".class", más el archivo "Manifest.txt" donde se especifica el nombre de la clase principal del programa, así como, la ruta de localización o "classpath" de las librerías utilizadas.

El arranque o inicialización de la aplicación QNSA lleva consigo la ejecución de una instancia de la máquina virtual Java mediante el comando: "java -jar guiGG1.jar". Éste proceso puede ser realizado por el usuario bien, mediante el icono lanzador de la aplicación diseñado para tal propósito o bien, mediante la definición de dicho comando, para su ejecución a través del botón derecho del ratón, sobre el ejecutable "guiGG1.jar" (sistemas Linux).

### 6.2.2. Barra de menú

La aplicación QNSA predefine al inicio de su ejecución una "carpeta de trabajo" donde se guardan los scripts ".tcl" creados, así como, aquellos que quieran ser simulados con NS-2. De igual modo, éste es el espacio de trabajo donde se almacenan los resultados con formato de traza, texto y gráficos.

El fichero de configuración inicial, donde se especifica dicho espacio y la ruta de otros programas externos (Gnuplot y OpenOffice) explotados en QNSA, tiene la siguiente estructura:

```
# guiGG1config File (guiGG1config.txt)
App=QNSA
# Program Locations Linux
L_gnuplot_path=/usr/local/bin/gnuplot
L_office_path=/usr/lib/OpenOffice/program
#My Working Directory to place Otcl Scripts and Results
L_WorkingD=/home/alumno/Escritorio/hola2
```

En el programa principal se accede a ésta información mediante la clase "Properties". Con esta clase se tiene acceso al contenido de las variables definidas en el fichero de configuración en forma "variable = valor" (ejemplo: L_WorkingD = /home/alumno/Escritorio/hola2).


**Figura 14. QNSA/Diálogo "QNSA Settings"**

Este fichero debe ser modificable por el usuario si se quiere mantener la portabilidad de la aplicación a otros sistemas GNU\Linux. Para realizar dicha operación de manera gráfica, se define en la barra de menú el item "**/Settings/QNSA Directories**" mediante el cual se muestra al usuario el diálogo modal de la Figura 14. De este modo, el usuario puede configurar la aplicación conforme a sus necesidades.

El item de barra de menú "**/help/QNSA Help**" es un espacio de ayuda y documentación donde el usuario puede informarse de diferentes aspectos relacionados con la aplicación. Aunque el uso de QNSA no entraña dificultad en sí mismo, sí es necesario que el usuario esté familiarizado con todos los conceptos que entrañan su funcionamiento, como ya se ha visto a lo largo de los capítulos anteriores.

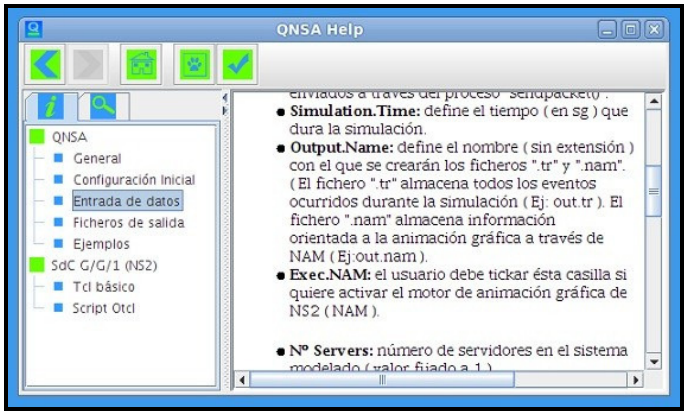


**Figura 15. QNSA/Ventana "QNSA Help"**

Básicamente, la ventana de ayuda, como se puede ver en la Figura15, está compuesta por:

- Una barra de herramientas en la que los diferentes iconos permiten al usuario la navegación a través de documentos (hacia delante, hacia atrás o posicionamiento en la primera hoja del índice), impresión y formateo de la impresión.

- Un índice con los títulos de ayuda:

  - General: explica las funcionalidades básicas de la aplicación; de esta forma, el usuario puede hacerse una idea de lo que QNSA le va a permitir hacer.

  - Configuración inicial: se explica al usuario cómo debe configurar la aplicación para correcto funcionamiento.

  - Entrada de datos: se indica cuál es el significado de los diferentes parámetros de entrada.

- Ficheros de salida: se explica cuál es el contenido y significado de las trazas de salida generadas en la simulación.
- Ejemplos: almacena diferentes casos de uso a fin de aclarar posibles dudas en la introducción de datos.
- Tcl básico: es una pequeña guía sobre el lenguaje tcl.
- Script Otcl: explica el significado general del script de simulación para SdC G/G/1 en NS-2.

La documentación mostrada en el índice está construida en forma de ficheros ".html".

La estructura Java de esta ventana se construye mediante tres ficheros ".xml":

- Uno guarda la relación de contenidos con formato ".html" e imágenes, y su identificador interno. Es el mapa interno de la ventana de ayuda.
- Otra almacena la estructura e información que compone el índice de ayuda. Es decir, el texto mostrado en los nodos y subnodos, y la información o documentación que muestran al usuario.
- Por último, un fichero que construye la estructura de la ventana en sí (barra de herramientas, visor de contenidos, tamaño, etc.).

La opción de menú "**File**" despliega al usuario dos items de menú: New y Exit.

- New: permite al usuario llevar a la aplicación a un estado inicial. Es decir, comenzar una nueva simulación desde cero.
- Exit: sirve para cerrar la ventana de aplicación, y terminar el proceso de JVM lanzado para ejecutar el programa QNSA.

### 6.2.3. Interfaz de entrada: Input

Como ya se ha comentado, el objetivo principal de esta pestaña denominada "Input", y que implementa la interfaz de entrada para datos en QNSA, es la creación del script Otcl necesario para la simulación de SdC G/G/1 con NS-2.

El estado inicial de la aplicación se define a través del método "IncializaFichero()" donde se especifica:

1. Qué componentes, inicialmente, deben estar habilitados o no.

2. El contenido, inicialmente vacío o null, de los componentes dedicados a la introducción de datos por teclado.

3. La opción predefinida en los componentes dedicados a la selección ("jComboBoxes", "jCheckBoxes" y "jRadioButtons").

4. Carga de plantilla o base de fichero Otcl en el panel "Script View". El texto que forma parte de esta plantilla está caracterizado por una fuente de texto de color negro.

Este método, como ya se ha visto, también puede ser accedido mediante la opción de menú "File/New", cada vez que el usuario quiere realizar una nueva simulación partiendo del estado inicial.

Los componentes "Create" y "Load" forman parte de un mismo "ButtonGroup"; esto quiere decir que la selección de uno de ellos excluye la del otro, así, nunca van a poder estar seleccionados los dos al mismo tiempo. En líneas generales, las acciones derivadas de seleccionar "Create" posibilitan la generación del script Otcl G/G/1 bajo las restricciones impuestas por la propia aplicación; evitando, que el usuario pueda realizar modificación alguna sobre el panel de edición de texto "Script View"; donde se codifica dicho script. Por contra, la selección de "Load" permite realizar modificaciones sobre el fichero Otcl cargado en el panel de edición de texto; esta facilidad hace posible también la simulación de otros scripts ".tcl" con carácter particular.

El método encargado de escribir o borrar sobre el fichero Otcl que se crea en el panel "Script View" se llama "EscribeFichero (String Field, String Movement, int Position)", donde:

- Field -- identifica al componente que desencadena una acción de escritura o borrado.
- Movement -- es el tipo de acción a realizar: "A" para escribir y "B" para borrar.
- Position -- es la posición dentro del documento donde se debe escribir texto o a partir de la cual, se debe borrar texto.

Para la programación de este método se definen dos arrays: scriptPosition[] y scriptLength[]. Con el primero se inicializan cada una de las posiciones fijas donde el método "EscribeFichero" puede realizar operaciones de escritura o borrado. La definición de dichas posiciones se realiza tomando como referencia la plantilla de script Otcl, y contando dentro de ella el número de caracteres existentes hasta la posición fija donde la aplicación QNSA va a necesitar manipular instrucciones de lenguaje Otcl.

A continuación, se muestra un ejemplo:

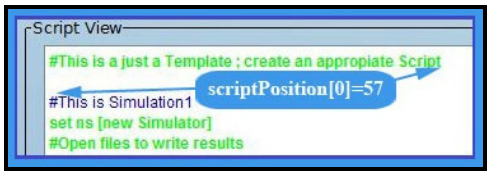


**Figura 16. QNSA/Ejemplo "EscribeFichero"**

En la Figura 16 se fija la posición de carácter número 57 como inicio de instrucción Otcl, dedicada a la definición del nombre de fichero ".tcl", a codificar. De igual modo, cada uno de los componentes gráficos de entrada encargados de desencadenar la escritura de instrucciones Otcl, sobre el panel de edición de texto, tienen fijada su posición de escritura de acuerdo al orden que debe seguirse para la correcta programación del script de simulación G/G/1.

La función del segundo array (scriptLength[]) consiste en almacenar la cantidad total de caracteres que conforman la instrucción Otcl que se desea escribir (dependiente del evento desencadenado) más el número total de caracteres existentes en el script Otcl anteriores a la posición de escritura definida por el evento desencadenado.

Este método es utilizado por cada uno de los elementos que, dentro de la interfaz de entrada "Input", desencadenan una acción de escritura o borrado en la codificación Otcl del script de simulación G/G/1.

En cada componente "jTextField", utilizado para la introducción de datos por teclado, se define:

- Un evento "focusGained ()".
- Un evento "focusLost ()".
- Un método de validación de datos.

En líneas generales, las acciones programadas en este componente funcionan de la siguiente manera:

- Cuando una caja de texto recibe el foco, es decir, el usuario sitúa el cursor del ratón sobre éste componente, se llama al método "EscribeFichero" para que borre del script Otcl el texto que previamente pudiera haber sido escrito como consecuencia de una acción de escritura anterior.
- Cuando la caja de texto pierde el foco, es decir, el usuario sitúa el cursor del ratón en otro componente de la interfaz. Primero, se ejecuta el método de validación de datos para comprobar si la información que el usuario introdujo en la caja de texto es correcta o no. En caso negativo, aparecerá un mensaje de error para advertir al usuario (el mensaje aparece en forma de diálogo modal construido a partir de un componente "jOptionPane"; en consecuencia, no se ejecutará la acción

de escribir sobre el fichero Otcl mediante "EscribeFichero"). Por otro lado, en caso de que la información introducida por el usuario sí sea correcta, el método "EscribeFichero" podrá escribir sobre el fichero Otcl.

- El método de validación de datos de entrada se ejecuta mediante tres clases que extienden la clase de Java "javax.swing.InputVerifier" y ejecutan el método "verify". El uso de este método hace imposible que la caja de texto, objeto de validación, pueda perder el foco en caso de error en la introducción de datos. Entre las clases definidas, una comprueba enteros, otra double y otra verifica caracteres; la utilización de una u otra depende, evidentemente, del tipo de dato esperado dentro de la caja de texto.

Los componentes con lista desplegable o "jComboBox" ("Arrival" y "Service") llevan implementado un modelo encargado de definir los elementos seleccionables de la lista que despliegan. Tanto en el combo titulado "Arrival" como en el de "Service", la selección de uno de los elementos de la lista excluye a los demás. Estos componentes muestran al usuario el tipo de distribuciones de entrada y salida disponibles para la simulación, y sus parámetros característicos. Cuando el usuario selecciona una de las opciones disponibles de la lista, automáticamente aparecen los parámetros requeridos para su caracterización.

Los componentes "jCheckBox" tienen asociado un evento del tipo "itemStateChanged". La activación o no de estos elementos es independiente entre sí. Su activación o desactivación desencadena la escritura o borrado de determinadas instrucciones Otcl.

El componente titulado "Check Ro<1" tiene implementado un evento "actionPerformed". Es utilizado para validar la estabilidad del SdC seleccionado por el usuario, e indicarle, cuál es la nomenclatura de dicho sistema según la notación Kendall. La respuesta a esta acción quedará reflejada en los dos componentes gráficos situados por debajo de éste botón.

- La caja de texto refleja una salida "Yes" si el SdC es estable, y un "No" en caso contrario.

- La etiqueta de salida "Q.M.S.", usada para reflejar el tipo de SdC configurado, puede tomar los siguientes valores: G/G/1, G/M/1, M/G/1, M/M/1, M/D/1, D/M/1, G/D/1, D/G/1, D/D/1, G/E/1, E/G/1, E/E/1, E/D/1, E/M/1, D/E/1, M/E/1.

La condición de estabilidad a evaluar es la siguiente:

$$Ro = \frac{\frac{1}{Ta}}{\frac{1}{Ts}} < 1$$

$Ro = Coeficiente\ de\ utilización\ del\ sistema$
$Ta\ es\ la\ esperanza\ del\ tiempo\ entre\ llegadas$
$Ts\ es\ la\ esperanza\ del\ tiempo\ de\ servicio$

Mediante el uso de los botones "CreateGG1Script" y "SaveScript", el usuario es capaz de guardar un fichero Otcl creado o modificado sobre el panel editor de textos "Script View". Para ello, se hace uso de las clases "ExtFileFilter" encargada de filtrar el tipo de documento a guardar con extensión ".tcl", y la clase "JFileChooser" encargada de abrir el diálogo de sistema con el usuario en modo de operación "guardar".

El botón "LoadScript" lleva implementado un evento "actionPerformed". El objetivo de la acción generada es abrir un diálogo de sistema con el usuario en modo de operación "carga"; así, el usuario puede cargar un fichero de tipo ".tcl" en el panel editor de textos "Script View" (también, se hace uso de las clases "ExtFileFilter" y "JFileChooser").

El botón "CopyScript[43]" tiene implementado un evento "actionPerformed". El objetivo es copiar la porción de texto seleccionada por el usuario desde el panel editor de textos "Script View" al portapapeles del sistema. Se hace uso de la clase "MngTextClipboard" para gestionar el contenido (con formato texto) del clipboard del sistema.

El componente "RUNinNS" es el botón utilizado para arrancar el motor NS, tomando como argumento el script ".tcl" seleccionado por el usuario. Éste "jButton" tiene implementado un evento "actionPerformed". Las clases más importantes utilizadas en este componente son: "Runtime", "Process" y "StreamProc".

- La clase "Runtime" es la que posibilita tener acceso al entorno de ejecución Java, para poder ejecutar programas externos, mediante el método "Runtime.getRuntime()". Es decir, es la clase Java utilizada para crear una interfaz de comunicación entre la instancia de la aplicación en ejecución (QNSA), y programas externos a la JVM.

- La clase "Process" permite identificar a un proceso, y obtener información del mismo. Utiliza el método "Runtime.getRuntime().exec()". Es decir, la JVM recoge el resultado de este comando como un subproceso del que ya se puede obtener información.

[43] Durante la realización de estas acciones se pudo comprobar la imposibilidad de manipular, dentro de sistemas Linux, el contenido del clipboard cuando el tipo de dato a tratar es una imagen. Esto llevó a desechar la facilidad de manipular imágenes a través del clipboard de sistema, que en el caso de S.O. Windows sí funciona correctamente. Se trata, en principio, de un Bug Linux-Java notificado desde hace ya tiempo a la espera de solución en posteriores revisiones.

- "StreamProc" es la clase utilizada para poder controlar la información generada por el proceso mediante los métodos: ".getInputStream()" y ".getErrorStream()". Con el primer método se recogen las salidas normales o habituales del proceso externo (ejemplo: salida generada con un simple puts "hola mundo" dentro del programa a ejecutar); con el segundo método, se obtienen los errores que la ejecución del programa externo haya podido generar (ejemplo: definición incorrecta de una variable dentro del script ".tcl" a simular con NS-2).

Existen diferentes formas de construir el comando "exec()" [37]:

```
public Process exec(String command);
public Process exec(String [] cmdArray);
public Process exec(String command, String [] envp);
public Process exec(String [] cmdArray, String [] envp);
public Process exec(String [] cmd, String [] envp,File dir);
public Process exec(String [] cmdArray, String [] envp,File dir);
```

***command*** -- especifica el comando de sistema a ejecutar.

***cmdarray*** -- especifica un array de comandos a ejecutar.

***envp*** -- array de strings, cada elemento de los cuales tiene una configuración de variables de entorno con el formato nombre = valor, o null si el subproceso debe heredar el entorno del proceso actual.

***dir*** -- el directorio de trabajo del subproceso, o null si el subproceso debe heredar el directorio de trabajo del proceso actual.

Para la ejecución de NS se ha utilizado la forma:

```
public Process exec(cmdArray, null, dir);
```

donde ***cmdArray*** se define como:

```
String[] cmdArray = new String[2];
   cmdArray[0] ="ns"; // comando de ejecución NS
   cmdArray[1] =" "// ruta del fichero ".tcl" a ejecutar
```

y ***dir*** toma el valor del espacio de trabajo definido para QNSA en el fichero de configuración inicial ***"guiGG1config File".***

El panel titulado "QNSA Console" es el contenedor donde se localiza, siempre visible, la herramienta de consola de la interfaz. La razón fundamental por la que se ha decidido colocar dicha funcionalidad en esta pestaña se debe a que es la parte de la interfaz donde la actividad del usuario puede generar mayor número de errores que deben ser controlados y reportados. Se construye a partir de un componente "jTextArea" sobre el que el método "MessagesToConsoleArea()" escribe mensajes de control generados a partir de acciones programadas en los distintos componentes que integran la interfaz. De este modo, todo mensaje de error o salida derivado de la ejecución de programas externos con los que comunica Java o los mensajes de error usados para controlar el uso correcto de la interfaz son reportados al usuario a través de este panel.

El botón "ClearConsole" lleva asociado un método "actionPerformed". Su labor consiste en borrar el histórico de mensajes escrito en la consola QNSA.

La interfaz implementa, como ya se ha visto anteriormente, otro método de reporte de errores que se basa en el uso del componente "jOptionPane". Se trata de mostrar el mensaje de error al usuario mediante un diálogo modal (una vez notificado el error, el usuario debe cerrar el diálogo para poder continuar su actividad con la interfaz). Por ejemplo, este tipo de notificación es utilizada cuando se ejecuta el método de validación de datos dentro de los componentes "jTextField" de la pestaña "Input", y el resultado es un mensaje de error. La Figura 17 muestra un ejemplo:


**Figura 17. QNSA/Mensaje de error**

#### 6.2.3.1. Resumen de parámetros de entrada

- Simulation.Name: especifica el nombre (sin extensión) con el que se guardará el script de simulación ".tcl" creado.

- Seed: define un número entero con el que se dotará de aleatoriedad a la simulación. Se utiliza para alimentar el generador de números aleatorios utilizado por NS-2.

- Packet.Size: define un tamaño de datos (bytes). Este número debe ser los suficientemente grande como para que el protocolo UDP, implementado en NS-2, no limite a un número demasiado pequeño el tamaño de los datagramas o paquetes que posteriormente serán creados y enviados a través del proceso "sendpacket()".

- Simulation.Time: define el tiempo (sg) que dura la simulación.

- Output.Name: define el nombre (sin extensión) con el que se crearán los ficheros ".tr" y ".nam". El fichero ".tr" almacena todos los eventos ocurridos durante la simulación (ejemplo: out.tr). El fichero ".nam" almacena información orientada a la animación gráfica a través de NAM (ejemplo:out.nam).

- Exec.NAM: el usuario debe ticar ésta casilla si quiere activar el motor de animación gráfica de NS-2 (NAM).

- S.Replication: el usuario debe ticar esta opción si quiere realizar un promedio de resultados, obtenidos en sucesivas simulaciones.

- Nº Servers: número de servidores en el sistema modelado (valor fijado a 1).

- C Server: capacidad del servidor (bps).

- Arrival/Service: permiten seleccionar el tipo de distribución de llegadas y de servicio a simular, y la definición de sus parámetros característicos. En la Tabla 8 se muestran las opciones disponibles.

| Distribuciones de Probabilidad ( Llegada / Servicio ) | | |
|---|---|---|
| **Distribuciones Aleatorias** | **Parámetros Otcl** | **Parámetros QNSA** |
| Exponencial ( /Exponential ) | avg_ | Lambda ( Llegada )<br>Mu ( Servicio ) |
| Uniforme ( /Uniform ) | min_ \| max_ | Min \| Max |
| Constante ( /Constant ) | val_ | d |
| Normal ( /Normal ) | avg_ \| std_ | Mu \| Sigma |
| Erlang ( /Erlang ) | lambda_ \| k_ | Lambda \| k |
| Gamma ( /Gamma ) | alpha_ \| beta_ | Alpha \| Beta |

**Tabla 8. QNSA/Distribuciones de llegada y servicio**

- Check Ro<1: permite valorar la estabilidad del SdC modelado por el usuario.

- Q.M.S.: indica al usuario el tipo de SdC que ha modelado con su selección; según la notación de Kendall. Puede tomar los siguientes valores: M/M/1, M/G/1, G/M/1, G/G/1, D/G/1, G/D/1, D/D/1, M/D/1, D/M/1, G/E/1, E/G/1, E/E/1, E/D/1, E/M/1, D/E/1 ó M/E/1.

- Queue Monitoring: permite crear un fichero de traza “qm(1).tr” donde se almacena información de monitoreo de cola.

- Finish: esta casilla debe ser ticada por el usuario para crear el proceso “finish()” dentro del script de simulación “.tcl”.

### 6.2.4. Interfaz de proceso: Process

Esta pestaña tiene como objetivo fundamental el procesamiento interno, y la presentación automática de resultados al usuario.

Para hacer posible la vista de todos los resultados en una misma pestaña se hace uso de un componente "jSplitPane". Su finalidad es dividir la vista en dos zonas separadas por una barra deslizante. El usuario, desplazando dicha barra hacia arriba o hacia abajo, puede especificar la cantidad de área visible de una zona u otra, y por tanto, maximizar la vista de resultados actual.

La zona superior alberga dos paneles ("Analitic Results" y "Simulation Results") dedicados a la visualización, con formato de texto, de resultados teóricos y simulados. Ambos paneles están colocados uno al lado del otro para facilitar al usuario una vista rápida y comparativa.

Desde un punto de vista funcional, los componentes que integran uno y otro panel tienen en realidad la misma finalidad:

- Un panel de texto, no editable, usado para presentar resultados.
- Un botón que activa la acción de procesar.
- Un botón para copiar los resultados, presentados en el panel de texto, al clipboard del sistema (evento "actionPerformed").

La diferencia se encuentra en la naturaleza de los ficheros a procesar y de los scripts de procesamiento utilizados para programar la acción de los botones "ProcessForAnaliticResults" y "ProcessForSimulationResults".

- El botón "ProcessForAnaliticResults" se encarga de procesar un fichero de texto llamado "analitic_data_input.txt" que contiene un registro con la siguiente información dispuesta en campos separados por espacios en blanco: número de fuentes (1), número de servidores (1), esperanza del tiempo entre llegadas, esperanza del tiempo de servicio, varianza del tiempo entre llegadas, varianza del tiempo de servicio.

El script AWK de procesamiento, llamado "analitics_results.awk" tiene programadas las tareas necesarias para procesar la información contenida en el

fichero anterior. El resultado es un fichero de texto con los parámetros de rendimiento teóricos o ideales del SdC tipo, que previamente seleccionó el usuario a través de la pestaña Input.

- En el caso del botón "ProcessForSimulationResults", el fichero a procesar es la traza "length_queue.tr", y el script de procesamiento utilizado se llama "simulation_results.awk". El resultado es un fichero de texto con los parámetros de rendimiento del SdC simulado por NS-2.

La zona inferior está dedicada fundamentalmente a la generación automática de gráficas, y su visualización.

Los componentes "jButton" situados en la parte izquierda, bajo la etiqueta Draw, tienen implementado un evento tipo "actionPerformed" cada uno. El objetivo es ejecutar, de forma transparente al usuario, el software de sistema para representación gráfica llamado "Gnuplot", y generar las gráficas a las que hace referencia cada uno de los botones.

El resultado gráfico se muestra a través de ventanas internas contenidas dentro del componente "jDesktopPane", el cual, actúa a modo de pantalla de escritorio interna para la aplicación QNSA. El control de las ventanas internas al escritorio QNSA (tamaño, redimensionado o movimiento) se realiza a través de la clase "ControlDesktop".

A continuación, se hace referencia a algunas consideraciones importantes respecto al proceso de ejecución de Gnuplot:

- Gnuplot necesita como argumento un script donde se especifican los comandos Gnuplot necesarios para construir la gráfica deseada. Dicho script es generado por QNSA en la carpeta temporal del sistema "/tmp/". La estructura de dicho script, en este caso, es como sigue:

```
#Ejemplo script Gnuplot generado en evento de botón"ArrivalsQueueB"
#Se establece el tipo de ventana - terminal con el que se debe ejecutar Gnuplot
```

```
set term png truecolor notransparent nocrop enhanced font arial 10 size 640,480
#Se dirige la salida gráfica a un fichero en formato "png"
set output '/tmp/gpResult5936703939422572185.png'
set title "ArrivalsQueue(+)"
set xlabel "Simulation Time sg"
set ylabel "Arrivals Bytes"
#Se indica a Gnuplot que debe aplicar autoescalado de los ejes xy set autoscale
plot  "/home/alumno/Escritorio/hola2/arrivals_queue.tr" using 2:3 with lines
exit
```

- Se vuelve a hacer uso de las clases "Runtime()" y "Process()" para comunicar a la máquina virtual Java con el software externo Gnuplot.

- Las gráficas se generan de manera automática en la carpeta temporal del sistema con formato ".png". Y se visualizan sobre paneles contenidos en ventanas internas al componente "jDesktopPane". Posteriormente, se verá cómo a través de la pestaña Output el usuario puede guardar las gráficas que desee.

- La clase "PaintPanelBG" se encarga de pintar el resultado gráfico sobre el panel contenido en las ventanas internas al "jDesktopPane". Esta clase, que extiende la clase "jPanel", básicamente sobrescribe el método "paintComponent()" de Java para poder establecer la imagen generada con Gnuplot como fondo del panel, y así hacer visible la gráfica al usuario.

El botón titulado "ResultsView" está localizado en esta zona para permitir que el usuario, sin salir de esta pestaña, tenga acceso visual a los parámetros que definieron su simulación u otros datos que considere importantes. En general, las acciones programadas en dicho botón posibilitan la apertura de ficheros de texto en ventanas internas al componente "jDesktopPane".

### 6.2.5. Interfaz de salida: Output

Como ya se indicó al comienzo del capítulo, la funcionalidad de esta interfaz de salida gira en torno a dos ideas fundamentales:

- La explotación del software de edición OpenOffice [38].

- La explotación del software de representación gráfica Gnuplot [39].

El paquete OpenOffice, disponible para diferentes plataformas como Linux, Mac o Windows, ofrece al programador la posibilidad de manipular su funcionamiento mediante un conjunto de librerías Java que conforman la API UNO [40]. Esta API se localiza, dentro del paquete OpenOffice, en la ruta: "...\OpenOffice.org 3\Basis\program\classes" (igualmente, es posible su descarga de manera independiente a través del Web).

La API se integra en el paquete QNSA, junto con el resto de librerías utilizadas, para su explotación a través de eventos programados en los paneles "My Notes" y "Select".

La razón de su integración en el proyecto QNSA tiene origen en la necesidad de ofrecer al usuario un entorno de edición gratuita y potente, a partir del cual poder generar y personalizar memorias de simulación de SdC G/G/1; además de, visualizar y tratar los ficheros de traza generados con la simulación.

El problema con respecto al uso de esta API se encuentra en el poco detalle ofrecido por la documentación existente, y en su relativa complejidad. En el anexo IX se pueden ver los programas desarrollados en QNSA para la explotación de OpenOffice.

QNSA, a través de los componentes gráficos dispuestos en el panel titulado "Graph", define una pequeña interfaz Java para la explotación del software de sistema Gnuplot. La tabla de este panel se proyecta como un contenedor donde el usuario puede almacenar diferentes casos de representación gráfica. El modelo implementado para definir el caso a representar, o registro de tabla, está construido de la siguiente manera: FPath -- ruta de archivo de datos o traza a representar (formato columnas separadas por espacio), Cx -- columna de datos del eje x, Cy -- columna de datos del eje y, Title -- título de la gráfica, xlabel -- título o etiqueta aplicable al eje x, ylabel -- título o etiqueta aplicable al eje y, [xmin,xmax] --intervalo de valores a representar

del eje x, [ymin, ymax] -- intervalo de valores a representar del eje y, LineType -- tipo de seña a utilizar para conformar la gráfica (líneas, puntos,etc.), Plot -- activa o desactiva el registro de la tabla para la representación gráfica con Gnuplot

Para poder añadir o eliminar registros de la tabla, el usuario hace uso de los eventos programados en los botones "AddRowToTable" y "DeleteRowFromTable", respectivamente. Cuando el usuario pulsa sobre el botón "AddRowToTable", aparece en pantalla el diálogo modal de la Figura 18 para que complete los campos indicados, y así, se cree un registro de la tabla.

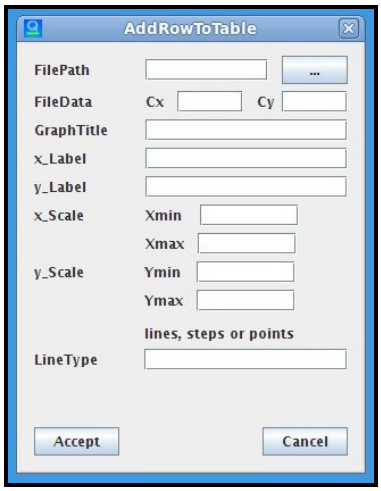


**Figura 18. QNSA/Diálogo "AddRowToTable"**

Una vez creado, el usuario puede modificar cada uno de los campos que definen el registro, directamente desde la tabla.

Al pulsar sobre el botón "DeleteRowFromTable", y si el usuario tiene seleccionado un registro, se borrará el registro de la tabla seleccionado.

Cuando el usuario quiere representar una gráfica, según características definidas en registro; lo primero que debe hacer es situarse sobre el registro deseado, y además, ticar el campo "plot" que habilita la creación interna del script Gnuplot correspondiente. Una vez hecho esto, los botones dispuestos bajo la tabla posibilitan las siguientes acciones:

- PlotYourFile: abre una instancia de interfaz Gnuplot (terminal "wxt") con la gráfica seleccionada. Con esta opción, como se puede apreciar

en la Figura 19, el usuario tiene activas opciones propias de la interfaz Gnuplot como el zoom, el autoescalado, uso de rejilla, etc.

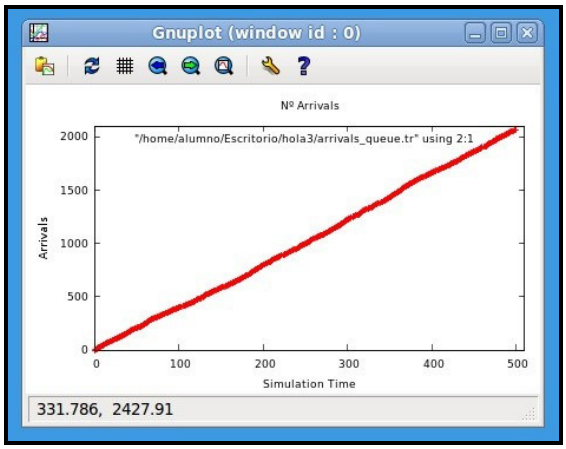


**Figura 19. QNSA/Terminal Gnuplot "wxt"**

El script Gnuplot generado, en este caso, tiene la siguiente estructura:

```
#Ejemplo
set term wxt
set title "Arrivals Queue"
set xlabel "Time"
set ylabel "Bytes"
set xrange [0:1000]
set yrange [0:25000]
plot "/home/alumno/Escritorio/hola3/arrivals_queue.tr" using 2:3 with lines
pause -1
```

- ViewChart: dibuja la gráfica seleccionada sobre el panel titulado "View Chart"; la ejecución de Gnuplot es transparente al usuario.

El script Gnuplot generado, en este caso, tiene la siguiente estructura:

```
#Ejemplo
#Se establece el tipo de ventana - terminal con el que se debe ejecutar Gnuplot
set term png truecolor notransparent nocrop enhanced font arial 10 size 640,480
#Se dirige la salida gráfica a un fichero en formato "png"
set output '/tmp/gpResult5936703939422572186.png'
set title "Arrivals Queue"
set xlabel "Time"
set ylabel "Bytes"
set xrange [0:1000]
set yrange [0:25000]
plot "/home/alumno/Escritorio/hola3/arrivals_queue.tr" using 2:3 with lines
exit
```

Las acción desencadenada en este botón es, en realidad, muy similar a lo visto en los botones de generación automática de gráficas de la pestaña "Process"; la diferencia se encuentra en que en este caso, el usuario tiene una función activa en la definición del script Gnuplot a través de la tabla diseñada para tal propósito dentro del panel "Graph".

- SavePlotAs: abre un diálogo modal con el usuario como el que se muestra en la Figura 20.

De esta manera, el usuario puede almacenar, en el directorio de trabajo QNSA, la gráfica que desee en formato: PNG, JPEG, Canvas, PS o SVG.

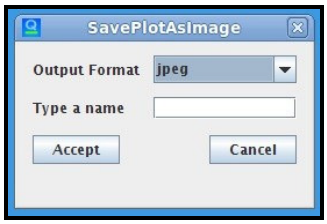


**Figura 20. QNSA/Diálogo "SavePlotAsImage"**

El script Gnuplot generado, en este caso, tiene la siguiente estructura:

```
#Las dos primeras líneas varían en función del tipo de formato seleccionado por el usuario
#Para format "jpeg"
set term jpeg nocrop font arial 12 size 640,480
set output '/home/alumno/Escritorio/hola3/Arrivals Queue.jpg'
#Para format "png"
set term png truecolor notransparent nocrop enhanced font arial 10 size 640,480
set output '/home/alumno/Escritorio/hola3/Arrivals Queue.png'
#Para format "Canvas"
set term canvas size 600,400 fsize 10 lw 1 standalone
set output '/home/alumno/Escritorio/hola3/Arrivals Queue.canvas'
#Para formato "ps"
set term postscript
set output '/home/alumno/Escritorio/hola3/Arrivals Queue.ps'
#Para format "svg"
set term svg size 600,480 fixed fname 'Arial' fsize 12 butt solid
set output '/home/alumno/Escritorio/hola3/Arrivals Queue.svg'
#Líneas comunes a todos los formatos, según edición de usuario, a través de registro de tabla
set title "Arrivals Queue"
set xlabel "Time"
set ylabel "Bytes"
set xrange [0:1000]
set yrange [0:25000]
plot "/home/alumno/Escritorio/hola3/arrivals_queue.tr" using 2:3 with lines
exit
```

## 6.3. Evaluación de la herramienta

Un paso importante a llevar a cabo cuando se realizan simulaciones consiste en verificar la fiabilidad de los resultados conseguidos. Con este propósito, y como ya se indicó en el punto 3.4, se utiliza la experiencia con otros simuladores encontrados en internet como muestra comparativa y valedora de los resultados obtenidos con QNSA.

En el punto siguiente, se exponen en tablas[44] algunos casos de SdC simulados y los valores de parámetros de rendimiento conseguidos.

### 6.3.1. Resultados de parámetros de rendimiento

| Caso1<br>QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 5<br>#Simulation.Time =1000<br># Output.name = out<br># Q.M.S. = M/M/1<br># Ro < 1 = yes<br># Arrival = Exponential<br># Lambda=26<br># Service = Exponential<br># Mu=32<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.8125 | 0.8087 | 0.812 | 0.8056 | 0.8084 |
| | *Lq* | 3.52083 | 3.3893 | x | 3.3747 | 3.4572 |
| | *Wq* | 0.135417 | 0.1296 | 0.136 | 0.1311 | 0.1326 |
| | *L* | 4.33333 | 4.1981 | 4.344 | 4.1804 | 4.2656 |
| | *W* | 0.166667 | 0.1679 | 0.167 | 0.1624 | 0.1636 |
| | *Ts* | x | 1000.0 | x | 600 | 10000 |
| | | | | | | |
| | | | | | | |

**Tabla 9. QNSA settings. Caso 1**

[44] La "x" indica que el dato no está disponible

| Caso2<br>QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 5<br># Simulation.Time = 1000<br># Output.name = out<br># Q.M.S. = M/G/1<br># Ro < 1 = yes<br># Arrival = Exponential<br># Lambda=0.8<br># Service = Uniform<br># Max=2<br># Min=0.1<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.84 | 0.8402 | 0.840 | 0.8503 | 0.8344 |
| | *Lq* | 2.80667 | 2.9509 | x | 3.2142 | 2.7828 |
| | *Wq* | 3.50833 | 3.7172 | 3.526 | 3.6971 | 3.5044 |
| | *L* | 3.64667 | 3.7910 | 3.663 | 4.1007 | 3.6172 |
| | *W* | 4.55833 | 4.9646 | 4.576 | 4.7386 | 4.5552 |
| | *Ts* | x | 1000.95 | x | 1000 | 10000 |
| | | | | | | |
| | | | | | | |

**Tabla 10. QNSA settings. Caso 2**

| Caso3<br>QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 4<br># Simulation.Time = 4000<br># Output.name = out<br># Q.M.S. = G/M/1<br># Ro < 1 = yes<br># Arrival = Uniform<br># Max=1.8<br># Min=1.1<br># Service = Exponential<br># Mu=0.8<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.862069 | 0.874969 | 0.862 | 0.8661 | 0.8661 |
| | *Lq* | 2.74629 | 2.54104 | x | 2.3272 | 2.3121 |
| | *Wq* | 3.98211 | 3.7015 | 3.566 | 3.2708 | 3.3471 |
| | *L* | 3.60835 | 3.416 | 3.321 | 3.1933 | 3.1782 |
| | *W* | 5.23211 | 5.15819 | 4.815 | 4.4906 | 4.6013 |
| | *Ts* | x | 4002.80 | x | 4000 | 4000 |
| | | | | | | |
| | | | | | | |

**Tabla 11. QNSA settings. Caso 3**

| Caso4<br>QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 3<br># Simulation.Time = 800<br># Output.name = out<br># Q.M.S. = G/G/1<br># Ro < 1 = yes<br># Arrival = Uniform<br># Max=1<br># Min=0.3<br># Service = Uniform<br># Max=0.6<br># Min=0.4<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.769231 | 0.771915 | 0.769 | 0.7558 | 0.7547 |
| | *Lq* | 0.141 | 0.095087 | x | 0.068 | 0.08397 |
| | *Wq* | 0.0916502 | 0.061382 | 0.059 | 0.0452 | 0.05526 |
| | *L* | 0.910231 | 0.867002 | 0.860 | 0.8241 | 0.8386 |
| | *W* | 0.59165 | 0.706927 | 0.559 | 0.5449 | 0.5519 |
| | *Ts* | x | 800.47 | x | 800 | 800 |
| | | | | | | |
| | | | | | | |

**Tabla 12. QNSA settings. Caso 4**

| Caso5 QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 3<br># Simulation.Time = 800<br># Output.name = out<br># Q.M.S. = M/D/1<br># Ro < 1 = yes<br># Arrival = Exponential<br># Lambda=0.2<br># Service = Constant<br># d=4<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.8 | 0.7712 | 0.8 | 0.8726 | 0.8381 |
| | *Lq* | 1.6 | 1.4193 | x | 1.5496 | 1.4947 |
| | *Wq* | 8 | 7.2675 | 8.125 | 7.1034 | 7.1344 |
| | *L* | 2.4 | 2.1905 | 2.424 | 2.4222 | 2.3328 |
| | *W* | 12 | 12.375 | 12.125 | 11.103 | 11.134 |
| | *Ts* | x | 803.9 | x | 600 | 800 |
| | | | | | | |
| | | | | | | |

**Tabla 13. QNSA settings. Caso 5**

| Caso6 QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 3<br># Simulation.Time = 800<br># Output.name = out<br># Q.M.S. = D/M/1<br># Ro < 1 = yes<br># Arrival = Constant<br># d=1.2<br># Service = Exponential<br># Mu=1<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.83333 | 0.8335 | 0.83 | 0.82022 | 0.8296 |
| | *Lq* | 2.08333 | 2.1315 | x | 1.3386 | 1.3330 |
| | *Wq* | 2.5 | 2.5609 | 2.152 | 1.6141 | 1.6006 |
| | *L* | 2.91667 | 2.9651 | 2.616 | 2.1588 | 2.1626 |
| | *W* | 3.5 | 3.6551 | 3.152 | 2.6002 | 2.5967 |
| | *Ts* | x | 801.36 | x | 500 | 800 |
| | | | | | | |
| | | | | | | |

**Tabla 14. QNSA settings. Caso 6**

| Caso7 QNSA settings | Parámetros | Resultados Teóricos | Resultados Simulados | | | |
|---|---|---|---|---|---|---|
| | | | NS2 | Q2.0 | WinQSB | QTSplus |
| # Seed = 3<br># Simulation.Time = 800<br># Output.name = out<br># Q.M.S. = M/G/1<br># Ro < 1 = yes<br># Arrival = Exponential<br># Lambda=28<br># Service = Normal<br># Mu=0.03<br># Sigma=0.01<br># Nº.Servers = 1<br># Capacity = 100000 | *Ro* | 0.84 | 0.8397 | x | 0.8366 | x |
| | *Lq* | 2.45 | 2.3522 | x | 2.4449 | x |
| | *Wq* | 0.0875 | 0.0842 | x | 0.0878 | x |
| | *L* | 3.29 | 3.1919 | x | 3.2815 | x |
| | *W* | 0.1175 | 0.1201 | x | 0.1178 | x |
| | *Ts* | x | 800.01 | x | 600 | x |
| | | | | | | |
| | | | | | | |

**Tabla 15. QNSA settings. Caso 7**

Como se puede apreciar de los resultados expuestos en las tablas 9, 10, 11, 12, 13, 14, y 15, el simulador NS-2 permite resolver los valores para parámetros de

rendimiento de SdC G/G/1 (sistema estable) por un lado, con proximidad a los valores teóricos y por otro, de manera bastante similar a los obtenidos con otros simuladores. No obstante, y haciendo mención a la bibliografía referenciada en el punto 4.3, la experiencia con el simulador y las distintas distribuciones que se utilizan evidencian, en muchos casos, desfases en la comparativa de resultados. El mayor problema en este sentido viene dado por la posible pérdida de reproducibilidad según utilización del RNG.

También, se ha de destacar que los casos presentados se corresponden con los valores obtenidos en una sola simulación. Las características del problema obligarían a realizar varias simulaciones y promediar el resultado de todas ellas a fin de ajustar con mayor veracidad la comparativa buscada.

En relación a la experiencia con los otros simuladores utilizados, también se dan casos en los que, dependiendo de las distribuciones consideradas, la semilla de aleatoriedad aplicada, el tiempo de simulación especificado o el número de iteraciones realizadas, las soluciones pueden alejarse mucho de los valores esperados.

## 6.4. Ejemplo de simulación – resultados

A continuación, se presenta una muestra general de los resultados que se pueden obtener de manera automática a través de la interfaz QNSA.

El usuario, mediante el botón “NewMyNotes” situado en la pestaña “Output”, puede fácilmente abrir un documento nuevo en blanco (OpenOffice) donde importar todos los resultados “.tcl”, “.tr”, “.txt” o gráficos que considere oportunos, y construir su propia memoria de simulación en tiempo real.

### 6.4.1. Script de simulación ".tcl"

Contiene un resumen inicial de los parámetros de entrada fijados por el usuario para configurar el SdC G/G/1 a simular, más el programa de simulación.

```
#Simu1.tcl
#*********************************************************************************
#*****************************DATA INPUT******************************************
#*********************************************************************************

# Seed = 909
# Simulation.Time = 500
# Output.name = out

# Q.M.S. = M/M/1
# Ro < 1 = yes

# Arrival = Exponential
# Lambda=3
# Service = Exponential
# Mu=5
# Nº.Servers = 1
# Capacity  = 10000.0

#*********************************************************************************
#*****************************END DATA INPUT**************************************
#*********************************************************************************

#This is Simu1

#Seeding the Simulation set rep 2
global defaultRNG;
$defaultRNG seed 909
set arrivalRNG [new RNG]
set serviceRNG [new RNG]

for {set r 1} {$r < $rep} {incr r} {
$arrivalRNG next-substream;
$serviceRNG next-substream;
}


set ns [new Simulator]
#Open files to write results
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$ns color fid Red
#Fixing some varibles
set PacketSize 20000
set duration 500
set execnam 0
set capacity 10000.0
```

```
#Our topology
set n(0) [$ns node]
set n(1) [$ns node]
$ns duplex-link $n(0) $n(1) $capacity 0.0001ms DropTail
set sink [new Agent/Null]
$ns attach-agent $n(1) $sink
#Generate Arrival type and Service type
set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr double(1.0/3)]
$InterArrivalTime use-rng $arrivalRNG
set medium_arrival [expr double(1.0/3)]
set variance_arrival [expr double(1.0/(3*3))]
set PSize [new RandomVariable/Exponential]
$PSize set avg_ [expr double($capacity/(8*5))]
$PSize use-rng $serviceRNG
set medium_service [expr double(1.0/5)]
set variance_service [expr double(1.0/(5*5))]
#define sources
…
…
$ns run
```

### 6.4.2. Resultado ".txt" de parámetros de rendimiento

Resultado de parámetros de rendimiento (teórico y simulado) para SdC G/G/1 obtenido en una simulación (ejemplo: Simu1.tcl - Seed 909 según punto 6.4.1).

```
*********************************************************************************
*****************************ANALITIC RESULTS*****************************
*********************************************************************************

*  Occupation Rate (Ro) : 0.6
*  Average Number of Customers in the Queue        (Lq) : 0.9
*  Average Time Customers Wait in the Queue        (Wq) : 0.3
*  Average Number of Customers in the System       (L) : 1.5
*  Average Time Customers Wait in the System       (W) : 0.5
*********************************************************************************
*****************************END OF REPORT*****************************
*********************************************************************************

*********************************************************************************
*****************************SIMULATION RESULTS *****************************
*********************************************************************************

*  Occupation Rate (Ro) : 0.636766
*  Average Number of Customers in the Queue        (Lq) : 1.05493
*  Average Number of Customers in the Queue        (Lq) : 1.0557      //con qm(1).tr
*  Average Time Customers Wait in the Queue        (Wq) : 0.338559
*  Average Number of Customers in the System       (L) : 1.6917
*  Average Time Customers Wait in the System       (W) : 0.542917
*  Number of Served Customers       : 1557
```

```
*  Number of Lost Customers          : 0
*  Max. Number of Customers in the Queue   : 11
*  Lost Customers Rate (%)  : 0
*  Simulation Time (sg)       : 499.688849
*****************************************************************************
*******************************END OF REPORT*******************************
*****************************************************************************
```

### 6.4.3. Resultado promedio de parámetros de rendimiento

En la Tabla 16 se puede observar el resultado promedio de 10 simulaciones ejecutadas con diferente semilla de aleatoriedad.

| Seed | Ro | Lq | Wq | L | W | P_servidos | P_perdidos | Max. Nº PKTs en Cola | Tiempo Simulación (sg) |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 0,62 | 1,1 | 0,35 | 1,72 | 0,55 | 1555 | 0 | 16 | 499,22 |
| 9 | 0,57 | 0,69 | 0,24 | 1,26 | 0,44 | 1443 | 0 | 11 | 499,56 |
| 15 | 0,63 | 1,03 | 0,33 | 1,66 | 0,54 | 1543 | 0 | 11 | 500 |
| 31 | 0,61 | 0,9 | 0,31 | 1,51 | 0,51 | 1473 | 0 | 11 | 499,79 |
| 73 | 0,57 | 0,67 | 0,23 | 1,24 | 0,43 | 1455 | 0 | 8 | 499,52 |
| 301 | 0,58 | 0,81 | 0,27 | 1,38 | 0,46 | 1502 | 0 | 11 | 499,41 |
| 505 | 0,63 | 1,07 | 0,35 | 1,69 | 0,55 | 1543 | 0 | 13 | 500,04 |
| 709 | 0,62 | 0,98 | 0,33 | 1,6 | 0,53 | 1494 | 0 | 12 | 500,08 |
| 803 | 0,58 | 0,94 | 0,32 | 1,52 | 0,53 | 1447 | 0 | 16 | 500,18 |
| 909 | 0,64 | 1,05 | 0,34 | 1,69 | 0,54 | 1557 | 0 | 11 | 499,69 |
| **Promedio** | **0,6** | **0,92** | **0,31** | **1,53** | **0,51** | **1501,2** | **0** | **12** | **499,75** |

**Tabla 16. QNSA/Resultado promedio de varias simulaciones**

En este caso, sólo es automática la recolección de resultados para varias simulaciones en un fichero ".txt" llamado "Simulation_Average". Es el usuario, quien a través del botón "Calc", de la pestaña "Output", puede hacer uso de una hoja de cálculo (OpenOffice) para abrir el fichero correspondiente (formateado de manera transparente al usuario), y ejecutar la función "Promedio" contenida en el paquete OpenOffice.

### 6.4.4. Resultados gráficos

En el anexo X se pueden ver algunos de los resultados gráficos obtenidos con QNSA (ejemplo "Simu1.tcl" según condiciones expuestas en punto 6.4.1.).

# 7. Virtualización

## 7.1. Introducción

En este último capítulo se presenta el estudio realizado sobre software de virtualización, como solución tecnológica adoptada, para el montaje y puesta en funcionamiento del laboratorio de simulación "sistemas de cola G/G/1".

Una de las necesidades fundamentales a cubrir para ejecutar la aplicación de escritorio QNSA viene impuesta por el entorno de trabajo del paquete NS-2. Como ya se ha visto, el simulador NS-2 corre exclusivamente bajo distribuciones de S.O. GNU\Linux. El objetivo es encontrar una herramienta robusta con la que satisfacer dicha necesidad, además de aportar facilidades en el montaje del laboratorio sin coste alguno.

Actualmente, la idea de optimizar recursos se ha impuesto como respuesta obligada a la mejora exponencial que los equipos de computación han experimentado. Los equipos integran elementos hardware cada vez más potentes (procesadores de última generación, Teras de disco duro, tarjetas gráficas con gran capacidad de memoria y procesamiento gráfico o tamaños de memoria RAM de 6GB y 8GB). La realidad es que, en muchas ocasiones, la capacidad del equipo es infrautilizada por parte del usuario; sin embargo, también es cierto que cada vez es más común utilizar programas de tipo profesional, aplicaciones o juegos que precisan dicha capacidad.

La industria tecnológica ha puesto de moda la virtualización para mejorar el rendimiento de estos equipos. De esta manera, las aplicaciones o servicios nativos de diferentes sistemas operativos funcionan al mismo tiempo y, lo que es más importante, sin ningún tipo de conflicto (ejemplo: "virtualización de servidores" en Data Centers). Igualmente, se puede hablar de beneficios colaterales como ahorro energético, aprovechamiento de espacio físico o reducción de contaminación medioambiental.

## 7.2. Tipos de virtualización. Justificación de uso dentro del proyecto

Existen diferentes tipos de virtualización o niveles de virtualización. Esta tecnología está en pleno desarrollo e implantación, y cada poco tiempo surgen nuevos conceptos relacionados con ella. A continuación, se detallan dos tipos en concreto, y de aplicación directa en el proyecto.

1. Virtualización de proceso o aplicación. Con Java.

2. Virtualización completa o nativa (sin apoyo hardware). Con Oracle VM VirtualBox.

El primero, ya referenciado en un capítulo anterior y cuya arquitectura básica se puede ver en la Figura 21, hace que el programa (QNSA) en cuestión se ejecute en una especie de entorno aislado (más seguro) o "sandbox" sobre el S.O. anfitrión.

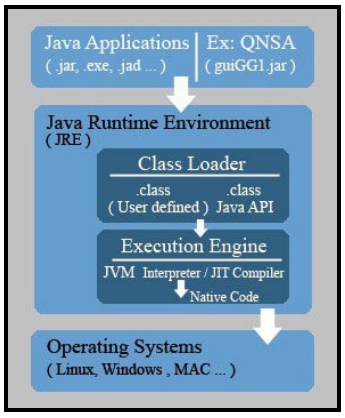


**Figura 21. Arquitectura Java Virtual Machine**

Este tipo de virtualización supone una abstracción del lenguaje de alto nivel o código fuente del programa con respecto a la plataforma de ejecución. El medio para conseguirlo se encuentra en la compilación de dicho código fuente como código "Bytecode" (el código fuente java o archivos ".java" es compilado para crear los archivos ".class" o Bytecode). De esta manera, el programa ya puede ser ejecutado en cualquier sistema o dispositivo (PC's, servidores o móviles) que contenga el JRE o máquina virtual adecuada para la plataforma. La máquina virtual de java (JVM), entendida como un proceso más dentro de la plataforma subyacente, interpreta dicho

código a modo de "procesador independiente", que posee su propio set de instrucciones, para su ejecución de forma nativa.

Gracias a la virtualización de proceso, se consigue que el programa QNSA sea portable a diferentes sistemas GNU\Linux (también, Windows y MAC) con idéntico resultado en su ejecución.

Existen más ejemplos con este tipo de filosofía como pueden ser la máquina virtual Parrot o el Common language Runtime de ".net" Framework.

El segundo, consiste en una virtualización de S.O. sin apoyo hardware. El software de virtualización, también llamado Virtual Machine Monitor (VMM) o Hypervisor, se encarga de ejecutar uno o varios SS.OO. "huésped" sobre otro que actúa como "anfitrión". En la Figura 22 se muestra un resumen de la arquitectura básica de este tipo de virtualización (con Oracle VM VirtualBox).

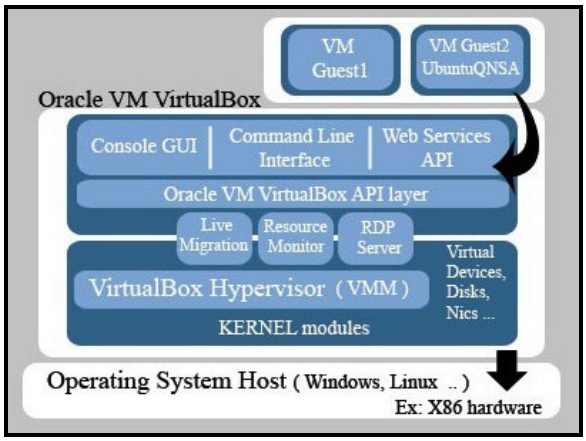


**Figura 22. Arquitectura Oracle VM VirtualBox**

En este caso, los diferentes SS.OO. que se deseen virtualizar no tienen que sufrir ningún tipo de cambio (aunque sí soportar la arquitectura hardware anfitrión), ya que el VMM administra los recursos (CPU, RAM, Disco, Red) de forma virtual o real según corresponda. El VMM se encargará de captar y transformar aquellas instrucciones del huésped que pudieran resultar.

Es importante notar que el S.O. anfitrión debe estar activo si se quiere ejecutar alguno de los SS.OO. huésped. Esto provoca una cierta competencia por los recursos

o un aprovechamiento compartido de los mismos; sin embargo, dada la capacidad actual de los equipos, la pérdida de eficiencia no tiene por qué ser demasiado significativa. En este sentido, se destaca que el software utilizado en el laboratorio para explotar este tipo de virtualización no está orientado a tener dos o más SS.OO. virtualizados funcionando constantemente, como si se tratasen de un servicio más.

En el proyecto se ha trabajado con la herramienta de virtualización Oracle VM VirtualBox, pero más adelante se verán otros ejemplos. Su aplicación directa en este proyecto tiene que ver con la posibilidad de levantar un S.O. completo GNU\Linux sobre una plataforma en la que el S.O. anfitrión es Windows (también otros). Con esta solución se alcanzan varias metas:

- No se tienen que realizar modificaciones hardware en los equipos del laboratorio de prácticas donde se quiera explotar este tipo de virtualización.

- No se tienen que realizar cambios en el software Windows de los equipos de laboratorio, puesto que la virtualización convive con éste sin problemas.

- El laboratorio de prácticas podrá seguir explotando los servicios y aplicaciones nativas de uno y otro sistema operativo sin ninguna clase de conflicto entre ellos.

- Se reducen a cero los errores derivados de utilizar otro tipo de políticas para hacer coexistir varios SS.OO. en una misma máquina física (ejemplo: particionado). En este caso, el cambio de un S.O. a otro no requiere reinicios; por lo que se reducen los tiempos de respuesta al cambiar de sistema operativo.

- Se aprovecha en espacio. No son necesarios nuevos equipos físicos.

- El alumno, con un pendrive o disco externo disponible, puede trasladar la máquina virtual con todo su contenido desde el laboratorio a su casa.

Es importante resaltar que esta acción se puede hacer con independencia del S.O. que tenga en su casa, y de forma gratuita.

- La virtualización permite al alumno crear un entorno de pruebas aislado para sus propios proyectos u otros sistemas operativos, y aplicaciones de testeo.

- El software de virtualización favorece la rápida distribución del paquete QNSA. Como se verá en el punto 7.4.1, este tipo de virtualización permite la clonación de las máquinas virtuales creadas de manera rápida y sencilla; sólo es necesario ejecutar una serie de comandos por consola.

## 7.3. Requisitos básicos

Las características principales que se buscan en el software de virtualización son las siguientes:

1. Efectividad. Se busca una máquina virtual que ofrezca una mínima pérdida de rendimiento; y de esta manera, utilice los recursos de la máquina anfitrión de la manera más eficiente posible.

2. Globalidad. Aunque se tenga claro el sistema operativo que se va a instalar en la máquina virtual, se busca aquel software que permita instalar máquinas virtuales con sistemas Windows o GNU/Linux. De esta manera, este software puede ser utilizado para otras aplicaciones.

3. Accesibilidad. Poder utilizar el máximo número de recursos de la máquina anfitrión (unidades de CD\DVD, impresoras, USB's, etc.).

4. Fiabilidad. No alterar de ningún modo el funcionamiento normal de los equipos donde esté instalado este software. Además, las máquinas virtuales no deben fallar en ningún momento.

5. Asequible. Es preferible un producto con licencia gratuita u open source.

6. Actualizable. La herramienta se tendrá que actualizar de forma regular, con el fin de evitar vulnerabilidades del software de virtualización que pudieran dañar el equipo donde esté instalado.

7. Viable. Que la implementación de la herramienta no sea complicada, y que se pueda utilizar en cualquier momento.

## 7.4. Herramientas de virtualización

La cantidad de herramientas orientadas a la virtualización, englobando dentro de ésta todas sus variantes, es elevada. Se trata de un producto que indudablemente está cosechando grandes beneficios, y con un futuro más que prometedor. Algunos nombres propios son : Qemu ( Linux, Windows, Mac OSX ... ), OpenVZ (Linux ), VirtualPC ( Windows ), KVM ( Linux ), Microsoft Virtual Server, Parallels Desktop / WorkStation ( Mac OSX ),etc.

En este apartado se referencian tres de los más utilizados. Dos de ellos, Oracle VM VirtualBox y VMWare Player (ver anexo XI) han sido explotados durante el proyecto en su versión de software para equipos de escritorio; siendo Oracle VM VirtualBox, la elección final. Xen (ver anexo XI) sobresale como solución en máquinas servidor.

### 7.4.1. Oracle VM VirtualBox

Desarrollado en primer lugar por Inotek GMBH, fue adquirida con posterioridad por Sun Microsystems, y finalmente por Oracle. Existen dos versiones del programa, una freeware (PUEL[45]) y otra open source (GNU General Public License versión 2) [41].

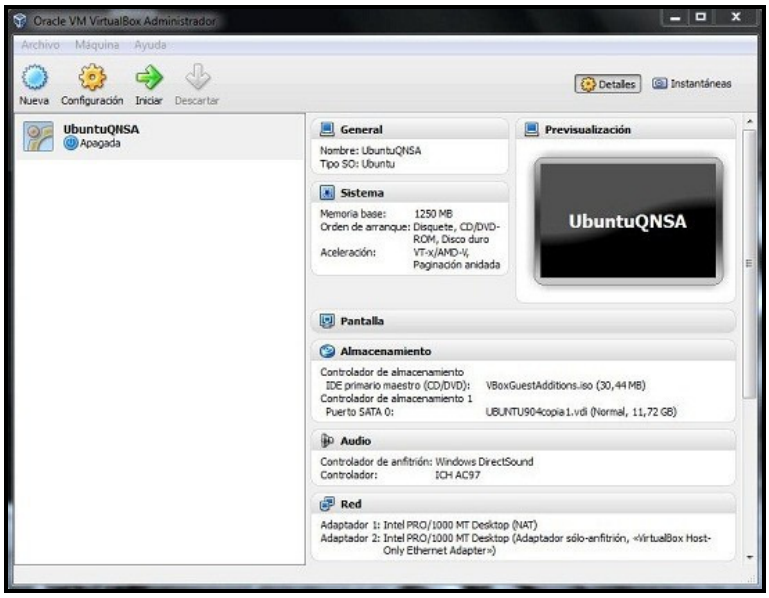


**Figura 23. Oracle VM VirtualBox/Captura de programa**

El tipo de virtualización que ofrece, como ya se ha visto, se denomina nativa o completa. Actualmente, Oracle VM VirtualBox también puede complementar y mejorar dicha virtualización mediante la explotación de arquitecturas con tecnologías como Intel VT-x y AMD-V.

Su ventaja principal es que sirve para cualquier sistema operativo, Windows, GNU/Linux, Mac OS y Solaris. Su instalación es muy sencilla; y para cada máquina virtual se puede elegir entre una configuración predeterminada o detallar por parte del usuario la cantidad de memoria RAM, disco duro, etc. que necesita. Incluso una vez

[45] PUEL es el acrónimo de Personal Use and Evaluation License

creada la máquina, podemos variar estos valores fácilmente. Las arquitecturas que soporta son x86 y AMD64/Intel64.

El software "VirtualBox Guest Additions", con que se puede acompañar a esta máquina virtual, mejora el rendimiento del sistema virtualizado. Su contenido son drivers y aplicaciones.

Por otro lado, gracias al gran uso del programa, hay una comunidad de usuarios muy amplia y por ello, numerosas páginas de soporte. Además, cada poco tiempo se liberan nuevas revisiones y mejoras que facilitan el mantenimiento del software (aprox. 6 meses).

Para este proyecto, como se puede ver en la Figura 23, se utiliza una versión de la herramienta bajo condiciones de licencia PUEL; por tanto, su uso a nivel personal y en ambientes académicos como la universidad, es libre y gratuito.

La herramienta se puede descargar directamente desde la página web del desarrollador. Se debe seleccionar el paquete destinado a equipos "anfitrión" con arquitectura x86 y S.O. Windows. Su instalación es sencilla, y no ofrece ningún tipo de problema o complicación.

El siguiente paso es crear la máquina virtual del S.O. seleccionado como "huésped". En este punto, se apuesta por virtualizar la distribución Ubuntu (basada en Debian) de GNU\Linux [42]. Las razones fundamentales son:

- Ubuntu goza de gran aceptación entre los usuarios de todo el mundo, muy por delante de otras distribuciones (o distros). Ha conseguido que se olvide la idea de que GNU\Linux es un S.O. sólo para personas con conocimiento en el área de la informática. De hecho, actualmente muchos equipos vienen con el S.O. ya instalado; lo cual era impensable hace relativamente poco tiempo.

- Es un S.O. de código abierto y gratuito con una amplia comunidad de desarrolladores (profesional y amateur) a sus espaldas. Esto facilita la liberación de nuevas versiones cada poco tiempo, acceso a

documentación formativa, solución de bugs y actualizaciones. Por tanto, el soporte y mantenimiento del S.O. está asegurado.

- Los requerimientos físicos de Ubuntu no son grandes (RAM 384MB, Disco 10GB, etc.). Esto facilita su virtualización en equipos no demasiado potentes o antiguos.

- El simulador NS-2 se instala fácilmente, y corre sin problemas en esta distro.

La imagen .iso de Ubuntu se puede descargar desde cualquier "mirror[46]" accesible desde su página Web.

Ya en la herramienta VirtualBox, se pulsa sobre la opción "nueva" para virtualizar el sistema "huésped". La herramienta nos guía durante todo el proceso permitiendo seleccionar diferentes opciones de configuración como: nombre que se asignará a la máquina virtual, S.O. "huésped" que se utilizará (Linux), procesador, memoria RAM, etc. El resultado es un fichero del tipo ".vdi" (accesible en el directorio VirtualBox) que realmente es nuestra máquina virtual o disco virtual.

Un paso importante la primera vez que se ejecuta la máquina virtual con Ubuntu consiste en añadirle el software "VirtualBox Guest Additions". Su instalación es automática a través de la barra de menú de VirtualBox\Dispositivos, o bien, mediante descarga de imagen ".iso" correspondiente e instalación manual. Con ello se consigue un mejor rendimiento general de la máquina virtual Ubuntu como:

- Integración del ratón. Es decir, se puede pasar del sistema "huésped" al "anfitrión" sin necesidad de tener que capturar o liberar el ratón mediante combinación de teclas.

- Soporte gráfico. Se mejora la resolución del sistema virtualizado.

[46] Mirror es un servidor de descargas

- Sincronización horaria.

- Carpetas compartidas. Se pueden crear carpetas para compartir archivos entre la máquina virtual y el sistema real. Para configurar dicha facilidad es necesario:

    - Seleccionar a través de consola VirtualBox la carpeta del sistema anfitrión a compartir. Ejemplo: XshareUbuntu.

    - En el sistema Ubuntu huésped se crea un ejecutable de shell para el montaje y desmontaje automático de la carpeta a compartir en la ruta de sistema seleccionada. Ejemplo:

```
#!/bin/sh
sudo mount -t vboxsf  XshareUbuntu
/home/alumno/Escritorio/SHARE
```

- Portapapeles compartido entre sistema "huésped" y "anfitrión".

- Integración de usb's, lectoras de CD\DVD, etc.

La distribución de la máquina virtual a otros equipos se realiza mediante "clonación" de la misma. La clonación consiste en realizar una copia o snapshot del estado actual de la máquina virtual. Por tanto, una vez que el S.O. "huésped" ha sido configurado conforme a las necesidades planteadas para ejecución de QNSA (actualización de sistema, instalación de software NS-2, Gnuplot y OpenOffice, etc.), se procede de la siguiente manera:

- En el sistema anfitrión (Windows), se realiza una copia del fichero [nombredelamáquinavirtual].vdi y se le asigna un nombre diferente.

- Se abre una instancia de la herramienta de consola. Y se ejecutan las siguientes órdenes:

```
>cd C:\Program Files\Oracle\VirtualBox
>VBoxManage.exe internalcommands sethduuid"[ruta al archivo copiado].vdi"
```

Con esto se consigue asignar un UUID (identificador de disco virtual) diferente a la máquina virtual copiada.

- Por último, en los equipos destino donde se desee ejecutar la máquina virtual; se especificará durante la instalación con VitualBox que el disco a utilizar será de tipo virtual, el ".vdi" copiado. El resto de parámetros se configurarán según disponibilidad y necesidad.

La línea de comandos de VirtualBox o CLI ofrece otros muchos servicios para la gestión de las máquinas virtuales creadas. Estas opciones, entre las que se encuentra el redimensionamiento del disco virtual con posterioridad a su creación, pueden ser exploradas escribiendo "VboxManage.exe" en la ventana de comandos.

Para la conexión de red, VirtualBox ofrece hasta 8 posibles adaptadores virtuales a elegir por cada máquina virtual creada (AMD PCNet PCI II, AMD PCNet FAST III, Intel PRO/1000 MT Desktop, Intel PRO/1000 T Server, Intel PRO/1000 MT Server y Virtio-net). De igual modo, el usuario debe seleccionar el modo de trabajo del adaptador de red entre 5 opciones (Not attached, Network Address Translation (NAT), Bridged, Host only e Internal mode). Por ejemplo, en el caso particular de la máquina virtual UbuntuQNSA (creada para el proyecto), sobre sistema anfitrión Windows (tarjeta de red Realtek), se ha configurado un adaptador Intel PRO/1000 MT Desktop en modo NAT para salida a internet.

# 8. Conclusiones y trabajos futuros

El documento que se presenta explica la implementación de una aplicación de escritorio Java encaminada a explotar la capacidad del motor NS-2 como simulador de sistemas de cola. En concreto, la aplicación se centra en la configuración, procesado y representación de modelos de cola tipo G/G/1. Además, facilita la simulación de otros escenarios que puedan ser de interés al usuario, así como la visualización y representación gráfica de los resultados traza obtenidos.

Las soluciones ofrecidas por la aplicación han sido verificadas mediante cálculo analítico y/o también, a través de su comparación con las soluciones obtenidas en otros simuladores de sistemas de cola encontrados en internet.

Su desarrollo ha supuesto la investigación en diferentes áreas como son: la teoría de colas, el simulador NS-2, la interfaz gráfica de usuario, el lenguaje Java, el lenguaje tcl, el procesamiento de datos mediante AWK, el software de edición OpenOffice, la representación gráfica mediante software Gnuplot o la vitualización. Se trata, por tanto, de un proyecto con un marcado carácter integrador y práctico que en lo particular ha significado un aprendizaje continuo, pese a las dificultades encontradas.

Por otro lado, el espacio de trabajo virtual y accesible configurado en el proyecto se considera suficiente para el desarrollo y testeo de nuevos proyectos sobre SdC, y otros muchos aspectos de relevancia dentro del ámbito de las telecomunicaciones contenidos en el paquete de simulación NS-2 (motivación y origen del proyecto presentado).

El marco de trabajo encontrado con este proyecto representa en general una buena fuente de futuros desarrollos. A continuación, se apuntan algunas ideas:

- Trabajo sobre el simulador NS-2. Estudio de otros modelos de sistemas de cola (prioridades, número de servidores, tipo de tráfico transmitido, número de fuentes, protocolos de transporte...). Mejora del grado de confianza de las simulaciones; sustitución del RNG actual

(MRG32k3a) por el RNG de Makoto Matsumoto (Mersenne Twister) [43].

- Mejoras sobre la aplicación diseñada. Considerar mayor número de parámetros de entrada y casos de simulación. Implementación de una interfaz de entrada, basada en panel de dibujo, complementaria o sustitutiva de la existente. Estudiar y mejorar el procesamiento de trazas.

- Dotar a la aplicación diseñada de nuevas funcionalidades que permitan explotar otros aspectos de simulación contenidos en el paquete NS-2.

- Trabajo sobre el simulador NS-3. Adaptación de la interfaz o aplicación diseñada para su trabajo sobre el motor de simulación NS-3.

- Estudio de los diferentes elementos software explotados en el proyecto (simulador NS-2 o NS-3, awk, Gnuplot, OpenOffice) para posible desarrollo de servlet[47]).

[47] Servlet es un programa Java que se carga y ejecuta en el contenedor web de un servidor de aplicaciones

# 9. Anexos

## 9.1. Anexo I: software - simuladores de sistema de cola

### 1-AQUAS[48]

Es una herramienta de simulación orientada a la resolución analítica y por simulación de problemas de colas. Los modelos de sistemas de cola que resuelve son:

| Resolución Analítica | Resolución por Simulación |
|---|---|
| *M/M/1* | *G/G/1* |
| *M/M/s* | *G/G/s* |
| *M/M/1/K* | *G/G/1/K* |
| *M/M/s/K* | *G/G/s/K* |
| *M/M/1/∞/H* | *G/G/1/∞/H* |
| *M/M/s/∞/H* | *G/G/s/∞/H* |
| *M/M/s/∞/H con Y repuestos* | *G/G/s/∞/H* con *Y* repuestos |
| *M/M/∞* | *G/G/∞* |
| Redes de Jackson Abiertas o Cerradas | |

**Tabla 17. AQUAS/Sistemas de cola admitidos**

- Las distribuciones de probabilidad de entrada y salida que soporta son: exponencial, uniforme, determinista, gamma, beta, lognormal, normal y weibull.

- Se trata de una aplicación realizada en MATLAB programa e interfaz gráfica). Su ejecución requiere tener instalado en el ordenador una versión del software de pago MATLAB 5.0 o superior y la librería Statistics Toolbox.

[48] AQUAS es el acrónimo de Application for solving Queuing problems Analytically and using Simulation

- Se distribuye de manera gratuita para los sistemas operativos Windows y Linux [44].

- Su instalación es sencilla. Sólo hay que indicar a MATLAB mediante un "set path", la ruta donde están los ficheros que conforman la aplicación AQUAS. Ejemplo de capturas del simulador en la Figura 24

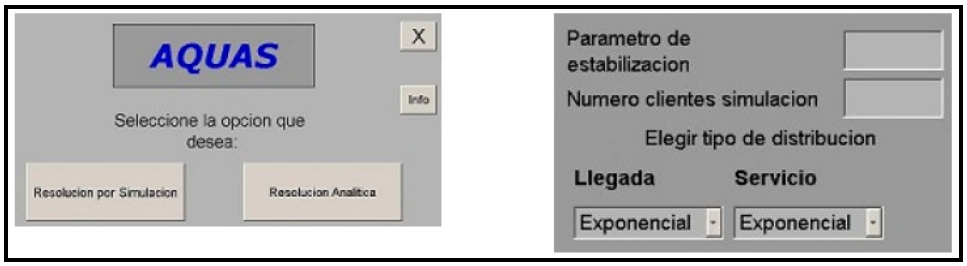


**Figura 24. AQUAS/Captura de programa**

### 2-Queue 2.0

Sus características principales son:

- Permite resolver sistemas de cola simple tipo G/G/S. Ofrece resultados analíticos y por simulación. Además, representa gráficamente los parámetros de rendimiento: número de clientes en el sistema y en cola, tiempo de espera en el sistema y en cola, y tasa de ocupación de los servidores.

- Las distribuciones de probabilidad que soporta en entrada y salida son: exponencial, Erlang, hiperexponencial, coxian, determinista, uniforme y general.

- El simulador está programado en lenguaje Java (interfaz applet). Para su ejecución es necesario tener instalado la máquina virtual de Java en el navegador web.

- Se trata de un software no distribuible. Se accede a través de la dirección web:

http://www.win.tue.nl/cow/Q2/

A continuación, la Figura 25 muestra una captura del simulador:

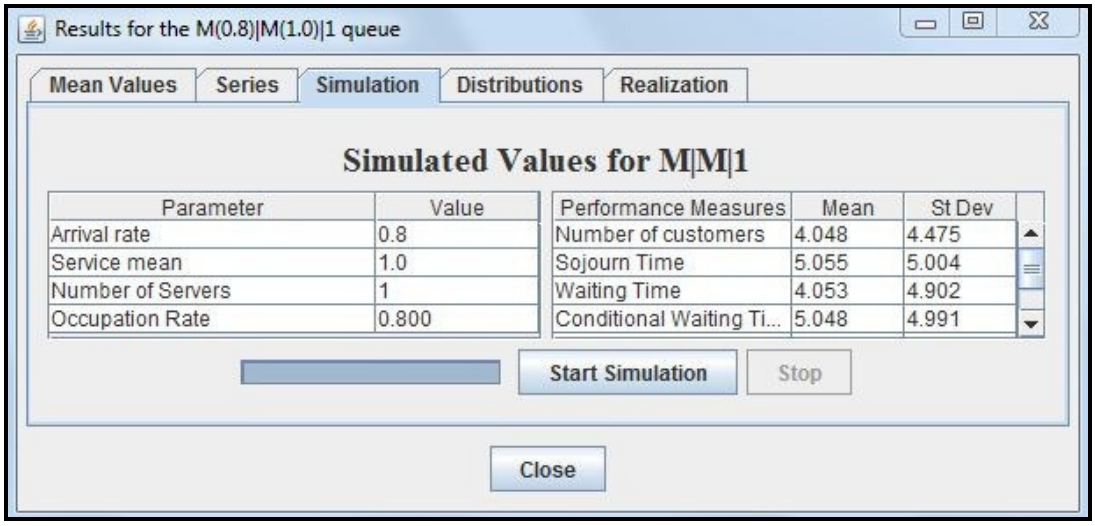


**Figura 25. Queue2.0/Captura de programa**

### 3-WinQSB 2.0

Sus características principales son:

- Es una aplicación formada por 19 módulos o subprogramas con los que se pueden resolver y automatizar problemas de cálculo lineal, investigación de operaciones, evaluación de proyectos, etc. Entre los subprogramas hay tres específicamente relacionados con la teoría de colas:
  - Resolución de procesos de Markov.
  - Análisis de sistemas de cola (resolución analítica y por simulación).
  - Simulación de sistemas de cola (resolución por simulación).

La aplicación se distribuye con un manual explicativo a base de ejemplos por cada módulo. Esto facilita el manejo de los módulos [45].

- El módulo "análisis de SdC" ofrece en su interfaz de entrada 2 formatos de resolución. Uno, resuelve analíticamente y por simulación SdC simple tipo M/M/S; con pocas opciones de configuración para facilitar su rápida ejecución.

El otro, resuelve analíticamente y por simulación SdC más complejos y generales. En este caso, las distribuciones de probabilidad soportadas son: general, weibull, uniforme, triangular, poisson, pareto, normal, lognormal, laplace, geométrica, hipergeométrica, gamma, exponencial, erlang, discreta, constante, binomial y geométrica.

En ambos casos, la solución analítica ofrece un buen número de parámetros de rendimiento del SdC. Además, en el caso de no existir fórmula cerrada para la resolución del SdC definido por el usuario, la aplicación ofrece la posibilidad de aproximar de manera automática por un modelo G/G/S conocido o a través de simulación Montecarlo.

Se destaca en la opción de simulación y en ambos formatos de entrada, la facilidad de poder elegir la semilla de aleatoriedad (por defecto, introducida por el usuario o reloj del sistema) y la disciplina de cola (LIFO, FIFO o Random).

Este módulo también ofrece la posibilidad de obtener los resultados en función del barrido de alguno de los parámetros característicos en un intervalo determinado y además, representarlo gráficamente.

A continuación, en la Figura 26 se muestra una captura de este módulo:

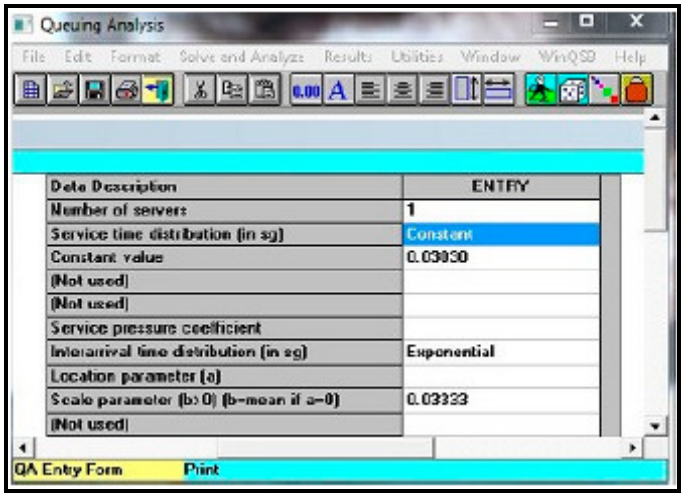


**Figura 26. WinQSB 2.0/Captura de módulo "Queuing Analysis"**

- El módulo "simulación de SdC", inicialmente más complejo en la introducción de parámetros de entrada, permite considerar 4 elementos dentro del ambiente simulado: tasa de llegada de clientes (C), cola o línea de espera (Q), servidores (S) y el recolector de basura (indica la posibilidad de que un cliente abandone el sistema sin ser procesado).

Las distribuciones soportadas en este caso son: beta, binomial, constante, discreta, erlang, exponencial, gamma, hipergeométrica, laplace, normal, pareto, poisson, triangular, uniforme y weibull.

Con el sistema definido, y terminada la simulación, la aplicación permite ver los resultados de rendimiento desde el punto de vista de los clientes, servidores o las colas.

Este módulo también ofrece la posibilidad de simular gráficamente el sistema; lo que facilita la comprensión del mismo.

A continuación, en la Figura 27 se muestra una captura de este módulo:

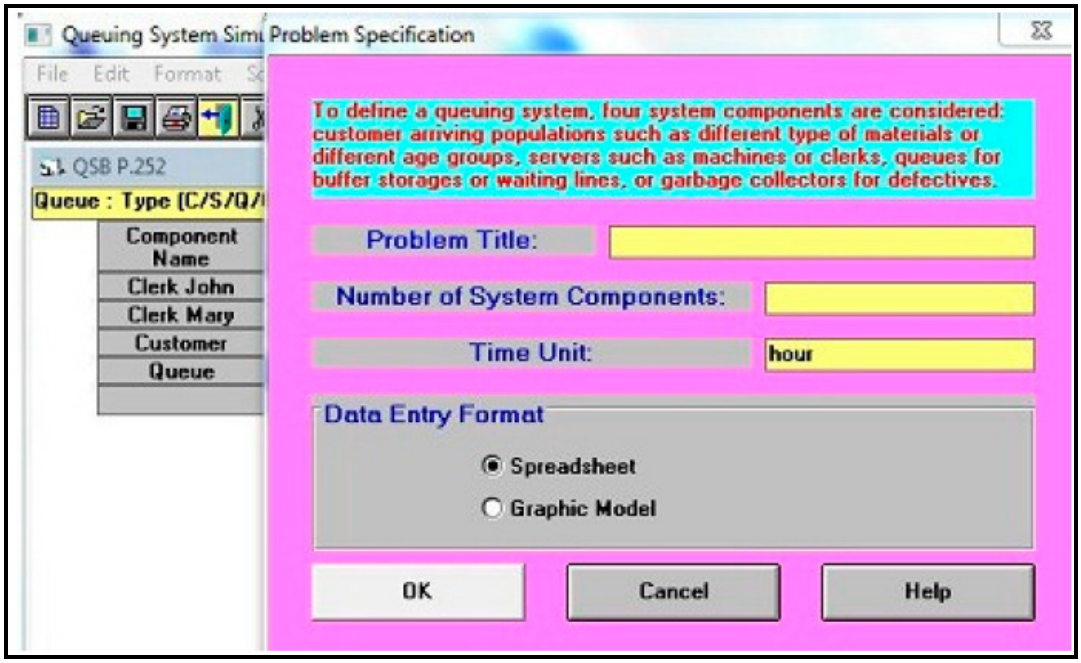


**Figura 27. WinQSB 2.0/Captura de módulo "Queuing System Simulation"**

- El simulador está muy extendido dentro de la comunidad docente universitaria, lo que avala sus virtudes a la hora de transmitir conocimientos que, como en el caso de la teoría de colas, pueden resultar demasiado engorrosos si sólo son vistos desde un punto de vista teórico.

- La aplicación WinQSB 2.0 se distribuye de manera gratuita para sistema operativo Windows.

### 4-QTSplus 3.0

Las características principales son:

- QTSplus permite la resolución analítica y mediante simulación de diversos problemas relacionados con la teoría de colas como cálculo de Erlang, procesos de nacimiento y muerte, resolución estadística, modelos de cola con uno o varios servidores, colas con prioridad y redes simples.

- Se trata de una aplicación realizada a base de macros de Excel. Existe una versión para su ejecución sobre hoja de cálculo de OpenOffice (QTSplus4Calc).

- Se muestran, a continuación, algunas de las macros implementadas para resolver modelos de sistema de cola simple:

| Un Servidor | Varios Servidores |
|---|---|
| M/M/1 | M/M/c |
| M/M/1/K | M/M/c/K |
| M/M/1 with Balking | M/G/c/c |
| M/G/1 | M/M/infinity |
| M/G/1 Sensivity Analysis | M/G/infinity |
| M/D/1 | M/Ek/c |
| M/Ek/1 | M/D/c |
| M/H/1 | D/M/c |
| D/M/1 | Ej/M/c |
| Ej/M/1 | H/M/c |
| H/M/1 | G/M/c |
| G/M/1 | **G/G/c Simulator** |
| Ej/Ek/1 | **Single Node Simulator** |
| G/G/1 Approximation model | |
| **G/G/1 Simulator** | |
| | |

**Tabla 18. QTSplus 3.0/Sistemas de cola admitidos**

- Las macros de simulación permiten el uso de las siguientes distribuciones de entrada y salida: exponencial, gamma, lognormal, triangular, uniforme y weibull.

- El software se distribuye de manera gratuita como complemento del libro de pago “Fundamentals of Queueing Theory” [46] [47].

- La Figura 28 muestra una captura del simulador:

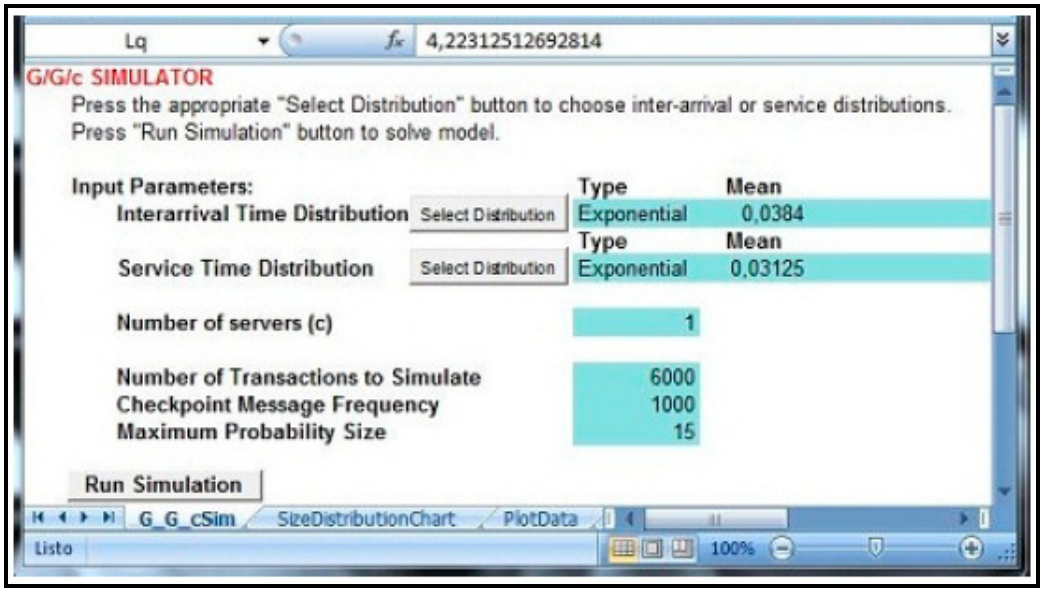


**Figura 28. QTSplus 3.0/Captura de programa**

## 9.2. Anexo II: instalación de NS 2.34 en Ubuntu 9.04

Los pasos que se han realizado han sido los siguientes:

1. Descargar el paquete "ns-allinone-2.34.tar.gz" desde: http://sourceforge.net

2. Descomprimir el paquete en el directorio de trabajo predeterminado (ejemplo: /home/alumno).

```
$cd /home/alumno
$tar -xzvf ns-allinone-2.34.tar
```

3. Instalar/actualizar el sistema con paquetes y librerías necesarios para evitar dependencias.

```
$sudo apt-get install build-essential autoconf automake libxmu-dev libx11-dev
```

4. Posicionarse en el directorio de archivos de ns-2.34, y ejecutar el instalador.

```
$cd ns-allinone-2.34
```

```
$./install
```

5. Validar la instalación.

```
$cd ns-2.34
$./validate
```

6. Editar el archivo ".bashrc" con las entradas siguientes:

```
$gedit .bashrc     # Se abre el archivo

#Para NS
# LD_LIBRARY_PATH OTCL_LIB=/home/alumno/ns-allinone-2.34/otcl-1.13
NS-2_LIB=/home/alumno/ns-allinone-2.34/lib
X11_LIB=/usr/X11R6/lib USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS-2_LIB
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$X11_LIB:$USR_LOCAL_LIB

# Para TCL_LIBRARY
TCL_LIB=/home/alumno/ns-allinone-2.34/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# Para PATH
XGRAPH=/home/alumno/ns-allinone-2.34/bin:/home/alumno/ns-allinone-
2.34/tcl8.4.18/unix:/home/alumno/ns-allinone-2.34/tk8.4.18/unix:/home/alumno/ns-
allinone-2.34/xgraph-12.1/
NS=/home/alumno/ns-allinone-2.34/ns-2.34/
NAM=/home/alumno/ns-allinone-2.34/nam-1.14/
export PATH=$PATH:$XGRAPH:$NS:$NAM

$source .bashrc   #Para actualizar los cambios realizados en el archivo ".bashrc"
```

7. Crear links al bin del sistema para que los programas instalados con el paquete NS-2.34 sean ejecutables desde cualquier parte del sistema.

```
$ sudo ln -s /home/alumno/ns-allinone-2.34/ns-2.34/ns /usr/bin/ns
$ sudo ln -s /home/alumno/ns-allinone-2.34/nam-1.14/nam /usr/bin/nam
$ sudo ln -s /home/alumno/ns-allinone-2.34/xgraph-12.1/xgraph /usr/bin/xgraph
```

**Nota:** Según la distro Ubuntu instalada, se puede generar otra dependencia que impida la correcta ejecución del simulador. NS-2.34 trabaja con la versión de compilador gcc-4.3, y algunas distribuciones de Ubuntu como por ejemplo la 10.4 usan otra versión de compilador por defecto. Si éste es el caso, es necesario descargar e instalar la versión de compilador gcc-4.3, modificar el Makefile.in de otcl de NS-2.34, y volver a ejecutar el ./install de NS-2.34. A continuación, se muestra el proceso:

```
$sudo apt-get install gcc-4.3
```

Se edita el fichero Makefile.in contenido en la ruta: /ns-allinone-2.34/otcl-1.13/Makefile.in, y se sustituye el valor de CC=@CC@ por CC=gcc-4.3

```
$cd ns-allinone-2.34
$make clean
$make
$./install          # Se reinstala NS-2.34
```

## 9.3. Anexo III: programación tcl básica

### 1-Variables

La creación de una variable y la asignación de valor se realiza mediante el comando "set".

El símbolo "$" precediendo a una variable se usa para que el intérprete de tcl tome el valor de dicha variable.

El comando "expr" se usa para realizar operaciones aritméticas.

Tcl considera la cadena contenida entre corchetes como un comando que el intérprete sustituirá por su resultado.

En la primera línea del ejemplo se asigna el valor 5 a la variable "c". En la segunda línea, se define el valor de la variable b como el resultado de la expresión suma de dos factores: el contenido de la variable "c" y el valor 8. Ejemplo:

```
set c 5
set b [expr $c+8]
```

### 2-Comandos de Control

| if | for | While |
|---|---|---|
| Ej:<br>set a 5<br>set i 3<br>if {$i==0} {<br>puts $a<br>} elseif { $i>2 } {<br>puts $i<br>} | Ej:<br>set i 0<br>for {$i==0} {$i<3} {incr i} {<br>puts "hola mundo"<br>} | Ej:<br>set i 3<br>while {$i<=5} {<br>puts $i<br>incr i<br>} |
| **Operadores de Comparación** | > , < , >= ; <= , == , != , \|\| , && | |

**Tabla 19. Comandos de control en lenguaje tcl**

### 3-Ficheros y Comandos de I/O

El comando "open" permite abrir el fichero "file0.txt".

El resultado, es un identificador de fichero que se almacena en la variable "$file1".

El comando "gets" permite leer líneas del fichero.

El comando "puts" permite escribir líneas en el fichero.

El comando "close" cierra el fichero.

Ejemplo:

```
set file1 [open "file0.txt" X ]  #X Puede tomar el valor r (leer), w (escribir) ó a
#(añadir)
…
…
puts $file1 "\nhola mundo\n"
…
close $file1
```

### 4-Procesos

Se crea el proceso denominado "record" con varios argumentos, y se devuelve el valor que finalmente contenga la variable " var3". Las variables utilizadas dentro

de un proceso tienen carácter local; el comando "global", se usa para hacer uso de variables globales dentro del proceso. Por ejemplo:

```
proc record { arg1 arg2 … } {
global var1 var2
...
return $var3
}
```

### 5-Llamadas a programa externo

El comando "exec" llama al programa externo de representación gráfica "Xgraph" para dibujar el contenido del fichero "Delay-end-to-end-tcp.txt". Los comandos "-nl -m -x tiempo (sg) -y delay(ms)" son argumentos requeridos por el programa "xgraph". Por ejemplo:

```
exec xgraph Delay-end-to-end-tcp.txt -nl -m -x tiempo(sg) -y delay(ms)
```

### 6-Comentarios

En tcl, los comentarios se identifican con líneas precedidas por el comando "#".

## 9.4. Anexo IV: script AWK - analitic_results.awk

```
function myfact(number)
{
  if (number == 0)
    fact = 1

  else
    fact = number

  for (x = 1; x < number; x++)
    fact *=x

  return fact
```

```
}


function mypow(num1,num2)
{
power=exp(log(num1)*num2)
return power
}


#****************************************
#*****Algoritmo para Resultados Analiticos****
#****************************************

{

RR=(1/$3)/(1/$4)
poisson=0
suma_poisson=0


for(p=0; p<=($2-1); p=p+1)
{
poisson=(exp(-RR) * mypow(RR,p)) / myfact(p)
suma_poisson=poisson + suma_poisson
#printf"salgo del for\n"
}
Cov_a=($5) * mypow(1/$3,2) Cov_s=($6) * mypow(1/$4,2)
p1=(Cov_s + Cov_a) / ( 2 * ($2 * (1/$4) - (1/$3)))
p2=p1 * mypow(RR,$2) / ( myfact($2) * ( 1 - (1/$3) / ($2* (1/$4))))
p3=p2 / ((exp(RR) * suma_poisson) + mypow(RR,$2) / (myfact($2) * (1 - (1/$3) / ($2*
(1/$4)))))
Wq_2=p3
Ro_2=(1./$3)/((1/$4)*$2) Lq_2=(1./$3)* Wq_2
L_2=Lq_2 + ( (1/$3) / (1/$4 )) W_2=L_2 / (1/$3)

#***************
#*****FIN******
#***************
print " ***********************************************" >"analitic_results.txt"
print " **************ANALITIC RESULTS************" >"analitic_results.txt"
print " ***********************************************\n" > "analitic_results.txt"
print "  *  Occupation Rate          (Ro) :" Ro_2                    >"analitic_results.txt"
print " *  Average Number of Customers in the Queue( Lq) :" Lq_2 >"analitic_results.txt"
print " *  Average Time Customers Wait in the Queue Wq) :" Wq_2 >"analitic_results.txt"
print " *  Average Number of Customers in the System(L) :" L_2 >"analitic_results.txt"
print " *Average Time Customers Wait in the System (W) :"W_2>"analitic_results.txt"
print " \n"
print " ***********************************************" >"analitic_results.txt"
print " ************END OF REPORT**************" >"analitic_results.txt"
print " ***********************************************" >"analitic_results.txt"
print "\n" >"analitic_results.txt"

}
```

## 9.5. Anexo V: script AWK - simulation_results.awk

```
#************************************************
#******* Resultados por Simulación*****************
#************************************************
#Script adaptado de proyecto "Modelamiento de sistemas de cola G/G/1 en NS". José
#Luis Sanzana Riffo [3]
#************************************************

BEGIN {
  i=j=z=k=0
  m=1
  acumula1=acumula2=acumula3=0
  resta1=resta2=resta3=resta33=resta333=0
  Ro=Lq=Wq=W=L=T_simulacion=P_servidos=P_exitqueue=P_outsystem=0
  num_paquetes=t_simulacion=perdidos=paq_ruteo=0

  maxPK=0
  queue_size =0
  lines_count =0

}

{

   if ($1==0 && m<=2) {
       b[m,1]  = $2
       b[m,2]  = $3
       m++
    }
    else {
        if ($1==0){
          perdidos++
          print perdidos > "grafico_paq_perdidos.tr"
          print perdidos " " $2 > "grafico_num_paq_perdidos.tr"
        }
        else {
          while(getline < "numero_servidores.data"){
              paq_ruteo = $1
          }

          if($3 > maxPK)
            maxPK = $3;

            b[m,1]  = $2
            b[m,2]  = $3
            acumula1 = acumula1 + ((b[m,1] - b[m-1,1]) * b[m-1,2])
            print $2 " " $3 > "grafico_length_queue.tr"
            m++

        }
  }

  while(getline < "arrivals_queue.tr"){
      a[i,1] = $1
```

```
        a[i,2] = $2
        total_tiempo_arrivo = a[i,2]
        num_paquetes = i
        i++

        print $2 " " $3 > "grafico_arrivals_queue.tr"
        print $2 " " num_paquetes+1 > "grafico_num_paq_enviados.tr"

    }


    while(getline < "entry_service.tr"){
        a[j,3] = $2
        j++
        P_exitqueue=j
       print $2 " " j > "grafico_entry_service.tr"

        bytes += $3;
        if(inicio == 0)
        inicio=$2;
        dif_tiempo=$2 - inicio;
        if(dif_tiempo > 0)
        printf("%f %f\n", $2, (8*bytes)/dif_tiempo) > "Throughput.tr"
    }

    while(getline < "out_services.tr"){
        a[k,4] = $2
        total_tiempo_out = a[k,4]
        k++
        P_outsystem = k
        print $2 " " k  > "grafico_num_paq_servidos.tr"
    }

    while (k-1>=z){
     # resta 1 representa la resta del tiempo cuando el paquete sale del servicio, menos el
#tiempo  cuando entra al servicio
      resta1 = a[z,4] - a[z,3]  #r -
      printf("%d %f %f\n", z, resta1, a[z,3]) > "Time_in_Service.tr"
      acumula2 = acumula2 + resta1
      # resta 2 representa la resta del tiempo cuando el paquete entra al servicio, menos el
#tiempo cuando llega al sistema
      resta2 = a[z,3] - a[z,2]  #- +
      printf("%d %f %f\n", z, resta2, a[z,3]) > "Q_Delay_End_To_End.tr"
      # resta 3 representa la resta del tiempo cuando sale del sistema, menos cuando entra del
#sistema
      resta3 = a[z,4] - a[z,2]  #r +
      printf("%d %f %f\n", z, a[z,2], resta3) > "S_Delay.tr"
      acumula3 = acumula3 + resta2
      z++
    }

    while(getline < "Q_Delay_End_To_End.tr"){

          if ( pk == 0 )  {
                              td = $2;
                              pk = $1;
                    }
```

```
                    else if ( $1 == (pk+1) ) {
            printf("%d %f %f\n", $1, $2-td, $3) > "Jitter_End_To_End.tr"
                        td = $2;
                        pk = $1;
                    }
                    else if ( $1 > (pk+1) ) {
                        td = $2;
                        pk = $1;
                    }
  }

  while(getline < "qm(1).tr")
  {
  mean_size = $5;
  queue_size = queue_size + mean_size;
  lines_count = lines_count + 1;

  }

  }

  END {

   if (total_tiempo_arrivo > total_tiempo_out) T_simulacion = total_tiempo_arrivo
   else T_simulacion = total_tiempo_out

   average = queue_size/lines_count
   P_servidos   = num_paquetes + 1
   P_perdidos   = perdidos
   P_totales    = P_servidos + P_perdidos
   P_P_perdidos = (P_perdidos*100)/P_totales
   Ro_original  = (acumula2/T_simulacion)
   Ro           = ((acumula2/T_simulacion)/num_servidores2)
   Lq_original  = (acumula1/T_simulacion)
   Lq           = (acumula1/T_simulacion/num_servidores2)
   L            = (Lq + Ro_original)
   Wq_temp      = (acumula1/P_servidos)
   Wq           = (Wq_temp)
   #w_temp = (Wq + acumula2/P_servidos)
   #W = w_temp
    if (acumula3==0) {
      W_temp   = (T_simulacion/P_servidos)
      W    = (W_temp)
   }
   else {
      W_temp=((T_simulacion/P_servidos)*num_servidores2)+
((acumula3/P_servidos)/num_servidores2)
      W   = (W_temp)
   }


   print "  ********************************************** " > "simulation_results.txt"
   print "*****SIMULATION RESULTS ************** " > "simulation_results.txt"
   print "  ************************************\n" > "simulation_results.txt"

   print " *   Occupation Rate              (Ro) : " Ro              > "simulation_results.txt"
   print  "*  Average  Number  of  Customers  in  the  Queue  (Lq)  :  "  Lq        >
```

```
"simulation_results.txt"
    print "  *   Average Number of Customers in the Queue    (Lq con qm(1),tr) : " average
> "simulation_results.txt"
    print "  *   Average Time Customers Wait in the Queue    (Wq) : " Wq                 >
"simulation_results.txt"
    print "  *   Average Number of Customers in the System     (L) : " L                >
"simulation_results.txt"
    print "  *   Average Time Customers Wait in the System     (W) : " W                >
"simulation_results.txt"
    print "  *    Number of Arrivals-Customers                  : " P_servidos          >
"simulation_results.txt"
    print "  *    Number of Customers Exit Queue                : " P_exitqueue         >
"simulation_results.txt"
    print "  *    Number of Served Customers                   : " P_outsystem          >
"simulation_results.txt"
    print "  *    Number of Lost Customers                     : " P_perdidos           >
"simulation_results.txt"
    print "  *   Max. Number of Customers in the Queue          : " maxPK               >
"simulation_results.txt"
    print "  *    Lost Customers Rate (%)                      : " P_P_perdidos        >
"simulation_results.txt"
    print "  *    Simulation Time (sg)                         : " T_simulacion        >
"simulation_results.txt"
    print "\n"

    print "  ************************************  " > "simulation_results.txt"
    print "  **************END OF REPORT*********  " > "simulation_results.txt"
    print "  ****************************************\n" > "simulation_results.txt"



   while(getline < "Sreplication.data")
   {
   if($1==1)
   printf("%d %f %f %f %f %f %f %f %f %f\n", $2, Ro, Lq, Wq, L, W, P_servidos,
P_perdidos, maxPK, T_simulacion) >> "Simulation_Average.tr"
   else
   printf("%s %s %s %s %s %s %s %s %s %s\n", "Seed", "Ro", "Lq", "Wq", "L", "W",
"P_servidos", "P_perdidos", "maxPK", "T_simulacion") > "Simulation_Average.tr"
   }
  }
```

## 9.6. Anexo VI: software - interfaces Java en torno a NS-2

### 1-NS Workbench y NSG2

Objetivo: en ambos casos, generar de manera automática scripts ".tcl" mediante la técnica "drag and drop".

Funcionalidad: en ambos casos, el usuario utiliza un panel de dibujo como

lienzo en el que disponer y configurar los distintos objetos gráficos disponibles a fin de crear el escenario de simulación (NSG2 permite configurar escenarios de red cableado y wireless, y NS Workbench, sólo cableado). Una vez terminado, el usuario debe pulsar un botón para convertir dicho escenario en un script ".tcl" simulable por NS-2.

### Captura del programa NSG2

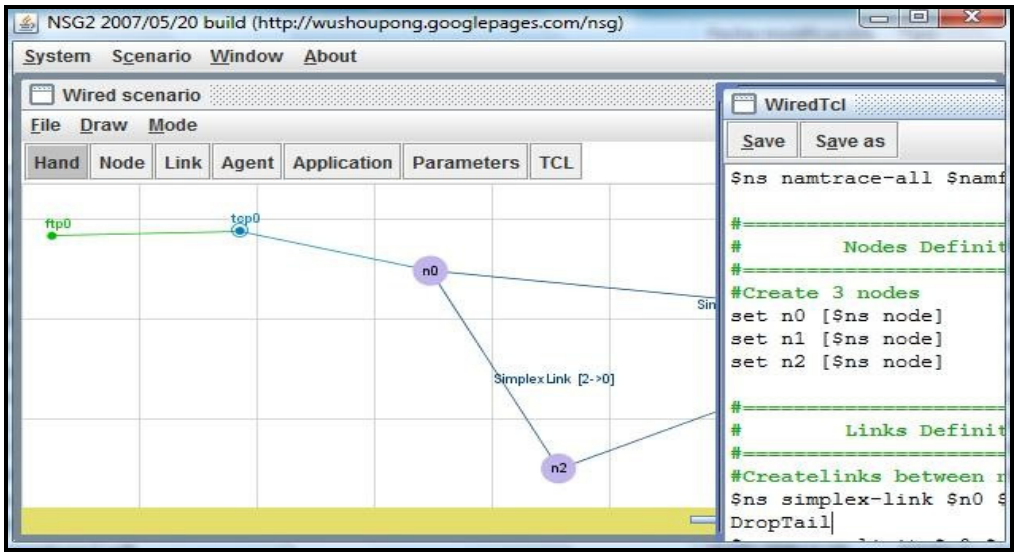


**Figura 29. Captura de interfaz Java "NSG2"**

### Captura del programa NS Workbench

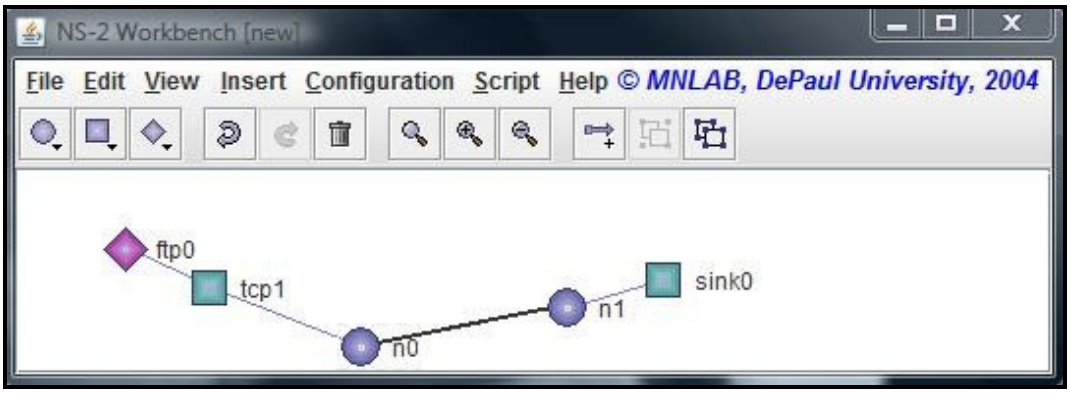


**Figura 30. Captura de interfaz Java "NS-2 Workbench"**

Se trata, como se puede observar, de dos aplicaciones de escritorio Java (o interfaces gráficas de usuario de entrada) muy parecidas. No obstante, se destaca una

mejor experiencia con NSG2, cuya compilación es más reciente (año 2007).

En ambos casos, en cuanto a sistemas de cola se refiere, se resalta la posibilidad de configurar el monitoreo por defecto en NS-2 de la cola del enlace.

### 2-SimBT

El objetivo y funcionalidad de esta interfaz de entrada son similares a los de los dos programas anteriores. La diferencia se encuentra en el tipo de escenario configurable, que en este caso se trata de redes bluetooth.

### Captura del programa SimBT

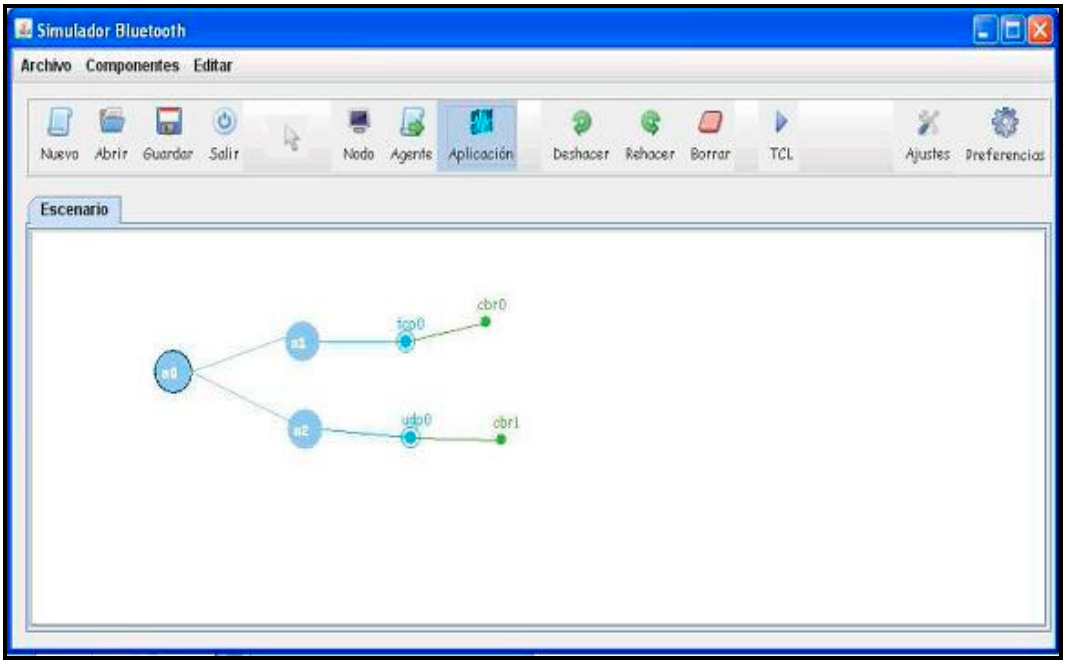


**Figura 31. Captura de interfaz Java "SimBT"**

### 3-Nans

Objetivo: analizar, mediante representación gráfica, ficheros de traza NS-2 standard de redes cableadas y wireless. Los parámetros representables son: "sequence number vs. time", "one way delay vs. time", "RTT vs time", "Throughput(one way) vs. time" y "Throughput(RTT) vs. time".

Funcionalidad: primero, se selecciona el fichero de traza ".tr" a estudiar.

Segundo, se indica a Nans cuál es el par de nodos entre los que hacer el análisis. Tercero, se selecciona el parámetro a representar. Finalmente, se pulsa el botón "draw graph" que dibuja la gráfica.

**Captura del programa Nans**

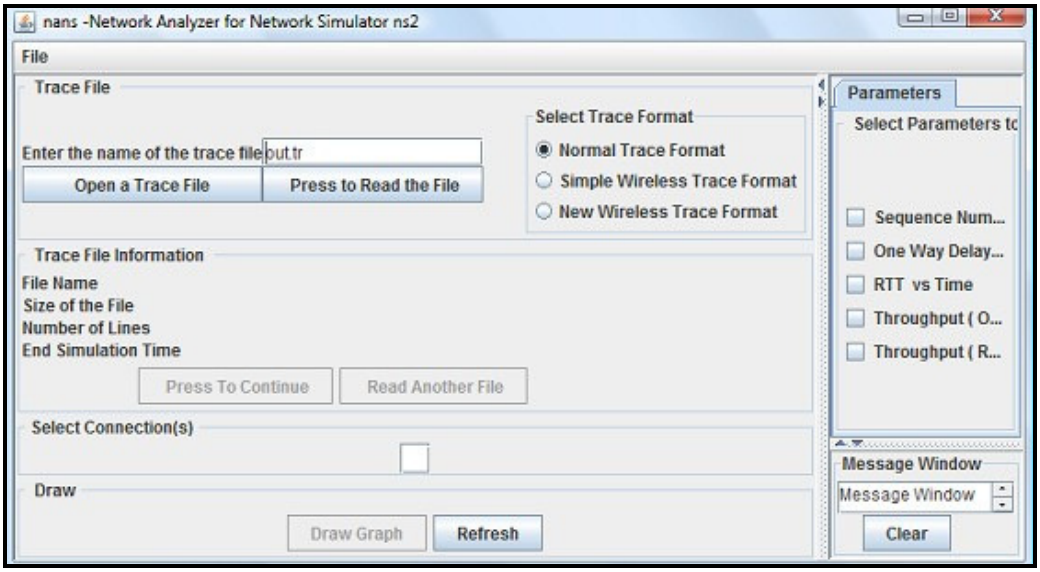


**Figura 32. Captura de interfaz Java "Nans"**

Se trata de una interfaz gráfica de salida en la que sobresale la explotación de su propio paquete Java de representación gráfica desarrollado en el departamento de la universidad de California en Berkeley. La aplicación requiere de compilación previa por parte del usuario.

## 9.7. Anexo VII: instalación de JRE

Los pasos realizados para este proyecto han sido:

1. Descargar el paquete "jre-6u<version>-linux-i586.bin", para sistemas Linux, desde la página web de Java http://www.java.com/. Donde <version> hace referencia a la última versión de Java disponible en la web.

2. Abrir un terminal de consola, y logarse como usuario root.

$sudo

3. Cambiar los permisos del archivo Java descargado para poder ejecutarlo.

   $chmod a+x jre-6u<version>-linux-i586.bin

4. Ejecutar el paquete Java.

   $./jre-6u<version>-linux-i586.bin

5. Aceptar el contrato de licencia, y comprobar directorio de instalación (donde se han descomprimido los archivos Java (ejemplo: /usr/local/jre1.6.0_16).

6. Indicar al sistema que se ha descargado una nueva versión de Java y que es ésta versión, y no otra antigua, la que se quiere utilizar por defecto.

$update-alternatives --install /usr/bin/java java /usr/local/jre1.6.0_16/bin/java 1

7. Seleccionar la versión de Java que se acaba de instalar.

   $sudo update-alternatives --config java

8. Verificar que la versión de Java por defecto es la deseada.

   $java -version

## 9.8. Anexo VIII: IDE Netbeans

A continuación, se presenta gráfica y funcionalmente el programa Netbeans como el entorno de programación utilizado para el desarrollo de la interfaz gráfica de usuario objeto de este proyecto.

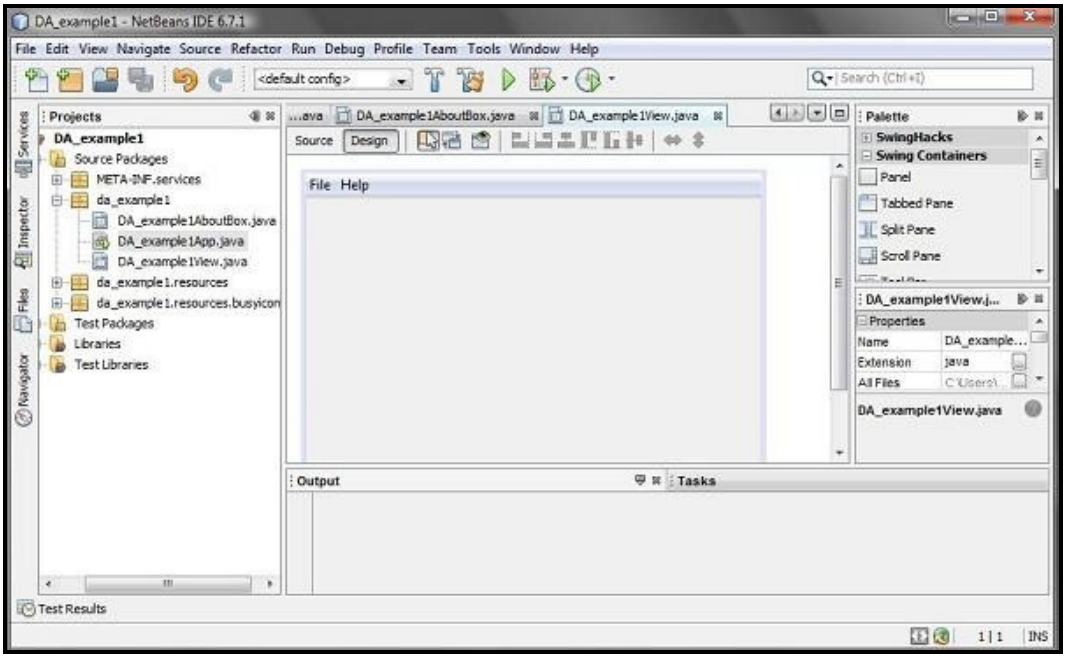


**Figura 33. Captura de programa "IDE NetBeans"**

Como se puede apreciar a simple vista en la Figura 33, Netbeans posee una interfaz muy intuitiva que facilita el descubrimiento de sus múltiples funcionalidades:

- A través de la opción del menú "File" se selecciona el tipo de proyecto a desarrollar y por tanto, la API de Java con la que se va a trabajar. También, a través de esta opción se pueden fijar de manera automática y rápida, alguna de las propiedades con las que se caracterizará finalmente a la aplicación de cara a su ejecución.

- El panel "Projects" muestra la organización de los archivos (código fuente en forma de archivos ".java" y ".form", recursos y librerías necesarias para la compilación) que conforman el proyecto. Esta vista facilita el trabajo del programador a la hora de estructurar el conjunto de paquetes fuente que conforman el trabajo.

- La incorporación de nuevas librerías al proyecto, necesarias para la compilación de alguna de las clases utilizadas en el proyecto, se puede realizar haciendo uso de la herramienta de menú "library manager" accesible desde "tools/libraries".

- El panel central permite alternar con rapidez entre el editor de código fuente y el editor gráfico denominado Matisse. Se destaca el automatismo del programa para identificar errores de programación mediante código de colores, y su capacidad para anticipar al programador estructuras o comandos útiles propios del lenguaje Java. Igualmente, en lo que al objeto gráfico se refiere, el programa facilita al programador herramientas con las que caracterizarlo inicialmente.

- La ventana "Output", situada en la parte inferior, permite ver los resultados de la depuración, compilación y ejecución del proyecto en desarrollo. El programa facilita en este punto la localización y direccionamiento hacia posibles errores en el código fuente.

- Las opciones "navigator" e "inspector" son tremendamente útiles a medida que el proyecto va creciendo, ya que permiten navegar a través del código fuente con fluidez o posicionarnos en alguno de los elementos gráficos que integran el proyecto.

- Las opciones de menú "Run" y "debug" permiten explotar las funcionalidades de compilación, ejecución y depuración contenidas dentro del paquete JDK.

## 9.9. Anexo IX: QNSA - código Java para explotación de OpenOffice

- Para abrir un fichero de traza ".tr" con swriter.exe:

```
// El fichero de configuración inicial QNSA establece la ruta al directorio de ejecutables
//"program"de OpenOffice (swriter.exe, scalc.exe, sbase.exe ...)

String L_office_path ="/usr/lib/OpenOffice/program";

//Se arranca OpenOffice a través de API UNO, estableciendo un socket con el centro de ejecución

XComponentContext  xContext = BootstrapSocketConnector.bootstrap(L_office_path);
```

```
//Se recogen el conjunto de servicios ofrecidos por OpenOffice

XMultiComponentFactory xMCF = xContext.getServiceManager();

//Se obtiene la interfaz de escritorio de OpenOffice

Object oRawDesktop = xMCF.createInstanceWithContext("com.sun.star.frame.Desktop",
xContext);

            XDesktop oDesktop = (XDesktop)
UnoRuntime.queryInterface(XDesktop.class,oRawDesktop);
XComponentLoader xCompLoader = (XComponentLoader)
          UnoRuntime.queryInterface(com.sun.star.frame.XComponentLoader.class, oDesktop);

//Se aplican los filtros necesarios según operación a realizar

PropertyValue[] mypv = new PropertyValue[3];
mypv[0] = new PropertyValue();
mypv[0].Name = new String("FilterName");
mypv[0].Value = new String("Text");

mypv[1] = new PropertyValue();
mypv[1].Name ="Hidden";
mypv[1].Value = new Boolean(false);
mypv[2] = new PropertyValue();
mypv[2].Name ="FilterOptions";
mypv[2].Value ="ASCII,LF";

// URL del fichero de traza ".tr" seleccionado por el usuario

   String sUrl="file:///"+this.yop.getText();

   //Se abre el fichero de traza según propiedades aplicadas

XComponent xComp = xCompLoader.loadComponentFromURL(sUrl.replace("\\","/"),"_blank",
   0, mypv);
```

- Para abrir fichero de traza ".tr" con scalc.exe:

Lo único que cambia con respecto al código anterior es el conjunto de propiedades aplicadas como filtro para abrir el documento.

```
…
//variablecolumnas es un string que almacena el número de columnas que contiene el fichero de
//traza
PropertyValue[] mypv = new PropertyValue[3];
mypv[0] = new PropertyValue();
mypv[0].Name = new String("FilterName");
mypv[0].Value = new String("Text - txt - csv (StarCalc)");
mypv[1] = new PropertyValue();
mypv[1].Name ="Hidden";
mypv[1].Value = new Boolean(false);
mypv[2] = new PropertyValue();
mypv[2].Name ="FilterOptions";
```

```
mypv[2].Value =“32,34,0,1,”+ variablecolumnas;
...
```

- Para exportar el fichero de traza “.tr” a PDF:

```
//Se realiza la conexión con OpenOffice igual que en los casos anteriores
...
PropertyValue[] mypv = new PropertyValue[3];
mypv[0] = new PropertyValue();
mypv[0].Name =“Hidden”;
mypv[0].Value = new Boolean(true);
mypv[1] = new PropertyValue();
mypv[1].Name = new String(“FilterName”);
mypv[1].Value = new String(“Text”);
mypv[2] = new PropertyValue();
mypv[2].Name =“FilterOptions”;
mypv[2].Value =“ASCII,LF”;

// URL del fichero de traza”.tr”seleccionado por el usuario

String sUrl=“file:///”+this.yop.getText();

XComponent xComp =
xCompLoader.loadComponentFromURL(sUrl.replace(“\\“,“/“),“_blank”,0,mypv);

//Se crea URL para almacenar el documento”.pdf”
String storeUrl=sUrl.concat(“.pdf”);

XModel xModel = (XModel)UnoRuntime.queryInterface(XModel.class,xComp);
XController xController = xModel.getCurrentController();
XFrame xFrame = xController.getFrame();

XMultiServiceFactory xMultiServiceManager =
(XMultiServiceFactory)UnoRuntime.queryInterface(XMultiServiceFactory.class,
xContext.getServiceManager());

Object objDispatchHelper=
xMultiServiceManager.createInstance(“com.sun.star.frame.DispatchHelper”);

XDispatchHelper xDispatchHelper =
(XDispatchHelper)UnoRuntime.queryInterface(XDispatchHelper.class, objDispatchHelper);
XDispatchProvider xDispatchProvider =
(XDispatchProvider)UnoRuntime.queryInterface(XDispatchProvider.class, xFrame);

PropertyValue[] mypvs = new PropertyValue[3];
mypvs[1] = new PropertyValue();
mypvs[1].Name =“DisplayPDFDocumentTitle”;
mypvs[1].Value = new Boolean(false);
mypvs[0] = new PropertyValue();
 mypvs[0].Name =“FilterName”;
mypvs[0].Value =“writer_pdf_Export”;
mypvs[2] = new PropertyValue();
mypvs[2].Name =“Overwrite”;
mypvs[2].Value = new Boolean(true);
```

```
XStorable xstorable = (XStorable) UnoRuntime.queryInterface(XStorable.class,xComp);
xstorable.storeToURL(storeUrl.replace("\\","/"),mypvs);

//IMPORTANTE LO QUE SIGUE
//HAY QUE descomentar esta ultima linea si quieres que te salte lo de exportar a pdf
//xDispatchHelper.executeDispatch(xDispatchProvider,".uno:ExportDirectToPDF","", 0,
mypvs);
//Fin del comentario de la ventana que salta para exportar a pdf
oDesktop1.terminate();
```

- Para abrir un nuevo documento de OpenOffice ".odt" en blanco:

```
/ El fichero de configuración inicial QNSA establece la ruta al directorio de ejecutables
//"program"de OpenOffice (swriter.exe, scalc.exe, sbase.exe ...)

String L_office_path ="/usr/lib/OpenOffice/program";

//Se arranca OpenOffice a través de API UNO, estableciendo un socket con el centro de
ejecución

XComponentContext xContext = BootstrapSocketConnector.bootstrap(L_office_path);

//Se recogen el conjunto de servicios ofrecidos por OpenOffice

XMultiComponentFactory xMCF = xContext.getServiceManager();

//Se obtiene la interfaz de escritorio de OpenOffice

Object oRawDesktop = xMCF.createInstanceWithContext("com.sun.star.frame.Desktop",
xContext);

XDesktop oDesktop = (XDesktop) UnoRuntime.queryInterface(XDesktop.class,oRawDesktop);
XComponentLoader xCompLoader = (XComponentLoader)
        UnoRuntime.queryInterface(com.sun.star.frame.XComponentLoader.class, oDesktop);
// Se crea el documento en blanco
XComponent document =
xCompLoader.loadComponentFromURL("private:factory/swriter","_blank", 0, new
PropertyValue[0]);
```

## 9.10. Anexo X: QNSA - resultados gráficos (ejemplo)

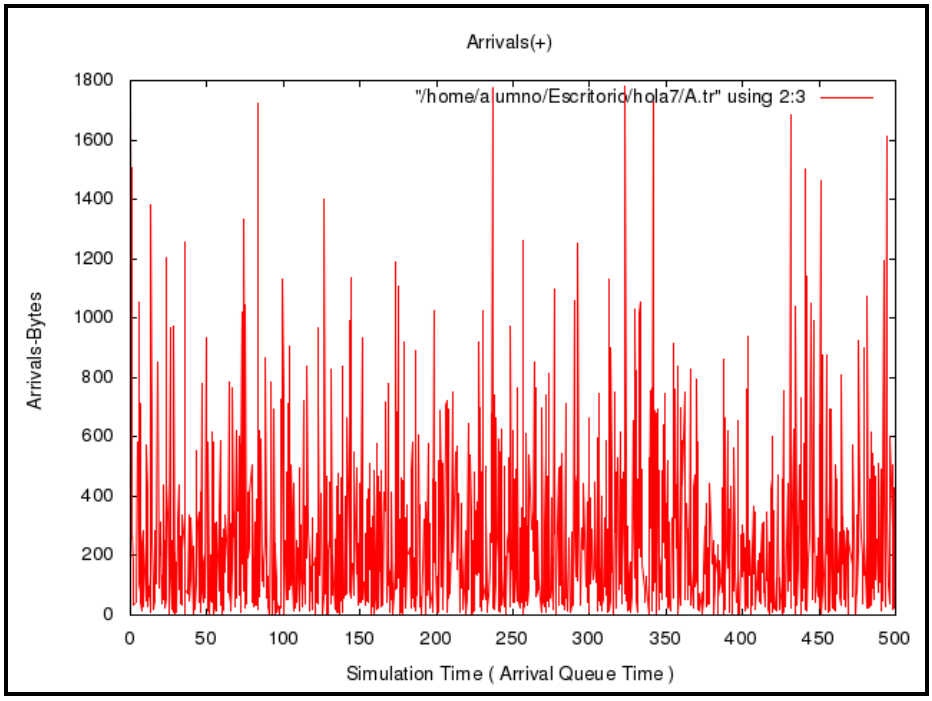


**Figura 34. QNSA/Gráfica 1**

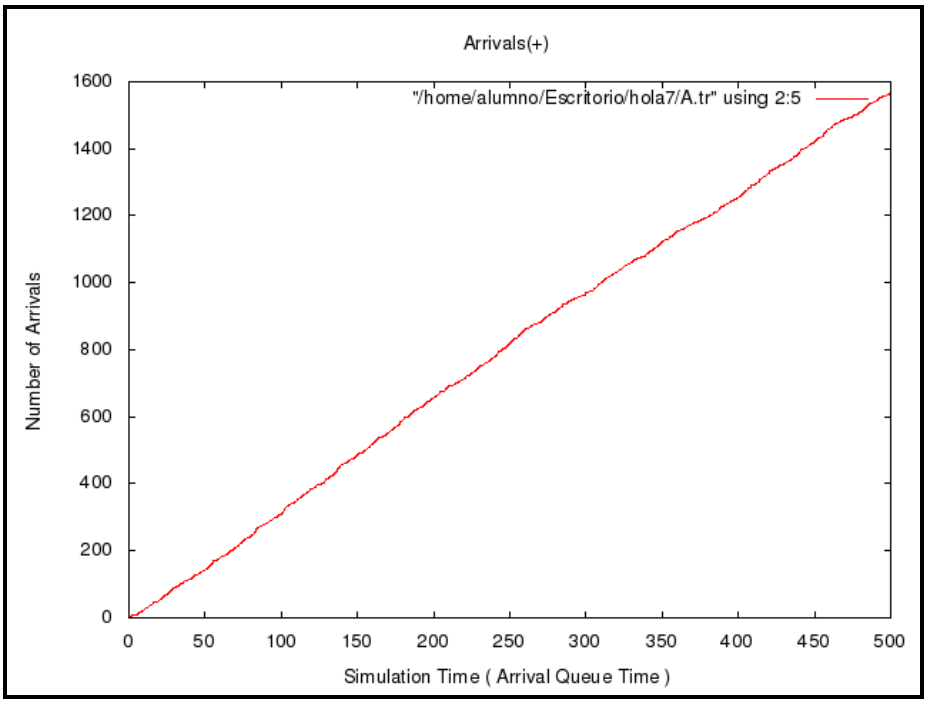


**Figura 35. QNSA/Gráfica 2**

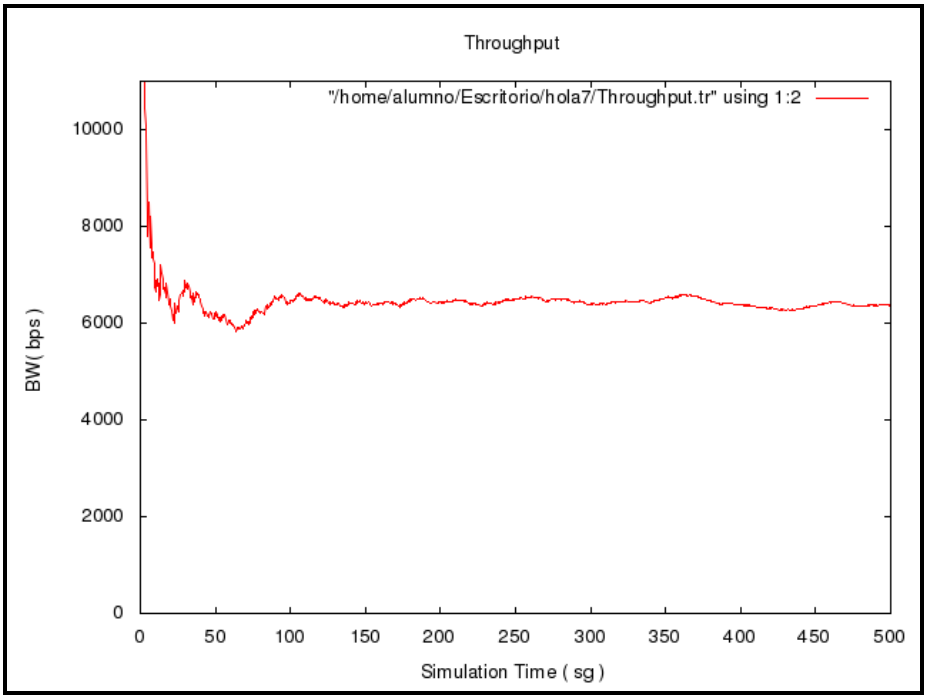


**Figura 36. QNSA/Gráfica 3**

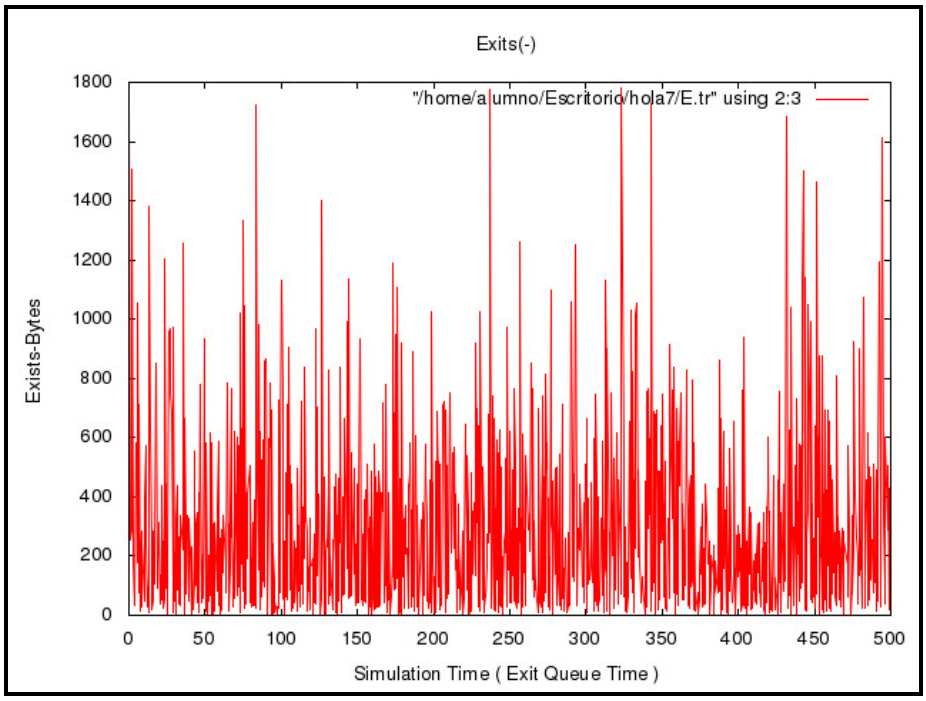


**Figura 37. QNSA/Gráfica 4**

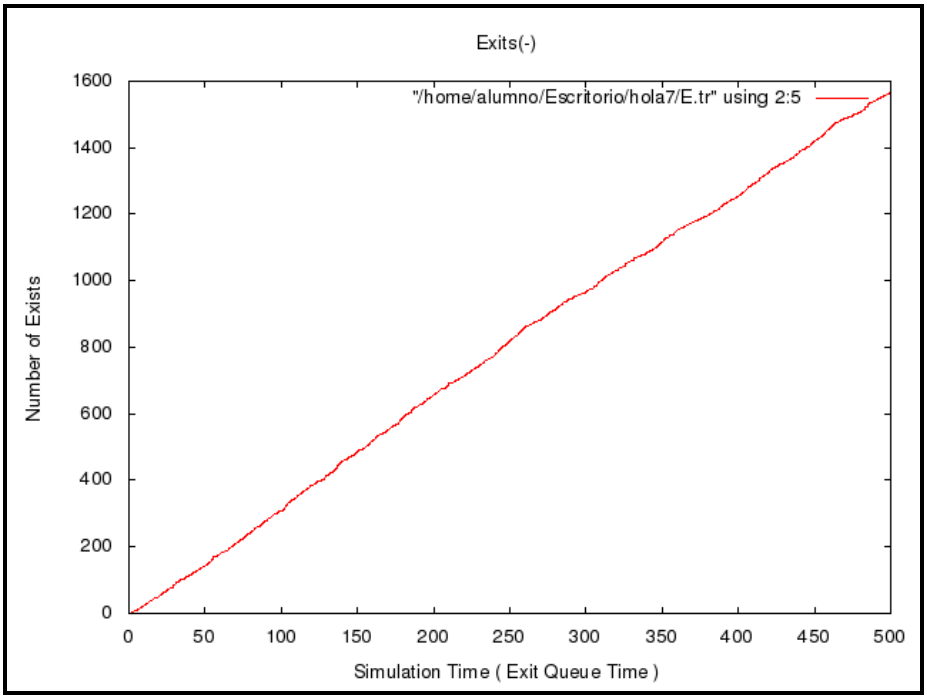


**Figura 38. QNSA/Gráfica 5**

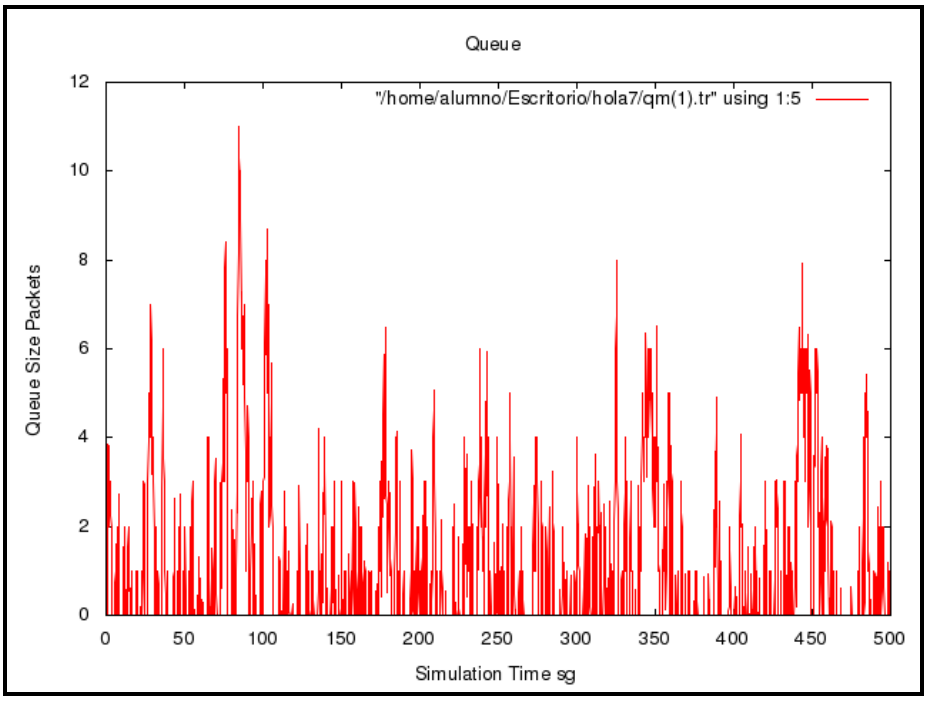


**Figura 39. QNSA/Gráfica 6**

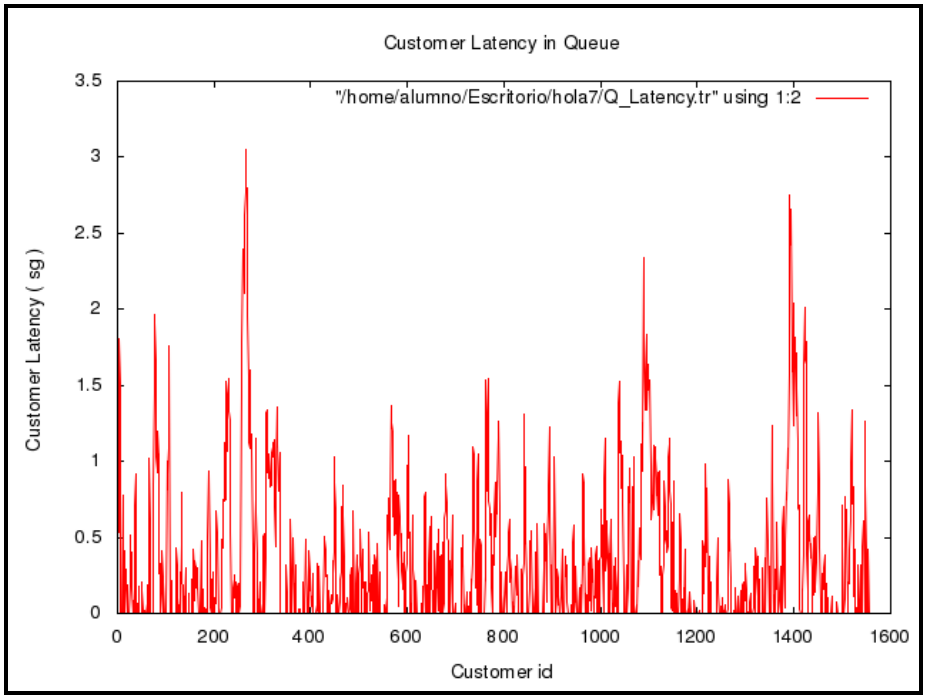


**Figura 40. QNSA/Gráfica 7**

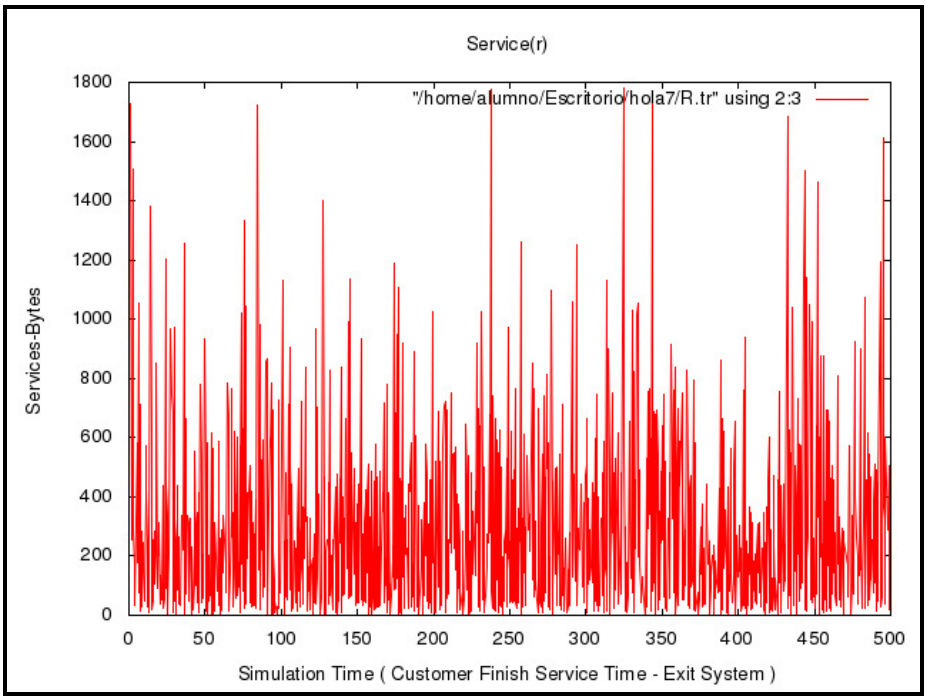


**Figura 41. QNSA/Gráfica 8**

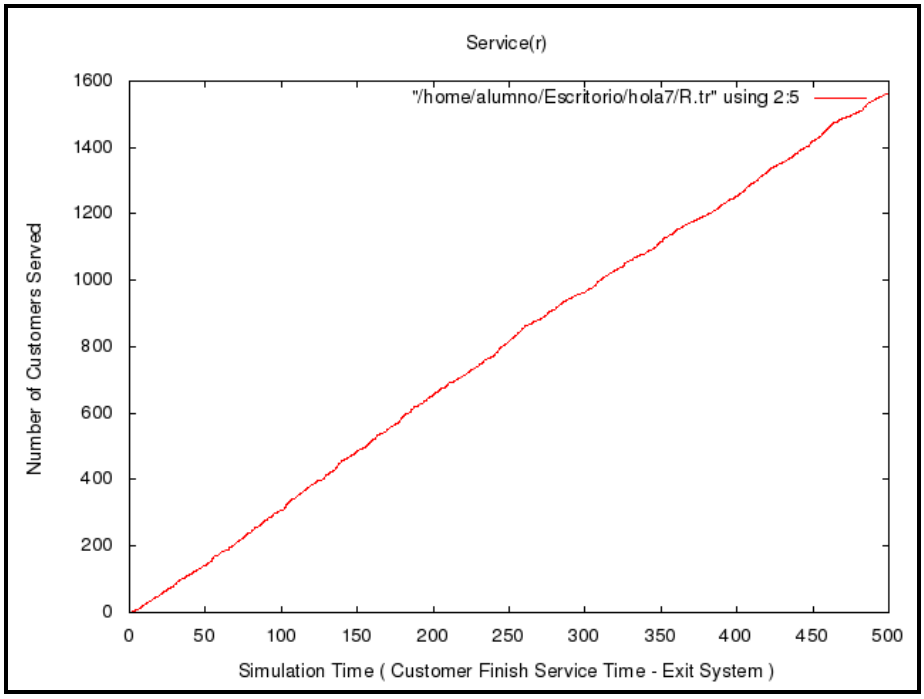


**Figura 42. QNSA/Gráfica 9**

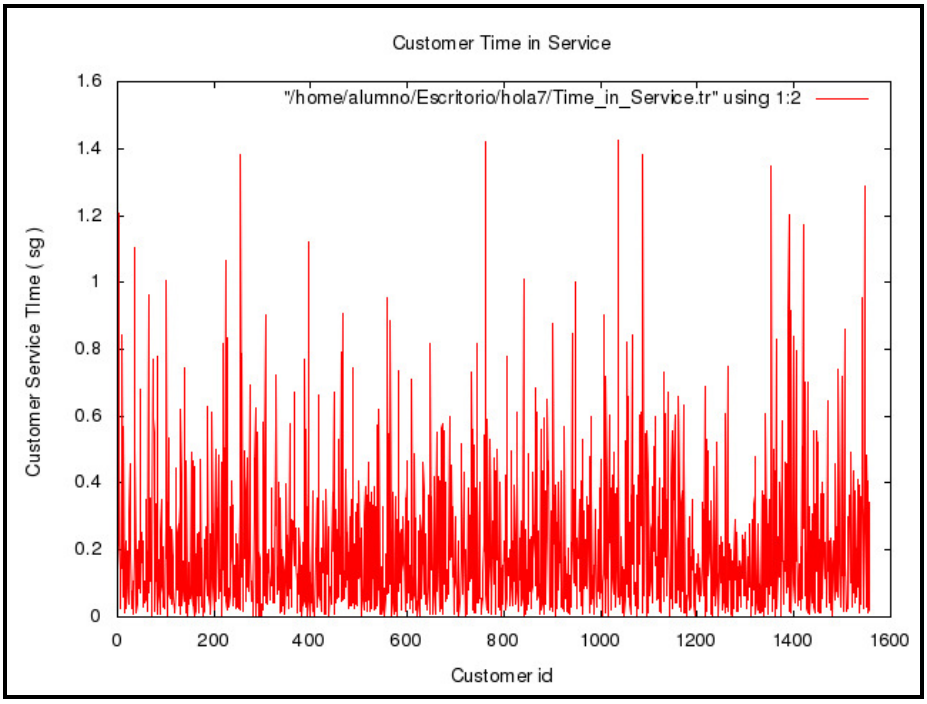


**Figura 43. QNSA/Gráfica 10**

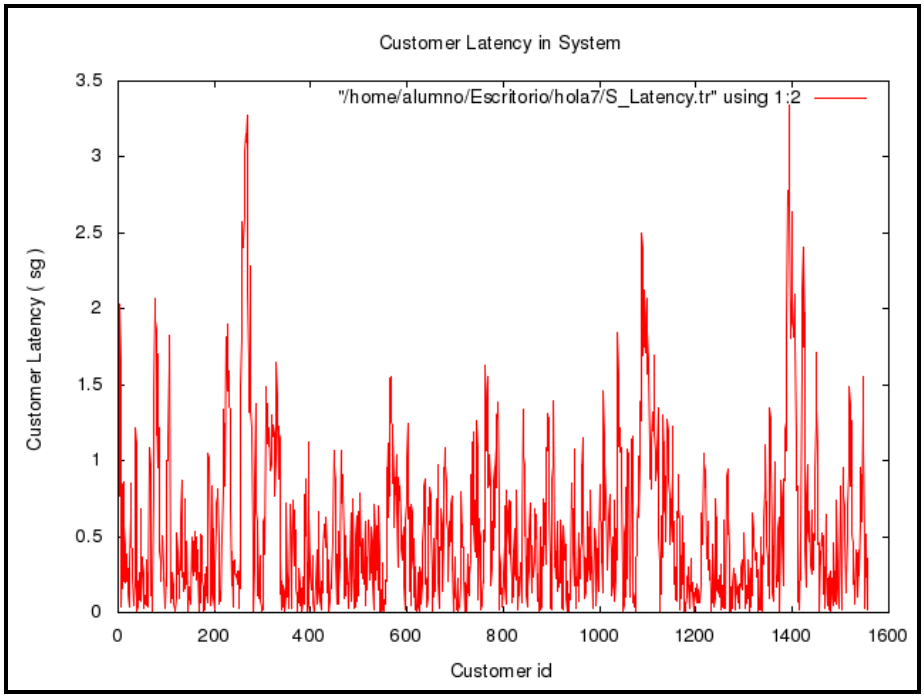


**Figura 44. QNSA/Gráfica 11**

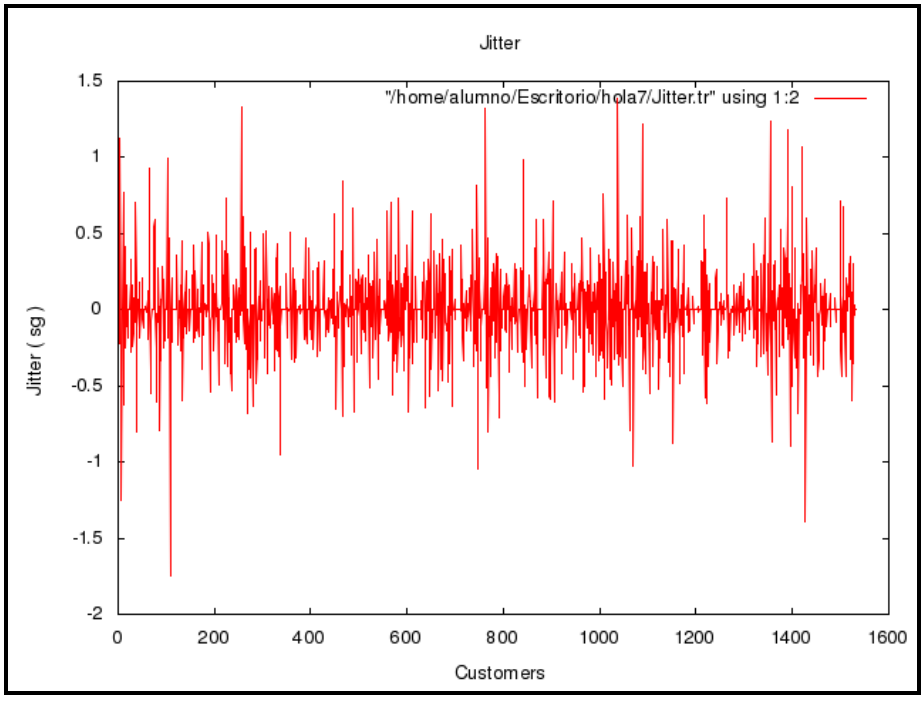


**Figura 45. QNSA/Gráfica 12**

## 9.11. Anexo XI: VMWare y Xen

### 1-VMWare

Uno de los programas de virtualización más conocidos a nivel mundial. Dispone de una variedad de productos donde destacan VMWare Server, Player o Workstation; los dos primeros gratuitos previo registro [48].

Ofrece virtualización completa para entorno doméstico-académico, y paravirtualización en empresa. Al igual que VirtualBox, su instalación es sencilla y destaca por un acceso web muy intuitivo al equipo anfitrión. La creación de máquinas virtuales también es sencilla, y soporta gran número de sistemas operativos. Es fácil encontrar en el WWW máquinas gratuitas y predefinidas con diferentes sistemas operativos según la necesidad (".vmdk" es la extensión de archivo que contiene una máquina virtual Vmware).

Este software permite, entre otras cosas, la opción de reservar una parte del disco duro del equipo anfitrión para que sea usado exclusivamente por una máquina virtual concreta. También existe la opción de que sólo utilice el disco que necesite la máquina virtual, evitando así reservar disco duro que no se vaya a utilizar, permitiendo aumentar dinámicamente según se vaya utilizando.

Dispone además de una herramienta llamada "VMWare Tools" que permite optimizar los recursos, facilitando la interacción entre huésped y cliente, como la posibilidad de copiar documentos directamente entre ambas máquinas o utilizar el ratón sin la necesidad de teclear "Ctrl. + Alt." dentro de la máquina virtual.

### 2-Xen

A pesar de no ser objeto de aplicación en este proyecto, con Xen se hace una mención especial al software de virtualización orientado a servidores. Constituye una de las políticas de gestión y administración informática dentro de los departamentos TIC de escuelas, universidades y empresas con mayor éxito en los últimos años [49]. Su ventaja y aplicación más importante consiste en la consolidación de servidores. En

este caso, las distintas máquinas virtuales configuradas (firewall, servidores web, consola de antivirus, servidor de dominio, etc.), sí ofrecen sus servicios de manera continuada. La Figura 46 muestra un ejemplo práctico.

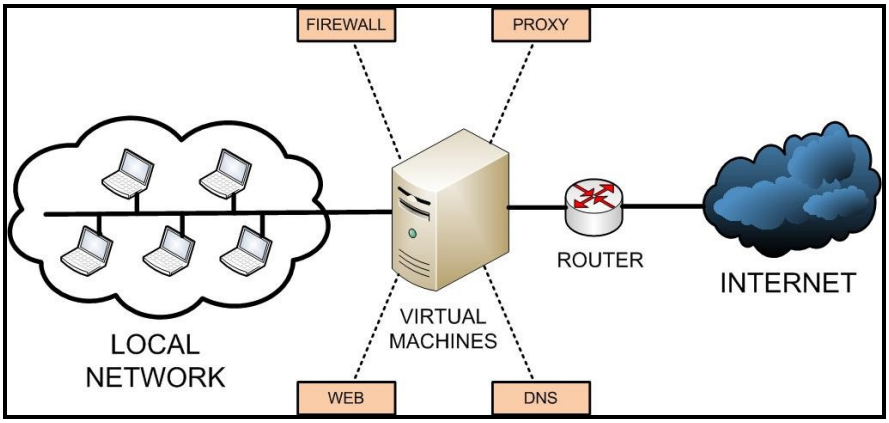


**Figura 46. Virtualización orientada a servidores**

Desarrollado por la universidad de Cambridge. Se trata de software código abierto, y totalmente gratuito [50].

El tipo de virtualización que ofrece se denomina paravirtualización. Aquí, el sistema a virtualizar es previamente modificado. La paravirtualización supone una mejora en el rendimiento puesto que el sistema modificado puede ejecutarse de manera más eficiente sobre el hardware de la máquina anfitrión.

Si el hardware anfitrión posee arquitectura con tecnología Intel-VTx o AMD-V se pueden explotar sistemas huésped sin modificar, ésto da lugar a la virtualización completa de Xen.

Soporta arquitectura x86, y se pueden ejecutar un gran número de sistemas operativos como Windows, GNU/Linux, Solaris, FreeBSD, etc. Una ventaja destacable es que permite la migración de las máquinas virtuales en caliente (ejemplo: balance de carga).

La principal desventaja, para los propósitos del proyecto, es que su configuración y utilización es más complicada que otro software de virtualización, y está más orientado a servidores que a ordenadores de sobremesa.

La comunidad de desarrolladores y usuarios profesionales es muy grande; y por eso, las diferentes soluciones y proyectos propuestos son cada vez más (http://xen.org/community/projects.html).

Su producto estrella es el llamado "Xen Hypervisor".

# 10. Lista de acrónimos

**NS-2** = Network Simulator 2

**TCL** = Tool Command Language

**OO** = Orientado a objetos

**C++** = Lenguaje de Programación C orientado a objetos

**Otcl** = Lenguaje de programación TCL orientado a objetos

**AWK** = Lenguaje de programación para procesar datos (Alfred Aho, Peter Weinberger y Brian Kernighan)

**PERL** = Lenguaje de programación que toma características de otros lenguajes como C, Sh, AWK, sed, Lisp y otros muchos (Practical Extraction and Report Language)

**SdC** = Sistema de Cola = Línea de espera

**GNU** = Sistema Operativo completo tipo Unix de software libre. Se usa habitualmente con un núcleo denominado Linux

**Linux** = Núcleo de Sistema Operativo libre tipo Unix. Se distribuye junto con un paquete de software dando lugar a la distribución Linux

**GNU/Linux** = Sistema Operativo combinación de GNU y Linux

**TIC** = Tecnologías de la Información y la Comunicación

**Cliente** = Entidad demandante de servicio

**Erlang** = Medida de tráfico telefónico que debe su nombre a los trabajos realizados por A.K. Erlang

**FIFO** = First in First out

**LIFO** = Last in First out

**RSS** = Random Service Selection

**CERN** = European Organization for Nuclear Research

**SICC** = Sea of Innovation Cantabria Cluster

**GPSS** = General Purpose Simulation System

**SLAM** = Simulation Language for Alternative Modeling

**SIMAN** = Simulation Analysis

**AQUAS** = Application for solving Queuing problems Analytically and using Simulation

**TCP** = Transmission Control Protocol

**UDP** = User Datagram Protocol

**GPL** = General Public License

**RED** = Random Early Drop

**CBQ** = Class-Based Queuing

**FQ** = Fair Queuing

**SFQ** = Stochastic Fairness Queuing

**DRR** = Deficit Round Robin

**SAMAN** = Simulation Augmented by Measurement and Analysis for Networks

**CONSER** = Collaborative Simulation for Education and Research

**ACIRI** = International Computer Science Institute

**NAM** = Network Animator

**API** = Application Programming Interface

**NS-3** = Network Simulator 3

**GPL** = General Public License

**RNG** = Random Number Generator

**GUI** = Graphical User Interface

**S.O.** = Sistema Operativo

**JVM** = Java Virtual Machine

**JDK** = Java Development Kit

**JRE** = Java Runtime Environment

**IDE** = Integrated Development Environment

**QNSA** = Queuing Network Simulator Application

**Open Source** = Código abierto

**UNO** = Universal Network Objects ( API OpenOffice )

**VMM** = Virtual Machine Monitor

**PUEL** = Personal Use and Evaluation License

# 11. Referencias

[1] Advances in Queueing Theory and Network Applications, Springer Edition. Wuyi Yue, Yutaka Takahashi, Hideaki Takagi

[2] http://www.tlmat.unican.es Asignatura: Redes de Comunicaciones

[3] http://www.buc.unican.es "Modelamiento de sistemas de colas G/G/1 en Network Simulator". José Luis Sanzana Riffo

[4] http://www.portalprogramas.com/software-libre/ranking-universidades/ Ranking de universidades en software libre (RuSL)

[5] http://www.ncr.com The Wait We Hate the Most: "NCR Queue Review" Survey Looks at Consumer Frustration with Waiting for Service

[6] http://www.telegraph.co.uk "Britons spend six months queuing"

[7] Fundamentals of Queueing Theory, 4th Edition. Donald Gross, John F. Shortle, James M. Thompson, Carl M. Harris

[8] Performance Analysis of Queuing and Computer Networks. University of Texas at Dallas U.S.A.

[9] Queuing Theory and Customer Satisfaction: A Review of Terminology, Trends, and Applications to Pharmacy Practice . Ronald Anthony Nosek, Jr., MS* and James P. Wilson, PharmD, PhD†

[10] http://www.scs.org/ Society for Computer Simulation

[11] http://public.web.cern.ch/public/ European Organization for Nuclear Research

[12] Applying Queuing Theory to Optimizing the Performance of Enterprise Software Applications. Henry H. Liu, Ph.D. BMC Software 1030 West Maude Avenue, Sunnyvale, CA 94085, USA

[13] http://www.seaofinnovation.com/home Sea of Innovation Cantabria Cluster

[14] www.tlmat.unican.es/siteadmin/submaterials/615.pdf Comnet III en Unican

[15] Simulación de Sistemas Discretos, ISDEFE, 1996. Jaime Barceló

[16] http://nsnam.isi.edu/nsnam/index.php/User_Information Web oficial NS-2

[17] http://ir.canterbury.ac.nz/bitstream/10092/3072/1/12618837_12618837_Pawlikowski.pdf "Survey of simulators of Next Generation Networks for Studying Service Availability and Resilience". L. Begg, W. Liu, K. Pawlikowski, S. Perera, H. Sirisena

[18] http://www.isi.edu/nsnam/ns/doc/index.html Documentación variada NS-2

[19] The NS Manual (Formerly NS Notes and Documentation). A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Kevin Fall hkfall@ee.lbl.govi, Editor. Kannan Varadhan hkannan@catarina.usc.edui, Editor

[20] NS Simulator for Beginners. Lecture notes, 2003-2004. Eitan Altman and Tania Jiménez

[21] http://ns-2.blogspot.com/ Blog NS-2 y novedades

[22] http://www.isi.edu/nsnam/ns/CHANGES.html Histórico de cambios en simulador NS-2

[23] http://www.cygwin.com/ Web oficial Cygwin

[24] http://www.nsnam.org/ Web oficial NS3

[25] http://www.grymoire.com/Unix/Awk.html AWK

[26] http://www.linuxfocus.org/Castellano/September1999/article103.html Artículo introductorio awk

[27] http://userver.ftw.at/~reichl/publications/WTS07.pdf "Experiences with the ns-2 Network Simulator Explicitly Setting Seeds Considered Harmful"

[28] http://www-st.inf.tu-resden.de/aquila/files/pub/cnds2002-spu-NS-2_rng_shortcomings.pdf "On Shortcomings of the ns-2 Random Number Generator"

[29] http://www.cs.odu.edu/~mweigle/papers/wintersim06.pdf "Improving Confidence in Network Simulators". 2006, Michele C. Weigle

[30] http://www-tkn.ee.tu-berlin.de/research/ns-2_akaroa-2/ns.html Proyecto Akaroa NS-2

[31] http://sourceforge.net/ Descarga de interfaces para NS-2

[32] Interfaces Gráficas y Aplicaciones para Internet. Fco. Javier Ceballos (RA-MA)

[33] http://www.infor.uva.es/~jmrr/TAD2003/Sesiones/TADONJava/JAVA.html Características Java

[34] http://www.programacion.com/Java/tutoriales/J2SE/ Java tutoriales

[35] http://www.java.com/es/download/ Descarga de software Java

[36] http://netbeans.org/downloads/ Descarga de software Netbeans

[37] "When Runtime.exec() won't" de Michael C. Daconta

[38] http://www.openoffice.org/es/ Web software de edición OpenOffice

[39] http://www.gnuplot.info/ Web software de representación gráfica Gnuplot

[40] http://www.openoffice.org/api/docs/Java/ref/overview-summary.html Java API UNO

[41] https://www.virtualbox.org/ Máquina virtual "Oracle VM VirtualBox"

[42] http://www.ubuntu.com Web oficial Ubuntu

[43] "Defects in Random Number Routines of Well-Known Network Simulators and Appropriate Improvements". 2004, Bernhard Hechenleitner

[44] http://www.udc.es/dep/mate/TeoriaColas/colas.htm Simulador Aquas

[45] http://winqsb.softonic.com/descargar Software WinQSB 2.0

[46] http://mason.gmu.edu/~jshortle/fqt4th.html Software QTSplus para Windows

[47] http://qtsplus4calc.sourceforge.net/ Software QTSplus4Calc para Linux

[48] http://www.vmware.com/es/ Descarga de máquina virtual "VMware"

[49] http://recursostic.educacion.es/observatorio/web/ca/software/software-general/462-monografico-maq Monográfico máquinas virtuales en el aula

[50] http://xen.org/ Descarga de máquina virtual "Xen"