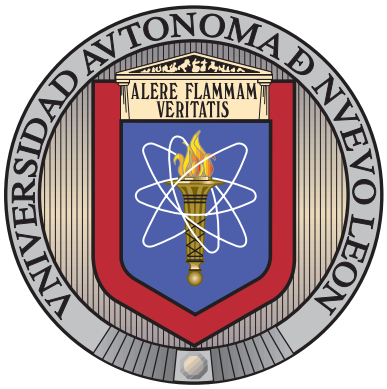
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Búsqueda de recursos para el balanceo dinámico de carga

por

David Juvencio Rios Soria

en opción al grado de

Maestro en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León julio 2009

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Búsqueda de recursos para el balanceo dinámico de carga

por

David Juvencio Rios Soria

en opción al grado de

Maestro en Ciencias

en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León julio 2009

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Búsqueda de recursos para el balanceo dinámico de carga», realizada por el alumno David Juvencio Rios Soria, con número de matrícula 1213374, sea aceptada para su defensa como opción al grado de Maestría en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

Dra. Satu Elisa Schaeffer
Asesor

Dr. Javier Bustos Jiménez
Revisor

Dr. José Arturo Berrones Santos
Revisor

Vo. Bo.

Dr. Moises Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, julio 2009

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco la oportunidad al PISIS y CONACYT por haberme permitido realizar mis estudios de maestría mediante una beca de estudios de tiempo completo.

A PROMEP por el apoyo para los proyectos 103,5/07/2523 y 103,5/08/4804.

A la Dra. Elisa Schaeffer por su paciencia, por su apoyo, confianza y compartir su tiempo y conocimientos.

A todos los maestros y alumnos de PISIS, por su disposición para ayudar a los compañeros.

Resumen

David Juvencio Rios Soria.

Candidato para el grado de Maestro en Ciencias en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Búsqueda de recursos para el balanceo dinámico de carga

Número de páginas: 89.

Objetivos y método de estudio: El objetivo de este trabajo es realizar un estudio experimental comparativo acerca de diferentes métodos para el balanceo de cargas de trabajo en redes dinámicas, esto bajo diversas condiciones ambientales tales como la estructura de la red de comunicación, la distribución de generación de carga y de las capacidades de los nodos.

El propósito de este estudio es conocer cúales son los factores que influyen en el desempeño del método de balanceo estos factores pueden ser la topología de la red o la manera en la cual se genera los trabajos.

Con el propósito de llevar a cabo este estudio, es necesario crear una herramienta computacional que permita simular el balanceo de carga en una red aplicando

diferentes métodos de generación de carga y balanceo.

Contribuciones y conclusiones: Se creó una herramienta de software que permite simular en el balanceo de carga en una red. Esta herramienta permite generar tres diferentes topologías de red, permite generar tareas en la red utilizando dos métodos diferentes de generación de carga, además permite aplicar cuatro diferentes métodos de balanceo.

Utilizando ésta herramienta se realizarón varias pruebas de diferentes métodos de balanceo para determinar bajo qué condiciones el método de balanceo tiene un mejor desempeño.

Firma del asesor: _____

Dra. Satu Elisa Schaeffer

Capítulo 1

Introducción

En la actualidad, las *redes de comunicación* están en constante crecimiento, tanto en tamaño como en complejidad; un ejemplo son las redes *punto a punto* o *P2P* (peer-to-peer en inglés) donde no se tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores de los demás nodos de la red [30]. Las redes P2P son redes que aprovechan el uso de recursos que acumulan de los demás usuarios, en una red por medio de la conectividad entre los mismos usuarios participantes de la red, obteniendo como resultado mucho más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales. Las redes P2P se basan principalmente en la idea de que todos los usuarios deben compartir recursos.

Las redes P2P tienen un alcance mundial con cientos de millones de usuarios potenciales. En general lo deseable es que cuantos más nodos estén conectados a una red P2P mejor será su funcionamiento. Esto es diferente en una arquitectura del servidor-cliente con un sistema fijo de servidores en los cuales la adición de más clientes podría significar una transferencia de datos más lenta para todos los usuarios. Cuando nodos llegan y comparten sus propios recursos en una red P2P, los recursos totales del sistema aumentan; en estos casos se pueden coordinar estos recursos para que la red trabaje como una supercomputadora [36] y de esta manera realizar las tareas de manera más rápida.

Debido al tamaño de las redes y que por lo general se desconoce la forma en que se genera carga de trabajos en éstas, es imposible poder resolver esto como un

problema tipo secuenciación (scheduling en inglés); es aquí donde entra el balanceo dinámico de carga, que es una técnica esencial que parece tener un gran potencial.

1.1 Objetivo

El objetivo de este trabajo es realizar un estudio experimental comparativo acerca de los métodos para el balanceo de cargas de trabajo en redes dinámicas, esto bajo diversas condiciones ambientales, tales como la estructura de la red de comunicación, la distribución de generación de carga y de las capacidades de los nodos. La comparación se basa en varias medidas de tiempo de terminación y de espera de los trabajos.

Para llevar a cabo este estudio, se pretende crear una herramienta que permita simular el balanceo de carga en una red, aplicando diferentes métodos de generación de carga y balanceo.

1.2 Hipótesis

La hipótesis en que se basa este trabajo es que si se tiene una red de computadoras, cada una con cierta capacidad de procesamiento y estas computadoras generan tareas con una cierta tasa, se busca que las tareas sean procesadas en el menor tiempo posible. Si el procesador no tiene suficiente capacidad para completar la tarea en un tiempo razonable, entonces se realiza una búsqueda dentro de la red para encontrar otra computadora cuyo procesador tenga capacidad actualmente disponible para esta tarea. Si se encuentra algún procesador disponible, se pasa la tarea para que sea procesada y luego devuelta. De ésta manera se aprovecha mejor los recursos de la red.

Existen varios métodos de balanceo de carga [8] y estudios de las *propiedades estructurales* [31] de las redes P2P. En esta tesis, queremos combinar estos dos ramos y desarrollar extensiones o métodos novedosos que aprovechan información ambiental

en buscar comportamiento óptimo de los nodos al balancear la carga. El desafío proviene de la naturaleza dinámica del ambiente como la distribución de las tareas que está sujeta a cambios constantes. Esperamos identificar ciertas propiedades en la topología de las redes que afectan el balanceo de carga. Esto permitiría elegir mejor a cuál nodo enviar la sobrecarga que se presenta.

Capítulo 2

Marco teórico

En este capítulo se presenta la documentación necesaria para entender el contexto de este trabajo. Se revisarán temas como sistemas distribuidos, *grids* y balanceo de carga. También veremos algunos conceptos necesarios que usaremos en este trabajo.

2.1 Conceptos

A continuación se enlistan algunos de los conceptos que estaremos utilizando durante el resto de este trabajo.

Proceso: Programa que se ejecuta en una máquina. En muchas máquinas pueden ejecutarse varios procesos a la vez.

Agente: Proceso conectado a la red. En ocasiones se denomina agente a un proceso que actúa sin control directo del usuario y que puede presentarse a la red como un usuario.

Cliente: Proceso que puede establecer conexiones a servidores y enviar peticiones a él.

Agente de usuario: Cliente que representa al usuario. A menudo tienen interfase de usuario, para que las personas puedan controlarlo directamente.

Servicio: Parte de un sistema de computadoras que gestiona una colección de recursos y presenta una funcionalidad a los usuarios y a las aplicaciones.

Servidor: Proceso que acepta conexiones de clientes y realiza servicios para ellos.

Anfitrión (host): Máquina conectada a la red que proporciona diversos servicios.

2.2 Sistemas distribuidos

Un *sistema distribuido* es un conjunto de computadoras enlazadas en una red que se comunican y coordinan sus acciones intercambiando mensajes y están soportados por aplicaciones que hacen que estos actúen como un servicio integrado [14].

El uso sistemas distribuidos tiene diferentes ventajas, son **funcionales**, esto significa que las computadoras que forman parte del sistema tienen diferentes funciones, por ejemplo pueden ser clientes o servidores. Los sistemas distribuidos son **económicos**, es más barato tener varias computadoras pequeñas que pocas muy grandes. Son **dispersos geográficamente** lo cual los hace más resistentes a fallas o ataques. La **distribución de trabajo** significa que las elementos que son parte de un sistema distribuido se reparten el trabajo para aprovechar mejor los recursos; es una de las principales ventajas y la más interesante para nuestro trabajo.

Algunas características utilizadas para definir los sistemas distribuidos son [13]:

- La ausencia de un *reloj compartido*: en un sistema distribuido es imposible sincronizar los relojes de todos los diferentes procesadores debido a que no hay certeza en los tiempos de retardo de las comunicaciones.
- La ausencia de *conocimiento global*: en un sistema distribuido es imposible para un procesador conocer el estado global del sistema. Debido a esto se vuelve difícil observar las propiedades globales del sistema.

- La ausencia de *detección exacta de fallas*: en un sistema distribuido asíncrono[1]

2.2.1 Grids

Una infraestructura o *grid* (en inglés) involucra el uso integrado y colaborativo de computadoras [9]. El concepto de grid es compartir coordinadamente recursos y resolver problemas de manera dinámica [17].

Los grid se enfocan en la autonomía, un sitio debe de tener control local sobre sus recursos, los usuarios deben tener cuentas y políticas de uso. Son heterogéneos, cualquier recurso hablando de un conjunto definido de protocolos puede ser usado. Involucran más recursos que solo computadoras y redes, por ejemplo instrumentos científicos especializados. En los grids las máquinas que son usadas para ejecutar una aplicación son elegidas desde el punto de vista del usuario, maximizando el desempeño de esa aplicación [32].

Foster [12] define a un grid como un sistema con que cuenta con las siguientes características:

- Son recursos coordinados no sujetos a un control centralizado.
- Usan protocolos e interfaces abiertos, estándares y de propósito general.
- Entregan servicios de calidad no trivial.

2.2.2 Tipos de grids

Las implementaciones actuales abarcan diversas aplicaciones, se pueden distinguir tres categorías de grids [29].

Los **grids de información** están formados por servicios de compartición de archivos. El servicio de intercambio se mantiene gracias a los participantes, es un

[1]Un sistema es asíncrono si no se conoce una cota superior para el tiempo de respuesta de los mensajes de comunicación, es difícil distinguir entre un procesador lento y una falla en el procesador.

ambiente dinámico y altamente flexible.

Los **grids de recursos** proveen mecanismos para el uso coordinado de recursos como computadoras, archivos de datos, servicios e instrumentos de laboratorio; a diferencia de la grid de información, los usuarios anónimos no pueden acceder sin credenciales.

Dentro de los grids de recursos están los **grids de datos**, los cuales proveen mecanismos para el almacenamiento seguro y redundante en sitios esparcidos geográficamente.

Por último, los **grids de servicio** entregan servicios y aplicaciones sin importar la ubicación geográfica, implementación o plataforma de hardware. La diferencia con el grid de recursos es que el el grid de servicios provee servicios abstractos sin importar su localización, mientras que el de recursos facilita accesos a recursos concretos ofrecidos en un sitio en particular.

2.2.3 Redes P2P

Una red P2P se refiere a un sistema totalmente distribuido, en donde todos los nodos son equivalentes en cuanto a su funcionalidad y a las tareas que realizan [4]. Una red P2P está compuesta por computadoras independientes que trabajan en conjunto con el propósito de compartir recursos como contenido, ciclos de procesamiento, almacenaje y ancho de banda. Existen diferentes tipos de arquitecturas para estos sistemas [22]; estas arquitecturas tienen el potencial de acelerar los procesos de comunicación y reducir los costos de colaboración.

Los sistemas P2P se pueden clasificar en tres diferentes categorías: comunicación y colaboración, computación distribuida y distribución de contenido.

1. **Centralizados**: En este tipo de estructura se tiene un *directorio central* el cual se actualiza constantemente con los nodos y sus contenidos. Un ejemplo de sistema centralizado es el sistema para compartir archivos Napster [14]. La

arquitectura de Napster constaba de índices centralizados en servidores, mientras que los archivos eran almacenados y accesados desde las computadoras personales de los usuarios. La desventaja de un sistema centralizado es que una falla en el servidor central ocasiona que no funcione la red. Las ventajas son que se realiza una búsqueda más eficiente y se puede limitar el uso del ancho de banda además de que no es necesario conocer el estado de cada nodo.

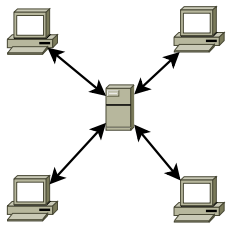


Figura 2.1: Sistema centralizado.

2. **Descentralizado pero estructurado**: No tienen un directorio central por lo que son descentralizados, pero las conexiones entre los nodos están controladas y los archivos están en lugares específicos que hacen que la búsqueda sea más fácil. Freenet [10] es un ejemplo de este tipo de estructura. Sistemas altamente estructurados [28, 37, 40] soportan el uso de *DHT* (Distributed Hash Tables en inglés) [2].

3. **Distribuido**: No se tienen directorios centrales ni control sobre la topología de la red o la ubicación de archivos. Esto hace que no se tenga un punto central de falla. Un ejemplo son las redes Gnutella [15] y Kazaa [20]. En este tipo de redes, la ubicación de los archivos, no está basada en el conocimiento de la topología. Para poder encontrar un archivo, un nodo debe buscar entre sus vecinos, por lo que las búsquedas son lentas.

[2]Una *tabla hash* o *tabla de dispersión*, es una estructura de datos que asocia claves con valores, permite el acceso a los elementos almacenados a partir de una clave generada con una función hash. Un hash resume o identifica un gran conjunto de información, dando como resultado un conjunto finito generalmente menor.

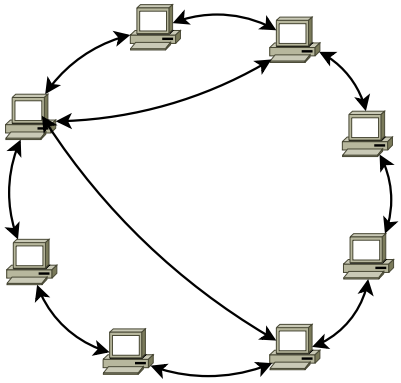


Figura 2.2: Sistema descentralizado pero estructurado.

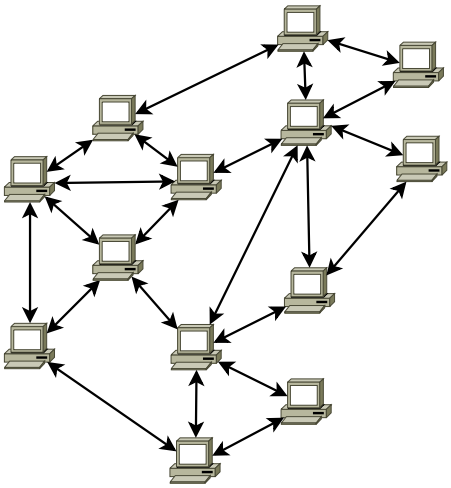


Figura 2.3: Sistema distribuido.

Sen et al. [33] analizan el tráfico que se genera en las redes tipo P2P, y llegan a la conclusión de que el tráfico generado por éstas redes no sigue distribuciones tipo *power law*, y que se necesitan desarrollar modelos más apropiados. Observaron que el tráfico generado es extremadamente variable, el 99 % del tráfico total es generado por menos del 10 % de los hosts.

2.3 Balanceo de carga

El *balanceo de carga* es distribuir tareas entre procesadores conectados en una red para equilibrar la carga de trabajo entre ellos. Un ejemplo para entender el

balanceo de carga [8] es la línea de cajas de un supermercado, donde se tienen varios servidores (cajas) y trabajos que necesitan ser procesados (carritos).

Figura 2.4: Ejemplo del supermercado.

Cuando una persona llega a la línea de de cajas debe tomar una decisión sobre en que caja formarse; para esto hay que tomar en cuenta varios aspectos: ¿Cuántas personas hay actualmente esperando?, ¿Cuántos artículos llevan?, ¿Cúal es la rapidez con la que el cajero procesa los artículos? Basados en ésta información podemos tomar la decisión de permanecer en esa caja o cambiarnos a otra para hacer más eficiente el proceso. Al hacer esto estamos realizando un balanceo de carga.

Los algoritmos utilizados para balanceo de carga pueden dividirse de dos maneras: algoritmos de *balanceo de carga dinámico* y algoritmos de *balanceo de carga estático*. Si se conoce exactamente el numero de procesadores conectados, sus tasas de servicio, las tasas de generación de las tareas y el tamaño de las tareas; se puede calcular la manera de distribuir las tareas entre los procesadores antes de que inicie el proceso; esto es balanceo de carga estático. El balanceo de carga dinámico intenta equilibrar la carga en *tiempo de ejecución*. Si no conocemos al menos alguno de estos parámetros, la tasa de servicio de los procesadores, la tasa de generación de las tarea o el tamaño de las tareas, entonces se dificulta demasiado encontrar una solución. El balanceo de carga es un tema crítico para la operación eficiente de redes P2P [18]. En este trabajo utilizaremos balanceo de carga dinámico. Esto es porque la generación

de los trabajos se realiza en cada paso, no se conoce con anticipación cúal será la carga del sistema.

Sharma et al. [34] analizan el desempeño de diferentes métodos de balanceo utilizando diferentes parámetros.

- **Rechazo de sobrecarga:** Cuando no es posible el balanceo de carga, son necesarias medidas de rechazo.
- **Tolerancia a fallas:** El algoritmo es capaz de tolerar fallas o no.
- **Precisión de previsión:** Es el grado de conformidad de los datos calculados con respecto al valor real generado después de la ejecución.
- **Estabilidad:** La estabilidad se caracteriza en terminos de los retrasos en la transferencia de información y las ganancias al obtener un rapido desempeño en una cantidad específica de tiempo.
- **Centralizado o descentralizado:** Esquemas centralizados almacenan información en un nodo designado. En un esquema distribuido cada nodo realiza el balanceo independientemente.
- **Naturaleza del algoritmo de balanceo de carga:** Balanceo estático o dinámico.
- **Cooperativo:** Si los procesadores comparten información entre ellos o no.
- **Migración de procesos:** Si el algoritmo es capaz de decidir hacer cambios en la distribución de la carga o no.
- **Utilización de recursos:** Si el algoritmo es capaz de utilizar todos los recursos más eficientemente.

En la sección 4.13 utilizamos estos parámetros para comparar nuestros métodos propuestos.

Parámetros	Round Robin	Aleatorio	Espera local	Espera central	Control central	Umbral
Rechazo de sobrecarga	No	No	Si	Si	No	No
Tolerancia a fallas	No	No	Si	Si	Si	No
Precisión de previsión	Más	Más	Menos	Menos	Más	Más
Estabilidad	Grande	Grande	Pequeña	Pequeña	Grande	Grande
Centralizado/Descentralizado	D	D	D	C	C	D
Dinámico/Estático	E	E	Di	Di	E	E
Cooperativo	No	No	Si	Si	Si	Si
Migración de procesos	No	No	Si	No	No	No
Utilización de recursos	Menos	Menos	Más	Menos	Menos	Menos

Tabla 2.1: Comparación de algoritmos de balanceo por Sharma et al. [34].

2.3.1 Medidas de desempeño

Existen varias medidas con las cuales se puede medir el desempeño de un algoritmo de balanceo de carga como:

1. **Rendimiento:** Es la medida cuantitativa de la mejora de la aplicación cuando el mecanismo controla los recursos.
2. **Eficiencia:** Son los costos producidos por el controlador de los recursos.
3. **Tiempo de retardo:** Tiempo máximo que tiene que esperar un trabajo para ser procesado.

Un algoritmo de balanceo de carga perfecto es aquel que logra obtener el mejor desempeño con el mínimo costo [8].

2.4 Teoría de grafos

Un *grafo* es un conjunto de objetos llamados *vértices* o *nodos* unidos por enlaces llamados *aristas* o *arcos*, que permiten representar relaciones entre elementos de un conjunto. Se representa gráficamente como un conjunto de puntos (vértices o nodos) unidos por líneas (aristas).

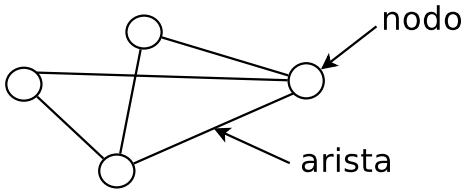


Figura 2.5: Ejemplo de un grafo con cuatro nodos y cinco aristas.

Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras. En nuestro caso, representaremos una red de computadoras mediante un grafo, en el cual los vértices representan terminales y las aristas representan conexiones. A continuación se presentan algunas definiciones importantes [11].

Grafo: Un grafo G es un par de conjuntos (V, E), donde V es un conjunto finito de puntos v_1, v_2, v_3, ..., v_n llamados vértices o nodos y E es un conjunto finito de aristas, en donde cada arista une pares ordenados de nodos.

Grado de un nodo: Si las aristas tienen un peso asignado el grado del nodo es la suma de todos los pesos de las aristas que inciden en él. Se denota por $d_i = \sum_j w_{ij}$. Es frecuente tomar

$$w_{ij} = \begin{cases} 1 & \text{si la arista } e_{ij} \text{ existe} \\ 0 & \text{en otro caso.} \end{cases}$$

en cuyo caso d_i es el número de aristas incidentes en el vértice i.

Grafo regular: G es regular de grado d si cada nodo tiene grado d, es decir, cada nodo del grafo G tiene el mismo grado.

Grafo completo: Un grafo G es llamado grafo completo si cada par de vértices está conectado por una arista. Si G tiene n nodos el número de aristas es $\frac{n(n-1)}{2}$. Este también es llamado grafo regular de grado $n-1$.

Grafo conexo: Un grafo G es conexo si existe un camino entre cualesquiera dos nodos.

Subgrafo: Sea $A(V_1, E_1)$ un grafo, A es una subgrafo de G si $V_1 \subset V$ y $E_1 \subset E$.

2.5 Modelos de red

En esta sección revisaremos algunos modelos de redes y sus características principales. Estos modelos de redes son los que utilizamos para este trabajo.

2.5.1 Modelo de Malla

Una *Malla* es una estructura donde se tienen $n \times n$ nodos y cada uno se conecta a los demás nodos que se encuentren dentro del rango de conexión; la distancia de un nodo a otro se calcula como *distancia Manhattan*, que es la cantidad de movimientos necesarios para moverse de un nodo a otro dentro de la malla. La distancia del nodo u con coordenadas (i, j) al nodo v con coordenadas (k, l) se calcula: $d(u, v) = |k - i| + |l - j|$.

En la figura 2.6 se muestra una red tipo Malla, la distancia Manhattan del nodo con coordenadas $(2, 2)$ al nodo con las coordenadas $(4, 4)$ es igual a 4.

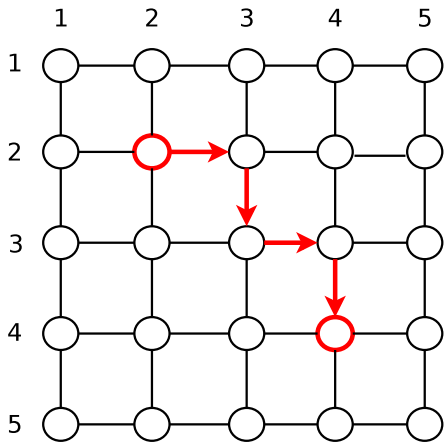


Figura 2.6: Ejemplo de distancia Manhattan en una red tipo Malla.

2.5.2 Modelo de Kleinberg

El fenómeno *Small-World* (mundo pequeño) está presente en una gran variedad de redes naturales y tecnológicas, y es una parte fundamental en la evolución de la *World Wide Web* [3]. En una red Small-World la mayoría de los nodos tienen solo unas pocas conexiones locales, mientras que una pequeña cantidad de nodos tiene conexiones de largo alcance.

Para la simulación de redes tipo Small-World utilizaremos el modelo de Kleinberg. El modelo de Kleinberg [19] es una malla de tamaño $s \times s$ Para una constante $p \geq 1$, el nodo u tiene una arista a todos los demás nodos con distancia p. Para las constantes $q \geq 0$ y $r \geq 0$, hay una arista desde u a q otros nodos. Hay una arista de u a v con probabilidad proporcional a $[d(u, v)]^{-r}$.

2.5.3 Modelo RGG

En el área de sistemas distribuidos, el estudio de los grafos aleatorios es una poderosa herramienta para entender el comportamiento de algoritmos y procesos distribuidos. También son relevantes en el estudio de las propiedades y el comportamiento de redes de gran escala como grids y redes P2P [8].

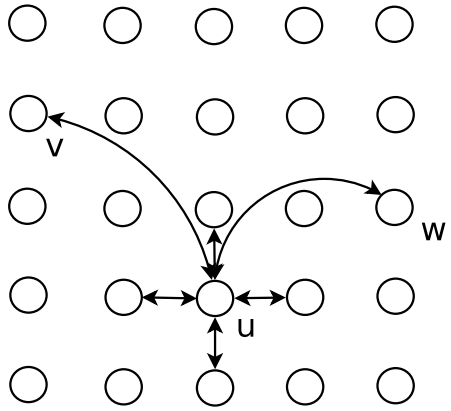


Figura 2.7: Ejemplo de red Kleinberg; se muestran los contactos del nodo u para $n = 5$, $p = 1$ y $q = 2$.

Un modelo de red de *Gráfo Geométrico Aleatorio* (RGG por sus siglas en inglés) [27], es una estructura donde se generan nodos aleatoriamente dentro de un área de 1×1 unidades, y cada nodo se conecta a los demás que se encuentren dentro de su radio de conexión. Un radio igual a $\sqrt{2}$ significa que todos los nodos están conectados con todos los demás. Con radios pequeños es más probable que el grafo obtenido sea no conexo.

Una cota superior para el *número esperado de aristas* en un RGG está dado por la ecuación 2.1. Donde r es el radio y n el número de nodos:

$$E = \frac{\pi r^2 (n-1) n}{2}. \tag{2.1}$$

Este valor obtenido nos representa una cota superior, ya que al ubicar los nodos en un área cuadrada no considera el efecto de los bordes. Ver figura 2.11.

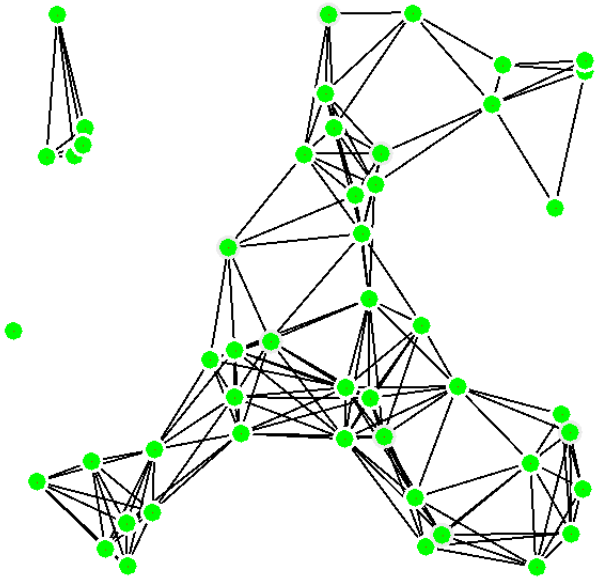


Figura 2.8: Ejemplo de un grafo aleatorio geométrico con 49 nodos y radio 0.2.

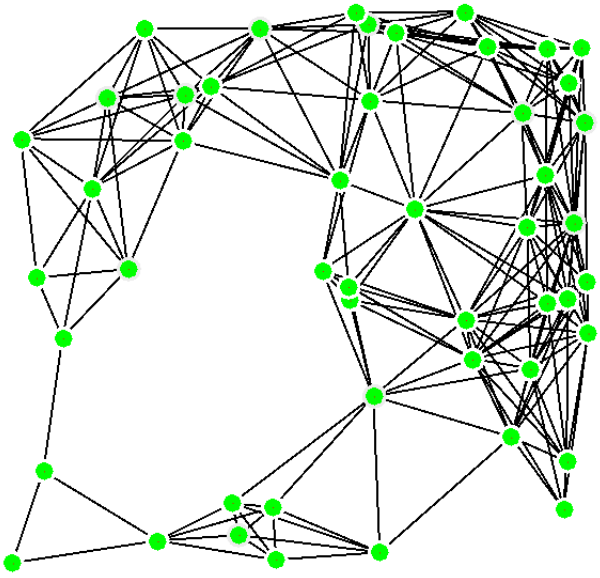


Figura 2.9: Ejemplo de un grafo aleatorio geométrico con 49 nodos y radio 0.25.

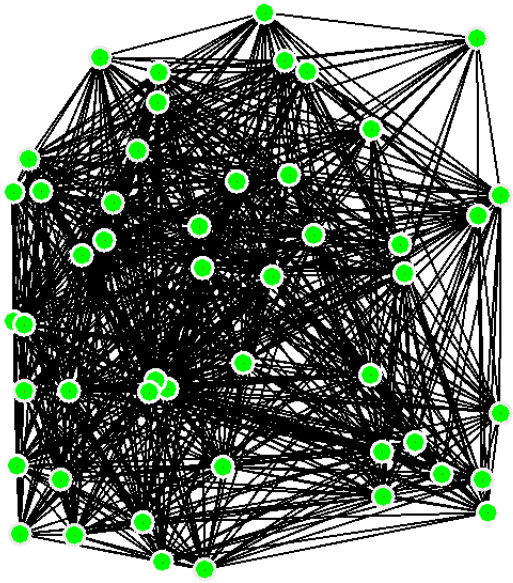


Figura 2.10: Ejemplo de un grafo aleatorio geométrico con 49 nodos y radio 0.5.

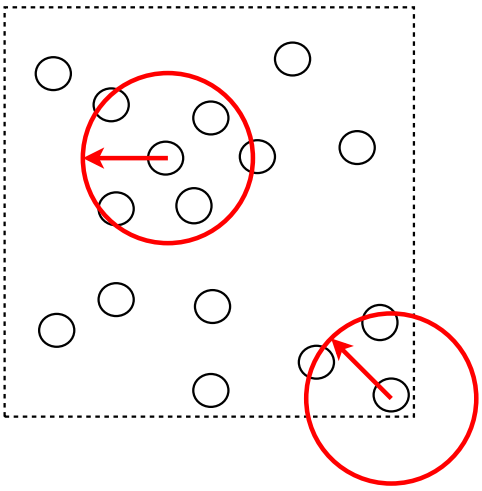


Figura 2.11: Efecto de bordes en un RGG al estimar el grado promedio.

2.6 Trabajo existente

Montesor, Meling y Babaolu [5, 6] demuestran que se puede realizar un balanceo de carga utilizando sistemas de agentes[3] que siguen tres reglas:

1. **Búsqueda máxima:** Un agente vaga a través de la red, buscando nodos con sobrecarga.
2. **Búsqueda mínima:** Un agente vaga a través de la red, buscando nodos sin carga.
3. **Transferencia:** Un agente transfiere tareas de el nodo con mas carga al nodo con menos carga.

Wang, Liu y Jin [39] utilizan el modelo de Montesor, Meling y Babaolu y agregan un parámetro de retardo a los agentes, demostrando que también se logra alcanzar un estado estable, aunque aparecen más oscilaciones y el tiempo de convergencia es más largo.

Si la carga se puede balancear bajo condiciones generales, se debe determinar que factor es el más importante para el balanceo de carga [38].

Stender, Kaiser y Albayrak [35] proponen un modelo de balanceo de carga basado en agentes, e introducen el concepto de *coordinador de balanceo de carga* LBC (por sus siglas en inglés), el cual realiza el balanceo de los agentes en tiempo de ejecución; sin embargo la comunicación entre los LBCs genera gran tráfico en la red.

Godfrey et al. [16] proponen un algoritmo de balanceo de carga para sistemas P2P estructurados. Este algoritmo está basado en *tablas de dispersión*. Godfrey et al.

[3]Los agentes son entidades autónomas diseñados para desempeñar un papel especifico, tienen control sobre su estado interno y sobre su propio comportamiento, están situados en un ambiente en particular, reciben entradas relacionadas con el ambiente a través de sensores y reaccionan con el ambiente a través de efectores.

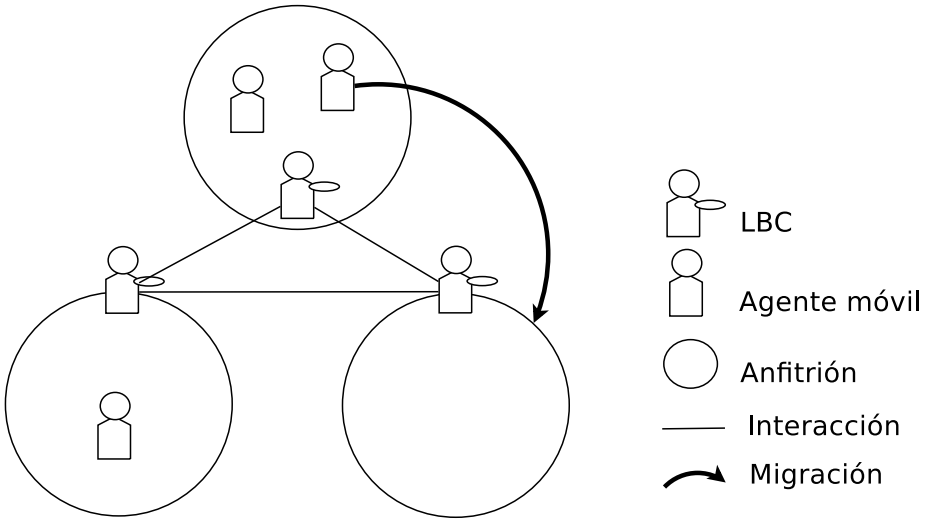


Figura 2.12: Coordinador de balanceo de carga [35].

demuestran que su algoritmo logra un balanceo de carga para el 90 % de utilización del sistema moviendo únicamente el 8 % de la carga que llega.

2.7 Simuladores

La simulación es la manera más popular de realizar investigación sobre redes. El costo de implementación es significativamente menor que el de experimentos a gran escala y se utlizan menos recursos computacionales. Si se construye cuidadosamente un modelo de simulación puede ser más realista que un modelo matemático. Naicken et al. [25] hacen una revisión de diferentes simuladores de redes P2P. Evaluan varios simuladores diferentes bajo diferentes criterios: Arquitectura de la simulación, Usabilidad, Escalabilidad, Estadísticas y Fundamentos de simulación de redes.

Simulador	Arquitectura	Usabilidad	Escalabilidad	Estadísticas	Fundamentos de redes
P2PSim	Eventos discretos para redes P2P estructuradas.	Documentación pobre.	3000 nodos.	Limitado	end-to-end, time graph, G2 graph, GT-ITM, aleatorio y euclidiano.
PeerSim	Eventos discretos para redes sin estructura. Simula nodos uniendose, pertiendo y fallando.	Solo la simulación Query-cycle esta documentada.	10^6 nodos.	Se pueden implementar componentes para reunir estadísticas.	No modelado.
Narses	Eventos discretos.	Sin documentación, código difícil de entender.	600 nodos, dependiendo de la topología.	Requiere modificación del código fuente.	Diferentes topologías, ejecución de balanco, velocidad y precisión.
Overlay Weaver	Emulación distribuida y un número de algoritmos estructurados.	API y codigo fuente bien documentado.	4000 nodos.	No es posible.	No modelado.
PlanetSim	Simulación de eventos discretos.	Diseño y API bien documentada; tutorial detallado	100000 nodos.	No, pero visualización es disponible.	Simulación limitada a las reeds subyacentes.
Neurogrid	Eventos doscretos para redes sin estructura, se puede modificar pra su uso con redes estructuradas.	Extensa documentación en la web.	300000 nodos.	Para variables predeterminadas, se necesita modificación de codifgo para otras	No modelado.

Tabla 2.2: Comparación simuladores de Naicken et al. [25].

Naicken et al. llegan a la conclusión de que el principal problema de la mayoría de los simuladores es la falta de documentación y la falta de mecanismos que permitan reunir diferentes estadísticas en una ejecución de la simulación. La mayoría de los investigadores utilizan herramientas como *ns-2* [2] debido a que se encuentran familiarizados con el funcionamiento, aunque sea frecuentemente inapropiado para la simulación de redes P2P. La elección más popular es la creación de simuladores a la medida, lo que complica la tarea de validar y reproducir resultados.

Capítulo 3

Balanceo de carga

En este capítulo presentamos los modelos y los métodos que utilizamos para llevar a cabo los experimentos, así como la herramienta que permite realizar estas simulaciones.

3.1 Modelo utilizado

Para este trabajo utilizamos un modelo de red síncrono [23] el cual se representa mediante un gráfo. Sea un grafo G se asocian los *procesadores* con los nodos de G y se permite que se comuniquen mediante los *canales* asociados con las aristas de G. Para cada nodo i se asocia un procesador P_i cada procesador P_i tiene una capacidad de procesamiento C_i y una cola de espera. El canal de comunicación entre los procesadores se asocia con cada arista (i, j) de G. Los canales pueden tener fallas, que son pérdida de mensajes y duplicación. Para este trabajo consideraremos que no existen fallas en los canales de comunicación. En la figura 3.1 se ejemplifica un sistema formado de tres procesadores y los canales de comunicación entre los procesadores.

El método utilizado para llevar a cabo el balanceo de carga consta de los siguientes pasos:

1. Se distribuyen generadores entre los procesadores, cada generador genera una tarea en cada ronda.

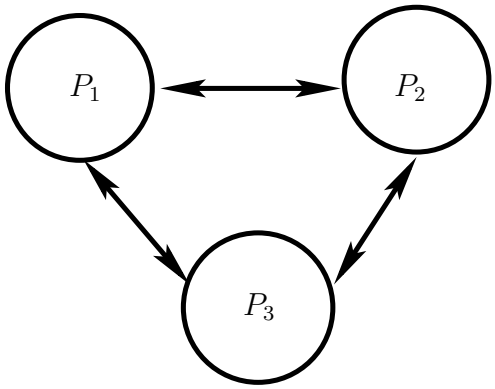


Figura 3.1: Modelo de red con tres procesadores y los canales de comunicación entre ellos.

2. Se balancea la carga entre los procesadores.

3. Los procesadores que no están vacíos eliminan tareas de acuerdo a su capacidad de procesamiento C_i.

3.2 Simulador

Con el fin de realizar pruebas del comportamiento de los diferentes métodos de balanceo en los distintos tipos de redes, se programó un simulador utilizando Java. En este simulador se crea un **modelo de red**, sobre el cual se aplicará un método de generación de carga y un método de balanceo, para analizar el comportamiento de la carga en el sistema.

El modelo de la red consta de nodos y aristas, donde cada uno de los nodos representa una computadora en la red. Cada nodo tiene asociado un procesador de cierta capacidad que es el encargado de procesar los trabajos. además cada nodo tiene una cola de espera donde van llegando cada uno de los trabajos que van llegando al nodo. Cada nodo tiene conocimiento de sus vecinos que tiene en la red.

Este programa consta de dos partes principales: El **Menú** y la **Ventana de visualización**. En la *ventana de visualización* se muestra el modelo de red generado,

se dibujan los nodos y las aristas de la red. El tamaño de los nodos se dibuja de acuerdo a la capacidad de procesamiento que tiene el nodo, entre más grande mayor es la capacidad del procesador. El color de los nodos varia entre verde y rojo; un nodo de color verde es un nodo que no está ocupado, mientras que un nodo rojo está ocupando toda la capacidad del procesador. Cada nodo tiene además un borde que varia de color entre blanco y negro, este borde representa la cola de espera del procesador. Un borde color blanco indica que no hay trabajos esperando en la cola, mientras más trabajos se van acumulando en la cola del procesador más oscuro se vuelve el color del borde del nodo.

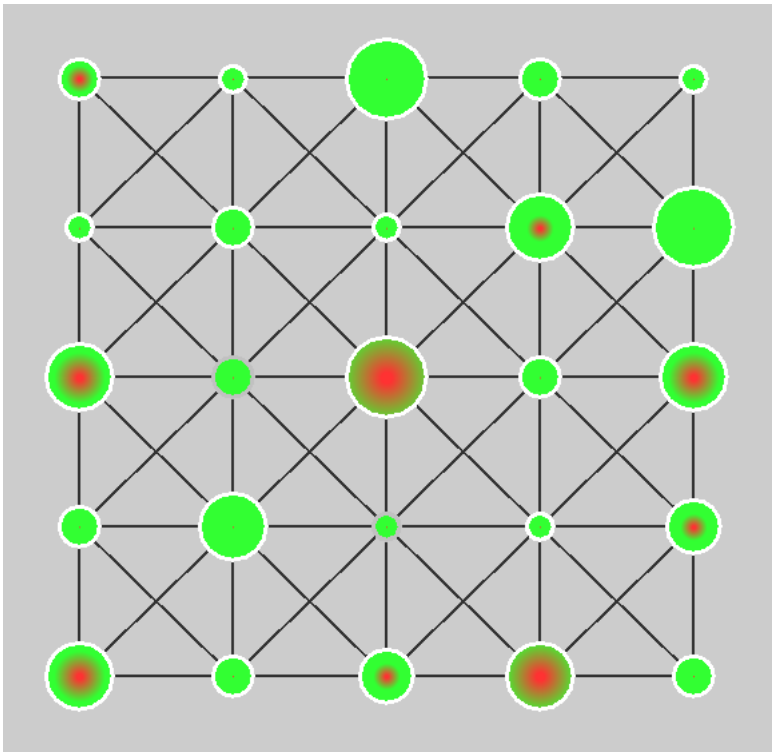


Figura 3.2: Ventana de visualización del simulador.

El *Menú del simulador* está dividido en tres partes principales: *Modelo, Carga y Método de balanceo.*

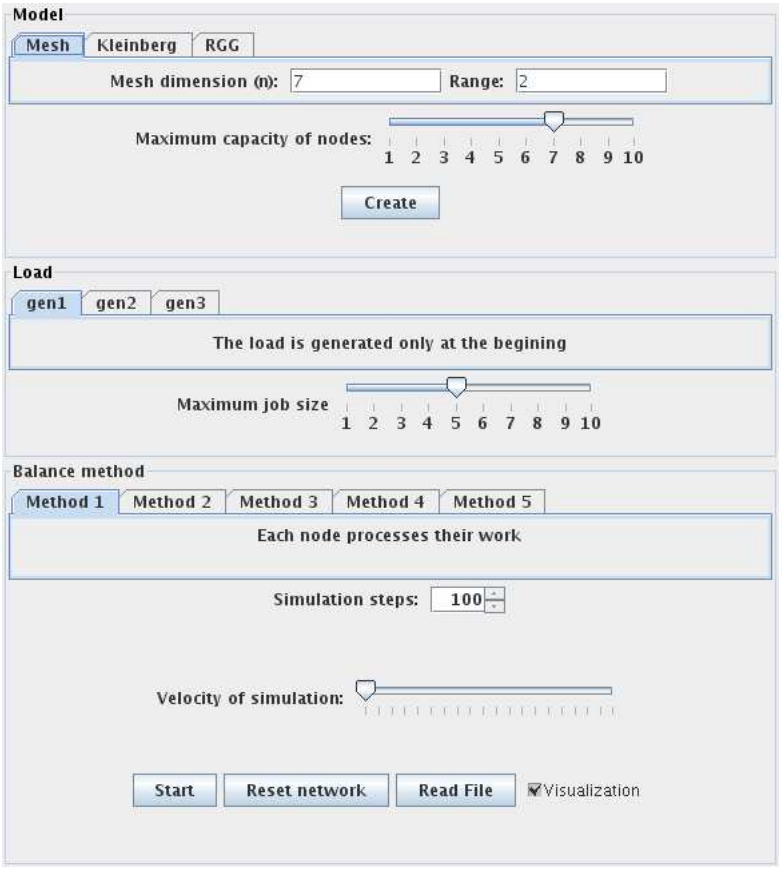


Figura 3.3: Menú del simulador.

3.2.1 Generación de modelo

En la sección referente al **Modelo**, se selecciona el tipo de modelo red que será utilizado en la simulación; se tienen tres opciones para generar el modelo de red: *Malla, Kleinberg* y *Grafo Aleatorio Geométrico* (RGG por sus siglas en inglés).

- **Malla:** Para generar una malla son necesarios dos parámetros: la *dimensión de la malla* (n) y el *rango*; n es el tamaño, por ejemplo un valor $n = 5$ crea un modelo de 25 nodos en una reja de tamaño 5×5; el *rango* es la distancia a la cual se conectaran cada uno de los nodos, un rango igual a n significa que los nodos se conectarán a todos los demás nodos que se encuentren a una distancia uno dentro de la malla.

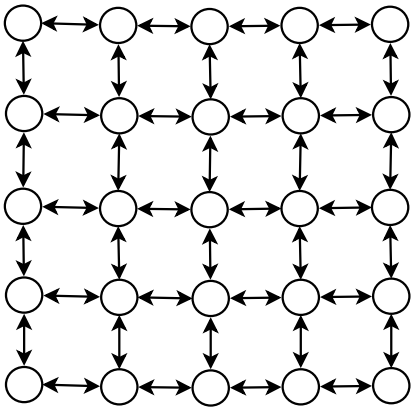


Figura 3.4: Ejemplo de una Malla con valores $n = 5$ y $p = 1$.

- **Kleinberg:** Utiliza los mismos parámetros *dimensión de malla* y *rango* para crear una Malla y además de estos parámetros se utiliza un parámetro q, que es la cantidad de aristas de largo alcance que se añadirán a la red.

- **RGG:** Para generar un modelo *RGG* son necesarios el *número de nodos* y el **radio** dentro del cual se conectarán los nodos; los nodos se crean dentro de un área de una unidad, el radio de conexión de los nodos debe de ser una fracción, un radio igual a $\sqrt{2}$ significaría que todos los nodos dentro del grafo se conectarán entre si.

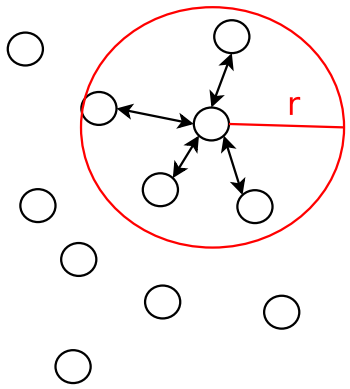


Figura 3.5: Vecinos de un nodo en un RGG.

Si se quisiera realizar una simulación utilizando un modelo de red diferente, se puede crear un archivo con los datos de la red el cual puede ser utilizado por el programa. El simulador utiliza un archivo de texto con el nombre “network.txt” para

guardar las redes utilizadas en la simulación; para utilizar una red diferente basta con reemplazar este archivo. Este archivo contiene una lista de nodos, cada nodo tiene un número que lo identifica, las coordenadas donde se encuentra localizado, la capacidad y al final una lista de sus vecinos.

```
Node 1 X: 1.0 Y: 1.0 c: 5 u: 3 neib: [ 4 3 2 ]
Node 2 X: 2.0 Y: 1.0 c: 4 u: 0 neib: [ 4 3 1 ]
Node 3 X: 1.0 Y: 2.0 c: 1 u: 0 neib: [ 4 2 1 ]
Node 4 X: 2.0 Y: 2.0 c: 8 u: 2 neib: [ 3 2 1 ]
```

Figura 3.6: Formato del archivo network.txt.

Despúes de seleccionar el modelo, se puede modificar la **capacidad máxima de los nodos** creados, la capacidad máxima de cada uno de los nodos se genera entre uno y el valor máximo seleccionado.

El botón **Crear**, genera el modelo con los parámetros seleccionados y lo muestra en la *ventana de visualización*; además se guarda el modelo en un archivo de texto.

3.2.2 Generación de carga

En la sección de **Carga** se selecciona el método de generación de los trabajos en el modelo. Hay tres métodos diferentes de generar la carga durante la simulación:

- **Inicio (I):** Solo se genera carga una vez al inicio de la simulación.
- **Aleatorio (A):** En cada paso de la simulación se genera carga de manera aleatoria para cada nodo con valores aleatorios entre cero y el valor máximo seleccionado. Los trabajos se generan con tamaños entre cero y el **Tamaño máximo del trabajo** en cada uno de los nodos. El número esperado de tra-

bajos generados utilizando este método es:

$$E(X) = \sum_{i=1}^{n} \frac{x_i}{n}. \tag{3.1}$$

Donde X es la variable aleatoria y x_i los valores generados.

- **Poisson (P):** Los trabajos son generados con una distribución de **Poisson**, como parámetro hay que seleccionar el valor de lambda (λ). El valor esperado para este tipo de distribución es igual a λ. Li et al. [21] y Mitzenmacher [24] utilizan una distribución de este tipo para la llegada de los trabajos. El valor esperado es:

$$E(X) = \lambda. \tag{3.2}$$

  Para generar los números aleatorios se utilizó la librería gratuita Colt [1] para Java (ver apéndice B).

Además es necesario seleccionar el tamaño máximo de las tareas generadas. Las tareas se generarán con tamaños entre cero y el valor seleccionado en **Tamaño máximo de los trabajos**; si la tarea creada tiene tamaño cero, entonces no se agrega al procesador, si el tamaño de la tarea es mayor a la capacidad del procesador se desecha la tarea.

3.2.3 Métodos de balanceo

En la parte de **Método de balanceo** se puede seleccionar el método de balanceo que será utilizado en el modelo, hay varias opciones:

- **Método Sin Balanceo (SB):** No se realiza balanceo de carga, cada uno de los nodos procesa sus trabajos.
- **Método Enviar Uno (EU):** En cada paso de la simulación el nodo procesa los trabajos de acuerdo a su capacidad, además si aún tiene trabajos en la cola pasa un trabajo al primer vecino desocupado de su lista de vecinos.

- **Método Enviar Varios (EV):** Similar al método anterior, con la diferencia que en vez de pasar un solo trabajo, pasa todos los trabajos que el vecino pueda recibir, de acuerdo ala capacidad de éste.

- **Método Robar Uno (RU):** Si el procesador está desocupado, roba un trabajo al vecino más cercano. En muchos de los trabajos de balanceo de carga se utiliza el método de robar trabajos (work-stealing en inglés), porque se ha observado funciona mejor para diferentes aplicaciones [7, 24].

- **Método Round Robin (RR):** Es uno de los métodos más tradicionales que se utiliza en redes [21]. En una implementación simple en este método si hay trabajos en la cola se envía un trabajo a cada uno de los vecinos en la lista, de manera secuencial.

Después de fijar estos parámetros, se puede también fijar el número de pasos máximo de la simulación, además se puede seleccionar la velocidad de la simulación.

Al hacer click en el botón **inicio**, comenzarán a correr los pasos de la simulación, realizando en cada uno el método de generación y de balanceo seleccionado. La simulación se detendrá cuando se alcance el número de pasos seleccionado. Como resultado de la simulación se tiene el valor de la carga total del sistema en cada paso de la simulación, cuantos trabajos fueron movidos entre los nodos y el tiempo máximo de espera de los trabajos para ser procesados.

El resultado se muestra en un archivo con el siguiente formato. Aparecen dos columnas, *step* (pasos en inglés), es el número de paso de la simulación y *load* (carga) es la carga total del sistema en ese paso. Al final aparece información adicional como el número total de trabajos movidos o transferidos a otro procesador (*Jobs Passed*) y el tiempo máximo de espera de los trabajos (*Job Max Wait*). Adicionalmente a estos datos, aparece la información de la parámetros utilizados en esa simulación, el tipo de red, el tipo de generación de carga y el método de balanceo aplicado.

```
step    load
97.0 564.0
98.0 531.0
99.0 478.0
100.0 511.0
101.0 527.0
Jobs passed: 407
Job Max Wait: 9
Network: Kleinberg, n: 7, r: 2, q: 2 node max capacity: 7
Load type: 3 lambda: 0.1 job max size: 5
Balance method: 4
```

Figura 3.7: Archivo de salida.

Para poder identificar los archivos, se les da una nomenclatura de acuerdo al tipo de red utilizada, el tipo de generación de carga y el método de balanceo usado.

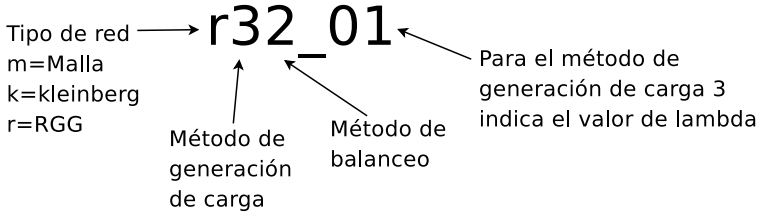


Figura 3.8: Nomenclatura de archivos.

Al finalizar la simulación se puede utilizar el botón de **Reiniciar red** para regresar el estado de la red al momento inicial de la simulación.

3.3 Diseño de experimentos

El propósito de la experimentación es conocer bajo qué condiciones se desempeña mejor cada uno de los diferentes métodos de balanceo mencionados anteriormente.

	Redes		
	Malla	Kleinberg	RGG
Tamaño (n)	7	7	49
	14	14	196
	38	38	1444

Tabla 3.1: Tamaños de las redes para los experimentos.

Se desea conocer bajo qué circunstancias la carga total del sistema se mantiene baja, el tiempo de espera de las tareas sea mínimo y la utilización de los procesadores esté balanceada.

3.3.1 Hipótesis

Con la experimentación se busca comprobar la siguiente hipótesis para cada tipo de generación de carga:

- La topología de la red y el método de generación de carga influyen de manera significativa en el desempeño del método de balanceo.

Se realizaron pruebas utilizando cuarto métodos de balanceo en tres tipos de redes y utilizando dos métodos diferentes de generación de carga. Se probaron en tres tamaños diferentes de las redes. Redes pequeñas 50 nodos, redes medianas 200 nodos y redes grandes 1500 nodos [26]. Estos tamaños fueron ajustados a 49, 196 y 1444, porque son los valores que se pueden obtener con rejas cuadradas.

3.3.2 Parámetros

Parámetros de las redes:

Los parámetros que serán fijados durante la experimentación son los siguientes para cada uno de los tipos de redes:

	Redes		
	Malla	Kleinberg	RGG
tamaño pequeño(n)	226	324	230
tamaño mediano	1038	1430	1024
tamaño grande	8286	11174	8182

Tabla 3.2: Número de aristas esperados para cada uno de los tipos de redes.

Malla: Rango = 2 **Kleinberg:** Rango = 2, q = 2 **RGG:** Tamaño pequeño: Radio = 0.25, tamaño mediano: Radio=0.13, tamaño grande: Radio = 0.05

El valor seleccionado para la capacidad máxima **(CM)** de los nodos generados será igual a 5.

Para que las redes sean comparables es necesario que sean similares en cuanto al número de nodos y el número de conexiones entre ellos. El número esperado de aristas para las redes tipo Malla es fijo. Para las tipo Kleinberg el número también es fijo, es el número de aristas de la malla más $q \times n$ aristas adicionales. Con estos parámetros el número esperado de aristas para cada una de las redes se muestra en la tabla 3.2.

Parámetros de generación de carga

Para el método **Poisson** se probaron diferentes valores de λ con valores enteros entre 1 y 10.

Para el tamaño de las tareas, se utilizó trabajos con tamaño igual a uno. Al realizar varias pruebas se observó que cuando el tamaño de los trabajos es variable se genera sobrecarga en el sistema, esto es porque cuando los trabajos generados son mayores que la capacidad de los nodos, estos trabajos se pasan de un nodo a otro hasta llegar a un nodo con una capacidad igual o mayor que la tarea. Para solucionar esto se podría modificar la manera en la que se procesan las tareas, por

ejemplo dividir las tareas en varias partes.

Parámetros de balanceo de carga

Número de pasos de la simulación = 100. Haciendo várias pruebas se determinó que 100 pasos son suficientes para la simulación, ya que los cambios en el comportamiento de la red se dan en los primeros pasos. Después de 100 el comportamiento de la carga de trabajo en la red es el mismo.

Capítulo 4

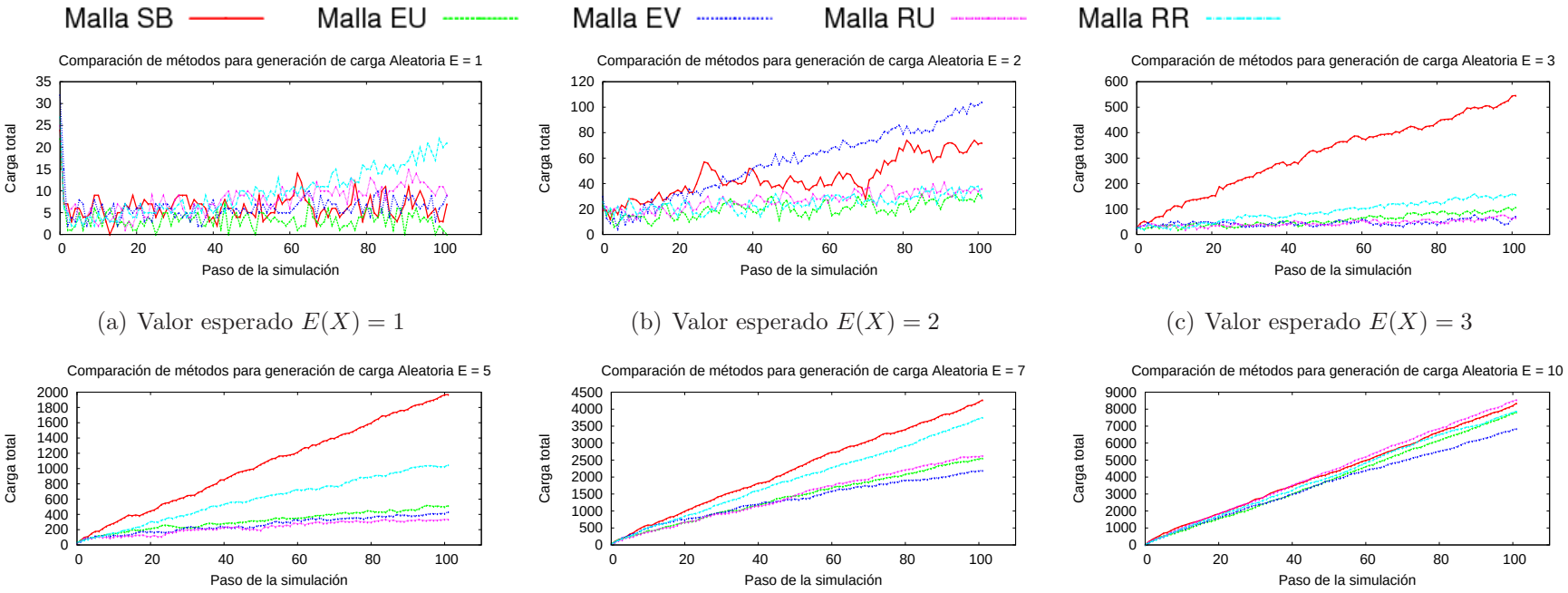
Resultados

A continuación se presentan algunos resultados de los experimentos donde se observa como utilizando diferentes métodos de balanceo para un mismo tipo de red, o diferentes métodos de generación de carga, la carga total de trabajo en el sistema puede aumentar o disminuir.

En las siguientes gráficas se muestra la carga total del sistema en cada uno de los pasos de la simulación. La carga total se obtiene sumando los trabajos de cada uno de los nodos. Conocer la carga total del sistema nos sirve para saber si la red es capáz de continuar procesando trabajos mientras estos sigan llegando, o si el tiempo de espera seguirá creciendo. Lo deseable es que la red pueda manejar las cargas de trabajo sin presentar sobrecarga. Las gráficas que se presentan corresponden a los experimentos de tamaño pequeño, para los tamaños grande y mediano de los tres tipos de redes se encontró que los resultados son similares.

4.1 Generación de carga tipo **Aleatoria**

Para este método de generación de carga se probaron diferentes valores esperados entre 0 y 15, se muestran solo las gráficas donde es más notorio el cambio en el comportamiento del sistema; se observó que para valores más pequeños los procesadores son capaces de manejar su propia carga, mientras que con valores más grandes, todos los nodos presentan sobrecarga.



(a) Valor esperado $E(X) = 1$

(b) Valor esperado $E(X) = 2$

(c) Valor esperado $E(X) = 3$

(d) Valor esperado $E(X) = 5$

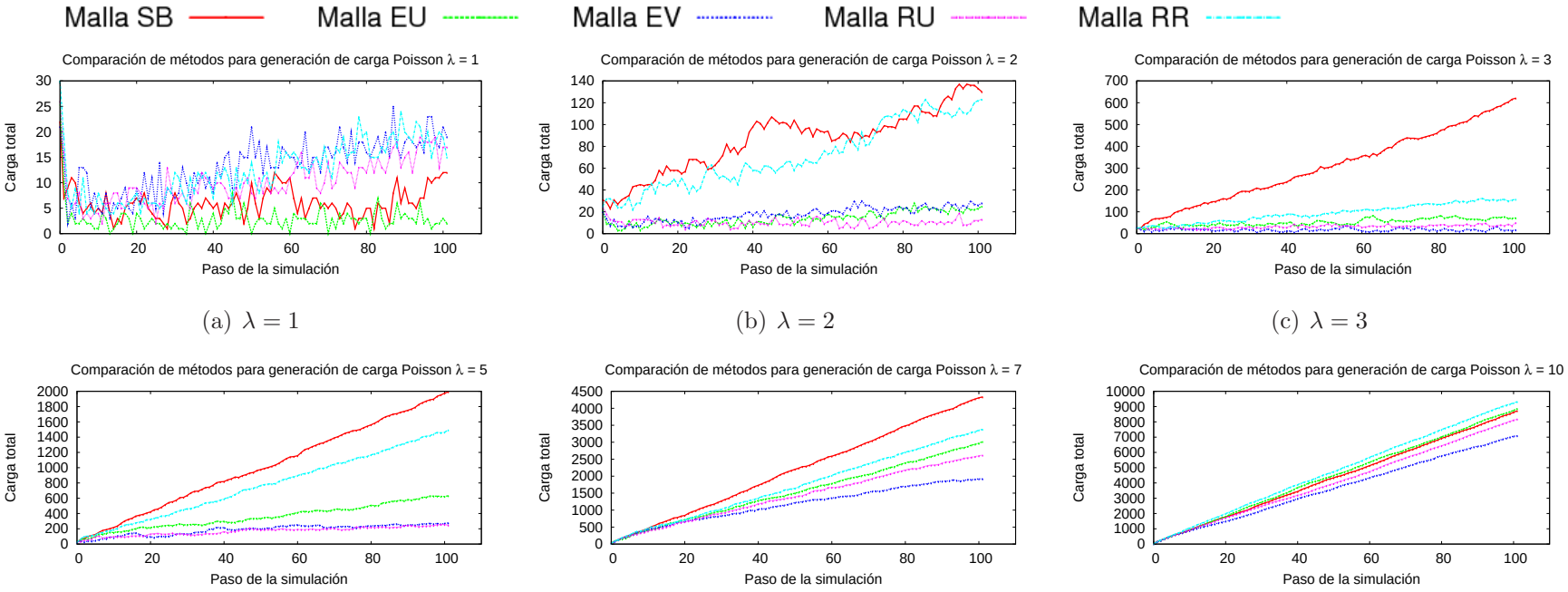
(e) Valor esperado $E(X) = 7$

(f) Valor esperado $E(X) = 10$

Figura 4.1: Comparación de métodos para la generación de carga **Aleatoria** con diferentes valores esperados para una Malla. Las gráficas se encuentran en diferente escala debido a la cantidad de trabajos generados.

4.2 Generación de carga tipo **Poisson** con diferentes valores de lambda

Para el método de generación de carga tipo **Poisson** se probaron diferentes valores de λ entre 0 y 15; se observó que para valores más pequeños los procesadores son capaces de manejar su propia carga, mientras que con valores más grandes, todos los nodos presentan sobrecarga.



(a) $\lambda = 1$ (b) $\lambda = 2$ (c) $\lambda = 3$

(d) $\lambda = 5$ (e) $\lambda = 7$ (f) $\lambda = 10$

Figura 4.2: Comparación de métodos para la generación de carga **Poisson** con diferentes valores de λ para una Malla.

Se puede observar que para los valor de λ igual a uno y valores más pequeños, no es necesario aplicar métodos de balanceo ya que los procesadores son capaces de manejar su propia carga. A medida que aumenta el valor de λ se observa que el nivel de carga cuando no se aplica balanceo aumenta y los métodos de balanceo mantienen la carga en un nivel más bajo. Los valores de λ entre 3 y 5 es donde los métodos de balanceo muestran un mejor desempeño. Conforme va aumentando el valor de λ los métodos en algunos casos mantienen un nivel de carga menor que sin aplicar balanceo, pero no evitan que la carga total se dispare. Con base en estas observaciones los siguientes resultados que se muestran corresponden a experimentos con un valor esperado para los trabajos $E(X) = 3$ que es el valor donde consideramos que los métodos tienen un mejor desempeño. Con este valor se espera que no se genere sobrecarga en el sistema ya que el valor esperado de trabajos es mayor que la capacidad esperada de los nodos $CM > E(X)$.

4.3 Comparación de métodos con generación de carga **Aleatorio**

En las gráficas se muestra el promedio de 30 repeticiones para el mismo experimento. Se realizaron 30 repeticiones para conocer la variabilidad del experimento y con esto comprobar si en realidad los métodos dan siempre los mismos resultados, o solo en un caso específico.

Se realizaron experimentos usando el método de generación de carga **Aleatorio** y probando los diferentes métodos de balanceo en cada tipo de red, para comprobar si el tipo de red y el método de generación influyen en el desempeño del método de balanceo.

En esta gráfica se muestra el comportamiento de los métodos de balanceo para una red tipo Malla con el método de generación de carga **Aleatorio**. Se observa que para los métodos **Enviar Uno Enviar Varios** y **Robar Uno** el nivel de carga

se mantiene similar, mientras que con el método de balanceo **Round Robin** la carga total del sistema va en aumento. En este método la mayoría de los trabajos solamente están pasando de un procesador a otro y no alcanzan a ser procesados, por eso se acumulan y ocasionan que la carga del sistema aumente. Con todos los métodos de balanceo se logra mantener la carga total del sistema en un nivel más bajo que cuando no se aplica el balanceo de carga.

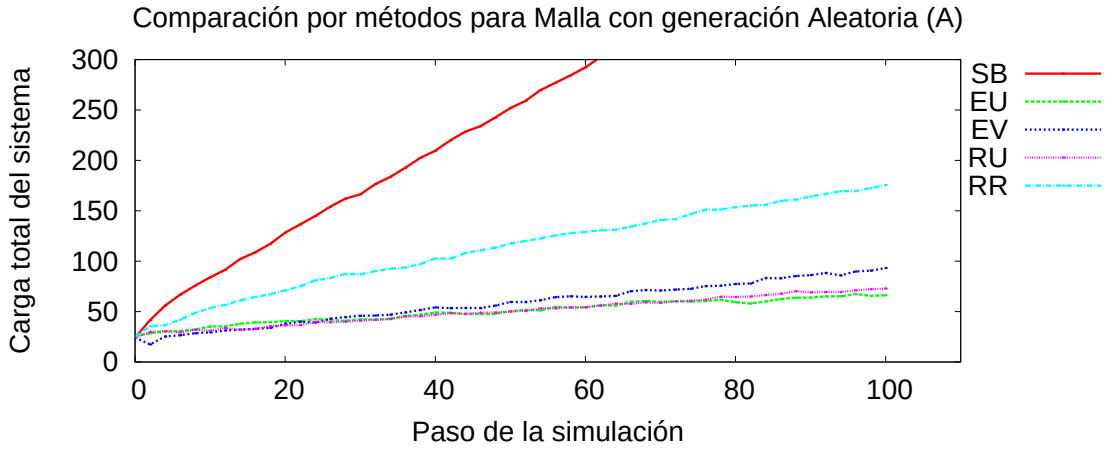


Figura 4.3: Comparación de métodos de balanceo para la generación de carga **Aleatorio** para una Malla.

El mismo comportamiento se observa con las redes tipo Kleinberg y RGG con el mismo método de generación de carga **Aleatorio**.

En todos los casos se observa el mismo comportamiento de los métodos de balanceo, siendo el método **Robar Uno** el que mantiene la carga del sistema en el nivel más bajo para las redes tipo Kleinberg y RGG.

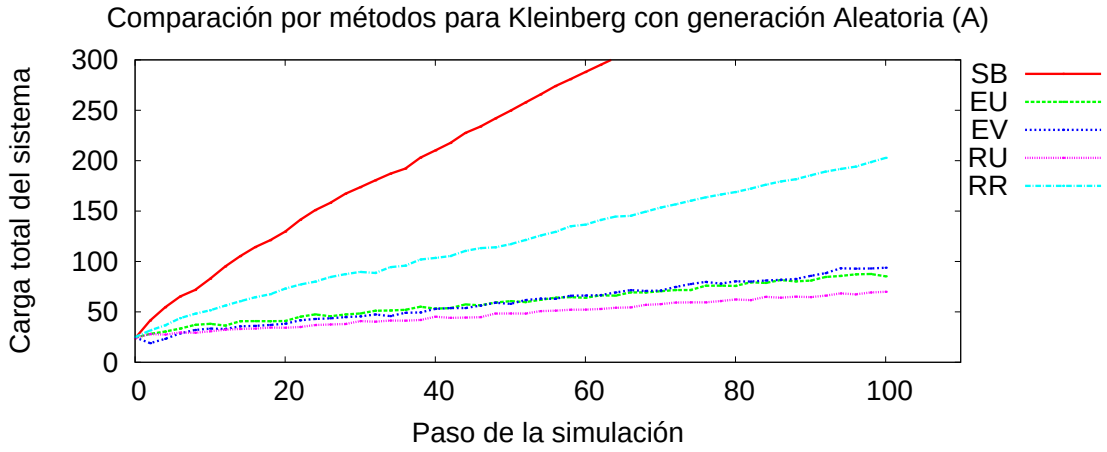


Figura 4.4: Comparación de métodos de balanceo para la generación de carga **Aleatorio** para red Kleinberg.

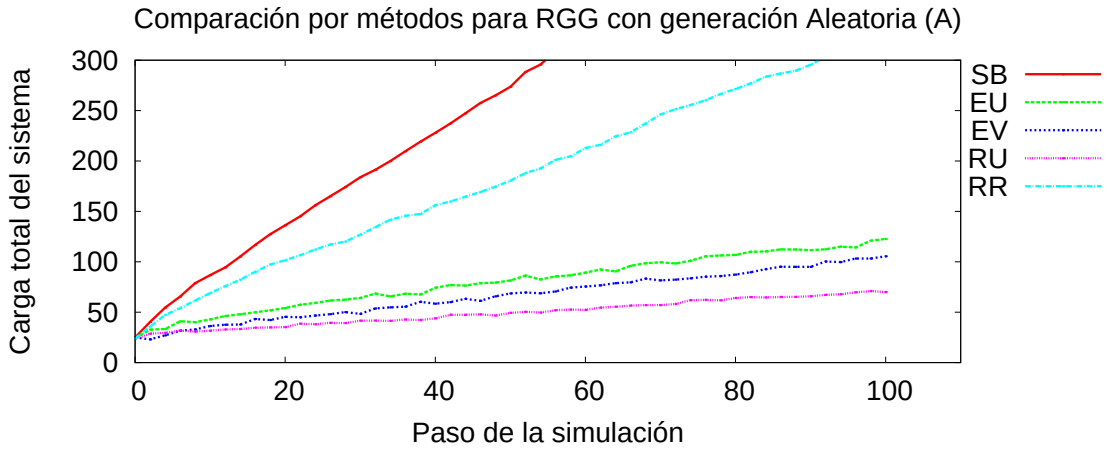


Figura 4.5: Comparación de métodos de balanceo para la generación de carga **Aleatorio** para una red RGG.

4.4 Comparación de métodos con generación de carga **Poisson**.

Para el método de generación de carga **Poisson**, en las tres tipos de redes se dispara el nivel de carga cuando no hay método de balanceo; se observa que se mantiene el nivel de carga total del sistema más bajo que sin aplicar balanceo con cualquiera de los métodos **Enviar Uno**, **Enviar Varios**, **Robar Uno**. Con el método de balanceo **Round Robin** de mantiene el nivel de carga más bajo que sin aplicar balanceo pero con valores por encima de los demás métodos. Al enviar un trabajo de un procesador a otro éste último todavía continúa generando sus propios trabajos y se pueden comenzar a acumular. El método **Round Robin** es el que involucra el mayor número de procesadores al momento de aplicarlo es por eso que el nivel de carga total del sistema al aplicar éste método es mayor que el de los otros.

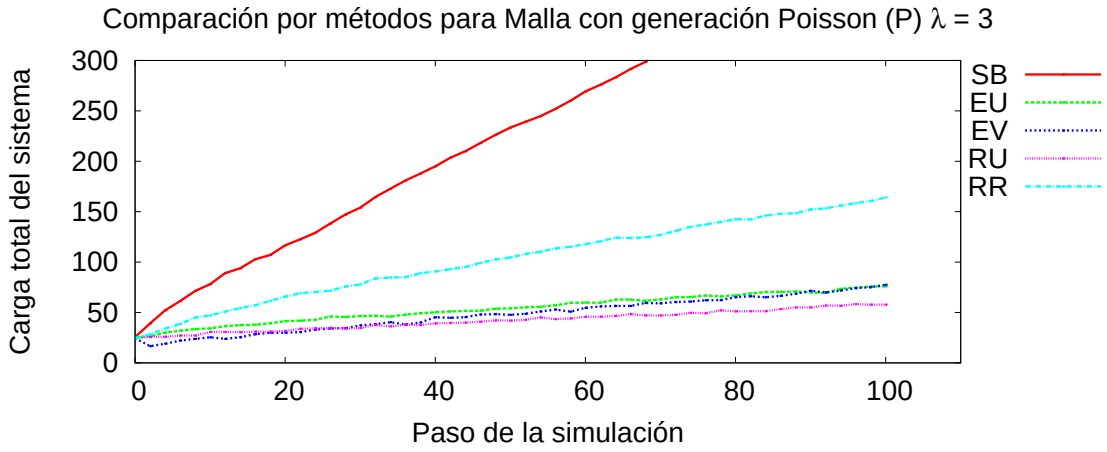


Figura 4.6: Comparación de métodos de balanceo para una Malla con generación de carga **Poisson**.

Para los tres tipos de redes, el método **Robar Uno** es el que mantiene la carga total del sistema en el nivel más bajo.

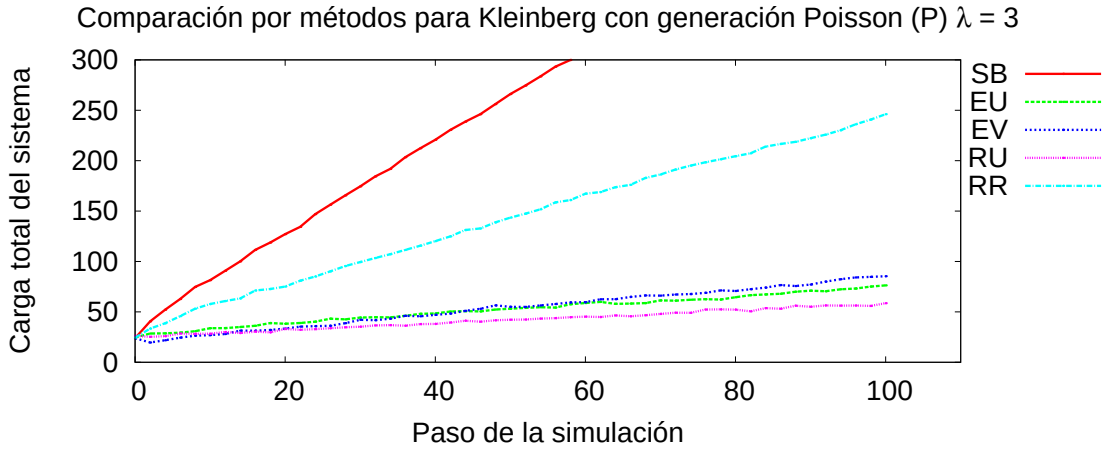


Figura 4.7: Comparación de métodos de balanceo para una red tipo Kleinberg con generación de carga **Poisson**.

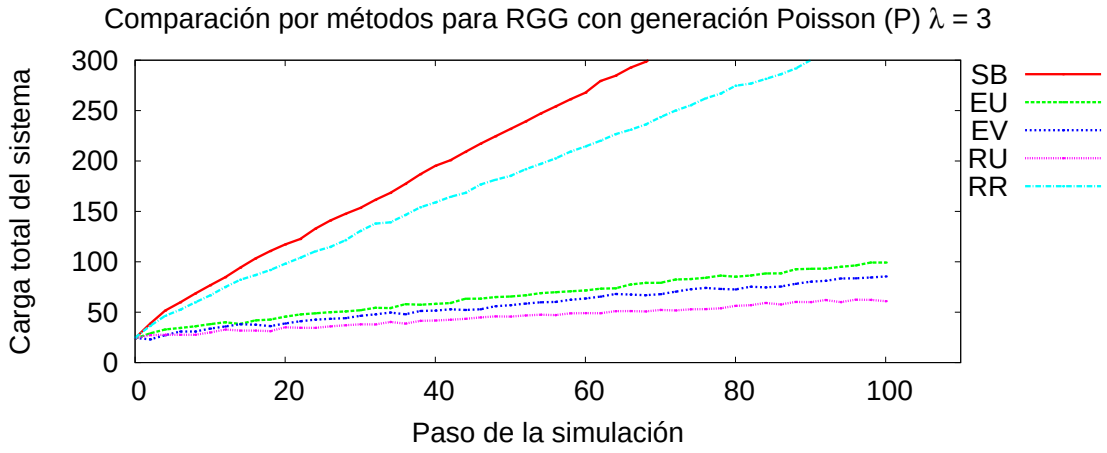


Figura 4.8: Comparación de métodos de balanceo para una red tipo RGG con generación de carga **Poisson**.

4.5 Comparación de métodos de balanceo por tipo de red

4.5.1 Método **Sin Balanceo**

Con el método **Sin Balanceo** (no hace ningún tipo de balanceo), se observa como los niveles de carga total del sistema se mantienen para los tres tipos de redes para ambos métodos de generación de carga se disparan.

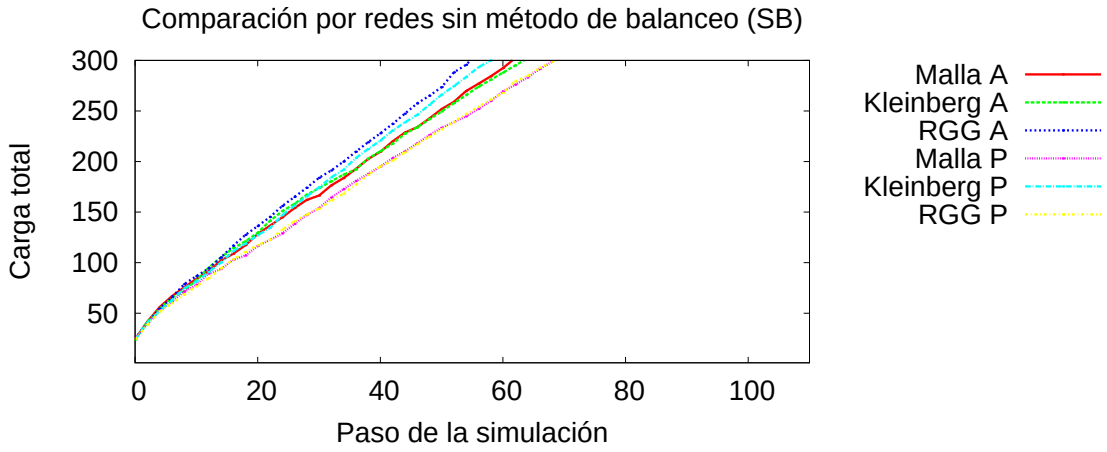


Figura 4.9: Comparación por redes del método de balanceo **Sin Balanceo**.

4.5.2 Método **Enviar Uno**

Aplicando el método **Enviar Uno** vemos que se mantiene el nivel de carga en un nivel más bajo para los tres tipos de redes que cuando no se aplica balanceo.

4.5.3 Método **Enviar Varios**

En este caso, la carga total del sistema crece de manera semejante para los tres tipos de redes.

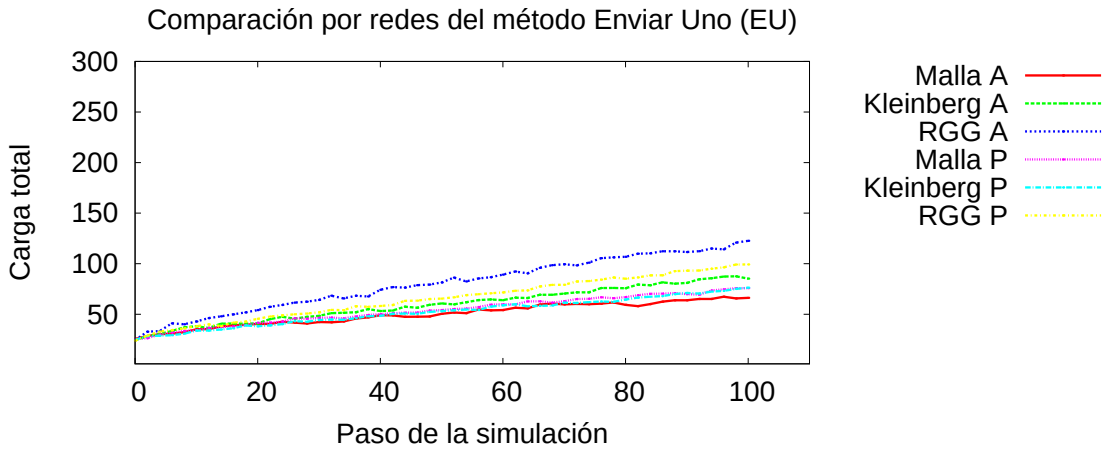


Figura 4.10: Comparación por redes del método de balanceo **Enviar Uno**.

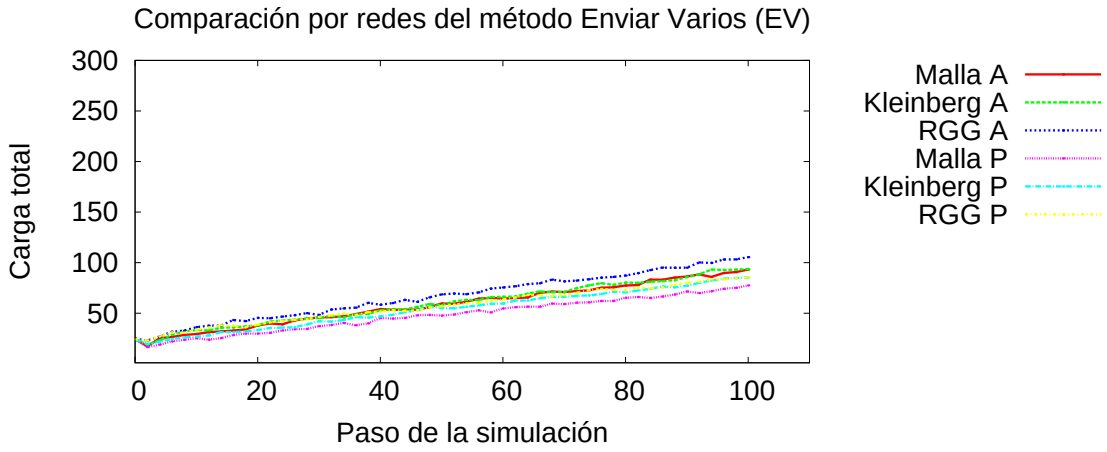


Figura 4.11: Comparación por redes del método de balanceo **Enviar Varios**.

4.5.4 Método **Robar Uno**

Para los dos tipos de generación de carga, se mantiene el nivel para los tres tipos de redes.

4.5.5 Método **Round Robin**

Para los dos tipos de generación de carga, se mantiene el nivel para los tres tipos de redes.

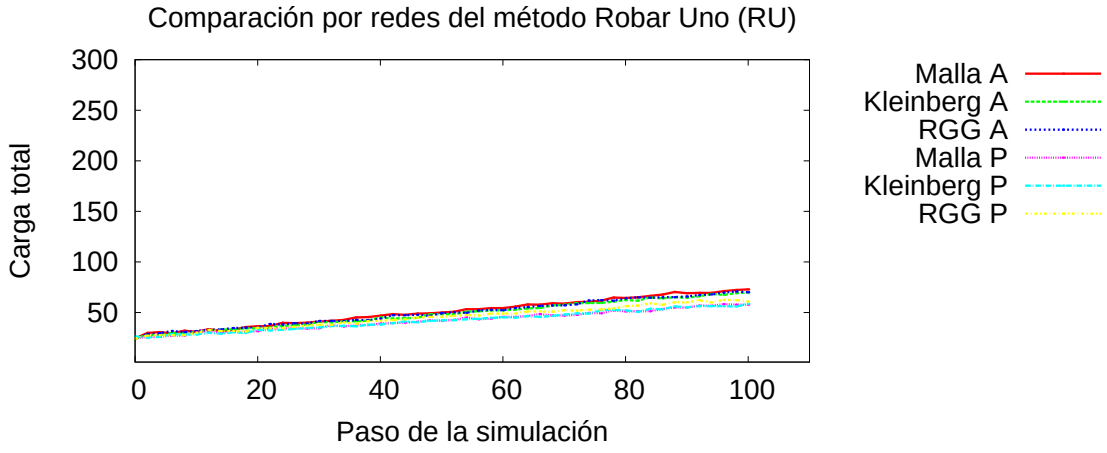


Figura 4.12: Comparación por redes del método de balanceo **Robar Uno**.

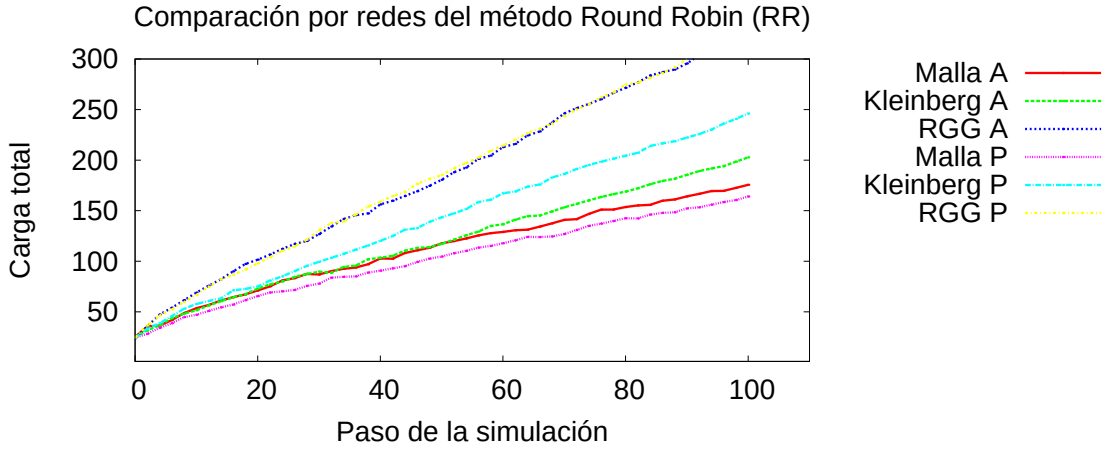


Figura 4.13: Comparación por redes del método de balanceo **Robar Uno**.

4.6 Comparación del método **Enviar Uno** para cada tipo de red

A continuación se muestran las gráficas para el método de balanceo **Enviar Uno** para cada tipo de red, utilizando los dos tipos diferentes de generación de carga.

4.6.1 Método **Enviar Uno** para Malla

Para una red tipo Malla con método de generación de carga **Aleatorio** y **Poisson** el método de balanceo **Enviar Uno** mantiene el nivel de la carga total en un nivel menor que cuando no se aplica un método de balanceo.

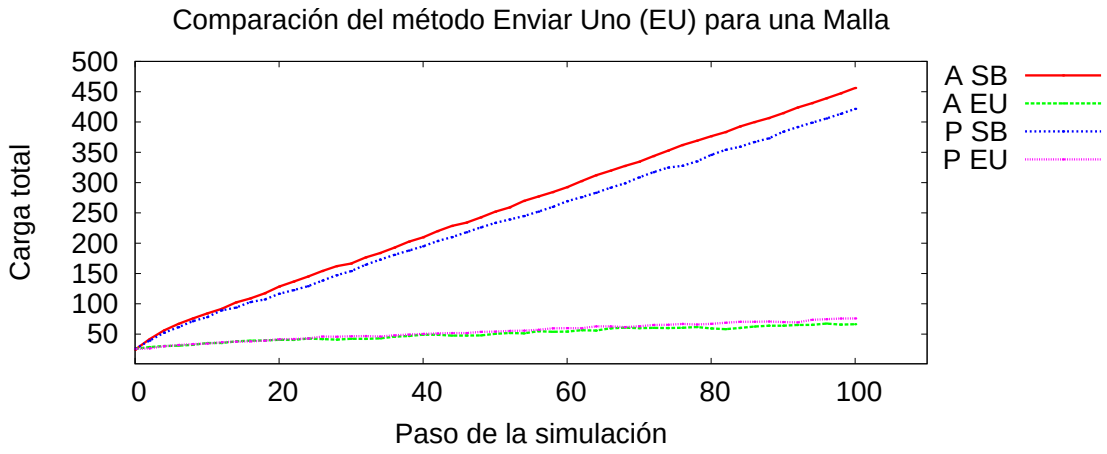


Figura 4.14: Método **Enviar Uno** para una red tipo Malla.

4.6.2 Método **Enviar Uno** para Kleinberg

Igual que en el caso anterior, para la generación **Aleatoria** y **Poisson** el nivel de carga es menor que cuando no se aplica método de balanceo.

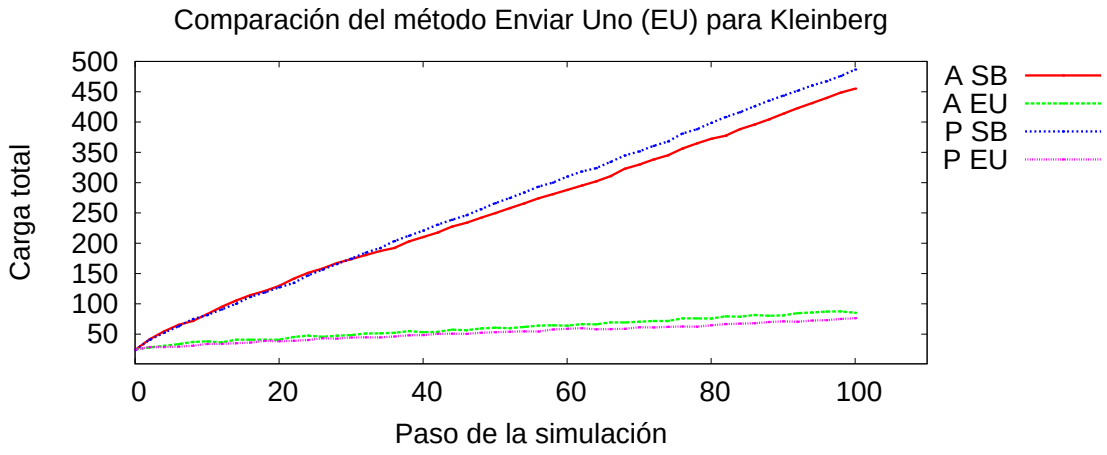


Figura 4.15: Método **Enviar Uno** para una red tipo Kleinberg.

4.6.3 Método **Enviar Uno** para RGG

Para la red tipo RGG, se observa un comportamiento similar al de los casos anteriores para la generación de carga tipo **Poisson** y para la generación de carga tipo **Aleatorio**.

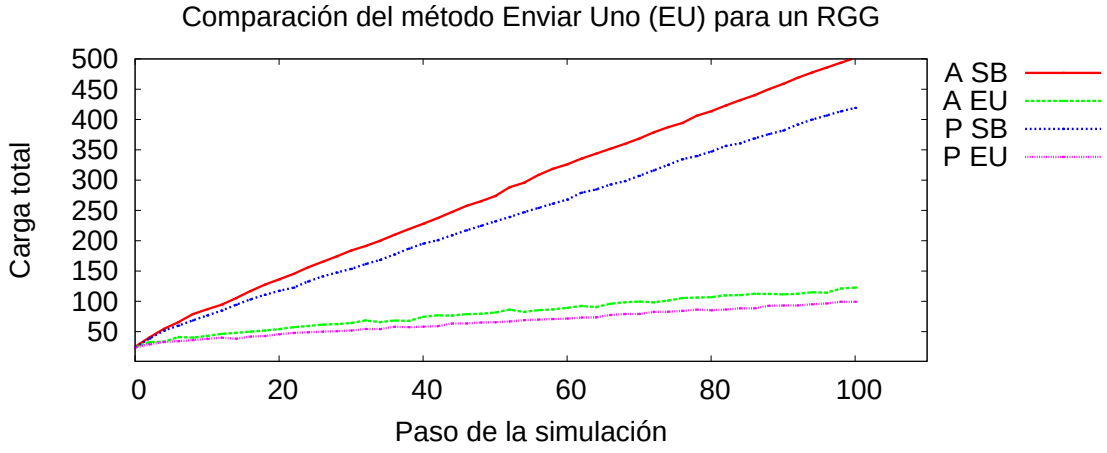


Figura 4.16: Comparación del método **Enviar Uno** para la red tipo RGG.

4.7 Comparación del método **Enviar Varios** para cada tipo de red

A continuación se muestran las gráficas para el método de balanceo **Enviar Varios** para cada tipo de red, utilizando los dos tipos diferentes de generación de carga.

4.7.1 Método **Enviar Varios** para Malla

Para una red tipo Malla con método de generación de carga **Aleatorio** y **Poisson** el método de balanceo **Enviar Varios** mantiene el nivel de la carga total en un nivel menor que cuando no se aplica un método de balanceo.

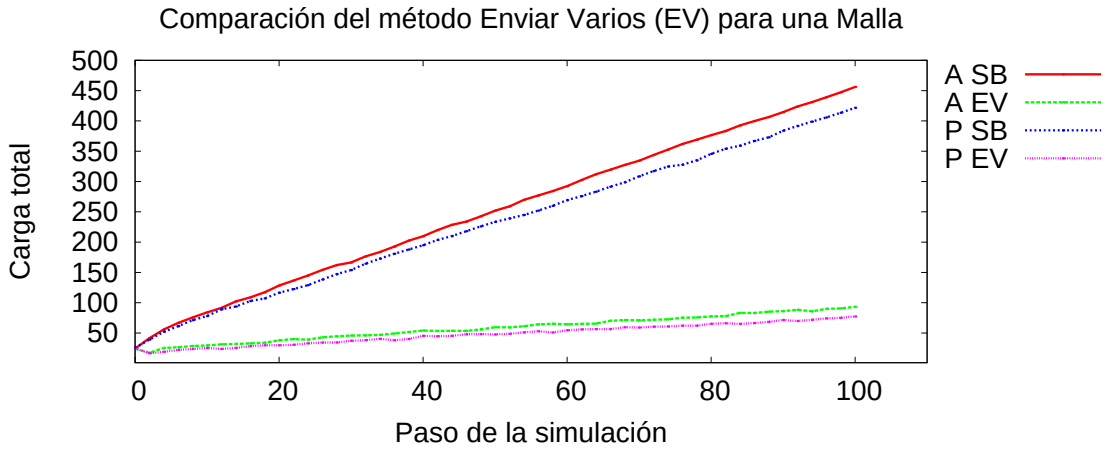


Figura 4.17: Método **Enviar Varios** para una red tipo Malla.

4.7.2 Método **Enviar Varios** para Kleinberg

Igual que en el caso anterior, la carga total del sistema se mantiene más baja para para el método de generación **Aleatorio** y para la generación tipo **Poisson**.

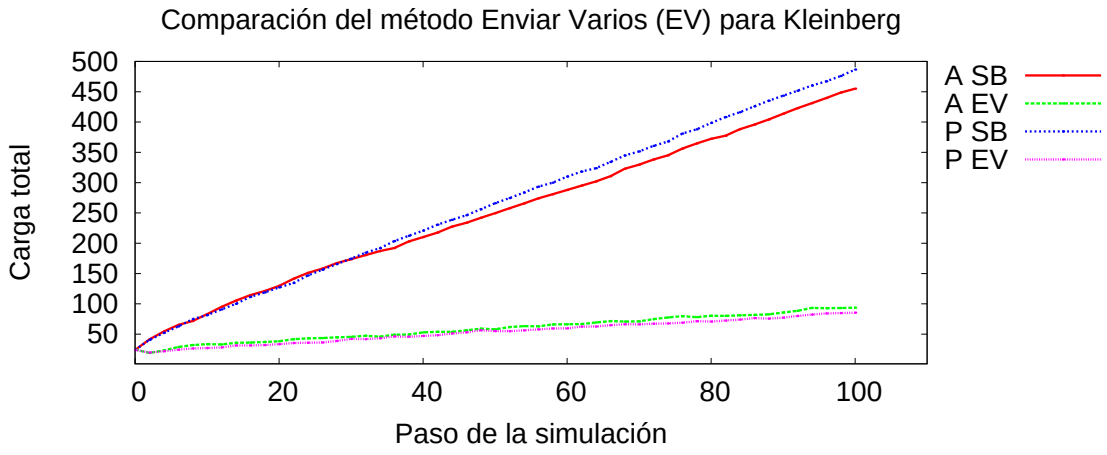


Figura 4.18: Método **Enviar Uno** para una red tipo Kleinberg.

4.7.3 Método **Enviar Varios** para RGG

Para la red tipo RGG, se observa un comportamiento similar al de los casos anteriores para la generación de carga tipo **Poisson** y para la generación de carga tipo **Aleatorio**.

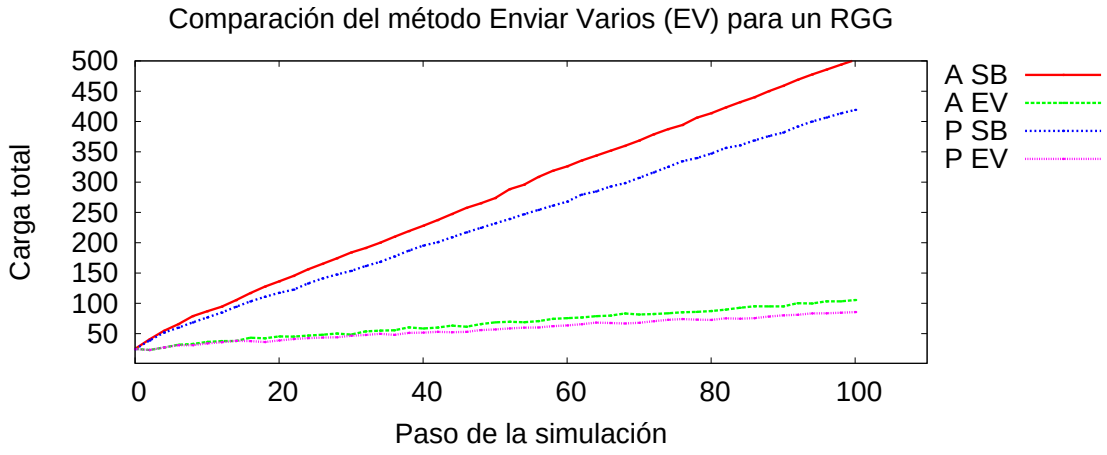


Figura 4.19: Método **Enviar Varios** para la red tipo RGG.

4.8 Comparación del método **Robar Uno** para cada tipo de red

A continuación se muestran las gráficas para el método de balanceo **Robar Uno** para cada tipo de red, utilizando los dos tipos diferentes de generación de carga.

4.8.1 Método **Robar Uno** para Malla

Al igual que con los métodos de balanceo anteriores, el método **Robar Uno** con el método de generación **Aleatorio** y **Poisson**, el nivel de la carga total es menor que cuando no se aplica un método de balanceo.

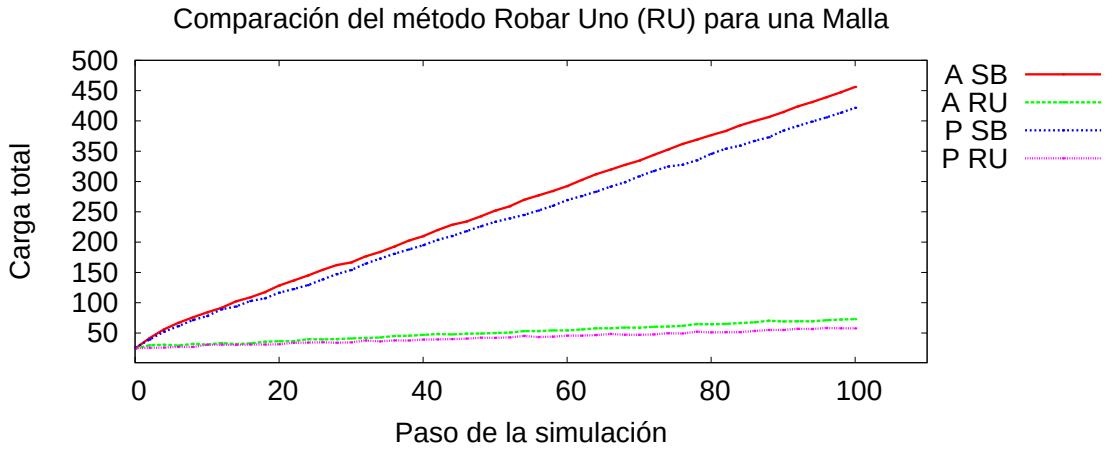


Figura 4.20: Método **Robar Uno** para una red tipo Malla.

4.8.2 Método **Robar Uno** para Kleinberg

Igual que en el caso anterior, la carga total del sistema es similar para para el método de generación **Aleatorio** y para la generación tipo **Poisson** el nivel de carga es menor que cuando no se aplica método de balanceo.

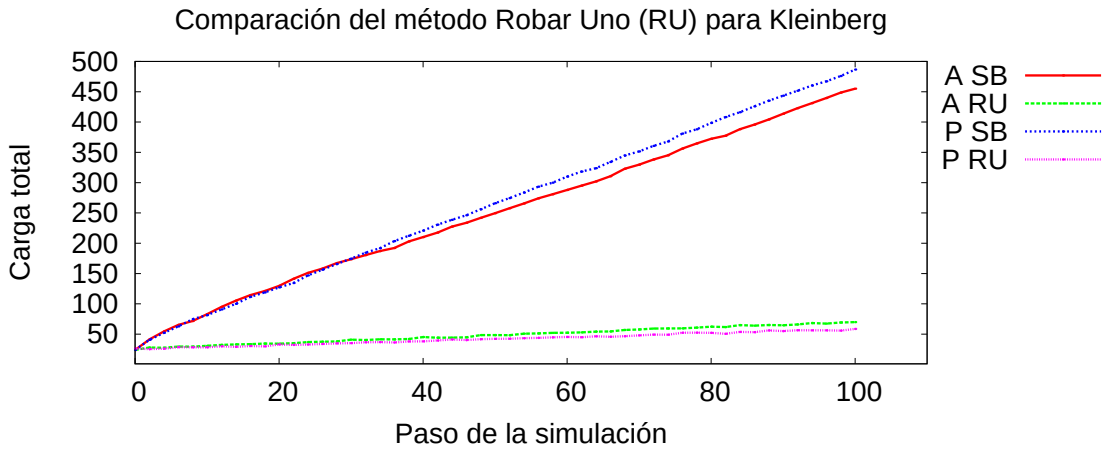


Figura 4.21: Método **Robar Uno** para una red tipo Kleinberg.

4.8.3 Método **Robar Uno** para RGG

Para la red tipo RGG, se observa un comportamiento similar al de los casos anteriores para la generación de carga tipo **Poisson** y para la generación de carga tipo **Aleatorio** se observa que la carga se mantiene en un nivel menor a cuando no se aplica balanceo.

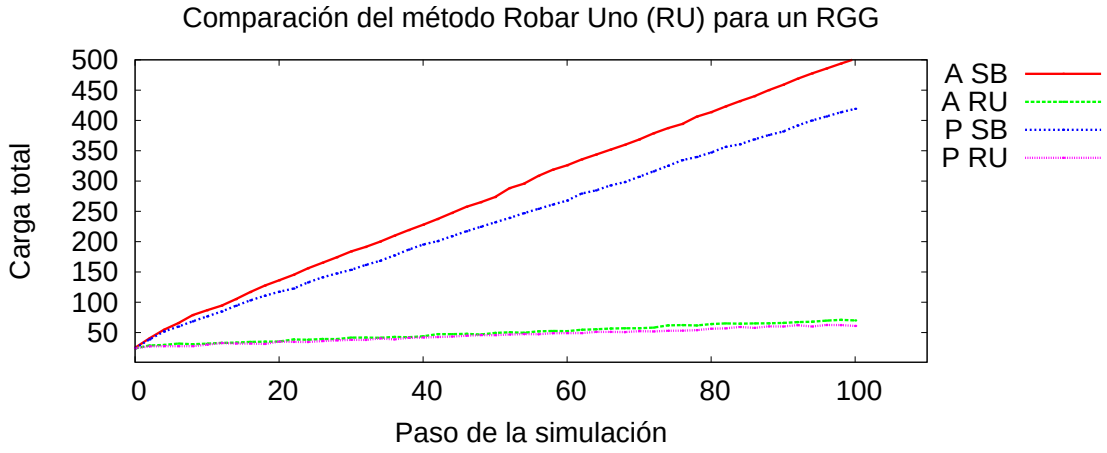


Figura 4.22: Comparación del método **Robar Uno** para la red tipo RGG.

Carga total del sistema durante la simulación						
	Malla		Kleinberg		RGG	
	A	P	A	P	A	P
SB	se dispara	se dispara	se dispara	se dispara	se dispara	se dispara
EU	bajo	bajo	bajo	bajo	bajo	bajo
EV	bajo	bajo	bajo	bajo	bajo	bajo
RU	bajo	bajo	bajo	bajo	bajo	bajo
RR	medio	medio	medio	alto	alto	alto

Tabla 4.1: Comparación de la carga total del sistema al aplicar los métodos de balanceo.

4.9 Comparación del método **Round Robin** para cada tipo de red

A continuación se muestran las gráficas para el método de balanceo **Round Robin** para cada tipo de red, utilizando los dos tipos diferentes de generación de carga.

4.9.1 Método **Round Robin** para Malla

Para una red tipo Malla, con el método de balanceo **Round robin** para el método de generación de carga **Aleatorio** y para el método **Poisson**, el nivel de la carga total es menor que cuando no se aplica un método de balanceo.

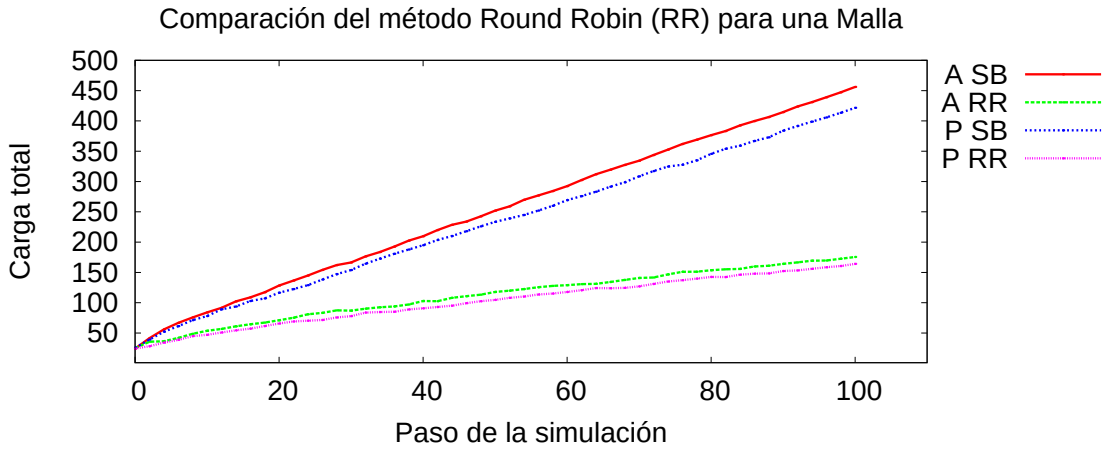


Figura 4.23: Método **Round Robin** para una red tipo Malla.

4.9.2 Método **Round Robin** para Kleinberg

Igual que en el caso anterior, para la generación tipo **Aleatoria** y **Poisson** el nivel de carga es menor que cuando no se aplica método de balanceo.

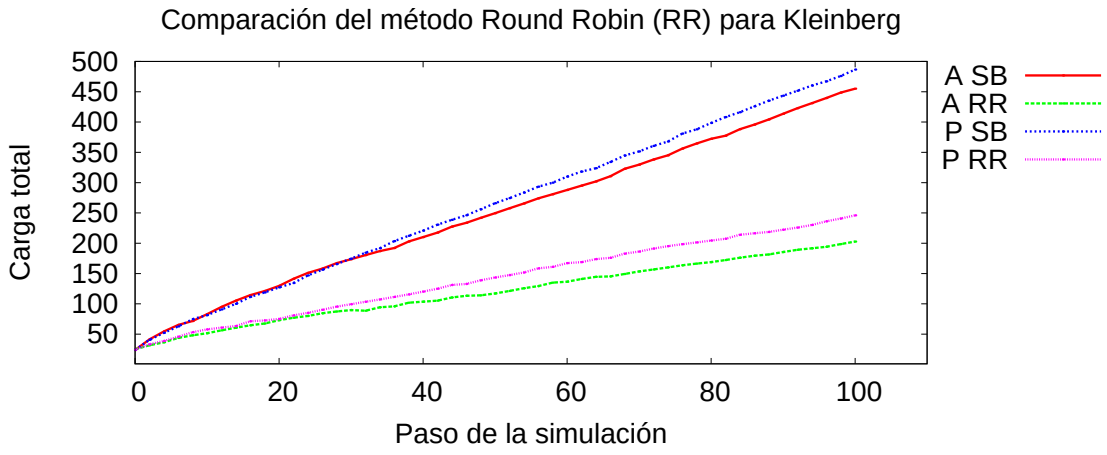


Figura 4.24: Método **Round Robin** para una red tipo Kleinberg.

4.9.3 Método **Round Robin** para RGG

Para la red tipo RGG, se observa un comportamiento similar al de los casos anteriores para la generación de carga tipo **Poisson** y **Aleatorio**.

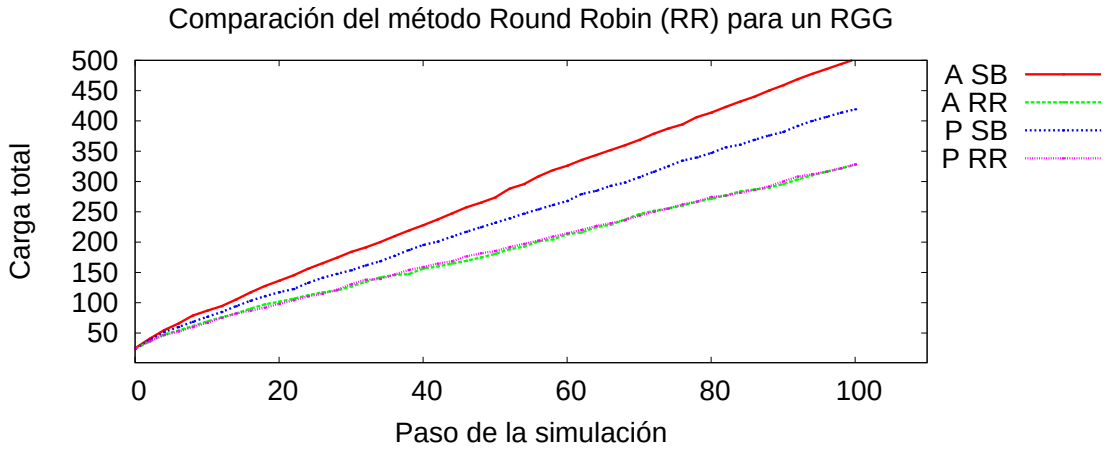


Figura 4.25: Comparación del método **Round Robin** para la red tipo RGG.

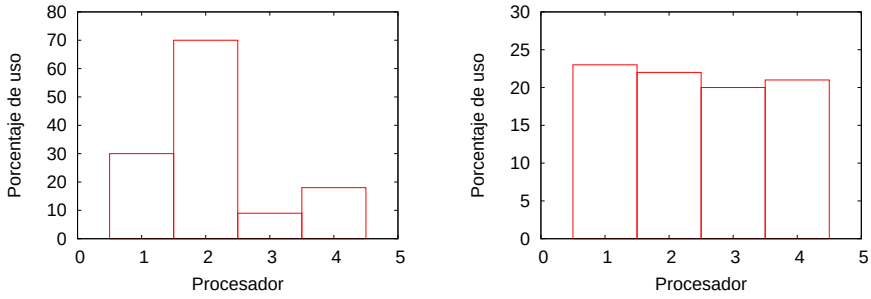
4.10 Medida de calidad de balanceo **CB**

Para comparar que tanto está en uso un procesador durante la simulación, se calcula un promedio del porcentaje de uso que el procesador tiene en cada paso de la simulación. Para medir la calidad de balanceo (CB) de la red, se propone la siguiente medida de balanceo:

$$CB = \frac{\sum_{i=1}^{n} \frac{\mu_i}{\text{máx}\{\mu_i\}}}{n} \tag{4.1}$$

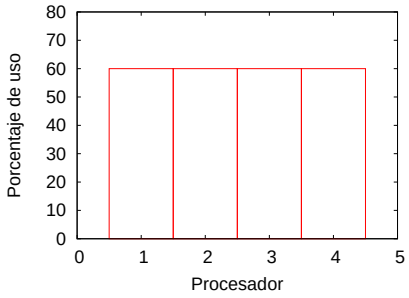
donde n es el número de procesadores y μ_i es el promedio de carga de procesador i.

CB es una medida de la diferencia en el uso de los procesadores durante una simulación. Cuando todos los procesadores tienen el mismo promedio de carga el valor de CB es igual a uno. El valor ideal de CB es igual a uno, entre menor sea el valor significa que existe una mayor diferencia entre el promedio de carga de los procesadores.



(a) Valor de CB=0.45

(b) Valor de CB=0.95

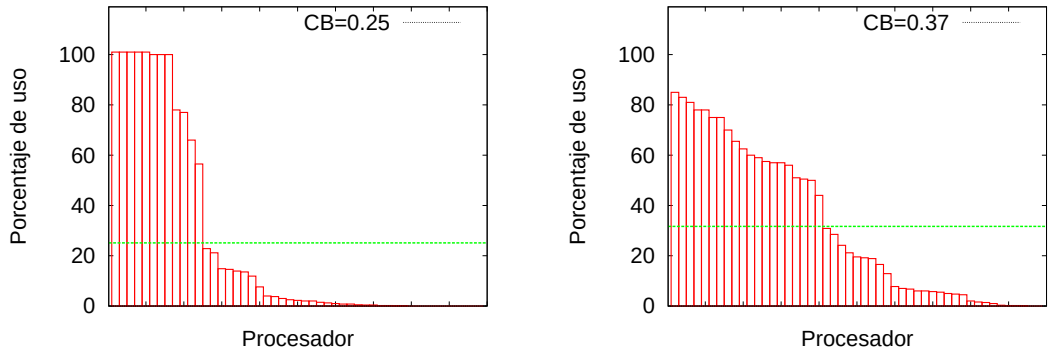


(c) Valor ideal de CB=1

Figura 4.26: Ejemplos de diferentes valores de la medida de Calidad de Balanceo: Malo (a), Bueno (b) e Ideal (c).

4.10.1 Comparación del promedio de uso por procesador en una red tipo RGG

En las siguientes gráficas se muestra el promedio de uso de los procesadores, la línea horizontal muestra el valor promedio; además se muestra el valor de la medida de calidad de balanceo *CB*. Los valores de porcentaje de uso se encuentran ordenados de mayor a menor.



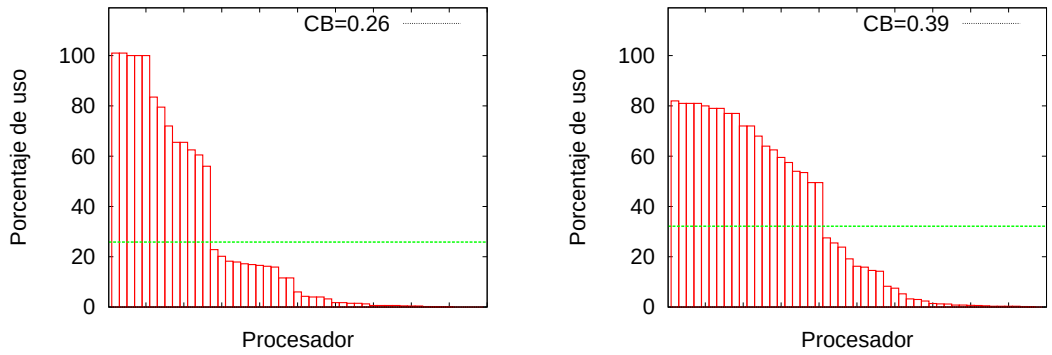
(a) Promedio de uso de procesadores en una red tipo RGG sin balanceo.

(b) Promedio de uso de procesadores en una red tipo RGG con método de balanceo **Robar Uno**.

Figura 4.27: Comparación del promedio de uso por procesador en una red tipo RGG.

En las gráficas anteriores se observa que aplicando el método de balanceo **Robar Uno**, se obtiene un valor de *CB* un poco mejor que si no se aplica ningún balanceo de carga. También se observa que el promedio de uso de los procesadores es menor en el segundo caso, eso significa que los procesadores están desocupados más tiempo.

4.10.2 Comparación del promedio de uso por procesador en una red tipo Kleinberg



(a) Promedio de uso de procesadores en una red tipo Kleinberg sin balanceo.

(b) Promedio de uso de procesadores en una red tipo Kleinberg con método de balanceo **Robar Uno**.

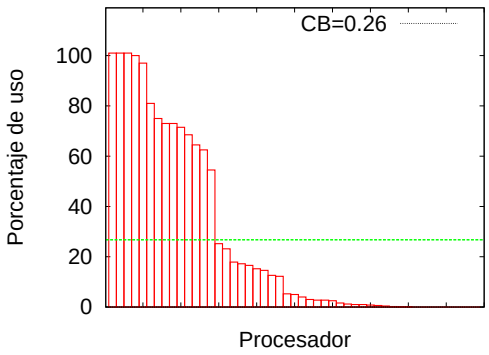
Figura 4.28: Comparación del promedio de uso por procesador en una red tipo Kleinberg.

Al igual que con las redes tipo RGG, en las tipo Kleinberg se observa que aplicando el método de balanceo **Robar Uno**, se obtiene un valor de *CB* un poco mejor que si no se aplica ningún balanceo de carga.

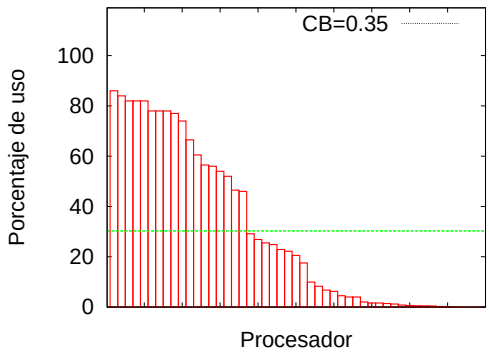
4.10.3 Comparación del promedio de uso por procesador en una red tipo Malla

Al igual que con los anteriores, en las redes tipo Malla se observa que aplicando el método de balanceo **Robar Uno**, se obtiene un valor de *CB* un poco mejor que si no se aplica ningún balanceo de carga; también que el promedio de uso de los procesadores es menor aplicando el método. Basados en los resultados de la figura 4.23 se probó también el otro método de generación de carga.

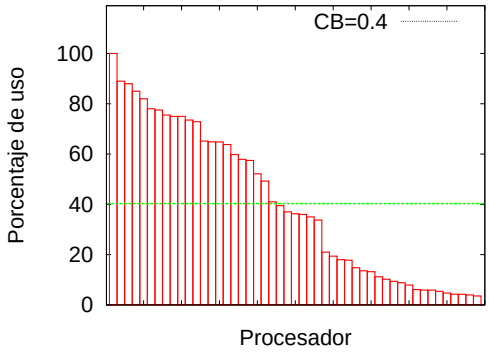
Como habíamos mencionado anteriormente aumentando el valor del número de trabajos esperados se presenta sobrecarga en la red. Como se puede observar en las siguientes gráficas, mientras algunos procesadores se encuentran en promedio al



(a) Promedio de uso de procesadores en una red tipo Malla sin balanceo.



(b) Promedio de uso de procesadores en una red tipo Malla con generación de carga **Poisson** y método de balanceo **Robar Uno**.



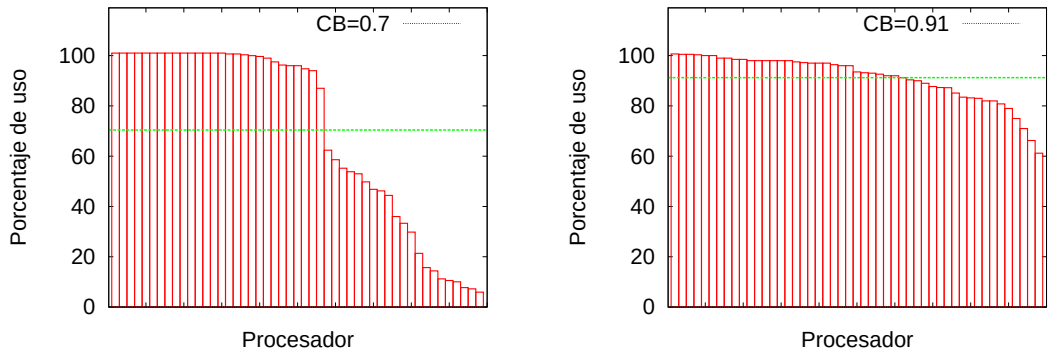
(c) Promedio de uso de procesadores en una red tipo Malla con generación de carga **aleatoria** y con método de balanceo **Robar Uno**.

Figura 4.29: Comparación del promedio de uso por procesador en una red tipo Malla.

100 % de su capacidad otros tienen muy poca utilización cuando no se aplica ningún método de balanceo.

4.10.4 Comparación del promedio de uso por procesador en una red RGG con sobrecarga

Si continúa aumentando el valor de la cantidad esperada de trabajos se observa que todos los procesadores se encuentran al 100 % de su capacidad sin importar el método de balanceo que se aplique en la red. En este caso la medida **CB** es igual a



(a) Promedio de uso de procesadores en una red tipo RGG sin balanceo.

(b) Promedio de uso de procesadores en una red tipo RGG método de balanceo **Enviar Varios**.

Figura 4.30: Comparación del promedio de uso por procesador en una red RGG con sobrecarga $\lambda = 8$.

uno porque todos los procesadores se encuentran con la misma carga 100 %.

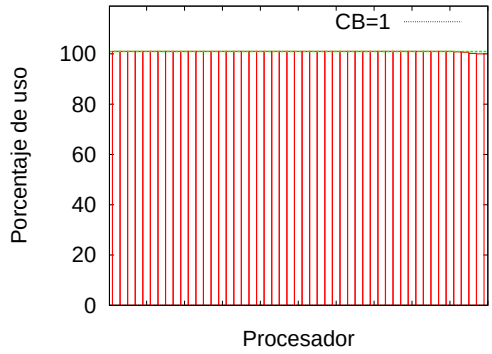
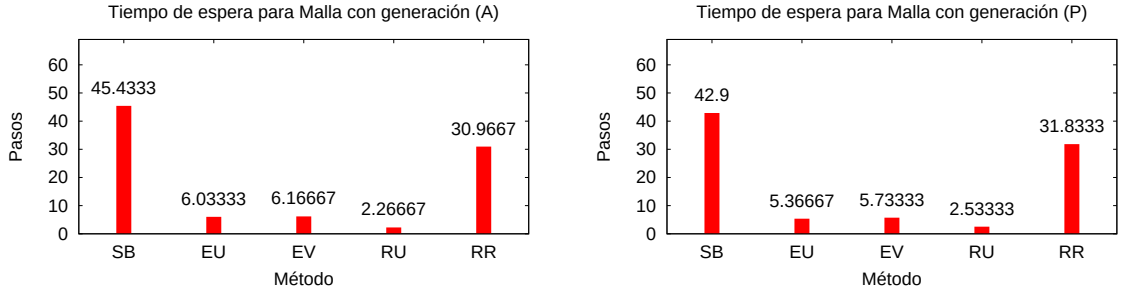


Figura 4.31: Medida **CB** para una red tipo RGG con sobrecarga.

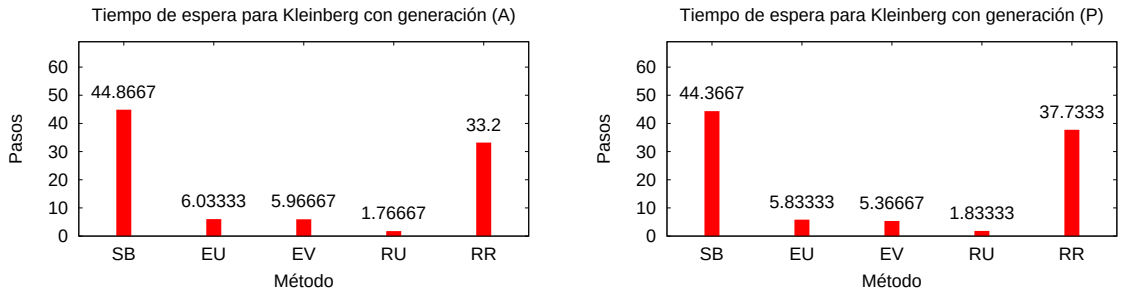
4.11 Tiempo de espera de los trabajos

Se midió el tiempo máximo en número de pasos que requiere un trabajo para ser procesado, el tiempo se mide desde el momento en que es generado el trabajo y hasta que es procesado. Se observó que el tiempo que esperan los trabajos para ser procesados es mucho menor cuando se aplica un método de balanceo que cuando no se aplica.



(a) Generación de carga tipo **Aleatoria** (b) Generación de carga tipo **Poisson**

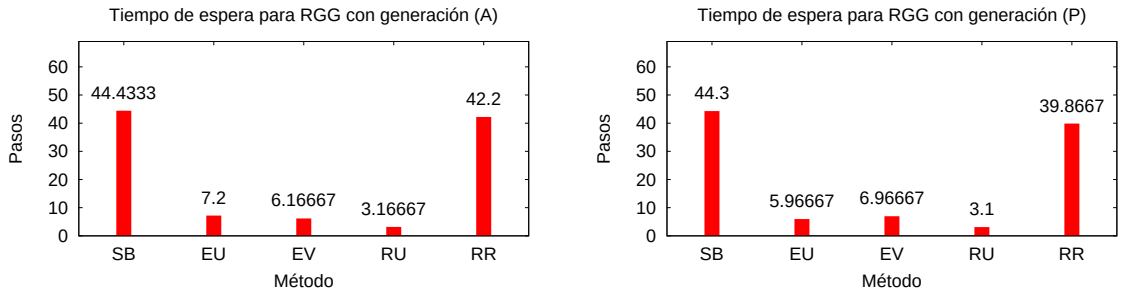
Figura 4.32: Comparación de tiempos de espera para una red tipo Malla.



(a) Generación de carga tipo **Aleatoria** (b) Generación de carga tipo **Poisson**

Figura 4.33: Comparación de tiempos de espera para una red tipo Kleinberg.

En todos los casos se observa que aplicar un método de balanceo disminuye considerablemente el tiempo de espera de los trabajos, con excepción del método **Round Robin**. El método de balanceo que tiene un menor tiempo de espera promedio es el método **Robar Uno**. Esto puede ser ocasionado porque en el método **Robar Uno** se mueven muchos menos trabajos en cada paso que en el método **Round Robin**.



(a) Generación de carga tipo **Aleatoria**

(b) Generación de carga tipo **Poisson**

Figura 4.34: Comparación de tiempos de espera para una red tipo RGG.

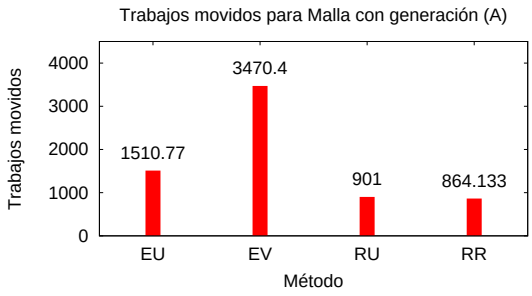
	Malla		Kleinberg		RGG	
	A	P	A	P	A	P
SB	*45.43*	42.9	44.86	44.36	44.43	44.3
EU	6.03	5.36	6.03	5.83	7.2	5.96
EV	6.16	5.73	5.96	5.36	6.16	6.96
RU	2.22	2.53	*1.76*	1.83	3.16	3.1
RR	30.96	31.83	33.2	37.63	42.2	39.86

Tabla 4.2: Comparación de tiempos de espera de trabajos.

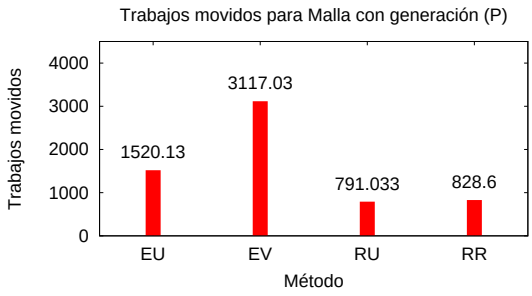
El menor tiempo de espera para los trabajos se presenta en las redes tipo Kleinberg con el método de balanceo **Robar Uno**.

4.12 Cantidad de trabajos enviados de un procesador a otro

Es importante conocer la cantidad de trabajos que son movidos de un procesador a otro ya que el mover un trabajo dentro de la red va a tener un costo. Durante la simulación se contabilizó el número de trabajos que se mueven de un procesador a otro para cada método de balanceo.

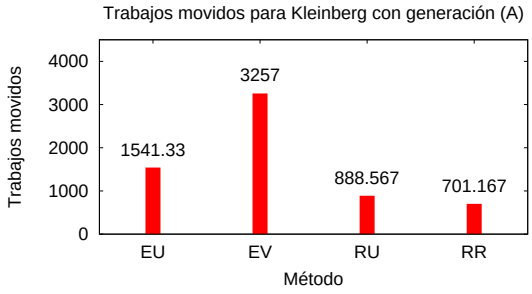


(a) Generación de carga tipo **Aleatoria**

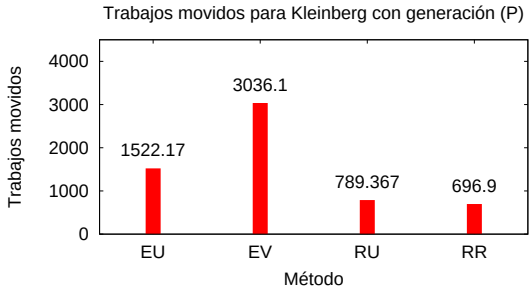


(b) Generación de carga tipo **Poisson**

Figura 4.35: Comparación de cantidad de trabajos movidos para una red tipo Malla.

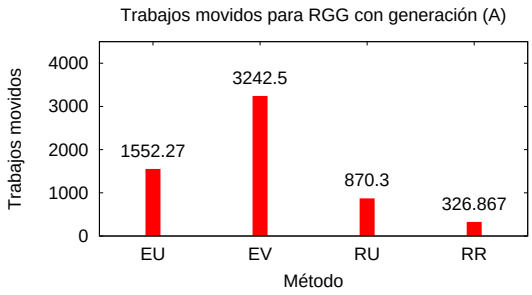


(a) Generación de carga tipo **Aleatoria**

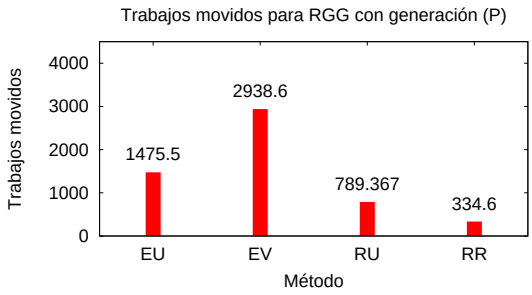


(b) Generación de carga tipo **Poisson**

Figura 4.36: Comparación de cantidad de trabajos movidos para una red tipo Kleinberg



(a) Generación de carga tipo **Aleatoria**



(b) Generación de carga tipo **Poisson**

Figura 4.37: Comparación de cantidad de trabajos movidos para una red tipo RGG.

Para todos los casos el método **Enviar Varios** es el que mueve una mayor cantidad de trabajos de un procesador a otro. En el método **Round Robin** también

	Malla		Kleinberg		RGG	
	A	P	A	P	A	P
EU	1510.77	1520.13	1541.33	1522.17	*1552.27*	1475.5
EV	3470.4	3117.03	3257	3036.1	3242.5	2938.6
RU	901	791.03	888.56	789.36	870.3	789.36
RR	864.13	828.6	701.16	696.9	*326.86*	334.6

Tabla 4.3: Comparación de cantidad de trabajos movidos.

se mueven varios trabajos a la vez, pero solamente se envía un trabajo a un vecino; por ejemplo si un nodo no tiene dos vecinos y varios trabajos en espera, con el método **Round Robin** solo podrá enviar dos trabajos, mientras que si se aplica el método **Enviar Varios** se moverán todos los trabajos que tiene en la cola. Es por esto que el método **Enviar Varios** mueve mucho más cantidad de trabajos que el método **Round Robin**.

4.13 Comparación general de métodos.

Utilizando los parámetros de Sharma et al. [34] presentados en la sección 2.3 se realizó una comparación de los métodos utilizados en este trabajo. Los resultados se presentan en la tabla 4.4.

Parámetros	EU	EV	RU	RR
Rechazo de sobrecarga	No	No	No	No
Tolerancia a fallas	Si	Si	Si	Si
Precisión de previsión	Menos	Menos	Menos	Menos
Estabilidad	Grande	Pequeña	Grande	Pequeña
Centralizado/Descentralizado	D	D	D	D
Dinámico/Estático	Di	Di	Di	Di
Cooperativo	No	No	Si	Si
Migración de procesos	No	No	No	No
Utilización de recursos	Más	Menos	Más	Menos

Tabla 4.4: Comparación de algoritmos de balanceo.

Capítulo 5

Conclusiones y trabajo futuro

Se programó una herramienta de software con una interfaz de fácil uso que permite simular el balanceo de carga en redes. Este simulador permite crear tres diferentes tipos de red Malla, Kleinberg y RGG con diferentes parámetros; se puede generar carga de trabajo sobre estas redes utilizando dos métodos diferentes de generación de carga. Esta herramienta también permite aplicar cuatro diferentes métodos de balanceo de carga sobre las redes generadas. Se puede visualizar la red y la carga que se genera en esta. Además de mostrar las gráficas de la carga total del sistema y de la medida *CB* en cada simulación.

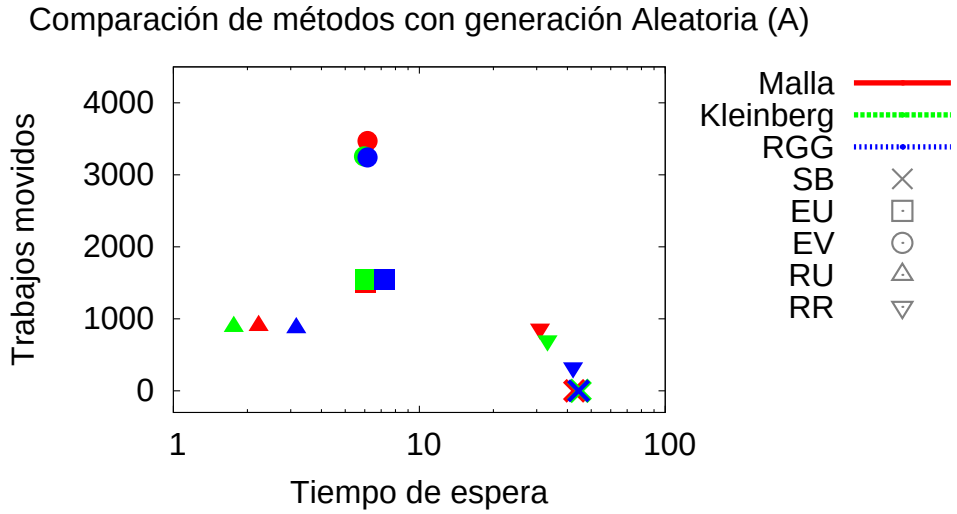
Utilizando la herramienta de simulación se llevaron a cabo varios experimentos. Se observó que los métodos de balanceo utilizados disminuyen la carga total del sistema y el tiempo de espera de los trabajos en la red desde que son creados y hasta que son procesados. De acuerdo con los experimentos realizados, podemos notar que el método de balanceo **Robar Uno** es el que tiene mejores resultados en cuanto a la carga total del sistema, esto en los tres tipos de redes Malla, Kleinberg y RGG utilizando el método de generación de carga **Poisson** presentando solo muy poca diferencia con el método **Aleatorio**. Para evitar que se incremente la carga del sistema cuando se aplican métodos que envían trabajos, lo conveniente sería medir la utilización de cada procesador durante un periodo de tiempo y enviar el trabajo a aquellos que han tenido un valor bajo de utilización. Si se envía un trabajo a cualquier procesador aunque éste desocupado, este va a seguir generando sus propias tareas, es por eso que se genera la sobrecarga. En el caso del método **Round Robin** como

más nodos se involucran en el método es por eso que se genera una sobrecarga mayor que en los demás métodos.

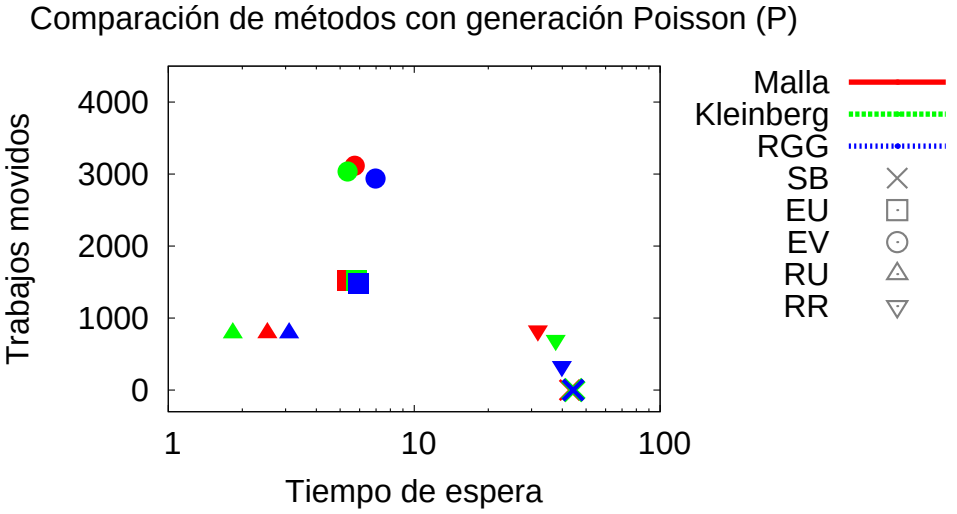
En cuanto a las condiciones ambientales, en los experimentos realizados también se observó que el tamaño de las redes no afecta el desempeño de los métodos de balanceo. El tipo de generación de carga no hace tampoco alguna diferencia, pero si la cantidad de trabajos que se espera que se generen en la red, ya que cuando son demasiados los trabajos que se están generando en la red ninguno de lo métodos utilizados logra mantener estable la carga total del sistema. Cuando la cantidad de trabajos generada es pequeña no tiene sentido aplicar métodos de balanceo ya que los procesadores son capaces de manejar su carga; aplicar métodos de balanceo en este caso solo ocasionaría costos innecesarios. Además de la cantidad de trabajos generados el grado promedio de los nodos también afecta los métodos de balanceo, ya que si un nodo tiene un grado muy alto tiene más opciones a donde enviar la sobrecarga y es más fácil realizar el balanceo de carga.

En la figura 5 podemos observar que de acuerdo a nuestras métricas de tiempo de espera y cantidad de trabajos movidos el método de balanceo **Robar Uno** es el que presenta un mejor desempeño ya que se consigue el menor tiempo de espera con la menor cantidad de trabajos que son movidos de un procesador a otro. Como se mancionó anteriormente se ha observado que los esquemas de robar trabajos son los que presentan mejores resultados [7, 24]. El segundo mejor método de acuerdo a éstas métricas es **Enviar Uno**.

Como trabajo futuro se podrían proponer otros métodos de balanceo más eficientes basados en las observaciones realizadas y compararlos con los actuales. También se pueden comparar y proponer mejoras a los métodos de balanceo que utilizan sistemas de agentes. Estos métodos se aprovecharían las propiedades de cada red para la cual se va a aplicar. Se podrían diseñar métodos específicos que se adapten a la topología de la red y a la cantidad de trabajos que se están generando en la red.



(a) Comparación para generación **Aleatoria**



(b) Comparación para generación **Poisson**

Figura 5.1: Comparación de métodos, tiempo de espera vs. trabajos movidos.

Bibliografía

[1] «Colt Project», URL `http://acs.lbl.gov/~hoschek/colt/`.

[2] «The Network Simulator ns-2», URL `http://www.isi.edu/nsnam/ns/`.

[3] Adamic, L. A., «The Small World Web», en S. Abiteboul y A.-M. Vercoustre (editores), *Proceedings of ECDL'99*, *Lecture Notes in Computer Science*, tomo 1696, Springer-Verlag GmbH, Berlin, Alemania, págs. 443–452, 1999.

[4] Androutsellis-Theotokis, S. y D. Spinellis, «A survey of peer-to-peer content distribution technologies», *ACM Computing Surveys*, **36**(4), págs. 335–371, diciembre 2004.

[5] Babaoglu, O., H. Meling y A. Montresor, «Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems», IEEE Computer Society, Los Alamitos, CA,EE.UU., pág. 15, 2002.

[6] Babaoglu, O., H. Meling y A. Montresor, «Messor: Load-Balancing through a Swarm of Autonomous Agents», en *Agents and Peer-to-Peer Computing*, *Lecture Notes in Computer Science*, tomo 2530, Springer-Verlag GmbH, Berlin/Heidelberg, Alemania, págs. 125–137, 2008.

[7] Berenbrink, P., T. Friedetzky y L. A. Goldberg, «The Natural Work-Stealing Algorithm is Stable», en *IEEE Symposium on Foundations of Computer Science*, págs. 178–187, 2001.

[8] Bustos Jiménez, J., *Dynamic load balancing for active objects on computer grids*, Tesis Doctoral, Universidad de Nice-Sophia, Antipolis, Francia, diciembre 2006.

[9] Buyya, R. y S. Venugopal, «A Gentle Introduction to Grid Computing and Technologies», *CSI Communications*, **29**(1), págs. 9–19, julio 2005.

[10] Clarke, I., O. Sandberg, B. Wiley y T. W. Hong, «Freenet: A Distributed Anonymous Information Storage and Retrieval System», en *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability 2000*, *Lecture Notes in Computer Science*, tomo 2009, Springer-Verlag GmbH, págs. 46–66, 2001.

[11] Diestel, R., *Graph Theory*, *Graduate Texts in Mathematics*, tomo 173, Springer-Verlag, New York, NY, EE.UU., 2000.

[12] Foster, I., «What is the Grid? A Three Point Checklist.», Columna in GRIDToday, julio 2002.

[13] Garg, V. K., *Elements of distributed computing*, John Wiley & Sons, Inc., New York, NY, EE.UU., 2002.

[14] George Colorius, T. K., Jean Dollimore, *Distributed Systems Concepts and Design*, cuarta edición, Addison Wesley, Norwell, MA, EE.UU., 2005.

[15] Ghosh, S., *Distributed Systems an Algorithmic Approach*, Chapman & Hall/CRC, Boca Raton, FL, EE.UU., 2007.

[16] Godfrey, B., K. Lakshminarayanan, S. Surana, R. Karp y I. Stoica, «Load Balancing in Dynamic Structured P2P Systems», en *Proceedings IEEE INFOCOM*, Hong Kong, 2004.

[17] I. Foster, S. T., C. Kesselman, «The Anatomy of the Grid: Enabling Scalable Virtual Organizations.», *International Journal Supercomputer Applications*, **15**(3), 2001.

[18] Karger, D. y M. Ruhl, «Simple efficient load balancing algorithms for peer-to-peer systems», en *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, ACM, New York, NY, EE.UU., págs. 36–43, 2004.

[19] Kleinberg, J. M., «The small-world phenomenon: an algorithmic perspective», en *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing (STOC)*, ACM Press, New York, NY, EE.UU., págs. 163–170, 2000.

[20] Leibowitz, N., M. Ripeanu y A. Wierzbicki, «Deconstructing the Kazaa Network», *IEEE Workshop on Internet Applications*, pág. 112, 2003.

[21] Li, D.-C. y F. M. Chang, «An In–Out Combined Dynamic Weighted Round-Robin Method for Network Load Balancing», *Computer Journal*, **50**(5), págs. 555–566, 2007.

[22] Lv, Q., P. Cao, E. Cohen, K. Li y S. Shenker, «Search and Replication in Unstructured Peer-to-Peer Networks», en *Proceedings of the Sixteenth Annual ACM International Conference on Supercomputing*, ACM, New York, NY, EE.UU, págs. 84–95, 2002.

[23] Lynch, N. A., *Distributed Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, EE.UU., 1996.

[24] Mitzenmacher, M., «Analyses of load stealing models based on differential equations», en *In Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures*, págs. 212–221, 1998.

[25] Naicken, S., B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman y D. Chalmers, «The state of peer-to-peer simulators and simulations», *SIGCOMM Computer Communication Review*, **37**(2), págs. 95–98, 2007.

[26] Pathan, M., comunicación personal, noviembre 2008.

[27] Penrose, M., *Random Geometric Graphs*, Oxford University Press, New York, EE.UU., 2003.

[28] Ratnasamy, S., P. Francis, M. Handley, R. Karp y S. Schenker, «A scalable content-addressable network», en *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM, New York, NY, EE.UU., 2001.

[29] Reinefeld, A. y F. Schintke, «Concepts and technologies for a worldwide Grid infrastructure», **2400**, págs. 62–71, 2002.

[30] Risson, J. y T. Moors, «Survey of research towards robust peer-to-peer networks: search methods», *Computer Networks: The International Journal of Computer and Telecommunications Networking*, **50**(17), págs. 3485–3521, diciembre 2006.

[31] Schaeffer, S. E., «Algorithms for Nonuniform Networks», Research Report A102, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finlandia, mayo 2006.

[32] Schopf, J. y B. Nitzberg, «Grids: Top Ten Questions.», *Scientific Programming*, **10**(2), págs. 103–111, special issue on Grid Computing, agosto 2002.

[33] Sen, S. y J. Wang, «Analyzing peer-to-peer traffic across large networks», en *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, ACM, New York, NY, EE..UU., págs. 137–150, 2002.

[34] Sharma, S., S. Singh y M. Sharma, «Performance Analysis of Load Balancing Algorithms», en *Proceedings of World Academy of Science, Engineering and Technology*, tomo 28, abril 2008.

[35] Stender, J., S. Kaiser y S. Albayrak, «Mobility-based Runtime Load Balancing in Multi-Agent Systems», en *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, Knowledge Systems Institute, pág. 688, 2006.

[36] Sterling, T., D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake y C. V. Packer, «BEOWULF: A parallel workstation for scientific computation», en *Proceedings of the 24th International Conference on Parallel Processing*, CRC Press, págs. 11–14, 1995.

[37] Stoica, I., R. Morris, D. Karger, M. F. Kaashoek y H. Balakrishnan, «Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications», págs. 149–160, 2001.

[38] Wang, Y. y J. Liu, «Dynamics of agent-based load balancing on grids», en *Proceedings of the 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, págs. 804–811, 2003.

[39] Wang, Y., J. Liu y X. Jin, «Modeling Agent-Based Load Balancing with Time Delays», en *IEEE International Conference on Intelligent Agent Technology*, págs. 189–195, 2003.

[40] Zhao, B. Y., B. Y. Zhao, J. Kubiatowicz, J. Kubiatowicz, A. D. Joseph y A. D. Joseph, «Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing», Informe técnico, 2001.

Apéndice A

Funcionamiento del simulador

En esta sección se explicará más a detalle las funciones de la herramienta de simulación.

A.1 Diagrama del simulador

En la figura A.1 se muestra el diagrama de las clases que conforman el simulador.

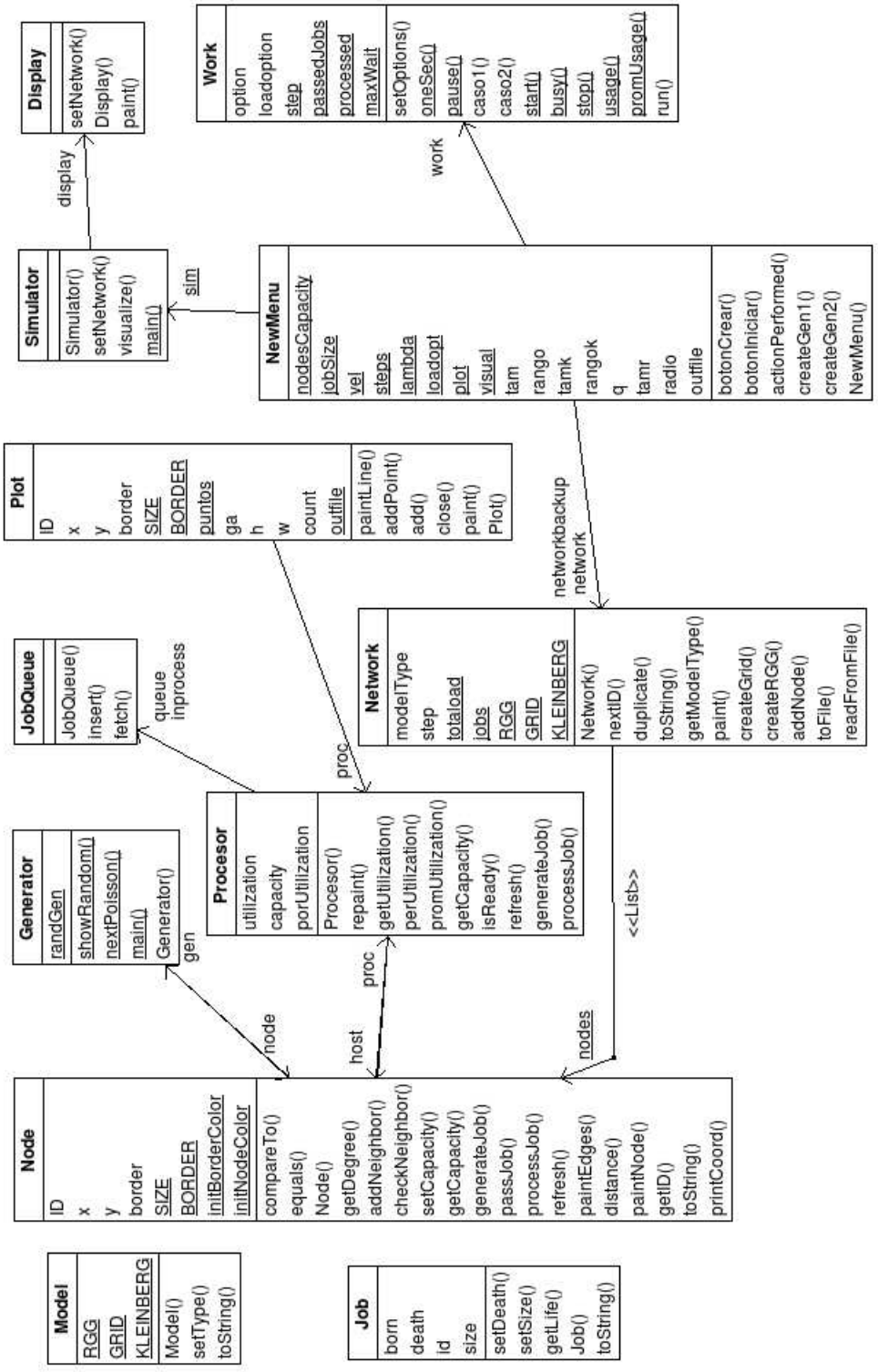


Figura A.1: Diagrama de clases del simulador.

A.2 Funcionamiento del simulador

A.2.1 Generación del modelo

Se tienen tres opciones para generar el modelo de red: *Malla, Kleinberg y RGG por sus siglas en inglés.*

- **Malla:** Son necesarios dos parámetros: la *dimensión de la malla* (n) y el *rango.*
- **Kleinberg:** Utiliza los parámetros *dimensión de malla* y *rango* y además se utiliza un parámetro q.
- **RGG:** Son necesarios el *número de nodos* y el **radio**.

A.2.2 Generación de tareas

Se pueden seleccionar tres diferentes métodos de generación de tareas para la simulación, cuando se genera una tarea en un nodo, esta tarea se añade a la cola del procesador asociado al nodo.

Cuando una tarea es generada tiene un *ID* único que conserva durante toda la simulación, además se fija el valor de nacimiento de la tarea con el paso de la simulación en el cual fue creada la tarea.

El método de generación de tareas es seleccionado al inicio de la simulación y no puede ser cambiado hasta una nueva simulación.

Los diferentes métodos de generación de tareas son los siguientes: Método G1, Método G2 y Método G3.

A.2.3 Método **Inicio**

Genera tareas aleatoriamente en cada uno de los nodos, solo al inicio de la simulación.

A.2.4 Método **Aleatorio**

Utilizando el método **Aleatorio** en cada paso de la simulación se generan tareas para cada uno de los procesadores de acuerdo a una distribución uniforme con valores entre cero y el valor máximo seleccionado. El tamaño de la tarea es seleccionado aleatoriamente entre cero y el tamaño máximo de tarea seleccionado.

A.2.5 Método **Poisson**

Cuando se selecciona el método de generación de tareas **Poisson** en cada paso de la simulación se generan tareas para cada uno de los procesadores utilizando una distribución de Poisson. El tamaño de la tarea es seleccionado aleatoriamente entre cero y el tamaño máximo de tarea seleccionado.

A.2.6 Procesamiento de tareas

Cada nodo tiene asociado un procesador, el cual tiene una capacidad de procesamiento y una cola de espera. Cuando se llama a la función de procesamiento de tareas para un procesador se lleva a cabo el siguiente procedimiento:

Si hay tareas en la cola del procesador se pasan las tareas al procesador, sin exceder su capacidad Las tareas permanecen en el procesador hasta el inicio del siguiente paso de la simulación cuando son marcadas como hechas y se fija el valor de muerte de la muerte de la tarea con el paso de la simulación.

Este procedimiento se repite para cada uno de los nodos, en cada paso de la simulación.

A.2.7 Funcionamiento de métodos

En el simulador se tienen cuatro tipos diferentes de métodos de balanceo, el método de balanceo se selecciona antes de iniciar la simulación. Los métodos de

balanceo son los siguientes: Método **Sin Balanceo**, Método **Enviar Uno**, Método **Enviar Varios**, Método **Robar Uno** y **Round Robin**.

Método **Sin Balanceo (SB)**

En el método **Sin Balanceo** no se realiza ningún balanceo de carga, este método es solo con fines comparativos.

Método **Enviar Uno (EU)**

El método de balanceo **Enviar Uno**, consiste en pasar un trabajo al vecino más cercano que este libre. Si después de que un procesador procesa sus trabajos aún tiene más trabajos en la cola de espera, entonces busca en su lista de nodos vecino aquel vecino que no tenga trabajos, si se encuentra un vecino libre se le pasa un trabajo. Para pasar un trabajo, el trabajo se elimina de la cola de espera del nodo que lo envía y se agrega en la cola de espera del nodo que recibe.Durante la simulación se contabiliza cada vez que un trabajo es movido a otro nodo. Este procedimiento se realiza una vez para cada uno de los nodos durante cada paso de la simulación.

Método **Enviar Varios (EV)**

El método de balanceo **Enviar Varios**, consiste en pasar todos los trabajos que el vecino más cercano que este libre pueda recibir. Si después de que un procesador procesa sus trabajos aún tiene más trabajos en la cola de espera, entonces busca en su lista de nodos vecino aquel vecino que no tenga trabajos, si se encuentra un vecino libre entonces le pasa todos los trabajos que pueda procesar.

Los trabajos movidos se eliminan de la cola de espera del nodo que lo envía y se agregan en la cola de espera del nodo que recibe. Durante la simulación se contabiliza cada vez que un trabajo es pasado a otro nodo. Este procedimiento se realiza una vez para cada uno de los nodos durante cada paso de la simulación.

Método **Robar Uno (RU)**

El método de balanceo **Robar Uno**, consiste en robar un trabajo a un vecino que tenga trabajos en la cola de espera. Cuanto un procesador termina de procesar sus trabajos busca en su lista de nodos vecinos aquel vecino que tenga trabajo en la cola, entonces toma el trabajo de la cola de ese procesador y lo agrega en su cola de espera.

Los trabajos movidos se eliminan de la cola de espera del nodo que lo envía y se agregan en la cola de espera del nodo que recibe. Durante la simulación se contabiliza cada vez que un trabajo es pasado a otro nodo. Este procedimiento se realiza una vez para cada uno de los nodos durante cada paso de la simulación.

A.2.8 Método **Round Robin**

En este método se envían todos los trabajos que se encuentran en la cola de espera del procesador a diferentes nodos vecinos, un trabajo a cada vecino de la lista de vecinos del nodo, hasta que ya no existan trabajos en la cola. Los trabajos movidos se eliminan de la cola de espera del nodo que lo envía y se agregan en la cola de espera del nodo que recibe. Durante la simulación se contabiliza cada vez que un trabajo es pasado a otro nodo. Este procedimiento se realiza una vez para cada uno de los nodos durante cada paso de la simulación.

Apéndice B

Generación de números aleatorios

Para generar los numeros aletaorios se utilizó la librería gratuita Colt [1] para Java. A continuación se muestran algunas gráficas de prueba para la generación de números aleatorios para una distribución uniforme y Poisson.

Un millón de datos generados de una distribución uniforme, con valores entre 0 y 100.

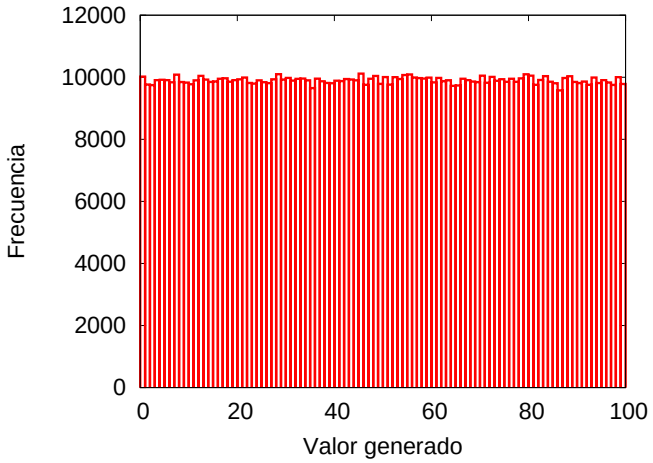


Figura B.1: Datos generados de una distribución uniforme

100000 datos generados de una distribución tipo Poisson con $\lambda = 50$.

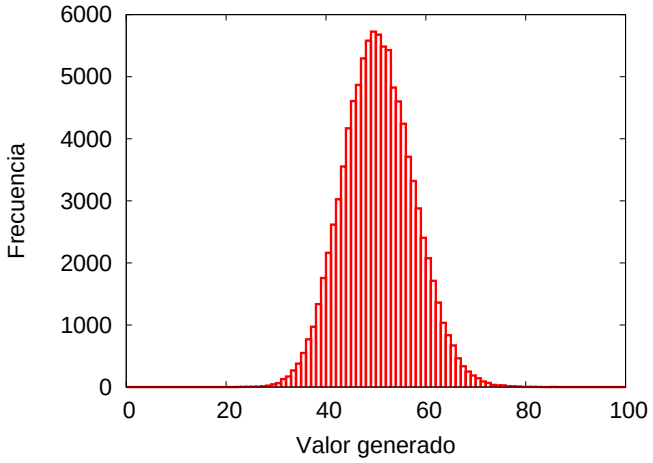


Figura B.2: Datos generados de una distribución Poisson

Apéndice C

Gráficas de experimentos de tamaño mediano

A continuación se muestran algunas gráficas de experimentos de tamaño mediano.

C.1 Gráficas de tamaño mediano para Malla

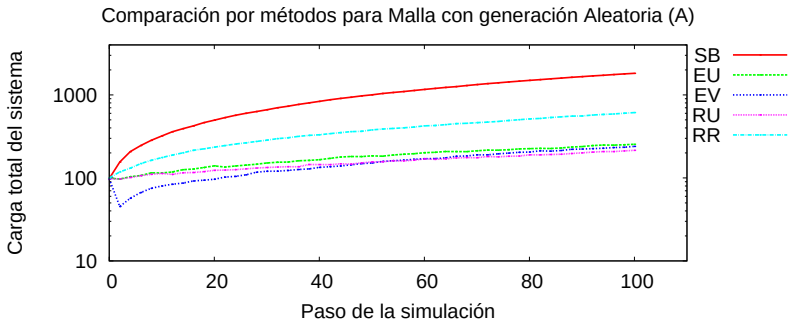
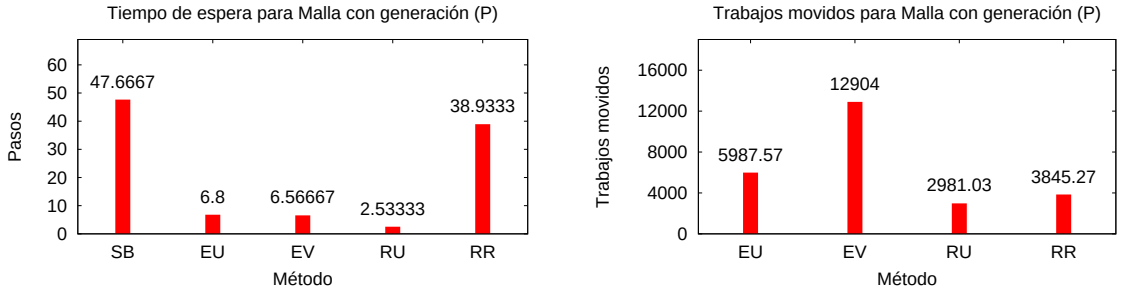


Figura C.1: Comparación de métodos para una malla tamaño mediano.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura C.2: Comparación de tiempos de espera y trabajos movidos para malla tamaño mediano.

C.2 Gráficas de tamaño mediano para Kleinberg

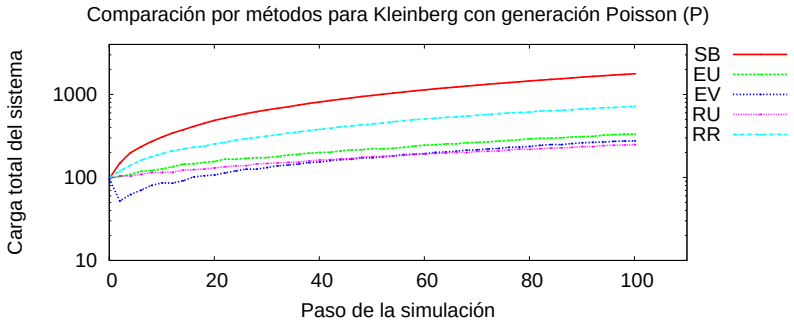
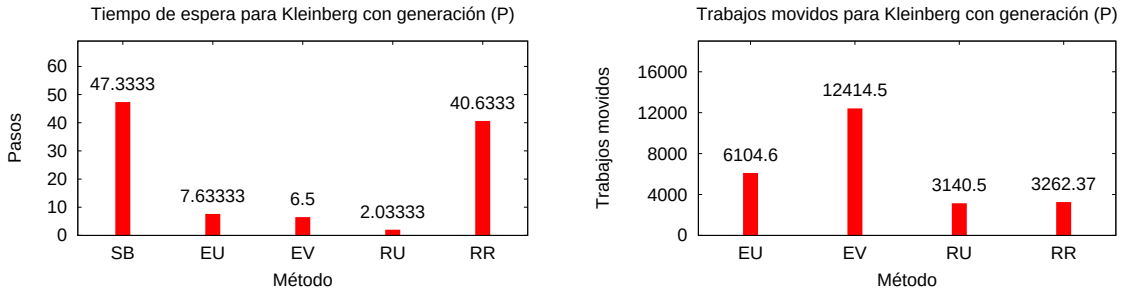


Figura C.3: Comparación de métodos para una Kleinberg tamaño mediano.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura C.4: Comparación de tiempos de espera y trabajos movidos para Kleinberg tamaño mediano.

C.3 Gráficas de tamaño mediano para RGG

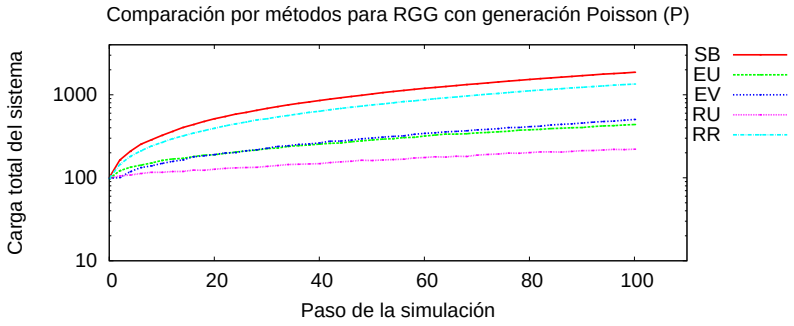
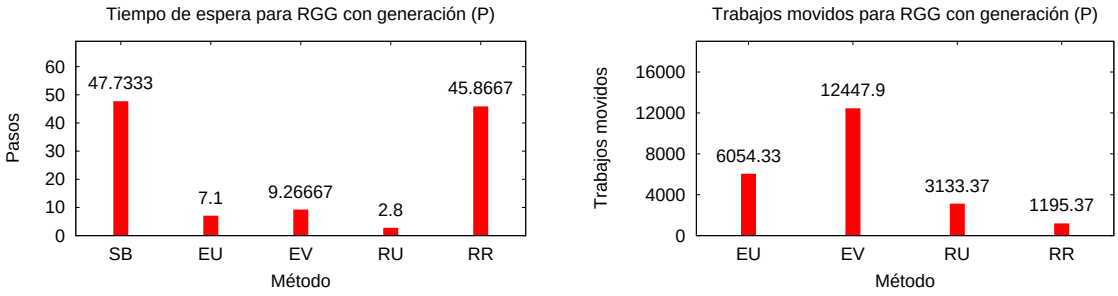


Figura C.5: Comparación de métodos para RGG tamaño mediano.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura C.6: Comparación de tiempos de espera y trabajos movidos para RGG tamaño mediano.

Apéndice D

Gráficas de experimentos de tamaño grande

A continuación se muestran algunas gráficas de experimentos de tamaño grande.

D.1 Gráficas de tamaño grande para Malla

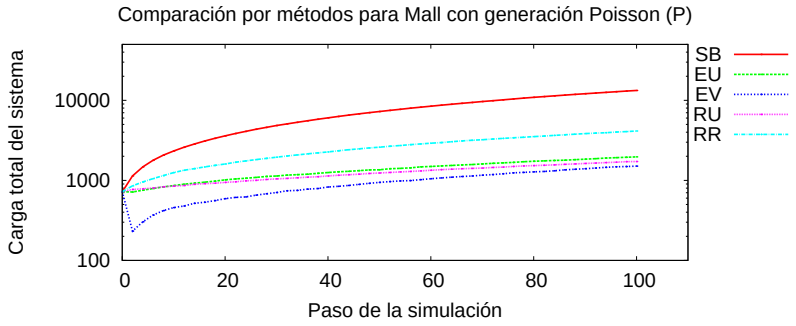
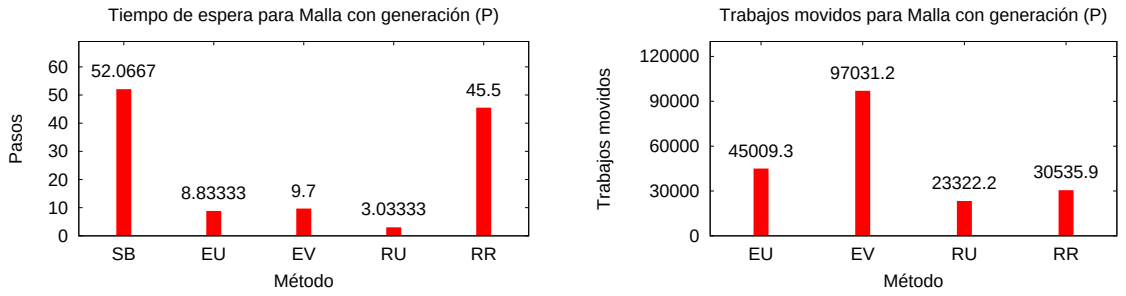


Figura D.1: Comparación de métodos para una malla tamaño grande.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura D.2: Comparación de tiempos de espera y trabajos movidos para malla tamaño grande.

D.2 Gráficas de tamaño grande para Kleinberg

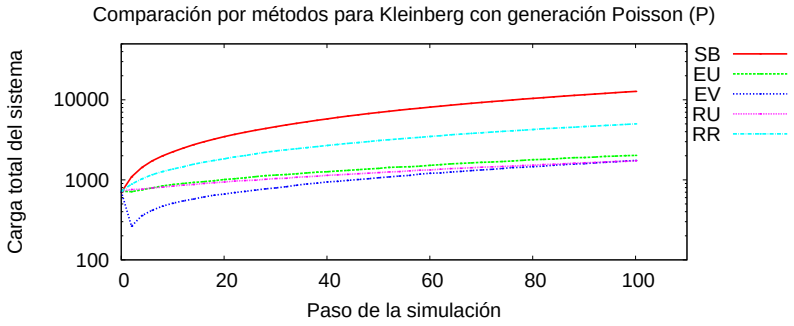
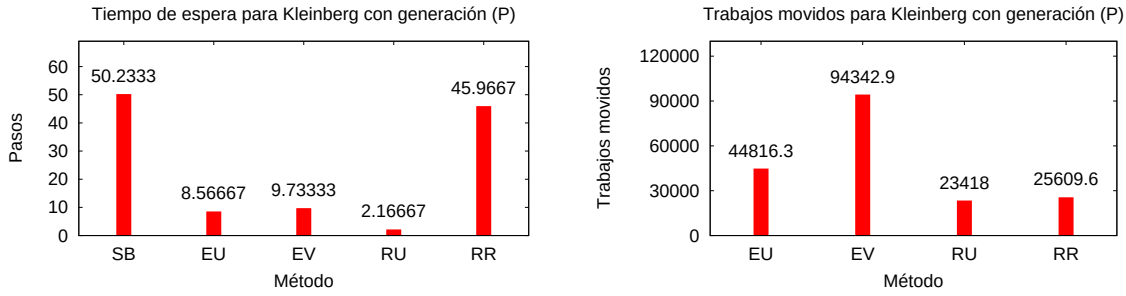


Figura D.3: Comparación de métodos para una Kleinberg tamaño grande.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura D.4: Comparación de tiempos de espera y trabajos movidos para Kleinberg tamaño grande.

D.3 Gráficas de tamaño grande para RGG

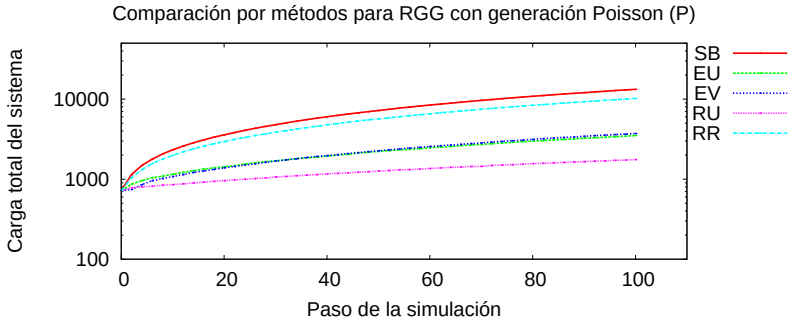
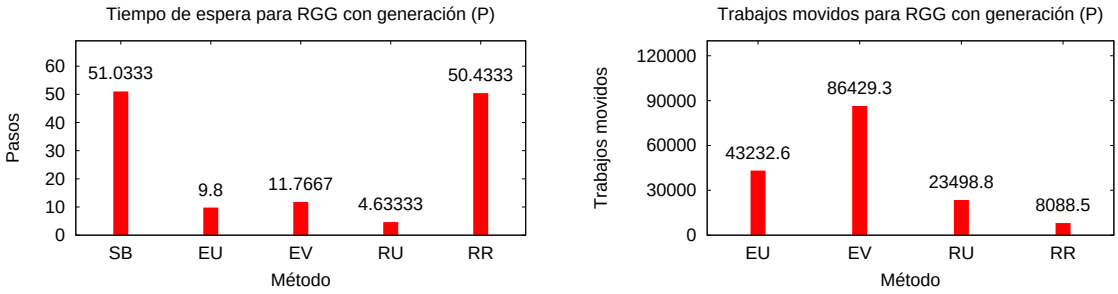


Figura D.5: Comparación de métodos para RGG tamaño grande.



(a) Comparación de tiempo de espera.

(b) Trabajos movidos.

Figura D.6: Comparación de tiempos de espera y trabajos movidos para RGG tamaño grande.

Ficha autobiográfica

David Juvencio Rios Soria

Candidato para el grado de Maestro en ciencias
en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Búsqueda de recursos para el balanceo dinámico de carga

Nací el 15 de noviembre de 1982 en la ciudad de Monterrey N.L., hijo primogénito de Daniel Rios Murguía y Rosa Ma. Soria Garza. Inicié mis estudios de Ingeniería en Electrónica y Comunicaciones en la Universidad Autónoma del Estado de Hidalgo que posteriormente finalicé en la Facultad de Ingeniería Mecánica y Eléctrica en la Universidad Autónoma de Nuevo León. Al finalizar continué con mis estudios de maestría en el Posgrado en Ingeniería de Sistemas de la misma facultad.