UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

DIRECCIÓN DE POSGRADO

MAESTRÍA EN REDES Y TELECOMUNICACIONES

Tema:

"EL USO DE BUENAS PRÁCTICAS DE PROGRAMACIÓN EN EL ALGORITMO DE CONSENSO Y SU INCIDENCIA EN EL CONSUMO DE RECURSOS DE HARDWARE Y RED PARA EL PROYECTO CLOUD-CEDIA DE LA UNIVERSIDAD TÉCNICA DE AMBATO"

Trabajo de Titulación

Previa a la obtención del Grado Académico de Magíster en Redes y Telecomunicaciones

Autor: Ingeniero Rubén Eduardo Nogales Portero
Director: Ingeniero Clay Fernando Aldás Flores Magister.

Ambato - Ecuador

2015

Al Consejo de Posgrado de la Universidad Técnica de Ambato.

El Tribunal de Defensa del trabajo de titulación presidido por el Ingeniero José Vicente Morales Lozada, Magister presidente del tribunal e integrado por los señores: Ingeniero Clay Fernando Aldás Flores Magister, Ingeniero Edgar Patricio Córdova Córdova Magister, Ingeniero Edwin Rodrigo Morales Perrazo Magister, Ingeniero Juan Pablo Pallo Noroña Magister, Ingeniero Carlos Humberto Sánchez Rosero Magister, miembros del tribunal de defensa, designados por Consejo Académico de Posgrado de la Facultad de Ingeniería en Sistemas Electrónica e Industrial, de la Universidad Técnica de Ambato, para receptar la defensa oral del trabajo de titulación con el tema: "EL USO DE BUENAS PRÁCTICAS DE PROGRAMACIÓN EN EL ALGORITMO DE CONSENSO Y SU INCIDENCIA EN EL CONSUMO DE RECURSOS DE HARDWARE Y RED PARA EL PROYECTO CLOUD-CEDIA DE LA UNIVERSIDAD TÉCNICA DE AMBATO", elaborado y presentado por el señor Ingeniero Rubén Eduardo Nogales Portero, para optar por el Grado Académico de Magister en Redes y Telecomunicaciones.

Una vez escuchada la defensa oral el Tribunal aprueba y remite el trabajo de titulación para el uso y custodia en las bibliotecas de la UTA.

Ing. José Vicente Morales Lozada, Mg

Presidente del Tribunal de Defensa

Ing. Edgar Patricio Córdova Córdova, Mg.

Miembro del Tribunal de Defensa

Ing. Edwin Rodrigo Morales Perrazo, Mg.

Miembro del Tribunal de Defensa

Ing. Juan Pablo Pallo Noroña, Mg.

Miembro del Tribunal de Defensa

AUTORÍA DE LA INVESTIGACIÓN

La responsabilidad de las opiniones, comentarios y críticas emitidas en el Trabajo de Titulación con el tema: "EL USO DE BUENAS PRÁCTICAS DE PROGRAMACIÓN EN EL ALGORITMO DE CONSENSO Y SU INCIDENCIA EN EL CONSUMO DE RECURSOS DE HARDWARE Y RED PARA EL PROYECTO CLOUD-CEDIA DE LA UNIVERSIDAD TÉCNICA DE AMBATO", le corresponde exclusivamente a: Ingeniero Rubén Eduardo Nogales Portero, Autor y bajo la Dirección del Ingeniero Clay Fernando Aldás Flores Magister, Director del trabajo de titulación; y el patrimonio intelectual a la Universidad Técnica de Ambato.

Ing. Rubén Eduardo Nogales Portero
Autor

Ing. Clay Fernando Aldás Flores, Mg
Director

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este trabajo de titulación como un documento disponible para su lectura, consulta y procesos de investigación.

Cedo los derechos de mi trabajo de titulación, con fines de difusión pública, además autorizo su reproducción, dentro de las regulaciones de la Universidad.

Ing. Rubén Eduardo Nogales Portero

CC: 1802668606

DEDICATORIA

Dedico este trabajo a toda mi familia en especial a mis hijos Luis Eduardo e Ibeth Analía, que lleven en su recuerdo que no importan los obstáculos que se presenten, cuan grandes sean las responsabilidades otorgadas por DIOS pero si ponemos esfuerzo, dedicación, constancia y perseverancia tarde o temprano se llega y se consigue la meta.

A mi esposa que con su paciencia y su carácter me ha acompañado todos los días de este arduo caminar.

Ing. Rubén Nogales Portero.

AGRADECIMIENTO

Mi agradecimiento primario es para DIOS quien ha hecho posible que los actos y circunstancias se vayan dando en mi vida conforme a su plan.

A mis padres, a mi hermana y a mi cuñado quienes han sabido afrontar con toda responsabilidad las labores de la fábrica mientras yo concluía con el trabajo de la tesis.

A la Universidad Técnica de Ambato, a sus docentes y en especial al director del proyecto Ingeniero David Guevara Magister quien a estado en todo momento brindando su apoyo y sus conocimientos.

Ing. Rubén Eduardo Nogales Portero.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

DIRECCIÓN DE POSGRADO

MAESTRÍA EN REDES Y TELECOMUNICACIONES

Tema: "EL USO DE BUENAS PRÁCTICAS DE PROGRAMACIÓN EN EL ALGORITMO DE CONSENSO Y SU INCIDENCIA EN EL CONSUMO DE RECURSOS DE HARDWARE Y RED PARA EL PROYECTO CLOUD-CEDIA DE LA UNIVERSIDAD TÉCNICA DE AMBATO".

Autor: Ing. Rubén Eduardo Nogales Portero.

Director: Ing. Clay Fernando Aldás Flores Mg.

Fecha: 14 de enero del 2015

RESUMEN EJECUTIVO

En el presente trabajo investigativo se pretende demostrar la incidencia del uso de buenas técnicas de programación en el algoritmo de consenso propuesto en el proyecto de investigación "SERVICIOS DE COORDINACIÓN EN LA NUBE CUANDO LOS ELEMENTOS INTERVINIENTES SON ANÓNIMOS" (CEDIA: CEPRA VII-2013-05) (Cloud-CEDIA).

La evaluación consistirá en medir el rendimiento de los recursos de memoria, CPU, disco y Red inicialmente programado en una versión lineal.

Este trabajo estará evaluado en tres sistemas operativos diferentes Fedora, Ubuntu y Centos, utilizando el lenguaje de programación C Sharp en mono developer.

Así mismo se simularán secuencias sucesivas de conexiones con tiempos de retardo entre procesos con el fin de estresar los sistemas.

Una vez estudiada y evaluada esta propuesta se procederá con la programación de una nueva versión del algoritmo de consenso utilizando técnicas de programación bajo el mismo lenguaje de programación (C Sharp en mono deloper) y en los mismos sistemas operativos (Fedora, Ubuntu y Centos). Evaluando de esta manera el rendimiento de los recursos de memoria, CPU, disco y red en las diferentes versiones de programación del algoritmo de consenso y en los distintos sistemas operativos.

Descriptores: Algoritmo, Consenso, C sharp, Lenguajes de Programación, Monodeveloper, Recursos, Rendimiento, Sistemas Operativos, Técnicas de Programación.

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

DIRECCIÓN DE POSGRADO

MAESTRÍA EN REDES Y TELECOMUNICACIONES

Theme: "USE OF PRACTICE PROGRAMMING ALGORITHM OF AGREEMENT AND ITS IMPACT ON THE USE OF RESOURCES FOR HARDWARE AND RED CLOUD-CEDIA PROJECT TECHNICAL UNIVERSITY OF AMBATO".

Author: Ing. Rubén Eduardo Nogales Portero.

Directed by: Ing. Clay Fernando Aldás Flores Mg.

Date: January 14th, 2015

EXECUTIVE SUMMARY

In this research work pretend to show the impact of the use of good programming techniques in the consensus algorithm proposed in the research project "SERVICE COORDINATION IN THE CLOUD WHEN THE ELEMENTS INVOLVED ARE ANONYMOUS" (CEDIA: CEPRA VII-2013- 05) (Cloud-CEDIA).

The test will measure the performance of memory resources, CPU, disk and network initially set into a linear version.

This work will be evaluated in three different operating systems Fedora, Ubuntu and Centos, using the C Sharp programming language in mono deloper.

Likewise successive connections sequences will be simulated whith delay times between processes in order to stress the system.

Having studied and evaluated this proposal will be programed a new version of the consensus algorithm using programming techniques under one programming language (C Sharp in mono deloper) and in the same operating systems (Fedora, Ubuntu and CentOS). Evaluating the performance of memory resources, CPU, disk and network in the different versions of programming algorithm consensus and on different operating systems.

Descriptors: Algorithm, Consensus, C Sharp, Programming Languages, Monodeveloper, Resources, Performance, Operating Systems, Programing Techniques.

Introducción.

A partir de la década de los 80's la computación va haciéndose cada vez mas parte de la vida cotidiana de las personas en su vida diaria, en el trabajo, en el hogar y de la mano con esta creciente necesidad a ido el desarrollo de software.

Debido a esto los desarrolladores de software se han visto en la urgente necesidad de implementar técnicas de programación que optimicen los recursos disponibles tanto en software como en hardware, pasando desde programaciones lineales a complejas técnicas como la programación orientada a objetos, programación orientada a aspectos con Modelos de programación como el modelo vista controlador (MVC), creando así complejos sistemas.

En la actualidad las empresas están migrando su software de control (programas, sistemas) de modelos cliente servidor a modelos MVC Web, es decir que la información que esta circulando está en expuesta en la red de redes (internet).

De aquí radica el nuevo reto que los programadores están enfrentando y es optimizar los recursos de red en la computación distribuida y sabiendo que el núcleo de la computación distribuida son los detectores de fallos y el consenso siendo el consenso un servicio que permite acordar un mismo valor entre todos los equipos intervinientes en la computación distribuida (Nube).

Y como se trata de optimización de recursos las grandes corporaciones en la actualidad están hablando de computación en la nube conocida como Cloud Computing, que no es otra cosa mas que acceder a los recursos en forma de demanda como IaaS infraestructura de servicios orientado a administradores de red, PaaS plataforma de servicios orientado a desarrolladores, SaaS software como servicios orientado a usuarios.

De tal forma que el capítulo I de la investigación enmarca el problema de no utilizar buenas técnicas de programación sus causas y sus efectos, la justificación de la investigación y sus objetivos.

El Capítulo II describe el marco teórico en relación a las variables de investigación, su fundamentación tanto filosófica como legal y la hipótesis.

El capítulo III explica la metodología y hace una breve descripción de la modalidad de investigación a utilizar, pasando por la operacionalización de las variables y las técnicas y métodos de recolección de datos.

En el capítulo IV se programa la primera versión del algoritmo de consenso, se preparan los servidores, se ejecuta y se toman datos con tiempos de retardo entre consensos de 0 y 5 seg., datos que son validos para verificar la hipótesis.

El capítulo V tiene las conclusiones y recomendaciones.

El capítulo VI contiene el desarrollo de la propuesta investigativa.

CAPÍTULO I

EL PROBLEMA

1.1. TEMA DE INVESTIGACIÓN

"El uso de buenas prácticas de programación en el algoritmo de consenso y su incidencia en el consumo de recursos de hardware y red para el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato"

1.2. PLANTEAMIENTO DEL PROBLEMA

1.2.1. Contextualización

1.2.1.1. Contexto macro

Cloud computing (computación en la nube) es una plataforma altamente escalable y de muy fácil acceso para usuarios que utilicen dicho servicio, existen tres tipos de nubes: Privada, pública e híbrida, incluso se puede hablar de un cuarto tipo y es cuando una *cloud* (nube) privada utiliza una nube pública y se denomina nube virtual (virtualización). Las *clouds* (nubes) se apoyan en tecnologías como la virtualización, en técnicas de programación como *multitenancy* (es una arquitectura de software donde una instancia de un servidor corre en un servidor) y escalabilidad.

1.2.1.2. Contexto meso

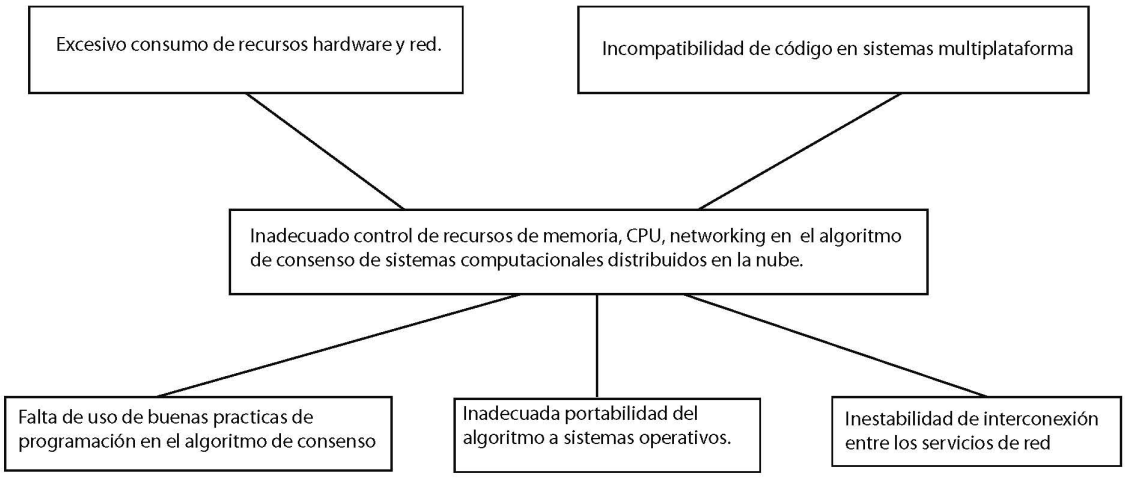
Es necesario hablar de recursos y servicios conectados por medio de una red de datos (sistemas distribuidos), desde el punto de vista del usuario un sistema distribuido proporciona una visión de maquina única. Mientras que desde el punto de vista del diseñador la programación está condicionada a la disposición física de los recursos.

1.2.1.3. Contexto micro

En el proyecto "SERVICIOS DE COORDINACIÓN EN LA NUBE CUANDO LOS ELEMENTOS INTERVINIENTES SON ANÓNIMOS" (CEDIA: CEPRA VII-2013-05) (Cloud-CEDIA) hablar de consumo de recursos es ingresar a analizar, entender y revisar el algoritmo de consenso, el mismo que permite a los procesos llegar a una decisión común a partir de valores iniciales a pesar de fallos.

1.2.2. Análisis crítico

1.2.2.1. Árbol de problemas



Árbol de problemas

Fuente: Investigación de campo

Elaborado por: Rubén Nogales Portero

1.2.2.2. Relación causa-efecto

Para el caso de la investigación se hace referencia al proyecto de Cloud-CEDIA. Consorcio Ecuatoriano para el Desarrollo de Internet Avanzado (CEDIA) en el mismo que intervienen tres universidades del país, Universidad Técnica Particular de Loja (UTPL), Universidad Técnica de Ambato (UTA), Escuela Politécnica Nacional (EPN), las que trabajan en solucionar el liderazgo de servicios de coordinación en la nube a través de la detección de fallos y consenso, para sistemas distribuidos asíncronos anónimos u homónimos, sabiendo que la coordinación distribuida de

la nube es un aspecto muy importante en los servicios que se ofertan a través del Internet. Los principales avances y retos a resolver dentro de la coordinación distribuida en la nube son: el consenso, el acuerdo en múltiples valores y detectores de fallos, en sistemas asíncronos anónimos u homónimos. Si consideramos el momento que en la nube tenemos una caída de un servidor (proveedor de servicios) o un enlace en un sistema asíncrono anónimo debe primero detectar el fallo y posterior acordar el valor de levantamiento o re direccionamiento de solicitudes de información de datos. En el instante en que esto sucede, el resto de servidores distribuidos en un sistema asíncrono intenta ponerse de acuerdo en el re direccionamiento de información, en este instante se puede evidenciar un inadecuado control de recursos de memoria, Unidad Central de Procesamiento (CPU), *networking* (red de trabajo) en el algoritmo de consenso de sistemas computacionales distribuidos en la nube debido a la falta y uso de buenas prácticas de programación ocasionando un excesivo consumo de recursos hardware y red.

1.2.3. Prognosis

En el caso dado de que los desarrolladores de software no tomen en cuenta las buenas prácticas de programación para el algoritmo de consenso y se incrementen los elementos que participen en la nube (usuarios, nodos, dispositivos) esta podrá generar retardos excesivos, perdidas de información implicando un uso desmedido de los recursos disponibles (*hardware*, *software* y red).

1.2.4. Formulación del problema

¿Es el inadecuado uso de buenas prácticas de programación en el algoritmo de consenso lo que conlleva a un inadecuado control de recursos de memoria, CPU, *networking* (trabajo en red) en el algoritmo de consenso de sistemas computacionales distribuidos en la nube lo que provocan un excesivo consumo de recursos de *hardware* y red en el segundo semestre del año 2014 para el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato?

1.2.5. Interrogantes

- ¿Cómo mantener un adecuado control de recursos de memoria, CPU y *networking* (trabajo en red) en el algoritmo de consenso en los sistemas computacionales distribuidos en la nube?

- ¿Cómo implementar buenas prácticas de programación?
- ¿Por qué se puede deducir que existe un excesivo consumo de recursos de hardware y red?

1.2.6. Delimitación del objeto de investigación

- Campo: Redes y Telecomunicaciones I edición en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.
- Área: Redes de computadores, tecnología de redes, arquitectura TCP/IP.
- Aspecto: Control de recursos en cuanto a memoria, CPU y *networking* (trabajo en red) se refiere en el algoritmo de consenso en los sistemas computacionales distribuidos en la nube.
- Temporal: La investigación debe culminar en un plazo de seis meses es decir en el segundo semestre del año 2014.
- Espacial: (ver Anexo 2) La investigación se desarrolla en el departamento de investigación de la unidad operativa de investigación y desarrollo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, en el proyecto "SERVICIOS DE COORDINACIÓN EN LA NUBE CUANDO LOS ELEMENTOS INTERVINIENTES SON ANÓNIMOS" (CEDIA: CEPRA VII-2013-05).

1.3. JUSTIFICACIÓN

Cloud computing (computación en la nube) es una tecnología que está revolucionando las técnicas de la información, siendo una *cloud* (nube) una abstracción de computación paralela, computación distribuida y mallas de computadoras, se estima que el computación en la nube cause una revolución y al ser el corazón de una nube la detección de fallos y el consenso es necesario profundizar el estudio de la misma. Como referencia la computación en la nube provee básicamente tres tipos de servicios: Servicios de infraestructura (IaaS), servicios de plataforma (PaaS) y servicios de software (SaaS).

El consenso es un servicio que permite acordar un mismo valor entre todos los equipos de un sistema distribuido propenso a fallos y/o caídas. Uno de los retos a resolver actualmente es diseñar nuevos protocolos para acordar valores comunes

que permitan sincronizar los recursos que están utilizando de forma distribuida los equipos pese a estar trabajando en la nube, proveyendo a los usuarios de la nube un servicio calidad.

1.4. OBJETIVOS

1.4.1. General

- Proponer buenas prácticas de programación en el algoritmo de consenso y su incidencia en el consumo de recursos de hardware, software y red en sistemas computacionales distribuidos en la nube para el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato.

1.4.2. Específicos

- Establecer el uso de buenas prácticas de programación para la optimización de recursos de memoria, CPU, *networking* (trabajo en red) en el algoritmo de consenso de sistemas computacionales distribuidos en la nube.
- Demostrar el consumo de recursos hardware y red en el algoritmo de consenso de sistemas computacionales distribuidos en la nube.
- Proponer la creación de métodos para un adecuado control de recursos de memoria, CPU y *networking* (trabajo en red) en Sistemas Asíncronos (AS).

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES INVESTIGATIVOS

A medida que la tecnología ha ido avanzando los lenguajes y técnicas de programación también han evolucionado, en la década de los años 50 la programación en sus inicios debía ser individualizada para cada *hardware* un programa especifico, hasta que en la década de los años 60 se crean los primeros lenguajes de programación denominados **Van Emden (1997, :Internet)** "FORTRAN, COBOL y ALGOL" que básicamente era una programación estructurada, posteriormente aparece un nuevo paradigma de programación con C++ que fue perfectamente adecuado al flujo de datos "técnicas de análisis y diseño estructurado" (*Structured Analysis and Design Technique*) y como se menciono anteriormente los avances tecnológicos han sido gigantescos y los desarrolladores y las empresas necesitaron que el *software* sea escalable, portable y reutilizable hasta que en los años 2000 y hasta la fecha (2014) se viene desarrollando la programación orientada a objetos (OOP), dicha programación se refiere a tres elementos como menciona **Omar and Razik (2008, :Internet)** "clases, variables/atributos y funciones/métodos", es necesario que los desarrolladores de OOP utilicen técnicas de procesamiento de lenguaje natural que analizan los documentos con requerimientos del sistema este documento fue escrito en ingles "*Object Oriented Conceptual Model*" donde se mencionan las clases, atributos y relaciones del sistema. Existen técnicas de OOP como LIDA, GOOAL, UML siendo este ultimo el mas usado.

A la par en la evolución del *hardware* y *software* están las redes de datos que también han incorporado trascendentales cambios en las tecnologías de la información llegando a dominar el concepto de información en la nube; **Kalagiakos and Karampelas (2011, :Internet)** *"Cloud Computing* se está desarrollando como una tecnología clave para los recursos compartidos. Tecnologías como *grid computing*,

computación distribuida, y virtualización en paralelo definen las formas de una nueva era".

Al ser la nube un desarrollo de computación paralela, computación distribuida y una malla de redes, esto añadido a la combinación y a la evolución de la virtualización tenemos tres tipos de servicios en la nube como se menciona en **Peng et al. (2009, :Internet)** "*Infrastructure-as-a-Service* (IaaS), *Platform-as-a- Service* (PaaS) and *Software-as-a-Service* (SaaS)", donde SaaS son las aplicaciones que se ejecutan en la nube solicitados por los usuarios a los proveedores de este servicio, el modo de acceso puede ser por browser, mientras que PaaS hace referencia al acceso a las aplicaciones creadas por desarrolladores ayudados por herramientas de desarrollo como JAVA, C# , etc. proporcionada por los proveedores de servicios de la nube, y el IaaS no es otra cosa que la renta de almacenamiento, red y procesamiento de datos.

2.2. FUNDAMENTACIÓN FILOSÓFICA

La presente investigación sera guiada por el enfoque paradigmatico positivista, llamado también paradigma cuantitativo, el mismo que se basa en hechos y leyes previamente investigados siendo estos indiferentes a los individuos, este tipo de investigación tiene como finalidad predecir fenómenos y verificar teorías.

Una breve definición propia de paradigma es la forma individualizada de ver y presentar el fenómeno que se persive o le presentan.

2.3. FUNDAMENTACIÓN LEGAL

En la Constitución Política de la República del Ecuador **LEY DE COMERCIO ELECTRONICO (2011, :Internet)**, Registro Oficial 557, del 17 de abril del 2002, última modificación 13 de octubre del 2011. En la ley de comercio electrónico, firmas electrónicas y mensajes de datos, capítulo 1, principios generales.

> Art. 4.- Propiedad intelectual.- Los mensajes de datos estarán sometidos a las leyes, reglamentos y acuerdos internacionales relativos a la propiedad intelectual.
>
> Art. 5.- Confidencialidad y reserva.- Se establecen los principios de confidencialidad y reserva para los mensajes de datos, cualquiera sea su forma, medio o intención. Toda violación a estos principios, principalmente aquellas referidas a la intrusión electrónica, transferencia ilegal de mensajes de datos o violación del secreto profesional, será

sancionada conforme a lo dispuesto en esta ley y demás normas que rigen la materia.

2.4. CATEGORÍAS FUNDAMENTALES

2.4.1. Visión dialéctica de conceptualizaciones que sustentan las variables del problema

2.4.1.1. Marco conceptual variable independiente

Como se menciona en el documento *Object-oriented programming as the end of history in programming languages* de **Van Emden (1997, :Internet)** el Asembler fue el primer lenguaje de programación orientado a máquina, posterior a esto aparecen los lenguajes orientado a procedimientos "*procedure-oriented*" que son llamados lenguajes de alto nivel, los principales paradigmas de estos lenguajes son: programación funcional, programación de flujo de datos y programación lógica.

Es necesario mencionar y describir la programación orientada a objetos (OOP) como el paradigma más estable que hasta el 2014 se viene usando a nivel académico e industrial, los principales elementos utilizados en la OOP son:

- Unidades u Objetos.
- Atributos.
- Métodos.
- Herencia.
- Clases.

Según el documento de **Omar and Razik (2008, :Internet)** el principal aspecto de la OOP es la construcción de grupos de unidades u objetos, los que deben estar habilitados para enviar y recibir mensajes a otros objetos, cada objeto tiene características (atributos) propias que lo definen y métodos o mensajes que no es más que el código de programación que describe el comportamiento de los objetos, basado en esto la OOP define la herencia como otro de los principales conceptos y como explicación de este concepto se visualiza la manipulación de las características propios, manteniendo las características intactas del objeto que recibe estas, y este tiene la posibilidad de enviar las características heredadas y propias hacia otro objeto, el mismo que llegara a tener todas las características de los objetos que recibe

el mensaje, otro concepto es la encapsulación de objetos similares y la agrupación de estos en clases.

Se debe mencionar también el nuevo paradigma de programación, la programación orientada a aspectos (AOP) como una abstracción de la OOP, la arquitectura de la AOP es diferente puesto que requiere un lenguaje de programación orientado a aspectos (AOPL), y un tejedor que descifre y concatene el código fuente nativo y los aspectos.

El lenguaje de programación que ha evolucionado junto a la informática es el lenguaje C, a C++, C# y C .NET. por tanto el algoritmo de consenso puede ser programado en una de estas variantes.

En el documento *Generic Construction of Consensus Algorithms for Benign and Byzantine Faults* de **Rutti et al. (2010, :Internet)** El consenso es el problema fundamental en la computación distribuida tolerante a fallos, esto explica los distintos algoritmos de consenso publicados con diferentes características y para diferentes modelos.

Los algoritmos pueden estar basados en líderes o en líderes libres basados en parámetros instanciados que permiten tener varios algoritmos, estos están compuestos de fases sucesivas, donde cada fase contiene tres rondas.

- Ronda de selección.
- Ronda de validación.
- Ronda de desición.

Para el algoritmo basado en líderes la primera acción es seleccionar el líder, los miembros de la red validan el líder propuesto, lo deciden y lo dan a conocer a todos los miembros de la red.

2.4.1.2. Marco conceptual variable dependiente

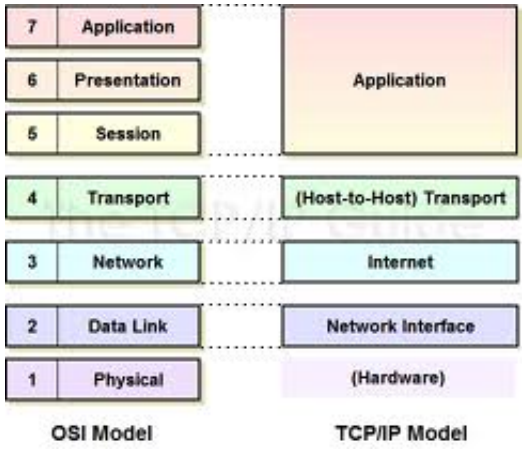
Las redes de datos fueron creadas por la necesidad de compartir información entre miembros de una misma organización, la evolución de las redes se visualiza por sus topologías como:

- Jerárquica.
- Bus.

- Anillo.
- Estrella.
- Malla.

La topología jerárquica consiste en niveles, la de mayor jerarquía controla la red, mientras que la de bus tiene un solo canal de conexión al que se adjuntan todos los miembros de la red, la topología de anillo también tiene un mismo canal de conexión y adiciona repetidores de señal en cada nodo (miembro de red o computador) y su distribución física es en círculo, la distribución en estrella se da con la incorporación de todos los nodos a un concentrador o *HUB* y este controla el acceso al medio físico, y la topología de malla tiene diferentes caminos para el traslado de información es una comunicación fiable y tolerante a fallos puesto que si una ruta deja de funcionar los datos se redireccionan, el núcleo de las redes son los protocolos.

Según **Nino et al. (2013, :Internet)**"el protocolo que se ha implementado de manera gradual con el crecimiento de las redes de datos como Internet es TCP (*Transport Control Protocol*); sobre este protocolo se han diseñado varias versiones que mejoran el funcionamiento de TCP". En la siguiente figura se observa los modelos de transmisión de datos.



Fuente: Internet arquitectura del modelo de transmisión.
Autor: Internet.

Figura 1: Modelo de transmisión de datos

Haciendo un compendio de las variables que intervienen en el comportamiento de las redes de datos se encuentran los recursos, el desempeño y los usuarios. El comportamiento se define como se relacionan estas variables; entre más usuarios tiene una red más recursos necesita y esto impacta directamente el desempeño. Entonces una forma

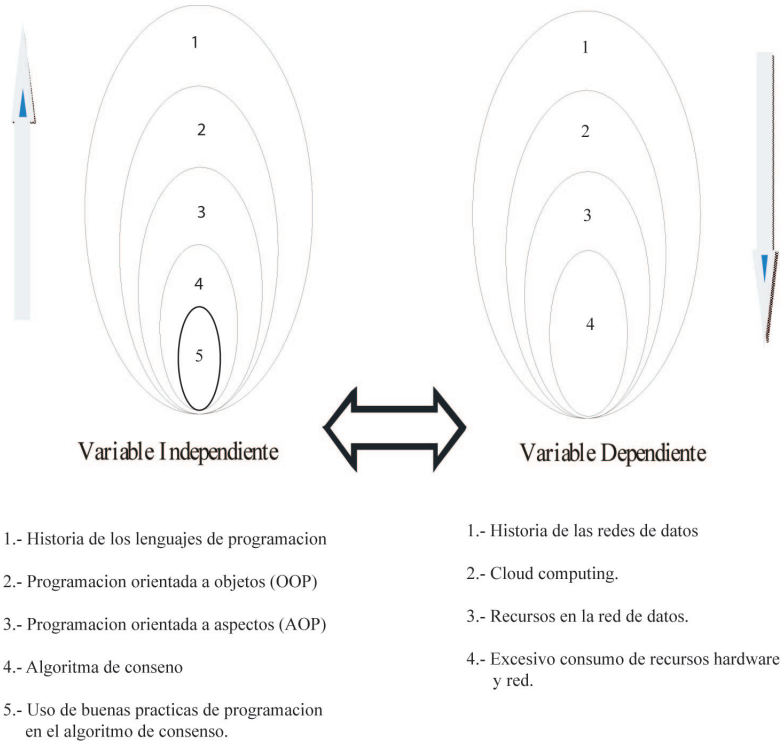
de combatir el bajo desempeño de una red que se traduce en congestión es hacer un dimensionamiento adecuado de los recursos disponibles.

La base de *Cloud Computing* es la topologia en malla con la variante computación distribuida y el protocolo TCP/IP. *Cloud Computing* como nuevo modelo de negocios enfrenta los retos de seguridad, disponibilidad, privacidad ofreciendo transparencia para el usuario de los servicios ofrecidos por las empresas proveedoras, manteniendo estándares de calidad y cumpliendo con políticas y procedimientos legales, las empresas ofrecen servicios como:

- *Infrastructure-as-a-Service* (IaaS).
- *Platform-as-a- Service* (PaaS).
- *Software-as-a-Service* (SaaS).

2.4.2. Gráficos de inclusión interrelacionados

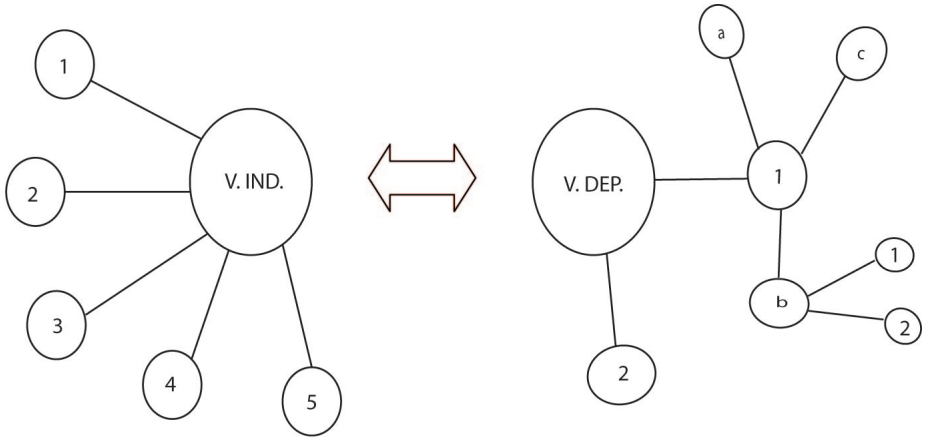
- Superordinación conceptual



Fuente: Variables de investigación / marco teórico
Elaborado: Ruben Nogales Portero

Figura 2: Superordinación conceptual

- Subordinación conceptual



1.- Programacion orientada a objetos. (OOP)

2.- Programacion orientada a aspectos (AOP)

3.- Programacion de flujo de datos.

4.- Programacion lineal.

4.- Algoritma de conseno

1.- Recursos.

1.a.- Hardware.

1.b.- Red.
1.b.1- Topologia
1.b.2- Modelos de transporte

1.c.- CPU.

2.- Cloud Computing.

Fuente: Variables de investigación / marco teórico
Elaborado: Ruben Nogales Portero.

Figura 3: Subordinación conceptual.

2.5. HIPÓTESIS

El inadecuado uso de buenas prácticas de programación en el algoritmo de consenso es la causa de un excesivo consumo de recursos hardware y red en el proyecto Cloud-CEDIA de la UTA.

2.6. SEÑALAMIENTO VARIABLES DE LA HIPÓTESIS

- **Variable independiente**: Buenas prácticas de programación en el algoritmo de consenso
- **Variable dependiente**: Excesivo consumo de recursos
- **Unidad de observación**: UTA proyecto Cloud-CEDIA

- **Términos de relación**: Excesivo consumo de recursos a causa de un inadecuado uso de buenas prácticas de programación.

CAPÍTULO III

METODOLOGÍA

3.1. ENFOQUE

Según **Pita Fernández S (2002, :Internet)** "La investigación cuantitativa es aquella en la que se recogen y analizan datos cuantitativos sobre variables" mientras que "la investigación cualitativa evita la cuantificación" y en este trabajo investigativo el método cuantitativo sera el predominante debido a que "la diferencia fundamental entre ambas metodologías es que la cuantitativa estudia la asociación o relación entre variables cuantificadas y la cualitativa lo hace en contextos estructurales y situacionales"

3.2. MODALIDAD BÁSICA DE LA INVESTIGACIÓN

3.2.1. Investigación de campo

Según **Franco (2011, :Internet)** investigación de campo es:

> El análisis sistemático de problemas en la realidad, con el propósito bien sea de describirlos, interpretarlos, entender su naturaleza y factores constituyentes, explicar sus causas y efectos, o predecir su ocurrencia, haciendo uso de métodos característicos de cualquiera de los paradigmas o enfoques de investigación conocidos o en desarrollo. Los datos de interés son recogidos en forma directa de la realidad; en este sentido se trata de investigaciones a partir de datos originales o primarios. Sin embargo, se aceptan también estudios sobre datos censales o muestrales no recogidos por el estudiante, siempre y cuando se utilicen los registros originales con los datos no agregados; o cuando se trate de estudios que impliquen la construcción o uso de series históricas y, en general, la recolección y organización de datos publicados para su análisis mediante procedimientos estadísticos, modelos matemáticos, econométricos o de otro tipo.

Aplicable en el proyecto Cloud de la Universidad Técnica de Ambato auspiciado por CEDIA.

3.2.2. Investigación bibliográfica documental

Es una secuencia lógica de actividades conducentes a la obtención de información necesaria para profundizar conocimientos a partir de información bibliográfica.

En este tipo de investigación se utilizan todas las investigaciones previamente realizadas en relación a la programación, a las técnicas que se utilizan, documentación y teoría sobre el algoritmo de consenso y los recursos que se consumen en la tecnología de cloud.

3.2.3. Experimental

La investigación experimental consiste en la manipulación de una variable experimental no comprobada, en condiciones rigurosamente controladas, con el fin de describir de qué modo o por qué causa se produce una situación o acontecimiento en particular.

Se trata de un experimento porque precisamente el investigador provoca una situación para introducir determinadas variables de estudio manipuladas por él, para controlar el aumento o disminución de esa variable, y su efecto en las conductas observadas. El investigador maneja deliberadamente la variable experimental y luego observa lo que sucede en situaciones controladas.

Para el caso del presente proyecto de investigación se simula una cloud entre las universidades integrantes del proyecto y también se simula un ambiente virtual.

3.3. NIVEL O TIPO DE INVESTIGACIÓN

3.3.1. Investigación exploratoria

Según **Roberto et al. (2010, :Internet)** la investigación exploratoria son:

> Los estudios exploratorios se efectúan, normalmente, cuando el objetivo es examinar un tema o problema de investigación poco estudiado o que no ha sido abordado antes. Es decir, cuando la revisión de la literatura reveló que únicamente hay guías no investigadas e ideas vagamente relacionadas con el problema de estudio.

El caso de investigación poco explorado es el algoritmo de consenso, como parte medular del nuevo paradigma de comunicación cloud.

3.3.2. Investigación descriptiva

El investigador necesita describir el fenómeno que investiga y evaluarlo (medir), como menciona **Roberto et al. (2010, :Internet)**"Los estudios descriptivos buscan especificar las propiedades importantes de personas, grupos, comunidades o cualquier otro fenómeno que sea sometido a análisis" adicional "en un estudio descriptivo se selecciona una serie de cuestiones y se mide cada una de ellas independientemente" una vez obtenido el resultado el investigar puede juntar los resultados para poder observar al fenómeno como es.

Es necesario implementar simulaciones del algoritmo de consenso en sus diferentes versiones, tomar los datos y poder compararlos.

3.3.3. Investigación asociación de variables (correlacional)

Según **Roberto et al. (2010, :Internet)** "este tipo de estudios tienen como propósito medir el grado de relación que exista entre dos o más conceptos o variables" las variables medidas son del mismo fenómeno.

Se medirá la incidencia de la programación de las versiones del algoritmo de consenso en el consumo de los recursos.

3.3.4. Investigación explicativa

Tomado del documento de **Sabino (1992, :Internet)**.

> Son aquellos trabajos donde nuestra preocupación se centra en determinar los orígenes o las causas de un determinado conjunto de fenómenos. Su objetivo, por lo tanto, es conocer por qué suceden ciertos hechos, analizando las relaciones causales existentes o, al menos, las condiciones en que ellos se producen.

En un proceso investigativo el actor periódicamente o temporalmente podrá irse ubicando en cualquier etapa de la investigación, el caso de este proceso se centra mayoritariamente en esta etapa, Se espera medir resultados causados por la simulación.

3.4. POBLACIÓN Y MUESTRA

En la investigación experimental el investigador selecciona o desarrolla la manera de como responder ante el fenómeno en estudio, respondiendo ante las interrogantes planteadas y tratando de demostrar con certeza la hipótesis.

En una experimentación al no tener población no se tiene una muestra, por tanto en este tipo de investigación se trabaja con variables, la variable independiente y la dependiente en relación al problema son estáticas, no son manipulables, pero para efectos de medición de resultados es necesario introducir variables adicionales en relación al mismo fenómeno u objeto de estudio, estas últimas si son manipulables.

En la investigación se utilizará:

- Virtualización de equipos en una nube.
  - Sistema Cloud computing
- Sistemas operativos GNU/Linux.
  - Centos server 6.5.
  - Fedora 20.
  - Ubuntu server 14.
- Utilización de recursos de Red.
- Utilización de recursos de Memoria.
- Utilización de recursos de CPU.
- Implementación de versiones de aplicación del algoritmo de consenso.

Tabla 1: Ficha para toma de muestras

Proceso	hora inicial	hora final	Tramas	No. Tomas

Fuente: Proceso de observación.
Elavorado: Ruben Nogales Portero

3.5. OPERACIONALIZACIÓN DE LA VARIABLES

La operacionalización de la variables es un proceso que inicia con la asignación de variables, las misma que pueden ser estrictamente medibles.

Según **Roberto et al. (2010, :Internet)** “Una definición operacional constituye el conjunto de procedimientos que describe las actividades que un observador debe realizar para recibir las impresiones sensoriales, las cuales indican la existencia de un concepto teórico en mayor o menor grado”, es decir como hago esto o aquello y como lo registro, en que lo registro.

> Los criterios para evaluar una definición operacional son básicamente tres: "adecuación al contexto", "confiabilidad" y "validez". De ellos se hablará en el apartado "Elaboración de los instrumentos de recolección de los datos". Una correcta selección de las definiciones operacionales disponibles o la creación de la propia definición operacional está muy relacionada con una adecuada revisión de la literatura. Cuando ésta ha sido cuidadosa, se puede tener una gama más amplia de definiciones operacionales para elegir o más ideas para crear una nueva.

3.5.1. Operacionalización de la variable independiente

Tabla 2: Operacionalización de la variable independiente

OPERACIONALIZACIÓN DE LA VARIABLE INDEPENDIENTE:				
Inadecuado uso de buenas prácticas de programación en el algoritmo de consenso.				
CONCEPTUALIZACIÓN	**CATEGORÍAS**	**INDICADORES**	**ITEMS BÁSICOS**	**TÉCNICAS E INSTRUMENTOS**
El inadecuado uso de buenas prácticas de programación en el algoritmo de consenso se conceptúa como: La evolución de las herramientas de programación y de la tecnología de redes con la aplicación de la programación orientada a objetos en el algoritmo de consenso como núcleo de la nube, constituye el avance tecnológico más importante en la categoría de servicios en red hasta el 2014.	- OOP - Nube (Cloud IT) - Algoritmo de Consenso	- La OOP es la técnica utilizada con mayor frecuencia en el desarrollo de software, desde su aparición (2002) hasta la fecha (2014). - Los servicios ofertados en la nube (IaaS, PaaS, SaaS) - Existen dos modelos de algoritmos de consenso (con fallos de parada y bizantino)	- Normas de programación. - Simulación en la nube -Versiones del algoritmo	- Observación de los resultados emitido por el software de evaluación.

Fuente: Marco Teórico.
Elavorado: Ruben Nogales Portero

3.5.2. Operacionalización de la variable dependiente

Tabla 3: Operacionalizacion de la variable dependiente

OPERACIONALIZACIÓN DE LA VARIABLE DEPENDIENTE:				
Excesivo consumo de recursos hardware y red.				
CONCEPTUALIZACIÓN	CATEGORÍAS	INDICADORES	ITEMS BÁSICOS	TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN
El Excesivo consumo de recursos hardware y red se conceptúa como: La inaplicación o aplicación incorrecta de normas de programación a conllevado a un consumo excesivo de recursos en la nuble.	- Red. - Memoria. - CPU.	- Software de medición de recursos de red. - Software de medición de recursos de memoria. - Software de medición de recursos de CPU.	- Consumo de memoria. - Consumo de red. - Consumo de CPU.	- Observación de software de instrumentos de evaluación. - Fichas.

Fuente: Marco Teórico.
Elavorado: Ruben Nogales Portero

3.6. RECOLECCIÓN DE INFORMACIÓN

Metodológicamente para **HERRERA (2004, :Internet)** "la construcción de la información se opera en dos fases: plan para la recolección de información y plan para el procesamiento de información".

3.6.1. Plan para la recolección de información

Este plan contempla estrategias metodológicas requeridas por los objetivos (ver Pág. 4 y 5) e hipótesis de investigación (ver Pág. 12), de acuerdo con el enfoque escogido que para el presente estudio es predominantemente cuantitativo (ver Pág. 13), considerando los siguientes elementos:

- Definición de los sujetos: personas u objetos que van a ser investigados. Los sujetos de investigación son los resultado obtenidos por el software de medición de procesos, haciendo referencia a la población y/o muestra de estudio (ver Pág. 18).
- Selección de las técnicas a emplear en el proceso de recolección de información. La recolección de datos será tomada en cada una de las implementaciones del algoritmo de consenso en las distintas versiones del mismo ayudado del software de medición (ver Pág. 18). La observación es el proceso sistemático de registro de los patrones de comportamiento de las personas, objetos y sucesos sin cuestionarlos ni comunicarse con ellos.
- Instrumentos seleccionados o diseñados de acuerdo con la técnica escogida para la investigación para documentar se utilizará fichas o lista de cotejos explicada en función del contenido de las columnas de técnicas e instrumentos de recolección de información de las diferentes matrices de operacionalización por variables (ver Pág. 19) las fichas de observación son instrumentos en los que se registran todos los sucesos y acontecimientos que se susciten al rededor de la variable en cuestión con una precisión casi fotográfica.
- Selección de recursos de apoyo (equipos de trabajo). Explicación sobre la(s) persona(s) que participarán en la investigación voluntariamente sin ningún tipo de remuneración económica, indicando la función exacta a realizar. NOTA: estas personas no pertenecen a la población de estudio.
- Explicación de procedimientos para la recolección de información.

Tabla 4: Procedimiento de recolección de información

TÉCNICAS	PROCEDIMIENTOS
	Se utilizará el método de investigación experimental.
Observación	La recolección de datos será en el departamento de investigación de la Universidad Técnica de Ambato
	La toma será de la segunda semana de septiembre del 2014 a la cuarta semana del mismo mes y año

Fuente: Investigación de campo
Elaborado por: Ruben Nogales Portero

El método experimental consiste en la introducción de variables a un proceso y observar y medir los resultados, tomado básicamente en tres momentos:

- Planificación.
- Experimentación.
- Toma de resultados.

Es decir se planifica e implementa la simulación de la nube, sobre esta implementa el algoritmo de consenso en sus distintas versiones y se toman medidas o dátos.

3.7. PROCESAMIENTO Y ANÁLISIS

3.7.1. Plan de procesamiento de información

En la investigación experimental se definen los pasos a seguir.

1. Simulación en la nube.
   a) Sistema operativo GNU/Linux
   b) Configuración de 3 servidores virtuales
2. Implementación en cada servidor virtual el algoritmo de consenso V01.
3. Descripción del lenguaje de programación en el cual esta realizado.
4. Descripción del método y técnica de programación utilizado.
5. Valiéndose del software de medición se observa.
   a) Consumo de recursos de red.
   b) Consumo de recursos de memoria.
   c) Consumo de recursos de CPU.

d) Consumo de disco.

6. Se simula casos de uso.

   a) Un servidor es el líder.

   b) Dos servidores no son líderes.

   c) Los valores propuestos para el consenso son diferentes.

7. La experimentación repite el paso 5 en los casos de uso.

8. Implementa en cada servidor virtual el algoritmo de consenso V(x).

9. Se repiten los pasos 3, 4, 5, 6.

10. Se repetirán los pasos 8 y 9 implementado versiones del algoritmo de consenso hasta que el investigador determine tener una versión estable.

El muestreo se lo realiza con 50 repeticiones en tanto que los intervalos de confianza son aceptables del experimento con 0 segundos de retardos en las corridas.

3.7.2. Plan de análisis e interpretación de resultados

Para cada métrica (memoria, CPU, red y disco) se generan archivos con los rendimientos de los cuales para la memoria se toma % de memoria usada, para la CPU se toma % de utilización, para la red se toma la tasa de KB recibidos y la tasa de KB transmitidos, y por último para el disco el % de utilización.

Los resultados recopilados en la ficha de observación vea (tabla 1) se tabularan en sus medias, de las diferentes mediciones, a partir de estos se puede obtener su tStudent, con estos valores graficar el diagrama de barras y los intervalos de confianza y se estará en capacidad de:

- Aplicar buenas técnicas de programación.
- Determinar si las buenas prácticas de programación mejoran el consumo de recursos de red, memoria y CPU.
- Sugerir el código de algoritmo de consenso $A\Omega$más estable.

CAPÍTULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1. ANÁLISIS DE LOS RESULTADOS

La investigación es de tipo experimental debido a que es necesario medir el rendimiento que se tendrá al programar el algoritmo de consenso en una programación lineal.

Como no se puede saber que parte del hardware se verá más afectado es primordial medir el rendimiento que tiene la:

- Utilización de memoria.
- Utilización de la CPU.
- Utilización del disco duro.
- Transmisión de paquetes en red.
- Recepción de paquetes en red.

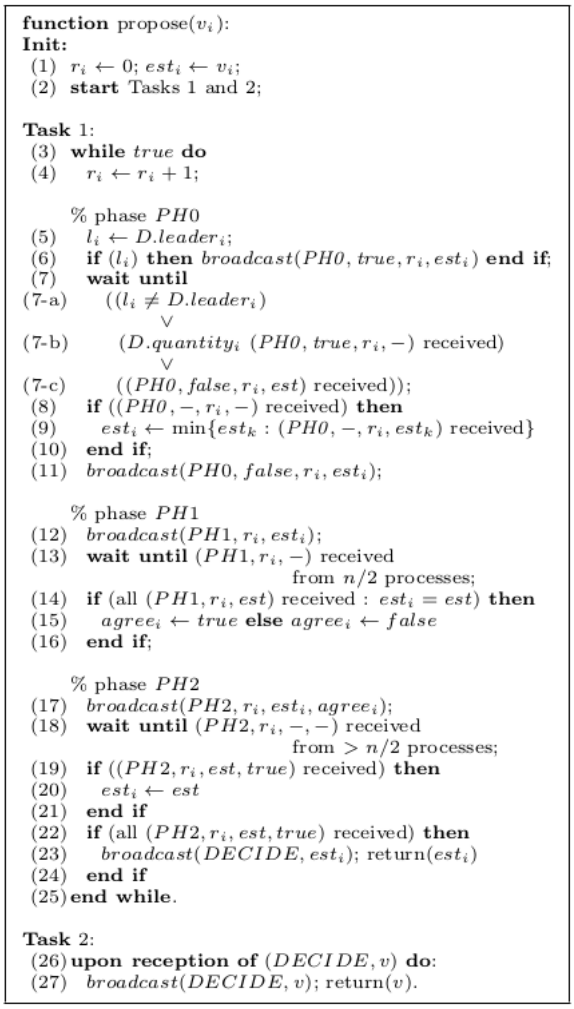
También se sabe que el consenso junto con el detector de fallos son el núcleo de la computación distribuida, y el presente trabajo de investigación consiste en:

1. Programar la versión del algoritmo de consenso presentada por el Dr. Ernesto Jiménez y su grupo de investigación
2. Probarlo en un ambiente virtualizado aislado.
3. Instalar en el ambiente virtualizado en sistemas operativos diferentes.
   a) Fedora 20
   b) Ubuntu LTS 14.04.1
   c) Centos server 6.5

4. Comunicarlos mediante la configuración de una red.
    a) Fedora 20 IP 172.18.0.21 / 24
    b) Ubuntu LTS 14.04.1 IP 172.18.0.14 / 24
    c) Centos server 6.5 IP 172.18.0.16 / 24
5. Instalar el algoritmo de consenso programado en cada uno de estos sistemas operativos.
6. El experimento se realizarán 50 veces con 0 segundos de delay (tiempo de retardo) en un minuto de tiempo, esto es una simulación de aproximadamente 7800 conexiones o solicitudes de conexión sucesivas por minuto.
7. El experimento se realizarán 50 veces con 5 segundos de delay (tiempo de retardo) en un minuto de tiempo, esto es una simulación de aproximadamente 12 conexiones o solicitudes de conexión sucesivas por minuto.
8. Durante este tiempo se realiza la captura de datos de rendimiento en memoria, CPU, disco y red; cada uno de estos en los sistemas operativos anteriormente mencionados.

4.2. PROGRAMACIÓN VERSIÓN 1 ALGORITMO DE CONSENSO $A\Omega'$

Para el experimento se ha procedido con la programación del algoritmo de consenso publicado por **Martin et al. (2007)**



function propose(v_i):
Init:
(1) $r_i \leftarrow 0$; $est_i \leftarrow v_i$;
(2) **start** Tasks 1 and 2;

Task 1:
(3) **while** $true$ **do**
(4) $r_i \leftarrow r_i + 1$;

% phase $PH0$
(5) $l_i \leftarrow D.leader_i$;
(6) **if** (l_i) **then** $broadcast(PH0, true, r_i, est_i)$ **end if**;
(7) **wait until**
(7-a) $((l_i \neq D.leader_i)$
$\vee$
(7-b) $(D.quantity_i\ (PH0, true, r_i, -)$ received)
$\vee$
(7-c) $((PH0, false, r_i, est)$ received));
(8) **if** $((PH0, -, r_i, -)$ received) **then**
(9) $est_i \leftarrow \min\{est_k : (PH0, -, r_i, est_k)$ received$\}$
(10) **end if**;
(11) $broadcast(PH0, false, r_i, est_i)$;

% phase $PH1$
(12) $broadcast(PH1, r_i, est_i)$;
(13) **wait until** $(PH1, r_i, -)$ received
from $n/2$ processes;
(14) **if** (all $(PH1, r_i, est)$ received : $est_i = est$) **then**
(15) $agree_i \leftarrow true$ **else** $agree_i \leftarrow false$
(16) **end if**;

% phase $PH2$
(17) $broadcast(PH2, r_i, est_i, agree_i)$;
(18) **wait until** $(PH2, r_i, -, -)$ received
from $> n/2$ processes;
(19) **if** $((PH2, r_i, est, true)$ received) **then**
(20) $est_i \leftarrow est$
(21) **end if**
(22) **if** (all $(PH2, r_i, est, true)$ received) **then**
(23) $broadcast(DECIDE, est_i)$; return(est_i)
(24) **end if**
(25) **end while**.

Task 2:
(26) **upon reception of** $(DECIDE, v)$ **do**:
(27) $broadcast(DECIDE, v)$; return(v).

Figura 4: Seudo código algorimo de consenso $A\Omega'$

La programación del algoritmo en su primera versión; programación lineal.

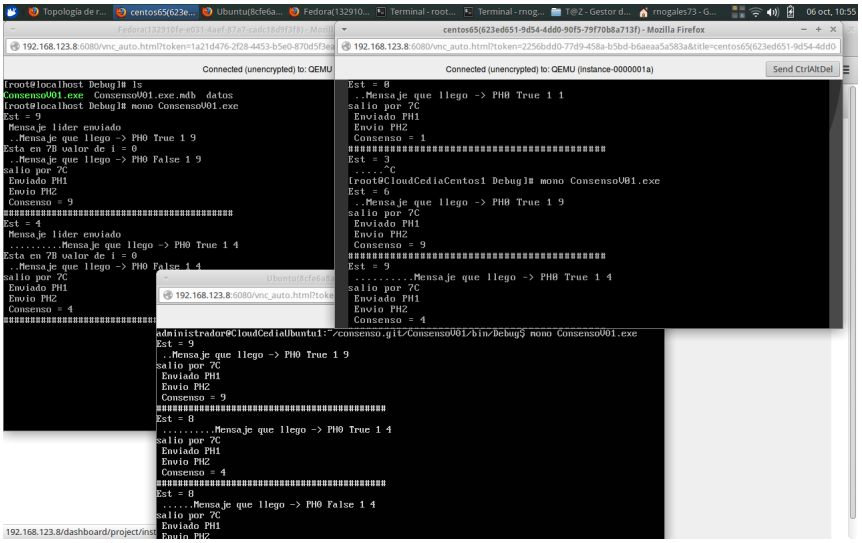
```
1  using System;
2  using System.Net;
3  using System.Net.Sockets;
4  using System.Text;
5
6  namespace ConsensoV01
7  {
8      class MainClass
9      {
10         public static void Main (string[] args)
11         {
12             if (args.Length == 0) {
13                 //Inicio de contador de vueltas
14                 int r = 0;
15                 //generación de valores aleatorios para el valor de est
16                 Random rnd = new Random (DateTime.Now.Millisecond);
17
18                 //int est = rnd.Next (0, 12);
19                 int est = 7;
20
21                 //Asigna el primer parámetro correspondiente al numero de nodos
22                 //double n = Int32.Parse (args[0]);
23                 double n = 3;
24
25                 //Asigna el segundo parámetro correspondiente al valor de Lider
```

```
26                  //String lider = args[1];
27                  String lider = "True";
28
29                  //Bandera para terminar la app una vez llegado a un consenso
30                  int salir = 0;
31
32                  //Variabla a utilizar como contador
33                  int i = 0;
34
35                  //Valor de n/2
36                  double media = 0;
37
38                  //Variable para guardar agree
39                  Boolean agree;
40
41                  //Valor de quantity
42                  //int quantity = Int32.Parse (args[2]);
43                  int quantity = 2;
44
45                  //Paquete de mensajes para enviar.
46                  byte[] paqueteEnv = null;
47
48                  //Paquete de mensajes para Recibir.
49                  byte[] paqueteRcv = null;
50
51                  //String para creación del String a enviar
52                  String envio = null;
53
54                  //String que va a recibir la cadena del broadcast
55                  String mensaje = null;
56
57                  //Datos de ip y puerto a utilizar
58                  String servidor = "192.168.1.255";
59                  int puerto = 11124;
60
61                  //Creacion de variables para utilizacion como cliente
62                  UdpClient client = new UdpClient ();
63
64                  //Creacion de variables para utilizacioón como servidor
65                  IPEndPoint remoteIPEndPoint = new IPEndPoint (IPAddress.Any, 0);
66                  UdpClient clientSrv = new UdpClient(puerto);
67
68                  Console.Write("Est = " + Convert.ToString(est) + " \n ");
69
70                  while(salir == 0){
71                      //Sumamos 1 en cada vuelta
72                      r++;
73
74                      //PHASE 0  //
75                      if(lider == "True"){
76                          //String de envio
77
78                          envio = "PH0 True " +Convert.ToString(r)+" "+Convert.ToString(est);
79                          paqueteEnv = Encoding.ASCII.GetBytes (envio);
80                          client.Send (paqueteEnv,paqueteEnv.Length,servidor,puerto);
81
82                          Console.Write("Mensaje lider enviado \n ");
83                      }
84                          //Espera que lleguen los valores est de todos los líderes segun el valor de
                                quantity
85
86                          //i empieza en 1 porque mi valor es el primero
87
88
89                          while(i < quantity){
90                          Console.Write (".");
91                              try{
92                                  //Recibe el paquete
93                                  paqueteRcv=clientSrv.Receive(ref remoteIPEndPoint);
94                                  mensaje = Encoding.ASCII.GetString(paqueteRcv,0,paqueteRcv.Length);
95
96                                  string[] words = mensaje.Split(' ');
97                                  Console.Write(".");
98                                  if((words[0] == "PH0" && Int32.Parse(words[2]) == r) && (words[1] == "True"
                                        || words[1] == "False")){
99                                  Console.Write("Mensaje que llego -> " + mensaje +"\n");
100                                     if(words[1] == "True" && lider == "True"){
```

```
101
102                                 if(i == 0){
103                                     est = Int32.Parse (words[3]);
104                                 }
105                                 //Aqui se comparan los valores que llegan para escoger el menor
106                                 if (Int32.Parse (words [3]) < est) {
107                                     est = Int32.Parse (words[3]);
108                                 }
109
110                                 Console.Write("Esta en 7B valor de i = "+ i +" \n ");
111                                 i++;
112                             }else{
113                                 est = Int32.Parse (words[3]);
114                                 i = quantity;
115                                 Console.Write("salio por 7C \n ");
116                             }
117
118                         }
119
120                     }catch(SocketException se)
121                     {
122                         Console.Write (se.ErrorCode+":"+se.Message);
123                     }
124
125
126                 }
127
128
129                 //Cuando ya llegan todos los valores, y se ha escogido el menor, se envia un broadcast
                        con el valor menor decidido
130                 envio = "PH0 False " +Convert.ToString(r)+" "+Convert.ToString(est);
131                 paqueteEnv = Encoding.ASCII.GetBytes (envio);
132                 client.Send (paqueteEnv,paqueteEnv.Length,servidor,puerto);
133
134                 //PHASE 1  //
135
136                 // Coger el valor superior de N/2
137                 if(n%2 == 0){
138                     media = n/2 + 1;
139                 }else{
140                     media = Math.Ceiling(n/2);
141                 }
142
143                 //Enviamos un broadcast con el valor decidido para est
144                 envio = "PH1 " +Convert.ToString(r)+" "+Convert.ToString(est);
145                 paqueteEnv = Encoding.ASCII.GetBytes (envio);
146                 client.Send (paqueteEnv,paqueteEnv.Length,servidor,puerto);
147
148                 Console.Write("Enviado PH1 \n ");
149
150                 //Booleano que lleva el control si todos los valores que se recibe en la phase 1 son
                        iguales
151                 agree = true;
152                 i = 0;
153                 //Captura de valores recibidos del broadcast de los valores decididos por todos los
                        nodos
154                     while(i < media){
155                         try{
156                             //Recibe el paquete
157                             paqueteRcv=clientSrv.Receive(ref remoteIPEndPoint);
158
159                             mensaje = Encoding.ASCII.GetString(paqueteRcv,0,paqueteRcv.Length);
160
161                             string[] words = mensaje.Split(' ');
162                             if(words[0] == "PH1" && Int32.Parse(words[1]) == r){
163                                 if(i == 0){
164                                     est = Int32.Parse(words[2]);
165                                 }
166
167                                 if(est != Int32.Parse(words[2])){
168                                     agree = false;
169                                     i = (int) media;
170                                 }
171                                 i++;
172                             }
173
174                         }catch(SocketException se)
```

```
175                             {
176                                 Console.Write (se.ErrorCode+":"+se.Message);
177                             }
178 
179                         }
180 
181 
182                     //PH2 //
183 
184                     //Enviamos el valor de Agree
185                     //Enviamos un broadcast con el valor decidido para est
186                     envio = "PH2 " +Convert.ToString(r)+" "+Convert.ToString(est)+" "+agree;
187                     paqueteEnv = Encoding.ASCII.GetBytes (envio);
188                     client.Send (paqueteEnv,paqueteEnv.Length,servidor,puerto);
189 
190                     Console.Write("Envio PH2 \n ");
191 
192                     agree = true;
193                     i = 0;
194                     //Captura de valores recibidos del broadcast de los valores decididos por todos los
                            nodos
195                         while(i < media){
196                             try{
197                                 //Recibe el paquete
198                                 paqueteRcv=clientSrv.Receive(ref remoteIPEndPoint);
199 
200                                 mensaje = Encoding.ASCII.GetString(paqueteRcv,0,paqueteRcv.Length);
201 
202                                 string[] words = mensaje.Split(' ');
203                                 if(words[0] == "PH2" && Int32.Parse(words[1]) == r){
204 
205                                     if(Convert.ToBoolean(words[3]) == true){
206                                         est = Int32.Parse(words[2]);
207                                     }else{
208                                         agree = false;
209                                     }
210                                     i++;
211                                 }
212 
213                             }catch(SocketException se)
214                             {
215                                 Console.Write (se.ErrorCode+":"+se.Message);
216                             }
217 
218                         }
219 
220                     //Si todos los valores son true entonces acabo el consenso, caso contrario se deja
                            listo el valor est para la siguiente ronda
221                     if(agree == true){
222                         //Enviamos un broadcast con el valor decidido para est
223                         envio = "Decide " +Convert.ToString(est);
224                         paqueteEnv = Encoding.ASCII.GetBytes (envio);
225                         client.Send (paqueteEnv,paqueteEnv.Length,servidor,puerto);
226 
227                         Console.Write("Consenso = " + Convert.ToString(est));
228                         salir = 1;
229                     }
230 
231 
232 
233                 } //Final del ciclo infinito
234 
235             } else {
236                 Console.Write ("Necesarios 3 parámetros: \n -Numero nodos\n -Lider=True-False \n -Quantity
                        \n ");
237             }
238         }
239     }
240 }
```

Toma de datos haciendo consenso.



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 5: Ejecución de Consenso.

Aquí se observa como cada uno de los servidores (Fedora 20, Centos server 6.5 y Ubuntu LTS 14.04.1) llegan a consensuar en un valor propuesto.

Gráficos.

Memoria.

Los datos capturados para el análisis son la columna cuarta referente al porcentaje de utilización de memoria como se muestra en la siguiente figura.

Linux 2.6.32-431.23.3.el6.x86_64 (CloudCediaCentos1) 24/11/14 _x86_64_ (1 CPU)

15:17:07 kbmemfree kbmemused %memused kbbuffers kbcached kbcommit %commit

15:17:09 196724 305532 60,83 139156 104312 84840 7,52

15:17:11 196600 305656 60,86 139156 104312 84840 7,52

15:17:13 196228 306028 60,93 139156 104316 85192 7,55

15:17:15 196104 306152 60,96 139156 104320 85192 7,55

15:17:17 196104 306152 60,96 139156 104324 85192 7,55

15:17:19 196104 306152 60,96 139156 104328 85192 7,55

15:17:21 195980 306276 60,98 139156 104332 85192 7,55

15:17:23 195980 306276 60,98 139156 104336 85192 7,55

15:17:25 195980 306276 60,98 139156 104340 85192 7,55

15:17:27 195980 306276 60,98 139156 104344 85192 7,55

15:17:29 195856 306400 61,00 139156 104352 85192 7,55

15:17:31 195856 306400 61,00 139156 104352 85192 7,55

15:17:33 195856 306400 61,00 139156 104356 85192 7,55

15:17:35 195856 306400 61,00 139156 104360 85192 7,55

15:17:37 195856 306400 61,00 139156 104364 85192 7,55

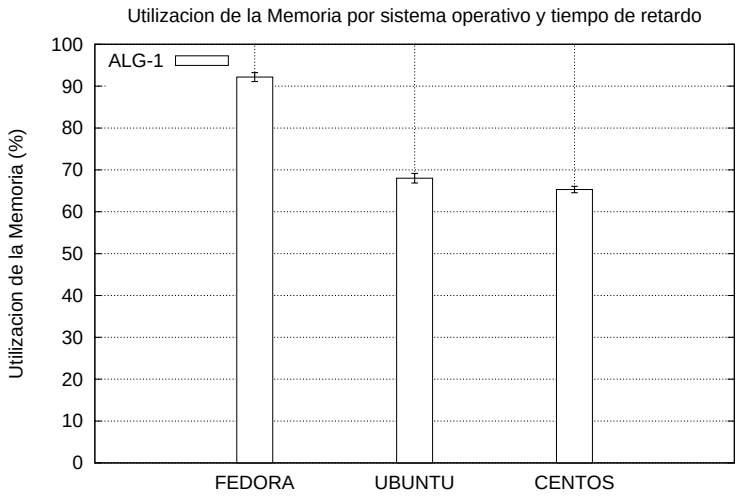
15:17:39 195856 306400 61,00 139156 104368 85192 7,55

15:17:41 195856 306400 61,00 139156 104372 85192 7,55

Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

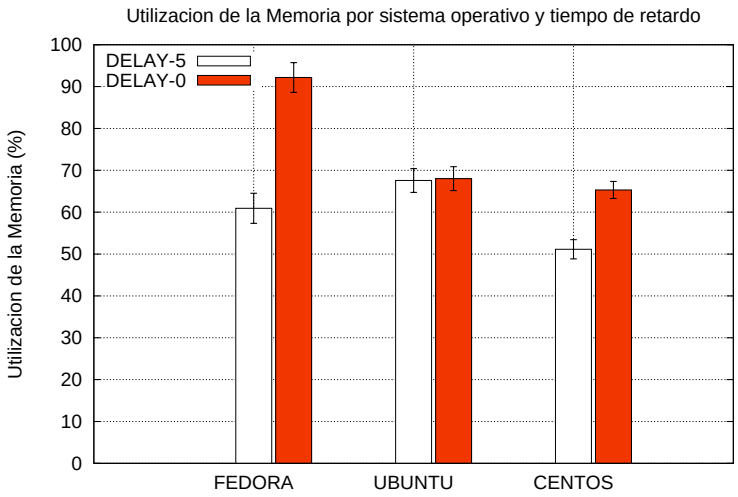
Figura 6: Muestra de datos capturados para la memoria

Muestra de datos.



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 7: Utilización de memoria V1 0 seg. de retardo



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 8: Comparativo de rendimiento con 0 seg y 5 seg de retardo

De las figuras 8 y 9 se puede concluir que:

- Fedora 20 tiene un porcentaje de uso del 92 % con 0 seg. de retardo mientras que con 5 seg. de retardo se tiene 62 % de utilización.
- Ubuntu LTS 14.04.1 con un porcentaje del 70 % de uso con 0 seg. de retardo y con 68 % de uso con 5 seg. de retardo.
- Centos server 6.5 con el 68 % de utilización con 0 seg. de retardo mientras con 5 seg. de retardo tiene 55 % de utilización.

Y con la figura 9 se puede evidenciar claramente el incremento de consumo de recursos.

CPU.

Para la CPU se referencia el % de utilización como se muestra.

Linux 2.6.32-431.23.3.el6.x86_64 (CloudCediaCentos1) 27/11/14 _x86_64_ (1 CPU)

15:42:03 CPU %user %nice %system %iowait %steal %idle

15:42:04 all 0,00 0,00 1,00 0,00 0,00 99,00

15:42:05 all 8,00 0,00 0,00 0,00 0,00 92,00

15:42:06 all 8,00 0,00 24,00 0,00 2,00 66,00

15:42:07 all 18,81 0,00 61,39 0,00 3,96 15,84

15:42:08 all 18,37 0,00 59,18 0,00 10,20 12,24

15:42:09 all 18,75 0,00 61,46 0,00 4,17 15,62

15:42:10 all 19,19 0,00 61,62 0,00 6,06 13,13

15:42:11 all 19,59 0,00 63,92 0,00 12,37 4,12

15:42:12 all 19,80 0,00 61,39 0,00 4,95 13,86

15:42:13 all 18,37 0,00 58,16 0,00 7,14 16,33

15:42:14 all 17,00 0,00 63,00 0,00 8,00 12,00

15:42:15 all 15,00 0,00 59,00 0,00 9,00 17,00

15:42:16 all 16,49 0,00 57,73 0,00 8,25 17,53

15:42:17 all 17,17 0,00 59,60 0,00 7,07 16,16

15:42:18 all 18,18 0,00 56,57 1,01 9,09 15,15

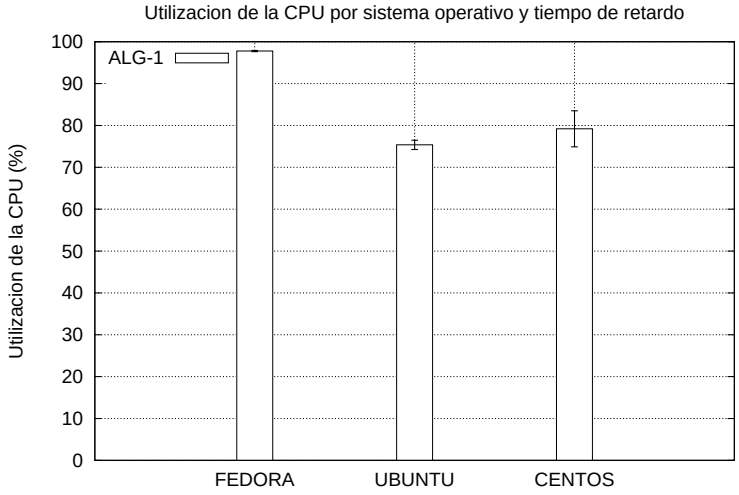
15:42:19 all 18,37 0,00 59,18 0,00 7,14 15,31

15:42:20 all 15,79 0,00 64,21 0,00 6,32 13,68

15:42:21 all 21,21 0,00 59,60 0,00 3,03 16,16

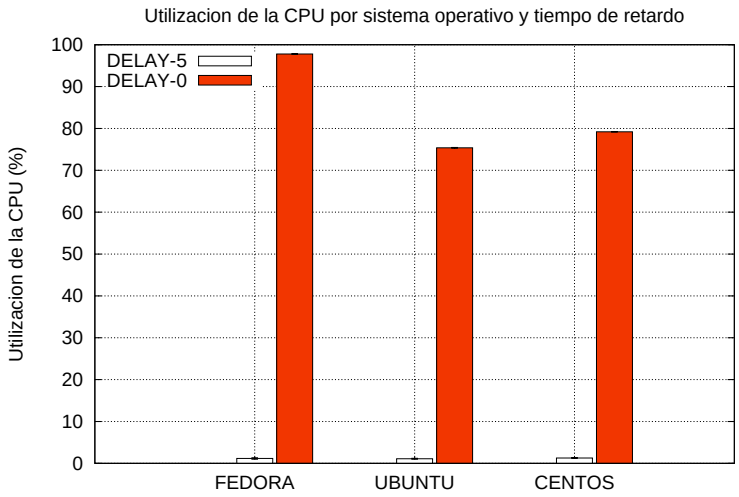
Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 9: Muestra del % de utilización de la CPU



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 10: Utilización de la CPU



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 11: Comparativo de rendimiento con 0 seg y 5 seg de delay

Según lo observado en la figura 11 se obtiene:

- El sistema operativo Fedora 20 alcanza el 98 % de su utilización.
- Mientras que Ubuntu LTS 14.04.1 alcanza un 78 % de utilización.
- Y Centos server 6.5 llega a utilizar un 80 %.

Este recurso es visiblemente estresado al simular 7800 solicitudes o 0 segundos de retardo como se muestra en la figura 12.

Disco.

Para analizar el rendimiento del disco duro se tomado el % de utilización del mismo mientras el algoritmo esta en funcionamiento y simulando las peticiones requeridas.

Linux 2.6.32-431.23.3.el6.x86_64 (CloudCediaCentos1) 27/11/14 _x86_64_ (1 CPU)

15:42:03 DEV tps rd_sec/s wr_sec/s avgrq-sz avgqu-sz await svctm %util

15:42:04 dev252-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:04 dev253-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:04 dev253-1 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:04 DEV tps rd_sec/s wr_sec/s avgrq-sz avgqu-sz await svctm %util

15:42:05 dev252-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:05 dev253-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:05 dev253-1 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:05 DEV tps rd_sec/s wr_sec/s avgrq-sz avgqu-sz await svctm %util

15:42:06 dev252-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:06 dev253-0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:06 dev253-1 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

15:42:06 DEV tps rd_sec/s wr_sec/s avgrq-sz avgqu-sz await svctm %util

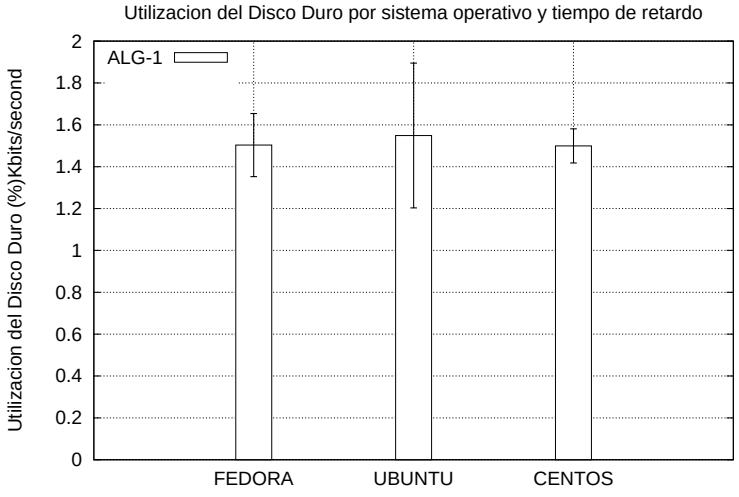
15:42:07 dev252-0 0,00 0,00 0,00 0,00 0,01 0,00 0,00 0,39

15:42:07 dev253-0 8,82 0,00 70,59 8,00 0,04 0,00 0,44 0,39

15:42:07 dev253-1 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

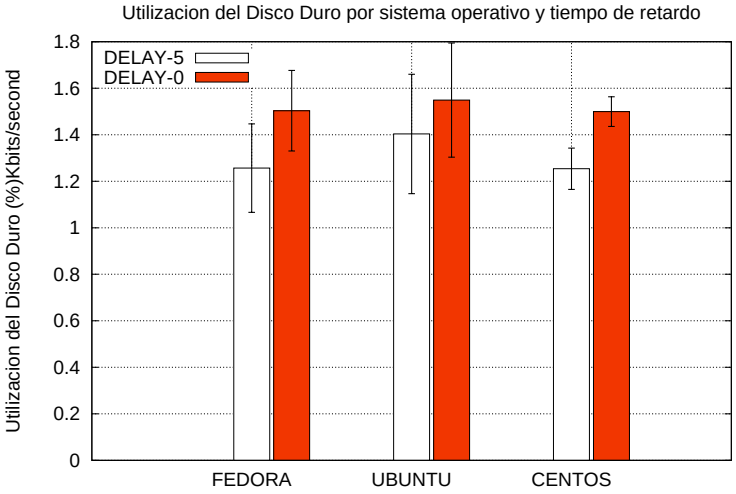
Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 12: Muestra del % de utilización del disco duro



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 13: Utilización de disco duro



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 14: Comparativo de rendimiento con 0 seg y 5 seg de delay

Según la observación el rango de valores de utilización de disco es muy bajo tanto que:

- Fedora 20 tiene un % de utilización del 1.8 %.
- Ubuntu LTS 14.04.1 1.9 % de utilización.
- Y Centos server 6.5 1.8 %.

Se puede ver que el rendimiento es de este recurso es casi homogéneo en los sistemas operativos probados y tampoco es significativo al estresar los sistemas con 0 segundos de retardo.

Red tasa de transmisión TxKB/s.

Para poder analizar la red se a tomado la columna 6 de la captura referente a la tasa de transmisión en KB por segundo

Red Tx.

Media: IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

Media: lo 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: eth0 1439,43 656,44 74,50 33,86 0,00 0,00 0,00

Media: IFACE rxerr/s txerr/s coll/s rxdrop/s txdrop/s txcarr/s rxfram/s rxfifo/s txfifo/s

Media: lo 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: eth0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: call/s retrans/s read/s write/s access/s getatt/s

Media: 0,00 0,00 0,00 0,00 0,00 0,00

Media: scall/s badcall/s packet/s udp/s tcp/s hit/s miss/s sread/s swrite/s saccess/s sgetatt/s

Media: 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: totsck tcpsck udpsck rawsck ip-frag tcp-tw

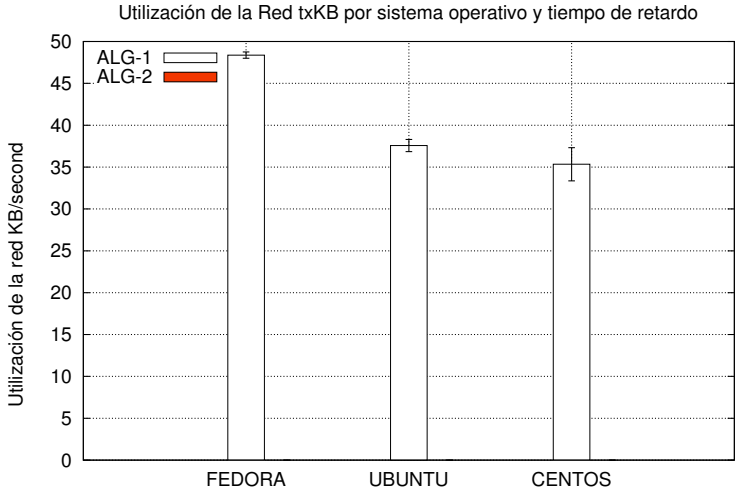
Media: 119 3 2 0 0 0

Media: irec/s fwddgm/s idel/s orq/s asmrq/s asmok/s fragok/s fragcrt/s

Media: 2095,86 0,00 2095,86 656,44 0,00 0,00 0,00 0,00

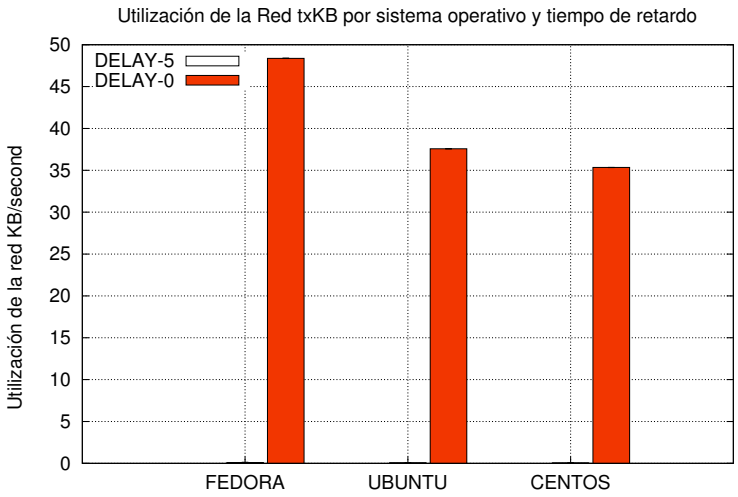
Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 15: Muestra de la tasa de transmisión de KB en la red



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 16: Tasa de transmisión Tx.



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 17: Comparativo de rendimiento con 0 seg y 5 seg de delay

En el análisis se puede concluir que los paquetes transmitidos con retardo de 0 segundos es inmensamente superior alcanzando:

1. Fedora 20 un 48 %
2. Ubuntu LTS 14.04.1 38 %
3. Centos server 6.5 35 %.

Mientras que con un retardo de 5 segundos el consumo de red es infimo.

Red tasa de transmisión RxKB/s.

Para poder analizar la red se a tomado la columna 5 de la captura referente a la tasa de recepción en KB por segundo.

Media: IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

Media: lo 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: eth0 1439,43 656,44 74,50 33,86 0,00 0,00 0,00

Media: IFACE rxerr/s txerr/s coll/s rxdrop/s txdrop/s txcarr/s rxfram/s rxfifo/s txfifo/s

Media: lo 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: eth0 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: call/s retrans/s read/s write/s access/s getatt/s

Media: 0,00 0,00 0,00 0,00 0,00 0,00

Media: scall/s badcall/s packet/s udp/s tcp/s hit/s miss/s sread/s swrite/s saccess/s sgetatt/s

Media: 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00

Media: totsck tcpsck udpsck rawsck ip-frag tcp-tw

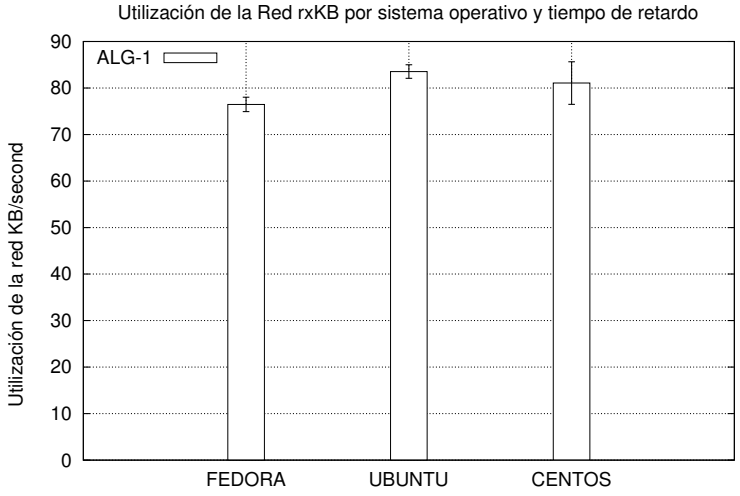
Media: 119 3 2 0 0 0

Media: irec/s fwddgm/s idel/s orq/s asmrq/s asmok/s fragok/s fragcrt/s

Media: 2095,86 0,00 2095,86 656,44 0,00 0,00 0,00 0,00

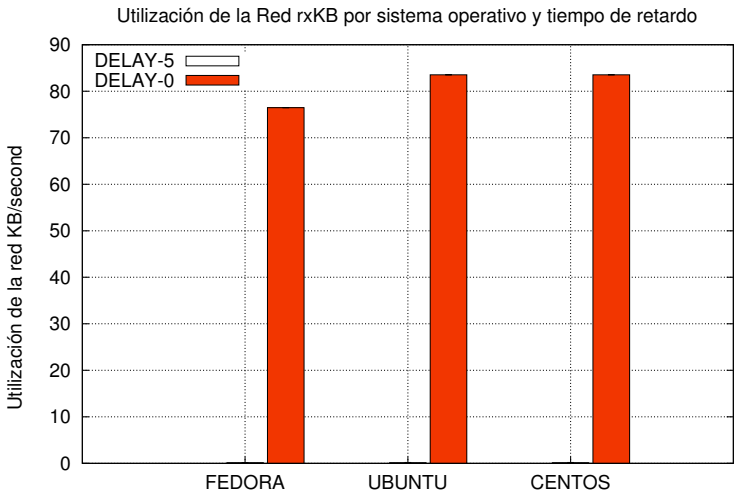
Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 18: Muestra de la tasa de recepción en KB/s



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 19: Tasa de transmisión de red Rx.



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 20: Comparativo de rendimiento con 0 seg y 5 seg de delay

Esta observación denota que:

- Fedora 20 alcanza una utilización del 78 %
- Ubuntu LTS 14.04.1 el 82 % y
- Centos server 6.5 el 82 %

La figura 21 demuestra claramente que la saturación de la red con 0 segundos es muy significativo sobre la utilización con 5 segundos de retardo.

4.3. INTERPRETACIÓN DE DATOS

Memoria.

La memoria es el recurso más utilizado dentro del procesamiento de datos, pues tiene que prestar servicio a todos los requerimientos del sistema. Y los sistemas operativos linux se basan en una estructura de capas, en donde las capas más internas van pasando parámetros a las externas para su funcionamiento.

Fedora 20.- Este sistema operativo separa la memoria de un proceso de la memoria de los demás procesos excepto los hilos del mismo proceso, entonces Fedora asigna

un espacio de direccionamiento de memoria para usuarios por proceso y asigna un espacio de memoria para el kernel para los demás procesos. Durante el experimento como existe un solo proceso corriendo se asigna casi toda su memoria a ese proceso.

Ubuntu LTS 14.04.1.- La utilización de la memoria de Ubuntu está relacionada a la utilización del disco, puesto que el inicio del kernel sube a memoria y lo guarda en su partición raíz.

Centos Server 6.5.- La administración de la memoria en este sistema operativo es similar a la de Ubuntu LTS 14.04.1.

CPU.

Probablemente el CPU sea la unida más importante de un ordenador, denominado también microprocesador o procesador.

El procesador básicamente limita su ámbito de funcionamiento en cinco etapas.

1. Búsqueda de la instrucción.
2. Decodificación de la instrucción.
3. Búsqueda de los operandos.
4. Ejecución de la instrucción.
5. Almacenamiento de los resultados.

Todo esto en sus dos unidades, la unidad de control y la unidad aritmético lógica.

Sin embargo la figura 11 demuestra que la utilización de la CPU en Fedora 20 es superior sobre Centos server 6.5 y más aún sobre Ubuntu LTS 14.04.1.

Disco.

Para el efecto del experimento la utilización de recursos de disco duro es ínfima sin embargo los datos arrojados han sido recogidos y efectivamente las figuras 14 y 15 demuestran que la utilización es inferior a 2 Kbits por segundo y el rendimiento es casi estándar en los tres sistemas operativos, Fedora 20, Ubuntu LTS 14.04.1 y Centos server 6.5.

Red Tx.

En el caso del algoritmo de consenso se esta utilizando un broadcast para hacer conocer a los servidores que ya se ha llegado a un conseso y se demuestra en la figura 20 que la utilización de la red en el caso de Fedora 20 es mayoritario debido a

que este sistema operativo tiene en su núcleo todas las implementaciones nuevas que la comunidad aporta, mientras que Ubuntu LTS 14.04.1 y Centos server 6.5 tienen todo lo estable.

Sin embargo la figura 21 representa el comportamiento de la recepción de paquetes en la red, siendo su desempeño muy lineal o aparecido en los sistemas operativos probados.

4.4. VERIFICACIÓN DE LA HIPÓTESIS

H1=Hipótesis alterna.- Se Necesita saber si existe una diferencia significativa en la utilización de recursos al correr el algoritmo de consenso con 0 segundos y con 5 segundos de retardo.

H0=Hipótesis nula.- No existe una diferencia significativa en la utilización de recursos al correr el algoritmo de consenso con 0 segundos y con 5 segundos de retardo.

Valor Alfa = 5 % = 0.05 valor de significancia o grado de error.

Se aplicará un estudio transversal, puesto que los experimentos se van a realizar en el mismo instante y analizar en dos circunstancias diferentes: Con 0 segundos y 5 segundos de retardo.

Para la comprobación se utilizará la variable aleatoria numérica que es el valor obtenido de la toma de datos del experimento.

Esta explicación determina que para la comprobación de la hipótesis se utiliza t-student para muestras independientes. (los resultados se muestran calculados en el software spss)

Estadísticas de grupo

Tiempo de retardo	*N*	*Media*	*Desviación Estándar*	*Err.Est.Media*
Valor 5 Seg Retardo	98	,06	,13	,01
0 Seg Retardo	51	37,57	2,03	,28

Prueba para muestras independientes

	Prueba de Levene para la igualdad de varianzas		Prueba T para la Igualdad de Medias						
								Intervalo de confianza 95% de la Diferencia	
	F	*Sign.*	*t*	*df*	*Sign. (2-colas)*	*Diferencia Media*	*Err.Est. de la Diferencia*	*Inferior*	*Superior*
ValorSe asume igualdad de varianzas	110,60	,00	-182,40	147,00	,00	-37,52	,29	-38,08	-36,95
Igualdad de varianzas no asumida			-131,55	50,21	,00	-37,52	,29	-38,09	-36,94

Figura 21: Prueba t-student para muestras independientes

El valor de significancia de 0,05 es > que el valor obtenido 0,00.

Por tanto acepto la hipótesis alterna, que dice que si existe una variación significativa al correr el algoritmo de consenso entre 0 y 5 seg. de retardo.

Rechazo la hipótesis nula.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones.

1. Se demuestra con la validación de la hipótesis (si existe una variación significativa al correr el algoritmo de consenso entre 0 y 5 seg. de retardo) que es necesario mejorar las técnicas de programación en el algoritmo de consenso $A\Omega'$.

2. Mediante el proceso de toma y representación de datos se demuestra un excesivo consumo de recursos hardware y red.

   - Se demuestra la incidencia de los valores correspondientes por tanto.

     - CPU.

	Delay 0	Delay 5	Diferencia
Fedora 20	98 %	2 %	96 %
Ubuntu LTS 14.04.1	78 %	2 %	76 %
Centos server 6.5	80 %	2 %	78 %

Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.
Tabla 5: Comparativo de rendimiento con 0 y 5 seg. de retardo

     - RED.

     - Tx

	Delay 0	Delay 5	Diferencia
Fedora 20	48 %	1 %	47 %
Ubuntu LTS 14.04.1	38 %	1 %	37 %
Centos server 6.5	35 %	1 %	34 %

Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.
Tabla 6: Comparativo de rendimiento con 0 y 5 seg. de retardo

- Rx

	Delay 0	Delay 5	Diferencia
Fedora 20	78 %	1 %	77 %
Ubuntu LTS 14.04.1	82 %	1 %	81 %
Centos server 6.5	80 %	1 %	79 %

Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.
Tabla 7: Comparativo de rendimiento con 0 y 5 seg. de retardo

3. Los valores que corresponden a Memoria y Disco en la medición del algortimo de conseno $A\Omega'$, no representan información relevante para el estudio debido a que sus diferencias con 0 seg. y 5 seg. de retardo no son significativas.

   a) En el caso de la memoria se instancian las mismas variables y utilizan el mismo espacio de memoria con 0 o con 5 seg. de retardo

   b) En el caso de la utilización del disco es referenciado únicamente al instante de la ejecución inicial del algoritmo y posteriormente no hay recurrencias a disco.

4. El método utilizado para la toma de datos y su correspondiente análisis estadístico es confiable.

Recomendaciones.

1. Es necesario programar en clases las fases PH0, PH1, PH2 del algoritmo de consenso $A\Omega'$, con el fin de mejorar la modularización y la reutilización del código así como también su rendimiento en cuanto al consumo de recursos hardware y red se refieren.

2. Se recomienda aislar los elementos de la experimentación (sistemas operativos Fedora 20, Ubuntu LTS 14.04.1 y Centos server 6.5) en un sistema Cloud

Computing para anular la posibilidad de influencia de elementos externos en la toma de datos.

3. Para verificar la incidencia del uso de las buenas prácticas de programación en el algoritmo de consenso se recomienda hacer un estudio comparativo entre las versiones programadas de dicho algoritmo.

CAPÍTULO VI

PROPUESTA

6.1. DATOS INFORMATIVOS

6.1.1. Tema.

"El uso de buenas prácticas de programación en el algoritmo de consenso y su incidencia en el consumo de recursos de hardware y red para el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato"

6.1.2. Institución ejecutora.

Proyecto Cloud-CEDIA multidisciplinario: participan Universidad Técnica Particular de Loja UTPL, Escuela Politécnica Nacional y Universidad Técnica de Ambato.

6.2. ANTECEDENTES DE LA PROPUESTA

Existen diferentes estudios y propuestas sobre el consenso en las redes distribuidas como también existen muchas reglas de programación que se han ido implementando acorde a la evolución del software y a las exigencias del mundo actual. Se ha implementado a nivel internacional varios protocolos de consenso utilizando diferentes tipos de fallos **Chandra and Toueg (1996)**, siendo esté un elemento importante en resolver problemas de consenso.

La presente propuesta investigativa cuenta con un estudio y programación del algoritmo de consenso $A\Omega'$ propuesto dentro del proyecto Cloud-CEDIA donde el grupo de investigación de la UTA tiene la responsabilidad de resolver el consenso.

Se pretende en el presente trabajo medir el rendimiento del consenso en tres sistemas

operativos diferentes y de estos el análisis de datos de métricas como la utilización de memoria, disco, CPU y red; utilizando dos versiones de programación del algortimo de consenso $A\Omega'$

6.3. JUSTIFICACIÓN

En el proyecto multidisciplinario planteado a CEDIA la Universidad Técnica de Ambato tiene la posibilidad de juntarse al mismo en la etapa de desarrollo del algoritmo de consenso $A\Omega'$ junto con la Universidad Técnica Particular de Loja, la Escuela Politécnica Nacional y la consultoría de Docentes de la Universidad Politécnica de Madrid.

Al programar el algoritmo de consenso utilizando técnicas de buenas prácticas de programación se espera mostrar la mejora en el rendimiento de consumo de recursos en la implementación del algoritmo de consenso $A\Omega'$, siendo un componente importante dentro de los objetivos a cumplir dentro del grupo de investigación del proyecto Cloud-CEDIA UTA, que utilizará la experiencia de esta investigación para su posterior integración con los componentes a desarrollar por la Escuela Politécnia Nacional y la Universidad Técnica Particular de Loja.

6.4. OBJETIVOS

6.4.1. Objetivo General.

- Demostrar que el uso de buenas prácticas de programación en el algoritmo de consenso inciden en el consumo de recursos de hardware y red en el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato.

6.4.2. Objetivos Específicos.

- Programar y aplicar el algoritmo de consenso basado en buenas prácticas de programación.
- Diseñar un protocolo para la captura de información del proceso experimental.
- Análizar los resultados de comparación para determinar la incidencia del consumo de recursos hardware y software en el proyecto Cloud-CEDIA de la Universidad Técnica de Ambato

6.5. ANÁLISIS DE FACTIBILIDAD

6.5.1. Factibilidad Técnica

El proyecto de investigación Cloud-CEDIA-UTA cuenta con el equipamiento y tecnología:

- Hardware
  - Servidor HP....
    - xxx MB RAM.
    - xxx GB HD.
    - xxx Red.
- Software.
  - Sistemas operativos GNU Linux.
  - Open Stack.
- Comunicaciones
  - Servicio de internet provisto por CEDIA (Corporación Ecuatoriana de Internet Avanzado), necesaria para la realización del presente estudio, además se cuenta con el asesoramiento y experiencia de los investigadores miembros del grupo de inversigación.

6.5.2. Factibilidad Operativa

La propuesta es factible debido a que el proyecto Cloud-CEDIA cuenta con la infraestructura adecuada (hardware, software, laboratorio de experimentación), además existe el apoyo de todo el personal involucrado en el proyecto

6.5.3. Factibilidad Económica

El presente estudio es parte de un proyecto de investigación financiado por CEDIA y por el DIDE de la UTA, garantizando el contar con el presupuesto y equipamiento necesario para el cumplimiento de los objetivos planteados.

6.6. FUNDAMENTOS

Buenas prácticas de programación

Según **Moroni and Señas (2005)** la programación eficaz se aborda de una forma general no de un lenguaje de programación específico o de una marca de procesador o de un sistema operativo específico.

Para la optimización de código en los lenguajes de programación es necesario saber si el código es compilado o será interpretado, si la gestión de la memoria es manual o automática y por último saber que tipo de aplicación se va a programar si es software para cliente servidor o para aplicativos web.

Para la obtener un código eficaz no es necesario corregir linea por linea la programación del software, tampoco es necesario ser un experto en el lenguaje en el que esta desarrollado el software.

El primer paso es identificar que tipo de problema si es de eficiencia o es de rendimiento.

Eficiencia.

Para determinar si un programa es eficiente primero se debe estar seguro que el código este funcionando correctamente. Un síntoma de poca eficiencia en un programa resulta en que este se cuelgue o se bloquee de forma constante.

Si el desarrollo es nuevo el código deberá ser:

- Claro
- Exacto
- Preciso
- Legible
- Comprensible
- Modular.

Rendimiento.

Cuando una aplicación tiene problemas de rendimiento suele presentar:

- Lenta al arrancar.
- El manejo de entrada y salida de datos es lento
- Excesivo consumo de recursos.
    - Memoria.
    - CPU.
    - Disco.
    - Red.

El segundo paso es realizar un análisis objetivo del programa o aplicación antes de sugerir que este tiene algún problema, este análisis puede ser realizado con herramientas especializadas (test de red, compiladores, depuradores).

Dentro de la eficiencia se debe abordar cuatro áreas de interés como:

- Memoria.
    - Tipos de datos.
    - Variables.
    - Matrices
    - Objetos.
- Algorítmica.
- Recursos de E/S archivos.
    - Minimizar el número de veces que se requiera hacer entradas o salidas de datos.
    - Usar un formato adecuado.
    - Un buen sistema de compresión de archivos
- Red.
    - Almacenamiento en la nube.
    - Servicios Web.

- Conexiones Pear to Pear.

Cada una de estas áreas son independientes entre si, esto hace que cada una de estas pueda ser medible por si sola.

El tercer paso se basa en la optimización que permite identificar el nivel de control que se posee dentro de la aplicación.

- Bajo control.- Del lado del cliente (front end).
- Alto control.- Del lado del servidor (software, Base de datos).

El cuarto paso es utilizar la solución mas simple y que proporcione mejores resultados

- Identificar cuellos de botella donde se vean comprometidos la cantidad de almacenamientos de memoria, o la velocidad de la CPU o la velocidad de escritura o lectura del disco o las comunicaciones de red.

Consenso

Es un servicio que permite acordar un mismo valor entre todos los equipos de la nube. Es un caso particular de "k-set-agreement". Concretamente, consenso es el caso cuando k=1. La consecución de este consenso se hace mucho más difícil de conseguir cuando los equipos pueden fallar (por ejemplo porque un computador ha dejado de funcionar).

Resolver el problema del consenso es muy importante porque se encuentra en el núcleo de muchas aplicaciones distribuidas. El consenso puede ser definido de forma que cada proceso propone un determinado valor, y todos los procesos correctos tienen que estar de acuerdo en tomar una misma decisión. Esta decisión tiene que ser uno de los valores propuestos por alguno de los procesos. Más formalmente, en el problema de consenso cada proceso propone un valor, y cada proceso correcto tiene que decidir un mismo valor "v" (propiedad de terminación), de tal forma que el valor decidido "v" es uno de los valores propuestos (propiedad de validez), y no pueden existir dos procesos que decidan de forma diferente del valor "v" (propiedad de acuerdo).

Son varios los protocolos que se pueden encontrar el día de hoy en la literatura donde se analizan la resolución del problema de consenso sobre un sistema distribuidos síncronos. En **Attiya and Welch (2004); Lynch (1996); Raynal (2010)** se puede encontrar ejemplos de estos protocolos. En los tres casos la resolución del problema se basa en diseñar algoritmos que actúan en distintas rondas. De los

artículos sobre consenso que se pueden encontrar hoy en día hay que destacar los siguientes artículos que implementan consenso con las siguientes características: • En **Chandra and Toueg (1996)** se han implementado varios protocolos de consenso utilizando varíos tipos de detectores de fallos (P, S, <>P, <>S ...). Este artículo es un trabajo que introduce el concepto de detector de fallos como una herramienta útil para resolver problemas de coordinación tales como consenso. El sistema asíncrono que es descrito en el artículo tiene las siguientes características: enlaces fiables, membresía conocida y estática, y fallos de caída-parada para los procesos. • En **Aguilera et al. (2000)**se han implementado varios protocolos para resolver consenso utilizando detectors de fallos. En todos los casos analizados, el sistema es estático, con membresía conocida, enlaces "fair-lossy", y con fallos de caída-recuperación en los procesos. • En **Arévalo et al. (2012)** se implementa un protocolo para resolver consenso. Utiliza un detector de fallos con homónimos (llamado $H\Omega$, es decir, "homonymous-omega"). Este trabajo es muy importante porque resuelve consenso en sistemas distribuidos con características muy avanzadas: membresía desconocida y existencia de procesos homónimos. En la actualidad ya existen varios proyectos a nivel internacional que están trabajando en el estudio e implementación de consenso y detectores de fallos en los sistemas distribuidos en la nube tradicionales. De entre ellas cabe resaltar dos:

ZooKeeper (http://zookeeper.apache.org). Es un proyecto para ofrecer servicios de coordinación en sistemas distribuidos de la Apache Software Foundation. Entre estos servicios están los de consenso y detectores de fallos.

Megastore (http://en.wikipedia.org/wiki/Google_Storage). Es un servicio de Google para ofrecer espacio en disco con alta disponibilidad y escalabilidad en la nube. Los accesos a este espacio en disco son coordinados de forma que se ofrezca una determinada coherencia en las lecturas y escrituras a los datos.

Computación en la Nube

Conocida en términos anglosajones como Cloud Computing es una nuevo campo de investigación multidisiplinario considerado hacer una evolución en lo que respecta a la computación. Es un modelo que permite acceder a los recursos computacionales a través de un modelo por demanda por ejemplo sea recursos de red, servers, espacio de almacenamiento, aplicaciones o servicios. Todo esto de una manera rápida y disminuyendo al mínimo los requerimientos de administición y mantenimiento de los recursos informáticos dentro de las empresas u organizaciones que hacen uso del

cloud computing. **Computing (2010)**

La computación en la nube se clasifica en:

IaaS Infraestructura como un Servicio, orientada al Administrador de red, provee poder de cómputo, Ej. Servidores Virtuales

PaaS Plataforma como un Servicio, orientada a los Desarrolladores de software, provee tiempo de ejecución Ej. Plataformas de desarrollo

SaaS Software como un Servicio, orientada a los usuarios finales y empresarios, proveer acceso a las aplicaciones, Ej, Aplicaciones y soluciones de software

XaaS Cualquier cosa como un servicio, es una nueva tendencia en los servicios de cloud computing.

Tipos de Computación en la Nube.

Las diferentes implementaciones de nubes dependen de las necesidades y propósitos de la Nube. Estas diferentes implementaciones determinan diferentes tipos de nubes las mismas que pueden estar orientadas a la investigación, a la producción, también pueden variar los tipos debido a sus costos, seguridad, personalización. **Ibou and Marquezan (2013)**La mayoría de tipos de nubes se clasifican dentro de las siguientes:

- Públicas
- Privadas
- Hibridas
- Comunitarias

OpenStack

Es un proyecto de computación en la Nube que se la cataloga como IaaS. Muchas empresas son parte de este proyecto entre las principales RedHat, Suse, Dell, HP, IBM, entre otros.

Openstack tiene una arquitectura modular con varios componentes los mismos que son **Ibou and Marquezan (2013)**:

Nova: Es el controlador de la estructura cloud computing, gestiona y automatiza

los recursos del equipo, compatible con diferentes tecnologías de virtualización como KVM, Xen, entre otras

Swift: Es el componente encargado del almacenamiento redundante y escalable, permite mantener la integridad y replicación de los datos en el cluster del centro de datos

Cinder: Gestiona los dispositivos de almacenamiento a nivel de bloque, permite usar diferentes plataformas de almacenamiento como GlusterFS, SAN, iSCSI, GPFS, entre otros

Neutron: Gestiona la red y el direccionamiento IP, permite crear redes independientes manejadas por los usuarios

Horizon: Es el entorno gráfico de administración de los componentes que forman parte de Openstack

Keystone: Es un sistema de autenticación que se puede integrar con LDAP. Además es compatible con otros sistemas de autenticación.

Glance: Es un servicio de imágenes, permite almacenar las imágenes de disco y transformarlas en plantillas.

Ceilometer: Es un sistema que permite centralizar la auditoría y los registros acerca del funcionamiento del sistema de computación en la nube que brinda OpenStack

Sysstat

Es un conjunto de herramientas que permiten realizar un monitoreo y captura de información acerca del rendimiento y la actividad de un servidor con el sistema operativo GNU/Linux. Es compatible con la mayoría de las distribuciones GNU/Linux disponibles en la actualidad. Todas las herramientas pueden ser ejecutadas de forma programada haciendo uso de cron para colectar y almacenar información acerca del rendimiento y la activada de los datos en un servidor **Godard (2010)**.

Sar

Esta utilidad permite colectar y generar reportes sobre el sistema relacionado a los siguientes parámetros:

- CPU
- Memoria
- Disco
- Interrupciones
- Interfaces de red
- TTY
- Kernel

6.7. METODOLOGÍA

La metodología para el cumplimiento de los objetivos planteados en la propuesta de la investigación se define en la realización de las siguientes actividades y procesos:

1. Sistema Cloud Computing a nivel de infraestructura
2. Simulación en la nube.
    a) Instalación y configuración de Fedora 20:
        1) 512 RAM.
        2) 1 Disco Duro de 8 GB.
        3) 1 CPU INTEL CORE 4000 MHz con un cache de 4MB.
        4) 1 tarjeta de red
        5) Instalación, actualización y configuración de paquetes requeridos para la ejecución del algoritmo de consenso $A\Omega'$
        6) Mono JIT Compiler 2.10.8 - Mono/NET 4.0
    b) Instalación y configuración de Ubuntu server LTS 12.4:
        1) 512 RAM.
        2) 1 Disco Duro de 6 GB.
        3) 1 CPU INTEL CORE 4000 MHz con un cache de 4Mb.
        4) 1 tarjeta de red

5) Instalación, actualización y configuración de paquetes requeridos para la ejecución del algoritmo de consenso $A\Omega'$

6) Mono JIT Compiler 3.2.8 - Mono/NET 4.0

c) Instalación y configuración de Centos server 6.5:

1) 512 RAM.

2) 1 Disco Duro de 6 GB.

3) 1 CPU INTEL CORE 4000 MHz con un cache de 4Mb.

4) 1 tarjeta de red

5) Instalación, actualización y configuración de paquetes requeridos para la ejecución del algoritmo de consenso $A\Omega'$

6) Mono JIT Compiler 2.10.8 - Mono/NET 4.0

3. Implementamos en cada servidor virtual el algoritmo de consenso versión y versión 2 .

4. Instalación y configuración del SAR (software utilizado para tomar datos).

5. Desarrollo e implementación de scripts en cada servidor.

6. Pruebas del lenguaje de programación seleccionado.

7. Pruebas del método y técnica de programación utilizado.

8. Con el software de medición y scripts generados se observará.

a) Consumo de recursos de red.

b) Consumo de recursos de memoria.

c) Consumo de recursos de CPU.

d) Consumo de disco.

9. Obtención de archivos y estandarización de salida:

a) REDAAAA-MM-DD-HH-mm.

b) MemAAAA-MM-DD-HH-mm.

c) HDAAAA-MM-DD-HH-mm.

d) CPUAAAA-MM-DD-HH-mm.

e) Siendo AAAA el año MM los minutos DD el dia HH la hora y mm los minutos en que se realizaron las capturas de los datos.

f) Esto en cada uno de los sistemas operativos seleccionados.

10. Procesamiento de los archivos del proceso 9 en:

    a) Obtención de la media aritmética del conjunto de datos de cada archivo generado con su respectivo valor de tstudent (para poder calcular los intervalos de confianza). Generando:

    b) net_tx_valuesV1_SO.dat

    c) net_tx_valuesV11_SO.dat

    d) net_rx_valuesV1_SO.dat

    e) net_rx_valuesV11_SO.dat

    f) mem_valuesV1_SO.dat

    g) mem_valuesV11_SO.dat

    h) hd_valuesV1_SO.dat

    i) hd_valuesV11_SO.dat

    j) cpu_valuesV1_SO.dat

    k) cpu_valuesV11_SO.dat

    l) Siendo SO el sistema operativo

11. Se repite los procesos 8 9 y 10 del experimento hasta que el investigador determine o considere tener datos confiables y que reflejen los objetivos planteados.

6.8. RESULTADOS

Para la obtención de resultados confiables es necesario la aplicación de la metodología propuesta para lo que se utilizaron recursos hardware, software y de comunicaciones del proyecto de investigación Cloud-CEDIA de la UTA.

6.8.1. Sistema Cloud Computing

Se accede al sistema IaaS con Openstack, donde se genera un proyecto llamado Cloud-CEDIA. Este proyecto tiene las siguientes características técnicas y dispone de los siguientes recursos:

Recursos	Características
Red de datos	172.18.0.0/24
Router virtual	172.18.0.1, 192.168.123.210
Instancias	Fedora 20, Ubuntu LTS 12.4, Centos 6.5
Hipervisor	KVM
Fedora 20	512 MB RAM, 8GB Disco, IP: 172.18.0.21/24
Ubuntu LTS 12.4	512 MB RAM, 6GB Disco, IP: 172.18.0.14/24
Centos 6.5	512 MB RAM, 6GB Disco, IP: 172.18.0.16/24

Tabla 8: Recursos disponibles server Openstack

En la Figura 23 se muestra el panel de administración de OpenStack donde se puede observar las instancias disponibles con las características técnicas de cada una de ellas.

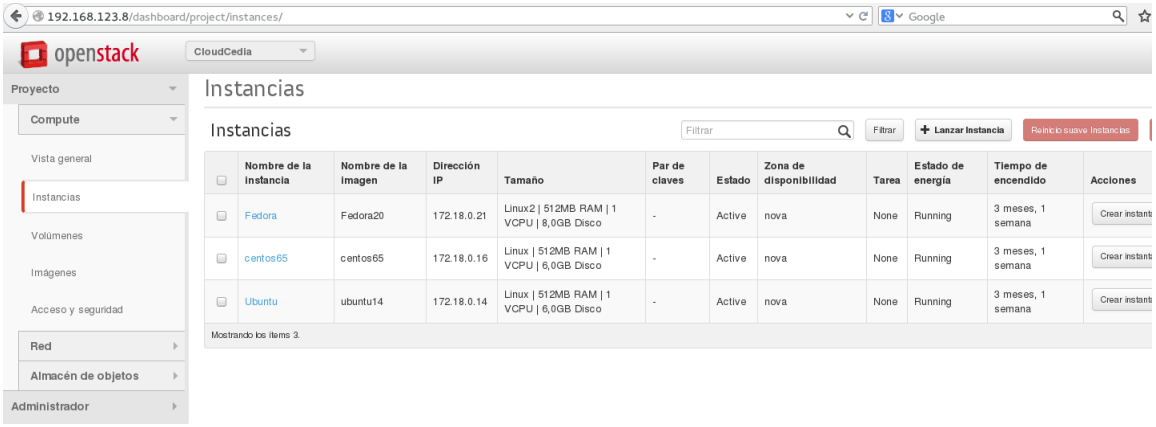


Figura 22: Instancias en Openstack

En la Figura 24 se observa la topología de la red virtual que permite aislar las instancias para obtener datos libres de interferencias de elementos externos a la red de datos real.

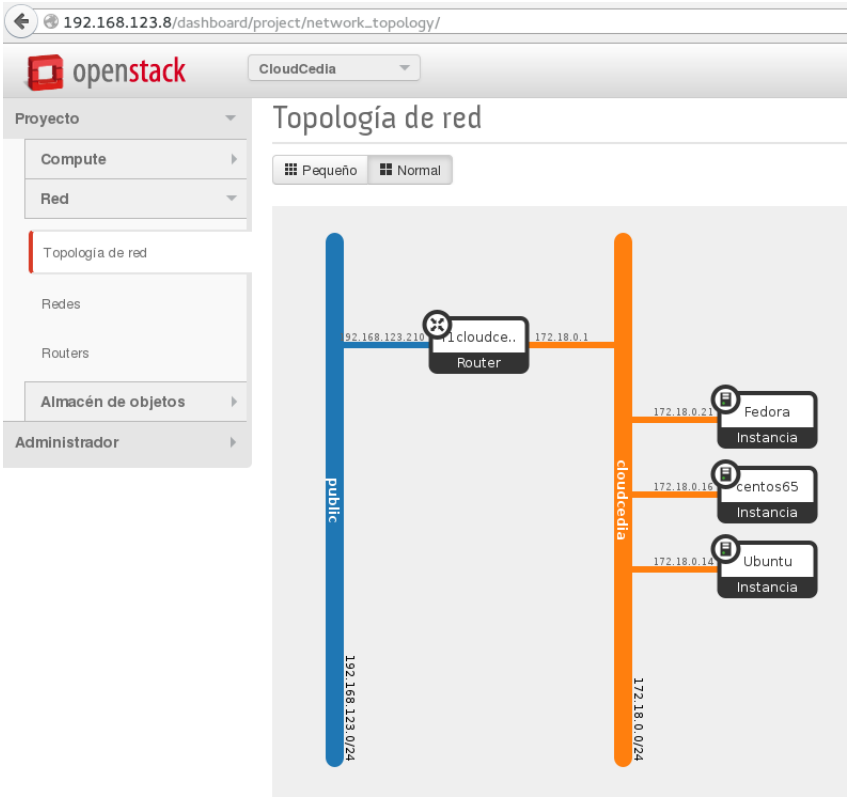


Figura 23: Topología de Red en Openstack

6.8.2. Versión 2 del programa para el algortimo de consenso $A\Omega'$

El algoritmo esta programado en cuatro clases, la clase principal es la que define las variables que serán utilizadas durante la ejecución del consenso.

Program.cs

```
1  using System;
2  using System.Net;
3  using System.Net.Sockets;
4  using System.Text;
5
6  namespace ConsensoV01
7  {
8      class MainClass
9      {
10         public static void Main (string[] args)
11         {
12             object_layer obj = new object_layer();
13
14
15                 //Inicializamos la variable randomica
16                 obj.setRnd(new Random(DateTime.Now.Millisecond));
```

```
17
18                  int est = obj.getRnd().Next(0 ,12 );
19
20                  //obj.setEst = obj.getRnd.Next (0,12);
21                  obj.setEst(est);
22
23                  //Asignamos el valor de control de fin de ciclo
24                  //obj.setSalir(0);
25
26                  string[] lines = System.IO.File.ReadAllLines("datos");
27                  string[] linesConfigs = System.IO.File.ReadAllLines("configs
                        ");
28
29                  //asignamos el primer parametro al numero de nodos
30                  obj.setN(Int32.Parse(lines[0]));
31
32                  //asignamos el segundo parametro correspondiente al valor de
                         lider
33                  obj.setLider(lines[1]);
34
35                  //Asignamos el valor del quantity del tercer valor pasado
                        como argumento
36                  obj.setQuantity(Int32.Parse(lines[2]));
37
38                  //asignamos ip del servidor
39                  obj.setServidor(linesConfigs[0]);
40
41                  //Asignamos numero del puerto
42                  obj.setPuerto(Int32.Parse(linesConfigs[1]));
43
44                  //Asignamos el tiempo de espera luego de consenso
45                  obj.setTiempo (Int32.Parse(linesConfigs[2]));
46
47                  //Inicializamos la variable del cliente
48                  //obj.setClient(new UdpClient());
49
50                  //iniciamos varialbe para utilizar como servidor
51                  //obj.setRemoteIPEndPoint(new IPEndPoint(IPAddress.Any, 0));
52                  obj.setClientSrv(new UdpClient(obj.getPuerto()));
53
54                  obj.setR(0);
55
56              while(true){
57
58                  //Inicializamos la variable randomica
59                  obj.setRnd(new Random(DateTime.Now.Millisecond));
60
61                  est = obj.getRnd().Next(0 ,15 );
62                  //obj.setEst = obj.getRnd.Next (0,12);
63                  obj.setEst(est);
64
65                  Console.Write ("Est = "+ Convert.ToString(obj.getEst())+" \n ");
66
67                  //Sumamos 1 en cada vuelta
68                  obj.setR(obj.getR()+1);
69
70                          try {
```

```
71                              PH0 fase0 = new PH0 (obj);
72                              Console.Write ("FASE 0 LISTO \n ");
73
74                          } catch (Exception ex) {
75
76                          }
77                          try {
78                              PH1 fase1 = new PH1 (obj);
79                              Console.Write ("FASE 1 LISTO \n ");
80                          } catch (Exception ex) {
81
82                          }
83
84                          try {
85                              PH2 fase2 = new PH2 (obj);
86                              Console.Write ("FASE 2 LISTO \n ");
87                          } catch (Exception ex) {
88
89
90                          }
91                      //Hastsa mientras
92
93                  }  //Final del ciclo infinito
94          }
95      }
96  }
```

Object_layer.cs

```
1   using System;
2   using System.Net;
3   using System.Net.Sockets;
4
5   namespace ConsensoV01
6   {
7       public class object_layer
8       {
9           public object_layer ()
10          {
11          }
12
13          #region Variables
14
15                  private int r;
16                  //generación de valores aleatorios para el valor de est
17                  // new Random (DateTime.Now.Millisecond)
18                  private Random rnd = new Random(DateTime.Now.Millisecond);
19
20                  //Valor de est para empezar a trabajar
21                  private int est;
22
23                  //Asigna el primer parámetro correspondiente al numero de nodos
24                  private double n;
25
26                  //Asigna el segundo parámetro correspondiente al valor de Lider
27                  private String lider;
```

```
28
29                  //Bandera para terminar la app una vez llegado a un consenso
30                  private int salir;
31
32                  //Valor de n/2
33                  private double media;
34
35                  //Variable para guardar agree
36                  private Boolean agree;
37
38                  //Valor de quantity
39                  //int quantity = Int32.Parse (args[2]);
40                  private int quantity;
41
42                  //Paquete de mensajes para enviar.
43                  private byte[] paqueteEnv;
44
45                  //Paquete de mensajes para Recibir.
46                  private byte[] paqueteRcv;
47
48                  //String para creación del String a enviar
49                  private String envio;
50
51                  //String que va a recibir la cadena del broadcast
52                  private String mensaje;
53
54                  //Datos de ip y puerto a utilizar
55                  private String servidor;
56                  private int puerto;
57
58                  //Creacion de variables para utilizacion como cliente
59                  private UdpClient client = new UdpClient();
60
61                  //Creacion de variables para utilizacioón como servidor
62                  private IPEndPoint remoteIPEndPoint = new IPEndPoint(IPAddress.
                        Any,0);
63                  private UdpClient clientSrv;
64
65                  private int tiempo;
66
67          #endregion
68
69
70          #region Acceso a las variables
71          //Acceso a R
72          public void setR (int valor){ this.r = valor; }
73          public int getR (){ return this.r; }
74
75          //Acceso a Random
76          public void setRnd (Random valor){ this.rnd = valor; }
77          public Random getRnd (){ return this.rnd; }
78
79          //Acceso a valor Est
80          public void setEst (int valor){ this.est = valor; }
81          public int getEst (){ return this.est; }
82
83          //Acceso al valor n
```

```
84          public void setN (double valor){ this.n = valor; }
85          public double getN (){ return this.n; }
86
87          //Acceso al valor Lider
88          public void setLider (String valor){ this.lider = valor; }
89          public String getLider (){ return this.lider; }
90
91          //Acceso al valor de salir
92          public void setSalir (int valor){ this.salir = valor; }
93          public int getSalir (){ return this.salir; }
94
95          //Acceso al valor de media
96          public void setMedia (double valor){ this.media = valor; }
97          public double getMedia (){ return this.media; }
98
99          //Acceso al valor agree
100         public void setAgree (Boolean valor){ this.agree = valor; }
101         public Boolean getAgree (){ return this.agree; }
102
103         //Acceso al valor quantity
104         public void setQuantity (int valor){ this.quantity = valor; }
105         public int getQuantity (){ return this.quantity; }
106
107         //Acceso al valor PaqueteEnv
108         public void setPaqueteEnv (byte[] valor){ this.paqueteEnv = valor; }
109         public byte[] getPaqueteEnv (){ return this.paqueteEnv; }
110
111         //Acceso al valor de PaqueteRcv
112         public void setPaqueteRcv (byte[] valor){ this.paqueteRcv = valor; }
113         public byte[] getPaqueteRcv (){ return this.paqueteRcv; }
114
115         //Acceso al valor de envio
116         public void setEnvio (String valor){ this.envio = valor; }
117         public String getEnvio (){ return this.envio; }
118
119         //Acceso al valor de Mensaje
120         public void setMensaje (String valor){ this.mensaje = valor; }
121         public String getMensaje (){ return this.mensaje; }
122
123         //Acceso al valor de Servidor
124         public void setServidor (String valor){ this.servidor = valor; }
125         public String getServidor (){ return this.servidor; }
126
127         //Acceso al valor del puerto
128         public void setPuerto (int valor){ this.puerto = valor; }
129         public int getPuerto (){ return this.puerto; }
130
131         //Acceso al valor de udpClient
132         public void setClient (UdpClient valor){ this.client = valor; }
133         public UdpClient getClient (){ return this.client; }
134
135         //Acceso a valor de remoteIPEndPoint
136         public void setRemoteIPEndPoint (IPEndPoint valor){ this.
                remoteIPEndPoint = valor; }
137         public IPEndPoint getRemoteIPEndPoint (){ return this.remoteIPEndPoint;
                }
138
```

```
139         //Acceso al valor del ClientSrv
140         public void setClientSrv (UdpClient valor){ this.clientSrv = valor; }
141         public UdpClient getClientSrv (){ return this.clientSrv; }
142
143         //Acceso al valor d etiempo
144         public void setTiempo (int valor) { this.tiempo = valor; }
145         public int getTiempo() {return this.tiempo; }
146
147         #endregion
148     }
149 }
```

PH0.cs

```
1  using System;
2  using System.Text;
3  using System.Net;
4  using System.Net.Sockets;
5
6  namespace ConsensoV01
7  {
8      public class PH0
9      {
10         public PH0 (object_layer obj)
11         {
12
13             IPEndPoint remoteIPEnPoint = new IPEndPoint(IPAddress.Any,0);
14
15
16             ///// Inicio de fases /////
17             //  while(obj.getSalir() == 0){
18
19
20                 //PHASE 0  //
21                 if(obj.getLider() == "True"){
22                     //String de envio
23
24                     obj.setEnvio("PH0 True " +Convert.ToString(obj.getR())+" "+
                           Convert.ToString(obj.getEst()));
25                     obj.setPaqueteEnv((Encoding.ASCII.GetBytes (obj.getEnvio()))
                           );
26                     obj.getClient().Send (obj.getPaqueteEnv(),obj.getPaqueteEnv
                           ().Length,obj.getServidor(),obj.getPuerto());
27
28                     Console.Write("Mensaje lider enviado \n ");
29                 }
30                 //Espera que lleguen los valores est de todos los líderes segun
                       el valor de quantity
31
32
33                 int i=0;
34                 int contador = 0;
35                 int[] valores = new int[obj.getQuantity()];
36
37                 while(i < obj.getQuantity()){
38
```

```
39                      try{
40                          //Recibe el paquete
41
42                          obj.setPaqueteRcv(obj.getClientSrv().Receive(ref
                                remoteIPEnPoint));
43                          obj.setMensaje ( Encoding.ASCII.GetString(obj.
                                getPaqueteRcv(),0,obj.getPaqueteRcv().Length));
44
45                          string[] words = obj.getMensaje().Split(' ');
46                          Console.Write(".");
47                          //ENTRADA TANTO PARA 7B COMO PARA 7C
48                          if((words[0] == "PH0" && Int32.Parse(words[2]) == obj.
                                getR()) && (words[1] == "True" || words[1] == "False
                                ")){
49
50                              Console.Write("Mensaje que llego -> " + obj.
                                    getMensaje() +"\n");
51                              if(words[1] == "True" && obj.getLider() == "True"){
52
53                                  //if (Int32.Parse (words [3]) < obj.getEst()) {
54                                  valores[contador]=Int32.Parse (words[3]);
55                                  //}
56
57                                  Console.Write("Esta en 7B valor de i = "+ i +" \
                                        n ");
58                                  i++;
59                                  contador++;
60
61                              }
62                              if(words[1] == "False"){
63                                  valores[contador]=Int32.Parse (words[3]);
64                                  i = obj.getQuantity();
65                                  Console.Write("salio por 7C \n ");
66                                  contador++;
67                              }
68
69                          }
70
71                      }catch(SocketException se)
72                      {
73                          Console.Write (se.ErrorCode+":"+se.Message);
74                      }
75
76                  }
77
78                  //Escoger el menor de los valores recividos
79                  int menor = valores [contador - 1];
80                  contador--;
81                  if (contador > 0) {
82                      while (contador > 0) {
83                          Console.WriteLine ("Valor "+Convert.ToString(valores[
                                contador]));
84                          if (menor > valores [contador-1]) {
85                              menor = valores [contador-1];
86                          }
87                          contador--;
88                      }
```

```
89                      Console.WriteLine ("Menos "+Convert.ToString(valores[
                            contador]));
90                  }
91
92                  obj.setEst (menor);
93                  //Cuando ya llegan todos los valores, y se ha escogido el menor,
                        se envi9a un broadcast con el valor menor decidido
94                  obj.setEnvio("PH0 False " +Convert.ToString(obj.getR())+" "+
                        Convert.ToString(obj.getEst()));
95                  obj.setPaqueteEnv(Encoding.ASCII.GetBytes (obj.getEnvio()));
96                  obj.getClient().Send (obj.getPaqueteEnv(),obj.getPaqueteEnv().
                        Length,obj.getServidor(),obj.getPuerto());
97                  //Console.WriteLine (Convert.ToString(obj.getEst()));
98          }
99      }
100 }
```

PH1.cs

```
1   using System;
2   using System;
3   using System.Text;
4   using System.Net;
5   using System.Net.Sockets;
6
7   namespace ConsensoV01
8   {
9       public class PH1
10      {
11          public PH1 (object_layer obj)
12          {
13              int i;
14              IPEndPoint remoteIPEnPoint = new IPEndPoint(IPAddress.Any,0);
15              // Coger el valor superior de N/2
16              if(obj.getN()%2 == 0){
17                  obj.setMedia(obj.getN()/2 + 1);
18              }else{
19                  obj.setMedia(Math.Ceiling(obj.getN()/2));
20              }
21
22              //Enviamos un broadcast con el valor decidido para est
23              obj.setEnvio("PH1 " +Convert.ToString(obj.getR())+" "+Convert.
                    ToString(obj.getEst()));
24              obj.setPaqueteEnv(Encoding.ASCII.GetBytes (obj.getEnvio()));
25              obj.getClient().Send (obj.getPaqueteEnv(),obj.getPaqueteEnv().Length
                    ,obj.getServidor(),obj.getPuerto());
26
27              Console.Write("Enviado PH1 \n ");
28              Console.WriteLine (obj.getEnvio());
29
30              //Booleano que lleva el control si todos los valores que se recibe
                    en la phase 1 son iguales
31              obj.setAgree(true);
32              i = 0;
33              //Captura de valores recibidos del broadcast de los valores
                    decididos por todos los nodos
```

```
34              while(i < obj.getMedia()){
35                  try{
36                      //Recibe el paquete
37                      obj.setPaqueteRcv(obj.getClientSrv().Receive(ref
                            remoteIPEnPoint));
38
39                      obj.setMensaje(Encoding.ASCII.GetString(obj.getPaqueteRcv()
                            ,0,obj.getPaqueteRcv().Length));
40
41                      string[] words = obj.getMensaje().Split(' ');
42                      if(words[0] == "PH1" && Int32.Parse(words[1]) == obj.getR())
                            {
43
44                          if(obj.getEst() != Int32.Parse(words[2])){
45                              obj.setAgree(false);
46                              i = (int) obj.getMedia();
47                          }
48                          i++;
49                      }
50
51                  }catch(SocketException se)
52                  {
53                      Console.Write (se.ErrorCode+":"+se.Message);
54                  }
55
56              }
57          }
58      }
59  }
```

PH2.cs

```
1   using System;
2   using System.Text;
3   using System.Net;
4   using System.Net.Sockets;
5
6   namespace ConsensoV01
7   {
8       public class PH2
9       {
10          public PH2 (object_layer obj)
11          {
12              IPEndPoint remoteIPEnPoint = new IPEndPoint(IPAddress.Any,0);
13              int i;
14              obj.setEnvio("PH2 " +Convert.ToString(obj.getR())+" "+Convert.
                    ToString(obj.getEst())+" "+obj.getAgree());
15              obj.setPaqueteEnv(Encoding.ASCII.GetBytes (obj.getEnvio()));
16              obj.getClient().Send (obj.getPaqueteEnv(),obj.getPaqueteEnv().Length
                    ,obj.getServidor(),obj.getPuerto());
17
18              Console.Write("Envio PH2 \n ");
19
20              obj.setAgree(true);
21              i = 0;
```

```
22              //Captura de valores recibidos del broadcast de los valores
                    decididos por todos los nodos
23              while(i < obj.getMedia()){
24                  try{
25                      //Recibe el paquete
26                      obj.setPaqueteRcv(obj.getClientSrv().Receive(ref
                            remoteIPEnPoint));
27
28                      obj.setMensaje(Encoding.ASCII.GetString(obj.getPaqueteRcv()
                            ,0,obj.getPaqueteRcv().Length));
29
30                      string[] words = obj.getMensaje().Split(' ');
31                      if(words[0] == "PH2" && Int32.Parse(words[1]) == obj.getR())
                            {
32
33                          if(Convert.ToBoolean(words[3]) == true){
34                              obj.setEst(Int32.Parse(words[2]));
35                          }else{
36                              obj.setAgree(false);
37                              i = (int) obj.getMedia();
38                          }
39                          i++;
40                      }
41
42                  }catch(SocketException se)
43                  {
44                      Console.Write (se.ErrorCode+":"+se.Message);
45                  }
46
47              }
48
49              //Si todos los valores son true entonces acabo el consenso, caso
                    contrario se deja listo el valor est para la siguiente ronda
50              if(obj.getAgree() == true){
51                  //Enviamos un broadcast con el valor decidido para est
52                  obj.setEnvio("Decide " +Convert.ToString(obj.getEst()));
53                  obj.setPaqueteEnv(Encoding.ASCII.GetBytes (obj.getEnvio()));
54                  obj.getClient().Send (obj.getPaqueteEnv(),obj.getPaqueteEnv().
                        Length,obj.getServidor(),obj.getPuerto());
55
56                  Console.Write("Consenso = " + Convert.ToString(obj.getEst())+"\n
                        ");
57
58                  //obj.setSalir(1);
59                  Console.WriteLine("####################");
60                  System.Threading.Thread.Sleep (obj.getTiempo());
61                  //obj.setR(0);
62              }
63
64
65          }
66      }
67  }
```

6.8.3. Programas Scripts

Para la obtención de los resultados se programaron diferentes scripts bash que ejecutaron instacias y aplicaciones como por ejemplo SAR y las versiones de consenso programadas en C#, que luego se utilizan en la generación de los resultados para la determinación del rendimiento del algoritmo en los entornos descritos.

Los principales scripts bash programados son:

- Captura de datos programa para el algoritmo de consenso $A\Omega'$ versión 1

```
#!/bin/bash
Path_toma='/home/rnogales73/Documentos/Tesis/Tomas/'
Path_mono='/home/rnogales73/consenso.git/ConsensoV01/bin/Debug/'
Fecha=$(date +%Y-%m-%d-%H:%M)
Nombre_archivo1='CPU'
Nombre_archivo2='Mem'
Nombre_archivo3='HD'
Nombre_archivo4='RED'
echo ..... Inicia sar
sleep 5s
sar -u 1 60 > "$Path_toma$Nombre_archivo1$Fecha" &
sar -r 1 60 > "$Path_toma$Nombre_archivo2$Fecha" &
sar -d 1 60 > "$Path_toma$Nombre_archivo3$Fecha" &
sar -n ALL 1 60 > "$Path_toma$Nombre_archivo4$Fecha" &
echo "$Path_toma$Nombre_archivo$Fecha"
bg
echo ..... Inicia MonoDeveloper
sleep 1s
cd $Path_mono
mono ConsensoV01.exe
```

- Captura de datos programa para el algoritmo de consenso $A\Omega'$ versión 2

```
#!/bin/bash
Path_toma='/home/rnogales73/Documentos/Tesis/Tomas/TomasV1.1/'
Path_mono='/home/rnogales73/ConsensoV01/ConsensoV01/bin/Debug/'
Fecha=$(date +%Y-%m-%d-%H:%M)
Nombre_archivo1='CPU'
Nombre_archivo2='Mem'
Nombre_archivo3='HD'
Nombre_archivo4='RED'
echo ..... Inicia sar
sleep 5s
sar -u 2 20 > "$Path_toma$Nombre_archivo1$Fecha" &
sar -r 2 20 > "$Path_toma$Nombre_archivo2$Fecha" &
sar -d 2 20 > "$Path_toma$Nombre_archivo3$Fecha" &
sar -n ALL 2 20 > "$Path_toma$Nombre_archivo4$Fecha" &
echo "$Path_toma$Nombre_archivo$Fecha"
bg
echo ..... Inicia MonoDeveloper
sleep 1s
cd $Path_mono
mono ConsensoV01.exe
```

- Copiar los archivos generados con los scripts anteriores en el ambiente aislado a la máquina de procesamiento.

```
#!/bin/bash
Cadena='rnogales73@192.168.30.36'
Cadena_Fedora='root@172.18.0.21'
Cadena_Centos='root@172.18.0.16'
Cadena_Ubuntu='administrador@172.18.0.14'

Path_FedoraV1='/usr/local/Tomas/'
Path_CentosV1='/home/Tomas/'
Path_UbuntuV1='/home/administrador/Tomas/'

Path_FedoraV11='/usr/local/TomasV1.1/'
Path_CentosV11='/home/TomasV1.1/'
Path_UbuntuV11='/home/administrador/TomasV1.1/'

FedoraV1='/home/rnogales73/Documentos/Tesis/Tomas/Fedora/'
CentosV1='/home/rnogales73/Documentos/Tesis/Tomas/Centos/'
UbuntuV1='/home/rnogales73/Documentos/Tesis/Tomas/Ubuntu/'

FedoraV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Fedora/'
CentosV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Centos/'
UbuntuV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Ubuntu/'

echo ..... Inicia Copia
sleep 5s
scp $Cadena_Fedora:$Path_FedoraV1* $FedoraV1
scp $Cadena_Centos:$Path_CentosV1* $CentosV1
scp $Cadena_Ubuntu:$Path_UbuntuV1* $UbuntuV1

scp $Cadena_Fedora:$Path_FedoraV11* $FedoraV11
scp $Cadena_Centos:$Path_CentosV11* $CentosV11
scp $Cadena_Ubuntu:$Path_UbuntuV11* $UbuntuV11

echo ..... Generando archivos de datos
sleep 5s
path_script='/home/rnogales73/'
$path_script./genera.sh
echo ..... Copiando archivos de datos al estadistico
sleep 5s
path_script='/home/rnogales73/'
$path_script./copia_estadistico.sh
```

- Ejecución de procesos de memoria, CPU, disco y red sustentado por **Peña Ortiz (2013)**

```
#!/bin/bash
path_FedoraV1='/home/rnogales73/Documentos/Tesis/Tomas/Fedora/'
path_CentosV1='/home/rnogales73/Documentos/Tesis/Tomas/Centos/'
path_UbuntuV1='/home/rnogales73/Documentos/Tesis/Tomas/Ubuntu/'

path_FedoraV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Fedora/'
path_CentosV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Centos/'
path_UbuntuV11='/home/rnogales73/Documentos/Tesis/TomasV1.1/Ubuntu/'

```

```
10 cd $path_FedoraV1
11 ./process_cpu_files.sh
12 ./process_hd_files.sh
13 ./process_mem_files.sh
14 ./process_red_tx_files.sh
15 ./process_red_rx_files.sh
16
17 cd $path_CentosV1
18 ./process_cpu_files.sh
19 ./process_hd_files.sh
20 ./process_mem_files.sh
21 ./process_red_tx_files.sh
22 ./process_red_rx_files.sh
23
24 cd $path_UbuntuV1
25 ./process_cpu_files.sh
26 ./process_hd_files.sh
27 ./process_mem_files.sh
28 ./process_red_tx_files.sh
29 ./process_red_rx_files.sh
30
31 echo Inicia la generacion de la version 2
32
33 cd $path_FedoraV11
34 ./process_cpu_files.sh
35 ./process_hd_files.sh
36 ./process_mem_files.sh
37 ./process_red_tx_files.sh
38 ./process_red_rx_files.sh
39
40 cd $path_CentosV11
41 ./process_cpu_files.sh
42 ./process_hd_files.sh
43 ./process_mem_files.sh
44 ./process_red_tx_files.sh
45 ./process_red_rx_files.sh
46
47 cd $path_UbuntuV11
48 ./process_cpu_files.sh
49 ./process_hd_files.sh
50 ./process_mem_files.sh
51 ./process_red_tx_files.sh
52 ./process_red_rx_files.sh
```

- Consumo CPU en Centos server 6.5, Ubuntu LTS 14.04.1 y Fedora 20

```
1  #!/bin/bash
2  FILES=`ls CPU*`
3  TMP_VALUES="cpu_valuesV11_Centos.dat"
4  rm -f $TMP_VALUES
5  for FILE in $FILES
6  do
7  cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{printf("%3.3f\n",(100-$NF))
       }' >> $TMP_VALUES
8  done
9
10 cat $TMP_VALUES | awk 'BEGIN{i=0;tStudent99[1]=31.82;tStudent99[2]=6.965;
```

```
    tStudent99[3]=4.541;tStudent99[4]=3.747;tStudent99[5]=3.365;tStudent99
    [6]=3.143;tStudent99[7]=2.998;tStudent99[8]=2.896;tStudent99[9]=2.821;
    tStudent99[10]=2.764;tStudent99[11]=2.718;tStudent99[12]=2.681;
    tStudent99[13]=2.650;tStudent99[14]=2.624;tStudent99[15]=2.602;
    tStudent99[16]=2.583;tStudent99[17]=2.567;tStudent99[18]=2.552;
    tStudent99[19]=2.539;tStudent99[20]=2.528;tStudent99[21]=2.518;
    tStudent99[22]=2.508;tStudent99[23]=2.500;tStudent99[24]=2.492;
    tStudent99[25]=2.485;tStudent99[26]=2.479;tStudent99[27]=2.473;
    tStudent99[28]=2.467;tStudent99[29]=2.462;tStudent99[30]=2.457;Z99
    =2.575}{

i++;
values[i]=$1;
}END{
N=i;
for (i=1;i<=N;i++) {
    valuesSum+=values[i];
}
valuesAvg=valuesSum/N;
valuesVar=0;
for (i=1;i<=N;i++) {
    valuesDiff=values[i]-valuesAvg;
    valuesVar+=(valuesDiff*valuesDiff);
}
valuesVar=valuesVar/N;
if (N<=30) {
   valuesI = tStudent99[N-1] *sqrt(valuesVar)/sqrt(N);
} else {
   valuesI = Z99 *sqrt(valuesVar)/sqrt(N);
}
printf("%10.10f %10.10f\n",valuesAvg,valuesI);

}' > "cpu_V11_Centos.dat"
#rm -f  $TMP_VALUES
```

- Consumo de memoria en Centos server 6.5, Ubuntu LTS 14.04.1 y Fedora 20

```
#!/bin/bash
FILES=`ls Mem*`
TMP_VALUES="mem_valuesV11_Centos.dat"
rm -f $TMP_VALUES
for FILE in $FILES
do
cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{printf("%3.3f\n",($4))}' >>
    $TMP_VALUES
done

cat $TMP_VALUES | awk 'BEGIN{i=0;tStudent99[1]=31.82;tStudent99[2]=6.965;
    tStudent99[3]=4.541;tStudent99[4]=3.747;tStudent99[5]=3.365;tStudent99
    [6]=3.143;tStudent99[7]=2.998;tStudent99[8]=2.896;tStudent99[9]=2.821;
    tStudent99[10]=2.764;tStudent99[11]=2.718;tStudent99[12]=2.681;
    tStudent99[13]=2.650;tStudent99[14]=2.624;tStudent99[15]=2.602;
    tStudent99[16]=2.583;tStudent99[17]=2.567;tStudent99[18]=2.552;
    tStudent99[19]=2.539;tStudent99[20]=2.528;tStudent99[21]=2.518;
    tStudent99[22]=2.508;tStudent99[23]=2.500;tStudent99[24]=2.492;
    tStudent99[25]=2.485;tStudent99[26]=2.479;tStudent99[27]=2.473;
    tStudent99[28]=2.467;tStudent99[29]=2.462;tStudent99[30]=2.457;Z99
```

```
        =2.575}{
11
12  i++;
13  values[i]=$1;
14  }END{
15  N=i;
16  for (i=1;i<=N;i++) {
17      valuesSum+=values[i];
18  }
19  valuesAvg=valuesSum/N;
20  valuesVar=0;
21  for (i=1;i<=N;i++) {
22      valuesDiff=values[i]-valuesAvg;
23      valuesVar+=(valuesDiff*valuesDiff);
24  }
25  valuesVar=valuesVar/N;
26  if (N<=30) {
27     valuesI = tStudent99[N-1] *sqrt(valuesVar)/sqrt(N);
28  } else {
29     valuesI = Z99 *sqrt(valuesVar)/sqrt(N);
30  }
31  printf("%10.10f %10.10f\n",valuesAvg,valuesI);
32  }' > "mem_V11_Centos.dat"
33  #rm -f  $TMP_VALUES
```

- Consumo de Disco en Centos server 6.5, Ubuntu LTS 14.04.1 y Fedora 20

```
1   #!/bin/bash
2   FILES=`ls HD*`
3   TMP_VALUES="hd_valuesV11_Centos.dat"
4   rm -f $TMP_VALUES
5   for FILE in $FILES
6   do
7   cat $FILE | sed "s/,/\./g" |sed '/ dev253-1/d'| awk 'BEGIN{}{}END{printf("
        %3.3f\n",($NF))}' >> $TMP_VALUES
8   done
9
10  cat $TMP_VALUES | awk 'BEGIN{i=0;tStudent99[1]=31.82;tStudent99[2]=6.965;
        tStudent99[3]=4.541;tStudent99[4]=3.747;tStudent99[5]=3.365;tStudent99
        [6]=3.143;tStudent99[7]=2.998;tStudent99[8]=2.896;tStudent99[9]=2.821;
        tStudent99[10]=2.764;tStudent99[11]=2.718;tStudent99[12]=2.681;
        tStudent99[13]=2.650;tStudent99[14]=2.624;tStudent99[15]=2.602;
        tStudent99[16]=2.583;tStudent99[17]=2.567;tStudent99[18]=2.552;
        tStudent99[19]=2.539;tStudent99[20]=2.528;tStudent99[21]=2.518;
        tStudent99[22]=2.508;tStudent99[23]=2.500;tStudent99[24]=2.492;
        tStudent99[25]=2.485;tStudent99[26]=2.479;tStudent99[27]=2.473;
        tStudent99[28]=2.467;tStudent99[29]=2.462;tStudent99[30]=2.457;Z99
        =2.575}{
11
12  i++;
13  values[i]=$1;
14  }END{
15  N=i;
16  for (i=1;i<=N;i++) {
17      valuesSum+=values[i];
18  }
19  valuesAvg=valuesSum/N;
```

```
20  valuesVar=0;
21  for (i=1;i<=N;i++) {
22      valuesDiff=values[i]-valuesAvg;
23      valuesVar+=(valuesDiff*valuesDiff);
24  }
25  valuesVar=valuesVar/N;
26  if (N<=30) {
27     valuesI = tStudent99[N-1] *sqrt(valuesVar)/sqrt(N);
28  } else {
29     valuesI = Z99 *sqrt(valuesVar)/sqrt(N);
30  }
31  printf("%10.10f %10.10f\n",valuesAvg,valuesI);
32  }' > "hd_V11_Centos.dat"
33  #rm -f  $TMP_VALUES
```

- Consumo de Red bytes transmitidos, en Centos server 6.5, Ubuntu LTS 14.04.1 y Fedora 20

```
1   #!/bin/bash
2   FILES=`ls RED*`
3   TMP_VALUES="net_tx_valuesV11_Centos.dat"
4   rm -f $TMP_VALUES
5   for FILE in $FILES
6   do
7   #cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{printf("%3.3f\n",(100-$NF))
        }' >> $TMP_VALUES
8   cat $FILE| sed "s/,/\./g" |egrep -A3  IFACE|egrep "Media:"|egrep -v ^$|
        egrep -A3 rxkB|egrep eth0|awk '{print $6}' >> $TMP_VALUES
9   done
10
11  cat $TMP_VALUES | awk 'BEGIN{i=0;tStudent99[1]=31.82;tStudent99[2]=6.965;
        tStudent99[3]=4.541;tStudent99[4]=3.747;tStudent99[5]=3.365;tStudent99
        [6]=3.143;tStudent99[7]=2.998;tStudent99[8]=2.896;tStudent99[9]=2.821;
        tStudent99[10]=2.764;tStudent99[11]=2.718;tStudent99[12]=2.681;
        tStudent99[13]=2.650;tStudent99[14]=2.624;tStudent99[15]=2.602;
        tStudent99[16]=2.583;tStudent99[17]=2.567;tStudent99[18]=2.552;
        tStudent99[19]=2.539;tStudent99[20]=2.528;tStudent99[21]=2.518;
        tStudent99[22]=2.508;tStudent99[23]=2.500;tStudent99[24]=2.492;
        tStudent99[25]=2.485;tStudent99[26]=2.479;tStudent99[27]=2.473;
        tStudent99[28]=2.467;tStudent99[29]=2.462;tStudent99[30]=2.457;Z99
        =2.575}{
12
13  i++;
14  values[i]=$1;
15  }END{
16  N=i;
17  for (i=1;i<=N;i++) {
18      valuesSum+=values[i];
19  }
20  valuesAvg=valuesSum/N;
21  valuesVar=0;
22  for (i=1;i<=N;i++) {
23      valuesDiff=values[i]-valuesAvg;
24      valuesVar+=(valuesDiff*valuesDiff);
25  }
26  valuesVar=valuesVar/N;
```

```
27  if (N<=30) {
28     valuesI = tStudent99[N-1] *sqrt(valuesVar)/sqrt(N);
29  } else {
30     valuesI = Z99 *sqrt(valuesVar)/sqrt(N);
31  }
32  printf("%10.10f %10.10f\n",valuesAvg,valuesI);
33  }' > "net_tx_V11_Centos.dat"
34  #rm -f  $TMP_VALUES
```

- Consumo de Red bytes recibidos, en Centos server 6.5, Ubuntu LTS 14.04.1 y Fedora 20

```
1  #!/bin/bash
2  FILES=`ls RED*`
3  TMP_VALUES="net_rx_valuesV11_Centos.dat"
4  rm -f $TMP_VALUES
5  for FILE in $FILES
6  do
7  #cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{printf("%3.3f\n",(100-$NF))
       }' >> $TMP_VALUES
8  cat $FILE | sed "s/,/\./g" |egrep -A3  IFACE|egrep "Media:"|egrep -v ^$|
       egrep -A3 rxkB|egrep eth0 |awk '{print $5}' >> $TMP_VALUES
9  done
10
11 cat $TMP_VALUES | awk 'BEGIN{i=0;tStudent99[1]=31.82;tStudent99[2]=6.965;
       tStudent99[3]=4.541;tStudent99[4]=3.747;tStudent99[5]=3.365;tStudent99
       [6]=3.143;tStudent99[7]=2.998;tStudent99[8]=2.896;tStudent99[9]=2.821;
       tStudent99[10]=2.764;tStudent99[11]=2.718;tStudent99[12]=2.681;
       tStudent99[13]=2.650;tStudent99[14]=2.624;tStudent99[15]=2.602;
       tStudent99[16]=2.583;tStudent99[17]=2.567;tStudent99[18]=2.552;
       tStudent99[19]=2.539;tStudent99[20]=2.528;tStudent99[21]=2.518;
       tStudent99[22]=2.508;tStudent99[23]=2.500;tStudent99[24]=2.492;
       tStudent99[25]=2.485;tStudent99[26]=2.479;tStudent99[27]=2.473;
       tStudent99[28]=2.467;tStudent99[29]=2.462;tStudent99[30]=2.457;Z99
       =2.575}{
12
13 i++;
14 values[i]=$1;
15 }END{
16 N=i;
17 for (i=1;i<=N;i++) {
18     valuesSum+=values[i];
19 }
20 valuesAvg=valuesSum/N;
21 valuesVar=0;
22 for (i=1;i<=N;i++) {
23     valuesDiff=values[i]-valuesAvg;
24     valuesVar+=(valuesDiff*valuesDiff);
25 }
26 valuesVar=valuesVar/N;
27 if (N<=30) {
28    valuesI = tStudent99[N-1] *sqrt(valuesVar)/sqrt(N);
29 } else {
30    valuesI = Z99 *sqrt(valuesVar)/sqrt(N);
31 }
32 printf("%10.10f %10.10f\n",valuesAvg,valuesI);
```

```
33
34  }' > "net_rx_V11_Centos.dat"
35  #rm -f  $TMP_VALUES
```

- Capturar la media de las medias generadas con los scripts descritos, y obtención de valores necesariso para las gráficas correspondientes.

```
1   #!/bin/bash
2   path_graphs_estadistico='/home/rnogales73/Documentos/Tesis/Estadistico/
        graphs/'
3   FILES=`ls mem_V*`
4   MEM_VALUES="mem.dat"
5   rm -f $MEM_VALUES
6   for FILE in $FILES
7   do
8   #cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{printf("%3.3f\n",($1))}' >>
        $MEM_VALUES
9   cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{print $1 , $2}' >>
        $MEM_VALUES
10  done
11  FILES=`ls cpu_V*`
12  MEM_VALUES="cpu.dat"
13  rm -f $MEM_VALUES
14  for FILE in $FILES
15  do
16  cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{print $1 , $2}' >>
        $MEM_VALUES
17  done
18  FILES=`ls hd_V*`
19  MEM_VALUES="hd.dat"
20  rm -f $MEM_VALUES
21  for FILE in $FILES
22  do
23  cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{print $1 , $2}' >>
        $MEM_VALUES
24  done
25  FILES=`ls net_rx_V*`
26  MEM_VALUES="net_rx.dat"
27  rm -f $MEM_VALUES
28  for FILE in $FILES
29  do
30  cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{print $1 , $2}' >>
        $MEM_VALUES
31  done
32  FILES=`ls net_tx_V*`
33  MEM_VALUES="net_tx.dat"
34  rm -f $MEM_VALUES
35  for FILE in $FILES
36  do
37  cat $FILE | sed "s/,/\./g" | awk 'BEGIN{}{}END{print $1 , $2}' >>
        $MEM_VALUES
38  done
```

- Scripts para graficar.
  - script1 y scrip2

Script 1

```
#!/bin/bash
function plot {
cat $PLT_TEMPLATE | sed "s/OUTPUTFILE/$2/g" | sed "s/INPUTFILE/$1/g" | sed
    "s/TITLE/$3/g" | sed "s/UNITS/$4/g" | sed "s/MAXY/$5/g"  > tmp.metric.
    plt

gnuplot  tmp.metric.plt

ps2pdf $2.ps
rm -f $2.ps
rm -f tmp.metric.plt
}
PLT_TEMPLATE="metric_graph.plt"
plot "cpu.dat" "cpu" "CPU Utilization by Operating System and Algorithm" "
    CPU Utilization (%)" ""
plot "mem.dat" "mem" "Memory Utilization by Operating System and Algorithm"
     "Memory Utilization (%)" ""
plot "disc_rd.dat" "disc_rd" "Disc Utilization by Operating System and
    Algorithm" "HD Utilization (%)Kbits\/second" ""
plot "net_rx.dat" "net_rx" "Network Utilization rxKB by Operating System
    and Algorithm" "Net Utilization (%)Kbits\/second" ""
plot "net_tx.dat" "net_tx" "Network Utilization txKB by Operating System
    and Algorithm" "Net Utilization (%)Kbits\/second" ""
```

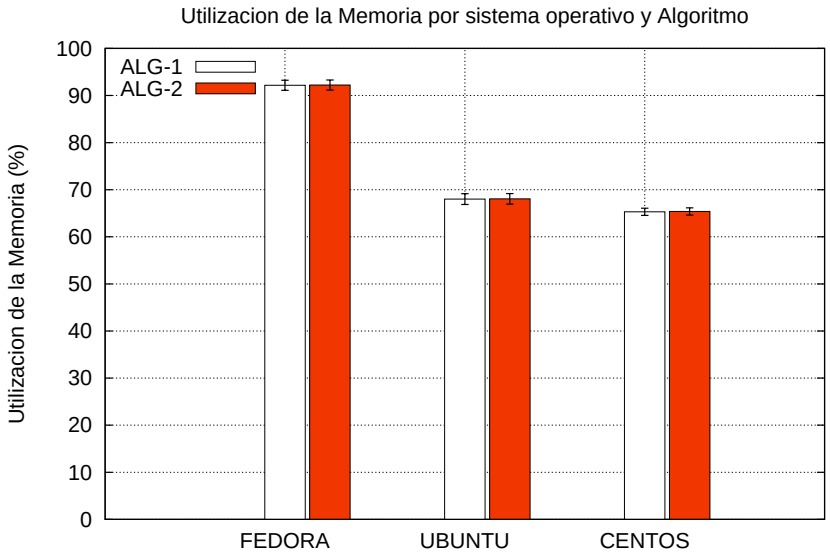
Script 2

```
#!/bin/bash
set terminal postscript color solid 18
set output "OUTPUTFILE.ps"
set key left top
set boxwidth 0.9 absolute
set style fill solid 1.00 border -1
set datafile missing '-'
set style data histograms
#set style histogram clustered gap 1 title  offset character 0, 0, 0
set style histogram errorbars gap 2 lw 2
set xtics border in scale 1,0.5 nomirror offset character 0, 0, 0
set xtics(" " 0.00000)
set title "TITLE"
set xlabel " "
set ylabel "UNITS"
set grid
set autoscale y
set yrange [0:MAXY]
plot "INPUTFILE" using 2:3:xtic(1) ti col fs solid lc rgb "#FFFFFF", '' u
    4:5 ti col fs solid lc rgb "#f33500"
reset
```

6.8.4. Obtención de resultados.

Memoria.

La utilización de la memoria en los sistemas operativos y en los algoritmos es simétrica, en Fedora 20 90 % de utilización, en Ubuntu LTS 14.04.1 el 68 % y en Centos server 6.5 el 65 %, en tanto que las variables instanciadas son las mismas necesarias para el algoritmo y la reserva de memoria es la misma en las dos versiones.

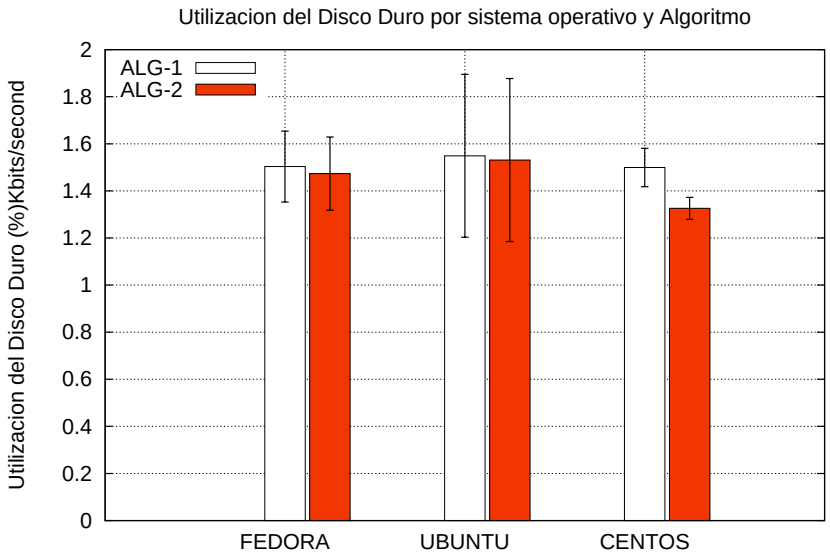


Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 24: Comparativo de Versiones del Algoritmo en los sistemas operativos

Disco.

La utilización y compotamiento del Disco es similar a la memoria en los tres sistemas operativos y en las dos versiones de algoritmo son similares, siendo el consumo de este recurso casi despreciable. En Fedora 20 y Ubuntu LTS 14.04.1 la utilización del Disco es de 1,5 % en Centos server 6.5 la versión 1 del algoritmo es de 1,5 % y con la versión 2 es de 1,3 %.

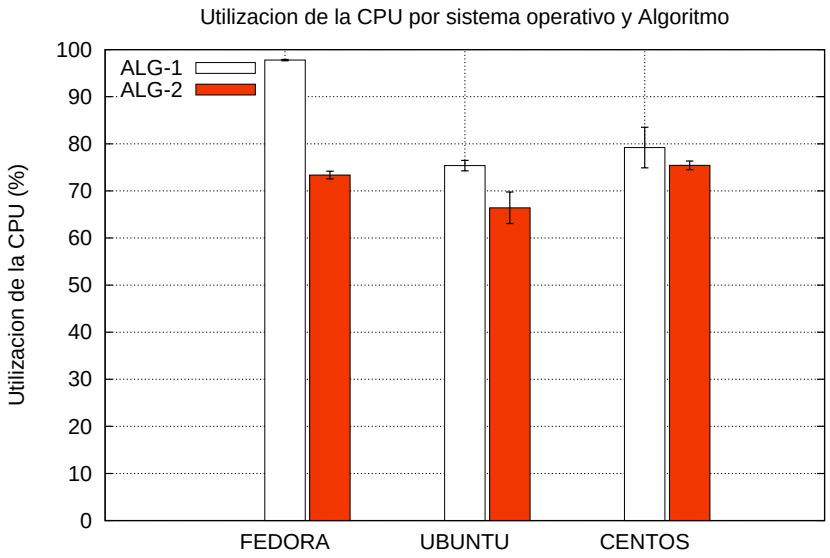


Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 25: Comparativo de Versiones del Algoritmo en los sistemas operativos

CPU.

La utilización del CPU en Fedora 20 con el algoritmo versión 1 de programación alcanza una saturación del 98 % mientras que con el algoritmo versión 2 el rendimiento disminuye a un 72 % la optimización del código es altamente visible. En Ubuntu LTS 14.04.1 la versión 1 utiliza 78 % del CPU en su procesamiento, con la versión 2 se tiene una utilización del 68 % y en Centos server 6.5 el rendimiento es del 80 % de la versión 1 versus el 75 % de la versión 2.



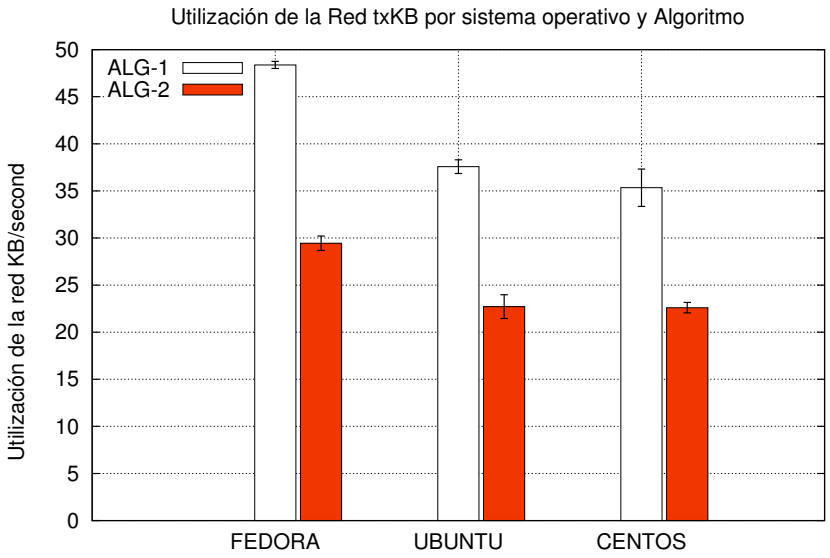
Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 26: Comparativo de Versiones del Algoritmo en los sistemas operativos

Red.

Bytes transmitidos.

La transmisión de paquetes en el algoritmo se realiza a través de broadcast y se obtiene que en Fedora 20 con el algoritmo versión1 utiliza la red en un 48 % mientras que con el algoritmo versión 2 su utilización disminuye al 29 %. La utilización de la red en Ubuntu LTS 14.04.1 con la versión 1 del algoritmo alcanza un 38 % con la versión 2 la utilización de la res es del 23 %, y en Centos server 6.5 con la versión 1 del algoritmo la utilización es del 35 % mientras que con la versión 2 es del 22 %.

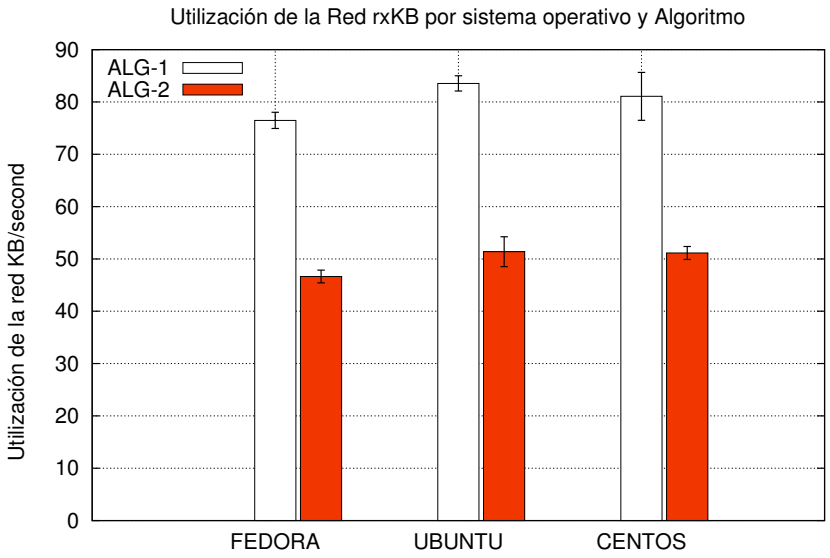


Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 27: Comparativo de Versiones del Algoritmo en los sistemas operativos

Bytes recibidos.

La recepción de paquetes al igual que la transmisión se realiza a través de broadcast y se obtiene que en Fedora 20 con el algoritmo versión 1 utiliza la red en un 78 % mientras que con el algoritmo versión 2 su utilización disminuye al 48 %. La utilización de la red en Ubuntu LTS 14.04.1 con la versión 1 del algoritmo alcanza un 82 % con la versión 2 la utilización de la red es del 51 %, y en Centos server 6.5 con la versión 1 del algoritmo la utilización es del 80 % mientras que con la versión 2 es del 50 %.



Fuente: Proceso de experimentación.
Autor: Rubén Nogales Portero.

Figura 28: Comparativo de Versiones del Algoritmo en los sistemas operativos

6.9. DISCUSIÓN

La experiencia investigativa con:

1. La programación del algoritmo de consenso $A\Omega'$ en su primera versión utilizando programación lineal.
2. Programación del algoritmo de consenso $A\Omega'$ segunda versión utilizando diferentes técnicas de programación.
3. Tomas de datos con retardo de 5 segundos simulando 12 conexiones por minuto.
4. Tomas de datos con retardo de 0 segundos simulando 7800 conexiones por minuto.

Es concluyente en razón de:

- La utilización de buenas prácticas de programación mejoran el rendimiento del hardware y red en el algoritmo de consenso $A\Omega'$.

- Se demuestra estadísticamente que la cantidad de tomas de datos en cada uno de los experimentos es suficiente según los intervalos de confianza expuestos en los gráficos.
- Los datos obtenidos en el proceso de toma de datos para la utilización de la memoria y del disco duro no son relevantes, por tanto son despreciables debido a:
    - Memoria.- El número de variables utilizado en las versiones del algoritmo de consenso $A\Omega'$ son las mismas, el tipo de datos de igual manera son los mismo por tanto la memoria reservada por los sistemas operativos es la misma.
    - Disco Duro.- Existe una diferencia mínima despreciable puesto que las técnicas de programación casi no disminuyen la cantidad de lineas de código y los llamados a ejecución del mismo no son itinerantes.
- Se demuestra la incidencia de los valores correspondientes por tanto.
    - CPU.

	Algoritmo Versión 1	Algoritmo Versión 2	Diferencia
Fedora 20	98 %	72 %	26 %
Ubuntu LTS 14.04.1	78 %	68 %	10 %
Centos server 6.5	80 %	75 %	5 %

Tabla 9: Comparativo de algoritmos y diferencia de rendimiento

- RED.
- Tx

	Algoritmo Versión 1	Algoritmo Versión 2	Diferencia
Fedora 20	48 %	29 %	19 %
Ubuntu LTS 14.04.1	38 %	23 %	15 %
Centos server 6.5	35 %	22 %	13 %

Tabla 10: Comparativos de algoritmos y diferencias de rendimiento

- Rx

	Algoritmo Versión 1	Algoritmo Versión 2	Diferencia
Fedora 20	78 %	48 %	30 %
Ubuntu LTS 14.04.1	82 %	51 %	31 %
Centos server 6.5	80 %	50 %	30 %

Tabla 11: Comparativos de algoritmos y diferencias de rendimiento

BIBLIOGRAFIA

Aguilera, M. K., Chen, W., and Toueg, S. (2000). Failure detection and consensus in the crash-recovery model. *Distributed computing*, 13(2):99–125.

Arévalo, S., Fernández Anta, A., Imbs, D., Jiménez, E., and Raynal, M. (2012). Failure detectors in homonymous distributed systems (with an application to consensus). In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 275–284. IEEE.

Attiya, H. and Welch, J. (2004). *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

Computing, C. (2010). Cloud computing.

Franco, Y. (2011). Investigación de campo. manual upel. *2011: Internet*.

Godard, S. (2010). Sysstat utilities home page.

HERRERA (2004). *Tutoría de la Investigación*. Number 252.

Ibou, H. M. and Marquezan, C. (2013). Openstack: The cloud os.

Kalagiakos, P. and Karampelas, P. (2011). Cloud computing learning. In *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pages 1–4.

LEY DE COMERCIO ELECTRONICO, F. Y. M. D. D. (2011). Ley de comercio electronico, firmas y mensajes de datos. Technical report.

Lynch, N. A. (1996). *Distributed algorithms*. Morgan Kaufmann.

Martin, C., Larrea, M., and Jimenez, E. (2007). On the implementation of the omega failure detector in the crash-recovery failure model. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 975–982.

Moroni, N. and Señas, P. (2005). Estrategias para la enseñanza de la programación. In *I Jornadas de Educación en Informática y TICs en Argentina*.

Nino, L., Ardila, E., and Sanchez, J. (2013). Congestion control model for local ip networks. In *Communications and Computing (COLCOM), 2013 IEEE Colombian Conference on*, pages 1–6.

Omar, N. and Razik, N. (2008). Determining the basic elements of object oriented programming using natural language processing. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 3, pages 1–6.

Peña Ortiz, R. (2013). Accurate workload design for web performance evaluation.

Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W., and Li, Q. (2009). Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27.

Pita Fernández S, P. D. S. (2002). Investigación cuantitativa y cualitativa.

Raynal, M. (2010). Fault-tolerant agreement in synchronous message-passing systems. *Synthesis Lectures on Distributed Computing Theory*, 1(1):1–189.

Roberto, H. S., Franco, Collado, F., and Pilar, B. L. (2010). MetodologÍa dela investigaciÓn. *Mexico: Editorial Mc. Graw Hill*.

Rutti, O., Milosevic, Z., and Schiper, A. (2010). Generic construction of consensus algorithms for benign and byzantine faults. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 343–352.

Sabino, C. (1992). El proceso de investigación. Technical report.

Van Emden, M. H. (1997). Object-oriented programming as the end of history in programming languages. In *Communications, Computers and Signal Processing, 1997. 10 Years PACRIM 1987-1997 - Networking the Pacific Rim. 1997 IEEE Pacific Rim Conference on*, volume 2, pages 981–984 vol.2.