INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Ambiente Visual de Simulación y Análisis de Redes de Colas

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ING. JUAN MANUEL HORTA MENDOZA

DIRECTOR DE TESIS

M. EN C. JOSÉ GIOVANNI GUZMÁN LUGO

CODIRECTOR DE TESIS

M. EN C. ROLANDO MENCHACA MÉNDEZ

MÉXICO D.F. MAYO 2007

SIP-14

INSTITUTO POLITECNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 16:00 horas del día 19 del mes de Diciembre de 2006 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"AMBIENTE VISUAL DE SIMULACIÓN Y ANÁLISIS DE REDES DE COLAS"

HORTA	**MENDOZA**	**JUAN MANUEL**
Apellido paterno	materno	nombre(s)

Con registro: | B | 0 | 4 | 1 | 2 | 4 | 6 |

aspirante al grado de: ***MAESTRO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Jesús Guillermo Figueroa Nazuno

Secretario

Dr. Felipe Rolando Menchaca García

Primer vocal
(Director de tesis)

M. en C. José Giovanni Guzmán Lugo

Segundo vocal
(Co-director)

M. en C. Rolando Menchaca Méndez

Tercer Vocal

Dra. Elsa Rubio Espino

EL PRESIDENTE DEL COLEGIO

INSTITUTO POLITECNICO NACIONAL
SECRETARIA DE INVESTIGACION Y POSGRADO
13 FEB 2007
RECIBIDO
DIRECCION DE POSGRADO

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION

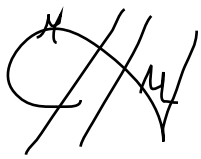
INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de ____México___el día __3__del mes__Mayo__del año ___2007__, la que suscribe___Juan Manuel Horta Mendoza_ alumna del Programa de___Maestría en Ciencias de la Computación____ __con número de registro _B041246__, adscrito al __Centro de Investigación en Computación_____, manifiesta que es autora intelectual del presente trabajo de Tesis bajo la dirección del _M. en C. José Giovanni Guzmán Lugo y M. en C. Rolando Menchaca Méndez_ y cede los derechos del trabajo intitulado __Ambiente Visual de Simulación y Análisis de Redes de Colas__, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección ___jmhortam@gmail.com___. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Juan Manuel Horta Mendoza__

Nombre y firma

Agradecimientos.

A mis asesores el M. en C. Rolando Menchaca Méndez y el M. en C. Giovanni Guzmán Lugo, por su dedicación y guía que recibí de ellos para la realización de esta tesis.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada para realizar mis estudios de maestría.

Además agradezco a todas las personas que me apoyaron y de una u otra forma me ayudaron a que este trabajo se pudiera realizar.

Resumen.

El estudio de las redes de colas es importante en varias áreas como en la industria, redes de computación, desempeño computacional y otras. Esto se debe a que los modelos de redes de colas son muy útiles en su funcionamiento, ya que representan el núcleo del negocio(o parte de este) en muchas de estas áreas. Además el funcionamiento de las colas, representa el desempeño total del sistema que a finalmente se traduce en costo para muchas de estas áreas.

Este trabajo presenta un ambiente visual de simulación y análisis de redes de colas llamado QSim, el cual permite al usuario modelar una red de colas de manera rápida y fácil para después simular y obtener resultados de desempeño. La aplicación está basada en un simulador de eventos discretos programado en el lenguaje de programación Java. Además se incluye un generador de código listo para simularse, apartir de un archivo descriptor de una red de colas en XML y una interfaz de usuario amigable. La interfaz de usuario permite modelar de manera visual una red de colas y se incluye un módulo de graficación además de un módulo de análisis estadístico para dar más información de los resultados obtenidos.

Con este simulador que se presenta se obtuvieron resultados acercados a los obtenidos de manera analítica, lo cual indica que la aplicación es confiable para simular muchos modelos de redes de colas.

Abstract.

The queueing network theory is important in many study areas like industry, computer networks, computer performance and more. This is because of the queueing networks models are very useful in their whole function while queueing networks represent the whole or part of their core business function. Furthermore the operation of this queueing networks represent the total system performance, the intime results in cost for many of this areas.

This theses presents a visual environment for queueing networks simulation and analysis called QSim. This application allows the user to model and simulate queueing networks fast and easy and then obtain the performance results for further analysis. The core of the application is based on a discrete event simulator developed in the programming language Java. In addition there is included a simulation ready code generator from a XML file descriptor. The graphic user interface let the user model a queueing network in a visual way; it is also included a plot module and a statistic anlysis module to provide more result information.

The results obtained by this simulator are close to the results obtained analytically, this concludes into a reliable application to simulate many queueing network models.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Glosario.

1. **Variables de Estado:** Estas variables definen el estado del sistema. Son importantes ya que si una simulación se detiene, solo se puede reiniciar después si y solo si todas las variables de estado son conocidas.

2. **Evento:** Se define como un cambio en el estado del sistema.

3. **Modelos de Tiempo Continuo o Discreto.** Un modelo en el que su estado está definido durante todo el tiempo es llamado modelo de tiempo continuo. Si solo esta definido en instantes específicos del tiempo, entonces se llamará modelo de tiempo discreto.

4. **Modelos de Estado Continuo o Discreto.** Un modelo es llamado de estado continuo o discreto dependiendo si las variables de estado son continuas o discretas. Esto es si sus variables pueden tomar cualquier infinidad de valores, entonces será un modelo de estado continuo. De manera análoga si solo pueden tomar algunos valores específicos, entonces será un modelo de estado discreto.

5. **Modelos Determinísticos o Probabilísticos.** Si se puede predecir la salida de un modelo con certeza, entonces será un modelo determinístico. Un modelo probabilística dará diferente resultado en repeticiones para un mismo conjunto de parámetros.

6. **Modelos Estáticos o Dinámicos.** Si en un modelo el tiempo no es una variable, entonces será estático. Si el estado del sistema cambia según el tiempo será dinámico.

7. **Modelos Lineales o No Lineales.** Un modelo será lineal si los parámetros de salida son una función lineal de los parámetros de entrada, de otra manera será no lineal.

8. **Modelos Abiertos o Cerrados.** Si la entrada para un modelo es externa y es independiente de éste, entonces es un modelo abierto. Si no existe una entrada será un modelo cerrado.

9. **Modelos Estables o Inestables.** Cuando el comportamiento dinámico de un modelo cesa hasta un estado quieto, y es independiente del tiempo, entonces se llama modelo estable. Si un modelo en el que su comportamiento cambia continuamente es llamado inestable.

10. **Disciplina de una cola.** Son una serie de reglas que determina el orden en que los trabajos de entrada son atendidos.

CAPÍTULO 1

Introducción.

Además de los sistemas computacionales, las colas son parte de la vida de todo individuo, para comprar boletos, para hacer operaciones bancarias, pagar en el supermercado, etc., en todas ellas se tiene que esperar un determinado tiempo, y de una u otra manera el individuo se acostumbra, pero a veces esta espera es molesta. Se calcula que en E.U. aproximadamente la gente gasta 37 billones de horas por año en una cola de espera, si se traducen estas horas a productividad, se hablaría de estimar cerca de 20 millones años-persona de trabajo útil cada año [1].

El impacto de esta espera excesiva, daña en muchos ámbitos, donde las colas de espera son parte de la lógica del negocio de una empresa. Hay máquinas que esperan a ser reparadas, y que en consecuencia se pierde producción. Los aviones esperan a despegar y aterrizar, y pueden alterar el retraso de los vuelos siguientes. En las comunicaciones, una espera por tráfico genera una conexión con fallas. Un trabajo de manufactura en espera puede interrumpir la producción subsiguiente lo cuál se traduce en costo de producción.

Por esto, los modelos de colas son importantes de analizar y evaluar, para así determinar la manera más eficiente de operarlos. Con esto se busca encontrar un balance entre el costo del servicio y la cantidad de espera, ya que si se tiene mucha capacidad de servicio tal vez esté muy sobrado, y el costo es muy alto; de manera contraria, si no se tiene mucha capacidad, la espera será muy larga y las consecuencias serán algunas de las mencionadas anteriormente.

En los sistemas computacionales, todo sistema en donde esté involucrado una entidad que preste un servicio, ya sea un servidor de recursos en línea, un procesador o una impresora, estará sujeto a tener una cola, en la que los usuarios o tareas, tendrán que esperar para poder ser atendidos. Esto lleva al

estudio del desempeño de estos sistemas. ¿Qué tanto tiempo tarda en la cola?, ¿Cuánto tiempo tarda el usuario dentro del sistema?, ¿Cuántos usuarios soporta el sistema?, son algunas de las preguntas que la evaluación del desempeño puede contestar, por medio de diferentes técnicas análisis.

Este documento da una solución a través de la programación de un simulador de redes de colas, con el fin de analizar cualquier sistema compuesto por n entidades de servicio y m servidores, de manera sencilla y rápida. El simulador obtendrá los resultados del desempeño de la red de colas simulada. También con los resultados obtenidos del simulador programado, se pueden determinar problemas comúnes de estos sistema, como identificar cuellos de botella, entidades de servicio con poca o nula ocupación, colas que crecen muy rápido, etc.

1.1. Estructura de la tesis.

La estructura de este documento será la siguiente:

Capítulo 1: Introducción, se da una introducción a la simulación de redes de colas y una breve revisión a la teoría de este tipo de simulación.

Capítulo 2: Procesos estocásticos y redes de Markov, se revisará la importancia de estas estructuras para entender mejor este documento ya que son la base para la solución analítica de los modelos de redes de colas.

Capítulo 3: Sistemas de colas, aquí se detallan algunos de los modelos más importantes y más estudiados de la teoría de redes de colas y se denotan sus resultados analíticos.

Capítulo 4: Redes de colas, se describe este tipo de modelos en teoría, también se revisan los modelos más importantes como lo son las Redes de Jackson.

Capítulo 5: Simulación, se centra en la teoría de simulación, describe la importancia de la generación de números aleatorios, análisis de resultados y finalmente se revisa la simulación de eventos discretos que es el núcleo del simulador que se presenta en este documento.

Capítulo 6: Diseño e implementación, se presenta cómo fue construido el simulador programado aquí y las bases que se siguieron para llevar a cabo la programación de todos sus componentes.

Capítulo 7: Pruebas y resultados, se muestran los resultados obtenidos con el simulador presentado en este documento y una comparación con los resultados que se obtienen de manera analítica.

Capítulo 8: Conclusiones, se presentan las conclusiones obtenidas de este trabajo y algunas mejoras que se pueden realizar al simulador presentado.

1.2. Planteamiento del problema.

Hoy en día en muchos lados se ven colas de espera por atención, es decir se pueden encontrar en muchos lugares que se tiene que esperar un determinado tiempo para finalmente obtener un servicio. Este fenómeno se puede observar en el banco, en una tortillería, en el supermercado, restaurantes, parques de diversiones también se ve en salas de espera del seguro social, dentista, etc, lugares muy comúnes en la vida cotidana de toda persona. Asimismo se puede observar que algunas veces la manera de espera no es una línea de personas, inclusive podría no ser en un lugar físico como lo son en los centros de atención telefónicos en dónde se tiene que esperar a que un ejecutivo atienda la llamada. Algunas de las compañías invierten mucho dinero en el estudio del comportamiento de su sistema de atención, esto va desde que llega un cliente, en cuánto tiempo es atendido, cuánto tiempo se toma atenderlo y sale. Esto es por que las empresas saben que parte de su negocio final (vender, realizar operaciones bancarias, prestar un servicio) involucra una cola de atención y para ellos es muy importante que el cliente espere a que se le atienda.

Así, existen muchos factores que se estudian desde que el cliente no tenga que esperar tanto a que se le atienda, hasta que cómo hacer que tenga una espera placentera. Estas empresas estudian todos estos factores de tiempo de atención, cantidad de clientes en espera, cuántos clientes se retiran antes de ser atendidos, etc. Usualmente estos estudios se realizan en simuladores computacionales, en los cuales se modelan estos sistemas de atención y se simula el flujo de clientes que se tienen.

En la actualidad estos simuladores son muy costosos, además los modelos y el análisis de resultados los tiene que hacer un experto en la materia. Existen algunos simuladores gratuitos, sin embargo necesitan que los programe un experto en el simulador y en materia de teoría de redes de colas, ya que no presentan algún tipo de interfaz gráfica y el lenguaje de programación es otra limitante. Es por eso, que se presenta un simulador con las capacidades de llevar a cabo la simulación de muchos de estos modelos de redes de colas, presentar y facilitar el análisis de resultados precisos al usuario y facilitar el modelado de estos sistemas al proveer una interfaz de usuario gráfica fácil de usar.

1.3. Antecedentes.

En la actualidad existen lenguajes para programar y modelar sistemas que requieren de la simulación como lo son *Simula*[2](uno de los lenguajes más antiguos Orientado a Objetos) *ModSim*[3] o *SimScript*[4], por mencionar algunos. Estos lenguajes proveen de funciones y librerías para programar (o modelar) sistemas que se necesitan simular, la principal ventaja que ofrecen es que el programador se enfoca más en el modelo del sistema que en lo que necesita para simular y obtener los resultados

estadísticos.

A pesar de que existen estos lenguajes especializados, los lenguajes de propósito general se imponen y son utilizados más para la modelación y simulación de sistemas de colas. A lo largo de la historia se han realizado varias librerías que proveen de la funcionalidad de los lenguajes de simulación. En 1987 M.H. Dougall programó una serie de funciones en C, que proveen de toda la funcionalidad para poder modelar y simular un sistema de colas y finalmente obtener resultados de interés para el desarrollador. Actualmente este conjunto de funciones sigue en uso, ya que tiene facilidades para modelar y obtener los resultados requeridos. Sin embargo, para modelar es necesaria la programación del sistema en lenguaje C, esto implica la compilación y la ejecución de un nuevo programa para obtener los resultados requeridos.

De manera similar, desde la salida de la primera versión del lenguaje de programación Java, se han realizado diferentes paquetes con un conjunto de clases para la simulación con eventos discretos. Algunos de los más importantes son *JSim*[5](Universidad de Georgia), *SimJava*[6](Universidad de Edinburgo), *JavaSim*[7](Universidad de New Castle). Todos proveen de clases para que el programador modele un nuevo sistema y lo pueda simular. A pesar de que los paquetes son muy completos, no proveen de un ambiente gráfico para la rápida modelación, aunque algunos proveen salidas de sistemas gráficos.

En el sector comercial, las aplicaciones están particularizadas a la simulación de sistemas muy específicos, como lo son la manufactura/procesos industriales (*Simul8*, *FlexSim*), industria hidroeléctrica, finanzas (*GoldSim*), etc.; uno de los programas que proveen más flexibilidad en los escenarios es *Promodel*, que integra diferentes sistemas de colas por ejemplo la simulación de una caseta de cobro en un estacionamiento de un banco, así como la simulación interna del banco. La mayoría de estos programas comerciales, proveen de interfaces de usuario amigables para modelar un sistema de manera rápida y de la misma manera obtener resultados de la simulación rápidamente. Sin embargo, no están al alcance de cualquier persona ya que estos programas son sumamente costosos ya que los precios varían desde los 700 usd. hasta los 17,000 usd. (precio aproximado de *Promodel*). Pero este costo es una gran inversión para las empresas, ya que en la mayoría de los casos en el que se utilizan estas aplicaciones, los sistemas modelados representan el negocio principal de la empresa, así la generación de modelos y obtener resultados factibles para la optimización de procesos a partir de las simulaciones, representa ahorros o ganancias a la empresa.

1.4. Objetivos y aportaciones.

1.4.1. Objetivo general.

Programar un área de trabajo que provea de un ambiente gráfico para que el usuario pueda modelar un sistema de colas y después pueda simularlo. Para así obtener resultados que se puedan analizar (ya sea por medios visuales o de cifras numéricas) y valorar para generar un análisis de desempeño del sistema modelado.

Para lo cual es necesario:

- Comprender la teoría de desempeño de sistemas,
- Analizar la relación y la equivalencia entre un sistema real y la modelación y simulación del mismo.
- Programar un núcleo para la simulación basada en eventos discretos.
- Realizar una interfaz gráfica de usuario sencilla de usar.

1.4.2. Aportaciones.

Se programó un simulador de eventos discretos en Java, con la capacidad de permitir al usuario modelar y simular redes de colas. En comparación con otros simuladores en Java, el de esta tesis provee una interfaz de usuario amigable en la cual el usuario puede modelar con iconos y flechas una red de colas para después simularla y obtener resultados de manera gráfica. Además se integró un módulo de análisis estadístico que permite hacer simulaciones de un mismo modelo con diferentes semillas y así obtener resultados más precisos. Por último se incluye un generador de código Java apartir de un archivo XML que describe una red de colas.

1.5. Marco teórico.

1.5.1. Evaluación de desempeño.

Un parte importante de la evaluación de un sistema es la medición del desempeño o también llamada monitoreo. Con el monitoreo se obtendrán diferentes vistas del sistema, como el ver qué operaciones toman más tiempo o a qué componentes se les da una carga mayor de trabajo.

Para que este monitoreo se pueda llevar a cabo se tendrán que modificar algunas partes del sistema, ya que se requieren tomar tiempos y otros datos; esto modificará el sistema (en el código) y también su desempeño real del sistema. A este tipo de monitoreo se le llama monitoreo por software. Si se requieren mediciones de hardware, habrá que instalar hardware especializado (usualmente sensores) para hacer mediciones acerca de la carga de direcciones, carga en los buses, etc.. De manera análoga al monitoreo anterior, a este se le llama monitoreo por hardware. También se pueden combinar ambos para obtener un monitoreo híbrido.

En ambos casos, el desarrollo de estas herramientas es un costo extra, en consecuencia el monitoreo no siempre se puede realizar. En vez de este monitoreo se puede utilizar una evaluación de desempeño basado en modelos. Este tipo de evaluación no requerirá que el sistema a estudiar, esté ya construido y en funcionamiento, solo se necesitará al menos de una descripción del sistema para así obtener un modelo abstracto.

En el contexto de la evaluación de desempeño, un modelo es una descripción abstracta (basada en conceptos matemáticos bien definidos) de un sistema en términos de sus componentes y sus interacciones con el ambiente. Este ambiente se refiere a como se usa el sistema por humanos u otro sistema, y es llamado el modelo de carga de trabajo.

Como en muchos escenarios, donde hay que modelar algo, para el caso de modelar un sistema de comunicaciones no es nada fácil. Existen algunas guías que se pueden seguir para crear un modelo, pero no existe un método establecido para realizar un modelo. Asimismo definir lo que es un buen modelo, es un tanto subjetivo, por lo tanto un modelo podrá ir de algo muy simple hasta uno demasiado complejo.

Un modelo dependerá mucho del tipo de medición que sea de interés hacer al sistema en cuestión. Estas mediciones pueden ser orientadas a usuario u orientadas a sistema.

Las mediciones usuales orientadas a usuario (también conocidas como orientadas a tarea) pueden ser:

- Tiempo de respuesta. (R)
- Salida de tareas. (X)
- Tiempo de espera de las tareas. (W_q)
- Tiempo de servicio de las tareas. (S)

En cualquiera de estos casos las medidas señalan algo acerca del desempeño de las peticiones al sistema (o tareas) que son hechas por los usuarios del sistema.

Algunas de las mediciones más comunes de las orientadas a sistema son:

- Número de tareas en el sistema. (N)
- Número de tareas en alguna cola de sistema. (N_q)
- Utilización de los componentes de sistema. (ρ)

Estas mediciones dicen algo acerca de la organización interna del sistema a estudiar.

Ya que se definió que se requiere medir, hay que verificar que tan detallado se requiere que estas mediciones sean. Si se necesita una medición muy exacta, entonces el modelo tendrá que ser lo suficientemente detallado, para que las mediciones sean lo más aproximadas posibles a algo real. De otra manera, si solo se necesitan promedios, y mediciones no tan exactas, entonces vale la pena mantener un modelo lo más simple posible. Es recomendable hacer un modelo abstracto justo con solo ciertas suposiciones, en vez de un modelo muy detallado al que no se le puedan dar datos de entradas requeridos. Entonces la salida del modelo dependerá y se interpretará de acuerdo a los datos de entrada, es decir, un modelo es tan bueno como sus datos de entrada lo sean.

Una vez que se construyó un modelo, deberá ser analizado, que se puede realizar con muchas técnicas. En muchos casos prácticos éste análisis deberá ser comprobado, preferentemente por herramientas de software, ya que los sistemas reales son muy complejos, y un análisis a mano, no tendrá suficiente certeza.

Finalmente el resultado numérico arrojado por el análisis deberá ser debidamente interpretado para verificar si se obtiene un resultado útil, de otra forma es posible que se tenga que modificar el modelo o usar alguna otra técnica. Si se obtienen los resultados requeridos estos resultados se deberán interpretar en términos de la operación del sistema modelado. A todo este proceso definido se le llama evaluación de sistema o evaluación de desempeño de sistema.

Todo este proceso también se puede combinar con el uso de monitoreo y así obtener algunos datos de entrada que le pueden servir a un modelo de sistema detallado justo. La salida que se obtendrá podrá llevar a conclusiones acerca de que ajustes requerirá el sistema realmente y que inversiones serán más necesarias.

1.5.2. Técnicas de solución por modelo.

Existen dos importantes clases de técnicas: las analíticas y las simulativas.

Si el modelo llena un número de requerimientos, se pueden calcular algunas mediciones directamente de la manera analítica. Las mediciones analíticas son convenientes, ya que son más exactas y de una manera u otra los resultados son inmediatos. Sin embargo, no muchos sistemas reales se pueden modelar de alguna forma que los requerimientos estén determinados. Éstas técnicas implican tiempo al derivarlas y aplicarlas. La mayor ventaja es que se puede obtener una vista rápida en algunos propósitos de diseño del sistema.

En estás técnicas existe una subdivisión: las llamadas técnicas analíticas de forma cerrada, las cuales darán como resultado mediciones de desempeño dadas expresiones explícitas en término de la estructura y parámetros del sistema, pero solo se pueden aplicar a los modelos más simples. Las otras técnicas son llamadas numéricas, con las cuales se obtendrán ecuaciones de las cuales se puede obtener la solución por medio de técnicas de análisis numérico, por ejemplo con procedimientos iterativos.

Para casi toda clase de modelos, no existen técnicas analíticas para obtener soluciones de los modelos. En estos casos se tienen que utilizar técnicas de simulación para resolver el modelo, es decir, obtener las medidas requeridas. Con la simulación se puede imitar al sistema en cuestión, donde se utiliza generalmente un software de simulación, y así se toman tiempos, se generan los eventos deseados, etc. Al final se tomarán estos datos para obtener las mediciones de interés.

Es complicado definir que tipo de técnica se va a utilizar ya que como se puede observar, las dos tienen sus ventajas y desventajas; las analíticas pueden ser menos costosas, pero los modelos no pueden ser acertados para analizarse por medio de ésta técnica; las simulativas pueden llevar a que el usuario modele un sistema muy complejo ya que no se restringe la simplicidad al utilizar esta técnica, debido al apoyo de una computadora. Usualmente se utiliza un punto intermedio como lo puede ser la técnica numérica que puede utilizar algo de las dos técnicas anteriores.

1.5.3. Modelos estocásticos.

Como se señaló en la sección anterior, con los modelos de los sistemas, interesa observar como se comportaría un sistema que no existe aún, así surge una incertidumbre de los modelos que serán desarrollados.

La incertidumbre en los parámetros del sistema se pueden confrontar con un análisis paramétrico, esto es, resolver un modelo muchas veces con diferentes parámetros. Como resultado se obtendrá un análisis de una medida de interés contra una serie de variaciones de un parámetro.

Otro tipo de incertidumbres se enfocan al uso del sistema, o también llamado carga de trabajo del sistema. Éste uso es dependiente de muchos factores, que no pueden ser descritos de manera determinista, es decir que no se puede predecir qué necesitarán los usuarios que haga el sistema, en

un futuro. Lo único que se puede saber, es por medio de las estadísticas acerca del uso en el pasado, o hacer especulaciones acerca de un comportamiento en el futuro. Estas incertidumbres llevan a usar variables aleatorias en los modelos, así expresará, de una manera estocástica[1], la incertidumbre acerca del comportamiento de uso.

Cuando se hacen suposiciones estocásticas, entonces se terminará con la creación de modelos estocásticos. El comportamiento total del sistema entonces será descrito como un proceso estocástico en el tiempo, es decir, una colección de variables aleatorias que cambian su valor a lo largo del tiempo. Las medidas de desempeño que interesen, tendrán que ser expresadas como funciones de los procesos estocásticos, y dependiendo de del tipo de procesos y medidas requeridas, estas funciones serán igualmente más o menos fáciles de determinar.

1.5.4. La notación de Kendall.

Usualmente se utiliza la notación de Kendall, para describir de manera corta, un sistema de colas en una manera no ambigua. Consiste de seis identificadores, separados por barras verticales, como:

Llegadas | Servicios | Servidores | Tamaño de Buffer | Población | Planeación,

donde las *Llegadas* significa el tiempo entre el proceso de llegada de cada cliente, *Servicio* se refiere a tiempo que requiere el servicio, *Servidores* indica el número de entidades que proveen el servicio, *Tamaño del Buffer* señala el número máximo de clientes en la estación de cola, que incluye al cliente que posiblemente está en el servidor, la *Población* indica el tamaño de la población de clientes, y finalmente, *Planeación* es la estrategia de planeación empleada. Los últimos tres indicadores, pueden ser omitidos, ya que si se omiten, se supondrá que el tamaño de buffer y la población son infinitos, y que la estrategia de planeación es FCFS (Primero en Llegar, Primero en ser Servido, por sus siglas en inglés). Los parámetros de llegada y servicio, pueden adquirir diferentes valores, los más comunes son:

- M (Markoviano), cuando el tiempo de llegada o de servicio son distribuidos de manera exponencialmente negativa.
- G (General), cuando el tiempo es arbitrariamente distribuido.
- D (Determinístico), cuando el tiempo es constante.
- E_r (estado-r Erlang), cuando el tiempo está distribuido conforme a una distribución Erlang-r.
- H_r, cuando el tiempo esta distribuido conforme a una distribución hiper-exponencial de estado-r.

[1]Estocástico: f. Mat. Teoría estadística de los procesos cuya evolución en el tiempo es aleatoria, tal como la secuencia de las tiradas de un dado. DRAE, http://buscon.rae.es/diccionario/cabecera.htm.

1.5.5. Soporte de herramientas.

No es posible construir de manera directa modelos estocásticos de sistemas muy complejos, que describan todos los aspectos importantes de las mediciones que se requieran realizar. Para esto, es recomendable construirlo en pasos, es decir, construir pequeños modelos para después unirlos en uno solo, así la derivación del modelo estocástico, será casi inmediata. Así se puede llevar la construcción de estos modelos al área de más del lado de un diseñador de sistemas y menos del lado del matemático, aunque siempre será necesario el conocimiento matemático detrás de los modelos.

Ahora bien, ya que se obtuvo el modelo estocástico, se tienen que extraer las mediciones requeridas. Por lo tanto, la complejidad de solución del modelo, como ya se explicó con anterioridad en la sección 1.5.2, va a depender del detalle de conocimiento de la simulación o de las técnicas analíticas-numéricas.

1.5.6. Construcción del modelo.

Las herramientas de software son de gran ayuda para la construcción de modelos estocásticos grandes. De aquí se pueden diferenciar dos representaciones de un modelo, la analítica, que usualmente estará lista para una evaluación numérica, y la representación del modelador, que es una descripción simbólica orientada a una aplicación específica, que es el sistema a ser modelado. Por esto, los diseñadores prefieren la representación del modelador.

La idea de tener diferentes puntos de vista de un modelo, dependiendo de lo que se quiera hacer con él, es la idea central del marco de trabajo de una herramienta general de modelado (*GMTF, General Modelling Tool Framework)*, para modelar de sistemas cuantitativos.

En resumen, la formalización de los modelos se lleva a una jerarquía de formalismos descriptivos, de una formalización F_0(el más bajo nivel) a una F_n(el nivel más alto). El nivel más bajo corresponderá a modelos que son directamente adecuados para evaluaciones de una u otra forma, con el uso de técnicas analíticas, numéricas o de simulación; el nivel más alto corresponderá a formalismos más cercanos al dominio de la aplicación.

Definición 1 Se define un $F_i - modelo$ como un proceso de abstracción, simplificación o de reescribir una descripción de sistema S de una forma que se amolde al formalismo F_i.

Definición 2 El resultado del proceso anterior se llamará un F_i-*modelo* M_i de S, que se puede reescribir a otro formalismo F_{i-1} (donde $i \geq 1$) que denotará un F_{i-1}-*modelo* M_{i-1} de S.

Definición 3 A este formalismo para un F_{i-1}-*modelo* se le llama un $F_{i-1} - modelado$, y F_0 coincidirá con la representación analítica.

Cuando F_i es el formalismo de más alto nivel, la mayoría de la actividad del usuario del modelado será $F_i - modelado$. Las actividades de más bajo nivel de modelado pueden ser parcialmente o completamente automatizado, entonces será importante contar con un soporte de herramienta.

Definición 4 La evaluación de un modelo $F_0 - modelo\ M_0$ de S dará como resultado un formalismo descriptivo R_0, que como se señaló antes se podrá realizar con el uso de técnicas numérico-analíticas o por simulación.

La evaluación de este tipo de modelos es llama una $V_0 - evaluación$. Los resultados están presentados en un formalismo o dominio R_i, y pueden ser sujetos a ser procesados para refinarlos a un nivel más alto R_{i+1}.

Definición 5 Un $E_{i+1} - refinamiento$, es el resultado de una evaluación presentado en un dominio R_i. Estos refinamientos usualmente implican el uso de una herramienta, ya que se podrán automatizar.

Cuando se tiene un $F_{i-1} - modelo\ M_{i-1}$ de S, generalmente se quiere evaluar este modelo y obtener las medidas dadas al mismo nivel, es decir, que se necesitan los resultados a un nivel R_i. Se define una evaluación virtual V_j como el proceso de modelar subsecuentemente M_i (con $1 \leq i \leq j$) en un formalismo F_{i-1}, hasta obtener un F_0-$modelo\ M_0$, seguido de una evaluación V_0 y sus refinamientos sucesivos E_i hasta E_j.

1.5.7. Solución del modelo.

Una vez que ya se obtuvo el modelo estocástico, se tienen que obtener las medidas de interés. En el ambiente de un GMTF, se puede decir que obtener los resultados de los modelos puede ser más o menos complejos, del nivel más bajo del formalismo. Cabe mencionar que también las transformaciones de un nivel de modelado a otro también será parte del proceso de solución. Como se explicará más tarde, estas transformaciones pueden ser más complejas y tardadas que la solución del modelo de nivel más bajo, una vez que se generó.

En general se prefiere que en una herramienta de evaluación de desempeño exista la manera de solucionar diferentes modelos, con diferentes técnicas. Así se podrán hacer comparaciones al solucionar el mismo modelo con una técnica u con otra, al momento de expandir el modelo o modificarlo.

CAPÍTULO 2

Procesos estocásticos y cadenas de Markov.

Cuando se trata con problemas de toma de decisiones, usualmente surge la necesidad de tomar decisiones de fenómenos que tienen cierto grado de incertidumbre asociada. Esta incertidumbre es natural del fenómeno, y es posible tratarla de modo cuantitativo. Esto es posible ya que estos fenómenos pueden ocurrir con regularidad, solo con algunas variaciones, por esto se pueden llevar a describir mediante un modelo probabilístico.

2.1. Procesos estocásticos.

Definición 6 Un proceso estocástico es una colección indexada de variables aleatorias $\{X(t)\}$, donde el índice t toma valores de un conjunto $\mathcal{T}$ dado [1]. El conjunto $\mathcal{T}$, es el conjunto de enteros no negativos y $X(t)$ representa una característica de interés medible en el tiempo t. Así un proceso estocástico $X(1), X(2), X(3), \ldots, X(n)$ podría representar la colección de niveles de usuarios diarios en un sistema.

Usualmente cuando se analiza un proceso estocástico, tendrá una estructura parecida a la siguiente. En puntos específicos del tiempo con $t = 0, 1, \ldots$, el sistema se encuentra exactamente en un estado, de un conjunto finito $\mathcal{I}$ de éstos, excluyentes entre si, también etiquetados como $0, 1, \ldots, M$. Los puntos en el tiempo pueden encontrarse a intervalos que dependen del comportamiento del sistema en el que se encuentra el proceso estocástico. La representación matemática del sistema a analizar es la de un proceso estocástico $\{X(t)\}$, donde las variables aleatorias se observan en $t = 0, 1, 2, \ldots n$ y en donde cada variable aleatoria puede tomar el valor de cualquiera de los estados de M.

Usualmente cuando el espacio o conjunto de estados es $\mathcal{I} = \{0, 1, 2, ..M\}$, se dice que se trata con un proceso estocástico de estados discretos, o también llamado cadena. De manera similar, se puede tratar con espacios continuos y se llamará proceso estocástico de estado continuo. Análogamente en el caso que el conjunto $\mathcal{T}$ puede ser discreto o continuo, se le llamará proceso estocástico de tiempo discreto o continuo.

La función de densidad acumulada o función de distribución de una variable aleatoria X_t usualmente se llama la distribución de primer orden del proceso estocástico $\{X(t)|t \in \mathcal{T}\}$ y se denota como $F(x,t) = P\{X(t_1) \leq x\}$ con x en $\mathcal{I}$ y t en $\mathcal{T}$ y P la función de probabilidad. Esto se puede generalizar a la distribución de orden n del proceso estocástico como:

$$F(x,t) = P(X(t_1) \leq x_1, \ldots, X(t_n) \leq x_n\}, \tag{2.1}$$

donde x pertenece al conjunto de estados $\mathcal{I}^n$ y t está en $\mathcal{T}^n$.

Si todas las distribuciones de orden n(con $n \in \mathbb{N}^+$) son invariantes de los saltos de tiempo para todos los valores de x y t, entonces se dice que el proceso es estrictamente estacionario.

Definición 7 Se define un proceso estocástico como independiente cuando una de sus n distribuciones cumple la siguiente condición:

$$F(x,t) = \prod_{i-1}^{n} F(x_i, t_i) = \prod_{i-1}^{n} P\{X(t_i) \leq x_i\}. \tag{2.2}$$

Un ejemplo de proceso estocástico independiente es un proceso de renovación (*renewal process).* Un proceso de renovación $\{X_n|n = 1, 2, \ldots\}$, es un proceso estocástico de tiempo discreto, donde $X_1, X_2, \ldots$ son variables aleatorias independientes, distribuidos idénticamente y no negativas. En este tipo de procesos existe la total independencia entre estados consecutivos. Sin embargo, se asume en algunos casos, que existe algún tipo de dependencia entre estos estados.

2.2. Procesos de renovación.

Definición 8 Se define un proceso de renovación como un proceso estocástico de tiempo discreto $\{X_n|n = 1, 2, \ldots, \infty\}$, donde $X_1, X_2, \ldots$ son variables aleatorias independientes, idénticamente distribuidas y no negativas. Se asume que todas las variables aleatorias X_i están distribuidas como la variable aleatoria X con una función de distribución $F_X(x)$. Así, se define:

$$S_k = X_1 + X_2 + ... + X_k \tag{2.3}$$

que denota el tiempo, desde la instancia de tiempo inicial 0 en adelante, hasta la k ocurrencia de una renovación ($S_0 = 0$).

Los procesos estocásticos que describen flujos de llegadas de clientes en una estación de colas, usualmente se consideran que son procesos de renovación. Esto implica que el tiempo entre cada llegada se asume que es independiente e idénticamente distribuido. Así, existen muchas formas de construir distribuciones de procesos de llegada.

2.3. Cadenas de Markov.

La dependencia mínima posible en un proceso estocástico es en la cual, el siguiente estado a tomar, solo depende del estado actual del proceso estocástico, y no de los estados que se tomaron anteriormente. Esto se llama dependencia de primer orden o dependencia de Markov, que se define como:

Definición 9 Un proceso estocástico $\{X(t)|t \in \mathcal{T}\}$ es llamado proceso de Markov si para cualquier $t_0 < \ldots t_n < t_{n+1}$ de la distribución de $X(t_{n+1})$, dados los valores $X(t_0), \ldots, X(t_n)$, solo depende de $X(t_n)$, es decir:

$$P\{X(t_{n+1}) \leq x_{n+1}|X(t_0) = x_0, \ldots, X(t_n) = x_n\} = P\{X(t_{n+1}) \leq x_{n+1}|X(t_n) = x_n\} \tag{2.4}$$

Esta ecuación, es también llamada la propiedad de Markov. En otras palabras se refiere a que la probabilidad condicional de cualquier evento futuro, dado un evento pasado y el estado presente $X_t = i$, es independiente del evento pasado y solo depende del estado presente. Este tipo de procesos que son usados para la evaluación de desempeño son invariantes a los cambios de tiempo, esto es, que para cualquier $s < t$, y x, x_s, se tiene:

$$P\{X(t) \leq x|X(s) = x_s\} = P\{X(t-s) \leq x|X(0) = x_s\} \tag{2.5}$$

En este caso se habla de procesos de Markov con tiempo homogéneo. Cabe denotar que en un proceso o cadena de Markov, el tiempo durante el que se reside en un estado debe de estar descrito por variables aleatorias que tengan una distribución sin memoria. Como se verá más tarde, esto implica que el tiempo de residencia de un estado en una cadena de Markov de tiempo continuo necesita ser distribuido exponencialmente, mientras que en una de tiempo discreto la cadena deberá ser distribuida de manera geométrica. Si no se cumplen ninguna de estas distribuciones, entonces se habla de semi-procesos de Markov.

Las probabilidades condicionales $P = \{X_{t+1} = j|X_t\}$ de una cadena de Markov son llamadas probabilidades de transición (de un paso). Si para cada i y j,

$$P\{X_{t+1} = j|X_t = i\} = P\{X_1 = j|X_0 = i\}, \tag{2.6}$$

para todo $t = 1, 2, \ldots$,entonces las probabilidades de transición son llamadas estacionarias. Entonces, al tener probabilidades de transición estacionarias implica que éstas no cambian en el tiempo. La existencia de este tipo de probabilidades también implican que para cada i, j y $n(n = 0, 1, 2, \ldots)$,

$$P = \{X_{t+n} = j | X_t = i\} = P\{X_n = j | X_0 = i\} \tag{2.7}$$

para todo $t = 0, 1, \ldots$. Estas probabilidades son llamadas probabilidades de transición de $n-pasos(p_{ij}^n)$. Esto es la probabilidad condicional, de que el sistema estará en el estado j después de exactamente n pasos (unidades de tiempo), dado que inicia en el estado i en cualquier instante de tiempo t.

Para hacer el cómputo de estas probabilidades se emplean las ecuaciones de Chapman-Kolmogorov:

$$p_{ij}^{(n)} = \sum_{k=0}^{M} p_{ik}^{(m)} p_{kj}^{(n-m)} \tag{2.8}$$

para todo $i = 0, 1, \ldots, M,$

$j = 0, 1, \ldots, M,$

y cualquier $m = 1, 2, \ldots, n-1,$

$n = m+1, m+2, \ldots$

Así, $p_{ik}^{(m)} p_{kj}^{(n-m)}$ es la probabilidad condicional que, dado un estado inicial i, el proceso va del estado k después de m pasos y después al estado j en $n-m$ pasos. Al sumar estas probabilidades para todos los posibles k resultará p_{ij}^n.

Existe una propiedad de las cadenas de Markov que aparece después de un determinado tiempo. Esta propiedad es una probabilidad limitante que señala que el sistema estará en un estado j después de un número grande de transiciones, y que esta probabilidad es independiente del estado inicial. Esto se define como:

Definición 10 Para cualquier cadena de Markov irreducible y ergódica (ver Glosario), existe el $\lim_{n\to\infty} p_{ij}^{(n)}$ y es independiente de i. Además,

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j > 0 \tag{2.9}$$

donde π_j únicamente satisface las siguientes ecuaciones de estado estable

$$\begin{aligned} &\pi_j = \sum_{i=0}^{M} \pi_i p_{ij}, \text{para } j = 0, 1, ..., M, \\ &\sum_{j=0}^{M} \pi_j = 1 \end{aligned} \tag{2.10}$$

Donde π_j son llamadas las probabilidades de estado estable de la cadena de Markov. Esta probabilidad significa que la probabilidad de encontrar un proceso en algún estado j, después de un número

grande de transiciones tiende al valor π_j, que es independiente de la distribución de probabilidad del estado inicial. Esta probabilidad no implica que el proceso se establezca en un estado. También estas probabilidades pueden ser interpretadas como probabilidades estacionarias (no confundir con probabilidades de transición estacionarias), si la probabilidad de estar en el estado j está dada por π_j para todo j, entonces la probabilidad de que el proceso esté en el estado j en el tiempo $n = 1, 2, \ldots$ también estará dado por π_j.

2.4. Cadenas de Markov de tiempo continuo.

Durante este capítulo se habló de cadenas de Markov asumiendo que eran de tiempo discreto, es decir el valor de t tomaba valores en $0, 1, 2, 3, \ldots, \infty$. Esto está bien para algunos problemas, pero en algunos casos (como en teoría de colas) no es una buena aproximación. Se requiere que t esté definido de manera continua (es decir que $t \in \mathbb{R}^+$), que se denotará como t'. La cadena de Markov estará definida de la misma manera que en las secciones anteriores, solo que ahora estará definida bajo un tiempo t'continuo.

2.4.1. Variables aleatorias claves.

En el análisis de las cadenas de Markov con tiempo continuo, existe un conjunto de variables aleatorias clave.

Cada que un proceso entra en un estado i, la cantidad de tiempo que está en ese estado antes de moverse a otro está definido por una variable aleatoria $\mathcal{T}_i$, donde $i = 0, 1, \ldots, M$.

Supongamos que el proceso entra a un estado $i$en un tiempo $t' = s$. Entonces, para cualquier cantidad de tiempo $t > 0$, nótese que $\mathcal{T}_i > t$si y solo si $X(t') = i$para todo t'en el intervalo $s \leq t' \leq s + t$.Entonces, por la propiedad Markoviana implica que:

$$P\{\mathcal{T}_i > t + s | \mathcal{T}_i > s\} = P\{\mathcal{T}_i > t\} \tag{2.11}$$

Esta es una propiedad inusual en alguna distribución de probabilidad. Esto básicamente dice que la distribución de probabilidad del tiempo restante hasta que el proceso sale de un determinado estado es siempre la misma, sin importar cuanto tiempo el proceso haya estado en ese estado. Es decir, la variable aleatoria no tiene memoria. Solo existe una probabilidad de distribución (continua) que posee esta propiedad, que es la distribución exponencial. Esta tiene un solo parámetro q, donde la media es $1/q$ y la función de distribución acumulada es

$$P\{\mathcal{T}_i \leq t\} = 1 - e^{-qt}, \text{para } t \geq 0 \tag{2.12}$$

Análogamente a las probabilidades de transición de un paso, en las cadenas de Markov de tiempo continuo existen intensidades de transición, definidas como:

Definición 11 Sea $p_{ij}(t)$ la función de probabilidad de transición en tiempo continuo, p_{ij} la probabilidad con la que un proceso se mueve de un estado j a uno i y q_i el parámetro de la distribución exponencial para $\mathcal{T}_i$, las intensidades de transición son:

$$\begin{aligned} q_i &= \frac{d}{dt}p_{ii}(0) = \lim_{t\to 0}\frac{1-p_{ii}(t)}{t}, \text{para } i = 0, 1, 2..., M \\ &\text{y} \\ q_{ij} &= \frac{d}{dt}p_{ij}(0) = \lim_{t\to 0}\frac{p_{ij}(t)}{t} = q_i p_{ij}, \text{para todo } j \neq i \end{aligned} \tag{2.13}$$

La interpretación que se les da a q_i y q_{ij} es que son tasas de transición. En particular, q_i es la tasa de transición fuera del estado i, es decir q_i es el número esperado de veces que el proceso se mueve fuera del estado i por unidad de tiempo empleado en el estado i. De manera similar, q_{ij} es el número de veces en que el proceso se mueve del estado i al estado j por unidad de tiempo empleado en i. Entonces

$$q_i = \sum_{j\neq i}^{q_{ij}} \tag{2.14}$$

Como q_i es el parámetro de la distribución exponencial para $\mathcal{T}_i$, cada q_{ij} es el parámetro de una distribución exponencial de una variable aleatoria $\mathcal{T}_{ij}$. Cada vez que el proceso entra en un estado i, esta variable aleatoria define el tiempo empleado en el estado i antes de que ocurra una transición al estado j (con $i \neq j$). El tiempo empleado en el estado i hasta que una transición ocurra ($\mathcal{T}_i$) es el mínimo (sobre $i \neq j$) de $\mathcal{T}_{ij}$. Cuando la transición ocurre, la probabilidad de que es hacía el estado j es $p_{ij} = q_{ij}/q_i$.

Además las cadenas de Markov de tiempo continuo también tienen la probabilidad limitante llamada probabilidad de estado estable descrita en la sección 3.1.4. Por último se establece que este tipo de cadenas también satisfacen las ecuaciones de Chapman-Kolmogorov.

CAPÍTULO 3

Sistemas de colas.

3.1. Introducción a la teoría de colas.

3.1.1. Teoría de colas.

La teoría de colas es el estudio de la espera en cualquier sistema que involucre una cola. Esta teoría usa los modelos de colas para representar varios tipos de sistemas de colas. Las fórmulas para cada modelo indican como se desempeña cada sistema de colas, incluyendo el promedio de la cantidad de espera que ocurrirá y otra variedad de mediciones.

Es por esto que los modelos de colas son de grana ayuda para determinar como manejar estos sistemas de la forma más eficaz.

3.1.2. Modelos de colas.

Un tipo de modelos estocásticos son los modelos de colas, que describen el fenómeno de encolamiento que ocurre en la realidad. El encolamiento se puede observar en muchas partes, en una cola del supermercado, en los cajeros automáticos, en los aeropuertos, parques de diversiones, etc. El encolamiento ocurre por que varía el tiempo en el que un cliente llega. También ocurrirá por que el tiempo de servicio varía dependiendo de cada cliente, y no solo de él, sino del que atiende también. A manera de regla general es que a más variabilidad, se necesitará más encolar. A esto también está asociada la espera. A una cola más larga, le corresponderá un tiempo de espera más largo de lo normal, a cada usuario.

Las colas son útiles tanto en sistemas técnicos, como lo son los sistemas de comunicaciones así como en sistemas de logística y líneas de manufactura. Incluso se utilizan técnicas similares para analizar y optimizar las operaciones, en todos estos sistemas.

En muchos de estos sistemas lo que se puede observar es el uso de componentes compartidos, donde muchos usuarios requieren del uso de éstos. Sin embargo, el uso no se puede dar siempre al mismo tiempo, así cada usuario tendrá que esperar para usar el componente deseado. Cabe mencionar que esto no es una debilidad de un sistema, sino que ofrece diferentes ventajas que se verán más adelante, además, no es costoso de implementar.

Entonces se requiere modelar todo tipo de recursos compartidos como entidades que proveen un servicio, cada una, precedida de una cola de espera.

3.1.3. Estructura básica de modelos de colas.

El proceso básico de un modelo de colas usualmente es el mismo. Una *fuente de entrada* genera en el tiempo un *cliente* que requiere un servicio. Estos clientes entran en un *sistema de colas* y se unen a una *cola*. Por medio de una regla específica, en determinados tiempos, se selecciona un miembro de la cola para servirle, esta regla será llamada la *disciplina de la cola*. El servicio requerido es realizado por el cliente por un *mecanismo de servicio*, después del cual, el cliente abandona el sistema de colas.

Una característica de la fuente de entrada es su tamaño, que determina el número total de clientes que pueden requerir el servicio de vez en cuando, es decir el número total de clientes potenciales. Esta población de la cuál llegan los clientes es la *población de llamada*, que su tamaño podrá ser finita o infinita. Se estudia el caso de un número grande finito, para así no tomarlo como verdaderamente infinita. El caso finito es más difícil analíticamente ya que el número de clientes en el sistema de colas afectará el número potencial de clientes afuera del sistema. Sin embargo, esta suposición finita se deberá hacer si la tasa a la cual la fuente de entrada genera clientes, es afectada significativamente por el número de clientes en el sistema.

Se asume que generalmente los clientes llegarán en un tiempo, conforme a una distribución de probabilidad [8], y así se podrá hacer la suposición que el tamaño de la fuente de entrada es infinito. Este tiempo es llamado el *tiempo entre llegadas*.

Una cola es donde un cliente espera antes de ser servido. La característica más importante es el número de clientes permitidos que puede contener. Este tamaño es finito o infinito; generalmente se asume que el tamaño de la cola es infinito.

La disciplina de la cola se refiere a la manera en que un cliente puede ser seleccionado para ser

servido. Usualmente se utiliza el esquema Primero en Llegar, Primero en ser Servido (FCFS, por sus siglas en inglés *First Come, First Served)*. Aunque, se pueden aplicar diferente esquemas como al azar, por prioridades, en un orden específico, etc.

El mecanismo de servicio consiste en una o más instalaciones de servicio, cada una podrá contener uno o más canales paralelos de servicio, conocidos como servidores. En estas entidades de servicio o servidores, se proporcionará el servicio requerido a cada cliente de la cola.

El tiempo transcurrido del comienzo del servicio de un cliente, hasta que finaliza, es llamado tiempo de servicio. Todo modelo de un sistema de colas, deberá especificar la distribución de probabilidad de los tiempos de servicio para cada servidor, que usualmente se utiliza la misma distribución para todos.

3.1.4. Funcionamiento.

El funcionamiento básico de un sistema de colas es el siguiente: se tiene una sola línea de espera (que puede estar vacía a veces) que se forma en frente de una instalación de servicio, en donde existe uno o más servidores. Cada cliente generado por una fuente de entrada es atendido por uno de los servidores, tal vez después de esperar en la cola de espera (o línea de espera). Esto se muestra en la siguiente figura:

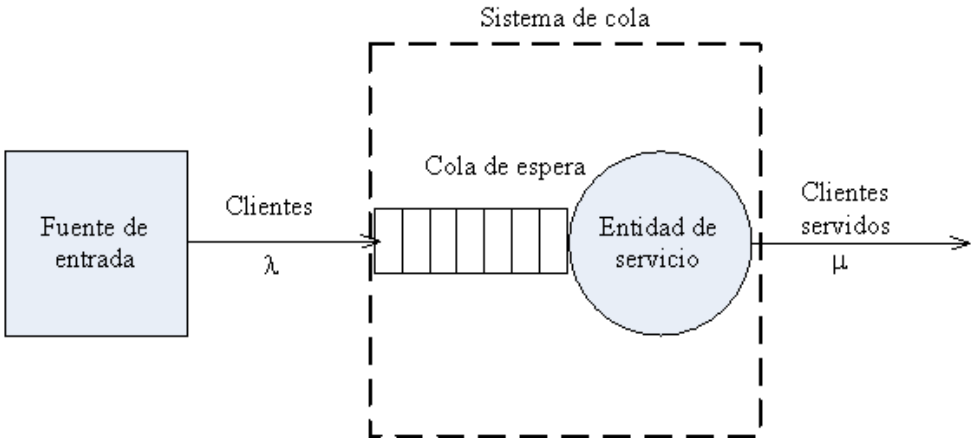


Figura 3.1: Funcionamiento básico de una cola.

Cabe notar que esta ilustración es solo una manera de mostrar el funcionamiento, ya que el funcionamiento y la estructura no están ligados. La fuente de entrada puede ser por ejemplo clientes que llegan a una sala de emergencia. Un servidor puede que no sea un individuo o una computadora, puede ser un grupo de personas, una máquina electrónica, etc. De igual modo, los clientes podrían no ser

personas, podrían ser piezas que esperan por una operación de una máquina, automóviles, etc. Finalmente no es necesario que la cola esté en una línea, los clientes podrían estar en un área distribuidos y ahí esperar a ser atendidos por el servidor. Todos estos factores no influyen en el funcionamiento, ya que el estudio de las variables: número de clientes esperando, tiempo de espera, tiempo de servicio, etc., no cambia.

3.1.5. Notación.

A continuación se presenta un conjunto de variables que se utilizarán como notación para el resto de esta tesis, todas relativas a la teoría de colas y su estudio:

- P_n = Probabilidad de que estén exactamente n clientes en el sistema.
- N = Estado del sistema, que es el número de clientes promedio en el sistema de cola.
- N_q = Longitud de la cola, indica el número de clientes promedio esperando en la cola.
- s = Número de servidores del sistema.
- λ_n = Tasa de llegadas promedio (número esperado de llegadas por unidad de tiempo) de nuevos clientes cuando n clientes están en el sistema.
- μ_n = Tasa de servicio promedio (número esperado de clientes atendidos por unidad de tiempo) de todo el sistema cuando n clientes están en el sistema.

Adicionalmente se definen las siguientes variables. Cuando λ_n es una constante para todo n, esta constante se denota por λ. Cuando la tasa de servicio promedio por cada servidor ocupado es una constante para todo $n \geq 1$, se denota por μ. Con estas condiciones se define $1/\lambda$ y $1/\mu$ como el tiempo esperado entre llegadas y el tiempo esperado de servicio, respectivamente. Finalmente se define $\rho = \lambda/(s\mu)$ como el factor de utilización para las instalaciones de servicio.

Se dice que después de un largo periodo de tiempo el sistema de cola llega a un estado estable. Este estado se alcanza cuando el estado del sistema es independiente del estado inicial y del tiempo transcurrido. Se establece que cualquier estado puede llegar a su condición de estado estable si $\rho = \mu/\lambda < 1$. La teoría de colas tiende a enfocarse ampliamente en esta condición de estado estable, una de las razones es por que cuando sí depende del caso inicial, es más difícil de estudiar analíticamente. Cuando el sistema se encuentra en estado estable se pueden estudiar las siguientes variables:

- $E[N]$ = Número esperado de clientes en el sistema. = $\sum_{n=0}^{\infty} nP_n$

- $E[N_q]$= Número esperado de clientes esperando en la cola. $= \sum_{n=s}^{\infty} (n-s)P_n$
- $E[R]$ = Tiempo de respuesta esperado, es decir el tiempo que espera un cliente a ser totalmente atendido. $(E[W] + E[W_q])$
- $E[W]$ = Tiempo de espera de la tarea en el ente de servicio.
- $E[W_q]$ = Tiempo de espera promedio de un cliente en la cola.

3.1.6. Relaciones entre variables y la Ley de Little.

En esta sección se establecerá la fórmula más general en la evaluación de análisis de desempeño, la ley de Little [9]. Esta ley relaciona el número promedio de clientes en un sistema de colas con el número promedio de llegadas por unidad de tiempo y el tiempo promedio que un cliente emplea en un sistema de colas.

Sea λ el promedio de llegadas por unidad de tiempo (también llamada la tasa de llegada o intensidad de llegada). Se asume que la disciplina de la cola será *FCFS*. Como lo denota el funcionamiento de la cola, el cliente pasará inmediatamente a ser atendido o esperará en una cola hasta pasar a ser atendido. Cabe notar que para este análisis no se toma en cuenta la distribución de probabilidad asignada de la fuente de entrada o del servidor, solo sus promedios[10]. Se define el tiempo promedio de espera en el sistema como $E[R]$ (tiempo de respuesta), y el número de clientes en el sistema como $E[N]$.

Se supone que se selecciona un cliente en particular, cuando el cliente entra al sistema y se marca con un tiempo t_i.Cuando sale del sistema se marca con un tiempo t_f. La diferencia entre $t_f - t_i$ será en promedio igual al valor de $E[R]$. Sin embargo, en el momento que el cliente deje el sistema, se sabe que mientras ese cliente pasó por todo el sistema de cola, otros clientes llegaron. Ya que en promedio pasaron $E[R]$ unidades de tiempo entre la llegada y salida del cliente seleccionado, se observa que en promedio han llegado $\lambda \times E[R]$ clientes después de este cliente. No obstante, este número, será igual al valor de $E[N]$. Esto dará por hecho de que cada cliente puede ser un cliente seleccionado, entonces, el producto $\lambda E[R]$ siempre será igual al número de clientes dejados atrás en el sistema por cualquier cliente seleccionado, es decir, esto puede ser interpretado correctamente como el número promedio de clientes en el sistema. Entonces se tiene:

$$E[N] = \lambda E[R] \tag{3.1}$$

Se puede asumir que la tasa de salida del sistema será igual a la tasa de entrada del sistema, esto ocurrirá siempre y cuando el sistema no esté sobrecargado, de otra manera se podrían perder algunos clientes.

Ahora se supone que se selecciona un cliente antes de que entre a la cola de espera del sistema. El tiempo de espera en la cola será $E[W]$. Al aplicar de nuevo la ley de Little, el número promedio de clientes en la cola $E[N_q]$, deberá ser igual al producto del tiempo de espera y la tasa de entrada (se asume que no se perderán clientes en el sistema), entonces se obtiene:

$$E[N_q] = \lambda E[W] \tag{3.2}$$

Finalmente se puede aplicar la ley al nivel del servidor y se obtiene que:

$$E[N_s] = \lambda E[S] \tag{3.3}$$

Donde N_s es el número esperado de clientes en el servidor y $E[S]$ es el tiempo promedio que el cliente tarda en el servidor.

De regreso a la vista de todo el sistema de cola, se puede obtener que el promedio de clientes dentro del sistema será igual a la suma de los clientes en todas las partes, es decir, $E[N] = E[N_q] + E[N_s]$. Cuando se aplican las derivaciones anteriores se obtiene el valor esperado:

$$E[N] = E[N_q] + E[N_s] = \lambda E[W] + \lambda E[S] = \lambda \left(E[W] + E[S]\right) = \lambda E[R] \tag{3.4}$$

3.1.7. Proceso de nacimiento y muerte.

Conocido en inglés como *the birth-and-death process*, este proceso asume que la entrada (llegada de clientes) y la salida (clientes atendidos) ocurren acorde a un proceso de nacimiento y muerte. Este proceso proviene de la teoría de probabilidad y tiene varias aplicaciones en diferentes áreas. En el caso de la teoría de colas, el término nacimiento se refiere a la llegada de un nuevo cliente y la muerte a la salida del sistema de este cliente ya atendido. El estado del sistema en cualquier tiempo t es el número de clientes $N(t)$ en el sistema en ese tiempo. El proceso de nacimiento y muerte describe probabilísticamente como $N(t)$ cambia conforme t se incrementa. Es decir, el proceso dice que los nacimientos y muertes individuales ocurren aleatoriamente, cuando su tasa promedio depende solo del estado del sistema actual. Más aún, se hacen las siguientes suposiciones del proceso:

- **Suposición 1.** Dada $N(t) = n$, la distribución de probabilidad actual del tiempo restante hasta el siguiente nacimiento (llegada) es exponencial con parámetro λ_n(con $n = 0, 1, 2\ldots$).
- **Suposición 2.** Dada $N(t) = n$, la distribución de probabilidad actual del tiempo restante hasta la próxima muerte (atención completada) es exponencial con parámetro μ_n(con $n = 1, 2\ldots$).
- **Suposición 3.** La variable aleatoria del punto 1 y la variable aleatoria del punto 2 son mutuamente independientes. La siguiente transición en el estado del proceso puede ser:

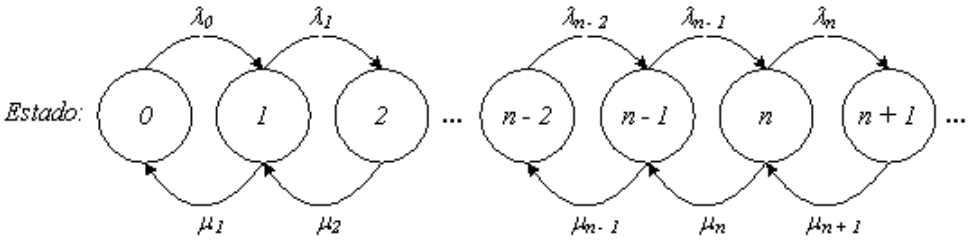


Figura 3.2: Diagrama de estados de un proceso de nacimiento muerte.

- $n \to n+1$ (un solo nacimiento), o
- $n \to n-1$ (una sola muerte).

Debido a esto, el proceso de nacimiento y muerte es un tipo especial de cadena de Markov de tiempo continuo. Así, los modelos de colas que pueden ser representados por una cadena de Markov de tiempo continuo, son por mucho más manejables analíticamente que cualquier otra.

Por la propiedad 4 del Apéndice A se implica que las tasas promedio son λ_n y μ_n, En Figura 3.2 se muestra el diagrama de estados de un proceso de nacimiento muerte, además se pueden resumir las 3 suposiciones pasadas, ya que se muestran las posibles transiciones del sistema y para cada flecha se da una tasa promedio para la transición.

Ya que los eventos que suceden en el sistema se alternan entre llegadas y salidas, se concluye en el siguiente principio:

Definición 12 Tasa de Entrada = Tasa de Salida. Para cualquier estado del sistema $n(n = 0, 1, 2, \ldots)$ la tasa de entrada promedio = tasa de salida promedio.

La ecuación de este principio también se llama ecuación de balance para el estado n. Después de construir las ecuaciones de balance para todos los estados en términos de la incógnita P_n, se puede resolver el sistema de ecuaciones para encontrar estas probabilidades.

Para ilustrar esta ecuación considere el estado 0. El proceso entra a este estado solo por el estado 1. Entonces, la probabilidad de estado estable de estar en el estado 1 (P_1) representa la proporción de tiempo en la que es posible que el sistema entre en el estado 0. Dado que el proceso está en el estado 1, la tasa promedio de entrar al estado 0 es μ_1, es decir para cada unidad de tiempo acumulada que el proceso emplee en el estado 1, el número de veces esperado de que el sistema deje el estado 1 para entrar al estado 0 es μ_1. De cualquier otro estado, esta tasa promedio será 0. entonces, la tasa

promedio resultante a la cuál el proceso deja su estado actual para entrar al estado 0 (la tasa de entrada promedio) es:

$$\mu_1 P_1 + 0(1 - P_1) = \mu_1 P_1 \tag{3.5}$$

Con el mismo razonamiento anterior, la tasa de salida promedio deberá ser $\lambda_0 P_0$, entonces la ecuación de balance para el estado 0 será:

$$\mu_1 P_1 = \lambda_0 P_0 \tag{3.6}$$

Para cualquier otro estado existen dos posibles transiciones ambas para entrar y salir del estado. Por consecuente, cada lado de las ecuaciones de balance para estos estados representa la suma de las tasas promedio para las dos transiciones involucradas. De lo contrario, se puede hacer el mismo razonamiento que para el estado 0.

En la tabla 3.1 se presentan las ecuaciones de balance para el proceso, nótese que la primera ecuación contiene dos variables para las cuáles hay que resolver, la segunda contiene tres y la siguiente una más para todos los casos, es decir, siempre hay una variable extra. El procedimiento para obtener el resultado de estas ecuaciones, es solucionarlas en términos de una de las variables, que la más conveniente es P_0. Entonces para resolver la primera ecuación se resuelve P_1 en términos de P_0,después se usarán estos resultados para resolver P_2 en términos de P_0 y de manera similar se continua para las demás ecuaciones. Al final, el requerimiento de que la suma de todas las probabilidades sea igual a 1 se puede usar para evaluar P_0.

Tabla 3.1: Ecuaciones de balance para el proceso de nacimiento y muerte.

Estado	**Tasa de entrada = Tasa de salida**
0	$\mu_1 P_1 = \lambda_0 P_0$
1	$\lambda_0 P_0 + \mu_2 P_2 = (\lambda_1 + \mu_1) P_1$
2	$\lambda_1 P_1 + \mu_3 P_3 = (\lambda_2 + \mu_2) P_2$
$\vdots$	$\vdots$
$n-1$	$\lambda_{n-2} P_{n-2} + \mu_n P_n = \left(\lambda_{n-1} + \mu_{n-1}\right) P_{n-1}$
n	$\lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} = (\lambda_n + \mu_n) P_n$
$\vdots$	$\vdots$

Cuando se aplica el procedimiento descrito se obtienen los siguientes resultados:

Estado:

$$
\begin{array}{lll}
0: & P_1 & = \frac{\lambda_0}{\mu_1} P_0 \\
1: & P_2 & = \frac{\lambda_1}{\mu_2} P_1 + \frac{1}{\mu_2}\left(\mu_1 P_1 - \lambda_0 P_0\right) = \frac{\lambda_1}{\mu_2} P_1 = \frac{\lambda_1 \lambda_0}{\mu_2 \mu_1} P_0 \\
2: & P_3 & = \frac{\lambda_2}{\mu_3} P_2 + \frac{1}{\mu_3}\left(\mu_2 P_2 - \lambda_1 P_1\right) = \frac{\lambda_2}{\mu_3} P_2 = \frac{\lambda_2 \lambda_1 \lambda_0}{\mu_3 \mu_2 \mu_1} P_0 \\
\vdots & \vdots & \\
n-1 & P_n & = \frac{\lambda_{n-1}}{\mu_n} P_{n-1} + \frac{1}{\mu_n}\left(\mu_{n-1} P_{n-1} - \lambda_{n-2} P_{n-2}\right) = \frac{\lambda_{n-1}}{\mu_n} P_{n-1} = \frac{\lambda_{n-1}\lambda_{n-2}.\cdots\lambda_0}{\mu_n \mu_{n-1}\cdots\mu_1} P_0 \\
n & P_{n+1} & = \frac{\lambda_n}{\mu_{n+1}} P_n + \frac{1}{\mu_{n+1}}\left(\mu_n P_n - \lambda_{n-1} P_{n-1}\right) = \frac{\lambda_n}{\mu_{n+1}} P_n = \frac{\lambda_n \lambda_{n-1}.\cdots\lambda_0}{\mu_{n+1}\mu_n\cdots\mu_1} P_0 \\
\vdots & \vdots &
\end{array}
$$

Para simplificar la notación se define:

$$C_n == \frac{\lambda_{n-1}\lambda_{n-2}.\cdots\lambda_0}{\mu_n \mu_{n-1}\cdots\mu_1}, \text{ para } n = 1, 2, ..., \tag{3.7}$$

Donde $C_n = 1$ cuando $n = 0$. Entonces, las probabilidades de estado estable son:

$$P_n = C_n P_0 \text{ para } n = 0, 1, 2, ... \tag{3.8}$$

El requerimiento de que la suma de las probabilidades sea igual a 1, implica que:

$$\sum_{n=0}^{\infty} P_n = \sum_{n=0}^{\infty} C_n P_0 = 1 \tag{3.9}$$

Por lo tanto:

$$P_0 = \left(\sum_{n=0}^{\infty} C_n\right)^{-1} \tag{3.10}$$

Entonces cuando un modelo de colas está basado en un proceso de nacimiento y muerte, tal que el estado del sistema n representa el número de clientes en el sistema de cola, las medidas claves de desempeño del sistema pueden ser obtenidas inmediatamente.

3.2. La propiedad PASTA.

Un resultado reconocido y usualmente aplicado a la teoría de colas es la propiedad de PASTA (*Poisson Arrivals See Time Averages*).

El teorema de la propiedad PASTA dice:

La distribución de los clientes en una estación de colas en el momento que un cliente nuevo de un proceso de llegada de Poisson se presenta, es la igual a la distribución a largo plazo o del estado estable.

Se dice que cada cliente "ve" la cola a la que llega como "si estuviera en equilibrio". La demostración de esta propiedad es relativamente simple. Considere un sistema de cola en el cuál el número de clientes presente es representado por un proceso estocástico $(X_t, t \geq 0)$. Se define el evento "existe al menos una llegada en ésta estación de cola en el intervalo $(t-h, t]$". Ya que las llegadas forman un proceso de Poisson (ver Glosario *Proceso de Poisson*) homogéneo, la probabilidad de este evento es igual a la probabilidad de que existan una llegada en el intervalo $(0, h)$ que es igual a $P\{N(h) \geq 1\}$, donde $N(t)$ es el proceso de conteo de procesos de renovación. Para procesos que no son de Poisson, este cambio al origen no es válido. Ya que los tiempos entre llegadas no tienen memoria, la probabilidad definida es independiente de la historia pasada del proceso de llegada y del estado de la estación de cola: $P\{N(h) \geq 1 | X_{t-h} = i\} = P\{N(h) \geq 1\}$ así que $P\{N(h) \geq 1 \cup X_{t-h} = i\} = P\{N(h) \geq 1\} P\{X_{t-h} = i\}$. A partir de esto se puede concluir también que:

$$P\{X_{t-h} = i | N(h) \geq 1\} = P\{X_{t-h} = i\} \tag{3.11}$$

Si se toma el límite de $h \to 0$, el lado izquierdo de esta igualdad simplemente expresa la probabilidad de que una llega en el momento t llega a una cola con i clientes en ella. Ésta probabilidad entonces es igual a la probabilidad de que la cola en el tiempo t tenga i clientes en ella, independientemente de cualquier llegada, por lo tanto es la probabilidad de estado estable de tener i clientes en la cola. Así se puede concluir que una llegada de Poisson actúa como un observador aleatorio y ve la cola como si estuviera en equilibrio.

3.3. Modelos de colas basados en procesos de nacimiento y muerte.

3.3.1. Modelo M | M | 1

Con la notación de Kendall descrita anteriormente (ver sección 1.5.4) este modelo es la representación de un sistema de colas que asume que el tiempo entre llegadas es independiente e idénticamente distribuido conforme a una distribución exponencial, así como otra distribución exponencial para el tiempo de servicio y que el número de servidores será uno. Por ello, este modelo es un caso especial de un proceso de nacimiento y muerte donde la tasa de llegada promedio y la tasa de servicio promedio por servidor ocupado del sistema son constantes (λ y μ respectivamente) cualquiera que sea el estado

del sistema. Cuando el sistema tiene un solo servidor ($s = 1$), la implicación es que los parámetros para el proceso de nacimiento y muerte son $\lambda_n = \lambda$ y $\mu_n = \mu$ (con $n = 0, 1, 2, 3, \dots$).

Sin embargo, cuando el sistema tiene múltiples servidores ($s > 1$) μ_n no puede ser expresado tan fácilmente. Se debe recordar que μ_n representa la tasa promedio de servicio para todo el sistema de colas, es decir la tasa promedio a la cual se completan servicios y así los clientes dejan el sistema, donde n son los clientes que están en el sistema actualmente. Conforme a la Propiedad 4 del Apéndice A de la distribución exponencial, la tasa de servicio promedio por servidor ocupado es μ, la tasa promedio para todo el sistema por n servidores ocupado deberá ser $n\mu$. Entonces $\mu_n = n\mu$ cuando $n \leq s$, como para $\mu_n = s\mu$ cuando $n \geq s$ para que todos los servidores s estén ocupados.

Cuando la tasa de servicio promedio máximo $s\mu$ excede la tasa de llegada promedio λ, esto es cuando la ocupación $\rho < 1$, un sistema de colas que se acople a este modelo eventualmente llegará a una condición de estado estable.

3.3.2. Resultados para el modelo M | M | 1.

Para $s = 1$, los factores C_n para el proceso de nacimiento y muerte se reduce a

$$C_n = \left(\frac{\lambda}{\mu}\right)^n = \rho^n, \text{para } n = 0, 1, 2, ... \tag{3.12}$$

Esto implica,

$$P_n = \rho^n P_0, \text{para } n = 0, 1, 2, ..., \tag{3.13}$$

donde

$$P_0 = \left(\sum_{n=0}^{\infty} \rho^n\right)^{-1} = \left(\frac{1}{1-\rho}\right)^{-1} = 1 - \rho \tag{3.14}$$

Entonces,

$$P_n = (1-\rho)\, p^n, \text{para } n = 0, 1, 2, ... \tag{3.15}$$

En consecuencia para obtener las medidas relevantes, se utiliza la teoría de las series geométricas [11] y se obtiene que:

$$\begin{aligned} E[N] = \sum\nolimits_{n=0}^{\infty} nP_n = \sum_{n=0}^{\infty} n\,(1-\rho)\,\rho^n = (1-\rho)\,\rho \sum_{n=0}^{\infty} \frac{d}{d\rho}\,(\rho^n) = \\ (1-\rho)\,\rho \frac{d}{d\rho}\left(\sum_{n=0}^{\infty} \rho^n\right) = (1-\rho)\,\rho \frac{d}{d\rho}\left(\frac{1}{1-\rho}\right) = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda} \end{aligned} \tag{3.16}$$

De manera similar para,

$$E[N_q] = \sum_{n=1}^{\infty} (n-1)\, P_n = E[N] - 1\,(1 - P_0) = \frac{\lambda^2}{\mu\,(\mu - \lambda)} \tag{3.17}$$

Cuando $\lambda \geq \mu$, y la tasa de llegada promedio excede la tasa de servicio promedio, la solución anterior no funciona (ya que la suma para calcular P_0 diverge). Para este caso, la cola crecerá sin límite, conocido como *explotar*. Si el sistema de cola inicia su operación sin clientes presentes, el servidor puede funcionar al atender a los clientes que lleguen en un corto periodo de tiempo, pero a largo plazo será imposible.

Ahora bien, si se regresa a la suposición que $\lambda < \mu$, se puede derivar la distribución de probabilidad del tiempo de espera en el sistema (que incluye el tiempo de servicio) $E[R]$ para una llegada aleatoria cuando la disciplina de la cola es FCFS. Si al llegar se encuentra con n clientes aún en el sistema, entonces el que llega tendrá que esperar por $n+1$ tiempos de servicio exponenciales, que incluye el suyo. Entonces, se definen $T_1, T_2, \ldots$ como las variables aleatorias independientes de tiempo de servicio con una distribución exponencial con parámetro μ, se define

$$S_{n+1} = T_1 + T_2 + ... + T_{n+1}, \text{para } n = 0, 1, 2, ... \tag{3.18}$$

tal que S_{n+1} representa el tiempo condicional de espera dado que n clientes están ya en el sistema. Como se verá más adelante S_{n+1} se dice que tiene un distribución de Erlang, ya que la probabilidad que la llegada aleatoria encuentre n clientes en el sistema es P_n, y por consiguiente

$$P\{E[R] > t\} = \sum_{n=0}^{\infty} P_n P\{S_{n+1} > t\}, \tag{3.19}$$

que se reduce después de cierta manipulación a

$$P\{E[R] > t\} = e^{-\mu(1-\rho)t}, \tag{3.20}$$

La conclusión es que $E[R]$ tiene una distribución exponencial con parámetro $\mu(1-\rho)$. Entonces se obtiene que,

$$E[R] = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu - \lambda} \tag{3.21}$$

Estos resultados incluyen el tiempo de servicio en el tiempo de espera. En algunos contextos, el tiempo de espera más relevante es solo hasta que el servicio inicia. Entonces, se considera el tiempo de espera en la cola $E[W]$ para una llegada aleatoria cuando la disciplina de la cola es FCFS. Si esta llegada no encuentra clientes en el sistema, entonces la llegada será atendida inmediatamente, tal que

$$P\{E[W] = 0\} = P_0 = 1 - \rho \tag{3.22}$$

De manera contraria si encuentra clientes, es decir cuando $n > 0$, entonces la llegada tiene que esperar durante n tiempos de servicio exponenciales hasta que su servicio inicie, entonces:

$$\begin{aligned} P\{E[W] > t\} &= \sum_{n=1}^{\infty} P_n P\{S_n > t\} = \sum_{n=1}^{\infty} (1-\rho)\rho^n P\{S_n > t\} \\ &= \rho \sum_{n=0}^{\infty} P_n P\{S_{n+1} > t\} = \rho P\{W > t\} = \rho e^{-\mu(1-\rho)t}, \text{para } t \geq 0 \end{aligned} \tag{3.23}$$

De la derivación del promedio de la distribución de $E[W]$ (o al aplicar cualquiera de $E[N_q] = \lambda E[W]$ o $E[W] = E[R] - 1/\mu$,

$$E[W] = \frac{\lambda}{\mu\,(\mu - \lambda)} \tag{3.24}$$

3.3.3. Modelo M | M | 1 | K

Este modelo indica que el sistema de colas tiene una cola de espera finita, es decir, que solo se permite un especificado número K de clientes en la cola. Cualquier cliente que trate de entrar al sistema mientras la cola esté llena, es rechazado a entrar en el sistema. Desde el punto de vista de un proceso de nacimiento y muerte, la tasa de entrada promedio del sistema se vuelve cero en esos momentos. Entonces, la modificación necesaria para introducir una cola finita es cambiar λ_n a:

$$\lambda_n = \begin{cases} \lambda \text{ para } n = 0, 1, 2, ..., K-1 \\ \\ 0 \text{ para } n \geq K. \end{cases} \tag{3.25}$$

Ya que para algunos valores de $n, \lambda_n = 0$, un sistema de cola de este tipo siempre llegará eventualmente a su estado estable, incluso cuando $\rho = \lambda/s\mu \geq 1$.

La interpretación que se le puede dar a este modelo es que solo se tiene un cuarto de espera reducido. También se puede interpretar que los clientes que llegan se retiran y buscarán el servicio en algún otro lado si encuentran muchos clientes más delante de ellos, ya que no pueden esperar tanto. Este tipo de frustración del cliente es muy común en los sistemas de servicio comerciales.

3.3.4. Resultados para el modelo M | M | 1 | K

Para este caso,

$$C_n = \begin{cases} \left(\frac{\lambda}{\mu}\right)^n = \rho^n \text{ para } n = 0, 1, 2, ..., K \\ \\ 0 \text{ para } n > K \end{cases} \tag{3.26}$$

Entonces, para $\rho \neq 1$[1],

$$P_0 = \frac{1}{\sum\limits_{n=0}^{K} (\lambda/\mu)^n} = \frac{1}{\left[\dfrac{1 - (\lambda/\mu)^{K+1}}{1 - \lambda/\mu}\right]} = \frac{1-\rho}{1-\rho^{k+1}} \tag{3.27}$$

Así para

$$P_n = \frac{1-\rho}{1-\rho^{K+1}}\rho^n, \text{para } n = 0, 1, 2, ..., K \tag{3.28}$$

[1]Si $\rho = 1$, entonces $P_n = 1/(K+1)$ para n= 0, 1, 2, . . . , K, así $L = K/2$.

Entonces sustituyendo en,

$$E[N] = \sum_{n=0}^{K} nP_n = \frac{1-\rho}{1-\rho^{K+1}}\rho \sum_{n=0}^{K} \frac{d}{d\rho}\left(\rho^n\right)$$
$$= \frac{1-\rho}{1-\rho^{K+1}}\rho \frac{d}{d\rho}\left(\sum_{n=0}^{K}\rho^n\right) = \frac{1-\rho}{1-\rho^{K+1}}\rho \frac{d}{d\rho}\left(\frac{1-\rho^{K+1}}{1-\rho}\right)$$
$$= \rho \frac{-(K+1)\rho^K + K\rho^{K+1} + 1}{(1-\rho^{K+1})(1-\rho)}$$
$$= \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}} \tag{3.29}$$

Cuando el número de servidores es 1 entonces,

$$E[N_q] = E[N] - (1 - P_0) \tag{3.30}$$

Los tiempos de respuesta y de espera se pueden obtener por medio de la Ley de Little (ver sección 3.1.6).

3.4. El modelo M | G | 1

Además de los modelos pasados, que utilizan distribuciones Markovianas y que son de los más usuales e importantes en el estudio de sistemas de colas, existe el modelo M | G | 1, el cual ya no se comporta como un caso especial de un proceso de nacimiento y muerte. En la práctica esto se da por que existen sistemas para los cuales los tiempos de servicio exponenciales negativos no son realistas. Este modelo es generalmente aplicable en ambientes donde múltiples usuarios (una población grande de clientes potenciales) están usando un recurso escaso, como una línea de transmisión o un servidor central, generalmente por periodos de tiempo distribuidos.

El modelo asume que el sistema de cola tiene un solo servidor y un proceso de Poisson de entrada (tiempos entre llegadas exponenciales) con una tasa de llegada fija λ. También se asume que los clientes tienen tiempos de servicio independientes con la misma distribución de probabilidad. Sin embargo, no se imponen restricciones sobre cual sea la distribución de tiempo de servicio.

3.4.1. Resultados para el modelo M | G | 1

Para este modelo se necesita saber o estimar el promedio $1/\mu$ y la varianza σ^2 de la distribución. Como ya se dijo, cualquier sistema puede alcanzar eventualmente su condición de estado estable si $\rho = \lambda/\mu < 1$. Los resultados para el estado estable para este modelo general son los siguientes:

$$P_0 = 1 - \rho \tag{3.31}$$

$$E[N_q] = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)} \tag{3.32}$$

$$E[N] = \rho - E[N_q] \tag{3.33}$$

$$E[W] = \frac{E[N_q]}{\lambda} \tag{3.34}$$

$$E[R] = E[W] + \frac{1}{\mu} \tag{3.35}$$

Considerando que se puede tomar cualquier distribución para el tiempo de servicio es notable la fórmula tan simple que se puede obtener para $E[N_q]$. Esta fórmula es uno de los resultados más importantes en la teoría de colas por su facilidad de uso y el predominio de los sistemas de colas M | G | 1 en la práctica. Esta ecuación para $E[N_q]$(o su contraparte $E[W]$) es referida comúnmente como la fórmula de Pollaczek-Khintchine, llamada así por los dos pioneros en el desarrollo de la teoría de colas quienes derivaron la fórmula a principios del año 1930.

Para cualquier tiempo de servicio fijo esperado $1/\mu$, se debe notar que $E[N_q]$, $E[N]$, $E[W]$ y $E[R]$ aumentarán conforme σ^2 aumente. Este resultado es importante por que indica que la consistencia del servidor tiene una relación mayor en el desempeño de la estación de servicio y no solo la velocidad promedio del servidor.

Cuando la distribución del tiempo de servicio es exponencial, $\sigma^2 = 1/\mu^2$, los resultados anteriores se reducirán a los resultados para el modelo M | M |1.

La flexibilidad de la distribución del tiempo de servicio que se da en este modelo es extremadamente útil, así es desafortunado el esfuerzo de derivar resultados similares para los resultados del caso de servidores múltiples no hayan tenido éxito. Sin embargo, se han obtenido algunos resultados para el caso de servidores múltiples para casos especiales como los modelos M | D | s y el modelo M | E_r | s [10].

CAPÍTULO 4

Redes de Colas.

En el mundo real, muchos sistemas no solo están estructurados con una estación de colas, sino de varias y cada estación con un servicio y una cola independiente. En este tipo de sistemas los clientes pasan de una estación a otra para así completar el servicio requerido. Por ejemplo, en un sistema computacional en el que un determinado número de usuarios esperan que se realicen operaciones por un conjunto de periféricos o procesadores, o en sistemas de comunicaciones donde los paquetes viajan por medio de conexiones independientes y enrutadores intermedios desde el origen hasta su destino. Este capítulo se centra principalmente en las redes de colas abiertas, es decir, que no se limita el número de clientes.

Así las redes de colas (RCs) están compuestas por n estaciones de colas interconectadas. Cada una de las estaciones o nodos son independientes de las otras. Las estaciones estarán conectadas de forma tal que la salida de un nodo sea la entrada para los n siguientes nodos. Se asume que existe una fuente de entrada que siempre tiene clientes para alimentar la RC.

Una manera más formal de describir las RCs es como un grafo dirigido en el cual los nodos son las estaciones de colas y los vértices son las rutas por las cuales los clientes son guiados de un nodo a otro. Los vértices pueden ser etiquetados con probabilidades de enrutamiento o tasas de llegadas. En las RCs los nodos de origen y final son generalmente denotados como un nodo especial.

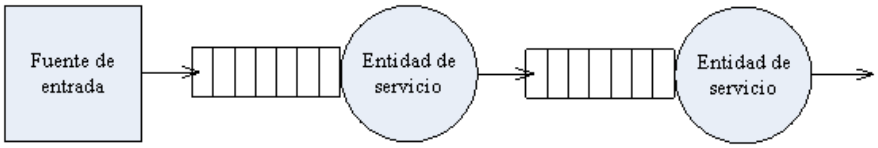


Figura 4.1: Red de dos M | M | 1 seriadas.

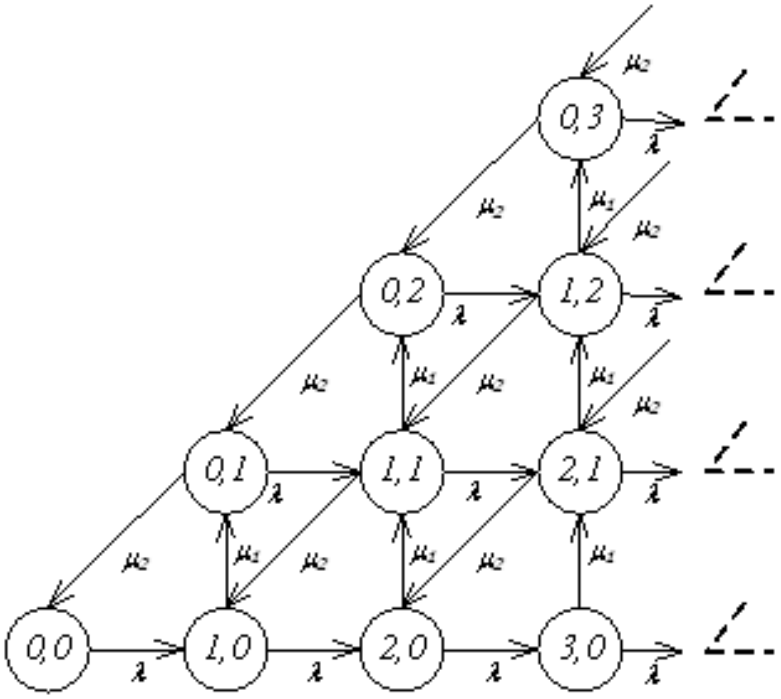


Figura 4.2: Diagrama de estados de transición de una CMTC con $M = 2$.

4.1. Redes de colas de prealimentación.

Este tipo de redes pueden ser ordenadas de tal forma que cuando un cliente va de una cola i a una cola j, implica que $i < j$, es decir, estas RCs son acíclicas. Por esta propiedad estas redes deben de ser abiertas.

Primero se considerará el caso de M colas en serie. Todas las estaciones serán del tipo M | M | 1. La tasa externa para la cola 1 es igual a λ y la tasa de servicio para la cola i es igual a μ_i. Para alcanzar la estabilidad se necesita que para todo $\rho_i = \lambda/\mu_i < 1$. Si existe alguna estación con el caso contrario, estas estaciones crearán colas de espera infinitas y por lo tanto el tiempo de respuesta de las series de colas también será infinito. La cola con el valor más grande para ρ_j se le llamará el nodo de cuello de botella (o el punto débil) de la RC. En la Figura 4.1 se puede observar un caso práctico de esta red.

Ya que no hay salidas de la RC, y no hay llegadas a la RC entre dos colas, la tasa de llegada en cualquier cola i será igual a λ. Más aún, una salida en una cola i(con $i = 1, ..., M-1$) será una llegada n la cola $i+1$. En la Figura 4.2 se muestra el diagrama de transición de estados de la cadena de Markov que modela una serie de RC con $M = 2$, y espacio de los estados $\mathcal{I} = \mathbb{N}^2$. Cada estado $(i, j) \in \mathcal{I}$ significa que hay i clientes en la cola 1 y j clientes en la cola 2. La suma de i y j es el número total de clientes en la RC. En la Figura 4.2 , una columna de estados representa los estados con el mismo número de clientes presentes en la RC. Así se reconocen básicamente 4 diferentes tipos de estados y se denotan las ecuaciones de balance a continuación:

estado (0,0)	:	$p_{0,0}\lambda = p_{0,1}\mu_2,$
estados $(i, 0), i \in \mathrm{N}^+$	:	$p_{i,0}(\lambda + \mu_1) = p_{i-1,0}\lambda + p_{i,1}\mu_2,$
estados $(0, j), j \in \mathrm{N}^+$	:	$p_{0,j}(\lambda + \mu_2) = p_{1,j-1}\mu_1 + p_{0,j+1}\mu_2,$
estados $(i, j), i, j \in \mathrm{N}^+$	:	$p_{i,j}(\lambda + \mu_1 + \mu_2) = p_{i-1,j}\lambda + p_{i+1,j-1}\mu_1 + p_{i,j+1}\mu_2$
normalización	:	$\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_{i,j} = 1$

Estas ecuaciones se pueden resolver de la misma manera como se explicó en la sección 3.3 y se obtendrá:

$$p_{i,j} = (1 - \rho_1)\rho_1^i \times (1 - \rho_2)\rho_2^j \tag{4.1}$$

En las RCs en serie se observa que se puede calcular la distribución de probabilidad total de clientes como el producto sobre cada distribución de probabilidad de cada cola. Así estas series de RCs son llamadas redes de colas de forma de producto. Esta propiedad se puede mostrar de manera sencilla, todos los flujos de clientes en series de RCs son flujos de Poisson. Esto fue demostrado por Burke y se conoce como el teorema de Burke que dice:

El proceso de salida de un solo servidor estable de una cola M | M |1 con una tasa de llegada y de servicio λ y μ respectivamente, es un proceso de Poisson con tasa λ.

Para mostrar la validez del teorema, se considera una cola M | M | 1. Mientras la cola no este vacía, los clientes dejarán la estación conforme el tiempo de distribución es igual a la distribución de tiempo de servicio. Si después de una salida la cola se vacía, tiene que esperar hasta la siguiente llegada, la cuál toma una longitud exponencialmente distribuida, más su periodo de servicio siguiente. Entonces, cuando se deja una cola vacía, el tiempo que pasa hasta la siguiente salida tiene una distribución hipo-exponencial con dos fases, y tasas λ y μ. La probabilidad de que después de la salida la cola no

esté vacía es igual a ρ, así se puede calcular la distribución entre llegadas $F_D(t)$ como:

$$F_D(t) = \rho(1 - e^{-\mu t}) + (1 - \rho)\left(1 - \frac{\mu}{\mu - \lambda}e^{-\lambda t} + \frac{\lambda}{\mu - \lambda}e^{-\mu t}\right), \tag{4.2}$$

la cuál se reduce a $F_D(t) = 1 - e^{-\lambda t} = F_A(t)$, donde F_A es la función de distribución de llegada.

Finalmente, se denota con N_i como el número de clientes en la cola i, la variable aleatoria $\underline{N} = (N_1, ..., N_M)$ representa el estado de las RC en serie. Debido a esto, el espacio de estados será $\mathcal{I} = \mathbb{N}^M$ y como resultado se obtienen las siguientes probabilidades de estado estable:

$$\begin{aligned} P\left\{\underline{N} = \underline{n}\right\} &= \prod_{i=1}^{M}(1 - \rho_i)\,\rho_i^{n_i} = \left(\prod_{i=1}^{M}(1 - \rho_i)\right)\left(\prod_{i=1}^{M}\rho_i^{n_i}\right) \\ &= \frac{1}{G}\prod_{i=1}^{M}\rho_i^{n_i}, \end{aligned} \tag{4.3}$$

donde G es una constante de normalización descrita como:

$$G = \left(\prod_{i=1}^{M}(1 - \rho_i)\right)^{-1} \tag{4.4}$$

lo cuál asegura que la suma de todas las probabilidades sea igual a 1. Se puede resaltar que esta probabilidad es resultado del producto de las probabilidades de estado estable de cada cola (ver sección 3.3).

Ahora bien, este tipo de redes no necesariamente tienen que estar en serie. Todas por si mismas son del tipo M | M | 1. Se denota el ambiente con "0" y se define el proceso de llegada total para el ambiente como un proceso de Poisson con tasa λ_0. Se define $r_{i,j}$ como la probabilidad (especificada por el usuario) de que un cliente deje la cola i y se vaya a la cola j. Por definición, se sabe que $r_{i,j} = 0$ toda vez que $j \leq i$. Las probabilidades $r_{0,i}$ indican como las llegadas son distribuidas en colas individuales y las probabilidades $r_{i,0}$ representan las salidas de la RC.

El flujo total de clientes que pasan por la cola j es igual a la suma de todo lo que viene del ambiente y de otras colas hasta que se complete el servicio, es decir:

$$\lambda_j = \lambda_0 r_{0,j} + \sum_{i<j}\lambda_i r_{i,j}, j = 1, ..., M \tag{4.5}$$

Estas ecuaciones son llamadas las ecuaciones de tráfico de primer orden. Habrá tantas ecuaciones como estaciones en la RC. Como este tipo de redes son acíclicas se pueden resolver de manera sucesiva, es decir se resuelve para cada variable y con el resultado obtenido se resuelve para la siguiente variable:

$$\lambda_0 \to \lambda_1 \to \lambda_2 \to ... \to \lambda_M \tag{4.6}$$

De manera similar que para los otros sistemas sencillos, si todos los $\rho_i = \lambda_i/\mu_i < 1$, se dice que la RC es estable. Si ese es el caso se puede ver la RC otra vez como el producto de las probabilidades de estado estable y se obtendrá el resultado de la ecuación 4.1.

4.2. Redes de Jackson.

Las redes de colas de Jackson (RCJs) son una extensión de las redes mencionadas en la sección 4.1, solo que la restricción de que un cliente solo esté atendido en nodos posteriores, no existirá. Es decir, estas redes permiten que los clientes sean enrutados a estaciones donde ya habían sido atendidos (ver Figura 4.3). Por esto, los flujos de clientes entre varias colas ya no son flujos de Poisson por que son compuestas por flujos dependientes.

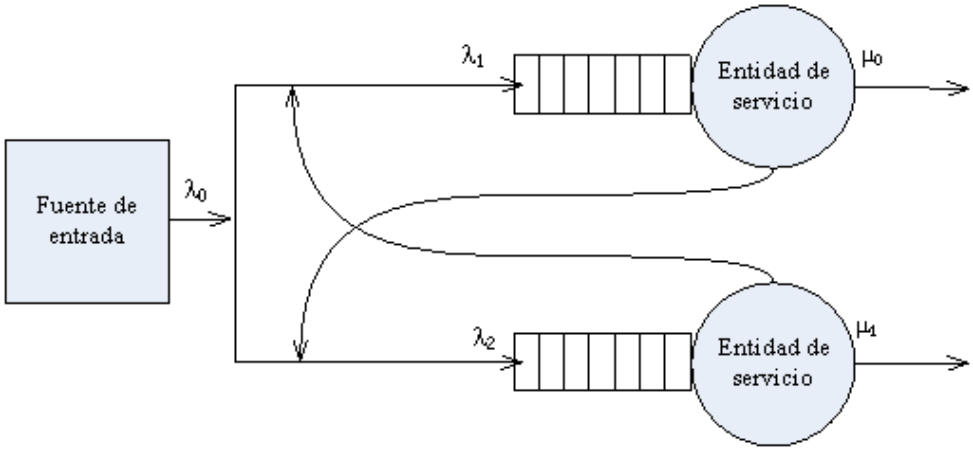


Figura 4.3: Red de Jackson simple.

Sin embargo, se puede descomponer el sistema en las diferentes estaciones que lo componen, y a pesar de que los flujos no sean de Poisson, los resultados analíticos comprueban que las probabilidades de estado estable tomarán la forma como si ciertamente fueran de Poisson[10]. Esto implicará que para las probabilidades de estado estable se obtendrá el mismo resultado de la ecuación 4.1 si para todas las i estaciones $\rho_i = \lambda_i/\mu_i < 1$. Para lo cuál se tendrán que resolver las ecuaciones de tráfico, que es la única diferencia con las redes de prealimentación serán las ecuaciones de tráfico que ahora serán:

$$\lambda_j = \lambda_0 r_{0,j} + \sum_{i=1}^{M} \lambda_i r_{i,j}, j = 1, 2..., M \tag{4.7}$$

Como ya no existe la restricción acíclica de las RC de prealimentación, entonces ya no se puede resolver de manera sucesiva, entonces se tendrá que resolver por medio de un procedimiento de eliminación Gaussiana o por medio de una técnica iterativa.

Cuando se cambia de estaciones con un solo servidor, a estaciones con varios servidores, se mantiene el resultado de los productos de la ecuación 4.1 . La única diferencia recae en la probabilidad del estado estable para las estaciones con múltiples servidores [10].

En la vida real las redes abiertas, son muy utilizadas para estudiar la muchos modelos que se adaptan a este tipo de funcionamiento, tal es el caso de una cola de espera de cualquier comercio (supermercado, banco, tortillas, etc), servidores web, dispositivos de red (enrutadores, switches), líneas de ensamble, etc, ya que en estos sistemas existe un número infinito de entradas al sistema y después de haber sido atendidos, las enradas saldrán del sistema de redes. Sin embargo, existen otro tipo de sistemas que se adaptan más a una red cerrada, dónde existen un número determinado de tareas rotando en la red, tal es el caso de una computadora, tanto el sistema operativo como el hardware, tendrán un número determinado de tareas en rotación en cierto instante de tiempo. Entonces, el estudio de estas redes va a depender del análisis del modelo del mundo real para que los resultados obtenidos sean los correctos.

CAPÍTULO 5

Simulación.

5.1. Introducción a la simulación.

La simulación es una de las técnicas usadas para la solución de modelos estocásticos, y muy útil en el análisis de desempeño de sistemas computacionales. Si el sistema aún no está implementado, la simulación es útil para probar el diseño y hacer comparaciones con diferentes alternativas. Si el sistema ya está construido y está preparado para que se le hagan mediciones, la simulación es útil, ya que se pueden probar más opciones de cargas y ambientes de trabajo.

Sin embargo, los modelos de simulación muchas veces fallan, no producen resultados útiles o totalmente verdaderos. Esto se debe a muchas razones, como modelar un nivel de detalle adecuado conforme a las mediciones que se requieran analizar. Para un modelo analítico se necesita menos detalle, en uno de simulación, el nivel de detalle está limitado solo por el tiempo de desarrollo de la simulación. Por lo tanto un nivel de detalle alto, tendrá como costo un tiempo mayor de desarrollo, incluso mayor tiempo para encontrar fallas, tiempo de cómputo, etc. Aunque un modelo con más detalle, dará mejores resultados, no es del todo cierto, ya que si se hacen suposiciones erróneas con algunos detalles en los datos de entrada del modelo, los resultados serán incongruentes. Lo mejor será iniciar por un nivel bajo de detalle, hacer pruebas, algunas mediciones, para después hacer el modelo más grande, hasta el nivel de detalle que se requiera.

Para obtener una buena simulación también es necesario tomar en consideración el lenguaje de programación a usar, si va a ser un lenguaje de uso general o uno especializado para simulación. Siempre es importante que el modelo sea estudiado hasta confirmar que se aproxima al sistema real, ya

que se puede incurrir en suposiciones erróneas que no representarán la realidad. Los resultados siempre tendrán que ser confirmados por modelos analíticos, mediciones o por simple intuición. El tiempo de la simulación dependerá de dos cosas, de los parámetros iniciales, ya que si no son bien establecidos los resultados no serán totalmente ciertos; y dependerá de la exactitud deseada, y la varianza de las cantidades. Ya que los modelos estocásticos dependen de variables aleatorias, se tendrá que poner especial atención en las funciones de generación de números aleatorios y en la semilla que se utilice, todo esto para obtener una varianza adecuada entre los números aleatorios[10].

5.2. Generación de números aleatorios.

Para simular modelos de desempeño de sistemas de comunicación por computadora con el uso de una aplicación de software se debe de tener la posibilidad de generar números aleatorios de ciertas distribuciones de probabilidad, como se mencionó en capítulos anteriores. La generación de números aleatorios (RNG, por sus siglas en inglés *Random Number Generation*) no es una tarea fácil y es muy importante, ya que cuando los números generados no conforman la distribución requerida, los resultados que se obtienen para una simulación no son del todo confiables.

Los números aleatorios no pueden ser generados por un algoritmo determinístico, es por esto que en computación se tiene que satisfacer esto con números pseudo-aleatorios. Para generar números pseudo-aleatorios usualmente se generan series de números pseudo-aleatorios en un subconjunto finito de $\mathbb{N}$, normalmente $\{0, \ldots, m-1\}, m \in \mathbb{N}$. De estas series se procesan números aleatorios uniformemente distribuidos. Después de esto, existen diferentes métodos para calcular variables no uniformemente distribuidas a partir de variables uniformemente distribuidas.

5.3. Tipos de simulación.

Existen diferentes tipos de simulación. Para el caso de las ciencias en computación algunos de los más importantes son la Emulación, Simulación de Monte Carlo, la Simulación Controlada por Seguimiento (*Trace-Driven Simulation)* y la Simulación de Eventos Discretos. La emulación está basada en la simulación con el uso de hardware o firmware, por ejemplo simular un procesador con el conjunto de instrucciones de otro. La simulación de Monte Carlo trata de modelar fenómenos probabilísticos, donde sus características no cambian con el tiempo. La simulación por seguimiento esta basada en el uso de trazas (registros de eventos ordenados por tiempo) y usualmente son usados para el análisis o la afinación de algoritmos para el manejo de recursos.

Existe una clasificación de simulaciones de eventos conforme a dos características: su espacio de

estado y su evolución en el tiempo [10]. En simulaciones de evento continuo se estudian sistemas en el que su estado cambia continuamente en el tiempo, usualmente estos sistemas se pueden describir con el uso de ecuaciones diferenciales (que a esta solución numérica a veces se le llama simulación). En sistemas físicos el tiempo es un parámetro continuo, a pesar de que solo se puede observar el sistema en ciertos puntos del tiempo, que lleva a un parámetro de tiempo discreto. Esta tesis se enfoca a las simulaciones de eventos discretos, DES (por sus siglas en inglés, *Discrete Event Simulations*), en el que los cambios del sistema ocurren en ciertos puntos del tiempo. El tiempo puede considerarse continuo o discreto, esto va a depender de cada aplicación.

5.4. Simulaciones de eventos discretos.

Una simulación que usa un modelo de estado discreto de un sistema es llamada simulación de eventos discretos, que de manera análoga se opone a la simulación de eventos continuos que utiliza modelos de estado continuos. En computación, se usan los modelos de eventos discretos, ya que el estado de un sistema está descrito por el número de tareas en varios dispositivos.

Para la simulación se utilizará un planificador de eventos que programe cada evento a un determinado tiempo. El tiempo tendrá que ser simulado también. Para avanzar el tiempo se tienen dos opciones, por unidad de tiempo, que se basa en avanzar el tiempo en pequeños incrementos y cada vez se verifica si un evento sucede. La otra opción es la controlada por eventos, que incrementa el tiempo hasta el evento con el tiempo más cercano. Esta última opción es la más usada en la simulación de sistemas computacionales.

El funcionamiento de este tipo de simulaciones está basado en guardar todos los eventos junto con su tiempo de ocurrencia en una estructura de datos (usualmente una lista ligada, una lista indexada o un árbol), la simulación se avanzará por cada evento que esté al inicio de la estructura utilizada, se ejecutará una rutina programada de atención al evento y finalmente el tiempo se incrementará conforme al tiempo del evento ocurrido. El tiempo de cada evento usualmente está descrito mediante tiempos relativos, es decir indican el incremento de tiempo al cuál ocurrirán. Como lo indica la teoría, para algunos casos (por ejemplo para el tiempo de llegada o el tiempo de servicio) este tiempo deberá ser calculado a partir de números aleatorios generados con cierta distribución probabilística.

Cada evento que suceda causa que otros eventos nuevos ocurran más adelante. Por ejemplo, cuando un cliente llega al sistema la rutina que atiende a este evento, causará que se genere el evento de la llamada de servicio para ese cliente, dicho evento creado se colocará en el lugar apropiado de la estructura de datos. Después se atenderá el evento que siga al inicio de la estructura de datos.

5.5. Análisis de resultados de la simulación.

Una simulación se lleva a cabo para obtener una visión cuantitativa de la operación de un sistema modelado. Cuando se ejecuta la simulación, cada evento se llevará a cabo en un determinado tiempo y finalizará en otro momento. Todas las muestras u observaciones tomadas pueden desplegarse o guardarse en archivos.

Para la evaluación final de la simulación, el responsable se tendrá que asegurar que los datos obtenidos son correctos y congruentes al la simulación programada y al modelo diseñado. Después se tendrá que hacer una evaluación de los datos a tomar en cuenta para obtener los resultados finales. Finalmente se calculan estadísticas a partir de la simulación que serán presentadas al usuario de una manera apropiada para su interpretación.

5.5.1. Validación y verificación del modelo.

La manera de medir un modelo de simulación si es bueno o malo está en función de que tanto se acerque la salida del modelo al sistema real. Cuando se desarrolla el modelo, se hacen varias suposiciones del comportamiento del sistema real. Así, se tienen dos vías para medir el modelo, primero es evaluar si las suposiciones son razonables y la segunda es evaluar si el modelo implementa esas suposiciones correctamente. Estos pasos son llamados validar y verificar, respectivamente. La validación se enfoca en lo que las suposiciones representan y la verificación en que tan correcta es la implementación.

Existen diferentes técnicas para verificar un modelo. En la literatura usualmente se refiere a la verificación como *debugging*. Ya que los modelos de simulación son programas grandes de computadora, todas las técnicas que sirven para el desarrollo, verificación o mantenimiento de programas grandes, también son aplicables a estos modelos. Algunas de estas son:

- **Diseño modular *top-down*.** Esta técnica se enfoca a la división del programa en módulos (subrutinas, subprogramas, procedimientos, etc.). Para organizar estos módulos de manera jerárquica y después seguirlos dividiendo en más sub-módulos. Esto ayudará ya que cada módulo pequeño es más fácil de verificar.

- **Anti-errores.** Se recomienda insertar algunas salidas o chequeos en el programa que denoten posibles errores, como contadores y algunas operaciones sencillas. Si el modelo de simulación es constantemente cambiado, estas salidas ayudarán a validar su funcionamiento.

- **Guía estructurada.** Esto consiste en explicar a alguien más como funciona el código línea por línea. Así el programador encontrará si existen errores.

- **Rastreo.** El rastreo consiste en tener una lista de eventos ordenados por tiempo con sus variables asociadas. Cuando se analiza esta lista al final de la simulación se pueden identificar fácilmente el origen de los errores.

- **Gráficas.** Las gráficas pueden ser la manera más simple que tiene el usuario de verificar como se está llevando a cabo la simulación y dependiendo de los resultados esperados puede identificar la validez del modelo.

- **Independencia de la semilla.** Se pueden realizar pruebas al cambiar la semilla de generación de números aleatorios, este cambio no debe de afectar los resultados finales.

- **Pruebas.** Ya que las simulaciones se basan en distribuciones de probabilidad y variables aleatorias, una prueba recomendable es indicar distribuciones constantes. Además también se deben de iniciar las pruebas con los casos más sencillos. Todo esto para poder deducir el resultado antes de que se lleve a cabo la simulación y así identificar el funcionamiento. Finalmente es aconsejable hacer pruebas con valores de entrada extremos (valores mínimos y máximos, opciones mínimas, etc.) y así verificar la robustez del modelo programado.

La validación del modelo se refiere a asegurar que las suposiciones usadas durante el desarrollo del modelo son razonables, y si son implementadas correctamente. Así el modelo arrojará resultados cercanos al sistema real. A diferencia de la verificación, las técnicas utilizadas para validar solo son aplicables solamente para estos casos.

La validación del modelo consiste usualmente en validar tres aspectos claves del modelo:

- Suposiciones.
- Parámetros de entrada (valores y distribuciones).
- Valores de salida y conclusiones.

Cada aspecto se debe de sujetar a una prueba de validación de tres posibles fuentes:

- Intuición experta.
- Medidas del sistema real.
- Resultados teóricos.

Se podría decir entonces que todo modelo está sujeto a la validación por medio de nueve pruebas. Sin embargo en la realidad usualmente no se tienen todas las pruebas, principalmente por falta de las fuentes de información.

Cuando se crea un modelo de simulación comúnmente es por que se requiere ahorrar en construir un sistema real, entonces no se tendrían las medidas reales con que comparar. A veces es posible tener algunos resultados teóricos en la literatura pero los modelos diseñados tal vez no estén. Lo que se hace, si es posible, es resolver el modelo de manera analítica, el problema sobresale cuando el modelo es muy complicado para resolverlo analíticamente. Esto nos lleva a que la manera más práctica de validar un modelo es a través de la intuición de un experto. El experto puede intuir y predecir la salida de un modelo por experiencia y analizar si existen errores en alguno de los tres puntos clave a validar.

5.5.2. Eliminación transitoria.

En la mayoría de las simulaciones los resultados de desempeño de interés se obtienen cuando el sistema ha llegado al estado estable. El estado inicial debe de ser descartado, a esta parte se le llama estado transitorio. El problema es identificar cuando se termina este estado transitorio que es llamado eliminación transitoria.

Existe una gran dificultad para definir exactamente este estado, por lo tanto la mayoría de los métodos para la eliminación transitoria son heurísticos. Los métodos más importantes son seis:

- **Corridas largas.** Este método es el más simple, ya que solo requiere de realizar una corrida de la simulación larga de tal forma que los resultados de la parte transitoria no afecten a los resultados finales. Sin embargo este método presenta dos problemas principales. El primero es que se consumen muchos recursos (tiempo de computación, tiempo de ocupación de equipos, etc.) y esta prueba suele ser muy cara. Segundo si es que esta prueba no involucra un costo excesivo, puede ser que no se sepa si se realizó una prueba lo suficientemente larga.

- **Inicialización correcta.** Se refiere a realizar la prueba de manera que se inicialice a un estado cercano al estado estable esperado. Por ejemplo que se inicie ya con un número de clientes avanzado en el sistema (tanto en la cola, como en servicio) para reducir la etapa transitoria y no afecte a los resultados finales.

- **Truncado.** Este método y los siguientes están basados en la suposición que la variabilidad durante el estado estable es menor que en el estado transitorio, que generalmente es verdad. El método consiste en dadas m observaciones $\{x_1, x_2, x_3, ..., x_n\}$, se ignoran las primeras l observaciones y después se calculan el mínimo y máximo de las $m-l$ observaciones restantes. Así, en este método

la variabilidad se mide en términos de un rango, el mínimo y máximo de observaciones. Si la trayectoria de las observaciones es graficada se puede determinar cuando la simulación entra al estado estable.

- **Supresión de los datos iniciales.** Al igual que el método de truncado, consiste en suprimir datos por medio de observaciones. Dado que los resultados cambian debido a la aleatoriedad de los números generados, en este método se recomienda hacer muchas observaciones con los mismos datos de entrada. Con esto se obtendrá que solo cambien un poco los resultados de una observación a otra solo por la semilla de generación de números aleatorios. Lo que se hace con estas observaciones es sacar un promedio de la trayectoria que sigue la simulación para después estimar la longitud del estado transitorio.
- **Promedio de lotes.** Consiste en hacer una simulación muy larga. Después se divide en partes iguales en duración igual llamadas lotes. El promedio de las observaciones en cada lote se llama promedio de lote, este método requiere el estudio de la varianza de estos promedios como función del tamaño de los lotes. El objetivo es encontrar el número n de lotes al cuál la varianza entre los lotes empieza a decrecer. Esta n será la longitud del periodo transitorio.

Aunque en la mayoría de los casos, el interés es el estudio y análisis del estado estable, existen algunos casos en que este estado nunca se alcanza. En estos casos se requiere estudiar el sistema en su estado transitorio. Este tipo de sistemas en simulación son llamados simulaciones terminadas (*terminating simulations).*

5.5.3. Muestras de eventos.

Las mediciones que se pueden obtener de la simulación pueden ser de dos tipos: orientadas al usuario u orientadas al sistema. Las orientadas al usuario son aquellas que se enfocan a un usuario o cliente en específico, esto se puede llevar a cabo con el monitoreo de un cliente en específico. Por ejemplo, cuando un cliente entra a una parte del sistema se toma la marca de tiempo t_a, de manera similar cuando el cliente deja esta parte del sistema se toma la marca de tiempo t_d. La diferencia $t_i = td_i - ta_i$ es el resultado del tiempo de residencia del cliente. Con la suma de las diferencias de todos los n clientes simulados, se obtiene un estimado del tiempo de residencia de cliente promedio como:

$$r = \frac{1}{n}\sum_{i=1}^{n}(td_i - ta_i) = \frac{1}{n}\left(\sum_{i=1}^{n} td_i - \sum_{i=1}^{n} ta_i\right) \tag{5.1}$$

Para las mediciones orientadas a sistema se tiene que monitorear el sistema en si. Por ejemplo, una medida de interés puede ser la probabilidad a largo plazo de que un buffer finito esté ocupado. Esta medición se puede llevar a cabo como se describió en el ejemplo anterior. Cuando el buffer esté lleno se

toma una marca de tiempo, cuando el buffer esté vacío se tomará la siguiente marca de tiempo. Con la diferencia obtendrá la medida de la variable aleatoria que denota el periodo del buffer lleno. Con la suma de las diferencias dividido entre el tiempo de simulación total se estima la probabilidad a largo plazo de que el buffer esté lleno.

5.5.4. Valores promedio e intervalos de confianza.

Supóngase que se ejecuta una simulación para estimar la media de la variable aleatoria X con $E[X] = a$. Para hacer esto, la simulación se utiliza para generar n muestras de x_i, con $i = 1, \ldots, n$ cada una se puede ver como una parte de la variable estocástica X_i. La simulación se construyó tal que las variables estocásticas X_i estén distribuidas de la misma manera que la variable aleatoria X. Entonces para calcular los intervalos de confianza, se necesita que las X_i sean independientes entre si.

Entonces, para estimar $E[X]$, se define una variable estocástica $\widetilde{X}$ llamada el estimador de X. Toda vez que $E[\widetilde{X}] = a$ el estimador $\widetilde{X}$ se le llamará imparcial. Ahora bien, toda vez que $P\left\{\left|\widetilde{X} - a\right| < \varepsilon\right\} \rightarrow 0$ cuando $n \rightarrow \infty$ el estimador $\widetilde{X}$ se llamará consistente. La condición posterior se traduce a un requerimiento de que la $var[\widetilde{X}] \rightarrow 0$, en todo momento que el número de muestras $n \rightarrow \infty$. Entonces las propiedades de imparcialidad y consistencia son propiedades deseables para los estimadores. Cuando sea que las observaciones $X_1, ..., X_n$ son independientes, entonces

$$\widetilde{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{5.2}$$

es un estimador imparcial y consistente para $E[X]$ ya que $E[\widetilde{X}] = E[X]$ (imparcialidad) y $var[\widetilde{X}] = var[X]/n$, tal que $var[\widetilde{X}] \rightarrow 0$ cuando $n \rightarrow \infty$ (consistencia).

Sin embargo aunque el estimador soluciona algunos problemas, el problema de la independencia de las X_i aún causa problemas, ya que generalmente las muestras tomadas de simulaciones sucesivas no son independientes. Por ejemplo si se toma una muestra para un X_i y es un valor grande o pequeño, para la siguiente X_i lo será más grande o pequeña, entonces esto muestra que son dependientes.

Para garantizar la independencia se pueden usar varios métodos. Cabe notar que esto es requerido para obtener los intervalos de confianza. Los métodos son:

- **Réplicas independientes.** Este método consiste en la ejecución de una simulación n veces, cada vez con una semilla diferente para la generación de números aleatorios. En la $i - ava$ simulación, se toman las muestras $x_{i,1}, ..., x_{i,m}$. Aunque estas muestras no son individuales entre si, los promedios $x_i = \left(\sum_{j=1}^{m} x_{i,j}\right) \Big/ m,\ i = 1, ...n$, son considerados que son independientes entre si. Estos n valores promedio son los que se consideran para el cálculo del valor promedio total y

los intervalos de confianza. La desventaja es que se tiene que ejecutar una y otra vez la simulación desde el inicio, esto implica el remover el comportamiento transitorio todas las veces.

- **Promedios de lotes.** Este método solo requiere de una simulación en donde se separan las muestras $x_1, ..., x_{n,m}$ en n lotes de tamaño m cada una. Para cada lote las muestras son promediadas como:

$$y_i = \frac{1}{m} \sum_{j=1}^{m} x_{(i-1)m+j} \tag{5.3}$$

Se asume que las muestras tomadas son independientes para calcular el promedio total y los intervalos de confianza. Ya que este método es una aproximación se dice que cada lote no es totalmente independiente, a pesar de esto este método es el más usado en la práctica.

- **Método Regenerativo.** Este método soluciona el problema de los lotes que no son totalmente independientes. También consiste en separar una sola ejecución en diferentes lotes, sin embargo esta separación se hace con puntos llamados de regeneración. Estos puntos de regeneración son definidos de tal forma que el comportamiento antes del punto es totalmente independiente del comportamiento después de este. Por ejemplo un buen punto de regeneración en una cola M | G | 1 es cuando la cola se queda vacía. Sin embargo, si el número de muestras por lotes no es constante, es más complicado, entonces se tienen que usar estimadores de razón[12].

Ya que se asume que las variables aleatorias X_i son independientes e idénticamente distribuidas, el estimador $\widetilde{X}$ de la sección pasada, de acuerdo al teorema del límite central[11], aproximadamente tendrá una distribución Normal con media a y varianza σ^2/n (se denota $var[X]$ como σ^2. Esto implica que la variable aleatoria

$$Z' = \frac{\widetilde{X} - a}{\sigma/\sqrt{n}} \tag{5.4}$$

tiene una distribución Normal de $(0, 1)$. Sin embargo como no se conoce la varianza de la variable aleatoria X, se tiene que calcular. Un estimador imparcial para σ^2, conocido como la varianza muestra, está dada por:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \widetilde{X}\right)^2, \tag{5.5}$$

Entonces la variable estocástica

$$Z = \frac{\widetilde{X} - a}{\widetilde{S/\sqrt{n}}} \tag{5.6}$$

tiene una distribución de Student [13] con $n-1$ grados de libertad (una $t_{n-1} - distribucion$). Cabe denotar que a y $\widetilde{\sigma}^2$ pueden ser calculados fácilmente cuando se conocen $\sum\limits_i x_i$ y $\sum\limits_i x_i^2$ de las muestras de la siguiente manera:

$$a = \frac{\sum\limits_{i=1}^{n} x_i}{n}, \quad \text{y} \quad \widetilde{\sigma}^2 = \frac{\sum\limits_{i=1}^{n} x_i^2}{n.-1} - \frac{\left(\sum\limits_{i=1}^{n} x_i\right)}{n\,(n.-1)} \tag{5.7}$$

para que durante la simulación, solo se tengan que mantener dos números por medición.

La distribución de Student con tres o más grados de libertad es una distribución de forma de campana simétrica, parecida en forma a la distribución Normal[14]. Cuando $n \longrightarrow \infty$, la $t_n - distribución$ se aproxima a una distribución Normal$(0, 1)$. Por medio del uso de tablas estándar para ésta distribución se puede obtener el valor $z > 0$ tal que la $P\{|Z| \leq z\} = \beta$ (x es llamado el valor crítico de ambos lados para la $t_n - distribución$, dado β). Con estos valores se puede escribir que:

$$\Pr\{|Z| \geq z\} = \Pr\left\{\frac{\left|\widetilde{X} - a\right|}{\sigma/\sqrt{n}} \leq z\right\} = \Pr\left\{\left|\widetilde{X} - a\right| \leq z\sigma/\sqrt{n}\right\} = \beta, \tag{5.8}$$

el cual establece que la probabilidad de que el estimador $\widetilde{X}$ se desvíe menos de $z\sigma/\sqrt{n}$ de la media a es β. La probabilidad β se le llama nivel de confianza. Como se puede observar, para hacer un intervalo de confianza un factor l menor, se necesitan l^2 observaciones más, para que la simulación deba de ser l^2 veces más larga.

CAPÍTULO 6

Diseño e implementación.

Basado en los capítulos anteriores se diseño e implementó una aplicación con las capacidades de modelar redes de colas y llevar a cabo su simulación, así como para poder visualizar los resultados importantes de desempeño de estas simulaciones. La aplicación está diseñada en base a componentes, además se crearon estos componentes adaptables al usuario. En este capítulo se describirán detalles sobre el diseño y la implementación de la aplicación que es el producto más importante de esta tesis.

6.1. Diseño General.

La aplicación llamada *QSim,* es un ambiente de desarrollo integrado, *IDE* (*Integrated Development Environment),* que se diseñó en base a tres componentes: un núcleo de simulación de eventos discretos, un módulo de herramientas para la lectura/escritura de XML y generación de código a partir de XML y un componente que provee el ambiente gráfico de interacción con el usuario que hace uso de los dos componentes anteriores. En la Figura 6.1 se muestra esta dependencia de los módulos.

Debido a que desde un inicio se pensño implementar la aplicación en el lenguaje de programación *Java*, es importante señalar que se tenían en mente todas las características importantes y las limitantes de la distribución del kit de desarrollo de *Java*. La aplicación está diseñada e implementada para la versión 1.5 de este kit. El diseño de está aplicación requería de una interfaz con las capacidades de arrastrar y dejar entidades que representaran los generadores de entrada y servidores, asimismo requiere de dibujar las conexiones entre estas entidades. Debido a que esto no es una tarea fácil con las clases que provee el kit de desarrollo de *Java* en los paquetes de gráficos y de componentes, se decidió utilizar

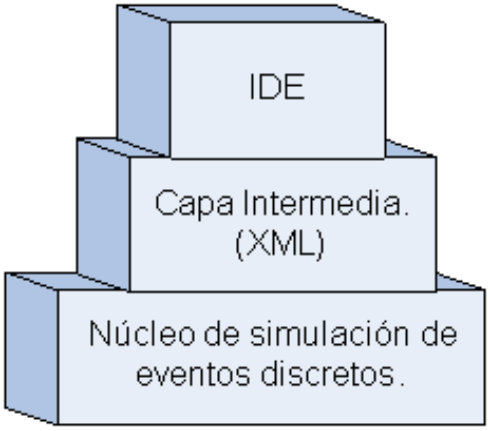


Figura 6.1: Estructura de la aplicación.

un paquete que brindará esta funcionalidad, el paquete utilizado fue *JHotDraw*.

JHotDraw es un marco de trabajo de interfaces de usuario gráficas (*GUI, Graphic User Interface*) para gráficos técnicos y estructurados [15]. Este poderoso paquete fue desarrollado por Erich Gamma y Thomas Eggenshwiler en *Java*, ambos son investigadores de la teoría y programación orientada a objetos. El paquete básicamente provee de todo el marco de trabajo para crear figuras que se pueden plasmar y mover en un lienzo de trabajo. Además de que también permite conectar éstas figuras, provee de muchas otras características como poner etiquetas, crear polígonos y otras funcionalidades más. Para el caso de ésta aplicación se usaron las primeras dos tareas principalmente.

La interfaz de usuario de esta aplicación desarrollada, está montada sobre este marco de trabajo de *JHotDraw,* el diseño y las clases desarrolladas se verán en una sección posterior. El diseño de 3 componentes permite al usuario tener 3 entradas a la aplicación como se muestra en la Figura 6.2:

Por lo tanto el usuario tendrá la opción de usar directamente el núcleo, o utilizarlo a través de las herramientas XML provistas o de igual forma hacer el uso del IDE para modelar y generar simulaciones de redes de colas. El diseño e implementación de estos tres componentes se detallan a continuación.

6.2. Núcleo de simulación de eventos discretos.

Primero se diseñó el núcleo de simulación de eventos discretos. Este módulo es el más importante de la aplicación ya que en éste, recae toda la funcionalidad de la simulación. Este componente no requería de más paquetes de *Java* de los que traía implementados para diseñar clases que realizaran las tareas

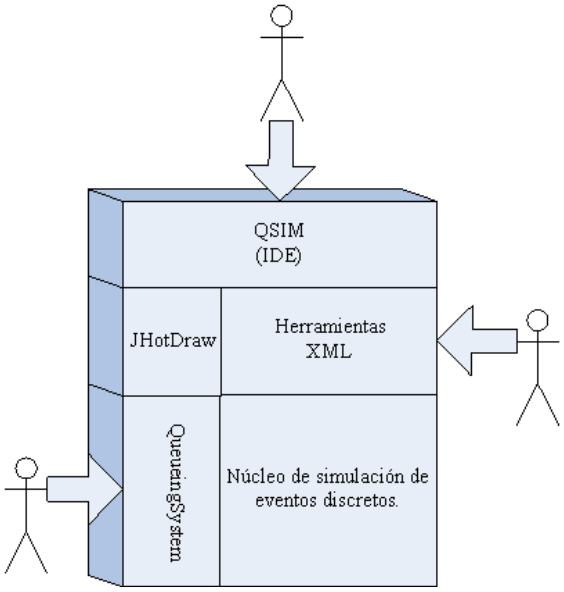


Figura 6.2: Entradas de usuario en la estructura de la aplicación.

básicas de la simulación de eventos discretos. Este módulo se diseñó de tal forma que se basara en una simulación orientada a eventos, es decir, que los eventos que suceden llevan el control del sistema, tanto en tiempo como en el estado general del sistema. Para lo cuál se implementaron las siguientes clases que se muestran en el diagrama de clases de la Figura 6.3:

- *QueingSystem*: esta clase modela un sistema de colas, el cuál contendrá un temporizador, n fuentes de entrada y m servidores, básicamente.
- *Clock*: la tarea principal de esta clase es de llevar el tiempo del sistema. Ya que lo necesitan varias clases, cada instancia se tendrá que compartir entre las clases del sistema que lo utilicen.
- *Scheduler*: modela el temporizador, el cuál su tarea básica es llevar una lista de eventos ordenados en base a su tiempo t en el que sucedan, por lo tanto se encargará de actualizar el reloj del sistema en cada evento que suceda.
- *DiscreteEvent*: éste es un evento discreto el cual contendrá el tiempo en el que sucede y el tipo de evento que envuelve.
- *InputGenerator*: este es un generador de entradas, es decir, generará los trabajos de llegada al sistema. Su tarea básica será generar los trabajos de llegada con un tiempo definido basado en una distribución de probabilidad.
- *Server*: la clase *Server* modela una entidad de servicio dentro del sistema de colas. Aquí la tarea

principal será brindar servicio a petición de un trabajo, pero si el servidor está ocupado, llevará el trabajo a una cola interna del servidor.

- *Job*: ésta es la clase base de un trabajo.
- *TimedQueue*: es una cola que tiene la capacidad de tomar el tiempo en el que los clientes se forman y salen de ésta.
- *RandomVariateGenerator*: este es un generador de números aleatorios basado en una distribución de probabilidad.

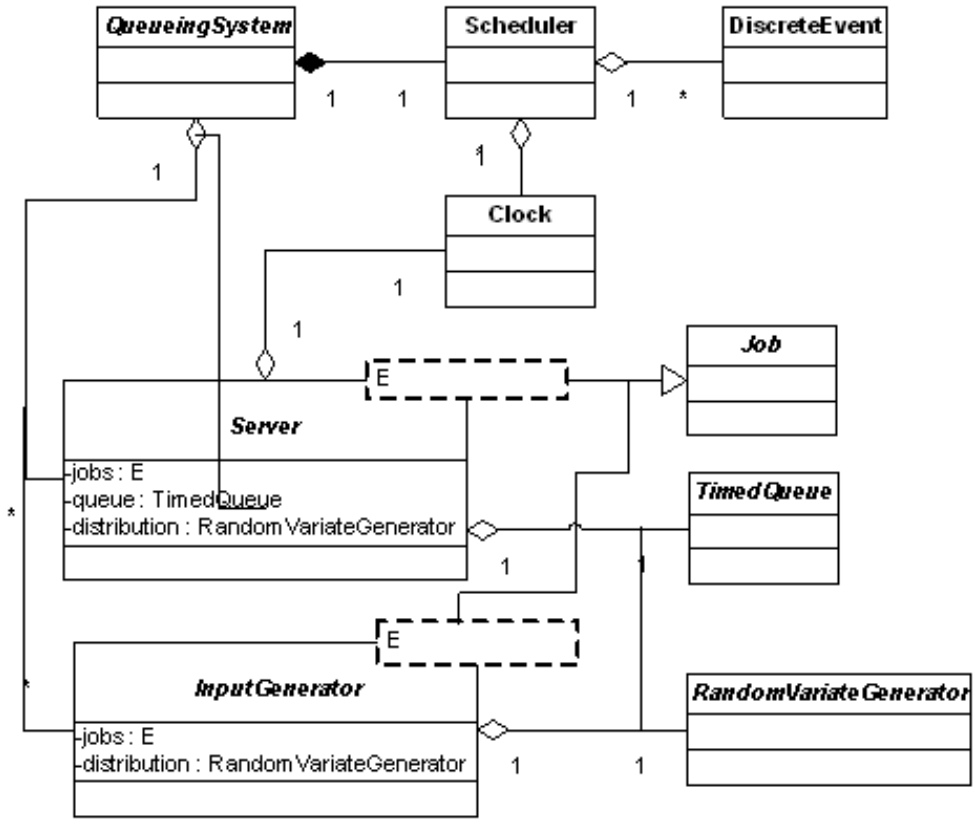


Figura 6.3: Diagrama de clases del núcleo de simulación.

Como se puede observar en la Figura 6.3 la mayoría de las clases se diseñaron de manera abstracta para que se implementarán de manera personalizada por el usuario. Para las clases `InputGenerator` y `Server`, se diseñaron de manera que reciban un tipo `E`, el cuál deberá ser un subtipo de clase `Job`, es decir, esto denota el tipo de trabajos que utilizarán estas clases.

El funcionamiento de este paquete se muestra en el diagrama de colaboración de la Figura 6.4. Este funcionamiento está totalmente apegado a lo descrito en las secciones 3.1.3 y 3.1.4.

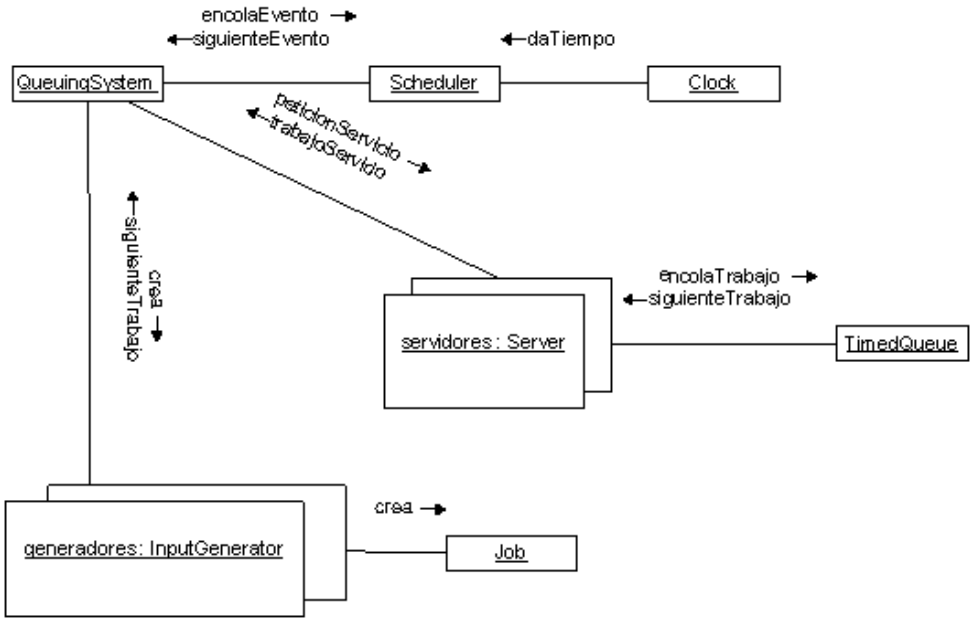


Figura 6.4: Diagrama de colaboración de las clases del núcleo de simulación.

Cabe mencionar que aunque no se vea claramente el uso de todas las clases en el diagrama de clases, una parte muy importante de este módulo y de toda la aplicación es la generación de números aleatorios. Esto se debe a que si estos número no son generados correctamente, el funcionamiento y por lo tanto los resultados de toda la simulación, serán erróneos. Para generarlos fue utilizada la clase `Random` que proporciona *Java* para generar números aleatorios entre 0 y 1. Esta clase genera estos números en base a una distribución uniforme [16]. Se decidió utilizar ésta clase y no una propia, debido a los años de desarrollo además de que está muy probado que en realidad si genera números (pseudo) aleatorios uniformemente distribuidos [17]. Las clases que se implementaron fueron a partir de la clase abstracta `RandomVariateGenerator` la cuál implementa una interfaz, para que el usuario implemente el método `getNextDouble()`, el cuál variará en las implementaciones para generar números aleatorios conforme a una distribución de probabilidad en particular.

Para las clases `InputGenerator` y `Server` se explotó la funcionalidad de los *Generics*, introducidos en la versión 5 del lenguaje *Java*. Esta funcionalidad permite al desarrollador definir una clase parametrizada, la cual podrá utilizar estos parámetros para contener objetos de cualquier tipo E, que cumplan ciertas características que desee el usuario. En el caso de estas dos clases, la restricción de implementación es que E sea un subtipo de la clase *Job*.

El funcionamiento básico de la clase `QueueingSystem`, está basado en la generación de casos para cada tipo de evento que suceda. Esto es, en cada ciclo del funcionamiento de esta clase, se hace suceder el siguiente evento que contenga el temporizador interno. De este evento se sacará su tipo de evento que

suceda, por omisión se definieron tres tipos de eventos: llegada, petición y terminación. El evento de llegada simboliza la llegada de un trabajo al sistema. Cuando llega al sistema el trabajo deberá hacer una petición al servidor, esto se traduce en encolar un evento de petición a algún servidor del sistema. El evento de petición encolará al trabajo en el servidor correspondiente y si no está ocupado lo pasará a servicio. Finalmente en el evento de terminación el trabajo sale del servidor, posteriormente, el trabajo continuará o saldrá del sistema según el modelo del usuario. Como se puede observar con estos tres tipos de eventos, un evento de llegada por cada fuente de entradas, un evento de petición y terminación por cada servidor, es posible llevar a cabo cualquier simulación de una red de colas tan complicada y con tantas entidades como se desee.

El usuario podrá implementar clases propias, con el uso de las clases y la funcionalidad provista por este núcleo y realizar simulaciones de sistemas de redes de colas.

6.2.1. Generación de resultados.

Como parte de la programación de la aplicación, se crearon clases concretas que implementaron la funcionalidad de entradas de tareas (clase `InputGenerator)` y servidores (clase `Server`) y cada clase generada ya sea por el usuario o por la herramienta de XML, extenderá de la clase `QueueingSystem`, para lo cuál es necesario crear la implementación de los métodos que hacen el cálculo de los resultados de la simulación.

Para cada variable de interés de tamaño (tamaño de las colas de espera y trabajos en el sistema) el cálculo se hace por medio de un algoritmo parecido al mencionado en la sección 5.5.3 y un promedio ponderado. Por ejemplo para el caso particular de el tamaño de las colas de espera(N_q), es importante considerar dos eventos: cuando se forma un nuevo trabajo y cuando sale un trabajo. Así, se toma el tiempo transcurrido entre llegadas a la cola Δt_n, se multiplica por el tamaño de la cola N_{qn}, que el trabajo ve (es decir, antes de que el se integre a la cola) y el resultado se acumula en una variable temporal. Al final el resultado se divide por el total de tiempo simulado. En resumen, este promedio se calcula de la siguiente manera:

$$N_q = \frac{\sum_{i=0}^{n} \Delta t_i N_{qi}}{\sum_{i=0}^{n} \Delta t_i} \tag{6.1}$$

Y de manera similar para los trabajos en el sistema.

Para las variables de tiempo basta con tomar el tiempo que se tarda cada trabajo ya sea en el sistema o en la cola. Para el caso particular de tiempo de espera en la cola(W_q) se deberá tomar el tiempo t_0 en el que un trabajo j se forme en la cola. Cuando salga, este trabajo j de la cola, se toma

el tiempo t_1 en el que salió y se realiza la diferencia de tiempos Δt_n. Para conocer el tiempo que se tardó en la cola y se acumula en una variable. Al final el resultado del acumulador se divide entre el número de clientes q que se formaron en la cola durante toda la simulación, para conocer el promedio de tiempo que se tardó un trabajo en la cola. Esto es:

$$W_q = \frac{\sum_{i=0}^{n} \Delta t_i}{q} \tag{6.2}$$

Para las demás variables de tiempo se hizo el cálculo de manera similar.

6.3. Herramientas para XML.

Las herramientas XML desarrolladas fueron diseñadas para tres tareas principalmente:

1. Proveer una interfaz de entrada al uso del núcleo de simulación por medio de un lenguaje de etiquetas completamente aparte de *Java* como lo es XML.
2. Modelar clases que sirvieran como contenedores de meta-datos para las clases del núcleo de simulación.
3. Y principalmente, proveer de una herramienta de descripción de sistemas de simulación en XML para a través de esta descripción, generar código que implemente el sistema descrito.

Para llevar a cabo el objetivo del primer punto fue necesario establecer un formato de descripción de sistemas de redes de colas. En resumen el archivo producido está formado por los siguientes campos:

- *system*: especifica el inicio de un sistema, es el miembro raíz del archivo. Es necesario especificar su nombre con el atributo *name.*
- *jobclass*: este miembro indica el nombre de la clase de trabajos que se usarán en el sistema. Además aquí se especifica el número de tipo prioridades que puede tener un trabajo por medio del atributo *priorititypes.*
- *distributionproperties*: dentro del uso de estas herramientas existe un archivo el cuál se utiliza para mapear el nombre corto de la distribución a el nombre de la clase que generará números con ésta distribución. Por ejemplo si el archivo contiene: `classname.distribution.M = ExponentialRandomGenerator`, indica que todo elemento que indique que genera números con una distribución nombrada M será generado por medio de la clase *ExponentialRandomGenerator*. Este campo indica el archivo que contiene estos mapeos.

- *inputgenerator:* indica el inicio de un elemento generador de entradas para el sistema. Se pueden insertar un número indeterminado de estos elementos, el número de elementos va a depender específicamente de las capacidades del equipo donde se desee ejecutar la simulación. En este elemento se deben especificar los elementos *meanarrivaltime* (tiempo entre llegadas promedio), *distribution* (distribución de probabilidad a la que va a generar las entradas), *capacity* (población que genera) y por último el elemento *linksto*, este es un elemento que se añade para indicar las conexiones que existen hacía otros elementos del sistema. Además se deben de indicar la clase del generador por medio del atributo *class*, y un identificador único en el atributo *id*.

- *server:* este elemento es muy similar al elemento *inputgenerator*, solo que éste describe un servidor del sistema. De la misma manera que el elemento pasado, se puede repetir para añadir servidores al sistema. La diferencia reside en que aquí se tiene que especificar un tiempo de servicio promedio por medio del elemento *meanservicetime* y un tamaño de la cola interna del servidor con el elemento *maxqueuelength*. También en la parte de los atributos de la clase además de la clase y el identificador, es necesario indicar la clase de la cola que tiene el servidor por medio del atributo *queueclass*. Por último se puede indicar cuantos servidores tendrá en su núcleo y la carga inicial que puede tener el servidor para algunos casos especiales.

Así por medio de estos elementos se puede describir un sistema de redes de colas como se desee. En el código 6.3 se muestra un ejemplo de este archivo XML.

El archivo de XML es validado por medio de un Esquema W3C XML (también llamado *XSD*), este archivo provee un medio para definir la estructura, contenido y la semántica de los documentos XML[18].

Internamente, para leer el archivo se hizo uso de XPath. XPath es un lenguaje para localizar partes de un documento XML[19], básicamente el lenguaje describe consultas a realizar a un archivo de XML y de esta forma acceder a los valores de los elementos que se desee. La versión del kit de desarrollo de *Java*, contiene clases de gran utilidad para hacer uso del lenguaje XPath, estas clases se encuentran en el paquete `javax.xml.xpath`.

Como se mencionó en el segundo punto al inicio de este capítulo, se diseñaron clases para guardar el contenido de este análisis. Además de guardar el contenido del análisis del archivo XML, es importante mencionar que sirven como objetos de peso ligero, así no se tendrán que utilizar las clases descritas en la sección 6.2 para guardar este contenido. En la Figura 6.5 se muestra el diagrama de clases y la relación entre éstas:

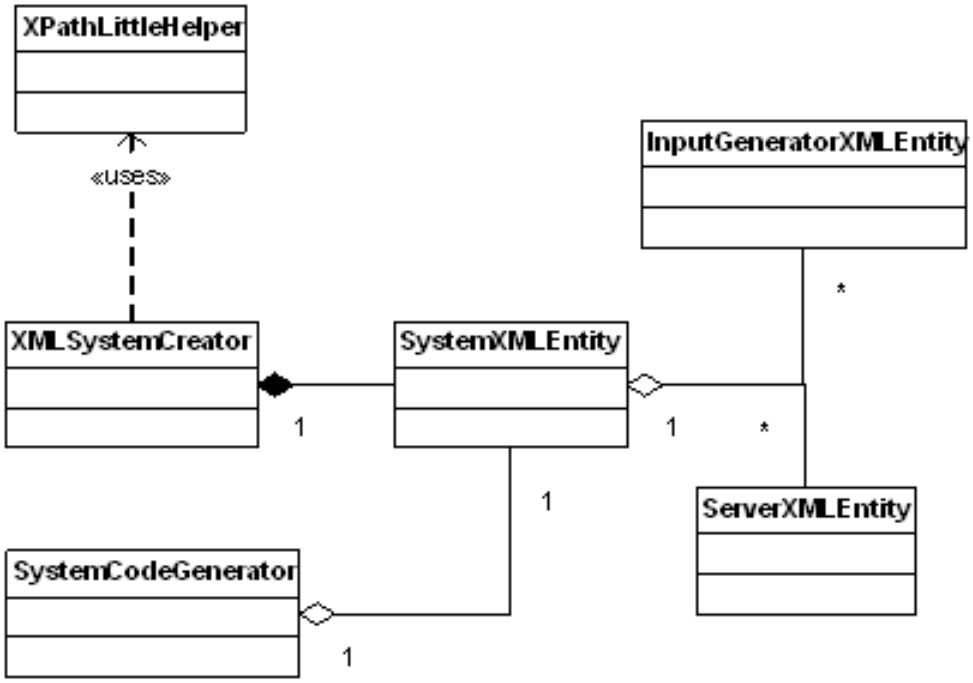


Figura 6.5: Diagrama de clases de herramientas XML.

En resumen, la clase `XMLSystemCreator` hace uso de un objeto de tipo `XPathLittleHelper` para el análisis del archivo XML y así construir un objeto de tipo `SystemXMLEntity`, el cuál contendrá los datos surgidos de este análisis.

Como parte de estas herramientas, se incluye la clase `SystemCodeGenerator`, que tiene la capacidad de generar código de una clase `QueingSystem`. Esta clase genera el código a partir de un objeto `SystemXMLEntity`. Así por medio del uso adecuado de estas clases, el usuario puede utilizar este paquete para generar un sistema de redes de colas listo para simularse. En la Figura 6.6 se muestra el diagrama de secuencia y muestra el uso que se le puede dar a este paquete:

6.4. Interfaz de usuario.

La interfaz de usuario para esta aplicación fue diseñada para que fuera un IDE. Como se mencionó al inicio de este capítulo, el diseño de éste módulo necesitaba que la interfaz tuviera mecanismos para que el usuario pudiera modelar sistemas con el uso de herramientas de arrastrar y dejar (*Drag and Drop*). Para lo cuál fue necesario el uso del paquete *JHotDraw*. Las clases generadas para esta interfaz, fueron derivadas de clases de *JHotDraw*, para que prestaran la funcionalidad adecuada al tipo de aplicación

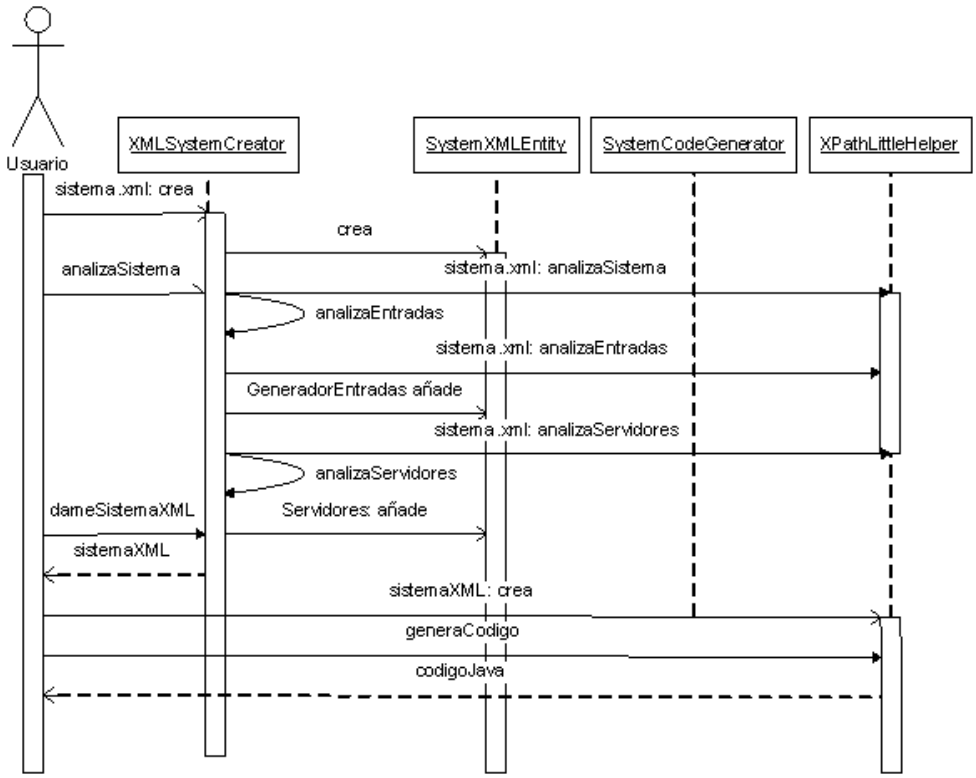


Figura 6.6: Diagrama de secuencia para herramientas XML.

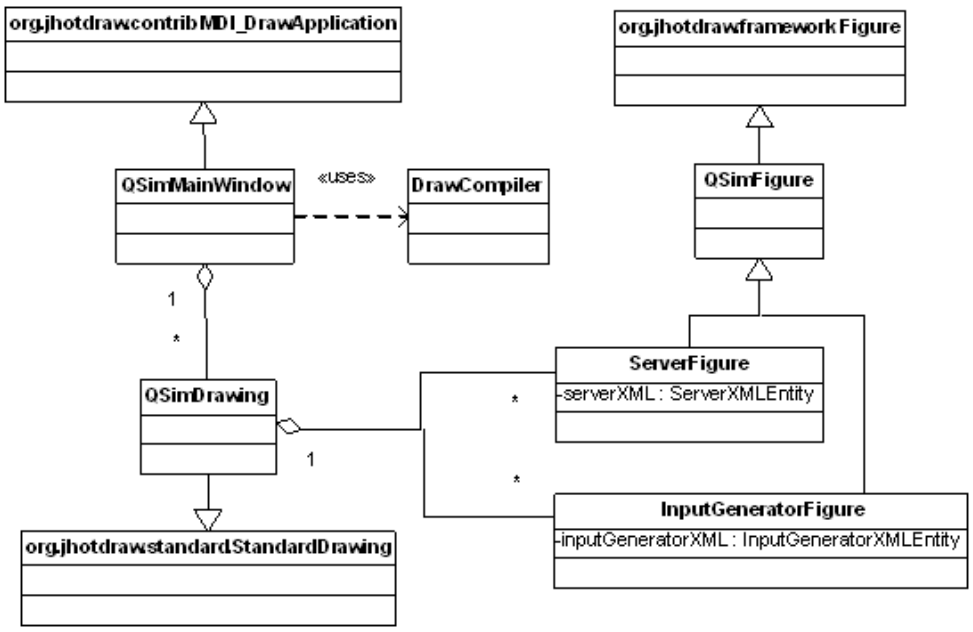


Figura 6.7: Diagrama de clases de la interfaz de usuario.

que se requería construir. En la Figura 6.7 se muestra el diagrama de clases del paquete de la interfaz de usuario.

La aplicación principal está construida en la clase `QSimMainWindow` la cuál genera la ventana principal de trabajo y dirige el uso de las demás clases. Además de las clases que proveen la funcionalidad para interactuar con el ambiente de trabajo, se provee la clase `DrawCompiler`. Esta clase analiza el dibujo realizado por el usuario para generar un archivo XML y así poder generar código del sistema descrito. Por medio de la entrada al compilador [20] y los mecanismos de reflexión [21] que provee el kit de desarrollo de *Java*, fue posible la compilación y ejecución de una nueva clase generada dentro del ambiente. Adicionalmente se construyeron clases de utilería para llevar a cabo la simulación en hilos de ejecución separados y animar la simulación. En el siguiente diagrama de colaboración se muestra la interacción entre las clases de este paquete:

6.5. Módulo de análisis de resultados.

Como ya se mencionó en el capítulo 5, una simulación depende estrictamente de datos probabilísticos y aún más importante, estos datos dependen a su vez de un generador de números aleatorios. Así se restringe que cada simulación dependerá de una instancia de un generador de números aleatorios, esto

Código 1 Ejemplo de descriptor XML de un sistema de redes de colas

```
<?xml version="1.0" encoding="UTF-8"?>
<system name="System0">
    <jobclass prioritytypes="1">Customer</jobclass>
    <distributionproperties>distributions.properties</distributionproperties>



    <inputgenerator class="CustomerGenerator" id="InputGenerator_0">
        <meanarrivaltime>100.0</meanarrivaltime>
        <distribution>M</distribution>
        <capacity>0</capacity>
        <linksto entity="server" id="Server_0"></linksto>
    </inputgenerator>

    <server class="CustomerServer" id="Server_0" queueclass="CustomerTimedQueue">
        <maxqueuelength>0</maxqueuelength>
        <leavesystem>true</leavesystem>
        <meanservicetime>100.0</meanservicetime>
        <distribution>M</distribution>
    </server>
</system>
```

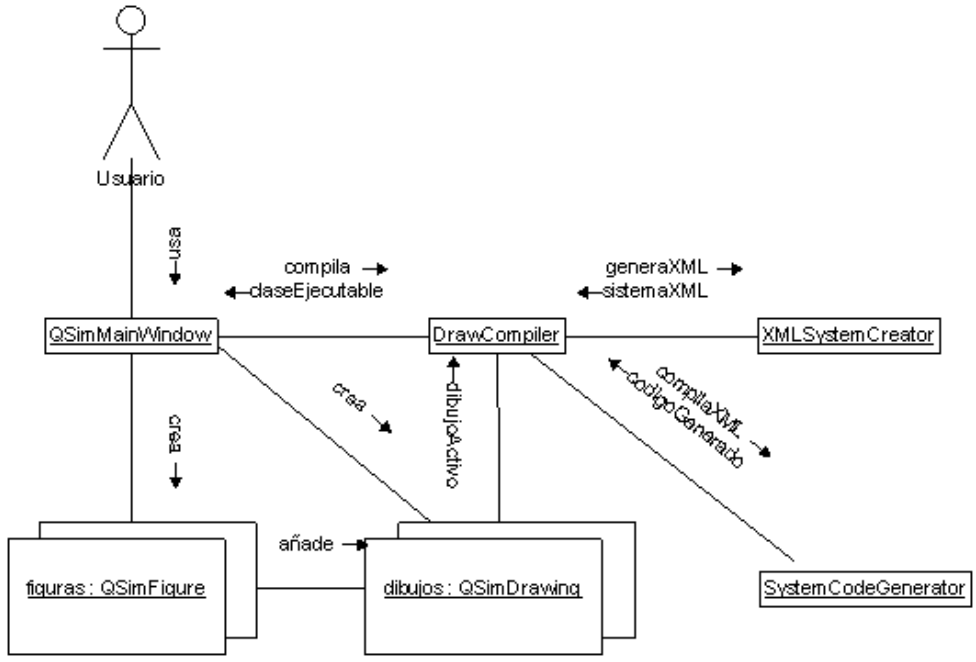


Figura 6.8: Diagrama de colaboración para las clases de la interfaz de usuario.

servirá para que toda simulación sea idéntica si y solo si la semilla del generador de números aleatorios es igual para cada simulación. Debido a esto si se modifica la semilla para el generador de números aleatorios se obtendrán resultados similares pero no idénticos. Además lo que importará en cada simulación es el estado estable. Entonces lo que se obtendrán son diferentes resultados o muestras, para cada variable de interés en un número determinado de simulaciones, lo cual lleva al análisis estadístico de estas muestras.

En este módulo se tomó en cuenta el método mencionado en la sección 5.5.2, para eliminación del estado transitorio se harán n simulaciones lo suficientemente largas en duración para que en el promedio de los datos de interés el estado estable prevalezca. Ya que para cada simulación se tomará una semilla de generación de números aleatorios diferente, entonces se asegura que cada variable de interés a calcular es independiente (ver 5.5.4). Como se aseguran estás dos propiedades se pueden calcular los intervalos de confianza para cada variable de interés con las fórmulas descritas al final de la sección 5.5.4.

La clase principal para este módulo es la clase `SimulatorAnalyzer` la cuál a partir de una instancia de un sistema de colas(`QueueingSystem`) realizará n simulaciones con diferentes semillas y tomará las n muestras para cada variable de interés del sistema. Para esto se implementó la clase `StatisticVariable` la cuál está encargada de guardar cada muestra de la simulación para después calcular la media, varianza e intervalos de confianza para cada variable. Asimismo, ya que se tienen un número determinado de servidores, y para cada servidor interesan los cálculos para diferentes variables, se creó la clase `ServerStatistics` la cuál contendrá las variables de interés de tipo `StatisticVariable` a calcular para cada servidor. En la figura 6.9 se muestra el diagrama de clases de este módulo y en la figura 6.10 se muestra el diagrama de secuencia del funcionamiento de estas clases.

Como se muestra en la figura 6.10, el usuario puede hacer uso de esta clase directamente desde su programa. Asimismo en la interfaz de usuario, se provee una entrada a este módulo para realizar un número determinado de simulaciones y calcular los intervalos de confianza para las muestras obtenidas.

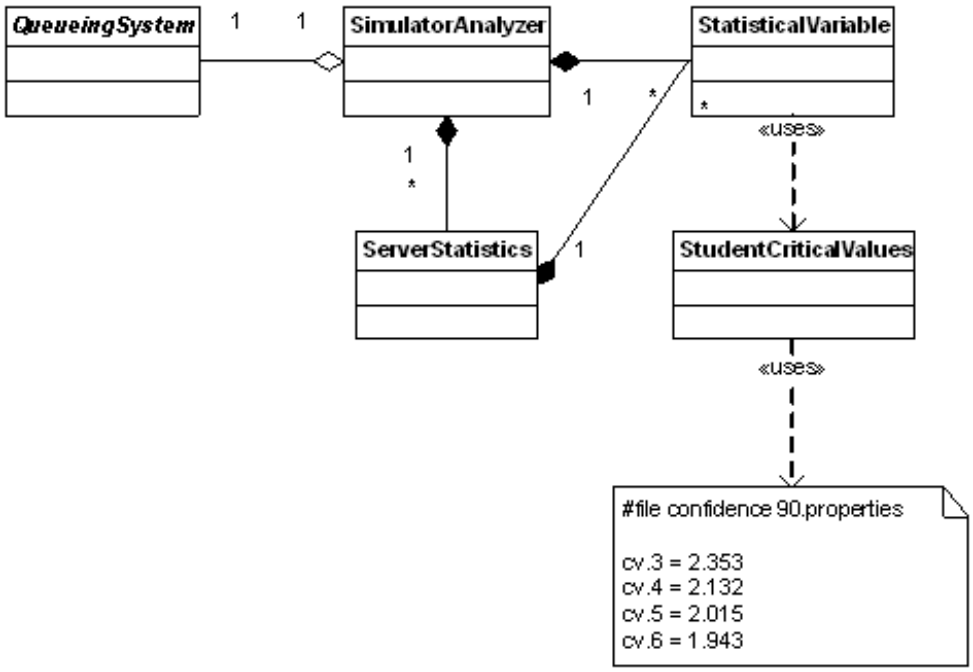


Figura 6.9: Diagrama de clases del módulo de análisis estadístico.

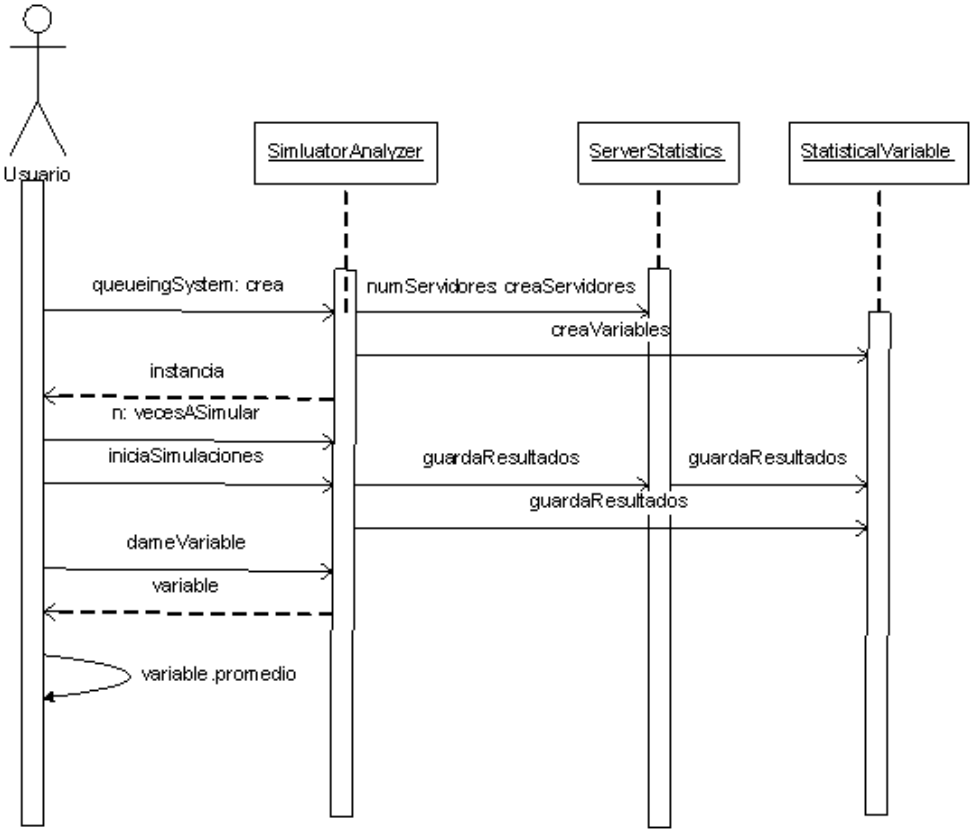


Figura 6.10: Diagrama de secuencia de las clases del módulo de análisis estadístico

CAPÍTULO 7

Pruebas y resultados.

Se realizaron varias pruebas con el sistema propuesto en esta tesis, en este capítulo se muestran un par de los resultados obtenidos en el sistema y su comparación con los resultados obtenidos analíticamente.

En primer lugar se revisará el caso más sencillo de una cola, con un origen de trabajos y un servidor de atención. En la segunda parte se revisará una red de colas en un modelo conocido como lo es el llamado de Servidor Central o *Central Server* en inglés.

En el Apéndice C se encuentran más resultados con sus respectivas comparaciones entre los resultados analíticos y los resultados obtenidos con simulación.

7.1. Resultados para prueba de sistema M | M | 1

Considérese el sistema mostrado en la Figura 7.1:

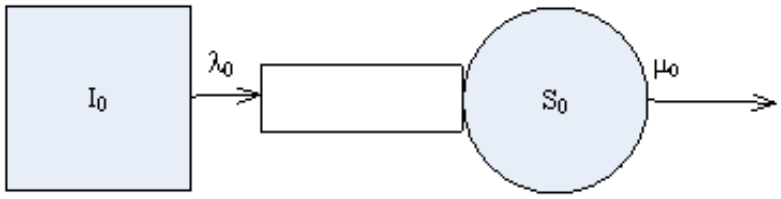


Figura 7.1: Sistema de prueba M | M | 1

Como se comentó anteriormente en la sección 3.3, este sistema pertenece a los del tipo de nacimiento y muerte. También es el sistema más sencillo que se puede encontrar de manear funcional en el mundo real. Este sistema cuenta con un generador de entradas (con tiempos de llegada distribuidos de manera exponencial), y una sola entidad de atención(con tiempos de servicio distribuidos de manera exponencial) para cada trabajo generado.

Las pruebas analíticas de este sistema se pueden evaluar con las fórmulas de la Ley de Little (ver sección 3.1.6), con las cuales se obtienen los siguientes resultados:

Dado un servidor de atención m y dados los siguientes datos de entrada para λ y μ:

$$m = 1 \tag{7.1}$$
$$\lambda = 1/30 \tag{7.2}$$
$$\mu = 1/20 \tag{7.3}$$

Se obtienen los siguientes resultados:

$$\rho = \frac{\lambda}{m\mu} \tag{7.4}$$
$$p_0 = \left(\sum_{j=0}^{m-1} \frac{(\frac{\lambda}{\mu})^j}{j!} + \frac{(\frac{\lambda}{\mu})^m}{m!} \frac{1}{1 - \frac{\lambda}{m\mu}} \right)^{-1} \tag{7.5}$$
$$p(n) = \begin{cases} \frac{(\lambda/\mu)^n}{n!} p_0 & \text{if} \quad 0 \leq n \leq m \\ \frac{(\lambda/\mu)^n}{m! m^{n-m}} p_0 & \text{if} \quad n \geq m \end{cases} \tag{7.6}$$
$$N = \sum_{n=0}^{\infty} n p(n) = 2 \tag{7.7}$$
$$N_q = \sum_{n=m}^{\infty} (n-m) p(n) = \frac{4}{3} \tag{7.8}$$
$$W_q = \frac{N_q}{\lambda} = \frac{120}{3} = 40 \tag{7.9}$$
$$W = W_q + \frac{1}{\mu} = 40 + 20 = 60 \tag{7.10}$$

Asimismo se obtuvieron resultados para $1/\lambda = \{30, 40, \ldots, 90, 100\}$, tanto analíticamente como dentro del sistema programado. En el sistema se introdujeron los mismos datos para las pruebas,

λ	μ	ρ	N	N_q	W_q	W
30	20	0.66667	2.0	1.3333	40.0	60.0
40	20	0.5	1.0	0.5	20.0	40.0
50	20	0.4	0.66667	2.6667	13.333	33.333
60	20	0.3333	0.5	0.16667	10.0	30.0
70	20	0.28571	0.4	0.11429	8.0	28.0
80	20	0.25	0.3333	0.08333	6.6667	26.667
90	20	0.2222	.28571	0.063492	5.7143	25.714
100	20	0.2	0.25	0.05	5.0	25.0

Tabla 7.1: Resultados analíticos para sistema M | M | 1.

se realizaron 5 simulaciones con tiempo de simulación fue por $1.0e^7$ y una colección de semillas para la generación de números aleatorios igual a $1973272912L$, $747177549L$, $20464843L$, $640830765L$, $1098742207L$. Los resultados analíticos se presentan en la la tabla 7.1 y los resultados obtenidos por el sistema (la media de las 5 simulaciones) en la tabla 7.2, la tabla 7.3 muestra los intervalos de confianza del 90 % calculados con el simulador.

Como se puede observar los resultados entre las dos tablas no varían en demasía (0.03 unidades en promedio), lo cual indica que el software realiza la simulación correcta del sistema introducido. En la tabla 7.4 se muestra el resultado de la diferencia entre los resultados y promediado, esto nos da un aproximado de la diferencia en unidades entre ambos resultados.

En la figura 7.2, se muestran las gráficas de $1/\lambda$ vs. ρ de ambos resultados, el analítico y el simulado respectivamente, en las cuales también se observa que la diferencia es muy poca. Aquí se puede señalar que en las gráficas de los resultados de QSim de aquí en adelante, se muestra también el intervalo de confianza, que se simboliza con dos barras horizontales que señalan el intervalo de confianza; ya que

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6663	1.9953	1.3290	39.8630	59.8470
40	20	0.4987	0.9964	0.4975	19.9234	39.8935
50	20	0.3984	0.6636	0.2651	13.2757	33.2271
60	20	0.33122	0.4985	0.1662	9.9796	29.9262
70	20	0.2849	0.3991	0.1141	7.9935	27.9502
80	20	0.2494	0.3329	0.0835	6.6815	26.6372
90	20	0.2216	0.2850	0.0634	5.7148	25.6705
100	20	0.1996	0.2497	0.0501	5.0124	24.9743

Tabla 7.2: Resultados obtenidos por QSim para sistema M | M | 1.

ρ	N	N_q	W_q	W
0.0018	0.0231	0.0214	0.5776	0.5964
0.0013	0.0067	0.0056	0.2029	0.2269
$9.89e^{-4}$	0.0035	0.0026	0.1041	0.1128
$5.39e^{-4}$	0.0027	0.0022	0.1368	0.1767
$4.13e^{-4}$	0.0012	$8.83e^{-4}$	0.0664	0.1053
$5.90e^{-4}$	0.0017	0.0012	0.0932	0.1173
$7.03e^{-4}$	0.0017	0.0012	0.1094	0.1297
$7.06e^{-4}$	0.0014	$9.49e^{-4}$	0.09186	0.1170

Tabla 7.3: Intervalos de confianza obtenidos por QSim para sistema M | M | 1.

ρ	N	N_q	W_q	W
$8.0887e^{-4}$	0.0019	0.0011	0.0407	0.0735

Tabla 7.4: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | M | 1.

las simulaciones no varian mucho entre ellas, solo se aprecia como si fuera una línea, en la Figura 7.3 se muestra un acercamiento a la gráfica para apreciar mejor el intervalo.

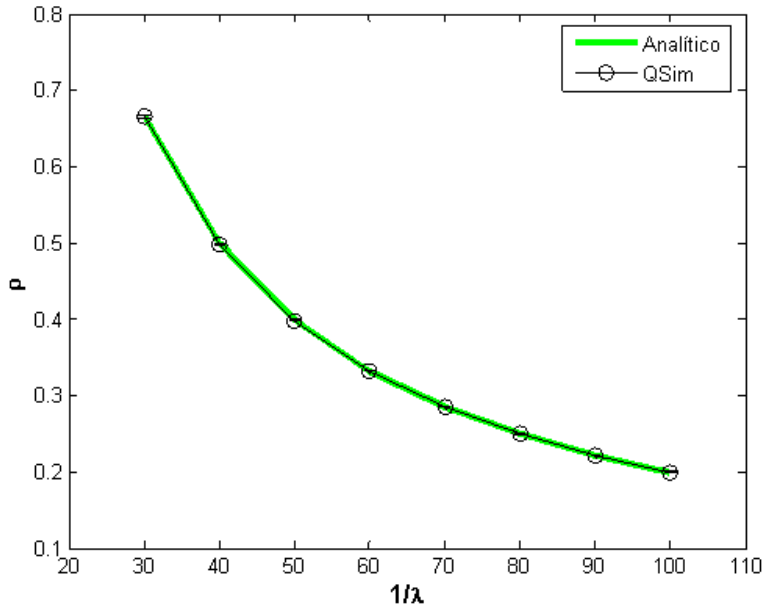


Figura 7.2: Gráfica $1/\lambda$ vs. ρ del sistema M | M | 1 de los resultados análiticos y obtenidos en QSim.

7.2. Resultados para prueba de sistema Servidor Central.

La segunda prueba importante del sistema programado para esta tesis, se baso en la simulación de un modelo de redes de colas conocido como lo es el del servidor central o *central server*. Este sistema es una red de colas cerrada y consta principalmente de 1 servidor central y M servidores, cada uno con su respectiva cola de atención e interconectados entre si como se muestra en la Figura 7.4.

En las conexiones de un servidor se involucra cierta probabilidad de que una tarea tome una ruta a un servidor u otro, esto por ejemplo podría simular un procesador de una computadora (servidor central) y M perífericos (servidores extras), a los cuales el procesador delegará tareas para realizar un ciclo de servicio.

El análisis de estos sistemas suele hacerse por medio de Redes de Jackson (ver Sección 4.2) o por un modelo que es una modificación a las Redes de Jackson llamado de Gordon-Newell [22], las cuales eliminan los generadores de tareas y suponen un número N de trabajos iniciales en circulación en la red. Las ecuaciones de balance están descritas por la siguiente expresión:

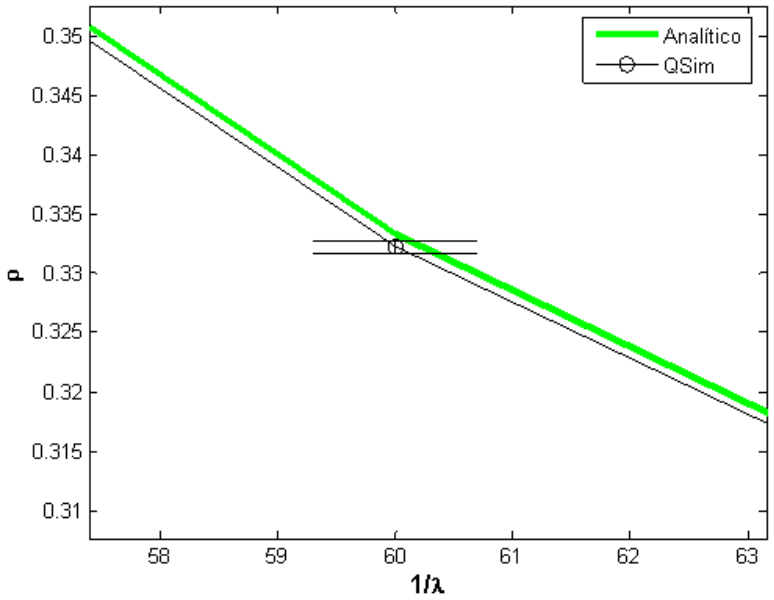


Figura 7.3: Acercamiento a la gráfica $1/\lambda$ vs. ρ del sistema M | M | 1 de los resultados análiticos y obtenidos en QSim.

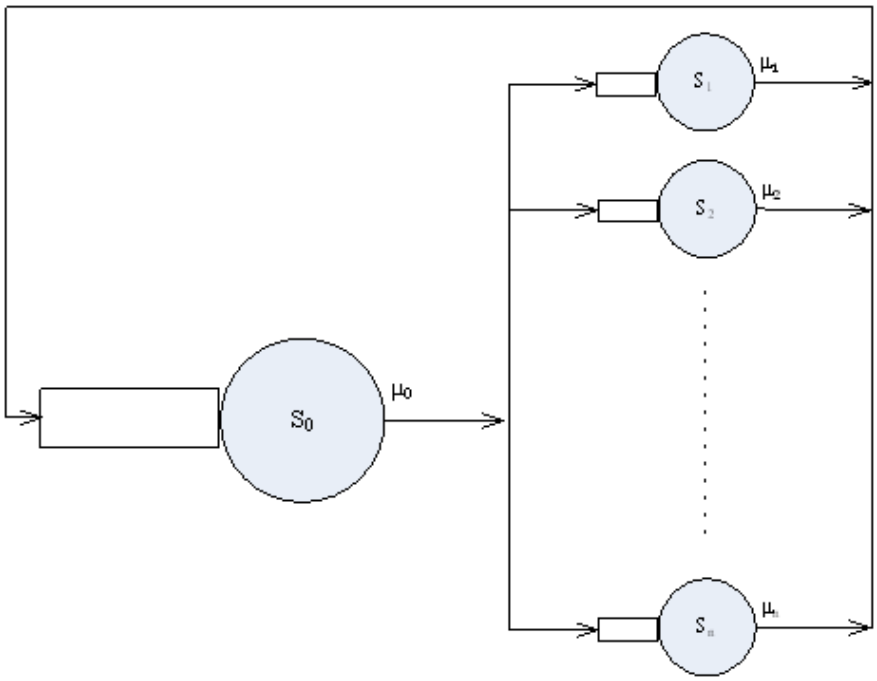


Figura 7.4: Sistema Servidor Central.

$$P(n_1, n_2, \ldots, n_M) = \frac{1}{G(N)} \prod_{i=1}^{M} (D_i)^{n_i}, \tag{7.11}$$

donde n_i son el número de clientes presentes en el servidor i, y $(D_i, D_2, \ldots, D_M)$ es una solución positiva real a las ecuaciones:

$$\mu_j D_j = \sum_{i=1}^{M} \mu_i D_i p_{ij},\ 1 \leq j \leq M, \tag{7.12}$$

y $G(N)$ es una constante de normalización con la cuál se asegura que la suma de las probabilidades $P(n_1, n_2, \ldots, n_M)$ sean igual a uno. Esto es, se define un vector $\overrightarrow{n} = (n_1, n_2, \ldots, n_M)$ el cual describe el estado de la red, entonces:

$$G(N) = \sum_{\overrightarrow{n} \in S(N,M)} \prod_{i=1}^{M} (D_i)^{n_i}, \tag{7.13}$$

donde

$$S(N, M) = \left\{ (n_1, n_2, \ldots, n_M) | \sum_{i=1}^{M} n_i = N \quad \text{y} \quad n_i \geq 0\ \forall_i \right\} \tag{7.14}$$

Cabe denotar que la suma de la ecuación 7.13 es realizada sobre todos los posibles $\binom{M+N-1}{N}$ estados del sistema $(n_1, n_2, \ldots, n_M)$. Así para la solución de la ecuación de balance 7.11 es necesario el cálculo de $G(N)$ el cual involucra la suma de $\binom{M+N-1}{N}$ términos y cada uno de ellos es un producto de M factores que son potencias de cantidades básicas (X_i). A pesar del número de operaciones que este cálculo involucra, Jeffrey P. Bunzen publicó un algoritmo iterativo en 1973, llamado de convolución, el cual realiza el cómputo de la variable $G(N)$ con un total de $N \cdot M$ multiplicaciones y $N \cdot M$ sumas, la descripción de este algoritmo se puede consultar en [23].

Así con la obtención de esta variable de nomalización se puede llevar a cabo el cálculo del resto de las variables importantes para este tipo de sistemas.

La prueba que se realizó consta de 1 servidor central y 2 servidores (ver Figura 7.5 y en la Figura 7.6 se muestra la pantalla de QSim con este sistema) extras y los siguientes datos:

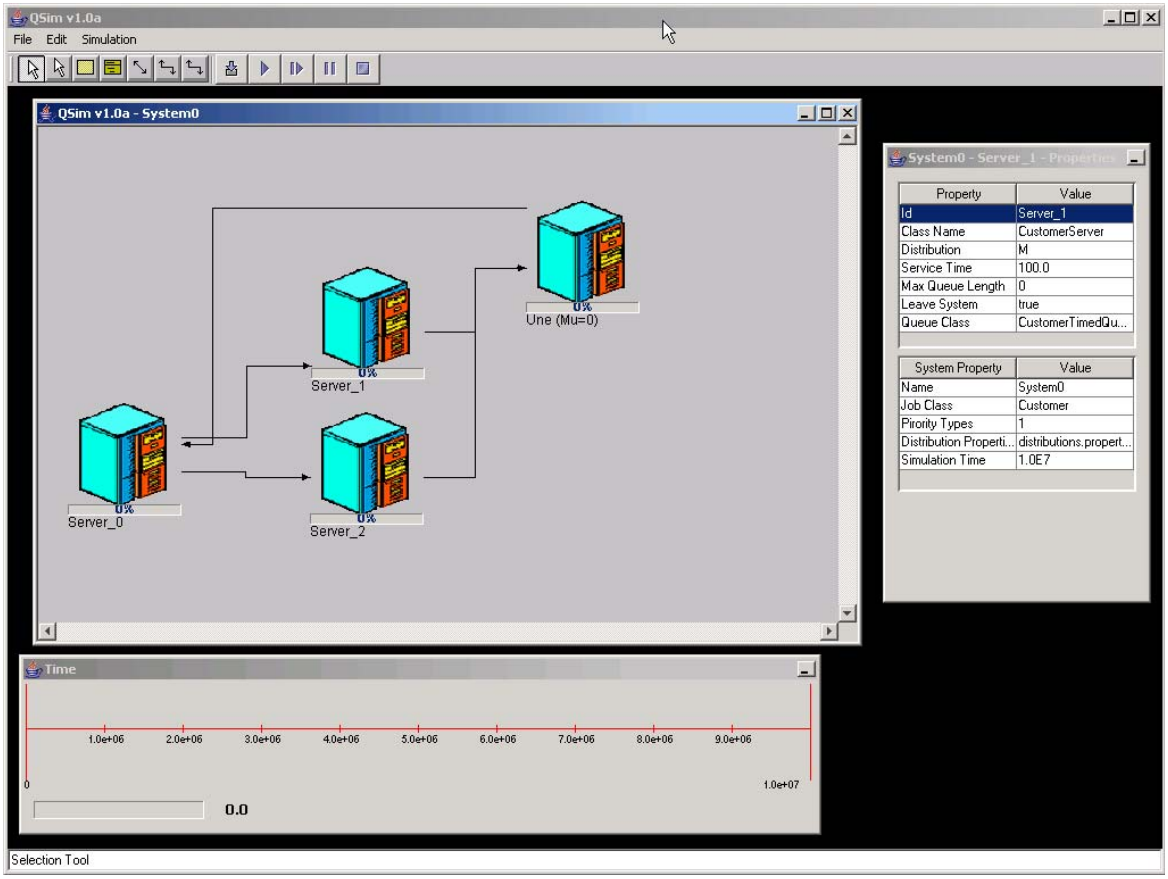


Figura 7.6: Sistema de red de colas en QSim.

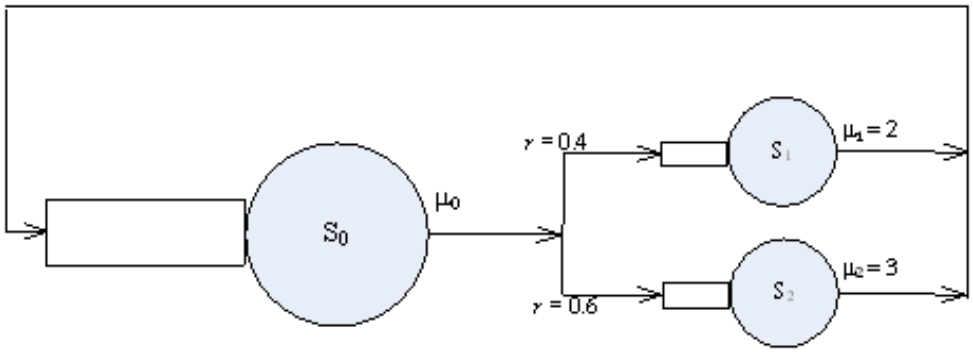


Figura 7.5: Modelo Servidor Central de prueba.

k	$G(1,k)$	$G(2,k)$	$G(3,k)$
0	1.0	1.0	1.0
1	1.0	1.8	3.6
2	1.0	2.44	8.92
3	1.0	2.952	19.008

Tabla 7.5: Resultados para la constante de normalización.

$$
\begin{aligned}
E[S_1] &= 1 \\
E[S_2] &= 2 \\
E[S_3] &= 3 \\
r_{1,2} &= 0.4 \\
r_{1,3} &= 0.6 \\
r_{2,1} &= 1 \\
r_{3,1} &= 1 \\
V_1 &= 1 \\
V_2 &= 0.4 \\
V_3 &= 0.6 \\
D_i &= V_i E[S_i] \\
D_1 &= 1 \\
D_2 &= 2(0.4) = 0.8 \\
D_3 &= 3(0.6) = 1.8
\end{aligned}
$$

donde los valores de V_i son valores dados que especifican la tasa de visita para cada servidor, es decir, esta variable expresa que por cada vez que se visite el servidor S_i el servidor S_j es visitado en promedio V_i veces. Y D_i expresa la cantidad de servicio que un trabajo necesitará en promedio de un servidor S_i en un ciclo. Inicialmente se estableció un número de 3 trabajos en la red. Los resultados para la constante $G(N)$ se muestran en la tabla 7.5.

$E[S_1]$	$\rho_1(3)$	$\rho_2(3)$	$\rho_3(3)$	$E[N_1(3)]$	$E[N_2(3)]$	$E[N_3(3)]$	$E[R_1(3)]$	$E[R_2(3)]$	$E[R_3(3)]$
1	0.4692	0.3754	0.845	0.7112	0.5235	1.7651	1.5156	2.7892	6.268
2	0.74218	0.29687	0.6679	1.1416	0.3852	1.1979	3.8182	2.595	5.3802
3	0.86804	0.23148	0.5208	1.8805	0.2850	0.8344	6.4991	2.4629	4.8065
4	0.92635	0.18521	0.41686	2.1646	0.2198	0.6155	9.3468	2.3733	4.43
5	0.9555	0.15288	0.3439	2.3451	0.1766	0.4783	12.271	2.3102	4.1717

Tabla 7.6: Resultados analíticos para servidor central.

El resultado importante del cómputo de la constante de normalización $G(N)$ es $X_j(N)$, que denota la salida del servidor j cuando hay N trabajos en la red, y de forma general del sistema:

$$X(N) = \frac{G(M, N-1)}{G(M, N)} \tag{7.15}$$

Con estos resultados se calculan las variables importantes con las siguientes expresiones:

$$\rho_i(N) = X(N)D_i \tag{7.16}$$

$$E[N_i(K)] = \sum_{k=1}^{K} D_i^k \frac{G(M, K-k)}{G(M, K)} \tag{7.17}$$

$$E[R_i(K)] = \frac{E[N_i(K)]}{X_i(K)} = \frac{E[N_i(K)]}{V_i X(K)} = \frac{\sum_{k=1}^{K} D_i^k G(M, K-k)}{V_i G(M, K-1)} \tag{7.18}$$

Con los mismos datos y la variación de $E[S_1] = \{1, 2, 3, 4, 5\}$ se obtienen los resultados de la tabla 7.6.

Para simular este sistema se utilizaron los mismos datos de entrada, se realizaron 5 simulaciones con una colección de semillas para la generación de números aleatorios igual a $1973272912L$, $747177549L$, $20464843L$, $640830765L$, $1098742207L$. Los resultados obtenidos se muestran en la tabla 7.7 (la media de las 5 simulaciones). También se muestra en la tabla 7.8 el intervalo de confianza al 90% obtenido con el simulador.

$E[S_1]$	$\rho_1(3)$	$\rho_2(3)$	$\rho_3(3)$	$E[N_1(3)]$	$E[N_2(3)]$	$E[N_3(3)]$	$E[R_1(3)]$	$E[R_2(3)]$	$E[R_3(3)]$
1	0.4693	0.3748	0.8438	0.7131	0.5231	1.7652	1.5186	2.7915	6.2691
2	0.7426	0.2961	0.6677	1.4151	0.3838	1.1979	3.8138	2.5949	5.3802
3	0.8680	0.2312	0.5242	1.8708	0.2859	0.8345	6.4495	2.4689	4.8065
4	0.9264	0.1863	0.4168	2.1615	0.2116	0.6156	9.3148	2.3904	4.4300
5	0.9554	0.1533	0.3436	2.3463	0.1773	0.4783	12.2345	2.3101	4.1717

Tabla 7.7: Resultados para servidor central obtenidos con QSim.

$E[S_1]$	$\rho_1(3)$	$\rho_2(3)$	$\rho_3(3)$	$E[N_1(3)]$	$E[N_2(3)]$	$E[N_3(3)]$	$E[R_1(3)]$	$E[R_2(3)]$	$E[R_3(3)]$
1	$1.72e^{-3}$	0.0025	$1.97e^{-3}$	$1.73e^{-3}$	$2.79e^{-3}$	0.0051	0.0016	0.0049	0.0039
2	$1.44e^{-3}$	$1.71e^{-3}$	$2.74e^{-3}$	$2.69e^{-3}$	$3.29e^{-3}$	0.0046	0.0042	0.0062	0.0119
3	$2.91e^{-3}$	$2.10e^{-3}$	$4.08e^{-3}$	$5.34e^{-3}$	$8.30e^{-4}$	$4.58e^{-3}$	0.0128	0.0104	0.0250
4	$2.45e^{-3}$	$3.05e^{-3}$	$4.50e^{-3}$	$6.88e^{-3}$	$7.38e^{-4}$	$4.11e^{-3}$	0.0167	0.0295	0.0235
5	$2.49e^{-3}$	$2.03e^{-3}$	$6.12e^{-3}$	0.0104	$6.98e^{-4}$	0.0077	0.0360	0.0180	0.0369

Tabla 7.8: Intervalos de confianza para servidor central obtenidos con QSim.

$\rho_1(3)$	$\rho_2(3)$	$\rho_3(3)$	$E[N_1(3)]$	$E[N_2(3)]$	$E[N_3(3)]$	$E[R_1(3)]$	$E[R_2(3)]$	$E[R_3(3)]$
0.0011	$5.2944e^{-4}$	$9.8508e^{-4}$	0.0578	0.0010	0.0032	0.0251	0.0051	0.0174

Tabla 7.9: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema Servidor Central.

Al igual que en los ejemplos mostrados anteriormente, la tabla 7.7 presenta resultados que no difieren en demasía en comparación a los resultados obtenidos de manera analítica. En la tabla 7.9 se muestra un promedio de la diferencia entre ambos conjuntos de resultados.

Por último se presenta una comparación entre las gráficas obtenidas (Figuras 7.7 y 7.8) para este sistema, y así concluir que la diferencia entre los resultados obtenidos analíticamente y los obtenidos por simulación, no es muy grande, así se puede dar cierto grado de confianza a las simulaciones que se hagan en el ambiente de simulación presentado en esta tesis.

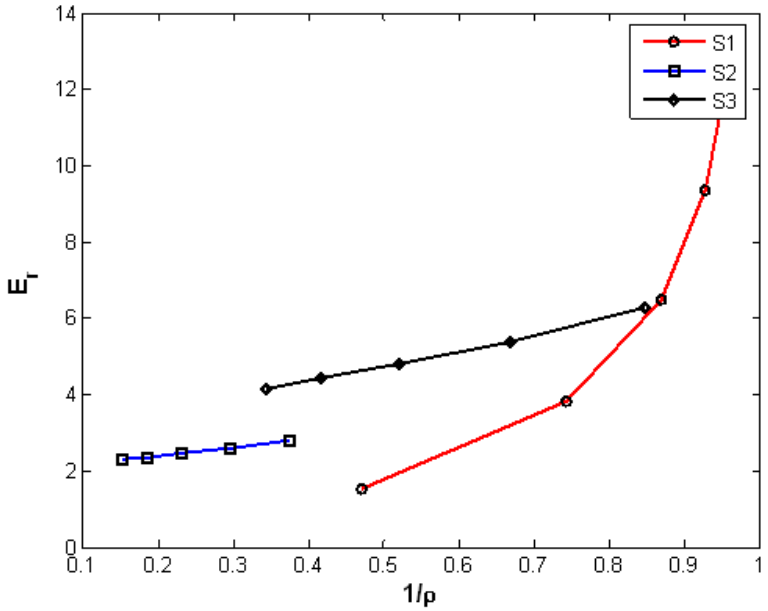


Figura 7.7: Gráfica $\rho(3)$ vs. $E[R(3)]$ del resultado analítico del sistema Servidor Central.

En el análisis de este sistema se observa que al aumentar el tiempo de servicio para un servidor, en este tipo de sistema impacta a los demás servidores interconectados ya que los trabajos que están

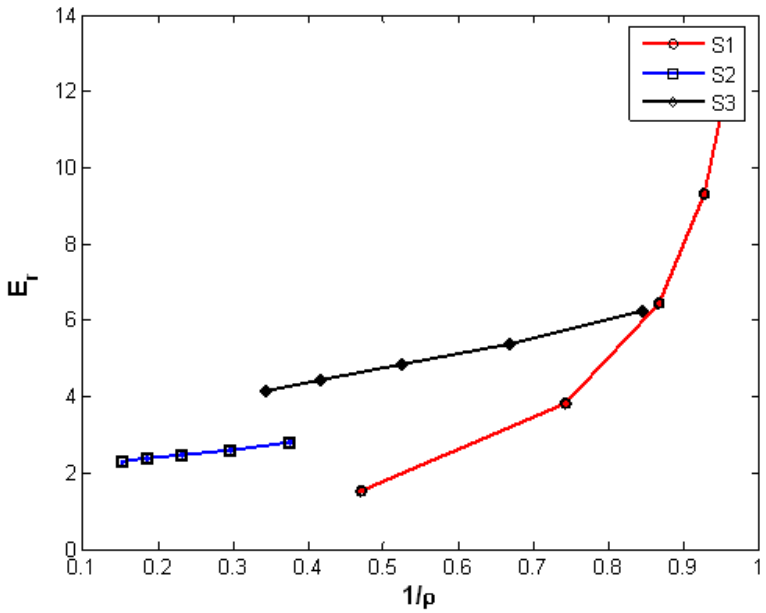


Figura 7.8: Gráfica $\rho(3)$ vs. $E[R(3)]$ del resultado obtenido con QSim del sistema Servidor Central.

fluyendo en el sistema se detienen en un nodo, lo cuál hace que en los demás servidores disminuya su ocupación y en el servidor dónde se detienen aumente. Esto denota ciertas características del sistema, por ejemplo:

- Con la simulación en el sistema se puede identificar rápidamente el cuello de botella en un sistema como éste de servidor central.
- También se puede identificar la ocupación o uso de cada servidor, y así poder balancear y optimizar el uso de cada servidor. Esto es porque en algunos sistemas como este, como podría ser una línea ensamble en donde el uso no optimizado de los servidores o maquinas de servicio (en el caso particular de una línea de ensamble) deriva en tiempo de los trabajos finales y costo de uso de cada máquina.
- Asimismo en este tipo de sistemas lo que se busca es que la cola de atención no crezca y que el tiempo de respuesta se comporte conforme a lo previsto y no se eleve desmesuradamente.

Uno de los modelos más estudiados de este tipo es el modelo de una computadora, donde se tiene una unidad de procesamiento central (CPU, que actúa como servidor central) y n entidades más de servicio, que en este caso serán los dispositivos o periféricos de entrada y salida de la computadora. El

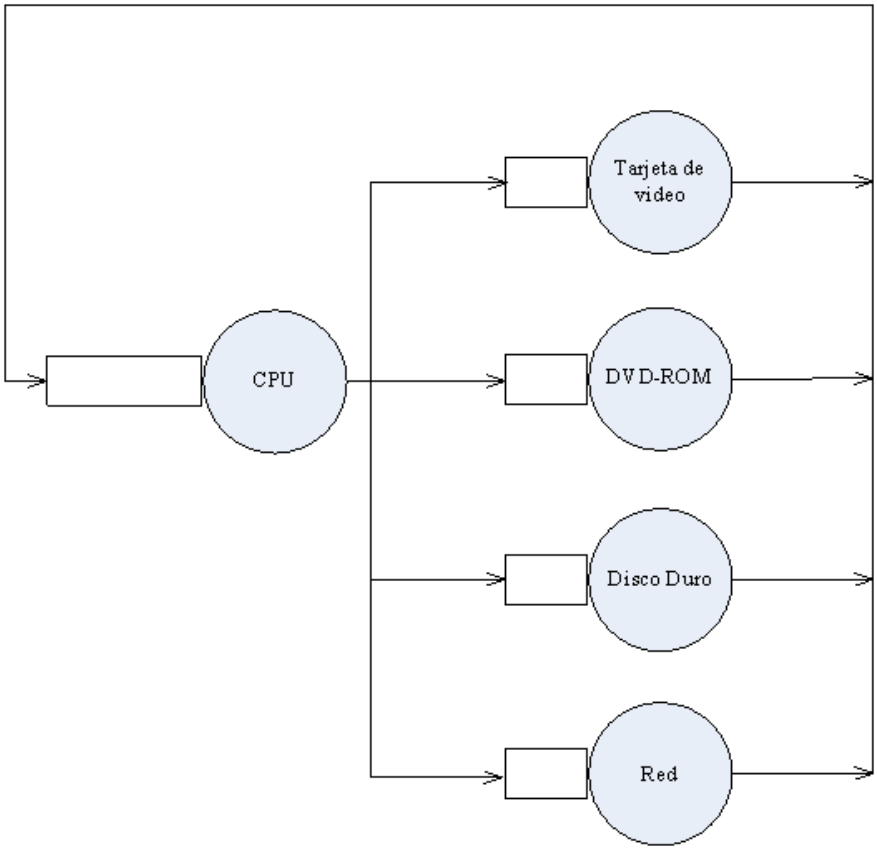


Figura 7.9: Diagrama de red de colas del funcionamiento en una computadora.

funcionamiento que se plantea es el que se tienen M tareas en rotación en el funcionamiento interno de la computadora, las cuales las atiende el CPU y después las delega a los demás periféricos de la computadora a que atiendan estas tareas, que serán en sí peticiones de entrada y salida a estos dispositivos como lo puede ser la tarjeta de video, lectores de discos, tarjetas de sonido, teclado, etc. Los resultados que se obtienen del estudio de este sistema son muy importantes, ya que se habla del desempeño de un procesador central, y muchas compañías como Intel o AMD realizan este tipo de estudios en todas sus investigaciones para nuevos desarrollos de procesadores. En la Figura 7.9 se muestra el diagrama de esta red de colas de la vida real.

.

CAPÍTULO 8

Conclusiones y trabajo futuro.

- Para llevar a cabo una simulación de un modelo real, es muy importante tomar en cuenta todo lo mencionado en esta tesis, acerca de los resultados que se pueden obtener de un modelo no conocido y las suposiciones que se hacen inicialmente. Es también importante tomar en cuenta la desviación que se tiene númericamente debido a la generación de números aleatorios y la precisión por naturaleza de la plataforma de hardware que se utilice. Finalmente el mejor punto de vista de resultados parciales de una simulación de un modelo real, la tendrá un experto que pueda saber el comportamiento aproximado de un sistema de redes de colas, o de algunos segmentos del sistema.

- En este trabajo de tesis se presentó un ambiente de desarrollo para la modelación y simulación de redes de colas que cuenta con una interfaz de usuario amigable en el lenguaje de programación *Java* lo cual permite que el programa se pueda ejecutar en diferentes plataformas y sistemas operativos. El núcleo programado lleva a cabo las simulaciones, es adaptable y se puede usar sin la necesidad de la interfaz gráfica. Asimismo permite la modelación de sistemas por medio de archivos XML. La simulación se basa en un simulador de eventos discretos los cuales son administrados por clases internas propias. Por último también se proveen las clases bases para la generación de números aleatorios y algunos generadores de números aleatorios con ciertas distribuciónes probabilísticas.

- Los resultados obtenidos por el núcleo de simulación son acercados (un .5 % de diferencia máximo) a los resultados obtenidos analíticamente, y permiten el análisis de sistemas muy complejos que pueden no ser si quiera analizados de forma analítica. Así este sistema representa una buena opción para la experimentación de este tipo de sistemas. Además de manera gratuita y la ventaja

de poder extenderlo o modificarlo de manera relativamente sencilla.

- La interfaz de usuario desarrollada, permite al usuario modelar sistemas muy complejos. En esta interfaz se pueden observar fácilmente fallas o posibles resultados no deseados en la práctica, como el que la cola de un servidor se sature o que un servidor no esté en uso. Además se provee de un módulo de análisis de resultados, en el cual, es posible realizar la misma simulación n veces, lo cuál es muy útil para así fijar un promedio de resultados más aproximado a la realidad. Por último también se incluye un módulo de graficación de resultados para un análisis en tiempo real del comportamiento de la simulación que se lleva a cabo.

- Hay trabajo futuro que desarrollar en el núcleo de la simulación. El más importante es el soporte para servidores dependientes de la carga, ya que existen muchos sistemas en el mundo real que funcionan de esta manera, cómo es el caso de los servidores computacionales. Otro punto a mejorar es la generación de resultados parciales en determinados puntos de la simulación, para que el usuario observe en ciertos puntos de interés el comportamiento actual del sistema. También el soportar colas con mayor número de trabajos, ya que esta propiedad esta arraigada totalmente en la computadora del usuario, ya que debido a las estructuras complejas de los objetos puede que se sature la memoria del ambiente donde se ejecute el sistema. También se podría mejorar la manera de seleccionar que datos de salida se requieren calcular ya sea desde la programación o desde el archivo de entrada XML, ya que esto podría aumentar el desempeño del tiempo de la simulación para sistemas muy complejos.

- Con respecto a la interfaz del usuario se podrían integrar muchas funcionalidades, cómo el de guardar el estado de un sistema en cualquier punto, para después poderla resumir. En un futuro se podrían generar gráficas de resultados de las múltiples simulaciones de un mismo sistema para así poderlo analizar de mejor manera. Una mejora importante sería la integración de esta interfaz a un sistema web, ya sea en un applet con llamadas a servlets con sockets o realizar una interfaz con la misma funcionalidad con llamadas a servicios web con *Adobe Flex* o *Flash* [24].

- Cabe señalar que esta aplicación, a pesar de las fallas existentes (con respecto a que puede tomar mucho tiempo realizar algunas simulaciones o que haya errores debido a problemas de memoria) esta aplicación realiza algunas simulaciones que no se pueden llevar a cabo en sistemas comerciales, como la simulación de redes de Gordon-Newell. También el sistema no está apegado a algún sistema o sector en específico (manufactura, finanzas, etc.), pero se pueden realizar adaptaciones sencillas para obtener resultados de interés relacionados con estos sectores, así el sistema puede ser muy útil en estos sectores.

- Por último se podrían realizar ciertas modificaciones o implementar nuevas clases con el uso del mismo núcleo para simular tráfico vehicular o redes de Petri. Para el primer caso se tendrían que

modificar principalmente las clases de los trabajos para que pudieran tener cierto comportamiento al llegar a un servidor. En el segundo caso se tendrían que modificar las clases de servidores, para que actúen como los estados de la red, y los trabajos ejemplificarían las acciones para llegar a cierto estado de la red.

REFERENCIAS

[1] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to operations research.* McGraw-Hill, Dubuque, Iowa, 8th edition, 2005. Frederick S. Hillier, Gerald J. Lieberman. Includes indexes.

[2] J. Sklenar. Introduction to simula, 1997. Disponible en: `http://staff.um.edu.mt/jskl1/talk.html`.

[3] John Goble. Modsim iii - a tutorial., 1997. Disponible en: `www.informs-cs.org/wsc97papers/0601.PDF`.

[4] Stephen V. Rice. Object-oriented simscript, April 18 2004. Disponible en: `www.simprocess.com/docs/O-O_Simscrip_IEEE.pdf`.

[5] John. A Miller. Jsim homepage. Disponible en: `http://chief.cs.uga.edu/~jam/jsim/`.

[6] Fred Howell. Simjava. Disponible en: `http://www.dcs.ed.ac.uk/home/hase/simjava/`.

[7] M.C. Little. Javasim home page. Disponible en: `http://javasim.ncl.ac.uk/`.

[8] William Feller. *An introduction to probability theory and its applications.* Wiley series in probability and mathematical statistics. Wiley, New York,, 3d edition, 1968. Bibliographical footnotes.

[9] John D. C. Little. A proof for the queueing formula: $L = \lambda \cdot W$.

[10] R. Haverkort Boudewijn. *Performance of Computer Communication Systems: A Model-Based Approach.* John Wiley & Sons, Inc., 1998. 521155.

[11] James Stewart. *Calculus : concepts and contexts.* Thomson Brooks/Cole, Australia Belmont, CA, 3rd edition, 2005.

[12] Paul Gerhard Hoel. *Introduction to mathematical statistics*. Wiley publication in mathematical statistics. Wiley, New York, 5th edition, 1984. Paul G. Hoel. Includes index.

[13] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908. 00063444 Cambridge University Press.

[14] Wikipedia. Normal distribution, 2006. Disponible en: `http://en.wikipedia.org/wiki/Normal_distribution`.

[15] Thomas Eggenschwiler Erich Gamma. Jhotdraw, 1996. Disponible en: `http://www.jhotdraw.org/`.

[16] Sun Microsystems. Class random, 05-Jun-06 2004. Disponible en: `http://java.sun.com/j2se/1.5.0/docs/api/java/util/Random.html`.

[17] L'Ecuyer Pierre. Software for uniform random number generation: distinguishing the good and the bad. In *Proceedings of the 33nd conference on Winter simulation*, Arlington, Virginia, 2001. IEEE Computer Society. 564139 95-105.

[18] W3C. W3c xml schema, 05-Jun-06 2006. Disponible en: `http://www.w3.org/XML/Schema`.

[19] W3C. Xml path language (xpath) version 1.0, 1999. Disponible en: `http://www.w3.org/TR/xpath`.

[20] Shawn Silverman. Java tip 131: Make a statement with javac!, 2002. Disponible en: `http://www.javaworld.com/javatips/jw-javatip131.html`.

[21] Ira Forman. Java reflection in action., 05-Jun-2006 2004. Disponible en: `http://www.developer.com/java/other/article.php/10936_3504271_1`.

[22] Gordon-Newell. Closed queueing systems with exponential servers. *Operations Research*, (15):254–263, 1967.

[23] Jeffrey P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Commun. ACM*, 16(9):527–531, 1973. 362345.

[24] Adobe. Flex, 2006. Disponible en: `www.adobe.com/es/products/flex/`.

[25] Wikipedia. Erlang distribution, 2006. Disponible en: `http://en.wikipedia.org/wiki/Erlang_distribution`.

APÉNDICE A

Propiedades de la distribución exponencial

Las características de los sistemas de colas están determinados a grandes rasgos por dos propiedades estadísticas, la distribución de probabilidad del tiempo entre llegadas y la distribución de probabilidad del tiempo de servicio. Para los sistemas de colas reales, estas distribuciones pueden tomar casi cualquier forma. Sin embargo, para formular un modelo como representación de un sistema real, es necesario especificar una forma para cada una de estas distribuciones. Para que sea útil, la forma supuesta deberá ser lo suficientemente realista para que el modelo provea predicciones razonables, y además suficientemente simple para que el modelo sea matemáticamente tratable. En base a esto, la distribución de probabilidad más importante en la teoría de colas es la distribución exponencial.

Se define la variable T que representa ya sea el tiempo entre llegadas o el tiempo de servicio. Esta variable aleatoria se dice que tiene una distribución exponencial con parámetro α si su función de densidad de probabilidad es:

$$f_T(t) = \begin{cases} \alpha e^{-\alpha t} & \text{para } t \geq 0 \\ 0 & \text{para } t < 0, \end{cases} \tag{A.1}$$

como se muestra en la Figura A.1. En este caso, las probabilidades acumuladas son:

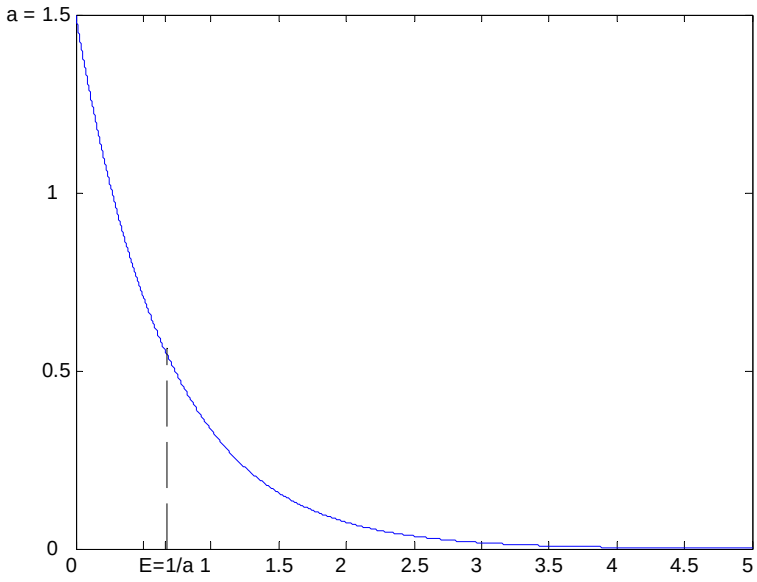


Figura A.1: Función de densidad de probabilidad de la distribución exponencial.

$$\begin{aligned} t &\geq 0 \\ P\{T &\leq t\} = 1 - e^{-\alpha t} \\ P\{T &> t\} = e^{-\alpha t} \end{aligned}$$

y el valor esperado y la varianza de T son, respectivamente:

$$\begin{aligned} E(T) &= \frac{1}{\alpha}, \\ var(T) &= \frac{1}{\alpha^2} \end{aligned}$$

Las implicaciones para suponer que T tiene una distribución exponencial para un modelo de colas se enlistan a continuación.

A.1. Propiedad 1.

$f_T(t)$ es estrictamente una función decreciente de $t(t \geq 0)$. Una consecuencia de esta propiedad es que:

$$P\{0 \leq T \leq \Delta t\} > P\{t \leq T \leq t + \Delta t\} \tag{A.2}$$

para cualquier valor estrictamente positivo de Δt y t. Esta consecuencia parte del hecho de que estas probabilidades son el área bajo la curva de $f_T(t)$ en el intervalo de longitud Δt, y el promedio de la altura de la curva es menor en la segunda probabilidad que en la primera. Entonces no es solo posible sino también relativo que T tomará un valor pequeño cercano a cero. De hecho,

$$P\left\{0 \leq T \leq \frac{1}{2}\frac{1}{\alpha}\right\} = 0.393 \tag{A.3}$$

como para

$$P\left\{\frac{1}{2}\frac{1}{\alpha} \leq T \leq \frac{3}{2}\frac{1}{\alpha}\right\} = 0.383 \tag{A.4}$$

tal que el valor que toma T es más preciso que sea "pequeño"(menor a la mitad de $E(T)$) a çercano"(no más allá de la mitad de $E(T)$) a su valor esperado, a pesar de que el segundo intervalo es el doble de longitud que el primero.

Si el servicio requerido es esencialmente idéntico para cada trabajo, es decir que el servidor siempre realice la misma secuencia de operaciones de servicio, entonces el tiempo de servicio actual tiende a estar cerca del tiempo de servicio esperado. Algunas desviaciones pequeñas pueden ocurrir usualmente por algunas variaciones menores en la eficiencia del servidor. Un tiempo de servicio pequeño lejos de la media es esencialmente imposible, por que se requiere un tiempo mínimo en la realización de las operaciones de servicio, aún cuando el servidor esté a su máxima velocidad. Claramente la distribución exponencial no provee una aproximación cercana para la distribución de tiempo de servicio para este tipo de situación.

Por el otro lado, si se considera una situación en el que el servidor atiende a trabajos que requieren de diferentes servicios dependiendo del usuario. La naturaleza del servicio puede ser la misma, pero el tipo y la cantidad de servicio puede variar. Por ejemplo, en un hospital, los doctores encuentran una gran variedad de problemas médicos. En la mayoría de los casos, pueden brindar el servicio requerido rápido, pero en algunos casos los clientes pueden requerir de un cuidado extensivo. De manera similar ocurre para los cajeros de banco o de un super-mercado que ocasionalmente el servicio se puede extender. Una distribución exponencial puede satisfacer este tipo de situación de servicio.

Por último si T representa el tiempo entre llegadas, esta propiedad modela la situación cuando un cliente potencial que se acerca al sistema de colas tiende a postponer su entrada si ve que un cliente

que entra adelante de él. Además es consistente con el fenómeno común e las llegadas que ocurren aleatoriamente.

A.2. Propiedad 2.

Ausencia de memoria.

Esta propiedad puede definirse matemáticamente como:

$$P\{T > t + \Delta t | T > \Delta t\} = P\{T > t\} \tag{A.5}$$

para cualquier t y Δt positivos. Es decir, la distribución de probabilidad del tiempo restante hasta que un evento(de llegada o servicio) ocurra siempre es la misma, sin importar el tiempo(Δt) que haya pasado. El efecto es que el proceso "se olvida"de su historia. La demostración de la ecuación A.5 se puede encontrar en la referencia [1].

Así para los tiempos entre llegadas, esta propiedad describe la situación en que el tiempo de la siguiente llegada es completamente independiente de la última llegada que ocurrió. Para los tiempos de servicio, aplica para el tipo de situación cuando cada cliente requiere de un servicio distinto, así un tiempo más largo o más corto dependerá del cliente que haya llegado.

A.3. Propiedad 3.

El mínimo de varias variables aleatorias exponenciales e independientes, tiene una distribución exponencial. Para establecer esta propiedad matemáticamente se definen $T_1, T_2, \ldots, T_n$ como variables aleatorias exponenciales e independientes con parámetros $\alpha_1, \alpha_2, \ldots, \alpha_n$ respectivamente. También se define U como una variable aleatoria que toma el valor igual al mínimo de los valores que de hecho toman $T_1, T_2, \ldots, T_n$, esto es:

$$U = \text{mín}\{T_1, T_2, \ldots, T_n\}. \tag{A.6}$$

Así, si T_i representa el tiempo hasta que un evento en particular ocurre, entonces U representa el tiempo hasta que el primer evento de n ocurre. Notesé que para cualquier $t \geq 0$,

$$P\{U > t\} = P\{T_1 > t, \ldots, T_n > t\} \tag{A.7}$$
$$= P\{T_1 > t\}P\{T_2 > t\} \ldots P\{T_n > t\} \tag{A.8}$$
$$= e^{-\alpha_1 t} e^{-\alpha_2 t} \ldots e^{\alpha_n t} \tag{A.9}$$
$$= \exp\left(-\sum_{i=1}^{n} \alpha_i t\right), \tag{A.10}$$

así U tiene una distribución exponencial con parámetro

$$\alpha = \sum_{i=1}^{n} \alpha_i \tag{A.11}$$

Esta propiedad tiene varias implicaciones para el tiempo entre llegadas. Se supone el caso donde existan n diferentes tipos de trabajos, pero el tiempo entre llegadas de cada tipo i tiene una distribución exponencial con parámetro $\alpha_i (i = 1, 2, \ldots n)$. Por la Propiedad 2, el tiempo restante de cualquier instantes específico hasta la siguiente llegada de tipo i tiene la misma distribución. Entonces, se define T_i como el tiempo restante, tomado desde el instante en que un trabajo de cualquier tipo llegue. Esta Propiedad 3 indica que U, el tiempo entre llegadas para el sistema de colas como un todo, tienen una distribución exponencial con parámetro α como se definió en la ecuación A.11. Como resultado, se puede elegir ignorar la distinción entre los clientes y aún así tener una llegada exponencial para el tiempo entre llegadas.

Esta propiedad es más importante en el caso de servidores múltiples que en el tiempo entre llegadas. Por ejemplo, se plantea la situación en que todos los servidores tengan la misma distribución para el tiempo de servicio con parámetro μ. Ahora se establece n como el número de servidores dando servicio, y T_i el tiempo de servicio restante para el servidor $i (i = 1, 2, \ldots, n)$, que también tiene una distribución exponencial con parámetro $\alpha_i = \mu$. Entonces U, el tiempo hasta el siguiente servicio de cualquier servidor, tiene una distribución exponencial con parámetro $\alpha = n\mu$. Esto marcará que el sistema de colas tiene un desempeño como si fuera un sistema de un solo servidor donde el tiempo de servicio tiene una distribución exponencial con parámetro $n\mu$.

A.4. Propiedad 4.

La relación con la distribución de Poisson.

Suponga que el tiempo entre las ocurrencias consecutivas de un evento en particular tiene una distribución exponencial con parámetro α. Esta propiedad tiene que ver con la implicación resultante

acerca de probabilidad distribución del número de veces que este tipo de evento ocurre en un tiempo especificado. Se define $X(t)$ como el número de ocurrencias en el tiempo $t(t \geq 0)$. donde el tiempo 0 designa el instante en el que la cuenta inicia. La implicación es se da como:

$$P\{X(t) = n\} = \frac{(\alpha t)^n e^{-\alpha t}}{n!}, \qquad \text{para } n = 0, 1, 2, \ldots; \tag{A.12}$$

esto es, $X(t)$ tiene un distribución de Poisson con parámetro αt. Por ejemplo, con $n = 0$,

$$P\{X(t) = 0\} = e^{-\alpha t}, \tag{A.13}$$

que es solo la probabilidad de la distribución exponencial de que el primer evento ocurra después del tiempo t. La media de esta distribución de Poisson es

$$E\{X(t)\} = \alpha t, \tag{A.14}$$

tal que el número de eventos esperados por unidad de tiempo es α. Entonces, se dice que α es la tasa promedio a la cual los eventos ocurren. Cuando estos eventos son contados con una base continua, el proceso de conteo $\{X(t); t \geq 0\}$ se dice que es un proceso de Poisson con parámetro α(la tasa promedio).

Esta propiedad provee de información útil acerca de las terminaciones de servicio cuando el tiempo de servicio tiene un distribución exponencial con parámetro μ. Esta información se obtiene por medio de la definición de $X(t)$ como el número de terminaciones de servicio logrados por un servidor continuamente ocupado en durante un tiempo t, donde $\alpha = \mu$. Para modelos con servidores múltiples, $X(t)$ puede definirse también como el número de terminaciones de servicio logrados por n servidores continuamente ocupados en un tiempo t, donde $\alpha = n\mu$.

Esta propiedad también es útil para describir el comportamiento probabilístico de las llegadas cuando el tiempo entre llegadas tiene una distribución exponencial con parámetro λ. En este caso, $X(t)$ es el número de llegadas en durante un tiempo t, donde $\alpha = \lambda$ que es la tasa de llegada promedio. Entonces, las llegadas ocurren de acuerdo a un proceso de entrada de Poisson con parámetro λ. A veces se dice que las llegadas ocurren aleatoriamente, que se traduce a que ocurren de acuerdo a un proceso de entrada de Poisson. Una interpretación de este fenómeno es que cada periodo de tiempo de tomaño fijo tiene la misma oportunidad de tener una llegada sin importar cuando ocurrió la llegada anterior.

APÉNDICE B

Manual de usuario.

El ambiente programado para esta tesis, presenta una interfaz de usuario sencilla de usar, para lo cuál es necesario el ambiente de ejecución de java (*Java Runtime Environment*) versión 1.5 como mínimo. Para ejecutar el programa se tiene que ejecutar el siguiente comando:

```
java -jar QSim.jar
```

Al ejecutar el programa se abrirá la ventana que se muestra en la figura B.1 y se describe a continuación:

- **Menú de tareas:** Esta es la barra de los menús de las tareas que se pueden realizar dentro del sistema. Aquí se econtrarán tres menús: Archivo, Edición y Simulación, que se describirán mas tarde en esta sección.

- **Barra de herramientas:** Aquí se muestran iconos con las tareas principales que se pueden realizar dentro del sistema.

- **Visor de tiempo de simulación:** En esta ventana se muestra la línea de tiempo cuando una simulación se esta lleva a cabo.

- **Inspector de propiedades:** El inspector de propiedades sirve para ingresar los datos principales de una simulación.

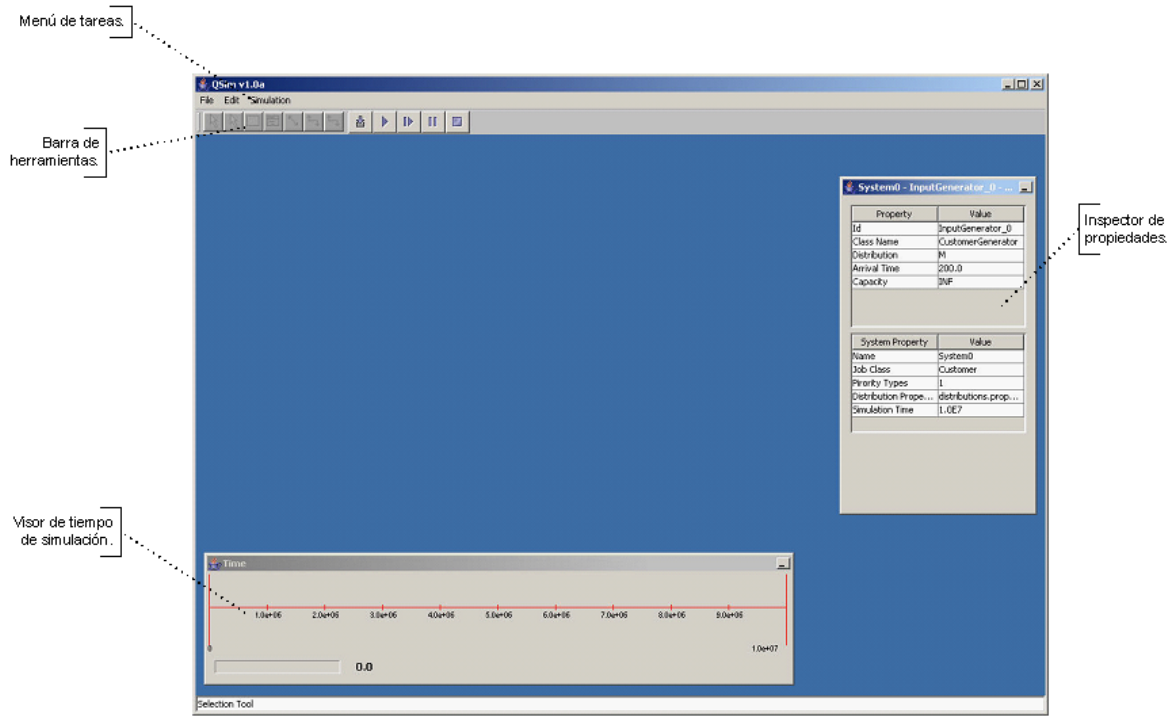


Figura B.1: Ventana principal de QSim.

B.1. Funcionamiento general.

El ambiente provee de herramientas básicas para que el usuario pueda crear un modelo de una red de colas con los parámetros requeridos por el usuario y así poder llevar a cabo la simulación de dicho modelo. Para esta función es necesario ir al menú de Archivo y dar click en la opción de Nuevo. En la ventana interna que se abrió se podrán dibujar las entidades y conexiones necesarias para la creación de un modelo, para lo cuál se encuentran los botones de Generador de entrada y Servidor en la barra de herramientas. Al presionarlos se creará una nueva entidad, la cual se podrá posicionar en donde el usuario desee siempre con la ayuda del mouse. Al posicionar las entidades, es necesario colocar las conexiones que el usuario desee con la ayuda de los botones de conectores que se encuentra en la barra de herramientas; con esta herramienta se podrán unir una o más entidades a otras como se desee para llevar a la construcción del modelo que se tenga en mente. En la figura B.2 se observa un modelo construido.

Los parámetros se cada entidad, servidor o generador de entrada, asimismo como del sistema, se pueden modificar en el inspector de propiedades.

En cada cambio del modelo, así sea en entidades o cualquier dato del sistema se deberá presionar el botón de Compilar de la barra de herramientas, para que la aplicación pueda simular el modelo

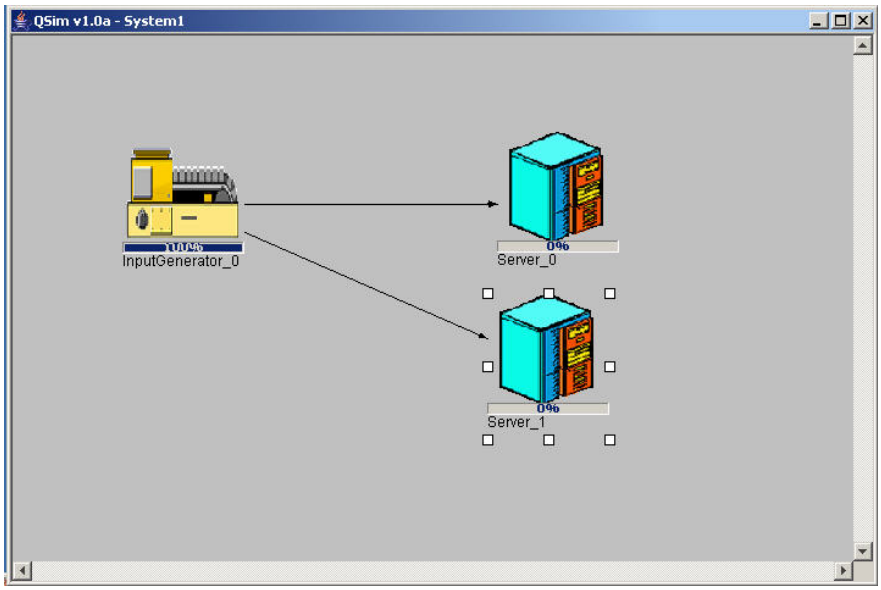


Figura B.2: Modelo creado en QSim.

creado. Al compilar con éxito un modelo se abrirá la ventana de Resultados Generales (Figura B.3) donde se podrán observar los resultados parciales y finales de la simulación seleccionada.

Si se desean ver las gráficas parciales de algunos de los resultados obtenidos, en el menú de simulación se encuentran opciones de graficación como la gráfica de Tiempo de Espera vs. Tiempo de simulación o Tamaño de la Cola vs. Tiempo de Simulación (ver Figura B.4).

B.2. Módulo de Análisis Estadístico.

En la aplicación se agregó un módulo de análisis estadístico, el cual permite realizar la simulación de un mismo modelo n veces, esto para obtener cierto nivel de confianza según se desee. En la figura B.5 se muestra la ventana de análisis estadístico en la que se pueden capturar los datos que se deseen para la simulación del modelo seleccionado.

B.3. Inspector de propiedades.

El inspector de propiedades permite la captura de datos para cada entidad del modelo creado, así como para el sistema en general, a continuación se muestra una imagen de la ventana del inspector de propiedades y los datos que se pueden capturar, dependiendo la entidad seleccionada.

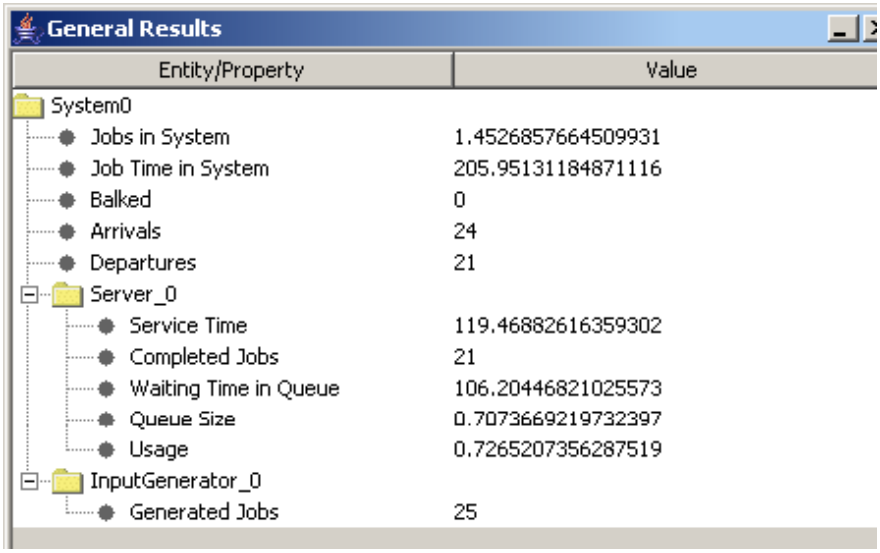


Figura B.3: Resultados Generales en QSim.

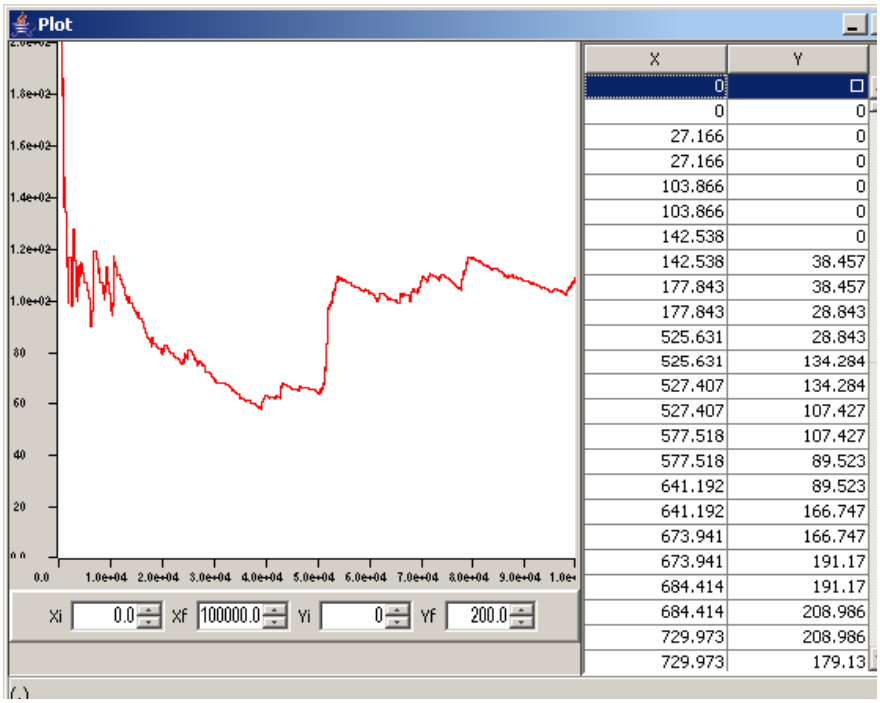


Figura B.4: Gráfica de resultados en QSim.

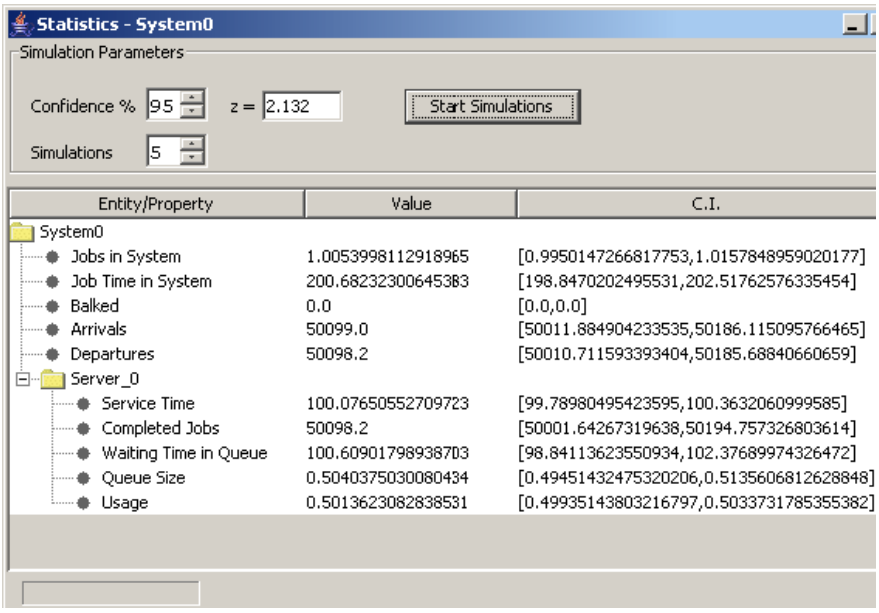


Figura B.5: Analizador Estadístico de QSim.

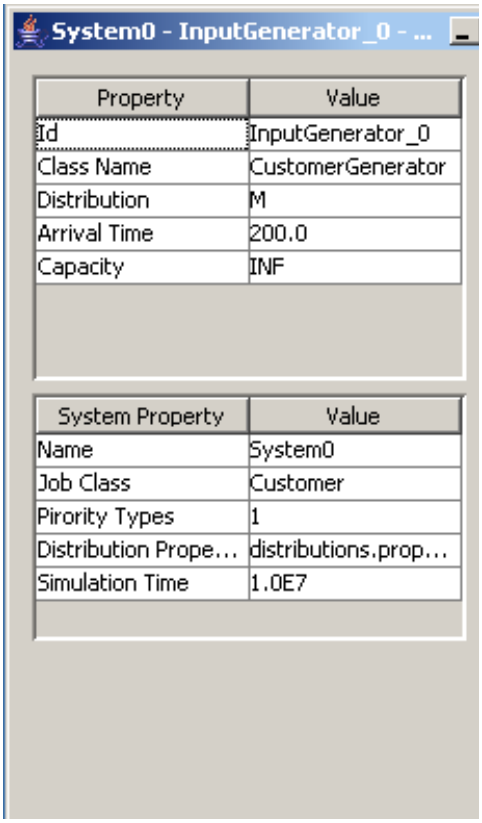


Figura B.6: Inspector de Propiedades de QSim.

1. **Datos del Sistema:**

   - **Nombre:** Indica el nombre del sistema.
   - **Clase de los trabajos:** Es la clase calificada de Java que implementa y representa un trabajo para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.Job`).
   - **Tipos de prioridad:** Son los n tipos de prioridad para cada trabajo a generar.
   - **Propiedades de Distribución:** Indica la ruta en la que se encuentra el archivo de propiedades de Java, en el cual se mapean las distribuciones a usar en el sistema y la clase de Java que implementa dicha distribución.
   - **Tiempo de simulación:** Es el tiempo total de simulación, se puede indicar en notación exponencial.

2. **Datos para un Generador de Entradas:**

   - **Id:** Identificador único de cada generador de entradas.
   - **Clase:** Es la clase calificada de Java que implementa y representa un generador de entradas para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.InputGenerator`).
   - **Distribución:** Es la distribución de cada trabajo que se genere, indicada por el identificador usado en el archivo de propiedades de distribución.
   - **Tiempo de llegada:** Es el tiempo promedio de llegada entre cada trabajo que se genere.
   - **Capacidad:** Indica la cantidad de trabajos que puede generar.

3. **Datos para un Servidor:**

   - **Id:** Identificador único de cada servidor.
   - **Clase:** Es la clase calificada de Java que implementa y representa un servidor para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.Server`).
   - **Distribución:** Es la distribución de cada trabajo que se atenderá, indicada por el identificador usado en el archivo de propiedades de distribución.
   - **Tiempo de servicio:** Es el tiempo promedio de servicio para cada trabajo que ingrese al servidor.
   - **Tamaño máximo de la cola:** Indica el límite de trabajos que se pueden encolar.
   - **Deja el sistema:** Es un dato booleano que indica si el servidor es terminal en el sistema o no.
   - **Clase de la cola:** Es la clase calificada de Java que implementa y representa una cola para el servidor(deberá implementar la interfaz `mx.ipn.cic.qsim.kernel.TimedQueue`).

- **Número de Servidores:** Indica el número de servidores de atención internos, los cuales compartirán la misma cola de espera.
- **Carga inicial:** Indica el número de trabajos que tendrá inicialmente encolados en el servidor seleccionado.

B.4. Barra de herramientas.

En esta barra se encuentran las tareas más importantes a realizar dentro del sistema. A continuación se presenta una imagen con la barra de herramientas y la descripción de cada botón:

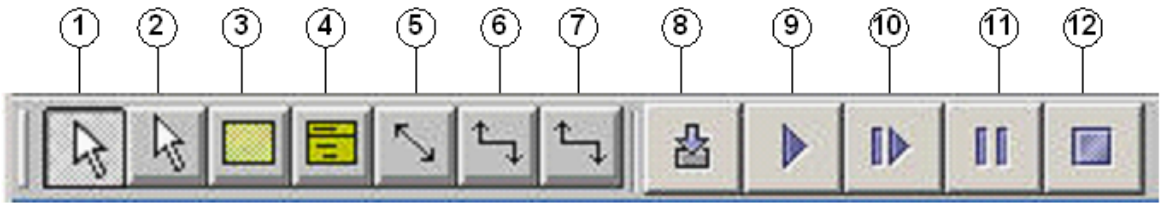


Figura B.7: Barra de herramientas de QSim.

- 1. **Seleccionar:** Esta herramienta permite seleccionar las entidades en el modelo actual.
  2. **Arrastrar y Dejar:** Permite mover fácilmente las entidades.
  3. **Generador de entradas:** Crea un nuevo generador de entradas.
  4. **Servidor:** Crea un nuevo servidor.
  5. **Conector:** Crea un conector, para lo cual se deberá dar click sobre la entidad origen, dejar presionado el botón, arrastrar el mouse y dejar sobre la entidad destino para generar la conexión.
  6. **Conector en codo:** Crea un conector con ángulos rectos según sea necesario.
  7. **Conector en partes:** Crea un conector. La diferencia principal es que para llegar a la entidad destino, se puede dar click donde se desee y en esos puntos se crearan puntos intermedios del conector.
  8. **Compilar:** Compila el modelo seleccionado.
  9. **Reproducir:** Inicia la simulación del modelo seleccionado.
  10. **Paso:** Realiza un paso de la simulación del modelo seleccionado.
  11. **Pausa:** Detiene la simulación y la deja lista para continuarla de nuevo al presionar de nuevo reproducir.

12. **Detener:** Detiene por completo la simulación, reinicializa todos los datos y tiempo de simulación.

B.5. Menús.

Dentro de los menús se encuentran las siguientes opciones:

1. **Menú de Archivo:**

   - **Nuevo.** Crea una nueva ventana de desarrollo de un modelo.
   - **Abrir.** Abre un modelo guardado.
   - **Guardar.** Guarda un modelo creado.
   - **Imprimir.** Imprime un modelo generado.
   - **Salir.** Sale de la aplicación.

2. **Menú de Edición:**

   - **Seleccionar Todos:** Selecciona todas las entidades en la venta de trabajo seleccionada.
   - **Cortar:** Corta al portapapeles la(s) entidad(es) del modelo seleccionado.
   - **Copiar:**Copia al portapapeles la(s) entidad(es) del modelo seleccionado.
   - **Pegar:** Pega del portapapeles la(s) entidad(es) guardad ahí.
   - **Duplicar:** Duplica la(s) entidad(es) seleccionada(s).
   - **Borrar:** Borra la(s) entidad(es) seleccionada(s).
   - **Agrupar:** Agrupa las entidades seleccionadas.
   - **Desagrupar:** Separa las entidades del grupo seleccionado.
   - **Enviar al fondo:** Envía al fondo la(s) entidad(es) seleccionada.
   - **Traer al frente:** Trae al frente la(s) entidad(es) seleccionada.
   - **Deshacer:** Deshace la tarea inmediata realizada.
   - **Rehacer:** Rehace la tarea inmediata realizada.

3. **Menú de Simulación:**

   - **Compilar:** Compila el modelo seleccionado.

- **Reproducir:** Realiza la simulación del modelo seleccionado.
- **Paso:** Realiza solo un ciclo de simulación del modelo seleccionado.
- **Detener:** Detiene por completo la simulación en progreso del modelo seleccionado.
- **Submenú de Estadísticas:**
  - **Analizador estadístico:** Muestra la ventana del analizador estadístico.
  - **Ver tiempo de espera vs.tiempo:** Muestra la ventana de graficación entre tiempo de espera vs. tiempo de simulación.
  - **Ver tamaño de la cola vs. tiempo:** Muestra la ventana de graficación entre tamaño de la cola vs. tiempo de simulación.

APÉNDICE C

Pruebas.

C.1. Sistema M | M | 2

Este sistema consta de un generador de trabajos y dos servidores (ver Figura C.1), los cuales tienen la particularidad que consiste en compartir la misma cola, en otras palabras se podría ver como una entidad de servicio pero que internamente tiene dos entes de servicio internos. Así este sistema y el presentado en la sección 3.3.1 son solo un caso especial del sistema M | M | s, donde s indica el número de entidades de servicio que comparten una misma cola. A continuación se presentan los datos con los que se realizaron las pruebas y sus respectivos resultados.

C.2. Sistema M | $\mathbf{E}_r$ | 1

Este sistema está compuesto por un generador de trabajos y una entidad de servicio, la principal característica es que los tiempos de llegada del generador están distribuidos conforme a una distribución Markoviana o exponencial, pero los tiempos de servicio están distribuidos conforme a una distribución de Erlang-r [25]. Los resultados se presentan a continuación.

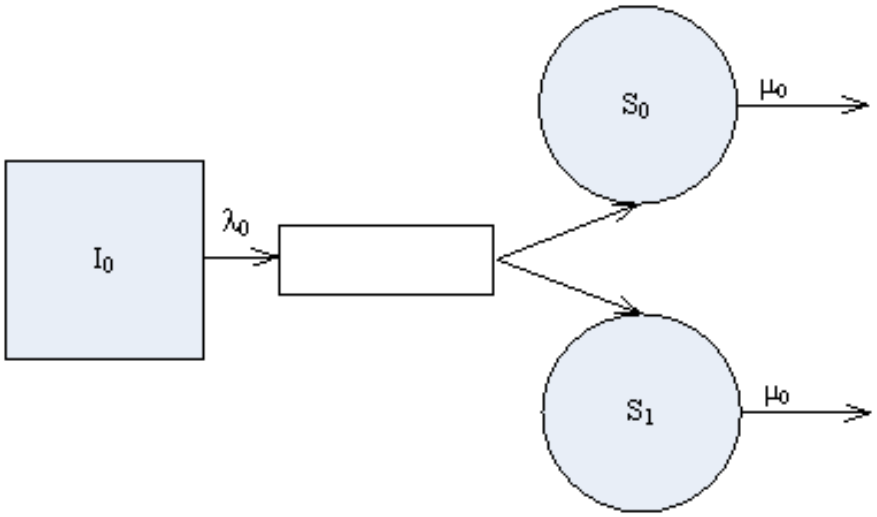


Figura C.1: Sistema M | M | 2

λ	μ	ρ	N	N_q	W_q	W
20	20	0.5	1.3333	0.3333	6.6667	26.667
30	20	0.3333	0.75	0.0833	2.5	22.5
40	20	0.25	0.5333	0.0333	1.3333	21.333
50	20	0.2	0.41667	0.0166	0.8333	20.833
60	20	0.1667	0.3428	0.0095	0.5714	20.571
70	20	0.1428	0.2916	0.0059	0.4166	20.417
80	20	0.125	0.25397	0.0039	0.3174	20.317
90	20	0.1111	0.225	0.0027	0.25	20.25
100	20	0.1	0.2020	0.0020	0.20202	20.202

Tabla C.1: Resultados analíticos para sistema M | M | 2.

λ	μ	ρ	N	N_q	W_q	W
20	20	0.4993	1.3302	0.3315	6.6329	26.6152
30	20	0.3326	0.7482	0.0829	2.4885	22.4580
40	20	0.2492	0.5314	0.0330	1.3230	21.2874
50	20	0.1991	0.4149	0.0165	0.8301	20.7781
60	20	0.1658	0.3412	0.0094	0.5679	20.4983
70	20	0.1422	0.2903	0.0058	0.4111	20.3492
80	20	0.1245	0.2530	0.0039	0.3144	20.2593
90	20	0.1107	0.2242	0.0027	0.2487	20.1985
100	20	0.0997	0.2014	0.0020	0.2017	20.1556

Tabla C.2: Resultados obtenidos con QSim, para sistema M | M | 2.

ρ	N	N_q	W_q	W
$4.77e^{-4}$	0.0021	0.3315	0.0296	0.0477
$6.00e^{-4}$	0.0027	0.0829	0.0450	0.0595
$6.04e^{-4}$	0.0017	0.3303	.0231	0.0433
$4..66e^{-4}$	0.0011	0.0165	0.0190	0.0412
$3.21e^{-4}$	$8.82e^{-4}$	0.0094	0.0202	0.0517
$3.33e^{-4}$	$7.57e^{-4}$	0.0058	0.0104	0.0473
$3.22e^{-4}$	$6.71e^{-4}$	0.0039	0.0099	0.0269
$3.41e^{-4}$	$7.05e^{-4}$	0.0027	0.0091	0.0271
$2.99e^{-4}$	$6.06e^{-4}$	0.0020	0.0079	0.0258

Tabla C.3: Intervalos de confianza obtenidos con QSim, para sistema M | M | 2.

ρ	N	N_q	W_q	W
5.9930^{-4}	0.0015	$3.1563e^{-4}$	0.0080	0.0544

Tabla C.4: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | M | 2.

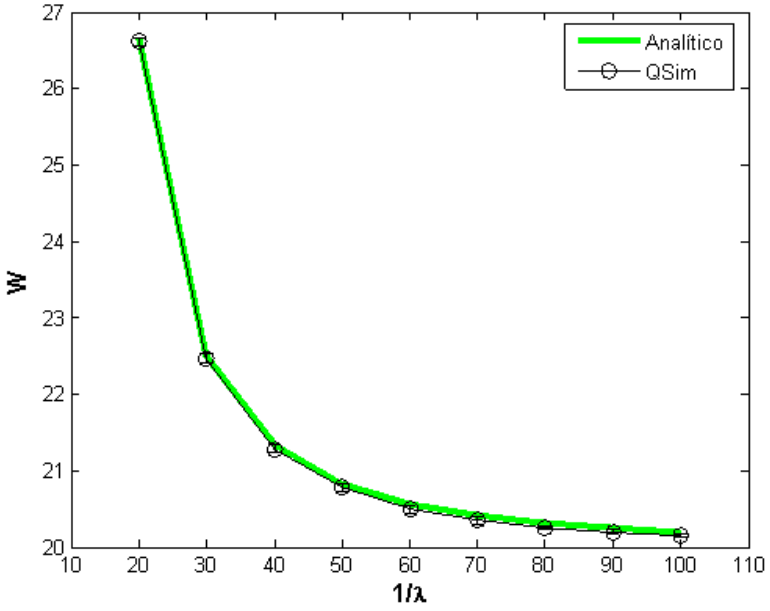


Figura C.2: Gráfica $1/\lambda$ vs. W, resultados analíticos y sobre QSim, para el sistema M | M | 2.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.4667	0.8	24.0	44.0
40	20	0.5	0.8	0.3	12.0	32.0
50	20	0.4	0.56	0.16	8.0	28.0
60	20	0.3333	0.4333	0.1	6.0	26.0
70	20	0.2857	0.3542	0.0685	4.8	24.8
80	20	0.25	0.3	0.05	4.0	24.0
90	20	0.2222	0.2603	0.0380	3.4286	23.429
100	20	0.2	0.23	0.03	3.0	23.0

Tabla C.5: Resultados analíticos para sistema M | E_r | 1.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.4643	0.7977	23.9410	43.9460
40	20	0.49967	0.7994	0.2997	11.9975	31.9962
50	20	0.3998	0.5589	0.1590	7.9566	27.9603
60	20	0.3331	0.4326	0.0995	5.9804	25.9918
70	20	0.2852	0.3536	0.0683	4.7928	24.8052
80	20	0.2495	0.2992	0.0497	3.9872	23.9971
90	20	0.2218	0.2598	0.0379	3.4250	23.4334
100	20	0.1994	0.2292	0.0298	2.9935	23.0020

Tabla C.6: Resultados obtenidos con QSim, para sistema M | E_r | 1.

ρ	N	N_q	W_q	W
0.0010	0.0100	0.0092	0.2682	0.2811
$7.4e^{-4}$	0.0021	0.0015	0.0567	0.7267
$6.20e^{-4}$	0.0016	0.0012	0.0604	0.0680
$6.26e^{-4}$	$7.34e^{-4}$	$3.15e^{-4}$	0.0190	0.0300
$6.46e^{-4}$	$8.89e^{-4}$	$4.12e^{-4}$	0.0278	0.0404
$6.12e^{-4}$	$9.49e^{-4}$	$3.50e^{-4}$	0.0234	0.0510
$5.66e^{-4}$	$7.95e^{-4}$	$2.38e^{-4}$	0.0172	0.0411
$5.24e^{-4}$	$6.48e^{-4}$	$1.46e^{-4}$	0.0130	0.0390

Tabla C.7: Intervalos de confianza obtenidos con QSim, para sistema M | E_r | 1.

ρ	N	N_q	W_q	W
$3.0713e^{-4}$	$8.9327e^{-4}$	$5.8123e^{-4}$	0.0150	0.0193

Tabla C.8: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | E_r | 1.

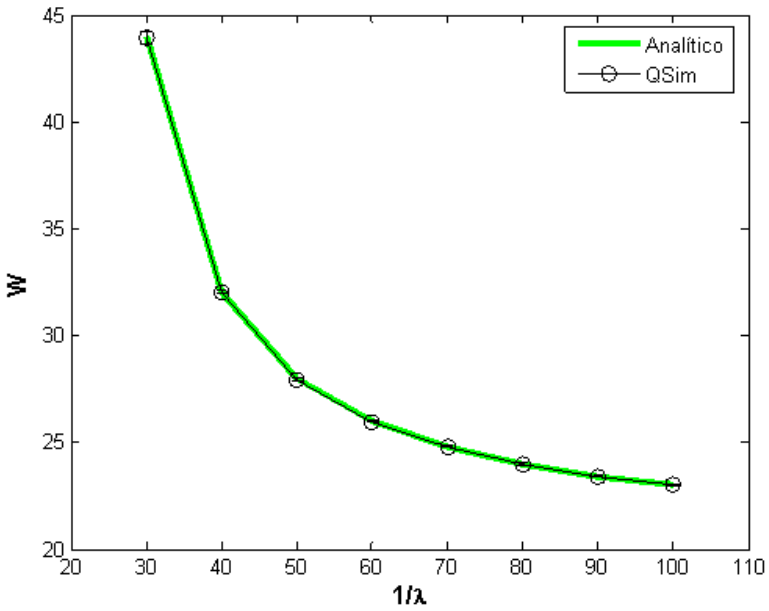


Figura C.3: Gráfica $1/\lambda$ vs. W, resultados analíticos y sobre QSim, para el sistema M | E_r | 1 .

C.3. Sistema M | G | 1

Como se mencionó en la sección 3.4 este sistema consiste de un generador de entrada y una entidad de servicio, en este caso G indica que los tiempos de servicio están distribuidos conforme a una función general, que para el caso de la prueba realizada es sobre un tiempo constante de servicio. A continuación se presentan los resultados de esta prueba.

Por último se muestra una comparación del comportamiento entre los tres sistemas en la Figura C.5, dónde se observa la gráfica de los tiempos de llegada $(1/\lambda)$ contra el uso del servidor (ρ), este tipo de análisis se puede realizar de manera rápida y sencilla con la solución de software propuesta en esta tesis, la cuál permite identificar que opción es la más adecuada conforme al desempeño y demás resultados que se esperen del sistema modelado.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.3333	0.6666	20.0	40.0
40	20	0.5	0.75	0.25	10.0	30.0
50	20	0.4	0.5333	0.1333	6.6667	26.667
60	20	0.3333	0.4166	0.08333	5.0	25.0
70	20	0.2857	0.3428	0.0571	4.0	24.0
80	20	0.25	0.2916	0.0416	3.3333	23.333
90	20	0.2222	0.2539	0.0317	2.8571	22.857
100	20	0.2	0.225	0.025	2.5	22.5

Tabla C.9: Resultados analíticos para sistema M | G | 1.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6671	1.3358	0.6684	20.0298	40.0297
40	20	0.5005	0.7514	0.2509	10.0254	30.0254
50	20	0.4004	0.5340	0.1336	6.6767	26.6766
60	20	0.3338	0.4174	0.0835	5.0051	25.0050
70	20	0.2860	0.3432	0.0572	4.0254	24.0024
80	20	0.2500	0.2916	0.0415	3.3235	23.3233
90	20	0.2222	0.2539	0.0316	2.8493	22.8490
100	20	0.2001	0.2250	0.0249	2.4911	22.4909

Tabla C.10: Resultados obtenidos con QSim para sistema M | G | 1.

ρ	N	N_q	W_q	W
0.0011	0.0048	0.0037	0.0834	0.0835
$3.46e^{-4}$	$4.57e^{-4}$	$3.35e^{-4}$	0.0156	0.0156
$3.55e^{-4}$	$8.54e^{-4}$	$6.30e^{-4}$	0.0291	0.0291
$3.36e^{-4}$	$8.66e^{-4}$	$5.67e^{-4}$	0.0299	0.0299
$2.84e^{-4}$	$5.30e^{-4}$	$2.84e^{-4}$	0.0171	0.0171
$3.12e^{-4}$	$5.70e^{-4}$	$3.26e^{-4}$	0.0238	0.0237
$3.19e^{-4}$	$6.37e^{-4}$	$3.25e^{-4}$	0.0253	0.0253
$3.72e^{-4}$	$6.19e^{-4}$	$2.52e^{-4}$	0.0208	0.0208

Tabla C.11: Intervalos de confianza obtenidos con QSim para sistema M | G | 1.

ρ	N	N_q	W_q	W
$3.5855e^{-4}$	$7.5141e^{-4}$	$4.5070e^{-4}$	0.0124	0.0124

Tabla C.12: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | G | 1.

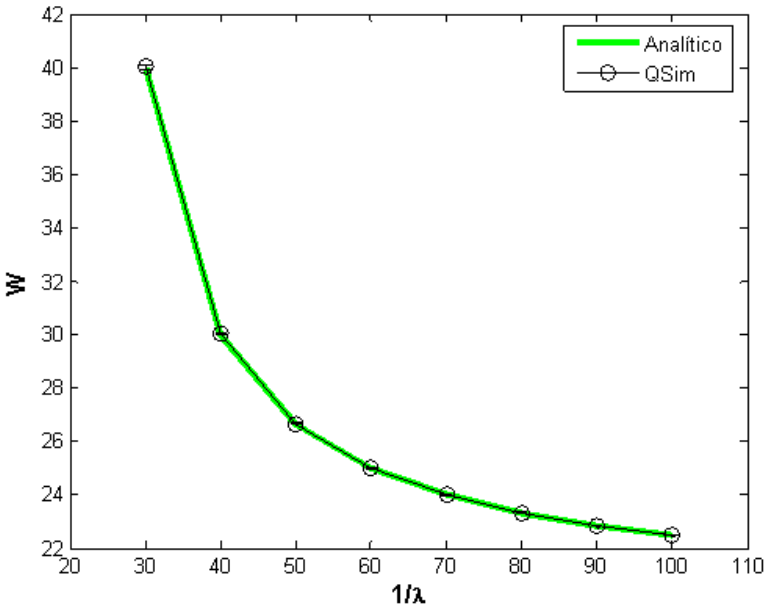


Figura C.4: Gráfica $1/\lambda$ vs. W , resultados analíticos y sobre QSim, para el sistema M | G | 1.

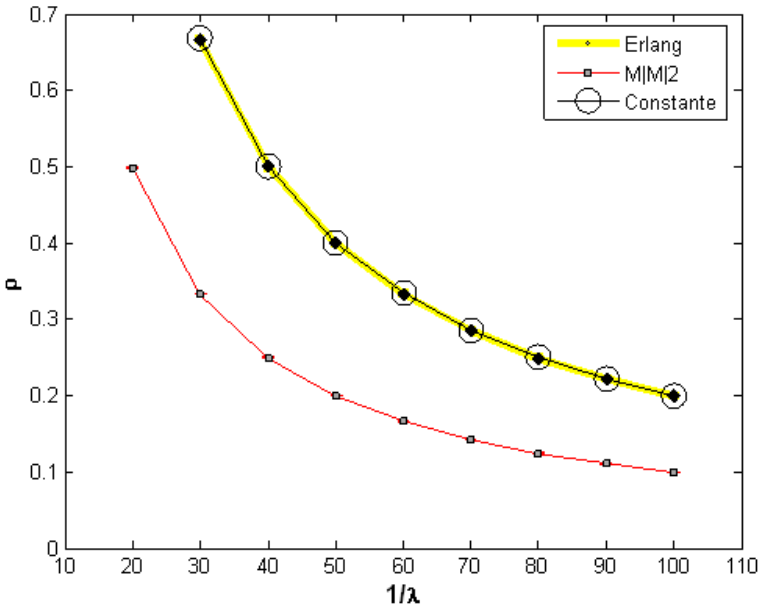


Figura C.5: Comparación entre tres sistemas (resultados obtenidos con QSim).

APÉNDICE A

Propiedades de la distribución exponencial

Las características de los sistemas de colas están determinados a grandes rasgos por dos propiedades estadísticas, la distribución de probabilidad del tiempo entre llegadas y la distribución de probabilidad del tiempo de servicio. Para los sistemas de colas reales, estas distribuciones pueden tomar casi cualquier forma. Sin embargo, para formular un modelo como representación de un sistema real, es necesario especificar una forma para cada una de estas distribuciones. Para que sea útil, la forma supuesta deberá ser lo suficientemente realista para que el modelo provea predicciones razonables, y además suficientemente simple para que el modelo sea matemáticamente tratable. En base a esto, la distribución de probabilidad más importante en la teoría de colas es la distribución exponencial.

Se define la variable T que representa ya sea el tiempo entre llegadas o el tiempo de servicio. Esta variable aleatoria se dice que tiene una distribución exponencial con parámetro α si su función de densidad de probabilidad es:

$$f_T(t) = \begin{cases} \alpha e^{-\alpha t} & \text{para } t \geq 0 \\ 0 & \text{para } t < 0, \end{cases} \tag{A.1}$$

como se muestra en la Figura A.1. En este caso, las probabilidades acumuladas son:

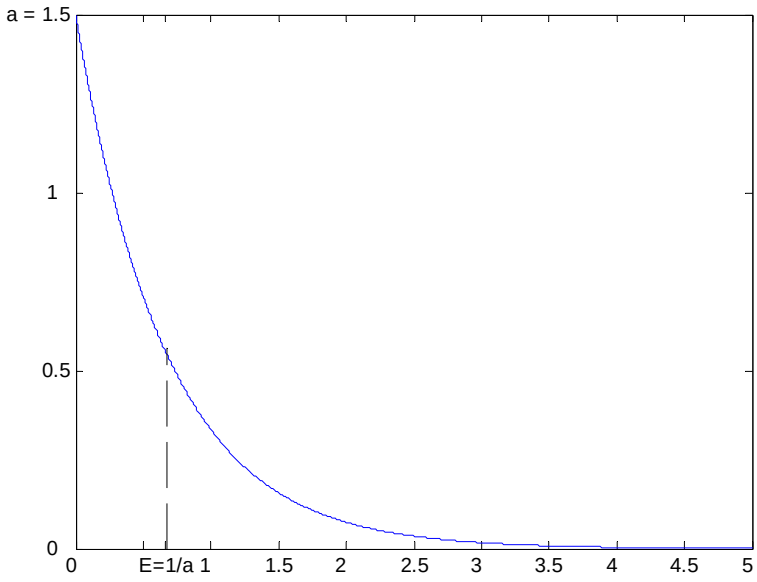


Figura A.1: Función de densidad de probabilidad de la distribución exponencial.

$$\begin{aligned} t &\geq 0 \\ P\{T &\leq t\} = 1 - e^{-\alpha t} \\ P\{T &> t\} = e^{-\alpha t} \end{aligned}$$

y el valor esperado y la varianza de T son, respectivamente:

$$\begin{aligned} E(T) &= \frac{1}{\alpha}, \\ var(T) &= \frac{1}{\alpha^2} \end{aligned}$$

Las implicaciones para suponer que T tiene una distribución exponencial para un modelo de colas se enlistan a continuación.

A.1. Propiedad 1.

$f_T(t)$ es estrictamente una función decreciente de $t(t \geq 0)$. Una consecuencia de esta propiedad es que:

$$P\{0 \leq T \leq \Delta t\} > P\{t \leq T \leq t + \Delta t\} \tag{A.2}$$

para cualquier valor estrictamente positivo de Δt y t. Esta consecuencia parte del hecho de que estas probabilidades son el área bajo la curva de $f_T(t)$ en el intervalo de longitud Δt, y el promedio de la altura de la curva es menor en la segunda probabilidad que en la primera. Entonces no es solo posible sino también relativo que T tomará un valor pequeño cercano a cero. De hecho,

$$P\left\{0 \leq T \leq \frac{1}{2}\frac{1}{\alpha}\right\} = 0.393 \tag{A.3}$$

como para

$$P\left\{\frac{1}{2}\frac{1}{\alpha} \leq T \leq \frac{3}{2}\frac{1}{\alpha}\right\} = 0.383 \tag{A.4}$$

tal que el valor que toma T es más preciso que sea "pequeño"(menor a la mitad de $E(T)$) a çercano"(no más allá de la mitad de $E(T)$) a su valor esperado, a pesar de que el segundo intervalo es el doble de longitud que el primero.

Si el servicio requerido es esencialmente idéntico para cada trabajo, es decir que el servidor siempre realice la misma secuencia de operaciones de servicio, entonces el tiempo de servicio actual tiende a estar cerca del tiempo de servicio esperado. Algunas desviaciones pequeñas pueden ocurrir usualmente por algunas variaciones menores en la eficiencia del servidor. Un tiempo de servicio pequeño lejos de la media es esencialmente imposible, por que se requiere un tiempo mínimo en la realización de las operaciones de servicio, aún cuando el servidor esté a su máxima velocidad. Claramente la distribución exponencial no provee una aproximación cercana para la distribución de tiempo de servicio para este tipo de situación.

Por el otro lado, si se considera una situación en el que el servidor atiende a trabajos que requieren de diferentes servicios dependiendo del usuario. La naturaleza del servicio puede ser la misma, pero el tipo y la cantidad de servicio puede variar. Por ejemplo, en un hospital, los doctores encuentran una gran variedad de problemas médicos. En la mayoría de los casos, pueden brindar el servicio requerido rápido, pero en algunos casos los clientes pueden requerir de un cuidado extensivo. De manera similar ocurre para los cajeros de banco o de un super-mercado que ocasionalmente el servicio se puede extender. Una distribución exponencial puede satisfacer este tipo de situación de servicio.

Por último si T representa el tiempo entre llegadas, esta propiedad modela la situación cuando un cliente potencial que se acerca al sistema de colas tiende a postponer su entrada si ve que un cliente

que entra adelante de él. Además es consistente con el fenómeno común e las llegadas que ocurren aleatoriamente.

A.2. Propiedad 2.

Ausencia de memoria.

Esta propiedad puede definirse matemáticamente como:

$$P\{T > t + \Delta t | T > \Delta t\} = P\{T > t\} \tag{A.5}$$

para cualquier t y Δt positivos. Es decir, la distribución de probabilidad del tiempo restante hasta que un evento(de llegada o servicio) ocurra siempre es la misma, sin importar el tiempo(Δt) que haya pasado. El efecto es que el proceso "se olvida"de su historia. La demostración de la ecuación A.5 se puede encontrar en la referencia [1].

Así para los tiempos entre llegadas, esta propiedad describe la situación en que el tiempo de la siguiente llegada es completamente independiente de la última llegada que ocurrió. Para los tiempos de servicio, aplica para el tipo de situación cuando cada cliente requiere de un servicio distinto, así un tiempo más largo o más corto dependerá del cliente que haya llegado.

A.3. Propiedad 3.

El mínimo de varias variables aleatorias exponenciales e independientes, tiene una distribución exponencial. Para establecer esta propiedad matemáticamente se definen $T_1, T_2, \ldots, T_n$ como variables aleatorias exponenciales e independientes con parámetros $\alpha_1, \alpha_2, \ldots, \alpha_n$ respectivamente. También se define U como una variable aleatoria que toma el valor igual al mínimo de los valores que de hecho toman $T_1, T_2, \ldots, T_n$, esto es:

$$U = \text{mín}\{T_1, T_2, \ldots, T_n\}. \tag{A.6}$$

Así, si T_i representa el tiempo hasta que un evento en particular ocurre, entonces U representa el tiempo hasta que el primer evento de n ocurre. Notesé que para cualquier $t \geq 0$,

$$P\{U > t\} = P\{T_1 > t, \ldots, T_n > t\} \tag{A.7}$$
$$= P\{T_1 > t\}P\{T_2 > t\} \ldots P\{T_n > t\} \tag{A.8}$$
$$= e^{-\alpha_1 t} e^{-\alpha_2 t} \ldots e^{\alpha_n t} \tag{A.9}$$
$$= \exp\left(-\sum_{i=1}^{n} \alpha_i t\right), \tag{A.10}$$

así U tiene una distribución exponencial con parámetro

$$\alpha = \sum_{i=1}^{n} \alpha_i \tag{A.11}$$

Esta propiedad tiene varias implicaciones para el tiempo entre llegadas. Se supone el caso donde existan n diferentes tipos de trabajos, pero el tiempo entre llegadas de cada tipo i tiene una distribución exponencial con parámetro $\alpha_i (i = 1, 2, \ldots n)$. Por la Propiedad 2, el tiempo restante de cualquier instantes específico hasta la siguiente llegada de tipo i tiene la misma distribución. Entonces, se define T_i como el tiempo restante, tomado desde el instante en que un trabajo de cualquier tipo llegue. Esta Propiedad 3 indica que U, el tiempo entre llegadas para el sistema de colas como un todo, tienen una distribución exponencial con parámetro α como se definió en la ecuación A.11. Como resultado, se puede elegir ignorar la distinción entre los clientes y aún así tener una llegada exponencial para el tiempo entre llegadas.

Esta propiedad es más importante en el caso de servidores múltiples que en el tiempo entre llegadas. Por ejemplo, se plantea la situación en que todos los servidores tengan la misma distribución para el tiempo de servicio con parámetro μ. Ahora se establece n como el número de servidores dando servicio, y T_i el tiempo de servicio restante para el servidor $i (i = 1, 2, \ldots, n)$, que también tiene una distribución exponencial con parámetro $\alpha_i = \mu$. Entonces U, el tiempo hasta el siguiente servicio de cualquier servidor, tiene una distribución exponencial con parámetro $\alpha = n\mu$. Esto marcará que el sistema de colas tiene un desempeño como si fuera un sistema de un solo servidor donde el tiempo de servicio tiene una distribución exponencial con parámetro $n\mu$.

A.4. Propiedad 4.

La relación con la distribución de Poisson.

Suponga que el tiempo entre las ocurrencias consecutivas de un evento en particular tiene una distribución exponencial con parámetro α. Esta propiedad tiene que ver con la implicación resultante

acerca de probabilidad distribución del número de veces que este tipo de evento ocurre en un tiempo especificado. Se define $X(t)$ como el número de ocurrencias en el tiempo $t(t \geq 0)$. donde el tiempo 0 designa el instante en el que la cuenta inicia. La implicación es se da como:

$$P\{X(t) = n\} = \frac{(\alpha t)^n e^{-\alpha t}}{n!}, \qquad \text{para } n = 0, 1, 2, \ldots; \tag{A.12}$$

esto es, $X(t)$ tiene un distribución de Poisson con parámetro αt. Por ejemplo, con $n = 0$,

$$P\{X(t) = 0\} = e^{-\alpha t}, \tag{A.13}$$

que es solo la probabilidad de la distribución exponencial de que el primer evento ocurra después del tiempo t. La media de esta distribución de Poisson es

$$E\{X(t)\} = \alpha t, \tag{A.14}$$

tal que el número de eventos esperados por unidad de tiempo es α. Entonces, se dice que α es la tasa promedio a la cual los eventos ocurren. Cuando estos eventos son contados con una base continua, el proceso de conteo $\{X(t); t \geq 0\}$ se dice que es un proceso de Poisson con parámetro α(la tasa promedio).

Esta propiedad provee de información útil acerca de las terminaciones de servicio cuando el tiempo de servicio tiene un distribución exponencial con parámetro μ. Esta información se obtiene por medio de la definición de $X(t)$ como el número de terminaciones de servicio logrados por un servidor continuamente ocupado en durante un tiempo t, donde $\alpha = \mu$. Para modelos con servidores múltiples, $X(t)$ puede definirse también como el número de terminaciones de servicio logrados por n servidores continuamente ocupados en un tiempo t, donde $\alpha = n\mu$.

Esta propiedad también es útil para describir el comportamiento probabilístico de las llegadas cuando el tiempo entre llegadas tiene una distribución exponencial con parámetro λ. En este caso, $X(t)$ es el número de llegadas en durante un tiempo t, donde $\alpha = \lambda$ que es la tasa de llegada promedio. Entonces, las llegadas ocurren de acuerdo a un proceso de entrada de Poisson con parámetro λ. A veces se dice que las llegadas ocurren aleatoriamente, que se traduce a que ocurren de acuerdo a un proceso de entrada de Poisson. Una interpretación de este fenómeno es que cada periodo de tiempo de tomaño fijo tiene la misma oportunidad de tener una llegada sin importar cuando ocurrió la llegada anterior.

APÉNDICE B

Manual de usuario.

El ambiente programado para esta tesis, presenta una interfaz de usuario sencilla de usar, para lo cuál es necesario el ambiente de ejecución de java (*Java Runtime Environment*) versión 1.5 como mínimo. Para ejecutar el programa se tiene que ejecutar el siguiente comando:

```
java -jar QSim.jar
```

Al ejecutar el programa se abrirá la ventana que se muestra en la figura B.1 y se describe a continuación:

- **Menú de tareas:** Esta es la barra de los menús de las tareas que se pueden realizar dentro del sistema. Aquí se econtrarán tres menús: Archivo, Edición y Simulación, que se describirán mas tarde en esta sección.

- **Barra de herramientas:** Aquí se muestran iconos con las tareas principales que se pueden realizar dentro del sistema.

- **Visor de tiempo de simulación:** En esta ventana se muestra la línea de tiempo cuando una simulación se esta lleva a cabo.

- **Inspector de propiedades:** El inspector de propiedades sirve para ingresar los datos principales de una simulación.

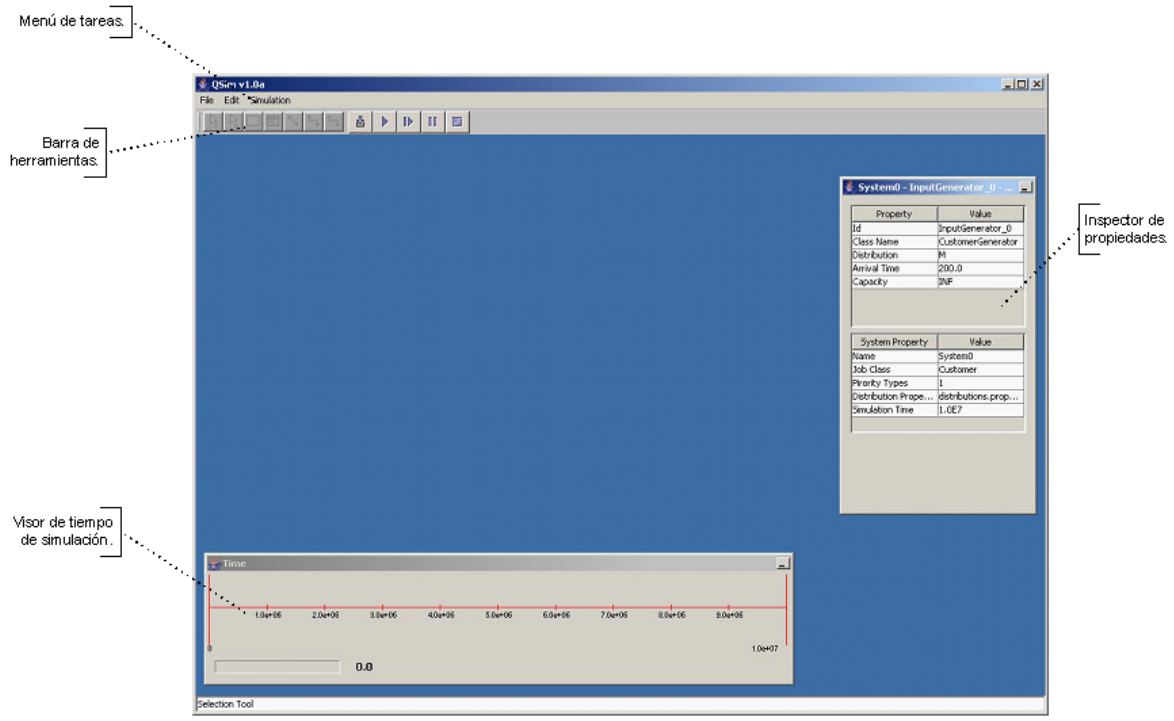


Figura B.1: Ventana principal de QSim.

B.1. Funcionamiento general.

El ambiente provee de herramientas básicas para que el usuario pueda crear un modelo de una red de colas con los parámetros requeridos por el usuario y así poder llevar a cabo la simulación de dicho modelo. Para esta función es necesario ir al menú de Archivo y dar click en la opción de Nuevo. En la ventana interna que se abrió se podrán dibujar las entidades y conexiones necesarias para la creación de un modelo, para lo cuál se encuentran los botones de Generador de entrada y Servidor en la barra de herramientas. Al presionarlos se creará una nueva entidad, la cual se podrá posicionar en donde el usuario desee siempre con la ayuda del mouse. Al posicionar las entidades, es necesario colocar las conexiones que el usuario desee con la ayuda de los botones de conectores que se encuentra en la barra de herramientas; con esta herramienta se podrán unir una o más entidades a otras como se desee para llevar a la construcción del modelo que se tenga en mente. En la figura B.2 se observa un modelo construido.

Los parámetros se cada entidad, servidor o generador de entrada, asimismo como del sistema, se pueden modificar en el inspector de propiedades.

En cada cambio del modelo, así sea en entidades o cualquier dato del sistema se deberá presionar el botón de Compilar de la barra de herramientas, para que la aplicación pueda simular el modelo

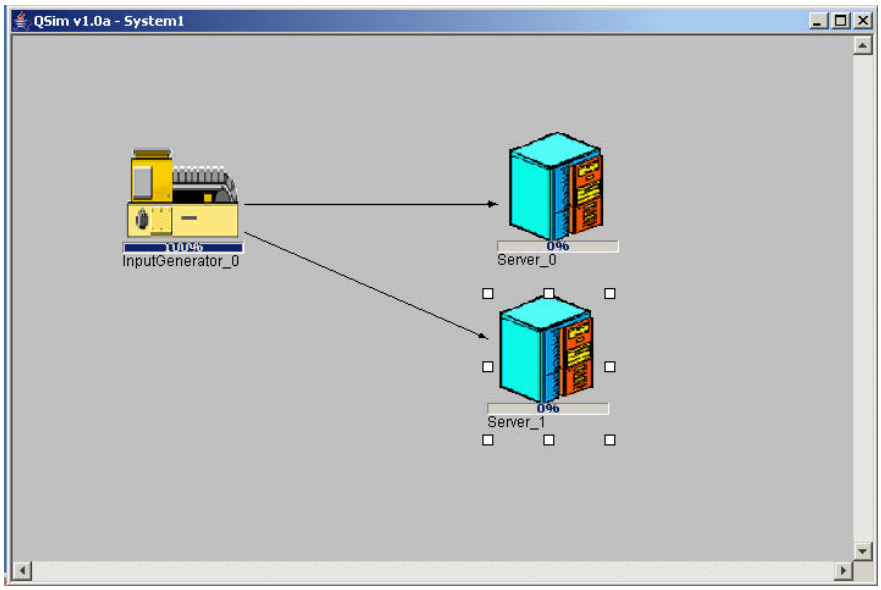


Figura B.2: Modelo creado en QSim.

creado. Al compilar con éxito un modelo se abrirá la ventana de Resultados Generales (Figura B.3) donde se podrán observar los resultados parciales y finales de la simulación seleccionada.

Si se desean ver las gráficas parciales de algunos de los resultados obtenidos, en el menú de simulación se encuentran opciones de graficación como la gráfica de Tiempo de Espera vs. Tiempo de simulación o Tamaño de la Cola vs. Tiempo de Simulación (ver Figura B.4).

B.2. Módulo de Análisis Estadístico.

En la aplicación se agregó un módulo de análisis estadístico, el cual permite realizar la simulación de un mismo modelo n veces, esto para obtener cierto nivel de confianza según se desee. En la figura B.5 se muestra la ventana de análisis estadístico en la que se pueden capturar los datos que se deseen para la simulación del modelo seleccionado.

B.3. Inspector de propiedades.

El inspector de propiedades permite la captura de datos para cada entidad del modelo creado, así como para el sistema en general, a continuación se muestra una imagen de la ventana del inspector de propiedades y los datos que se pueden capturar, dependiendo la entidad seleccionada.

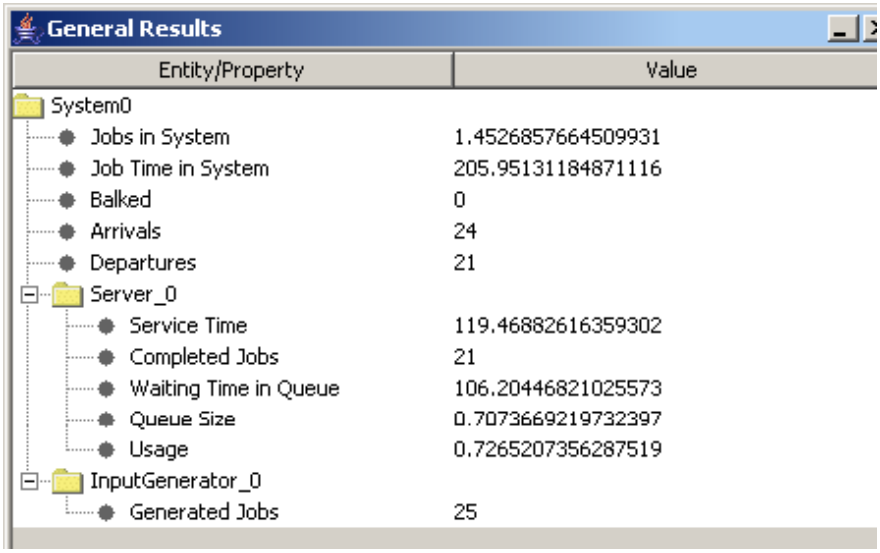


Figura B.3: Resultados Generales en QSim.

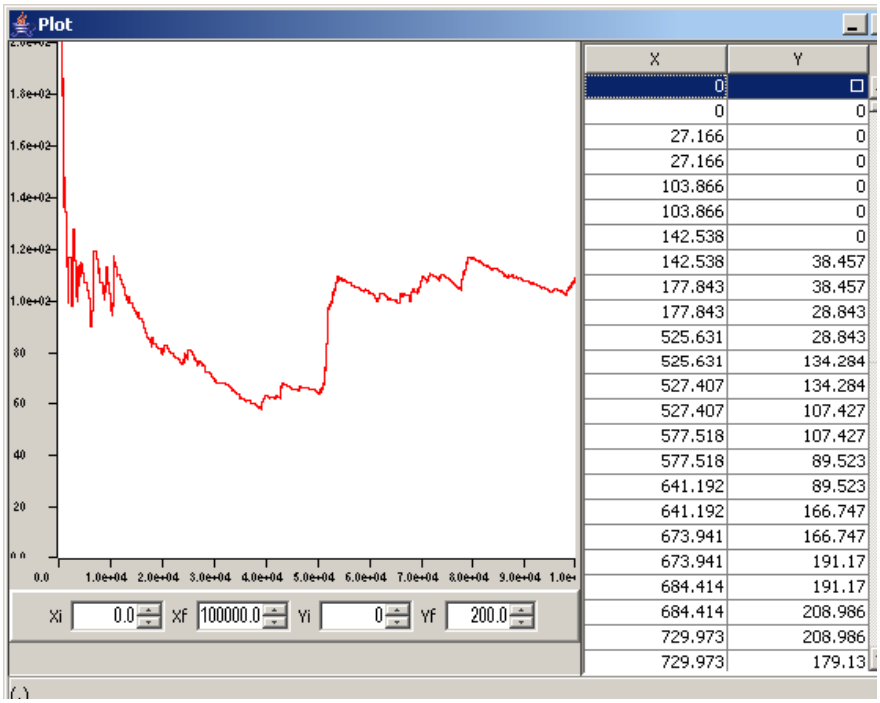


Figura B.4: Gráfica de resultados en QSim.

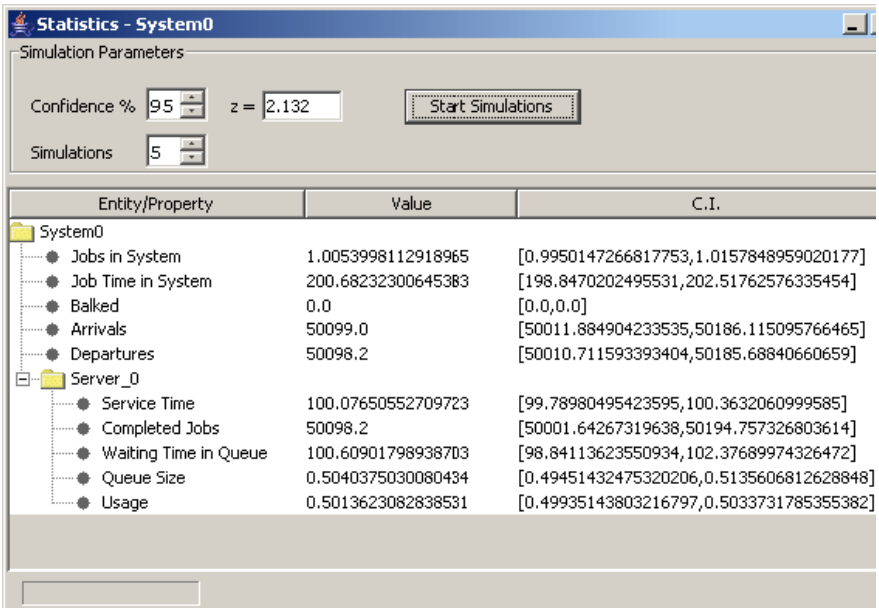


Figura B.5: Analizador Estadístico de QSim.

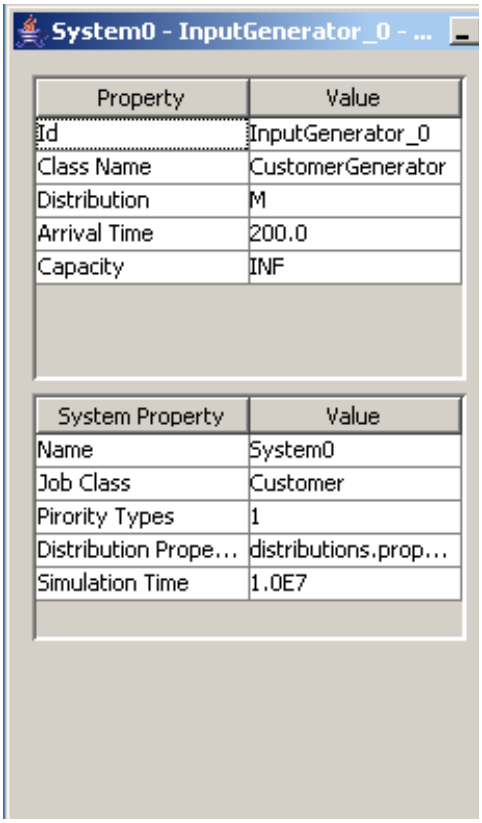


Figura B.6: Inspector de Propiedades de QSim.

1. **Datos del Sistema:**

   - **Nombre:** Indica el nombre del sistema.
   - **Clase de los trabajos:** Es la clase calificada de Java que implementa y representa un trabajo para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.Job`).
   - **Tipos de prioridad:** Son los n tipos de prioridad para cada trabajo a generar.
   - **Propiedades de Distribución:** Indica la ruta en la que se encuentra el archivo de propiedades de Java, en el cual se mapean las distribuciones a usar en el sistema y la clase de Java que implementa dicha distribución.
   - **Tiempo de simulación:** Es el tiempo total de simulación, se puede indicar en notación exponencial.

2. **Datos para un Generador de Entradas:**

   - **Id:** Identificador único de cada generador de entradas.
   - **Clase:** Es la clase calificada de Java que implementa y representa un generador de entradas para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.InputGenerator`).
   - **Distribución:** Es la distribución de cada trabajo que se genere, indicada por el identificador usado en el archivo de propiedades de distribución.
   - **Tiempo de llegada:** Es el tiempo promedio de llegada entre cada trabajo que se genere.
   - **Capacidad:** Indica la cantidad de trabajos que puede generar.

3. **Datos para un Servidor:**

   - **Id:** Identificador único de cada servidor.
   - **Clase:** Es la clase calificada de Java que implementa y representa un servidor para el sistema(deberá heredar de la clase `mx.ipn.cic.qsim.kernel.Server`).
   - **Distribución:** Es la distribución de cada trabajo que se atenderá, indicada por el identificador usado en el archivo de propiedades de distribución.
   - **Tiempo de servicio:** Es el tiempo promedio de servicio para cada trabajo que ingrese al servidor.
   - **Tamaño máximo de la cola:** Indica el límite de trabajos que se pueden encolar.
   - **Deja el sistema:** Es un dato booleano que indica si el servidor es terminal en el sistema o no.
   - **Clase de la cola:** Es la clase calificada de Java que implementa y representa una cola para el servidor(deberá implementar la interfaz `mx.ipn.cic.qsim.kernel.TimedQueue`).

- **Número de Servidores:** Indica el número de servidores de atención internos, los cuales compartirán la misma cola de espera.
- **Carga inicial:** Indica el número de trabajos que tendrá inicialmente encolados en el servidor seleccionado.

B.4. Barra de herramientas.

En esta barra se encuentran las tareas más importantes a realizar dentro del sistema. A continuación se presenta una imagen con la barra de herramientas y la descripción de cada botón:

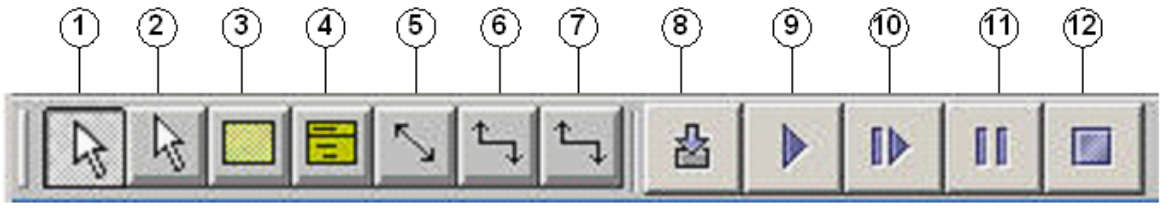


Figura B.7: Barra de herramientas de QSim.

- 1. **Seleccionar:** Esta herramienta permite seleccionar las entidades en el modelo actual.
  2. **Arrastrar y Dejar:** Permite mover fácilmente las entidades.
  3. **Generador de entradas:** Crea un nuevo generador de entradas.
  4. **Servidor:** Crea un nuevo servidor.
  5. **Conector:** Crea un conector, para lo cual se deberá dar click sobre la entidad origen, dejar presionado el botón, arrastrar el mouse y dejar sobre la entidad destino para generar la conexión.
  6. **Conector en codo:** Crea un conector con ángulos rectos según sea necesario.
  7. **Conector en partes:** Crea un conector. La diferencia principal es que para llegar a la entidad destino, se puede dar click donde se desee y en esos puntos se crearan puntos intermedios del conector.
  8. **Compilar:** Compila el modelo seleccionado.
  9. **Reproducir:** Inicia la simulación del modelo seleccionado.
  10. **Paso:** Realiza un paso de la simulación del modelo seleccionado.
  11. **Pausa:** Detiene la simulación y la deja lista para continuarla de nuevo al presionar de nuevo reproducir.

12. **Detener:** Detiene por completo la simulación, reinicializa todos los datos y tiempo de simulación.

B.5. Menús.

Dentro de los menús se encuentran las siguientes opciones:

1. **Menú de Archivo:**

   - **Nuevo.** Crea una nueva ventana de desarrollo de un modelo.
   - **Abrir.** Abre un modelo guardado.
   - **Guardar.** Guarda un modelo creado.
   - **Imprimir.** Imprime un modelo generado.
   - **Salir.** Sale de la aplicación.

2. **Menú de Edición:**

   - **Seleccionar Todos:** Selecciona todas las entidades en la venta de trabajo seleccionada.
   - **Cortar:** Corta al portapapeles la(s) entidad(es) del modelo seleccionado.
   - **Copiar:**Copia al portapapeles la(s) entidad(es) del modelo seleccionado.
   - **Pegar:** Pega del portapapeles la(s) entidad(es) guardad ahí.
   - **Duplicar:** Duplica la(s) entidad(es) seleccionada(s).
   - **Borrar:** Borra la(s) entidad(es) seleccionada(s).
   - **Agrupar:** Agrupa las entidades seleccionadas.
   - **Desagrupar:** Separa las entidades del grupo seleccionado.
   - **Enviar al fondo:** Envía al fondo la(s) entidad(es) seleccionada.
   - **Traer al frente:** Trae al frente la(s) entidad(es) seleccionada.
   - **Deshacer:** Deshace la tarea inmediata realizada.
   - **Rehacer:** Rehace la tarea inmediata realizada.

3. **Menú de Simulación:**

   - **Compilar:** Compila el modelo seleccionado.

- **Reproducir:** Realiza la simulación del modelo seleccionado.
- **Paso:** Realiza solo un ciclo de simulación del modelo seleccionado.
- **Detener:** Detiene por completo la simulación en progreso del modelo seleccionado.
- **Submenú de Estadísticas:**
  - **Analizador estadístico:** Muestra la ventana del analizador estadístico.
  - **Ver tiempo de espera vs.tiempo:** Muestra la ventana de graficación entre tiempo de espera vs. tiempo de simulación.
  - **Ver tamaño de la cola vs. tiempo:** Muestra la ventana de graficación entre tamaño de la cola vs. tiempo de simulación.

APÉNDICE C

Pruebas.

C.1. Sistema M | M | 2

Este sistema consta de un generador de trabajos y dos servidores (ver Figura C.1), los cuales tienen la particularidad que consiste en compartir la misma cola, en otras palabras se podría ver como una entidad de servicio pero que internamente tiene dos entes de servicio internos. Así este sistema y el presentado en la sección 3.3.1 son solo un caso especial del sistema M | M | s, donde s indica el número de entidades de servicio que comparten una misma cola. A continuación se presentan los datos con los que se realizaron las pruebas y sus respectivos resultados.

C.2. Sistema M | $\mathbf{E}_r$ | 1

Este sistema está compuesto por un generador de trabajos y una entidad de servicio, la principal característica es que los tiempos de llegada del generador están distribuidos conforme a una distribución Markoviana o exponencial, pero los tiempos de servicio están distribuidos conforme a una distribución de Erlang-r [25]. Los resultados se presentan a continuación.

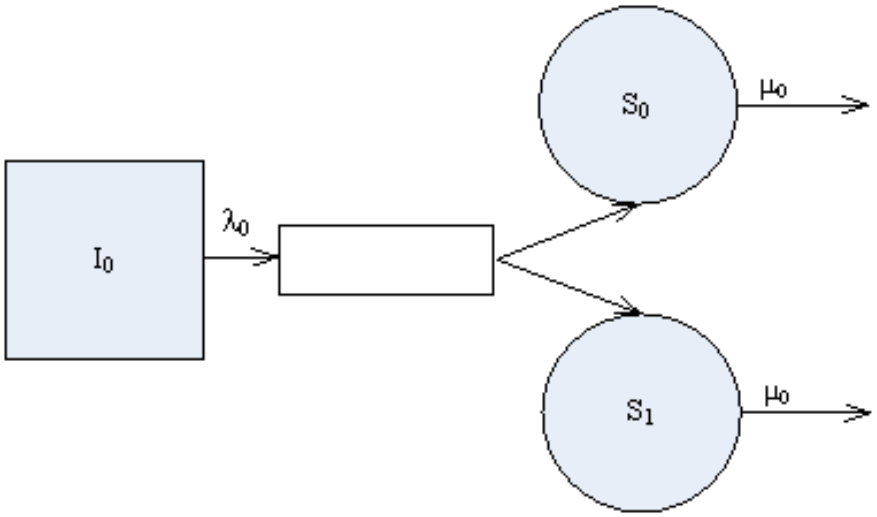


Figura C.1: Sistema M | M | 2

λ	μ	ρ	N	N_q	W_q	W
20	20	0.5	1.3333	0.3333	6.6667	26.667
30	20	0.3333	0.75	0.0833	2.5	22.5
40	20	0.25	0.5333	0.0333	1.3333	21.333
50	20	0.2	0.41667	0.0166	0.8333	20.833
60	20	0.1667	0.3428	0.0095	0.5714	20.571
70	20	0.1428	0.2916	0.0059	0.4166	20.417
80	20	0.125	0.25397	0.0039	0.3174	20.317
90	20	0.1111	0.225	0.0027	0.25	20.25
100	20	0.1	0.2020	0.0020	0.20202	20.202

Tabla C.1: Resultados analíticos para sistema M | M | 2.

λ	μ	ρ	N	N_q	W_q	W
20	20	0.4993	1.3302	0.3315	6.6329	26.6152
30	20	0.3326	0.7482	0.0829	2.4885	22.4580
40	20	0.2492	0.5314	0.0330	1.3230	21.2874
50	20	0.1991	0.4149	0.0165	0.8301	20.7781
60	20	0.1658	0.3412	0.0094	0.5679	20.4983
70	20	0.1422	0.2903	0.0058	0.4111	20.3492
80	20	0.1245	0.2530	0.0039	0.3144	20.2593
90	20	0.1107	0.2242	0.0027	0.2487	20.1985
100	20	0.0997	0.2014	0.0020	0.2017	20.1556

Tabla C.2: Resultados obtenidos con QSim, para sistema M | M | 2.

ρ	N	N_q	W_q	W
$4.77e^{-4}$	0.0021	0.3315	0.0296	0.0477
$6.00e^{-4}$	0.0027	0.0829	0.0450	0.0595
$6.04e^{-4}$	0.0017	0.3303	.0231	0.0433
$4..66e^{-4}$	0.0011	0.0165	0.0190	0.0412
$3.21e^{-4}$	$8.82e^{-4}$	0.0094	0.0202	0.0517
$3.33e^{-4}$	$7.57e^{-4}$	0.0058	0.0104	0.0473
$3.22e^{-4}$	$6.71e^{-4}$	0.0039	0.0099	0.0269
$3.41e^{-4}$	$7.05e^{-4}$	0.0027	0.0091	0.0271
$2.99e^{-4}$	$6.06e^{-4}$	0.0020	0.0079	0.0258

Tabla C.3: Intervalos de confianza obtenidos con QSim, para sistema M | M | 2.

ρ	N	N_q	W_q	W
5.9930^{-4}	0.0015	$3.1563e^{-4}$	0.0080	0.0544

Tabla C.4: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | M | 2.

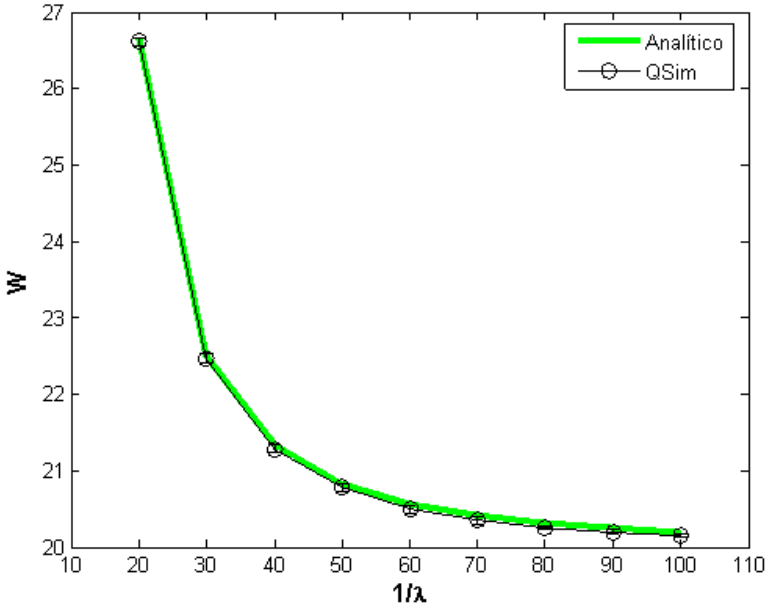


Figura C.2: Gráfica $1/\lambda$ vs. W, resultados analíticos y sobre QSim, para el sistema M | M | 2.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.4667	0.8	24.0	44.0
40	20	0.5	0.8	0.3	12.0	32.0
50	20	0.4	0.56	0.16	8.0	28.0
60	20	0.3333	0.4333	0.1	6.0	26.0
70	20	0.2857	0.3542	0.0685	4.8	24.8
80	20	0.25	0.3	0.05	4.0	24.0
90	20	0.2222	0.2603	0.0380	3.4286	23.429
100	20	0.2	0.23	0.03	3.0	23.0

Tabla C.5: Resultados analíticos para sistema M | E_r | 1.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.4643	0.7977	23.9410	43.9460
40	20	0.49967	0.7994	0.2997	11.9975	31.9962
50	20	0.3998	0.5589	0.1590	7.9566	27.9603
60	20	0.3331	0.4326	0.0995	5.9804	25.9918
70	20	0.2852	0.3536	0.0683	4.7928	24.8052
80	20	0.2495	0.2992	0.0497	3.9872	23.9971
90	20	0.2218	0.2598	0.0379	3.4250	23.4334
100	20	0.1994	0.2292	0.0298	2.9935	23.0020

Tabla C.6: Resultados obtenidos con QSim, para sistema M | E_r | 1.

ρ	N	N_q	W_q	W
0.0010	0.0100	0.0092	0.2682	0.2811
$7.4e^{-4}$	0.0021	0.0015	0.0567	0.7267
$6.20e^{-4}$	0.0016	0.0012	0.0604	0.0680
$6.26e^{-4}$	$7.34e^{-4}$	$3.15e^{-4}$	0.0190	0.0300
$6.46e^{-4}$	$8.89e^{-4}$	$4.12e^{-4}$	0.0278	0.0404
$6.12e^{-4}$	$9.49e^{-4}$	$3.50e^{-4}$	0.0234	0.0510
$5.66e^{-4}$	$7.95e^{-4}$	$2.38e^{-4}$	0.0172	0.0411
$5.24e^{-4}$	$6.48e^{-4}$	$1.46e^{-4}$	0.0130	0.0390

Tabla C.7: Intervalos de confianza obtenidos con QSim, para sistema M | E_r | 1.

ρ	N	N_q	W_q	W
$3.0713e^{-4}$	$8.9327e^{-4}$	$5.8123e^{-4}$	0.0150	0.0193

Tabla C.8: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | E_r | 1.

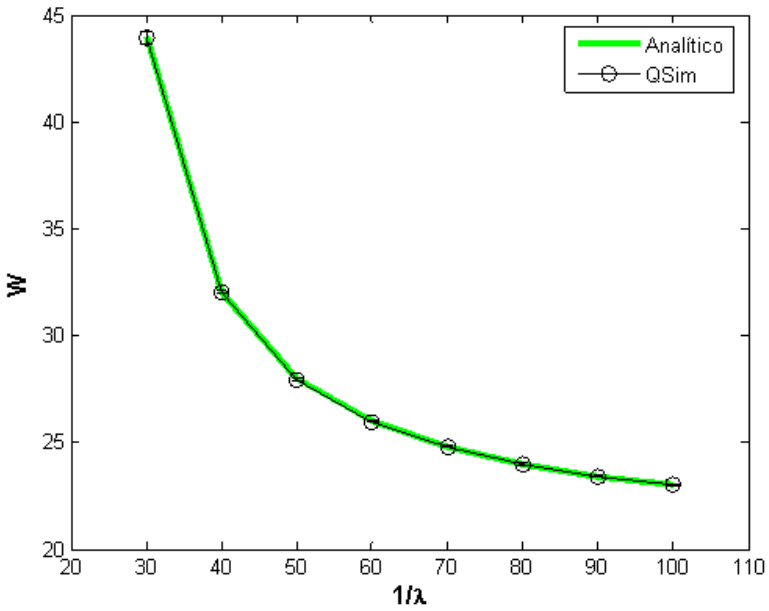


Figura C.3: Gráfica $1/\lambda$ vs. W, resultados analíticos y sobre QSim, para el sistema M | E_r | 1 .

C.3. Sistema M | G | 1

Como se mencionó en la sección 3.4 este sistema consiste de un generador de entrada y una entidad de servicio, en este caso G indica que los tiempos de servicio están distribuidos conforme a una función general, que para el caso de la prueba realizada es sobre un tiempo constante de servicio. A continuación se presentan los resultados de esta prueba.

Por último se muestra una comparación del comportamiento entre los tres sistemas en la Figura C.5, dónde se observa la gráfica de los tiempos de llegada $(1/\lambda)$ contra el uso del servidor (ρ), este tipo de análisis se puede realizar de manera rápida y sencilla con la solución de software propuesta en esta tesis, la cuál permite identificar que opción es la más adecuada conforme al desempeño y demás resultados que se esperen del sistema modelado.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6666	1.3333	0.6666	20.0	40.0
40	20	0.5	0.75	0.25	10.0	30.0
50	20	0.4	0.5333	0.1333	6.6667	26.667
60	20	0.3333	0.4166	0.08333	5.0	25.0
70	20	0.2857	0.3428	0.0571	4.0	24.0
80	20	0.25	0.2916	0.0416	3.3333	23.333
90	20	0.2222	0.2539	0.0317	2.8571	22.857
100	20	0.2	0.225	0.025	2.5	22.5

Tabla C.9: Resultados analíticos para sistema M | G | 1.

λ	μ	ρ	N	N_q	W_q	W
30	20	0.6671	1.3358	0.6684	20.0298	40.0297
40	20	0.5005	0.7514	0.2509	10.0254	30.0254
50	20	0.4004	0.5340	0.1336	6.6767	26.6766
60	20	0.3338	0.4174	0.0835	5.0051	25.0050
70	20	0.2860	0.3432	0.0572	4.0254	24.0024
80	20	0.2500	0.2916	0.0415	3.3235	23.3233
90	20	0.2222	0.2539	0.0316	2.8493	22.8490
100	20	0.2001	0.2250	0.0249	2.4911	22.4909

Tabla C.10: Resultados obtenidos con QSim para sistema M | G | 1.

ρ	N	N_q	W_q	W
0.0011	0.0048	0.0037	0.0834	0.0835
$3.46e^{-4}$	$4.57e^{-4}$	$3.35e^{-4}$	0.0156	0.0156
$3.55e^{-4}$	$8.54e^{-4}$	$6.30e^{-4}$	0.0291	0.0291
$3.36e^{-4}$	$8.66e^{-4}$	$5.67e^{-4}$	0.0299	0.0299
$2.84e^{-4}$	$5.30e^{-4}$	$2.84e^{-4}$	0.0171	0.0171
$3.12e^{-4}$	$5.70e^{-4}$	$3.26e^{-4}$	0.0238	0.0237
$3.19e^{-4}$	$6.37e^{-4}$	$3.25e^{-4}$	0.0253	0.0253
$3.72e^{-4}$	$6.19e^{-4}$	$2.52e^{-4}$	0.0208	0.0208

Tabla C.11: Intervalos de confianza obtenidos con QSim para sistema M | G | 1.

ρ	N	N_q	W_q	W
$3.5855e^{-4}$	$7.5141e^{-4}$	$4.5070e^{-4}$	0.0124	0.0124

Tabla C.12: Diferencia entre resultados analíticos y obtenidos por QSim para el sistema M | G | 1.

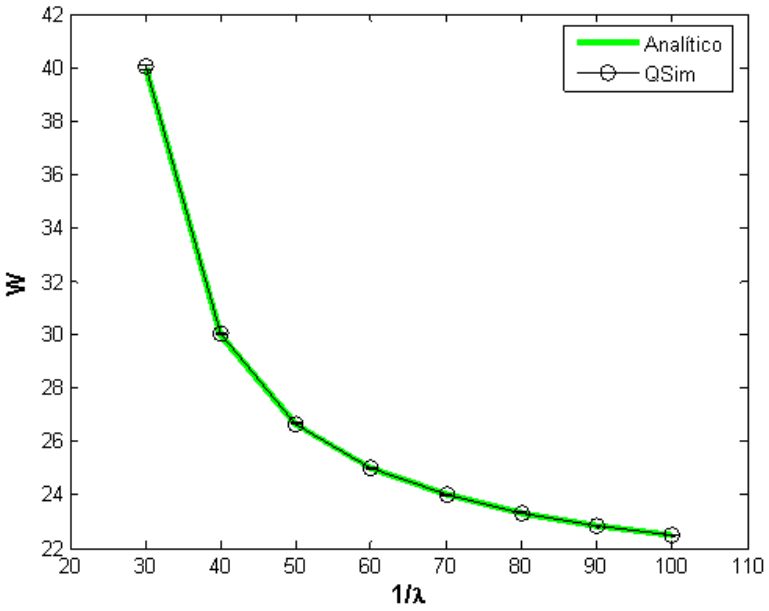


Figura C.4: Gráfica $1/\lambda$ vs. W , resultados analíticos y sobre QSim, para el sistema M | G | 1.

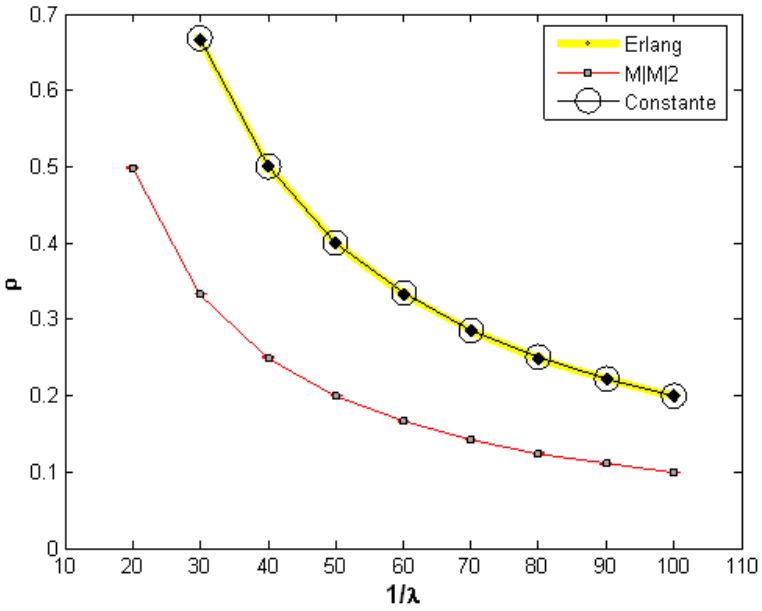


Figura C.5: Comparación entre tres sistemas (resultados obtenidos con QSim).

Glosario.

1. **Variables de Estado:** Estas variables definen el estado del sistema. Son importantes ya que si una simulación se detiene, solo se puede reiniciar después si y solo si todas las variables de estado son conocidas.

2. **Evento:** Se define como un cambio en el estado del sistema.

3. **Modelos de Tiempo Continuo o Discreto.** Un modelo en el que su estado está definido durante todo el tiempo es llamado modelo de tiempo continuo. Si solo esta definido en instantes específicos del tiempo, entonces se llamará modelo de tiempo discreto.

4. **Modelos de Estado Continuo o Discreto.** Un modelo es llamado de estado continuo o discreto dependiendo si las variables de estado son continuas o discretas. Esto es si sus variables pueden tomar cualquier infinidad de valores, entonces será un modelo de estado continuo. De manera análoga si solo pueden tomar algunos valores específicos, entonces será un modelo de estado discreto.

5. **Modelos Determinísticos o Probabilísticos.** Si se puede predecir la salida de un modelo con certeza, entonces será un modelo determinístico. Un modelo probabilística dará diferente resultado en repeticiones para un mismo conjunto de parámetros.

6. **Modelos Estáticos o Dinámicos.** Si en un modelo el tiempo no es una variable, entonces será estático. Si el estado del sistema cambia según el tiempo será dinámico.

7. **Modelos Lineales o No Lineales.** Un modelo será lineal si los parámetros de salida son una función lineal de los parámetros de entrada, de otra manera será no lineal.

8. **Modelos Abiertos o Cerrados.** Si la entrada para un modelo es externa y es independiente de éste, entonces es un modelo abierto. Si no existe una entrada será un modelo cerrado.

9. **Modelos Estables o Inestables.** Cuando el comportamiento dinámico de un modelo cesa hasta un estado quieto, y es independiente del tiempo, entonces se llama modelo estable. Si un modelo en el que su comportamiento cambia continuamente es llamado inestable.

10. **Disciplina de una cola.** Son una serie de reglas que determina el orden en que los trabajos de entrada son atendidos.