VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencias e Ingeniería de la Computación

"ESTUDIO DEL BALANCE DE CARGA EFICIENTE EN SISTEMAS DISTRIBUIDOS".

TRABAJO DE TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
EDGAR TISTA GARCÍA

TUTOR:
DR JORGE LUIS ORTEGA ARJONA
Facultad de Ciencias UNAM

MÉXICO, D. F. AGOSTO 2015

Gracias a todos aquellos que directa o indirectamente hicieron posible este trabajo

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

Un sistema distribuido es aquél donde los procesadores no comparten memoria o reloj. Pueden ubicarse en diferentes localizaciones geográficas, y la comunicación entre ellos se realiza generalmente mediante el paso de mensajes. [15]

El uso de estos sistemas ha crecido ámpliamente en los últimos años. Es por ello que la compartición de los recursos de cómputo se vuelve cada vez más importante para sistemas de mediana y gran escala. Paralelamente, las redes de computadoras han adquirido una mayor rapidez, permitiendo una mejora considerable en la comunicación de los procesadores que conforman sistemas distribuidos.

El objetivo principal de un sistema distribuido, y una de las principales razones que guían a su construcción, es (a) tener una cantidad mayor de recursos de cómputo ubicados en diferentes puntos geográficos, y (b) proporcionar un ambiente eficiente y conveniente para la compartición de los mismos cuyos nodos se puedan interconectar para realizar procesamiento de datos de manera más rápida que en un sistema centralizado. [2].

Otros objetivos importantes en la implementación y el uso de los sistemas distribuidos son la reducción en el tiempo de respuesta de las tareas, la disponibilidad y la escalabilidad. En principio, una estrategia adecuada para balance de carga presupone alcanzar tales objetivos[14].

El balance de carga es la técnica utilizada para distribuir o compartir el trabajo a realizar entre varios procesadores. Al tener todos los NODOS una carga de trabajo similar, se pueden aprovechar los recursos disponibles, e inclusive incrementar el número de tareas a ejecutar

1.2. Problemática

Entre los principales desafíos que surgen en la construcción, el uso y mantenimiento de un sistema distribuido se encuentran: la disponibilidad, la escalabilidad, la seguridad, la heterogeneidad y el desempeño. Es precisamente en este último donde el presente trabajo se enfoca.

Algunos de los principales problemas asociados al desempeño de sistemas distribuidos son los siguientes[2, 15]:

- Adquirir la información general del sistema y mantenerla lo más actualizada posible.
- Mantener la transparencia en el sistema por completo.
- Realización de trabajo equitativo por parte de los nodos.
- Alcanzar el mayor aprovechamiento posible de los recursos disponibles en elementos que lo conforman.

En la implementación de herramientas y técnicas para la resolución de estos problemas, destaca el balance de carga, que consiste en la redistribución inteligente de las tareas que se ejecutan. El balance de carga se asocia con el surgimiento y la ejecución de las tareas mediante su distribución transparente y equitativa.

Lo que se persigue es que los recursos se compartan de una manera equitativa entre los nodos participantes. Al mismo tiempo evitando que en un determinado momento se tengan nodos sin tareas por ejecutar y otros con una larga cola pendiente. El objetivo es alcanzar un equilibrio en la carga de trabajo.

En la literatura existe una amplia variedad de estrategias para alcanzar la mejora del desempeño de los sistemas distribuidos mediante el balance de carga. En este documento se analizan algunas de ellas, describiendo sus características, ventajas y desventajas. Basado en dicho análisis, se propone una estrategia de balance de carga para colaborar en la resolución de problemas asociados al desempeño de un sistema distribuido.

1.3. Hipótesis

"Un algoritmo de balance de carga implementado con lógica difusa en las reglas para migración de tareas, permite aprovechar de manera más equilibrada los recursos disponibles en sistema distribuido en relación con un algoritmo de balance de carga dinámico, basado en reglas convencionales."

1.4. Objetivo

El principal objetivo del presente trabajo es aportar una algoritmo que pueda conducir a obtener una mejora en el desempeño general de un sistema distribuido, mediante el alcance de un nivel de equilibrio en la distribución de carga de los nodos participantes en el sistema.

1.5. Aproximación

Para alcanzar el objetivo propuesto, se realiza un análisis detallado de algunas de las estrategias más relevantes que se encuentran en la literatura, se propone una estrategia basada en dicho análisis y se evalúa experimentalmente.

La forma de realizar la evaluación experimental de la solución propuesta es mediante la simulación de un sistema distribuido. Para ello se toma en cuenta un caso de estudio generado a partir de un modelo general, en el que se manejan diferentes valores de carga que representan un conjunto de tareas a procesar.

Se realiza una simulación debido a que es una forma adecuada en la que se puede verificar la validez de la propuesta así como el cumplimiento de los objetivos planteados en las condiciones definidas para ello.

1.6. Contribuciones

- La contribución principal es un algoritmo dinámico con elementos de lógica difusa, mediante el cual se establecen reglas para la toma de decisiones en la migración de carga y con ello lograr el balance de carga de forma eficiente.

- Descripción de las características de los algoritmos de balance de carga estáticos y dinámicos así como los elementos esenciales que los componen.

- La simulación realizada para el caso de estudio del sistema distribuido, junto con la descripción de las pruebas correspondientes.

1.7. Estructura general

El presente documento se ordena de la siguiente manera:

- En el **Capítulo 2** se describen los conceptos fundamentales, tales como: sistemas distribuidos, balance de carga y una introducción al campo de conocimiento de la lógica difusa, la cual se utiliza como herramienta auxiliar en las reglas del algoritmo presentado. Se enuncian las definiciones que se han propuesto en la literatura y las particulares de este trabajo.
- El **Capítulo 3** es una descripción del trabajo relacionado. Se analizan las características más importantes, ventajas y desventajas de las estrategias que se han planteado. El propósito principal es brindar un contexto de algunos de los avances que existen en la actualidad.
- En el **Capítulo 4** se describe la propuesta para solucionar el problema. Se plantea la arquitectura del sistema en el que se realizan las pruebas, la descripción del algoritmo propuesto, su aplicación y parámetros de funcionamiento.
- El **Capítulo 5** presenta los experimentos realizados para comprobar la propuesta de solución al problema de balance de carga. Se describen las características de simulación realizada y los aspectos particulares del modelo de sistema propuesto.
- Finalmente en el **Capítulo 6** se encuentran las conclusiones generales del trabajo. Así como una reenunciación de las aportaciones del mismo.

Capítulo 2

Antecedentes

En este capítulo se presentan los conceptos fundamentales para lograr un entendimiento básico en el dominio de los sistemas distribuidos. En el tema de balance de carga, se considera la división entre estrategias estáticas y dinámicas. En el caso de las primeras, se describen las utilizadas con mayor frecuencia, y en el caso de las últimas, se detallan las características fundamentales que las componen. Finalmente se brinda una descripción de los elementos básicos de lógica difusa para un entendimiento más amplio acerca de la construcción de las reglas que se incluyen en la propuesta de solución.

2.1. Sistemas Distribuidos

Los sistemas distribuidos surgen junto con el desarrollo las redes de computadoras. Al conectar dos o más dispositivos para compartir información, inmediatamente surge la necesidad de explotar el potencial que se tiene al contar con un conjunto de dos o más equipos interconectados y extenderlo de manera gradual.

Éstos no solo comparten información, sino también hacen uso de los otros recursos que puede tener un equipo de cómputo, (memoria disponible, espacio de almacenamiento, ciclos de reloj en el procesador, etc.) de manera transparente para los usuarios.

A continuación se analizan los aspectos esenciales que conforman un sistema distribuido, proporcionando un panorama general de las características, desafíos, y necesidades, como punto de partida para el estudio.

2.1.1. Definición

En la literatura se encuentra una amplia variedad de definiciones para el concepto de sistema distribuido. La mayor parte de ellas hacen énfasis en la compartición de los recursos, y otras, en la transparencia de los mismos.

Dentro de las definiciones más conocidas, aceptadas y citadas en la literatura, destacan las siguientes:

> *"Un sistema distribuido es una colección de computadoras independientes que se muestran a los usuarios del sistema como una sola"* [15].

> *"Sistema en el cual, los componentes de hardware y software ubicados en redes de computadoras comunican y coordinan sus acciones mediante paso de mensajes"*[2].

En la primera definición, implícitamente se define el concepto de transparencia. Además, al decir que se trata de computadoras independientes, se entiende que cada una tiene recursos propios específicos (procesador, memoria, HW). Sin embargo, la definición no establece cómo se conectan o comunican las computadoras.

En el caso de la segunda definición, por el contrario, se hace particular énfasis en la manera en la que se establece la comunicación en este tipo de sistemas. Pero no contempla la compartición de recursos.

A partir de las definiciones anteriores y mediante la combinación de ellas, se construye una definición para los fines de este trabajo;

Un sistema distribuido es un conjunto de computadoras independientes de características particulares, conectadas entre sí mediante una red, que comparten sus recursos de cómputo de manera transparente, y que se comunican entre sí mediante el modelo de paso de mensajes.

Un sistema distribuido se puede entender y analizar de manera similar a una red de datos. La diferencia fundamental entre ambos es que al ser distribuido, no solo se comparte información, sino también recursos de cómputo, sin importar dónde se ubiquen geográficamente los elementos que lo componen. Gráficamente, se pueden representar de diversas formas, desde simples gráficas, esquemas que representen la ubicación de equipos hasta complejos diagramas con un amplio nivel de detalle. (Figura 2.1).

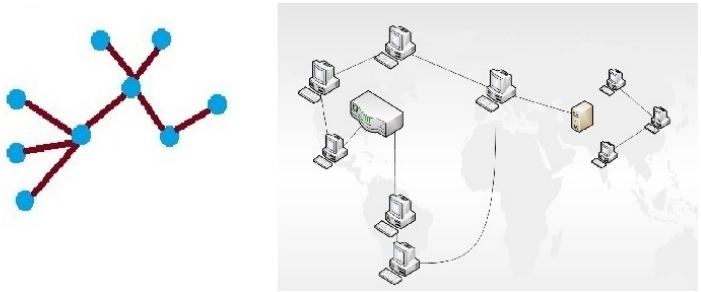

Figura 2.1: Ejemplos de Representaciones de Sistemas Distribuidos

2.1.2. Características Generales

Los sistemas distribuidos varían de acuerdo al número de computadoras/nodos que los conforman, y de las propiedades particulares de éstos, inclusive en la topología de la red mediante la que se conectan. No obstante, analizados en conjunto, se tienen las siguientes características generales en los sistemas distribuidos. [2]:

- Concurrencia.
  En el mismo instante de tiempo, cada usuario puede realizar un trabajo diferente utilizando los mismos recursos de uno o varios nodos. Por lo tanto, la ejecución simultánea de programas es factible. Comúnmente, la capacidad del sistema de manejar la concurrencia se puede incrementar añadiendo recursos a la red.

- Compartición transparente de recursos.
  La distancia geográfica entre los nodos que integran el sistema y la comuni-

cación entre los mismos no es una limitante para que un usuario ubicado en un extremo geográfico del sistema pueda utilizar recursos de un procesador ubicado a varios kilómetros de distancia.

- Ausencia de reloj global.
  Un sistema distribuido tiene componentes independientes que cuentan cada uno con su respectivo reloj. Sin embargo, no existe un reloj global que controle a todos. Cuando los programas necesitan cooperar, deben coordinar sus acciones mediante el intercambio de mensajes. La mayoría de las veces, dicha coordinación depende del orden en el que ocurren los eventos, o en su defecto, la dependencia de datos para iniciar una nueva ejecución.

- Independencia de fallas.
  Existen muchos factores que pueden ocasionar el surgimiento de fallas: pueden aparecer tanto en la red de comunicación, como en el funcionamiento de algún equipo en particular. En el diseño del sistema, se hacen consideraciones para que en caso de alguna falla, se aísle el nodo o nodos en cuestión, sin que esto signifique que se detengan el sistema en su conjunto o que deje de funcionar.

- Escalabilidad.
  Es la capacidad del sistema para crecer de manera transparente, y que al añadir elementos (nodos y sus recursos) se mantenga la comunicación, el orden de los eventos, la transparencia y demás aspectos de integración entre los componentes.

Clasificación

No existe una manera formal o única de clasificar la amplia variedad de sistemas distribuidos, ya que por su naturaleza pueden ser muy diversos en aspectos como la topología de la red, la distancia geográfica entre nodos, la cantidad de los mismos, etc. Sin embargo, una manera de agruparlos para su estudio se establece de acuerdo a las características de los nodos. Si los procesadores del sistema son iguales, se trata de un sistema **homogéneo**, mientras que si las características de estos son diferentes se trata de un sistema **heterogéneo** [4].

2.1.3. Desafíos de los Sistemas Distribuidos

El cumplimiento de las características descritas anteriormente no es una tarea trivial, y en el cumplimiento de las mismas, surgen los desafíos inherentes a los

sistemas distribuidos. Es por ello que van de la mano con las características que se establecen al diseñarlos, construirlos y mantenerlos en operación correcta.

Los principales desafíos en los sistemas distribuidos son los siguientes [15]:

- Heterogeneidad
  Los nodos que integran un sistema distribuido pueden ser muy diferentes en aspectos como la arquitectura de hardware, la capacidad de memoria, la velocidad de procesamiento, etc. Esto contribuye a convertirse en un desafío puesto que la manera de programar, de organizar la memoria, o la capacidad de procesar un mayor número de tareas pueden variar.

- Apertura
  Se determina primordialmente por el grado en el cual nuevos servicios pueden ser añadidos y estar disponibles para usar por los programas clientes.

- Seguridad
  Se debe proteger, tanto física como lógicamente, la información que se encuentra en el sistema sin que esto cause conflictos en la compartición o transparencia adecuada de ésta.

- Escalabilidad
  El sistema debe ser capaz de crecer, pudiendo añadir nuevos componentes, sin perder ninguna de sus otras funcionalidades

- Tolerancia a fallas
  El sistema debe ser capaz de detectar fallos en uno o mas de sus componentes, adaptarse a estos fallos para continuar funcionando, o en el mejor de los casos, resolver tales fallas de manera transparente.

- Desempeño
  Un sistema distribuido debe ser capaz de ofrecer un mejor desempeño que un sistema centralizado expresado en la disminución del tiempo de ejecución de las tareas. Una forma de mejorarlo es verificando que el aprovechamiento de los nodos en conjunto se maximice tanto como sea posible evitando tener ciertos elementos sobrecargados y otros en estado ocioso. [1]

[1] EL término en inglés para éste estado en los procesadores es *idle*, no existe una traducción única para el término pero para efectos de este trabajo se utiliza "ocioso"

Estos desafíos han sido objeto de estudio en las últimas décadas. Producto de dicho estudio, han surgido técnicas o estrategias que buscan solucionar eficientemente los problemas que aparecen en ellos

Es precisamente en el desempeño donde se centra el presente trabajo, puesto que con el avance de la tecnología, el surgimiento de nuevas herramientas y equipo, el aprovechamiento de los recursos se mantiene como un reto fundamental en el estudio de este tipo de sistemas. Diversas soluciones se han propuesto para mejorar el desempeño, siendo una de las más relevantes la técnica de balance de carga.

2.2. Balance de Carga

La carga de trabajo de un procesador es la cantidad de tiempo que éste necesita para ejecutar todos los procesos que le son asignados. Debido a la naturaleza de un sistema distribuido, la llegada de nuevas tareas a un nodo puede realizarse de diferentes formas, y por consiguiente, es muy probable que no sean asignadas uniformemente de acuerdo a la cantidad de nodos disponibles. Así, es factible incrementar el desempeño general, implementando mecanismos para que los nodos más cargados puedan repartir algunas de sus tareas a los nodos con carga más ligera, con el objetivo de completar las tareas con mayor rapidez. [14].

El balance de carga es el proceso de distribuir recursos computacionales y la cantidad de trabajo de manera transparente en el sistema. Para un sistema distribuido, el balance de carga se considera eficiente cuando a través de su ejecución, se reflejan en el sistema características como un alto desempeño, alta disponibilidad y escalabilidad [11].

Un esquema de balance de carga proporciona reglas para determinar si una tarea debe ser ejecutada localmente o por un nodo remoto. No obstante, antes de iniciar una estrategia de balance de carga, se debe considerar la suma del tiempo de transferencia de una tarea y el tiempo de ejecución en otro nodo. Si el tiempo estimado es mayor que en el procesador original, resulta inconveniente realizar el proceso de balance, ya que ésto solo degradaría el desempeño del sistema en su totalidad [6].

El tema del balance de carga puede ser estudiado en diferentes ambientes de cómputo, usando diversas estrategias y siendo aplicado en diferentes niveles de profundidad. Los algoritmos pueden requerir o no información acerca de las tareas individuales, o pueden tomar decisiones basados en la situación actual de carga. La

transferencia de una tarea puede ser iniciada por el emisor o por el receptor de la misma. La unidad de ejecución que es transferida o redistribuida puede variar desde tareas completas, generadas por usuarios, procesos individuales o incluso pequeños módulos de programas. También pueden ser ciertos componentes de cómputo con requerimientos de comunicación específica. Finalmente, la transferencia de una tarea puede estar restringida a ser realizada antes de inicio de su ejecución o puede ser realizada en tiempo de ejecución de la misma [1].

Sin embargo, es importante considerar que la ejecución de algoritmos cuyo objetivo el balance de carga implica un costo de recursos. En su elaboración se debe tener en consideración de manera importante, una serie de reglas y criterios para estimar los niveles de carga del sistema. El uso de los diferentes enlaces entre los nodos, y otros aspectos. No en todos los esquemas de ejecución de tareas distribuidas, es factible realizar la redistribución de la carga, debido a que en ciertos casos, al realizar el proceso, el exceso en el uso de los canales de comunicaciones (puertos, canales, protocolos, tipos de mensaje) afecta negativamente en el desempeño, generando un problema más grande que el original que se pretendía resolver.

Un gran número de estrategias para el balance de carga han sido propuestas. A grandes rasgos, los diferentes algoritmos que se han propuesto se pueden clasificar en dos categorías: estáticos y dinámicos

2.2.1. Balance de Carga Estático

En este tipo de algoritmos, las asignaciones de las tareas por ejecutar en los nodos se realiza *a priori*, tomando en consideración la información de las tareas, (tiempo de llegada, tiempo promedio de ejecución, cantidad de recursos necesarios, y requerimientos de comunicación entre ellos, si fuera el caso) para realizar un estimado del tiempo que transcurre en su ejecución. Sin embargo, no siempre es posible conocer toda esta información o hacerlo de manera precisa, por lo que se puede volver más complejo cuando se trata de sistemas heterogéneos y la capacidad de los nodos varía de uno a otro [1].

La asignación para la ejecución es realizada normalmente por un procesador principal o maestro, basado en determinadas reglas definidas previamente. Los otros nodos del sistema ejecutan las tareas, y al finalizar los resultados pueden ser enviados o no al procesador principal dependiendo del tipo de tarea que se trate.

Las estrategias estáticas toman decisiones considerando valores promedio estimados de tiempo de ejecución de procesos y retrasos en la comunicación durante el

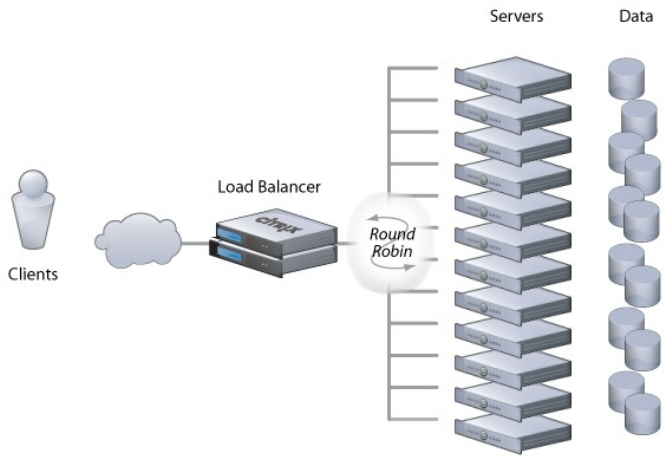

Figura 2.2: En un algoritmo de tipo round robin el nodo repartidor, asigna las tareas de manera equitativa entre el resto de los elementos

tiempo de compilación[14].

Algoritmos estáticos convencionales

Algunos de los algoritmos más utilizados en el balance de carga estático son los siguientes [13]:

- **Round Robin**

  En este algoritmo, las tareas se asignan de manera equitativa a todos los nodos encargados de la ejecución. Esto significa que la elección de procesador se realiza de forma serial y una vez que se asigna una tarea a cada nodo, se regresa al primero y se vuelve a realizar la asignación en el mismo orden. Lo anterior se hace de manera similar a un juego de naipes, donde se reparten de manera equitativa a cada uno de los participantes siguiendo el mismo orden preestablecido (Figura 2.2).

- **Asignación aleatoria**

  El procesador encargado de la asignación genera una serie de números aleatorios en las tareas, los cuales utiliza para asignarlas a los respectivos nodos. Dichos números se generan entre otras formas por distribuciones estadísticas (Figura 2.3).

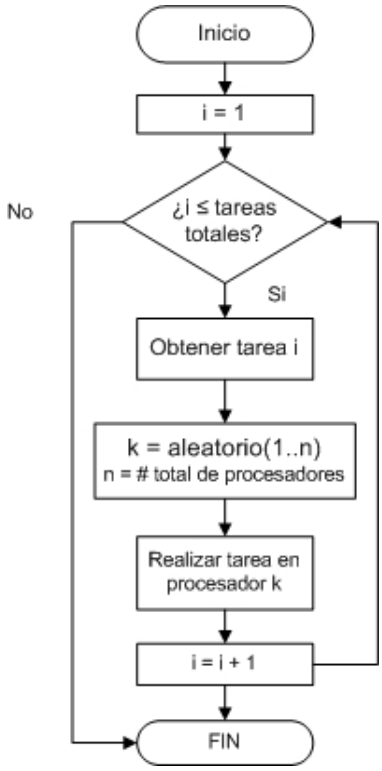

Figura 2.3: Algoritmo de asignación aleatoria
[13]

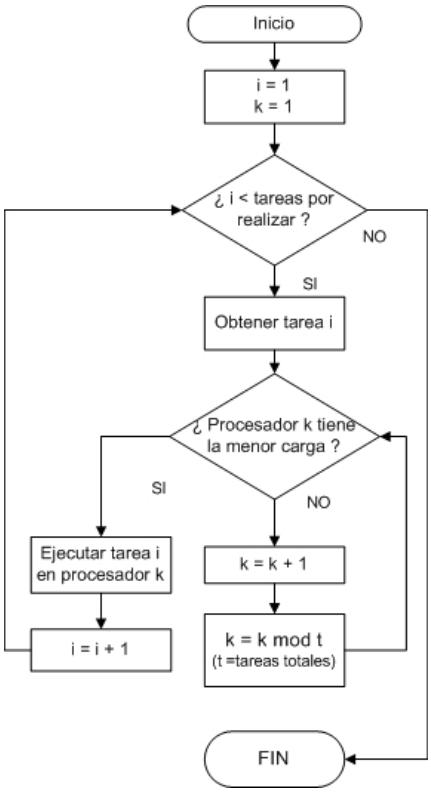

Figura 2.4: Algoritmo de Administrador Central
[13]

- **Administrador Central**

  En cada paso de la ejecución de este algoritmo, el criterio de selección para asignar tareas se basa en decidir cuál de los nodos tiene la carga de trabajo más baja. Por ello el nodo central debe de reunir la información de la carga de los otros nodos (Figura 2.4). Este algoritmo tiene características similares a "Round Robin.[en] cuanto a la asignación de las tareas. La principal diferencia es que en el caso de "Round Robin"la asignación es equitativa mientras que en este algoritmo, si se asigna una tarea considerada como pesada a un determinado nodo, en la siguiente ronda de asignación no se considera dicho nodo.

- **Umbral**

  Previamente a la ejecución del algoritmo, se define una constante que se conoce como umbral de carga, la cual es parámetro para medir la carga de trabajo de

los nodos.

La elección del nodo al cual se asigna la tarea o conjunto de tareas a ejecutar se realiza tomando en cuenta dos valores para dicho umbral. El primero de ellos es para el límite superior de carga. Una vez alcanzado éste, ya no se le asignan tareas. El otro valor de umbral es asignado para el límite inferior de carga.

Basado en la estimación del tiempo de ejecución de las tareas asignadas en una primera iteración, al respectivo nodo se le asignan nuevas tareas cuando se calcule que el nodo termine de ejecutarlas, y su nivel de carga llega a este limite inferior.

Los valores de umbral son utilizados para caracterizar estados de un procesador esclavo. (Figura 2.5)

2.2.2. Balance de Carga Dinámico

Por naturaleza, este tipo de algoritmos son más flexibles que los de tipo estático, ya que las asignaciones de tareas pueden tener muchas variaciones de acuerdo a las características que se tomen en cuenta para elaborarlos. Son adaptables a las condiciones cambiantes del sistema. Si en un momento dado una tarea toma mucho tiempo en ejecutarse, el proceso de balance se adapta de tal manera que ya no asigna tareas a ese nodo, e inclusive puede, reubicar las otras tareas que se encuentren en la cola de ejecución [14].

La carga de los nodos se monitorea continuamente, y cuando se alcanza un nivel predefinido de disparidad, se procede con el proceso de redistribución de carga. Al realizar el procedimiento, se utiliza tiempo de procesamiento y recursos adicionales, los cuales se deben estimar de alguna forma antes de efectuar la ejecución. Inevitablemente, la redistribución incurre en una sobrecarga *(overhead)* en la ejecución del proceso por sí mismo y en el tráfico de la red. La clave es que dicha sobrecarga no afecte el sistema en manera mayor que si la redistribución no se realiza [1].

Un algoritmo de balance de carga dinámico utiliza información de los estados del sistema lo más actualizada posible en ese momento, y busca tomar mejores decisiones en la redistribución de las tareas en el sistema. Las estrategias dinámicas son adaptables a situaciones cambiantes, puesto que las decisiones de envío a otro procesador, de ubicación de nuevas tareas o de manejo de tiempos de espera se pueden tomar en tiempo de ejecución [9].

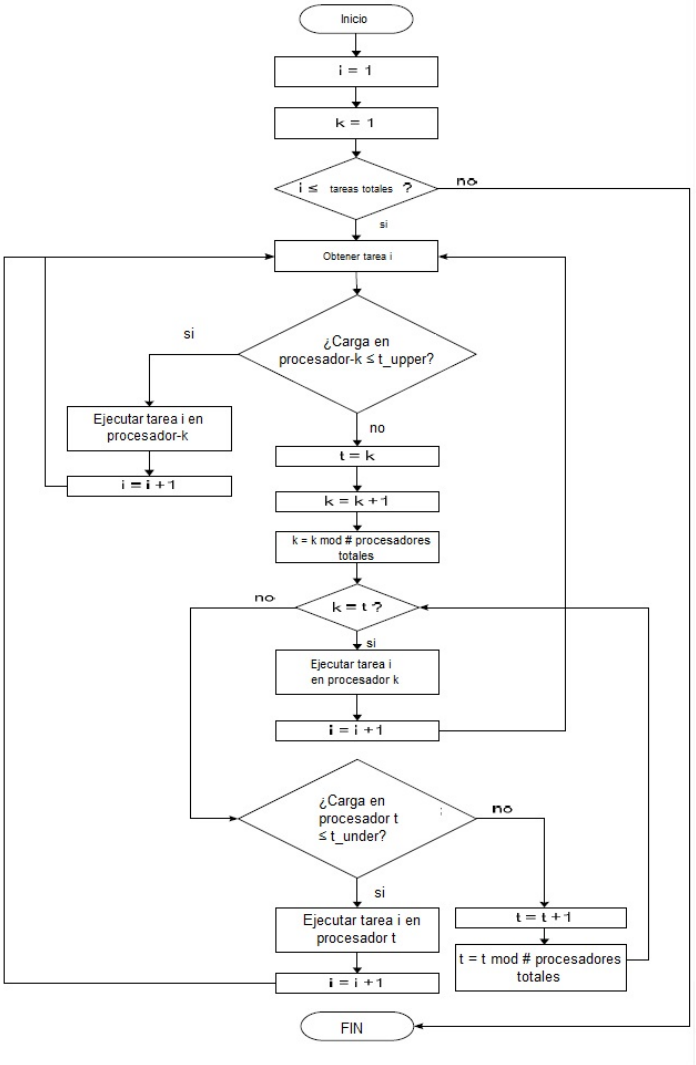

Figura 2.5: Algoritmo de Distribución por Umbral [13]

Debido a la amplia variedad de soluciones que consideran estrategias dinámicas, es difícil realizar una clasificación donde se engloben las particularidades de cada algoritmo. Sin embargo, una gran parte de éstos comparten determinados componentes esenciales.

Ventajas

Las principales ventajas que ofrece un algoritmo de balance de carga dinámico son las siguientes [9]:

- El desempeño global del sistema evaluado en la cantidad de tareas ejecutadas suele ser mayor que el de un algoritmo estático.
- Reducción en el tiempo de respuesta promedio.
- Mayor utilización de los recursos.
- Adaptabilidad a las fluctuaciones de carga del sistema.

Desventajas

En una buena parte de los algoritmos dinámicos aparecen las siguientes desventajas:

- Mayor dificultad en la implementación.
- Si no se selecciona una política adecuada, se genera una sobrecarga considerable en las comunicaciones del sistema.
- Se requiere siempre información actualizada del estado de los nodos en el sistema.
- El algoritmo en sí mismo provoca una sobrecarga en la red lo cual en un momento dado puede generar congestión de la misma.

Aspectos a considerar en el Balance de carga dinámico

El diseño de algoritmos de balance de carga dinámico se construye siguiendo diferentes políticas que son las que establecen las características particulares del algoritmo Las políticas generales de los algoritmos de balance de carga dinámico son las siguientes [11]:

- **La política de información**

  En esta política se define la manera de recolectar la información de los estados del sistema. En cuanto a la información local, se reunen los datos de los nodos vecinos, mientras que para recabar estadísticas globales, se trata de datos de cada uno de los nodos pertenecientes al sistema. Esto con el propósito de tener una mayor cantidad de información y poder tomar decisiones más acertadas en la ubicación de tareas.

  Los factores que se toman en cuenta al diseñar una política de información adecuada son, entre otros: las estimaciones de la carga del procesador, la caducidad de la información (principalmente ocasionada por el factor de latencia de comunicación en la red), y la frecuencia de intercambio de información.

  Algunos ejemplos de Políticas de Información son los siguientes [9]:

  - **Periódica:** Los nodos envían mensajes de manera periódica con la información de sus estados. Este esquema es adecuado para sistemas donde la información se transmite hacia un nodo central o es intercambiada entre los dispositivos.
  - **Bajo Demanda:** Cada nodo recolecta la información del estado del sistema, consultando los otros nodos solamente cuando se alcanza un estado de sobrecarga (en cuyo caso se convierte en emisor) o cuando tiene carga muy baja (se cataloga como nodo receptor). Existen tres enfoques en esta política: iniciada por el emisor, iniciada por el receptor, o iniciada simétricamente.
  - **Por cambios de Estado:** Los nodos diseminan información del estadoo en el que se encuentran, siempre que las condiciones del nodo varian en cierto grado. Los dos enfoques de esta política son: centralizado, y descentralizado. A diferencia de la política periódica, solo se transmite información cuando hay un cambio.
  - **Petición-Respuesta:** El intercambio de información de los estados se hace mediante retroalimentación entre nodos. Se ejecuta solo cuando un nodo lo requiere y se minimiza la sobrecarga de recolectar dicha información.
  - **Por Eventos:** Esta aproximación es similar a la política por cambios de estado. La diferencia es que, el intercambio de información se realiza de

manera asíncrona, ocurriendo solo cuando ciertas condiciones se cumplen en un conjunto de nodos.

- **Variabilidad de Índice:** En esta política, en el sistema se define una constante conocida como índice de carga. Si el índice de uno o varios nodos en el sistema está cambiando frecuentemente, el nodo informa a los demás en el sistema, mientras que si la carga de trabajo cambia poco, el intercambio no es necesario.

- **La política de transferencia**

  Después de que la información de la carga se intercambia, un nodo central o cada nodo posee la información de uno o varios nodos del sistema. Utilizando dicha información la política de transferencia puede decidir el grado de carga de un nodo. Normalmente se ordenan de ligeramente cargado a altamente cargado. Con base en este grado se seleccionan los nodos que son susceptibles de realizar una transferencia de carga.

  Ejemplos de Políticas de Transferencia son:

  - **Umbral:** Este esquema funciona de manera similar al algoritmo estático de distribución por umbral. Los umbrales son expresados en unidades de carga y la política define a un nodo ligeramente cargado como receptor, y a un nodo altamente cargado como emisor.

    Estos algoritmos, por lo general, tienen menor comunicación entre los procesos y un mayor número de ubicaciones locales, lo que reduce la sobrecarga de accesos a memoria remotos.

  - **Disparidad en la carga:** Inicia la transferencia de procesos solamente cuando, de acuerdo a la política de intercambio de información, se detecta un desequilibrio en la carga entre los nodos, que normalmente se define con una constante de disparidad en la carga.

- **La política de selección**

  En la política de selección, se eligen una o varias tareas para ser transferidas a otro nodo. Un criterio básico para que un proceso, o un conjunto de tareas sean seleccionadas para ser transferidas, es que la sobrecarga que se va a generar con la transferencia del proceso debe ser compensada con la reducción en el tiempo de respuesta.

Algunos de los factores que se consideran en la selección de procesos/tareas para ser transferidos son los siguientes:

- El número de llamadas al sistema dependientes de ubicación debe ser mínimo.
- La tarea seleccionada debe ser lo suficientemente larga, de modo que valga la pena incurrir en la sobrecarga de ser transferida.
- Un proceso solo debe ser seleccionado para ser transferido si su tiempo de respuesta se puede reducir después de la transferencia.

- **La política de ubicación**

  La política de ubicación es responsable de seleccionar el nodo más apto entre los que se encuentren disponibles en el sistema para ejecutar una tarea. El nodo seleccionado debe tener un ambiente adecuado en cuanto a recursos disponibles y nivel de carga para ejecutar la tarea.

  Alguno ejemplos de políticas de ubicación son los siguientes:

  - Aleatorio

    El nodo para ejecución es seleccionado aleatoriamente, y no hay intercambio de información que ayude en la toma de decisión. Un problema de esta aproximación es que pueden ocurrir transferencias sin sentido ya que el nodo seleccionado puede estar altamente cargado y requiera reenviar la tarea, que a su vez vuelva a llegar a un nodo altamente cargado, ocasionando en los peores casos que las tareas solo se estén transfiriendo entre nodos y nunca se ejecuten.
  - Polling (Sondeo)

    Un nodo sondea a uno o varios de los otros nodos del sistema para encontrar alguno que sea adecuado para transferir una o un conjunto de tareas. El nodo al cual se hace el sondeo puede ser seleccionado de diferentes formas, las más utilizadas son: selección aleatoria, basado en la información de previos sondeos, o por el criterio del vecino más cercano.

2.2.3. Aspectos cualitativos de los algoritmos de balance de carga

Los algoritmos de balance de carga son muy variados debido a su naturaleza. Es por ello que no hay un método formal o único para elaborarlos. Sin embargo, en su

construcción se consideran los siguientes aspectos, los cuales también proporcionan elementos para la evaluación del desempeño y rendimiento del sistema. [1]

- **Naturaleza del algoritmo.** Lo primero que se debe definir es si el algoritmo será estático o dinámico.
- **Sobrecarga (Overhead)** Se refiere a la influencia del algoritmo en el sistema, consecuencia de la reubicación de las tareas, la comunicación entre procesos, nodos, grupos, tareas etc.
- **Utilización de recursos.** Se trata de la cantidad de recursos empleados por cada uno de los nodos involucrados en el sistema.
- **Desperdicio de procesador.** Es medido cuando los procesadores emplean más recursos en enviar una tarea para ejecución remota, que en ejecutarla localmente.
- **Predecibilidad** La característica referente a la capacidad de determinar si el algoritmo es o no determinístico en su comportamiento.
- **Adaptabilidad** Se refiere a la capacidad del algoritmo para cambiar su forma de ejecución bajo ciertas circunstancias de sobrecarga en el procesamiento.
- **Confiabilidad** Éste término se refiere a la capacidad del algoritmo para continuar su ejecución en caso de que exista una o varias fallas en el sistema. Además de la recuperación a tales fallas.
- **Tiempo de respuesta** Es la cantidad de tiempo que lleva en la ejecución del algoritmo además de la realización del balance de carga.
- **Estabilidad** Se refiere a que el intercambio de la información relacionada con los estados de los nodos y la carga de los mismos se mantengan de manera consistente en su ejecución.

La eficiencia de los algoritmos dinámicos de balance de carga depende de la selección de los criterios para estimar la carga. En general, tales métodos muestran mayor desempeño, velocidad de ejecución y mayor escalabilidad que los métodos estáticos. [5, 8, 14].

Sin embargo, esta mejora en dichas características implica un costo, reflejado en los siguientes aspectos:

- La dificultad inherente en el diseño e implementación de los algoritmos.
- El incremento en la sobrecarga de comunicaciones al recolectar los datos del sistema.
- el incremento de tráfico en la red, el cual es un factor importante cuando se transmite la carga de un nodo a otro.

Tales aspectos pueden afectar en el desempeño general del sistema. Uno de los pprincipales objetivos de un algoritmo de balance de carga es reducirlos lo más posible.

Diversas técnicas se han propuesto en la literatura para alcanzar un mejor desempeño en los algoritmos de balance de carga. Entre ellas, destacan las estrategias que utilizan elementos de lógica difusa en la construcción de sus reglas.

El uso de la lógica difusa se introduce para lidiar con la incertidumbre que existe de manera inherente en los algoritmos dinámicos. Se utiliza como una herramienta desde el punto de vista de Inteligencia Artificial, como esquema para la toma de decisiones en la ubicación de tareas y en la evaluación de la carga en cada nodo, con el fin de establecer un destino adecuado para el envío y recepción de tareas.

2.3. Conceptos básicos de lógica difusa

En esta sección se presenta la introducción a los conceptos de lógica difusa, que se utiliza en un sistema distribuido en la propuesta para lograr el balance de carga.

2.3.1. Introducción

La lógica difusa surge como alternativa a la lógica clásica. El objetivo principal que persigue es introducir el manejo de ambigüedad en las variables y categorías que se evalúan.

En la lógica clásica se busca establecer una dicotomía en cualquier modelo matemático de un concepto dado. Tradicionalmente cuando se analiza un determinado objeto, sin importar qué tan complejo sea, es preciso asignarlo entre dos categorías predefinidas. (por ejemplo: bueno-malo, blanco-negro, lejano-cercano, etc.)

Tales clasificaciones son adecuadas en diversos tipos de análisis. Por ejemplo, en el estudio de números enteros se pueden definir *pares* e *impares* como dos categorías establecidas, y cualquier número puede ser fácilmente clasificado en alguna de estas dos. [12]

En otros ámbitos, especialmente en aquellos relacionados a la percepción que tenemos los humanos, lo anterior no es muy preciso. Algunos ejemplos pueden ser: clasificar la temperatura como *confortable*, designar a una persona como de estatura *baja*, definir una distancia como *lejana* etc.

La lógica difusa es propuesta por primera vez por Lofti Zadeh en 1965 [17]. Se basa en la idea de que en muchos casos del mundo físico real, las clases de objetos que se encuentran no tienen criterios de pertenencia definidos de manera rígida. Postula los conjuntos difuso, como clases de objetos con grados de pertenencia contínuos.

Con el paso del tiempo, la lógica difusa ha evolucionado como una herramienta en el desarrollo de la tecnología. Algunas de las principales áreas del conocimiento donde la lógica difusa juega un rol son las siguientes: [16]

- Modelado matemático
- Lenguaje natural
- Minería de datos y aplicaciones de ingeniería
- Bioinformática
- Sistemas de control
- Reconocimiento de patrones
- Geología
- Administración e investigación de operaciones

En el área de la computación. La lógica difusa es una herramienta que permite realizar diferentes procesos para tomar decisiones de una manera más parecida a los humanos, es decir, en una forma ambigua: lidiar con incertidumbre, imprecisión o problemas cuya resolución requiere toma de decisiónes basada en características cualitativas.

2.3.2. Teoría de conjuntos difusos

La cuestión fundamental de los conjuntos difusos es que permiten extender el significado del concepto de *conjunto*, admitiendo diferentes grados de pertenencia. Entre más alto sea el valor de pertenencia de cierto objeto x a un conjunto difuso, digamos $A(x)$, más fuerte es el vínculo de x a la categoría descrita por A [12].

Conjuntos Clásicos

La teoría de conjuntos clásicos es sumamente extensa y ha sido estudiada ampliamente en diversos campos de conocimiento. A continuación se presentan algunos elementos fundamentales, con el fin de establecer la base para la descripción de los conjuntos difusos.

Un conjunto es una colección de objetos que puede ser representado de tres maneras diferentes [7]:

- Listando todos sus elementos. Esta forma solo se puede utilizar cuando los conjuntos son finitos.
$$A = \{a_1, a_2, ..., a_n\} \tag{2.1}$$

- Mediante una propiedad que cumplen los elementos pertenecientes al conjunto.
$$A = \{x|P(x)\} \tag{2.2}$$

- Mediante una función, normalmente llamada *función característica*, que declara qué elementos de X pertenecen al conjunto y cuáles no.
Ejemplo: El conjunto A se define por su función característica X_A como sigue:
$$X_A = \begin{cases} 1 & \text{for } x \in A \\ 0 & \text{for } x \notin A \end{cases} \tag{2.3}$$
Esto es, la función característica mapea elementos de X a elementos del conjunto [0,1] que es formalmente expresado por
$$X_A : X \to \{0, 1\} \tag{2.4}$$
Para cada $x \epsilon X$, cuando $X_A(x) = 1, x$ pertenece a A; y cuando $X_A(x) = 0$, entonces x no pertenece a A

De modo que en los conjuntos tradicionales, los elementos del universo del discurso pertenecen o no de manera definitiva.

Conjuntos Difusos

Un conjunto difuso se define de la siguiente manera [17]:

Sea X el espacio de objetos, con un elemento genérico de X denotado por x. Entonces, $X = \{x\}$. Un *conjunto difuso* A es caracterizado por una *función de*

pertenencia $f_A(x)$ que asocia para cada objeto en X un número real en el intervalo [0,1] con el valor de $f_A(x)$ en x representando el "grado de pertenencia"de x en A. Entonces entre más cercano a la unidad se encuentre el valor $f_A(X)$, más alto es el grado de pertenencia de x en A.

Para representar una función de pertenencia que expresa un conjunto difuso se emplea la siguiente notación:

$$\mu_A : X \to [0, 1] \tag{2.5}$$

En relación a dicha función se puede señalar lo siguiente:

- Si $\mu_A = 0$ implica que tal elemento definitivamente no es un elemento del conjunto difuso.
- Si la función de pertenencia cumple con el valor de la unidad ($\mu_A = 1$) significa que el elemento pertenece completamente al conjunto.
- Un grado de pertenencia intermedio corresponde a la pertenencia difusa del conjunto.

Los primeros dos casos se trata de la misma situación que en los conjuntos clásicos , mientras que con el tercer caso, se expande éste concepto tanto como se desee introducior el grado de ambiguedad. El significado de predicado fundamental en teoría de conjuntos '$\in$'(que pertenece) es expandido significativamente aceptando una pertenencia parcial en un conjunto.

Otra forma de representar conjuntos difusos en un universo del discurso X es mediante listas de información estructurada acerca de grados de pertenencia en puntos individuales de X. Por ejemplo una lista compuesta de sublistas de dos elementos:

$$[[x_1, 1.0], [x_2, 0.8], [x_3, 0.5], [x_4, 0.0]] \tag{2.6}$$

Que denota un conjunto difuso A con esos grados de pertenencia:

$$A(x_1) = 1.0, A(x_2) = 0.8, A(x_3) = 0.5, A(x_4) = 0.0 \tag{2.7}$$

2.3.3. Ejemplos de algunas funciones de pertenencia

Aún cuando en principio cualquier función puede ser válida para definir un conjunto difuso, existen ciertas funciones que son más comúnmente utilizadas por su

simplicidad de representación y facilidad al implementar. Entre ellas se encuentran las funciones de tipo triangular, trapezoidal o gaussiana. [14]

$$\mu(x)=\begin{cases}0 & \text{para } x\le a\\ \dfrac{x-a}{m-a} & \text{para } a<x\le m\\ \dfrac{b-x}{b-m} & \text{para } m<x\le b\\ 0 & \text{para } x>b\end{cases}$$

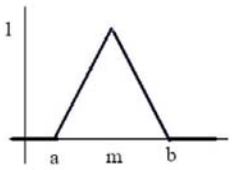

Figura 2.6: Ejemplo de función triangular

$$\mu(x)=\begin{cases}0 & \text{para } x\le a\\ \dfrac{x-a}{b-a} & \text{para } a<x\le b\\ 1 & \text{para } b<x\le c\\ \dfrac{d-x}{b-c} & \text{para } c<x\le d\\ 0 & \text{para } x>d\end{cases}$$

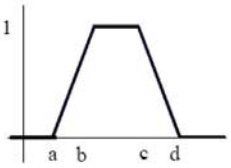

Figura 2.7: Ejemplo de función trapezoidal

$$\mu(x)=\begin{cases}0 & \text{para } x\le a\\ 2\left\{\dfrac{(x-a)}{(b-a)}\right\}^2 & \text{para } x\in (a,m]\\ 1-2\left\{\dfrac{(x-b)}{(b-a)}\right\}^2 & \text{para } x\in (m,b)\\ 1 & \text{para } x\ge b\end{cases}$$

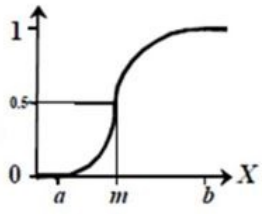

Figura 2.8: Ejemplo de función S

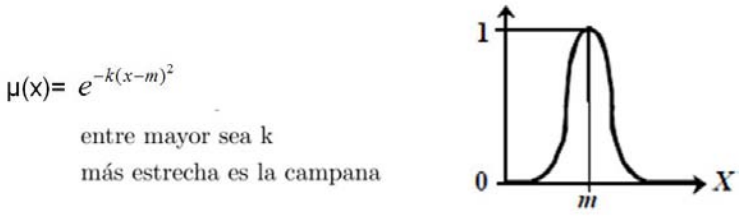

Figura 2.9: Ejemplo de función gaussiana

2.3.4. Módulo Difuso

La lógica difusa se aplica en sistemas que utilizan expresiones ambiguas, el objetivo es formar reglas que definen la forma en que se comporta un sistema en su totalidad.

De manera general, cuando se aplica la lógica difusa en algún campo de conocimiento, se hace a través de un módulo, en el que se establece cada una de las partes que lo componen. Mediante su implementación, se describe un conjunto de reglas, buscando realizar el procedimiento de abstracción que una persona utilizaría para la toma de decisiones para un conjunto de acciones, y que el conjunto de estas acciones puedan conducir a la resolución de un problema, generando diferentes resultados donde sea aplicado [5].

El principal propósito de la aplicación de la lógica difusa desde el punto de vista de inteligencia artificial, es el de involucrar a la *intuición* como elemento participante en la toma de decisiones. Esto se realiza mediante la inclusión de reglas de tipo *if-then* y la inferencia de conclusiones para la toma de decisiones a partir de dichas reglas. Una regla difusa *if-then* es un esquema que representa un cierto conocimiento que es inexacto e impreciso naturalmente. La parte *if* de la regla difusa se conoce como el antecedente de la regla, la parte *then* es el consecuente [14]

En la utilización de las reglas difusas de tipo *if-then* se busca que de acuerdo al comportamiento del sistema, en un momento dado exista una coincidencia con una de las reglas en los datos de entrada, y en consecuencia, hacer una inferencia. Además, el sistema combina las conclusiones inferidas de todas las reglas para formar una conclusión general

En la literatura se encuentran diversos tipos de módulos difusos para la implementacion de las reglas y la toma de desiciones, algunos de los más utilizados y considerados como entre lós más relevantes son los siguientes [12]:

- **Sistemas difusos puros**

  La configuración básica de un sistema difuso puro consiste solamente en dos componentes: la base de conocimientos, que representa la colección de reglas difusas de la forma IF-THEN; y el módulo de inferencia el cual combina dichas reglas en un mapeo de conjuntos difusos en el espacio de entrada R^n a un conjunto difuso en el espacio R basado en los principios de lógica difusa.

  El principal problema con este tipo de sistemas, es que las entradas y salidas son conjuntos difusos, en tanto que en sistemas reales, las entradas y salidas son variables de valores reales, por lo que no siempre se puede aplicar. (Ver Figura2.10).

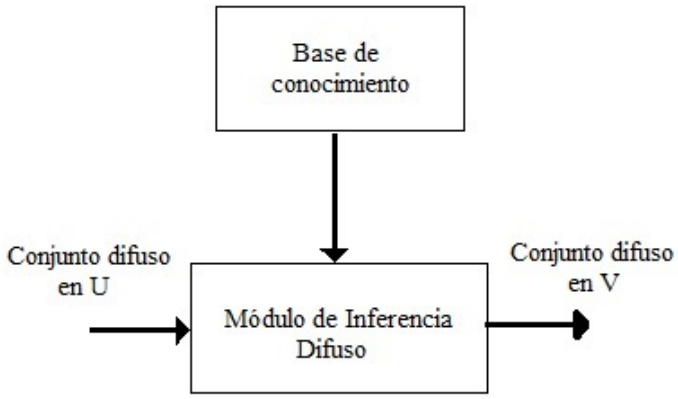

Figura 2.10: Módulo de sistema difuso puro

- **Takagi-Sugeno-Kang**

  Para resolver el problema del modelo anterior, Takagi, Sugeno y Kang propusieron otro tipo de sistema difuso cuyas entras y salidas son variables de valores reales. (Ver Figura 2.11)

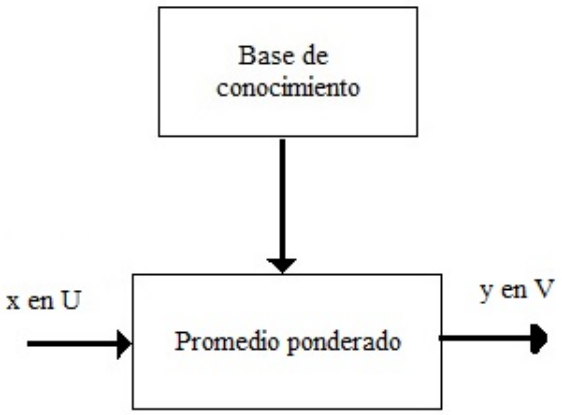

Figura 2.11: Módulo de sistema difuso Takagi-Sugeno-Kang

Las reglas difusas en este tipo de sistemas son de la forma siguiente:

IF la velocidad x en un auto es alta,
THEN la fuerza del acelerador es $y = cx$

El cambio en una regla de este tipo con respecto a una regla básica IF-THEN radica en el uso de una fórmula matemática. Este cambio en las reglas permite combinarlas de manera más simple. Ésta forma es un promedio ponderado de los valores en las partes THEN de las reglas.

Los principales problemas con éste tipo de sistemas radican en que la parte "THEN"de la regla de inferencia al ser una fórmula matemática puede no proveer un marco de trabajo para representar conocimiento humano; as'mismo, no hay mucha "libertad"para aplicar los principios difusos y la versatilidad de los mismos no siempre es representada completamente.

- **Sistemas difusos con bloques complementarios**

  Surge como mejora al modelo Takagi-Sugeno-Kang. Éste tipo de sistema integra lo mejor de los modelos anteriores superando las desventajas encontradas en ellos [**?**].

  La estructura se compone de cuatro etapas: una base de conocimiento, de una etapa que transforma la entrada nítida en difusa, un mecanismo de inferencia y una sección que transforma la salida difusa en nítida (figura 2.12).

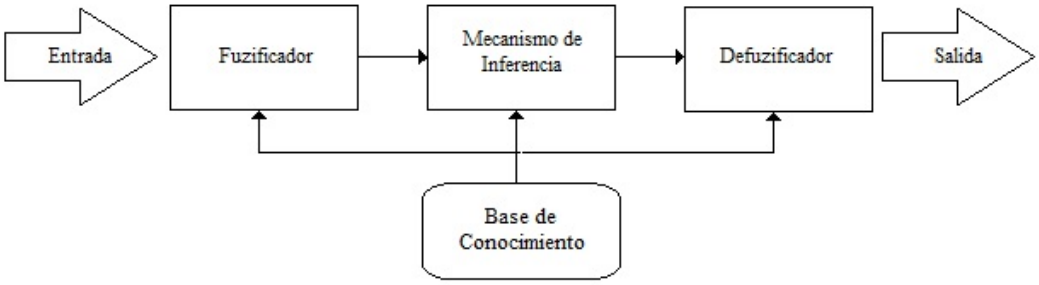

Figura 2.12: Módulo difuso completo
[5][7][14]

En este trabajo se utiliza el módulo con bloques complementarios debido a que con la integración de éstas etapas, es posible ir del universo nítido al difuso y viceversa, lo cual proporciona elementos auxiliares en las decisiones de balance de carga.

- **Entrada**. Es un valor nítido congruente con el respectivo sistema y que va a ser procesado por el módulo difuso.
- **Bloque para convertir la entrada**. Este bloque descompone el valor de la entrada en uno o más conjuntos difusos. Tiene como objetivo hacer la conversión o asociación de esos valores nítidos (También conocidos como valores *crisp*) en difusos.

  En este bloque se asignan grados de pertenencia a cada una de las variables de entrada, con relación a los conjuntos difusos previamente definidos, utilizando las funciones de pertenencia asociadas a dichos conjuntos difusos.

  El proceso permite a las entradas del sistema ser expresadas en términos lingüísticos tales que las reglas pueden ser aplicadas de forma simple para expresar un sistema complejo.
- **Base de conocimiento**. Contiene las reglas asociadas con el dominio de la aplicación y los objetivos que persigue el módulo. En esta etapa se definen las reglas lingüisticas que se encargarán de tomar las decisiones mediante las cuales debe actuar el sistema. Por lo general, las reglas que definen la base de conocimiento difusa son del tipo *if-then* y generan la clasificación para los elementos del sistema.
- **Mecanismo de inferencia**. Relaciona los conjuntos difusos de entrada

y salida para representar las reglas que definen el sistema. En la inferencia se utiliza la información de la base de conocimiento para generar reglas mediante el uso de condiciones.

- **Bloque para convertir la salida**. El proceso inverso a la entrada consiste en adecuar los valores difusos generados en la inferencia en valores nítidos, que posteriormente se utilizan en el proceso de la toma de decisión de un valor para el respectivo análisis. Para regresar a estos valores, se utilizan diversos métodos, desde matemáticos aplicados a las funciones de pertenencia; así como otros métodos más simples que consideran la fuerza de las reglas aplicadas.
- **Salida**. Finalmente la salida será otro valor concreto de acuerdo a la clasificación previamente establecida en el módulo difuso y que va a permitir ser procesado de manera más precisa.

2.4. Resumen

Los sistemas distribuidos se componen de una serie de computadoras interconectadas mediante una red. Comparten recursos de manera transparente para el usuario final y se comunican mediante el paso de mensajes.

El balance de carga es el proceso de compartir recursos computacionales mediante la distribución transparente de la carga de trabajo del sistema. Los algoritmos de balance estáticos toman decisiones considerando la asignación de tareas a procesadores basados en valores promedio estimados de tiempo de ejecución de procesos.

Los algoritmos de balance de carga estático realizan todas las consideraciónes de información de la carga del sistema antes de la ejecución de las tareas, mientras que aquellos con enfoque dinámico tienen la cualidad de adaptarse a los cambios que se van presentando durante de la ejecución en el sistem. De tal manera, toman decisiones para transferir una parte de su carga en tiempo de ejecución.

Existe una amplia variedad de algoritmos de balance de carga dinámico. A pesar de que la variedad puede ser muy amplia, todos ellos se construyen considerando las suguientes políticas:

- La política de información indica cuándo, de dónde y qué es lo que se requiere recolectar en los nodos del sistema.

- La política de transferencia, que determina si un nodo está en un estado adecuado para participar en la transferencia de tareas.
- La política de selección, donde se decide cual o cuales tareas deben ser consideradas para ser transferidas.
- Finalmente la política de ubicación, donde se elige a cuál nodo debe ser migrada una tarea seleccionada.

La lógica difusa es una alternativa a la lógica convencional. Busca introducir ambigüedad en los entes que evalúa. Surge como una herramienta para el control de sistemas y procesos industriales complejos, así como para aspectos de electrónica y sistemas expertos.

La aplicación de la lógica difusa en sistemas de control, sistemas expertos, o como en este caso, en redes de computadoras, se especifica mediante un módulo difuso. En la literatura existen diferentes tipos de estos, que se utilizan de acuerdo al propósito que se desee. Uno de los más utilizados en aplicaciones relacionadas con las redes de computadoras es aquél formado por una base de conocimiento donde se encuentran todas las reglas lingüísticas, un módulo para la conversión de valores de entrada en valores difusos, que descompone la entrada en uno o más conjuntos difusos, un módulo de inferencia donde se aplican las reglas especificadas en la base de conocimiento y un módulo inverso al inicial cuyo propósito es adecuar los valores difusos en valores nítidos[2] que se utilizan en el proceso de aplicación del módulo difuso.

[2]El término en inglés para este tipo de valores es conocido como *crisp*, no existe una traducción única para dicho término, por lo tanto, para efectos del trabajo se utiliza el término "valor nítido"

Capítulo 3

Trabajo Relacionado

En este capítulo se presentan algunas de las técnicas más relevantes que se encuentran en la literatura cuyo propósito es la mejora del desempeño en sistemas distribuidos mediante la aplicación de algoritmos de balance de carga. Con el propósito de brindar un panorama más amplio de la diversidad de los algoritmos.

Se describe la aportación principal de los trabajos analizados, las características distintivas y las principales ventajas y limitantes de ésta. Se consideran las estrategias seleccionadas, por ser clasificadas como entre las más relevantes así como por ser estudios recientes en el área.

3.1. Simulación de algoritmos estáticos de balance de carga.

En ésta publicación, Rahmawan y Gondokaryono[13], llevan a cabo una análisis de los algoritmos de balance estático. El estudio tiene como principal ventaja, el fundamentarse en resultados obtenidos mediante la ejecución y análisis de los algoritmos en un simulador de eventos discretos. Los índices de carga empleados son: procesamiento en CPU, memoria utilizada y cantidad de instrucciones en CPU.

En el balance de carga estático, la asignación de tareas se realiza previamente a su ejecución. Existen diferentes enfoques para llevar a cabo dicha asignación, sin embargo el mas utilizado es mediante el modelo Manager Worker [3], es decir por medio de un nodo destinado para tal propósito y que por consiguiente no ejecuta ninguna otra.

Simulación de los algoritmos

El sistema utilizado para llevar a cabo la simulación que se presenta es definido por dos elementos:

- Sistema Distribuido

  Definido por una computadora o nodo destinado para equilibrar la carga, el cual distribuye y asigna las tareas, a N nodos denominados "nodos procesantes", los cuales básicamente son responsables de ejecutar completamente las tareas.

  Finalmente, se tienen los programas paralelos que se van a ejecutar.

  A su vez, los nodos procesantes se componen de CPU, memoria y disco duro.

- Simulador de eventos discretos

Resultados y aportaciones

Los resultados que presentan en su documento son divididos en dos categorías.

- Tiempo de Ejecución.

  El algoritmo de administración central que considera al uso de cpu o disco duro como parámetro de balance, brinda el tiempo de ejecución más rápido.

- Distribución de la Carga.

  Los algoritmos de Administrador central y Distribución por Umbral con uso de CPU, memoria y acceso a disco duro como índices de carga son los que alcanzan las distribuciones de tareas más equitativas

Como aspectos destacables del trabajo, se tienen los siguientes:

- La implementación realizada considera el uso de un sistema distribuido homogéneo, de tal forma que se tiene el control de las características de los nodos involucrados.

- La simulación se realiza con 3 tipos de programas que tienen en cuanto a instrucciones de CPU, diferente dominio de carga , acceso a memoria y disco duro. Por lo que se evalúa la propuesta en distintos niveles de carga de trabajo

Limitaciones

Las principales aspectos limitantes de esta propuesta son:

- El aspecto que resulta desfavorable en este algoritmo se basa en la eficiencia del balance de carga, la cual es inherente a un algoritmo estático; se trata de que este tipo de algoritmos son inflexibles en el sentido de la obligatoriedad derivada en la asignación de las tareas, la cual tiene que ser realizada antes de la ejecución, y sobre la cual no se puede hacer algún ajuste que mejore la distribución de carga.

- El sistema donde se realizan las pruebas es homogéneo, es decir, los nodos tienen las mismas características, lo que puede limitar las variaciones que se encuentran al momento de utilizar los algoritmos estáticos; además de que en sistemas distribuidos reales es muy difícil que exista homogeneidad.

- Otra limitante es derivada de que la simulación es realizada con solamente los algoritmos mencionados lo cual es una buena base para comparación pero deja del lado diversas propuestas que existen en la literatura y que pueden ofrecer variantes en los resultados obtenidos en la distribución de la carga.

3.2. Componentes significativos para el diseño de un algoritmo de balance de carga dinámico.

Mehta y Jinwala [9] describen los principales componentes que constituyen los algoritmos dinámicos de balance de carga. Además sugieren modificaciones a las políticas de información y ubicación para definir un nuevo algoritmo.

Modificación en las políticas del algoritmo dinámico

Se proponen políticas de información bajo demanda modificadas y políticas de ubicación por *broadcast* limitado.

En el caso de las primeras, cuando un nodo entra a un estado de sobrecargado, o con carga baja, inicialmente recolecta solamente información de la carga de los nodos locales vecinos, como locales se refiere a los que se encuentren en el mismo grupo, bajo la idea de que sea factible encontrar un nodo destino, o emisor en el mismo grupo. Si esto ocurre se reduce significativamente el número de mensajes totales en el sistema. Si tal nodo no se encuentra, entonces la búsqueda se extiende hacia los demás grupos de manera secuencial, siendo seleccionados de forma aleatoria con lo que el número de mensajes aumentar de manera gradual y siempre que se encuentre un nodo receptor en algún grupo, ya no es necesario buscar en nodos más lejanos.

En el caso de políticas de ubicación, la solicitud de información de carga se realiza dentro de un grupo en vez de hacerlo en el sistema completo. Si se encuentra un nodo adecuado disponible en el mismo grupo, la carga se transfiere para ejecución.

Resultados y Aportaciones

Las políticas propuestas son analizadas en un sistema compuesto por veinte nodos, que se dividen en cuatro grupos virtuales pero que en conjunto pertenecen al mismo sistema. (Figura 3.1).

Se realiza una clasificación de las diferentes políticas involucradas en un algoritmo distribuido. Tal clasificación es descrita de manera detallada y permite entender de manera más profunda los aspectos inherentes a los diferentes tipos de política conociendo de qué manera afectan en la aplicación del balance de carga.

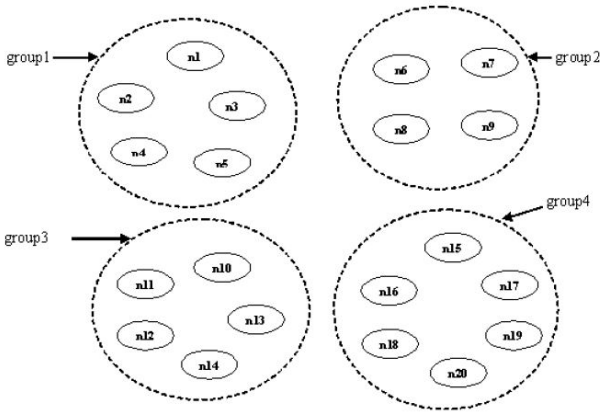

Figura 3.1: Caso de estudio utilizado en [9]

El funcionamiento se da de la siguiente manera: Si un nodo se sobrecarga, inicialmente busca un nodo ligeramente cargado para hacer la transferencia de las tareas dentro de su grupo, en caso de no encontrarlo, buscará en alguno de los otros grupos y así sucesivamente hasta, en el peor de los casos buscar en todos los otros nodos del sistema. Para hacer las transferencias de tareas (y en consecuencia el balance de carga) puede haber casos favorables donde el nodo receptor se encuentre en el grupo original o puede haber casos muy desfavorables donde se tienen que verificar todos los nodos del sistema. En casos promedio, el algoritmo reduce significativamente la sobrecarga de comunicaciones limitando el número de nodos donde se revisa el estado de carga.

Limitaciones

El sistema en el que se desarrollan las pruebas del algoritmo propuesto es uno de baja escala. Los resultados de funcionamiento del algoritmo no son significativos para un sistema de mayor escala.

Otro aspecto que el propio estudio menciona como limitante es el tipo de sistema construido, el cual se cataloga como homogéneo. En este caso el análisis puede ser ampliado tanto en la variación de las características de los nodos como en el uso de recursos. Además de ampliar pruebas correspondientes pero en un sistema heterogéneo.

3.3. Desempeño en balance de carga dinámico utilizando lógica difusa.

Saxena, Khan y Singh [14] proponen una estrategia para alcanzar el balance de carga mediante el uso de un módulo difuso para establecer las reglas de migración de las tareas. También presenta, un análisis comparativo con otro sin tales elementos.

Los módulos de tipo difuso, utilizan funciones de pertenencia en los datos que evalúan. En la propuesta analizada, se aplica el módulo difuso en la ubicación y la migración de tareas. [5, 14]

Integración de módulos difusos

Debido al uso de funciones de pertenencia, existe un criterio más amplio aplicado en cada nodo que integran el sistema para realizar la transferencia de tareas, y en consecuencia, su redistribución en el sistema. Los módulos difusos propuestos para realizar la redistribución de la carga son los siguientes

- Inferencia para la ubicación de tareas. Se utiliza la longitud de la cola de ejecución y el nivel de utilización de CPU, como variables de entrada del módulo difuso; en tanto que la salida es la elección para una posible ubicación. Se define un conjunto de reglas para determinar el grado de pertenencia de los elementos combinando las variables de entrada mencionadas.

- Inferencia para la migración de la carga. Las variables de entrada del módulo difuso las constituyen la carga registrada tanto por el emisor como por el receptor. La salida es un valor de posibilidad (y no de probabilidad) de hacer una transferencia de tareas (migración de carga), entre mayor sea el valor de la posibilidad, más adecuado es el nodo para la transferencia.

Resultados y Aportaciones

En en análisis pudo comprobarse el resultado de la aplicación del algoritmo en la ubicación y migración de tareas mediante la medición de dos parámetros: tiempo de respuesta y desempeño, éste último expresado en la cantidad de tareas ejecutadas.

Se muestra que el tiempo de respuesta disminuye proporcionalmente y el desempeño del sistema se degrada cuando las tareas se incrementan respectivamente sobre el sistema.

La simulación y las pruebas del algoritmo se realizan en un modelo de sistema organizado de manera jerárquica. Tal modelo se ha vuelto común y es utilizado en diversos trabajos relacionados con el balance de carga y con sistemas distribuidos debido a que en sus características es similar a sistemas reales y es un modelo suficientemente representativo de las características tanto de carga como de comunicaciones. Figura3.2.

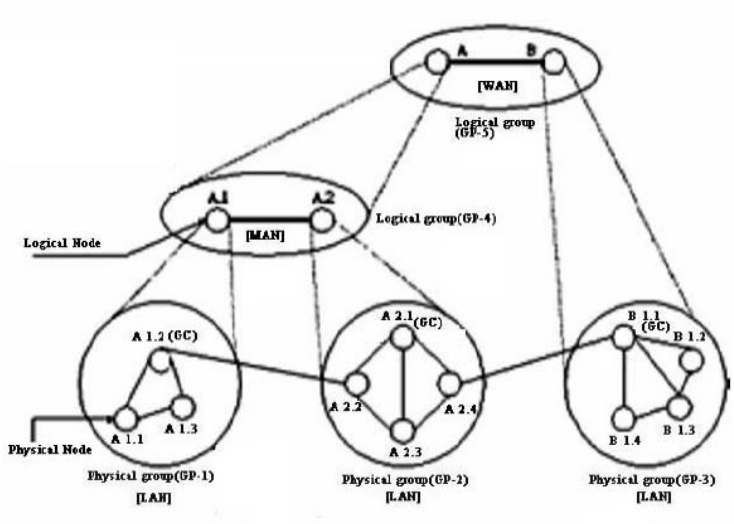

Figura 3.2: Sistema distribuido establecido como un modelo jerárquico.[5, 14]

Limitaciones

El número y la distribución de nodos en el sistema, son parámetros constantes en las simulaciones realizadas, por lo que es factible incrementar su cantidad o modificar las funciones de membresía e inclusive en la topología, con el propósito de verificar si el algoritmo es eficiente bajo otras condiciones en el sistema.

3.4. Resumen

En este capítulo se describen los detalles más importantes acerca de algunas aproximaciones que se encuentran en la literatura sobre algoritmos para resolver el problema de balance de carga en sistemas distribuidos.

En el caso del balance de carga estático, se describe un estudio realizado mediante una simulación de cuatro algoritmos convencionales: round robin, administrador central, aleatorio y distribución por umbral. En los resultados del estudio, se menciona que bajo las características analizadas, el algoritmo de administración central es el que alcanza el mejor desempeño en cuanto a tiempo de ejecución y repartición de la carga como indicadores.

En cuanto a los estudios relacionados con algoritmos dinámicos principalmente se recapitulan tres propuestas:

- Se analizan los elementos significativos de balance de carga dinámico. Se mencionan las características principales de un algoritmo dinámico. Se realiza una propuesta de balance de carga con modificaciones en las políticas de información y de ubicación de tareas, basadas en la búsqueda de los nodos más adecuados para tales fines en grupos de nodos cercanos en primer lugar, y gradualmente en grupos de nodos más lejanos. Los resultados de la propuesta indican que en los mejores casos se disminuye considerablemente la sobrecarga de comunicaciones en el sistema, en tanto que en los peores casos, la carga del sistema relativa a las comunicaciones entre procesadores no se reduce.

- Se analiza el desempeño del balance de carga utilizando lógica difusa, se propone un conjunto de reglas que se aplican mediante un módulo difuso, a la migración de tareas. Dicha estrategia es el punto de partida de la propuesta, ya que presenta una mayor eficiencia con respecto a los algoritmos dinámicos convencionales, que respectivamente muestran una mayor eficiencia que las estrategias de tipo estático.

El propósito de este trabajo de tesis es explorar y extender la propuesta original de los autores pero utilizando elementos de lógica difusa para evaluar parámetros diferentes con el objeto de enriquecer y en el peor de los casos, solamente comprobar que efectivamente, la lógica difusa es un elemento que puede servir en esquemas de balance de carga en sistemas distribuidos

Capítulo 4

Balance de carga eficiente

En este capitulo se describe detalladamente la propuesta planteada para contribuir en el estudio de los sistemas distribuidos a través del alcance de un mayor desempeño utilizando un balance de carga eficiente basado en lógica difusa

Se trata de una propuesta de los criterios para la elaboración de un conjunto de reglas para un algoritmo de balance de carga, tales reglas cuentan con elementos de lógica difusa en las políticas de migración de tareas.

El modelo consiste que las reglas que se establecen para el balance de carga se ejecutan mediante la implementación de un módulo difuso, del cual se mencionan sus características esenciales, la forma en la que se construyen las reglas del mecanismo de inferencia y las funciones de membresía.

Se explica el funcionamiento del algoritmo y finalmente se presenta el caso de estudio en el que se implementa la propuesta. Adicionalmente, se define la forma general para representar un sistema distribuido. Se selecciona un ejemplo que se establece como base para la realización los experimentos respectivos.

4.1. Aspectos básicos de la propuesta

A continuación se describen los elementos teóricos para la mejor descripción de la propuesta.

4.1.1. Definiciones

Sistema Distribuido

Sea un sistema distribuido S, un conjunto de grupos de nodos conectados entre sí mediante una red:

$$S = \{G_1, G_2, ..., G_n\} \tag{4.1}$$

donde cada grupo G_i está constituido por m nodos, y es representado ante el resto del sistema por un coordinador de grupo CG

$$G_i = \{n_{i\cdot 1}, n_{i\cdot 2}, ..., n_{i\cdot m}\} \tag{4.2}$$

$$CG_i = \{n_j\} \tag{4.3}$$

Índice de carga

El índice de carga en cada uno de los nodos que pertenecen al sistema es la métrica preestablecida que se utiliza para evaluar la cantidad de carga que tiene un nodo. La medición del nivel de carga en los nodos que integran un sistema distribuido es importante para contar con información precisa, y en consecuencia, tener más elementos que permitan llevar a una toma de decisiones adecuadas en términos de migración de carga o de ubicación de nuevas tareas que se generen en el sistema.

El índice utilizado para medir la carga de nodo es *el porcentaje de utilización del procesador*. La selección de esta métrica se establece debido a la simplicidad tanto en la implementación como en la medición del mismo, ya que se trata un elemento representativo con el que es posible describir adecuadamente el comportamiento de un nodo.

Umbral de balance

El umbral de balance, también conocido como la tolerancia del sistema, es la diferencia en el índice de carga del nodo con mayor actividad con respecto al nodo

con menor actividad, dependiendo del tipo de algoritmo que se define, este umbral pude variar en tiempo de ejecución o definirse previamente.

4.1.2. Organización e interacción de los nodos

Una de las principales desventajas en las estrategias de balance de carga, es que su ejecución genera una sobrecarga en diferentes aspectos de la red, tales como los canales de comunicación, el tráfico asociado a la ejecución del algoritmo, o la propia carga que se transmite de un nodo a otro. Por otro lado, se genera un incremento en el nivel de utilización de los nodos lo que puede llegar a ser totalmente inconveniente, y en algunos casos no implica un beneficio real con respecto a un sistema que no aplica alguna estrategia de balance.

En diversos trabajos que se encuentran en la literatura, se presenta algunas técnicas para reducir dicha sobrecarga en las comunicaciones del sistema, una de las más comúnmente utilizadas es la de formación de grupos de nodos.

Para efectos de la propuesta no se evalúan los aspectos relacionados a las comunicaciones como intermitencias o variaciones en la velocidad de estas, únicamente se tratan los datos referentes a la carga asociada a cada nodo, sin embargo, se plantea la constucción del sistema a través de la formación de grupos debido a que ello conduce a la reducción en general del envío y recepción de mensajes que se transmiten en la red.

La construcción de grupos en el sistema se establece de la siguiente manera.

- Los grupos físicos (G_i) representan una red de área local LAN.
- Los nodos coordinadores de grupo (CG) se comunican entre sí para intercambiar los aspectos clave de la información de carga y tener información aproximada del estado general del sistema.
- Los grupos del nivel superior inmediato representan redes de área metropolitana (MAN).
- Los grupos más altos en la jerarquía representan redes de área amplia (WAN).

4.1.3. Tipo de tareas y tiempos de respuesta

Se utiliza un modelo para un sistema con tareas independientes entre sí, con procesos, datos, bloques de memoria y operaciones que no necesitan información contenida en otras para realizar su ejecución, esto debido a que uno de los principales

aspectos del balance de carga es la migración de éstas. Así mismo no se considera el tiempo de respuesta de las tareas como un factor de decisión en el balance de carga.

En la generación y ejecución de tareas del sistema, no se consideran aspectos relacionados con respuestas en tiempo real, ya que lo que se persigue no es obtener un resultado específico después de determinado tiempo, sino de una forma eficaz de distribución de las tareas y de equilibrio de la carga.

4.1.4. Planificación

La planificación en un sistema distribuido es la forma en la que se mapean las tareas a los procesadores así como el procedimiento para determinar la secuencia de ejecución en dichos procesadores. Esto se realiza a través de dos componentes: un ubicador *(allocator)* y un planificador *(scheduler)*. El primero decide donde se ejecutará una tarea, y el segundo, el momento en el que recibe su porción de CPU y espacio de memoria correspondientes.

La planificación se maneja principalmente cuando se trata de algoritmos de tipo estático. El caso de este ejemplo, al tratarse de un conjunto de reglas de balance de carga aplicadas a un algoritmo dinámico, no hace énfasis en los aspectos de planificación, sino que se centra en la carga de los nodos y las decisiones para determinar si ejecutar una tarea en el nodo en el que esta se encuentra o transferirla a otro para su ejecución.

4.1.5. Lógica difusa en el balance de carga

En los sistemas distribuidos el comportamiento general de la carga de los nodos suele ser impredecible. Por lo tanto, no existe una manera determinística o basada en un modelo matemático para la ubicación de las nuevas tareas que ofrezca resultados óptimos. Debido a ello, es necesario buscar estrategias alternativas que permitan mejorar gradualmente el rendimiento mediante soluciones aceptables.

Como se mencionó anteriormente, la lógica difusa permite lidiar con ambigüedad e incertidumbre en los aspectos que evalúa, por lo tanto es posible aplicarla para enfrentar los problemas relacionados con los niveles de equilibrio de carga en un sistema distribuido.

La inclusión de un módulo difuso en la construcción de las reglas para realizar el balance de carga tiene la ventaja de que permite considerar un mayor número de elementos en la toma de decisiones del balance y obtener una mayor certeza cuando se realiza una migración de carga.

Con el uso de las reglas difusas se busca que solo se realicen tareas de migración de carga, cuando el hecho de hacerlo garantice un mejor desempeño que el no hacerlo. Ya sea entre aquellos nodos con niveles de carga muy disparejos o evitando realizar transferencias entre nodos con niveles de carga muy similares que resulten más costosas que las propias ejecuciones originales.

4.2. Módulo difuso

El propósito del uso de este módulo, es redistribuir la carga entre los nodos de manera eficiente, maximizando el aprovechamiento de los recursos y reduciendo las transferencias innecesarias.

4.2.1. Variables de Entrada

Los datos de entrada al módulo son: el valor de **carga del nodo** (porcentaje de uso de cpu) y el numero de tareas que tiene para ejecutar es decir la **longitud de cola de las tareas**. Para cada variable respectivamente, se definen las variables lingüísticas correspondientes.

Se seleccionan estos dos valores de datos como entrada debido a que, la carga es el principal valor que nos permite conocer el estado del nodo, mientras que la longitud de fila de tareas pendientes, es un valor complementario que ofrece mayor información para la toma de decisiones de migración. Por ejemplo, si se tiene un nodo con 80 % de carga con cinco tareas, es factible pensar que ambas son pesadas y que migrar alguna de ellas sería poco conveniente para el desempeño del sistema, en tanto que si se tiene el mismo nodo con una carga del 80 % pero con treinta tareas en la cola de ejecución, es factible pensar que dichas tareas son más ligeras y que en algún momento dado se pueden migrar hacia otro(s) nodo(s)

Carga de Nodo

Se utiliza el valor de porcentaje de uso de procesador porque es un dato que no es complejo en su implementación y es representativo en cuanto al comportamiento de un nodo. Los valores lingüísticos propuestos para la variable de carga de nodo son: muy ligera, ligera, moderada, pesada, muy pesada. (Ver figura 4.1)

Se definen cinco valores para las funciones de membresía porque se cuenta con una transición gradual de un estado a otro lo que permite al sistema intuir cuál podría ser el siguiente valor inmediato (ya sea superior o inferior) basado en el

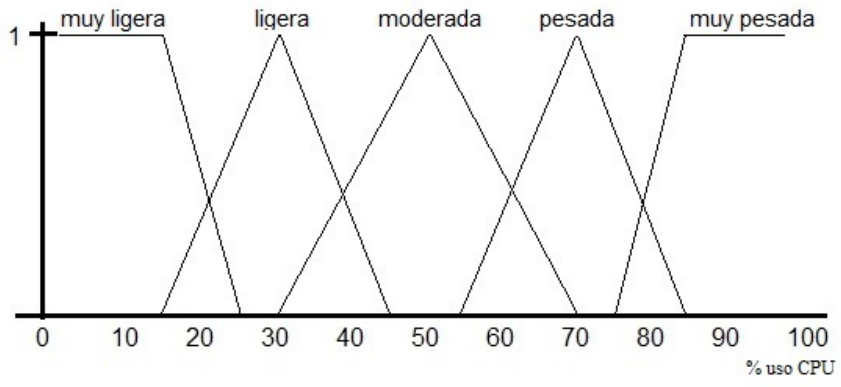

Figura 4.1: Carga de nodo

estado actual, si en un momento dado se agrega una tarea. Si en el sistema hay menos estados, se tiene menor precisión en los datos y las transiciones se asemejan a las de un sistema convencional, mientras que con una cantidad mayor de funciones de membresía, la implementación se puede complicar al grado de que el sistema consumiría una mayor cantidad de recursos únicamente en evaluar el nodo.

Longitud de la cola de tareas

El valor de la longitud de cola de las tareas se refiere al número de tareas pendientes para ejecutar en el nodo. No se toma en cuenta el tipo de tareas o su complejidad, simplemente se verifica la cantidad correspondiente para asociarla con el nivel de carga. Es importante volver mencionar que en el modelo de sistema en general, se considera que no existen tareas con dependencias entre si.

Los valores lingüísticos (funciones de membresía) para esta variable son definidos como: larga, media, corta y muy corta. (Figura 4.2.)

En este caso se proponen cuatro valores debido a que se trata de un parámetro complementario al de la carga y por la naturaleza de la variable que se maneja, se tiene un menor numero de transiciones de un valor hacia otro.

4.2.2. Variable de Salida

Tipo de nodo

Como valor de salida en el módulo de migración se propone la variable *tipo_nodo*. En este caso existen tres posibles opciones de valores, cada una de estas describiendo

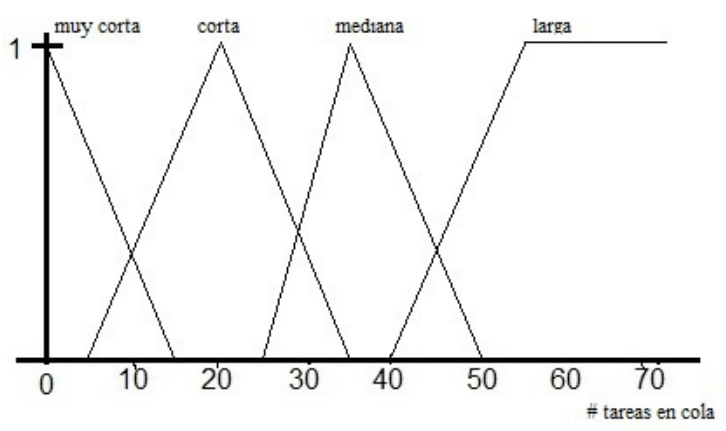

Figura 4.2: Longitud de cola de las tareas

los posibles comportamientos del nodo dentro de un proceso de migración.

Al participar en el procedimiento de migración de carga, un nodo puede enviar parte de su ejecución a otro, recibir una o varias tareas para ejecutar, de igual forma, puede haber casos en los que se encuentre en un nivel de carga intermedio en el que no transfiere ni recibe carga de trabajo. Cada nodo puede ser *emisor*, *receptor* o *neutro*. Si el resultado de la evaluación para los nodos resulta en emisor o receptor, significa que es factible, debido a la cantidad de tareas y carga que tiene, participar en una migración.

4.2.3. Conversión a valores Difusos

El procedimiento para convertir una entrada en variable difusa es el siguiente. Se recibe el valor nítido de la entrada y se ubica dentro de la gráfica que representa la variable difusa. Al ubicarlo es posible que solo sea parte de un valor lingüístico, si este ese el caso, el valor adquirido será 100 %. En caso de que se encuentre en un punto de la gráfica en el que sea parte de dos conjuntos difusos, su valor será el porcentaje de cada una de esas variables

4.2.4. Mecanismo de inferencia

Una vez que se tienen los valores difusos, estos ingresan al mecanismo de inferencia y se evalúan contra el conjunto de reglas definidas en él.

A la salida del mecanismo de inferencia, se obtiene un valor difuso que está con-

formado por un cierto porcentaje de cada uno de los posibles valores de salida.

Las reglas del mecanismo de inferencia propuesto para realizar el balance de carga en sistemas distribuidos son las siguientes.

Regla [1]: ***If*** (Carga de nodo es muy ligera AND longitud de cola de tareas es muy corta) ***then*** (nodo es receptor en un 100 %)
Regla [2]: ***If*** (Carga de nodo es muy ligera AND longitud de cola de tareas es corta) ***then*** (nodo es receptor en un 80 %)
Regla [3]: ***If*** (Carga de nodo es muy ligera AND longitud de cola de tareas es mediana) ***then*** (nodo es receptor en un 50 %)
Regla [4]: ***If*** (Carga de nodo es muy ligera AND longitud de cola de tareas es larga) ***then*** (nodo es receptor en un 30 %)
Regla [5]: ***If*** (Carga de nodo es ligera AND longitud de cola de tareas es muy corta) ***then*** (nodo es receptor en un 80 %)
Regla [6]: ***If*** (Carga de nodo es ligera AND longitud de cola de tareas es corta) ***then*** (nodo es receptor en un 60 %)
Regla [7]: ***If*** (Carga de nodo es ligera AND longitud de cola de tareas es mediana) ***then*** (nodo es neutral en un 70 %)
Regla [8]: ***If*** (Carga de nodo es ligera AND longitud de cola de tareas es larga) ***then*** (nodo es neutral en un 30 %)
Regla [9]: ***If*** (Carga de nodo es moderada AND longitud de cola de tareas es muy corta) ***then*** (nodo es receptor en un 60 %)
Regla [10]: ***If*** (Carga de nodo es moderada AND longitud de cola de tareas es corta) ***then*** (nodo es neutral en un 60 %)
Regla [11]: ***If*** (Carga de nodo es moderada AND longitud de cola de tareas es mediana) ***then*** (nodo es neutral en un 100 %)
Regla [12]: ***If*** (Carga de nodo es moderada AND longitud de cola de tareas es larga) ***then*** (nodo es neutral en un 40 %)
Regla [13]: ***If*** (Carga de nodo es pesada AND longitud de cola de tareas es muy corta) ***then*** (nodo es neutral en un 50 %)
Regla [14]: ***If*** (Carga de nodo es pesada AND longitud de cola de tareas es corta) ***then*** (nodo es neutral en un 30 %)
Regla [15]: ***If*** (Carga de nodo es pesada AND longitud de cola de tareas es mediana) ***then*** (nodo es neutral en un 40 %)
Regla [16]: ***If*** (Carga de nodo es pesada AND longitud de cola de tareas es larga)

then (nodo es emisor en un 80 %)
Regla [17]: ***If*** (Carga de nodo es muy pesada AND longitud de cola de tareas es muy corta) ***then*** (nodo es emisor en un 40 %)
Regla [18]: ***If*** (Carga de nodo es muy pesada AND longitud de cola de tareas es corta) ***then*** (nodo es emisor en un 60 %)
Regla [19]: ***If*** (Carga de nodo es muy pesada AND longitud de cola de tareas es mediana) ***then*** (nodo es emisor en un 80 %)
Regla [20]: ***If*** (Carga de nodo es muy pesada AND longitud de cola de tareas es larga) ***then*** (nodo es emisor en un 100 %)

4.2.5. Conversión a valores nítidos

Existen diferentes métodos para el proceso de convertir una salida difusa en una salida nítda. En el balance de carga, se procede a verificar en cual de los valores difusos tiene un mayor porcentaje, es decir, se toma en cuenta la fuerza de aplicación de las reglas.

4.3. Caso de Estudio

4.3.1. Descripción general

- El caso de estudio se basa en un sistema distribuido compuesto por tres grupos físicos. Cada uno con cinco nodos, entre los cuales se encuentra el respectivo coordinador de grupo. Cada grupo es un nodo lógico que representa el grupo completo para el siguiente nivel. Entre los coordinadores de grupo hay un coordinador general del sistema, el cual ejecuta el balance de carga global. En cada grupo, el coordinador se encarga de recabar la información de los otros nodos, ejecutar el procedimiento de balance de carga local y tomar la decisión respecto a la ubicación de las nuevas tareas y a la migración de las mismas.

4.3.2. Representación de la carga.

En la ejecución convencional de un sistema, las tareas nuevas aparecen o se generan de manera aleatoria y en algún momento a lo largo de dicha ejecución, se genera la disparidad entre la cantidad de tareas de un nodo con respecto a otro, por lo que es necesario un proceso de balance. En la representación que se plantea, la

idea principal es analizar un sistema que ya se encuentre con determinados valores de carga en los nodos para realizar el procedimiento de balance de carga. Se parte de valores conocidos para analizar de manera más adecuada el comportamiento del algoritmo, ya que el uso de valores aleatorios puede generar confusión en el comportamiento del sistema y del algoritmo.

4.3.3. Ejecución

Al inicio se genera para cada nodo del sistema distribuido del caso de estudio, una serie de datos que representan el nivel de carga, un valor promedio de nivel de carga por tarea, que a su vez, genera un valor para la cantidad de tareas que se ejecutan por nodo. El propósito es mantener un ambiente controlado que tenga condiciones donde se puede evaluar la forma en que se ejecuta el algoritmo, y por consiguiente pueda ser aplicado a un sistema distribuido convencional.

Los coordinadores de cada grupo de nodos, verifican los niveles de carga de los elementos pertenecientes a su grupo y una vez que conocen esta información, la comparten entre sí, posteriormente uno de estos coordinadores previamente designado como el coordinador general, analiza la información y realiza la aplicación del balance de carga.

4.3.4. Aplicación del balance de carga

Se trata de un algoritmo estático cuyas bases se pueden ampliar para convertirse posteriormente en un algoritmo dinámico. La ejecución es centralizada pero de igual forma es posible aplicarla de manera totalmente distribuida. La toma de decisión para llevar a cabo el procedimiento de balance de carga, está a cargo del coordinador general debido a se cuenta con información aproximada del sistema y al realizar el análisis de las condiciones de los nodos, determina si el sistema se encuentra balanceado o si es factible un intercambio de carga en algunos de los nodos.

Es importante mencionar que por la propia naturaleza de un sistema distribuido, siempre se tiene un cierto grado de incertidumbre, ya que de un instante de tiempo a otro, la información de carga de los nodos puede cambiar. Por ello, se utiliza la lógica difusa en la elaboración de las reglas, en las que se busca transición más gradual de un valor a otro en los niveles de carga, además es posible lidiar con ese nivel de incertidumbre.

Comúnmente, en un algoritmo de balance de carga convencional ya sea de tipo estático o de tipo dinámico, el umbral de balance se define previamente, sin embargo,

en diversas ocasiones puede que dicho umbral nunca se alcance, entre otras razones por la generación de nuevas tareas o por el cambio de estado de un nodo al recibir un porcentaje de carga de otro nodo, y al no alcanzar el umbral predefinido puede ocurrir que una o varias tareas únicamente se esten moviendo de un nodo a otro y nunca se ejecuten. Lo cual se evita con el uso de lógica difusa ya que en un momento dado cuando todos los nodos se clasifiquen con un cierto valor, el proceso de balance termina.

4.4. Integración del módulo difuso en el sistema

La forma en la que se utiliza el módulo difuso en el sistema es la siguiente:

1. Conversión de valor nítido a difuso (entrada)

   Cada nodo evalúa su estado tanto de carga como de longitud de cola de las tareas. Determina su valor difuso de acuerdo al punto donde se encuentre en las gráficas que representan las variables difusas.

2. Base de conocimiento

   La base de conocimiento son las gráficas de las variables difusas tanto de la carga como la de la longitud de cola de las tareas. Los cuales se van a combinar en el mecanismo de inferencia para producir una salida difusa, y posteriormente una salida nítida.

3. Mecanismo de inferencia

   Una vez que se tienen los valores difusos, se evalúan con respecto a cada una de las reglas que se encuentran en el mecanismo de inferencia. Al evaluar estas reglas, el nodo se va a catalogar con un valor difuso que tendrá algún porcentaje para cada uno de los valores de salida

4. Conversión a valor nítido (salida)

   En el proceso inverso, se selecciona el valor de salida final para el nodo, esto es, emisor, receptor o neutro. El cual establece la capacidad del nodo de realizar envío o recepción de carga.

4.4.1. Balance de carga

Una vez que los nodos han evaluado su condición de carga por medio del módulo difuso, envían esa información al coordinador general del sistema, éste nodo es el

encargado de realizar el análisis de las condiciones del sistema y de informar a los nodos emisores, hacia qué nodos migrarán parte de su carga. El algoritmo que ejecuta el coordinador general es el siguiente

Algoritmo 1 Balance de carga

Entrada: Información del estado de los nodos del sistema
Salida: **cierto** si el sistema esta balanceado Sistema Balanceado.

```
1: mientras N_T > 0 hacer
2:   si N_R > 0 entonces
3:     Elige N_T con mayor valor de carga
4:     Elige N_R con menor valor de carga
5:     Tranfiere carga de N_T a N_R
6:   si no
7:     El sistema está balanceado
8:     devolver cierto
9:   fin si
10: fin mientras
11: devolver cierto
```

4.4.2. Constantes

Los elementos que permanecen constantes en la representación del sistema, es decir sin cambios antes o después de la ejecución del algoritmo son:

- **Características de los nodos.** Se utilizan nodos homogéneos con el propósito de que no existan variaciones entre ellos que puedan afectar el desempeño del algoritmo.
- **Canales de Comunicación.** Debido a que los retardos en las comunicaciones no se consideran en el análisis como elementos para tomar las decisiones de balance de carga, las velocidades y características de los enlaces definidos, se consideran sin cambios.
- **Asignación de coordinadores de Grupo.** En cada grupo de nodos, el coordinador designado no cambia en ningún momento, ya que por el modelo definido se asume que la comunicación no será interrumpida en ningún momento.
- **Número de nodos por grupo.** Se trata del mismo numero de nodos ya que se analizan las condiciones de ejecución del algoritmo bajo esta circunstancia.

- **Identificador de Grupo.** Cada grupo tiene un valor que lo identifica en el sistema, dicho identificador permanece sin cambios para facilitar la identificación en el proceso de balance

4.4.3. Variables

Son elementos que se modifican en diferentes instancias de la simulación del sistema así como de manera previa y posterior a la ejecución del algoritmo para balance. Entre estos:

- **Carga de los nodos.** Representa la carga que tienen los nodos sobre la cual se establece el módulo difuso
- **Numero de tareas por nodo.** Se trata de la cantidad de tareas por nodo reflejada en la longitud de la cola de tareas pendientes por ejecutar.
- **Valor promedio de carga por tarea.** Se trata de un valor estimado para la simulación que consiste en la cantidad de carga que va a representar cada una de las tareas presentes en el sistema. Por ejemplo. Si se tienen cinco tareas con una carga promedio igual a uno, se tiene un porcentaje de uso de procesador de 5. En tanto que si se tienen las mimas cinco tareas con un valor promedio de carga por tarea igual a cinco, entonces se tiene un porcentaje de uso de procesador de 25

4.4.4. Datos a evaluar

- **Umbral de balance** Tanto el umbral que determina la aplicación del balance, como los umbrales definidos en la aplicación de los módulos difusos (sección 4.3) se mantienen fijos en el experimento.
- **Carga de los nodos.** Como resultado de la migración de carga, se analiza el valor final de la misma en los nodos del sistema.
- **Ejecuciones del algoritmo.** Se trata del análisis del número de iteraciones necesarias en las que el algoritmo finaliza el equilibrio de la cantidad de carga.

4.5. Resumen

La propuesta presentada consiste en la definición de los criterios para la elaboración de un conjunto de reglas para un algoritmo de balance de carga, tales reglas cuentan con elementos de lógica difusa en las políticas de migración de tareas.

El índice de carga es la métrica establecida que se utiliza para evaluar la cantidad de carga que tiene un nodo. El índice utilizado para medir la carga de nodo es *el porcentaje de utilización del procesador*. Se utiliza un modelo para un sistema con tareas independientes entre sí, con procesos, datos, bloques de memoria y operaciones que no necesitan información contenida en otras para realizar su ejecución, esto debido a que uno de los principales aspectos del balance de carga es la migración de éstas.

La inclusión de un módulo difuso en la construcción de las reglas permite considerar un mayor número de elementos en la toma de decisiones y obtener mayor certeza cuando se realiza una migración de carga. Con el uso de las reglas difusas se busca que solo se realice migración de carga, cuando sea realmente necesario.

Las variables de entrada del módulo difuso son la carga y el número de tareas en los nodos, mientras que la salida será la clasificación para el nodo, ya sea emisor, receptor, o neutro, en tal caso no realizará migración de tareas.

Al finalizar la ejecución del algoritmo, se obtiene el umbral de carga resultante, el cual es la diferencia entre la carga del mayor nodo con respecto a la carga del menor nodo.

Capítulo 5

Evaluación Experimental

En este capítulo se presentan los experimentos realizados con el propósito de comprobar si se obtiene un balance de carga más eficiente con el uso de un algoritmo estático cuya construcción de reglas se basa en lógica difusa. Se describe la herramienta de software utilizada para la realización de simulaciones, así como los aspectos particulares de los mismos.

5.1. Simulación de Sistemas Distribuidos

La evaluación experimental de la propuesta se lleva acabo mediante el uso de un simulador, debido a que una simulación permite efectuar diversos tipos de pruebas en ambientes controlados. Es por ello que una simulación permite la representación adecuada del comportamiento de nodos, los diversos tipos de enlaces, la interacción entre diferentes tipos de hardware etc.

En una simulación aplicada a balance de carga, la principal ventaja que se tiene es que es posible distinguir las variaciones del sistema como consecuencia de agregar diferentes niveles de carga de trabajo, conocer el valor de carga que representa cada tarea, y por consiguiente, el comportamiento de los grupos al aplicar la estrategia propuesta.

Existen diversos simuladores para representar redes de datos y sistemas distribuidos. En este caso se utiliza el Software de Matlab, particularmente una biblioteca desarrollada por la universidad de Lund, *True-Time*, la cual es utilizada para la simulación de redes y sistemas de control con elementos en tiempo real. (Ver Figura 5.1). Utilizando dicha biblioteca, es posible ejecutar diversas funciones, para el envío y recepción de datos, para la creación de grupos y algunas otras características aplicables tanto a un controlador de tiempo real, o en este caso, un sistema distribuido [10]. El software simula un proceso de tiempo real ejecutando tareas definidas por el usuario y manejo de interrupciones. Los diferentes bloques de red permiten a los nodos comunicarse sobre redes alambradas o inalámbricas, para efectos de esta tesis, únicamente se utilizan aspectos de redes alambradas.

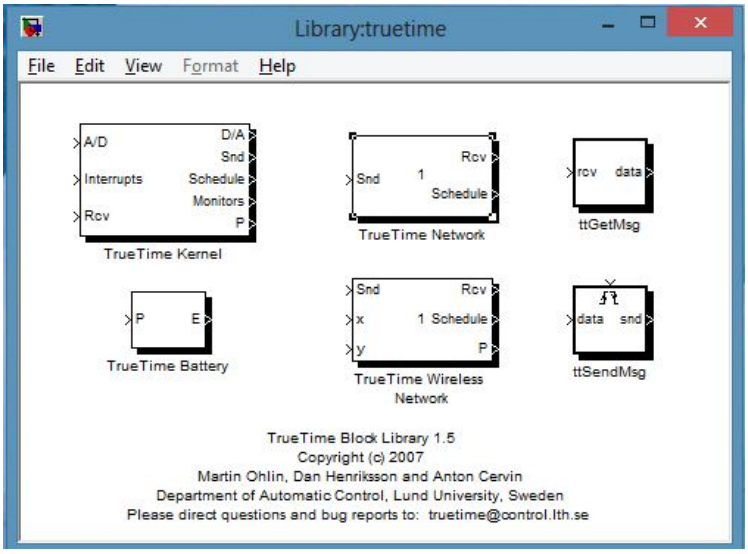

Figura 5.1: Bloques básicos de True-Time

5.2. Experimentos de Balance de carga

En esta sección se describen los diferentes escenarios en los que se pone a prueba la ejecución del algoritmo de balance de carga basado en lógica difusa. Inicialmente se definen los elementos participantes en los escenarios experimentales

- **Umbral Objetivo.-** Es el valor deseado de la diferencia de carga del nodo mas alto con respecto al más bajo cuando el algoritmo finaliza su ejecución.

- **Identificador de Grupo.-** Es el valor identificador para cada uno de los grupos de nodos con conforman el sistema. Dicho valor es utilizado como referencia para la organización y la ubicación de los nodos, sin embargo no tiene inferencia en las del algoritmo o en el comportamiento de la migración de la carga

- **Variables Difusas.-** Elementos cuyos valores determinan el curso de acción del algoritmo de balance. Los valores para la carga se establecen previamente para todos los escenarios debido para contar con un mayor control de la situación de carga del sistema. En tanto que los valores para el número de tareas se modifican en cada uno de los escenarios para analizar el comportamiento del algoritmo en ellos.

- **Valor promedio de carga por tarea.-** Es la cantidad de carga que representa una tarea. Dicho valor influye directamente en la cantidad de tareas con respecto a la carga que se tiene en el sistema. En un sistema distribuido real, se encuentra una diversidad de tareas con valor de carga diferente, sin embargo en este caso se analizan valores constantes para tener una base del comportamiento del algoritmo.

- **Numero de nodo.-** Valor identificador para cada uno de los nodos pertenecientes a un grupo.

- **Coordinador de Grupo.-** Nodo encargado de enviar y recibir información relacionada con los otros nodos del sistema. La elección del coordinador de grupo es arbitraria, ya que al no considerar los tiempos y retardos en comunicaciones, es indistinta la ubicación que tengan estos nodos en el sistema.

5.2.1. Escenario 1

Para el primer escenario se define un valor promedio de carga por tarea igual a uno, cada tarea representa 1 % de la carga del nodo. El Umbral de balance objetivo para este escenario es de 20 unidades de carga

Caso 1.1 Sistema completamente desbalanceado

Se establece el sistema donde los nodos del primer grupo se encuentran cargados prácticamente a su máxima capacidad mientras que los nodos de los otros dos grupos de encuentran en estado ocioso con una carga de 5.

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	100	100	5	5	5	5
Nodo 2	100	100	5	5	5	5
Nodo 3*	100	100	5	5	5	5
Nodo 4	100	100	5	5	5	5
Nodo 5	100	100	5	5	5	5

* Nodo coordinador de grupo

Tabla 5.1: Condiciones iniciales del experimento para escenario 1 con el sistema en un estado completamente desbalanceado

Caso 1.2 Sistema parcialmente desbalanceado

Para esta prueba los nodos de los primeros dos grupos se encuentran altamente cargados mientras que los nodos del tercer grupo se encuentran en estado ocioso.

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	80	80	80	80	5	5
Nodo 2	80	80	80	80	5	5
Nodo 3*	80	80	80	80	5	5
Nodo 4	80	80	80	80	5	5
Nodo 5	80	80	80	80	5	5

Tabla 5.2: Condiciones iniciales del experimento para escenario 1 con el sistema en un estado parcialmente desbalanceado

5.2.2. Escenario 2

En los experimentos del segundo escenario se utiliza un valor promedio de carga por tarea igual a dos. Se mantiene como umbral de balance deseado, 20 unidades de carga.

Caso 2.1 Sistema completamente desbalanceado

Se inicia con el caso donde el primer grupo de nodos se encuentra altamente cargado y los otros grupos se encuentran en estado ocioso.

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	100	50	6	3	6	3
Nodo 2	100	50	6	3	6	3
Nodo 3*	100	50	6	3	6	3
Nodo 4	100	50	6	3	6	3
Nodo 5	100	50	6	3	6	3

Tabla 5.3: Condiciones iniciales del experimento para escenario 2 con el sistema completamente desbalanceado

Caso 2.2 Sistema parcialmente desbalanceado

El siguiente caso contempla la carga de los nodos en estado alto en los primeros dos grupos y en estado ocioso para el tercer grupo.

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	80	40	80	40	6	3
Nodo 2	80	40	80	40	6	3
Nodo 3*	80	40	80	40	6	3
Nodo 4	80	40	80	40	6	3
Nodo 5	80	40	80	40	6	3

Tabla 5.4: Condiciones iniciales del experimento para escenario 2 con el sistema parcialmente desbalanceado

5.2.3. Escenario 3

Finalmente en este escenario se propone un valor promedio de carga por tarea de 5 unidades. El propósito de este escenario es evaluar el desempeño del algoritmo con tareas más "pesadas". Es decir, tareas cuyo valor de uso de cpu es más alto que en los escenarios anteriores.

Caso 3.1 Sistema totalmente desbalanceado

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	100	20	5	1	5	1
Nodo 2	100	20	5	1	5	1
Nodo 3*	100	20	5	1	5	1
Nodo 4	100	20	5	1	5	1
Nodo 5	100	20	5	1	5	1

Tabla 5.5: Condiciones iniciales del experimento para escenario 3 con el sistema en un estado parcialmente desbalanceado

Caso 3.2

Umbral de Balance	20					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	80	16	80	16	5	1
Nodo 2	80	16	80	16	5	1
Nodo 3*	80	16	80	16	5	1
Nodo 4	80	16	80	16	5	1
Nodo 5	80	16	80	16	5	1

Tabla 5.6: Condiciones iniciales del experimento para escenario 3 con el sistema en un estado parcialmente desbalanceado

5.3. Resultados Obtenidos

A continuación se presentan los resultados obtenidos en cada escenario experimental. Los elementos a considerar son los siguientes:

- **Resultado de balance.-** Una vez que el algoritmo finaliza su ejecución, se verifica el valor tanto de carga como de número de tareas en los nodos del sistema con el fin de verificar si el algoritmo logra su propósito de realizar el balance de carga.
- **Umbral Alcanzado.-** Es el valor obtenido entre la diferencia del nodo mas alto con respecto al más bajo cuando el algoritmo finaliza su ejecución. Si dicho umbral se encuentra dentro de los límites definidos en el inicio, (Umbral objetivo) se puede considerar al sistema como balanceado.

5.3.1. Escenario 1

Resultado de Balance.- Se presenta en la tabla 5.7 para el sistema totalmente desbalanceado y en la tabla 5.8 para el sistema parcialmente desbalanceado.

Umbral alcanzado.- El umbral alcanzado para el primer caso es de 12 unidades de carga y 19 unidades de carga para el segundo.

Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	40	40	28	28	28	28
Nodo 2	40	40	40	40	40	40
Nodo 3	40	40	28	28	28	28
Nodo 4	40	40	40	40	40	40
Nodo 5	40	40	40	40	28	28

Tabla 5.7: Resultado de balance para escenario 1, sistema desbalanceado

5.3.2. Escenario 2

Los resultados para la representación de tareas con un valor promedio de carga de 2 son los siguientes:

Resultado de Balance.- Se describe en la tabla 5.9 para el sistema totalmente desbalanceado y en la tabla 5.8 para el sistema parcialmente desbalanceado.

Umbral alcanzado.- El umbral alcanzado para el primer caso es de 12 unidades de carga y 18 unidades de carga para el segundo.

Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	61	61	61	61	42	42
Nodo 2	61	61	61	61	42	42
Nodo 3	61	61	42	42	61	61
Nodo 4	61	61	61	61	42	42
Nodo 5	42	42	61	61	61	61

Tabla 5.8: Resultado de balance para escenario 1, sistema parcialmente desbalanceado

Umbral Alcanzado	12					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	40	20	42	21	42	21
Nodo 2	30	15	40	20	30	15
Nodo 3	30	15	42	21	40	20
Nodo 4	30	15	42	21	42	21
Nodo 5	30	15	40	20	40	20

Tabla 5.9: Experimento con Umbral de Balance de 20 para valor promedio de carga de 1

Umbral Alcanzado	18					
Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	62	31	44	22	60	30
Nodo 2	44	22	62	31	60	30
Nodo 3	44	22	62	31	60	30
Nodo 4	62	31	44	22	60	30
Nodo 5	44	22	62	31	60	30

Tabla 5.10: Experimento con Umbral de Balance de 20 para valor promedio de carga de 1

5.3.3. Escenario 3

Ahora se presentan los resultados para la representación de tareas con un valor promedio de carga de 5 unidades por cada tarea.

Resultado de Balance.- Se puede ver en las tablas 5.11 para el sistema totalmente desbalanceado y 5.12 para el sistema parcialmente desbalanceado.

Umbral alcanzado.- El umbral alcanzado para ambos casos fue de 12 unidades

de carga en este caso podemos apreciar que el umbral de balance fue el mismo debido a que hay un número menor de tareas que participan en la migración de carga, y por la naturaleza de las mismas transferir más no resulta conveniente para el sistema.

Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	45	9	30	6	45	9
Nodo 2	40	8	45	9	40	8
Nodo 3	40	8	30	6	45	9
Nodo 4	45	9	40	8	30	6
Nodo 5	30	6	30	6	40	8

Tabla 5.11: Experimento con Umbral de Balance de 20 para valor promedio de carga de 1

Identificador de Grupo	1		2		3	
Variables	Carga	Tareas	Carga	Tareas	Carga	Tareas
Nodo 1	60	12	65	13	40	9
Nodo 2	60	12	60	12	40	8
Nodo 3	65	13	60	12	40	9
Nodo 4	65	13	65	13	40	6
Nodo 5	60	12	65	13	40	8

Tabla 5.12: Experimento con Umbral de Balance de 20 para valor promedio de carga de 1

5.4. Resumen

Este capítulo presenta los experimentos realizados para comprobar la propuesta de solución al problema de balance de carga.

Se proponen tres escenarios con casos de evaluación por cada uno de ellos, para todos los casos, se propone un umbral de balance objetivo de 20 unidades de carga, y en cada escenario se propone un caso donde se encuentra un sistema en un estado totalmente desbalanceado y uno en un estado parcialmente desbalanceado. Con estos casos se analiza el comportamiento del algoritmo. En los diferentes escenarios cambia el valor promedio de carga por tarea, donde se analiza el comportamiento del algoritmo para tareas muy ligeras de 1 % de carga por tarea y para tareas más pesadas de 2 y de 5 unidades de carga por tarea.

Se presentan los resultados obtenidos en cada uno de los escenarios anteriores mostrando el umbral de balance alcanzado así como el resultado de balance en el sistema (la carga final para cada uno de los nodos).

Se evalúa cada uno de los casos en 10 ocasiones, y se encuentra que en todos los casos se alcanzaron los mimsos niveles tanto en el resultado final de balance como en el umbral de balance alcanzados. Lo cual nos conduce a pensar que el algoritmo realiza su trabajo de forma eficaz de acuerdo a los resultados esperados.

Capítulo 6

Conclusiones

En este capítulo se presenta un análisis basado en la propuesta planteada así como los resultados obtenidos en el capítulo anterior, se describen las aportaciones de la investigación en el área de las redes de datos y los sistemas distribuidos. Se listan las conclusiones finales con respecto a la hipótesis planteada en este trabajo.

En la comparación con estrategias existentes se presentan las diferencias principales con la propuesta realizada, resaltando las aportaciones que se alcanzan. Finalmente se proponen nuevos objetivos para posible trabajo futuro siguiendo la misma línea de investigación.

6.1. Análisis Global

El objetivo primordial del balance de carga es obtener un mayor aprovechamiento de los recursos disponibles en el sistema, así como una mas alta disponibilidad para ejecutar diferentes tareas en un sistema distribuido que conduzcan a una mejora en el desempeño global de los mismos.

Es posible alcanzar ese objetivo mediante el balance de carga, sin embargo no todos los sistemas son susceptibles de ser balanceados, ya que la cantidad de tares y la capacidad de los sistemas para poder enviar o recibir carga de trabajo puede variar por diferentes circunstancias. Inclusive en aquellos que sí se pueden balancear, siempre existe un límite en el nivel de tolerancia que se puede alcanzar.

6.1.1. Resultados

Cada uno de los respectivos casos de cada escenario de experimentación fue ejecutado en 10 ocasiones, en cada una de las cuales se verifica el resultado final del sistema considerado como balanceado y el umbral de balance alcanzados.
Los resultados nos indican que el algoritmo cumple con el propósito de alcanzar el balance de carga considerando las condiciones planteadas en la propuesta (ver capítulo 4).

En el primer y segundo escenarios donde las tareas representan poca carga para el sistema, los resultados de balance fueron similares. En ambos casos se obtuvo una tolerancia de 12 unidades para el sistema totalmente sin balancear y una diferencia de balance menor a 20 para el sistema parcialmente sin balancear

Se piensa que este umbral se pudiera reducir, pero precisamente una de las características que brinda el uso de la lógica difusa es tener una mayor tolerancia en la incertidumbre, ya que al intentar reducir este umbral de balance puede resultar más costoso que continuar la ejecución en el estado alcanzado.

En el caso del tercer escenario, se puede ver que al reducir el número de tareas que participan en la migración de carga se alcanza el mismo umbral de balance en ambos casos (total y parcialmente sin balancear).

Con un mayor número de tareas (con menor carga por tarea) se tiene un mayor número de eventos de migración de tareas sin embargo, en cada nodo, al tener una evaluación con los módulos difusos no se pierde el control de la situación de carga al enviar o recibir información.

6.2. Conclusiones finales de la propuesta

6.2.1. Conclusiones

Finalmente partiendo de los resultados de los experimentos realizados se puede establecer lo siguiente.

- Un algoritmo de balance de carga cuyas reglas se basan en lógica difusa logra el objetivo de balancear un sistema distribuido. Debido a que el resultado de su ejecución en los nodos se encuentra dentro de los umbrales definidos como objetivo.

- El uso de reglas basadas en lógica difusa para las decisiones de balance de carga permite que se realice un menor número de eventos de migración, ésto debido

a que cuando las variables difusas indican que los nodos se encuentran en un estado neutral, el algoritmo se detiene, por lo que los procesos de migración solo se van a llevar acabo cuando sean estrictamente necesarios y el realizarlo no resulte más costoso.

- El hecho de no alcanzar umbrales de balance muy pequeños no implica que el algoritmo no es eficiente en su ejecución, sino que implica que dichos umbrales tan bajos no son factibles de alcanzar sin degradar el desempeño de un sistema.

6.2.2. Contribuciones

Las contribuciones globales del trabajo son las siguientes:

- La contribución principal es el algoritmo de balance de carga con elementos de lógica difusa para las reglas de migración de carga.
- Se analizan las características que debe tener un algoritmo de balance de carga así como las condiciones bajo las cuales se puede o no aplicar dicho balance.
- Se establecen las bases de una técnica que con su aplicación conduce al mayor aprovechamiento de los recursos de un sistema distribuido, y en consecuencia, la mejora en el desempeño.

6.3. Comparación con estrategias existentes

En el capítulo 3, se describieron algunas de las propuestas más relevantes en relación con algoritmos de balance de carga. A continuación se describen las diferencias de dichas propuestas en relación a la propuesta planteada.

En el caso de la propuesta referente a la simulación de algoritmos de balance de carga estático [13], se evalúan algoritmos convencionales, los cuales en un momento dado brindan eficacia pero que tienen el problema de que pueden caer en sobrecarga más grande generada por la aplicación del algoritmo, lo cual no ocurre en la propuesta realizada debido a que al evaluar los nodos con la característica de emisor, receptor o neutro, puede determinar que en un momento dado.

La propuesta referente a un algoritmo dinámico con políticas [9] modificadas indica que se reduce la cantidad de mensajes en el sistema, dicho esquema se incorpora

en la propuesta realizada y se extiende mayormente con el uso de la lógica difusa para las decisiones de migraciones de carga.

Finalmente la propuesta que planeta una solución de balance de carga con el uso de lógica difusa, que es la base de la propuesta actual, plantea el uso de lógica difusa bajo ciertas variables, que en esta propuesta se extiende combinando la carga y la longitud de cola de las tareas.

6.4. Trabajo Futuro

Algunas de las posibles lineas de investigación que pueden surgir a partir de la propuesta realizada son las siguientes:

6.4.1. Inclusión de un módulo difuso adicional para la ubicación de nuevas tareas

Una primer idea para continuar en el ámbito del balance de carga con el uso de lógica difusa en sus reglas es la inclusión de un segundo módulo difuso. El módulo utilizado en esta propuesta tiene el propósito de alcanzar un equilibrio de carga en los nodos del sistema, pero una vez que ese equilibrio ha sido alcanzado, una posible forma de mantenerlo es con el uso de un segundo módulo difuso para la ubicación de las nuevas tareas que surgen en el sistema. La naturaleza de dichas tareas es aleatoria, sin embargo se puede acotar indicando el valor promedio de carga por tarea. A continuación se presenta un esbozo de posible módulo difuso.

Variables de entrada

Las variables de entrada para este modulo pueden ser la carga de trabajo (de la cual ya se puede tener un adelanto en esta propuesta) y la memoria disponible de cada nodo.

Salida

Como función de pertenencia para la salida del módulo de ubicación de nodos se puede utilizar la variable *elección*, la cual determina qué tan adecuada es la ubicación de tarea para un determinado nodo.

Los posibles valores de funciones de membresía para esta variable pueden ser: Excelente, buena, aceptable y mala. Para tomar las decisiones finales de elección de

nodo, para aquellos cuya variable resulte con una valoración mala, deberán buscar otro nodo para ubicar las tareas nuevas, en tanto que en el caso del valor aceptable, solamente cuando sea con el 100 %, se podrá ubicar una tarea en ese nodo, y finalmente, a partir del 50 % en el caso de una elección buena y para cualquiera que resulte con excelente las tareas se ubican en ese nodo.

Procedimiento

El procedimiento de aplicación del módulo difuso de ubicación puede realizarse de la siguiente manera:

1. Conversión de variables nítidas a Difusas.
2. Aplicación de reglas de inferencia.
3. Conversión de variables difusas a valores nítidos de salida.
4. Ubicación de la tarea.

6.4.2. Tipos de tareas analizadas

Un aspecto importante que se puede poner a prueba en las reglas planteadas en esta propuesta es considerar las tareas que se analizan, en esta propuesta se proponen tareas que no tienen dependencias entre sí, lo cual se puede extender con tareas dependientes de información entre si, considerando siempre un ambiente controlado en el que se pueda estimar la carga que cada una de esas tareas representa en los nodos involucrados en ejecutarlas.

6.4.3. Manejo de Comunicaciones.

Se puede extender el uso de las comunicaciones y las consideraciones de los retardos en el envío de carga o en el envío de datos para la ejecución de nuevas tareas en el sistema.

6.4.4. Implementación en Hardware

Finalmente un aspecto que puede ser interesante basado en la propuesta planteada, es implementarla físicamente en un sistema construido físicamente ya que los aspectos evaluados en la evaluación experimental se realizaron por medio de una simulación, la cual brinda elementos válidos para construir el sistema, y que dado que en un ambiente simulado el algoritmo logró su objetivo, ahora se puede proceder a analizar si dicho objetivo se logra en un sistema de Hardware.

Bibliografía

[1] A. Chhabra y G. Singh. Qualitative parametric comparison of load balancing algorithms in distributed computing environment. En *Advanced Computing and Communications, 2006. ADCOM 2006. International Conference on*, págs. 58–61. 2006.

[2] George E. Coulouris. *Distributed Systems. Concepts and Design.* Addison-Wesley, 2012.

[3] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Parallel programming / scientific computing. ADDISON WESLEY Publishing Company Incorporated, 1995. URL http://books.google.com.mx/books?id=r5JsQgAACAAJ.

[4] TV Gopal, NS Nataraj, C Ramamurthy, y V Sankaranarayanan. Load balancing in heterogenous distributed systems. *Microelectronics Reliability*, 36(9):1279–1286, 1996.

[5] Ming-Chang Huang y S Hossein. Load balancing in computer networks. *International Society for Computers and Their Applications ISCA*, 227, 2002.

[6] Asha Kalyur. Process migration in sparta. proposal for master's writing project.

[7] George J. Klir y Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[8] V Kun-Ming, Y Chou, y Y Wang. A fuzzy-based dynamic load-balancing algorithm. *Journal of Information, Technology and Society*, 4(2):55–63, 2004.

[9] M.A. Mehta y D.C. Jinwala. Analysis of significant components for designing an effective dynamic load balancing algorithm in distributed systems. En *In-*

telligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on, págs. 531–536. Feb. 2009.

[10] Martin Ohlin, Dan Henriksson, y Anton Cervin. Truetime 1.5 reference manual. *Department of Automatic Control, Lund University, Sweden*, 2007.

[11] Chulhye Park y J.G. Kuhl. A fuzzy-based distributed load balancing algorithm for large distributed systems. En *Autonomous Decentralized Systems, 1995. Proceedings. ISADS 95., Second International Symposium on*, págs. 266–273. 1995.

[12] W. Pedrycz. *Fuzzy control and fuzzy systems*. Research Studies Press, 1993.

[13] H. Rahmawan y Y.S. Gondokaryono. The simulation of static load balancing algorithms. En *Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference on*, tomo 02, págs. 640–645. 2009.

[14] S. Saxena, M.Z. Khan, y R. Singh. Performance analysis in distributed system of dynamic load balancing using fuzzy logic. En *Engineering and Technology (S-CET), 2012 Spring Congress on*, págs. 1–5. May 2012.

[15] Andrew S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall, 1994. ISBN 0132199084.

[16] PaulP. Wang, Da Ruan, y EtienneE. Kerre. Why fuzzy logic? – a spectrum of theoretical and pragmatics issues. En *Fuzzy Logic*, tomo 215 de *Studies in Fuzziness and Soft Computing*, págs. 1–13. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-71257-2.

[17] Lotfi Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.