VICERRECTORADO DE INVESTIGACIÓN INNOVACIÓN Y TRANSFERENCIA TECNOLÓGICA

MAESTRÍA EN REDES DE INFORMACIÓN Y CONECTIVIDAD III

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN REDES DE INFORMACIÓN Y CONECTIVIDAD

TEMA:

¨PROPUESTA PARA INCREMENTAR EL DESEMPEÑO EN UNA INFRAESTRUCTURA VIRTUALIZADA¨

AUTOR: NÚÑEZ SOLÍS MARITZA ALEXANDRA

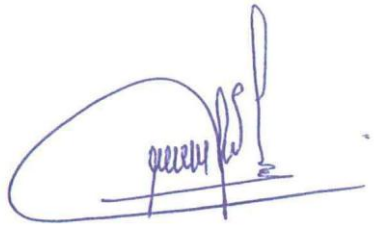
DIRECTOR: FUERTES DIAZ WALTER

ECUADOR - SANGOLQUÍ

2015

CERTIFICADO

Certifico que el presente trabajo fue desarrollado por la Ing. Maritza Alexandra Núñez Solís, bajo mi dirección.



Dr. Walter Fuertes, PhD.

Director de Tesis

AUTORÍA DE RESPONSABILIDAD

Yo, Maritza Alexandra Núñez Solís, declaro que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o calificación profesional; y, que se han consultado las referencias bibliográficas que se incluye en este documento.

Ing. Maritza Alexandra Núñez Solís

AUTORIZACIÓN

Yo, Maritza Alexandra Núñez Solís, autorizo a la Universidad de las Fuerzas Armadas ESPE, publicar la tesis que tiene como título ¨ PROPUESTA PARA INCREMENTAR EL DESEMPEÑO EN UNA INFRAESTRUCTURA VIRTUALIZADA¨, en el repositorio público de la ESPE.

Ing. Maritza Alexandra Núñez Solís

DEDICATORIA

A mis padres y hermanos.

Siempre me he sentido feliz por la familia que tengo, siempre me han formado para salir victoriosa ante las adversidades de la vida, sus enseñanzas y consejos nunca han cesado y hoy estoy aquí con un nuevo logro. A ustedes padres queridos, a ustedes hermanos adorados, Andrea y Luis; les dedico este trabajo por todo su amor y su apoyo incondicional, por haber creído en mí en todo momento sin titubear.

A mis amigos.

Siempre he pensado que los amigos, los buenos amigos, son la familia que uno escoge. Con mucho cariño para todos mis amigos quienes fueron un gran apoyo emocional durante el tiempo en que escribía esta tesis.

AGRADECIMIENTO

En primer lugar agradezco a Dios ya que sin su venia bendita nada sería posible. Agradezco a mis padres por ser ejemplo de perseverancia, responsabilidad y sacrificio, valores que me ayudaron a elaborar el presente trabajo de investigación.

A mi tutor de tesis Ing. Walter Fuertes, por el arduo trabajo de transmitirme sus conocimientos para la culminación exitosa de mi tesis. Gracias a su orientación paciencia y motivación hoy cumplo una meta muy importante en mi vida.

ÍNDICE DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

Virtualizar una infraestructura de TI ha permitido reducir costos de despliegue y gestión de lo que sería un sistema equivalente realizado de forma convencional con equipos reales. Sin embargo, debido a esto se ha sumado una nueva responsabilidad y preocupación para los administradores de red, que es el de garantizar el eficiente desempeño de la red y que la eficiencia de los servicios virtualizados no disminuya. Es preciso señalar que la virtualización provoca una penalización provocada por la capa de virtualización conocida como (overhead). Este artículo presenta una propuesta para disminuir dicha penalización, utilizando entornos virtuales de red. Para llevarlo a cabo se diseñó e implementó una topología de experimentación, la cual estuvo conformada por 6 máquinas virtuales en donde se desarrollaron tanto, las pruebas de saturación a los servidores con el fin de medir el consumo del procesador y la memoria RAM, como los mecanismos de gestión de recursos computacionales propuestos como son: la utilización de hilos, la configuración de los parámetros de red mediante un emulador y la detección y eliminación de programas zombis; que permitieron contrarrestar la penalización provocada por la capa de virtualización y así incrementar el desempeño en infraestructuras virtualizadas. Los resultados muestran la funcionalidad de dicha propuesta.

PALABRAS CLAVES:

- **VIRTUALIZACIÓN**
- **OVERHEAD**
- **PROCESADOR**
- **MEMORIA RAM**
- **MECANISMOS DE GESTIÓN.**

ABSTRACT

Virtualizing an IT infrastructure has allowed reducing costs of deployment and management of what would be an equivalent system performed conventionally with real equipment. However, due to this has been added a new responsibility and concern for network administrators, that is the ensure the efficient performance of the network and the efficiency of virtualized services does not decrease. It should be noted that virtualization causes a penalty provoked by the layer of virtualization known as (overhead). This article presents a proposal to reduce the penalty, using virtual network environments. To carry this work, it has been designed and implemented a topology of experimentation, which was formed by 6 virtual machines where developed, proofs of saturation to the servers in order to measure the consumption of the processor and RAM both mechanisms of management of computing resources are proposed: the use of threads, the configuration of network parameters via an emulator and the detection and elimination of zombies programs; that he was allowed to counteract the penalty caused by the layer of virtualization and thus increase performance in virtualized infrastructures. The results show the functionality of such a proposal.

KEY WORDS

- **VIRTUALIZATION**
- **OVERHEAD**
- **PROCESSOR**
- **RAM MEMORY**
- **MANAGEMENT MECHANISMS**

CAPITULO I INTRODUCCIÓN

1.1 Introducción

Hoy en día la tecnología ha dado un salto importante en utilizar las tecnologías de virtualización para aumentar la disponibilidad de los recursos de TI y aplicaciones, eliminando así el antiguo modelo de una aplicación por servidor, permitiendo que el personal de TI invierta tiempo en innovación en lugar de la administración de servidores. (Arias, 2011).

La utilización de la virtualización ha aumentado considerablemente en estos últimos años, debido principalmente a la disponibilidad de herramientas que permiten su manejo, además de la incorporación de procesadores y circuitería específica para soportar esta tecnología. (Valls, S. C., Castelló, F. J. C., Mayo, R., & Quintana-Ortí, E. S.) .

Virtualizar una infraestructura de TI ha permitido reducir costos de despliegue y gestión de lo que sería un sistema equivalente realizado de forma convencional con equipos reales, pero debido a esto se ha sumado una nueva responsabilidad y preocupación para los administradores de red, que es el de garantizar el eficiente desempeño de la red y que la eficiencia de los servicios virtualizados no se reduzca, debido a la penalización provocada por la capa virtual (overhead).

1.2 Justificación e Importancia

La disminución de costos de inversión de hardware, la disminución del precio de mantenimiento, el aumento de los servicios, aplicaciones y el aprovechamiento del desempeño de una infraestructura de TI, son aspectos que el personal de tecnología busca y que pueden conseguirse con las infraestructuras virtualizadas cuya utilización ha

madurado durante la última década (Qureshi, 2007) . Sin embargo aún persiste la pérdida del desempeño por la penalización provocada por la capa de virtualización.

La virtualización desde el punto de vista técnico resuelve el problema del desaprovechamiento de recursos del servidor, al ofrecer la posibilidad de que diferentes aplicaciones funcionen de modo independiente sin necesidad de un servidor físico para cada servicio. Además, los eventos que ocurran dentro de un servidor virtual no tendrán ningún impacto sobre otro, gracias al aislamiento, característica que significa que un fallo general de sistema de una máquina virtual no afecta al resto de las máquinas virtuales. (Cerrada, 2012).

Algunas de las ventajas citadas anteriormente pasan a un segundo plano al momento que se analiza el desempeño de la infraestructura de TI que ha sido virtualizada, ya que por el uso de esta tecnología se introduce una penalización u overhead debido al consumo de recursos, afectando al desempeño del entorno de red.

Por lo tanto, garantizar un desempeño eficiente en un entorno virtualizado de red permitirá que todas las aplicaciones centralizadas en dicho entorno funcionen eficientemente, dando como resultado el incremento en la velocidad de la infraestructura de TI.

Por lo expuesto, esta investigación pretende buscar una solución que permita incrementar el desempeño de un entorno virtualizado, ya que hasta hora la mayor parte de proyectos de virtualización han estado encaminados hacia el segmento de consolidación de servidores y hacia la implementación de herramientas de computación en nube. Sin embargo, los proyectos en los que se plantea la utilización de la virtualización como una herramienta dentro de la computación de altas prestaciones no han implementado soluciones reales. (Valls et al.)

1.3 Planteamiento del problema

El desempeño de una red es un tema de preocupación permanente para los profesionales de TI. En este contexto se debe buscar una solución unificada, eficiente y segura para gestionar servicios y aplicaciones aprovechando los recursos de ésta infraestructura.

Sin embargo, una infraestructura de TI virtualizada no tiene el mismo desempeño que si se utiliza equipos físicos, debido a una capa intermedia de virtualización que produce una penalización llamada overhead en las máquinas virtuales, para gestionar las peticiones de acceso y la concurrencia del mismo. Esto provoca la pérdida de información y la disminución del desempeño en la prestación de servicios; y, la confiabilidad.

1.4 Formulación del Problema

La virtualización es una tecnología que va creciendo día a día. El aspecto económico está firmemente detrás de esta tendencia; pero a pesar de este particular, existe un gran problema que no ha sido resuelto y es la penalización provocada por la capa de virtualización (overhead).

Por lo expuesto anteriormente, la realización de esta tesis, tiene como meta proponer mecanismos técnicos de gestión de recursos computacionales para incrementar el desempeño en infraestructuras virtualizadas.

1.5 Hipótesis

Si se diseña y desarrolla mecanismos de gestión de recursos computacionales para mejorar el comportamiento del CPU y la memoria; basado en métricas de desempeñó de

redes IP o de sistemas computacionales, entonces se puede incrementar el desempeño en una infraestructura virtualizada.

1.6 Objetivo general

Implementar una propuesta que permita incrementar el desempeño en una infraestructura virtualizada, mediante la evaluación sistémica y la implementación de mecanismos técnicos de gestión de recursos computacionales.

1.7 Objetivos específicos

1.7.1 Analizar el marco teórico referencial y el estado del arte relacionado al desempeño en infraestructuras virtualizadas.

1.7.2 Diseñar, implementar y poner en funcionamiento un entorno virtualizado experimental para evaluar el desempeño en su línea base.

1.7.3 Diseñar e implementar mecanismos técnicos de gestión de recursos computacionales para incrementar el desempeño del CPU y memoria en entornos virtualizados.

1.7.4 Verificar, validar y evaluar los resultados mediante emulación, simulación y procesamiento estadístico.

CAPITULO II MARCO TEORICO

2.1 Antecedentes del estado del arte

(Fuertes, Jácome, Grijalva, López de Vergara y Fonseca, 2009), en su estudio comparativo al evaluar el rendimiento de la red tanto en un entorno simulado como virtualizado, detectaron que en el caso de la simulación los parámetros deben ser rigurosamente programados para mejorarlos, mientras que en la virtualización, se deben adaptar otras condiciones operacionales como servidores dedicados, temporización y el mejoramiento del hardware base. (Fuertes, Vilac y Gallo)**,** en su evaluación del experimento para la aplicación de técnicas de consolidación de servidores y virtualización de aplicaciones se fundamentó en el rendimiento de la red, el consumo de CPU y memoria, el tiempo de respuesta y la carga de equipos soportados por parte del servidor, donde para dicha evaluación emplearon la herramienta SAR[1]. Explican que se pudo observar un buen desempeño de los equipos clientes; sin embargo el performance de la red disminuyó considerablemente cuando todos los equipos hicieron uso de aplicaciones con multimedia y navegación Web concurrentemente. (Quintero y Silva, 2013), en los resultados de la medición del desempeño de un servidor web virtual, indican que existe una disminución del ancho de banda en el ambiente virtual, pero no es tan significativo puesto que los usuarios se conectan al mismo servidor, con respecto al uso de la memoria RAM demuestran que en un ambiente virtualizado se puede mejorar el rendimiento de la aplicación virtualizada, pues mencionan que, en una computadora física usó un 88 % de los recursos de RAM, mientras que en el ambiente virtualizado usó un 51%, demostrando que en ambientes virtualizados podemos ganar mejoras,

[1] Herramientas de monitoreo en Linux. Recuperado de: http://oracleexperiences.wikispaces.com/file/view/monitorizar_sistemas_linux.pdf. . Última comprobación 10/10/2013.

[2]Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., & Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.

porque tenemos RAM disponible para los demás procesos. En lo que se refiere a la utilización del disco duro mencionan que existe una mejora en el rendimiento de lectura de sectores en disco duro, pues tienen poco uso de recursos pero mayor cantidad de sectores que se han leído, es decir, menor esfuerzo por mayor trabajo realizado. Finalmente se hace referencia al uso del procesador donde indican que el este bajó a menos de la mitad del ambiente real. Por lo cual tienen un mejor uso del procesador. (Gad, Kappes, Mueller-Bady, y Ritter, 2011), en los resultados obtenidos en su investigación mencionan que el rendimiento en redes virtuales pueden será alto o bajo, esto dependen altamente de su implementación y de la tecnología de virtualización elegida, además demuestran que las características cuantitativas medidas en su investigación como son tiempo de retorno, la variación del retardo y la pérdida de paquetes difieren significativamente con la implementación de redes reales. Indican también que en el caso de, que el rendimiento de la red virtualizada sea inferior, afinar la configuración de red virtualizada puede ser factible en algunos casos después de un análisis minucioso. Por otro lado indican si el rendimiento de la red virtualizada es alto, en cuyo caso especial, deben tomarse medidas para estrangular artificialmente el tráfico, para que coincida con la velocidad de una red física. Las pruebas de evaluación la realizaron en diferentes plataformas, entre ellas XEN[2], mediante la técnica de virtualización completa. **(**Li, Hao, Xiao y Xu, 2009), en su investigación para mejorar la virtualización de un entorno de red mediante la utilización de la tecnología de virtualización KVM, realizaron varias pruebas donde el rendimiento de la red cae drásticamente, por lo que presentan dos técnicas para mejorar el rendimiento de la red virtualizada. La primera hace referencia al rediseño de la arquitectura de la interfaz virtual y el uso de un método directo para la trasmisión y recepción de paquetes en lugar de algunos dispositivos virtuales y también el soporte de múltiples máquinas virtuales compartiendo los paquetes de red del host anfitrión, lo cual previene la copia innecesaria de paquetes, realizando una mejora en la eficiencia de la trayectoria de virtualización

[2]Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., & Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.

conectando la interfaz de red virtual y física. En la segunda técnica utilizan mapas de memoria fija y la tecnología de copia zero, lo cual optimiza el mecanismo de transferencia de datos entre las máquinas virtuales y el host anfitrión. Evitando copiar dos veces la memoria durante el trayecto de la transmisión y recepción. El mapa de memoria de página fijo la tecnología de copia zero resulta ser más barato porque el tamaño de los datos es mucho más pequeño que el tamaño de página. (Menon, Cox, y Zwaenepoel, 2006), en su investigación proponen tres técnicas para optimizar el rendimiento de la red en un entorno virtualizado con XEN. En primer lugar, redefinen las interfaces de red virtuales de los dominios huésped para incorporar características offload de red de alto nivel. Aquí demuestran las ventajas de rendimiento de la funcionalidad de offload de alto nivel en la interfaz virtual, incluso cuando dicha funcionalidad no está disponible en la interfaz física subyacente. En segundo lugar, optimizan la aplicación de la trayectoria de transferencia de datos entre el huésped y dominios. La optimización evita la cara reasignación de datos en el trayecto de transmisión y reemplaza la página de reasignación de copia de datos en el trayecto de recepción. Por último, mencionan el soporte para los sistemas operativos invitados para utilizar de manera efectiva las funciones de memoria virtual avanzada como superpages y las asignaciones de páginas globales. (Regola, y Ducom, 2010), realizan una evaluación sobre varias plataformas de virtualización entre ellas XEN, llegando a la conclusión de que la virtualización no se ha usado en computación de alto rendimiento debido a que el rendimiento aún no ha madurado hasta el nivel de eficiencia de un CPU físico. (Chen, Shou, Hu, y Guo, 2012), en su investigación propone un algoritmo de mapeo de redes virtuales que puede mejorar la robustez de la red. El algoritmo incluye el mapeo de nodos y el mapeo de enlace. Mencionan que la robustez de la red se mejora en la etapa de mapeo de enlace. Como conclusión manifiestan que en el algoritmo de mapeo de nodos, se redefine los recursos de los nodos, como una combinación de los recursos de sí mismo y de los recursos de enlace. Esta nueva métrica refleja la condición de la red cerca del nodo, evitando de este modo el cuello de botella en la red. En el algoritmo de asignación de enlace, introducen el concepto de la criticidad de la red y la

sensibilidad intermedia del enlace, y el uso de este último como el costo de un enlace cuando se aplica el algoritmo k-shortest-path en el algoritmo de mapeo de enlace. Finalmente indican que los resultados de la simulación que han realizado demuestran que el algoritmo propuesto puede mejorar la robustez de la red.

2.2 Marco Conceptual

2.2.1 Virtualización de red

La virtualización es una técnica que permite ocultar, a través de la encapsulación, las características físicas de los recursos informáticos cuando otros sistemas, aplicaciones o usuarios finales interactúan con estos recursos. IBM fue quien empezó a implementar la virtualización en la década de los sesenta, como una manera lógica de particionar ordenadores mainframe en máquinas virtuales independientes. Estas particiones permitían a estas computadoras, realizar varias tareas al mismo tiempo. Dado que en aquella época los mainframes eran recursos caros, se diseñaron para ser particionados para aprovechar de esta manera, al máximo la inversión (García, 2009).

El objetivo de la virtualización es tener uno a varios sistemas operativos sobre uno ya existente, permaneciendo este sin verse afectado y pudiendo arrancarlos de manera independiente a diferencia de la instalación en el mismo equipo gracias a una capa de software llamada Virtual Machine Monitor o VMM que crea una capa de abstracción entre el equipo físico o host y el software del sistema operativo de la máquina virtual o guest (Belén, Guerrero, Abad, & Astudillo, 2011).

Para ello existen varias técnicas, que van desde la virtualización del hardware (CPU, memoria, almacenamiento, dispositivos y red) hasta la separación de los procesos utilizando un único núcleo de sistema operativo. A través de la virtualización podemos, por ejemplo, utilizar servidores virtuales para cada servicio que deseemos implementar

en nuestra red, independientemente del hardware disponible. Estos servidores pueden ser administrados por personas distintas. Más adelante, al aumentar las necesidades, pueden añadirse nuevos servidores físicos y migrar los servidores virtuales existentes entre las máquinas físicas con un coste mínimo (Martínez, 2008).

Las plataformas de virtualización son una gran alternativa para la implementación de escenarios de experimentación de redes IP. Estas plataformas facilitan el dimensionado tal como si fuera en un entorno real, con lo cual se puede reducir el riesgo de falla, el costo de inversión así como el costo de la experimentación (Fuertes y de Vergara, 2008).

Por otro lado, la mayoría de las plataformas de virtualización tienen un enfoque centralizado (es decir, mono-host). Esto significa que el despliegue y gestión de cada entorno virtual de red tienen que hacerse en un equipo físico, que resulta en una baja escalabilidad que impide la creación de grandes y complejos entornos virtuales de red (Meneses, Fuertes, Guerra, de Vergara, & Aules. 2011).

2.2.2 Métricas de desempeño de una red IP

En la gestión de grandes sistemas, las pruebas de desempeño son importantes para predecir qué hardware actualizar y cuándo hacerlo. También nos proporciona datos con los que justificar estas actualizaciones.

El grupo de trabajo IPPM ha desarrollado un conjunto de medidas estándar que pueden ser aplicados a la calidad, desempeño y fiabilidad de los datos entregados en los servicios de Internet. Estas métricas están diseñadas de manera que puedan ser realizadas por los operadores de red o los usuarios finales. Es importante que los indicadores representan un juicio de valor (es decir, define "bueno" y "malo"), sino más bien proporcionar cuantitativamente medidas de desempeño.

El grupo de trabajo IPPM ha producido documentos que definen métricas específicas y procedimientos para medir con precisión y documentar estas métricas. Esta es la lista actual de los indicadores fundamentales y el conjunto existente de derivado métricas.

- Métrica para medir la conectividad: La conectividad es el material básico del que está hecha la Internet. Por lo tanto, la métrica para determinar si dos pares de hosts (direcciones IP) pueden alcanzarse entre sí, deben formar la base de un conjunto de medición.
- Retardo en un solo sentido: Este memo define una métrica de retardo unidireccional de paquetes a través de rutas de Internet. En el Internet el camino que se toma para el transporte de datos desde el origen hacia el destino es diferente que el camino que se toma del destino hacia el origen, por lo tanto medir el retardo de cada camino independientemente nos indicara el desempeño de la red en las dos trayectorias.
- Pérdida de paquetes en un solo sentido: Este memo define una métrica para la pérdida de paquetes de un solo sentido a través de Internet. La pérdida excesiva de paquetes, hace que sea difícil mantener ciertas aplicaciones en tiempo real. Cuanto mayor sea el valor de la pérdida de paquetes más difícil es para los protocolos de la capa de transporte mantener un ancho de banda óptimo.
- Retardo de ida y vuelta: Este memo define una métrica de retardo de ida y vuelta de paquetes a través de Rutas de Internet. Valores de esta métrica por encima del mínimo proporcionar un indicador de la congestión presente en el camino.
- Pérdida de paquetes de ida y vuelta: Muchas aplicaciones de usuario y los protocolos de transporte que los hacen posible, requiere una comunicación bidireccional, por ejemplo, el TCP SYN->, <-SYN-ACK, ACK->. Así, la medición de la pérdida de paquetes de ida y vuelta en el Internet proporcionara una base para medir el desempeño más fácilmente.

- Entre otras.

2.2.3 Métricas de desempeño de un sistema computacional

El desempeño de un computador depende de muchos factores, pero en los que vamos a enfocarnos debido a nuestra investigación son los siguientes:

- Procesador: dispositivo que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora. Sus características principales son:
  · Frecuencia de reloj (en MHz).
  · Número de núcleos.
  · Capacidad de la memoria cache (en MB).
  · Número de niveles de memoria cache.
- Disco duro interno: es un dispositivo no volátil, que conserva la información aun con la pérdida de energía, que emplea un sistema de grabación magnética digital. Sus características principales son:
  · Capacidad (en MB).
  · Velocidad rotacional (en rpm).
  · Tiempo medio de acceso (en ms).
  · Tiempo medio de posicionamiento (en ms).
  · Latencia media (en ms).
- Memoria RAM: es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados. Es el área de trabajo para la mayor parte del software de un computador (Álvarez, Gimeno & Padellano, 2010). Sus características principales son:
  · Capacidad (en MB).
  · Tiempo de medio de acceso (en ms).

2.2.4 Algoritmos de planificación de la CPU y administración de memoria

A. Algoritmos de planificación del CPU

El sistema operativo decide qué proceso entra en la CPU cuando ésta queda libre; o en qué momento el proceso que está en ejecución debe abandonar la CPU. En otras palabras el S.O. debe aplicar una política de planificación de procesos.

Se usan varias magnitudes para medir el rendimiento de los algoritmos de planificación:

- Utilización de la CPU
- Tiempo de retorno
- Tiempo de espera
- Tiempo de respuesta (Santos, Quesada y Santana).

Un sistema de tiempo real se caracteriza por la interacción que mantiene con el entorno que le rodea, debiendo responder a los diferentes eventos generados por éste en unos plazos de tiempo preestablecidos. Puesto que el número máximo de tareas que es posible ejecutar simultáneamente en un computador es limitado (como máximo igual al número de procesadores), es necesario definir un conjunto de reglas que permitan determinar qué tarea o tareas deben ser ejecutadas en cada momento. Estas reglas constituyen los denominados algoritmos o políticas de planificación, y de su elección depende en gran medida el que se satisfagan o no las restricciones temporales impuestas al sistema.

Dependiendo de si la prioridad de las tareas es constante o cambia en función del estado del sistema los algoritmos de planificación en tiempo de ejecución se dice que están basados en prioridades estáticas (FIFO, ROUND-ROBIN, SERVIDOR ESPORÁDICO, INTERCAMBIO DE PRIORIDADES, SERVIDOR DIFERIDO, SLACK STEALING) o dinámicas (EDF, LLF, BEST – EFFORT, SERVIDOR DE HOLGURA, CBS) (Rivas, 2002).

i. Planificación Primero en Entrar-Primero en Salir (FIFO, *First In First Out*)

Cuando se tiene que elegir a qué proceso asignar la CPU se escoge al que llevará más tiempo listo. El proceso se mantiene en la CPU hasta que se bloquea voluntariamente. La ventaja de este algoritmo es su fácil implementación, sin embargo, no es válido para entornos interactivos ya que un proceso de mucho cálculo de CPU hace aumentar el tiempo de espera de los demás procesos. Para implementar el algoritmo sólo se necesita mantener una cola con los procesos listos ordenada por tiempo de llegada. Cuando un proceso pasa de bloqueado a listo se sitúa el último de la cola.

ii. Planificación por Turno Rotatorio (Round Robin)

Este es uno de los algoritmos más antiguos, sencillos y equitativos en el reparto de la CPU entre los procesos, muy válido para entornos de tiempo compartido. Cada proceso tiene asignado un intervalo de tiempo de ejecución, llamado **cuantum o cuanto**. Si el proceso agota su *cuantum* de tiempo, se elige a otro proceso para ocupar la CPU. Si el proceso se bloquea o termina antes de agotar su *cuantum* también se alterna el uso de la CPU. El *round robin* es muy fácil de implementar. Todo lo que necesita el planificador es mantener una **lista** de los procesos listos.

iii. Algoritmo de planificación SJF (*shortest job frist*)

El algoritmo de primero el trabajo más corto (SJF, shortest job frist), que asocia a cada proceso la longitud de la siguiente ráfaga de CPU de ese proceso. Cuando la CPU queda disponible, asigna al proceso cuya siguiente ráfaga de CPU sea más corta. Si hay dos procesos cuyas siguientes ráfagas de CPU tienen la misma duración, se emplea lanificación FCFS (first come, first served o FIFO) para romper el empate.

iv. Algoritmo de planificación SRT (Es un SJF apropiativo)

Este algoritmo siempre ejecuta primero aquellos procesos a los que les queda menos tiempo para terminar. Este algoritmo también es conocido como 'optimo', pues con él se obtienen los mejores resultados.

v. Algoritmo de Prioridades

En este algoritmo a cada proceso se le asocia un número entero de prioridad. Mientras menor sea este entero pues mayor prioridad tiene el proceso, por lo que la esencia del algoritmo es planificar la entrada de procesos a la CPU de acuerdo a la prioridad asociada de cada uno de ellos.

Un caso particular del algoritmo por prioridad es el SJF, donde el valor del próximo ciclo de CPU representa la prioridad. El algoritmo por prioridad corrige algunas deficiencias del SJF, particularmente el retraso excesivo de procesos largos y el favoritismo por procesos cortos.

Para comprender su funcionamiento podemos indicar que, un algoritmo consiste en establecer que la prioridad sea "1 / f", donde "f" es la fracción del último cuanto utilizado por el proceso. Un proceso que utilice 2 ms (dos milisegundos) de su cuanto de

100 ms (cien milisegundos) tendrá prioridad 50 (cincuenta). Un proceso que se ejecutó 50 ms antes del bloqueo tendrá prioridad 2. Un proceso que utilizó todo el cuanto tendrá prioridad 1.

Frecuentemente los procesos se agrupan en "Clases de Prioridad", en cuyo caso se utiliza la Planificación con Prioridades entre las clases y con Round Robin (RR) dentro de cada clase. Si las prioridades no se reajustan en algún momento, los procesos de las clases de prioridad mínima podrían demorarse indefinidamente.

El inconveniente, es que puede producir 'inanición', es decir si tenemos un proceso de prioridad baja, y muchos de alta, puede ocurrir que el primero no se ejecute nunca. Se puede llevar a cabo un proceso de envejecimiento, el cual hace ganar prioridad al primer proceso, permitiendo que se ejecute.

B. Administración de memoria

La memoria es uno de los principales recursos de la computadora, la cual debe de administrarse con mucho cuidado. Aunque actualmente la mayoría de los sistemas de cómputo cuentan con una alta capacidad de memoria, de igual manera las aplicaciones actuales tienen también altos requerimientos de memoria, lo que sigue generando escasez de memoria en los sistemas multitarea y/o multiusuario.

Se denomina gestión de memoria al acto de gestionar la memoria de un dispositivo informático. De forma simplificada se trata de proveer mecanismos para asignar secciones de memoria a los programas que las solicitan, y a la vez, liberar las secciones de memoria que ya no se utilizan para que estén disponibles para otros programas.

La necesidad de repartir memoria entre varios usuarios redujo la cantidad de memoria para cada uno, e hizo necesaria la introducción de algún mecanismo de protección para

aislar entre si las actividades de los programas. Así tenemos que en la gestión de memoria se debe perseguir los siguientes objetivos:

- Protección: Si varios procesos comparten la memoria principal, se debe asegurar que ninguno de ellos pueda modificar posiciones de memoria de otro proceso. Debe disponerse de un sistema de permisos de acceso que especifique los derechos que tiene cada proceso en el acceso a zonas de memoria de otros procesos.
- Comportamiento: El comportamiento de la memoria parece estar en contradicción con la protección, pero es que a menudo también es necesario que vario procesos puedan compartir y actualizar estructura de datos comunes, por ejemplo en un sistema de base de datos. En otras ocasiones, lo que se requiere es compartir zonas de código, por ejemplo, en rutinas de biblioteca, para no tener en memorias distintas copias de la misma rutina. En este caso, se hace necesaria alguna protección para que un proceso no modifique inadvertidamente el código de las rutinas.
- Reubicación: La multiprogramación requiere que varios procesos residan simultáneamente en memoria. Lo que no se puede saber antes de llevarlo memoria es la dirección absoluta en la que se va a cargar el proceso, por lo que no es práctico utilizar direcciones absolutas en el programa. En su lugar es preferible realizar direccionamientos relativos para permitir que un programa pueda ser cargado y ejecutado en cualquier parte de la memoria.
- Organización de la memoria: La memoria se debe organizar tanto física como lógicamente.

A la memoria RAM normalmente se necesita ampliarla con memoria secundarias más barata y más lenta, utilizando para esto discos; o por el contrario se añade memoras de acceso más rápido que la RAM principal, como es el caso de la memoria cache. Esta

jerarquía física de memorias hace necesario un sistema que controle el flujo de información entre los distintos dispositivos de almacenamiento, por esto es conveniente que sea el gestor de memoria el que se ocupe de esta labor.

Aunque la mayoría de la memorias están organizadas como un único espacio lineal de direcciones secuenciales, que van desde 0 hasta un máximo, esto no refleja la estructura lógica de los programas, que utilizan estructuras lógicas de instrucciones y datos, tales como módulos, rutinas o procedimientos, matrices, registros, etc. Si una gestión de memoria pudiera proporcionar varios espacios de direcciones, cada estructura lógica podría ser una entidad independiente: un segmento. Esto sería ventajoso pues los segmentos pueden cargarse y compilarse de forma independiente, teniendo cada uno de ellos sus propios derechos de acceso.

Los sistemas de gestión de memoria pueden dividirse en dos clases: los que mueven los procesos entre la memoria principal y secundaria (intercambio y paginación), y los que no lo hacen.

C. Técnicas de gestión de memoria

i. Intercambio de memoria

El objetivo del intercambio es dar cabida a la ejecución de más aplicaciones de las que pueden residir simultáneamente en la memoria del sistema.

Consiste en trasladar el código y los datos de un proceso completo de memoria al sistema de almacenamiento secundario, para cargar otro previamente almacenado, no permiten a un proceso utilizar más memoria RAM de la que realmente existe en el sistema. Debe quedar claro que para que pase a ejecución algún proceso que se han llevado a disco, antes hay que traerlo a memoria. Cuando al proceso en ejecución se le acabe su porción de tiempo, es posible que se le vuelva a llevar a memoria secundaria

para dejar espacio al proceso que haya seleccionado el planificador. Al trasiego de procesos que se lleva de memoria a disco y de disco a memoria se le denomina intercambio (swapping).

En principio, un sistema de intercambio podría estar basado en particiones de tamaño fijo, de tal forma que cuando un proceso queda bloqueado en espera, se le puede mover al disco y traer otro a la partición que queda libre. Pero este sistema no es aconsejable cuando se dispone de poca memoria principal, pues los programas pequeños desperdician mucha memoria cuando ocupan particiones grandes (fragmentación interna). Otro enfoque mejor es el basado en particiones de tamaño variable.

La diferencia entre las particiones fijas y las de tamaño variable es que en estas últimas el número, la dirección y el tamaño de las particiones varían constantemente a medida que los procesos van y vienen: mientras que en las de tamaño fijo no varían. La flexibilidad de no estar sujeto a un número fijo de particiones que resulten demasiado grandes o demasiado pequeñas mejora la utilización de la memoria, pero también complica los algoritmos de asignación, liberación y contabilidad de memoria disponible.

Existen tres métodos utilizados por los sistemas operativos para llevar la cuenta de la memoria utilizada y de los huecos libres: mapa de bits, listas de bloques y el sistema buddy.

ii. Paginación

En sistemas operativos de computadoras, los sistemas de paginación de memoria dividen los programas en pequeñas partes o páginas. Del mismo modo, la memoria es dividida en trozos del mismo tamaño que las páginas llamados marcos de página. De esta forma, la cantidad de memoria desperdiciada por un proceso es el final de su última página (Agramon, 2011).

Los programas de los usuarios se dividen en zonas consecutivas (páginas). Cada página es de tamaño fijo (de 512 Bytes a 64 KB). Cada página, dentro del programa, se identifica con un número correlativo. Si, por ejemplo, la página es de 4KB, y el programa de 64KB, éste estaría formado por 16 páginas (0 a F).

La memoria principal se estructura en marcos de página de igual longitud que las páginas del programa. Cada marco se identifica con un número correlativo. Si el tamaño de la memoria principal fuese de 1MB, y el tamaño del marco 4KB, habría 256 marcos de página (del 00 al FF).

El fundamento de la paginación está en que no es necesario que un programa se almacene en posiciones consecutivas de memoria. Las páginas se almacenan en marcos de página libres independientemente de que estén o no contiguos (Prieto, Lloris y Torres, 1995).

Las páginas sirven como unidad de almacenamiento de información y de transferencia entre memoria principal y memoria auxiliar o secundaria. Cada marco se identifica por la dirección de marco, que está en la posición física de la primera palabra en el marco de página.

La paginación evita el considerable problema de ajustar los pedazos de memoria de tamaños variables que han sufrido los esquemas de manejo de memoria anteriores. Dado a sus ventajas sobre los métodos previos, la paginación, en sus diversas formas, es usada en muchos sistemas operativos.

Características de la paginación:

- El espacio de direcciones lógico de un proceso puede ser no contiguo.
- Se divide la memoria física en bloques de tamaño fijo llamados marcos (frames).
- Se divide la memoria en bloques de tamaño llamados páginas.

- Se mantiene información en los marcos libres.
- Para correr un programa de n páginas de tamaño, se necesitan encontrara n marcos y cargar el programa.
- Se establece una tabla de páginas para trasladar las direcciones lógicas a físicas.
- Se produce fragmentación interna.

Ventajas de la paginación:

- Es posible comenzar a ejecutar un programa, cargando solo una parte del mismo en memoria, y el resto se cargara bajo la solicitud.
- No es necesario que las paginas estén contiguas en memoria, por lo que no se necesitan procesos de compactación cuando existen marcos de páginas libres dispersos en la memoria.
- Es fácil controlar todas las páginas, ya que tienen el mismo tamaño.
- El mecanismo de traducción de direcciones (DAT) permite separar los conceptos de espacio de direcciones y espacios de memoria. Todo el mecanismo es transparente al usuario.
- Se libera al programador de la restricción de programar para un tamaño físico de memoria, con lo que s e aumenta su productividad. Se puede programar en función de una memoria mucho más grande a la existente.
- Al no necesitarse cargar un programa completo en memoria para su ejecución, se puede aumentar el número de programas multiprogramándose.
- Se elimina el problema de fragmentación externa.

Desventajas de la paginación:

- El costo de hardware y software se incrementa, por la nueva información que debe manejarse y el mecanismo de traducción de direcciones necesario. Se consume mucho más recursos de memoria, tiempo en la CPU para su implantación.
- Se deben reservar áreas de memoria para las PMT de los procesos. Al no ser fija el tamaño de estas, se crea un problema semejante al de los programas (como asignar un tamaño óptimo sin desperdicio de memoria, u "ovearhead" del procesador).
- Aparece el problema de fragmentación interna.

iii. Segmentación

La segmentación es una técnica de gestión de memoria que pretende acercarse más al punto de vista del usuario. Los programas se desarrollan, generalmente, en torno a un núcleo central (principal) desde el que se bifurca a otras partes (rutinas) o se accede a zonas de datos (tablas, pilas, etc).

Desde este punto de vista, un programa es un conjunto de componentes lógicos de tamaño variable o un conjunto de segmentos, es decir, el espacio lógico de direcciones se considera como un conjunto de segmentos, cada uno definido por un identificador, y consistente de un punto de inicio y el tamaño asignado.

La segmentación de un programa la realiza el compilador y en ella cada dirección lógica se expresará mediante dos valores: Número de segmento (s) y desplazamiento dentro del segmento (d).

Una de las implementaciones más obvias y directas de un espacio de memoria segmentado es asignar un segmento distinto a cada una de las secciones del espacio en memoria de un proceso.

La segmentación también ayuda a incrementar la modularidad de un programa: Es muy común que las bibliotecas enlazadas dinámicamente estén representadas en segmentos independientes.

Esta técnica permite reducir la fragmentación interna de la memoria provocada por la paginación, ya que asigna a cada programa la cantidad de memoria que requiere.

La carga de un programa en memoria exige la búsqueda de los huecos adecuados a sus segmentos, y puesto que éstos son de tamaño variable, se ajustarán lo más posible a las necesidades, produciéndose huecos pequeños. En este caso se produce fragmentación externa. La eficiencia de la segmentación requiere, de igual forma que la paginación, el uso de memorias caché para lograr unos tiempos de acceso adecuados. De igual forma que en la paginación, se pueden compartir segmentos entre varios procesos.

Una de las principales ventajas del uso de segmentación es que nos permite pedir a la unidad de gestión de memoria que cada uno de los segmentos tenga un distinto juego de permisos para el proceso en cuestión: El sistema operativo puede indicar, por ejemplo, que el segmento de texto (el código del programa) sea de lectura y ejecución, mientras que la sección de datos es de lectura y escritura. De este modo podemos evitar que un error en la programación resulte en que datos proporcionados por el usuario o por el entorno modifiquen el código que está siendo ejecutado.

iv. Híbrido

La paginación y la segmentación se pueden combinar para potenciar las ventajas de cada técnica. Ejemplo de una arquitectura de este tipo es Intel, la memoria es segmentada, y los segmentos se conforman de páginas.

En lugar de tratar un segmento como una unidad contigua, este puede dividirse en páginas. Cada segmento puede ser descrito por su propia tabla de páginas.

El esquema de segmentación paginada tiene todas las ventajas de la segmentación y la paginación:

- Debido a que los espacios de memorias son segmentados, se garantiza la facilidad de implantar la compartición y enlace.
- Como los espacios de memoria son paginados, se simplifican las estrategias de almacenamiento.
- Se elimina el problema de la fragmentación externa y la necesidad de compactación.

Desventajas de la segmentación paginada:

- Las tres componentes de la dirección y el proceso de formación de direcciones hace que se incremente el costo de su implantación. El costo es mayor que en el caso de segmentación pura o paginación pura.
- Se hace necesario mantener un número mayor de tablas en memoria, lo que implica un mayor costo de almacenamiento.

Sigue existiendo el problema de fragmentación interna de todas o casi todas las páginas finales de cada uno de los segmentos. Bajo paginación pura se desperdician solo la última página asignada, mientras que bajo segmentación – paginada el desperdicio puede ocurrir en todos los segmentos asignados.

v. Memoria Virtual

La memoria virtual es una técnica para proporcionar la simulación de un espacio de memoria mucho mayor que la memoria física de una máquina. Esta "ilusión" permite que los programas se hagan sin tener en cuenta el tamaño exacto de la memoria física.

La ilusión de la memoria virtual está soportada por el mecanismo de traducción de memoria, junto con una gran cantidad de almacenamiento rápido en disco duro. Así en cualquier momento el espacio de direcciones virtual hace un seguimiento de tal forma que una pequeña parte de él, está en memoria real y el resto almacenado en el disco (swapping), y puede ser referenciado fácilmente.

El sistema operativo gestiona niveles de memoria principal y memoria secundaria: transferencia de bloques entre ambos niveles (normalmente basada en paginación), de memoria secundaria a principal por demanda y de memoria principal a secundaria por expulsión.

2.2.5 Herramientas de medición de desempeño de la red

Existen varias herramientas para la medición del rendimiento de la red, a continuación mencionaremos algunas de ellas.

- NetEm: es una herramienta de Linux que realiza el control de tráfico, además permite añadir retardo y pérdida de paquetes. NetEm consta de dos partes, un pequeño módulo de kernel para una disciplina de cola y una línea de comando para configurarlo (Hemminger, 2005).

  Netem es un conjunto de herramientas que permiten emular las condiciones propias de una gran variedad de redes mediante técnicas que reproducen retardo variable, pérdida, duplicación y desorden de los paquetes de datos.

  En el artículo técnico EMULACIÓN DE ESCENARIOS DE RED MEDIANTE UN TESTBED utilizan NetEm para emular el comportamiento a nivel de red, ya que permite introducir retardos controlados en los paquetes con distintas distribuciones estadísticas (Saldaña et al.).

- SAR: El monitor SAR (System Activity Reporter) es una de las herramientas software más potentes disponibles actualmente para monitorizar sistemas informáticos. Es un monitor muy utilizado por los administradores de sistemas Unix para la detección de cuellos de botella. Al ser un monitor de tipo software el sistema tendrá una pequeña sobrecarga debido a su ejecución (Álvarez et al., 2010).
- HTTPERF: es una herramienta para medir el desempeño de los servidores Web. Se proporciona una instalación flexible para generar varias cargas de trabajo HTTP y para medir el desempeño del servidor. El enfoque de httperf no está en implementar un punto de referencia en particular, sino en proporcionar una solución robusta, de alto desempeño. Las tres características distintivas de httperf son su robustez, que incluye la capacidad de generar y mantener la sobrecarga del servidor, soporte para el HTTP/1.1 y SSL y su extensibilidad a los generadores de carga de trabajo y nuevos mediciones de desempeño.
- Tcpdump: es una herramienta en línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red. Permite al usuario capturar y mostrar a tiempo real los paquetes transmitidos y recibidos en la red a la cual el ordenador está conectado.
- Wireshark: antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones. La funcionalidad que provee es similar a la de tcpdump, pero añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite ver todo el tráfico que pasa a través de una red (usualmente una red Ethernet, aunque es compatible con algunas otras) estableciendo la configuración en modo promiscuo.

- IPERF: Es una herramienta que se utiliza para hacer pruebas en redes informáticas. El funcionamiento habitual es crear flujos de datos TCP y UDP y medir el rendimiento de la red. Iperf permite al usuario ajustar varios parámetros que pueden ser usados para hacer pruebas en una red, o para optimizar y ajustar la red. Iperf puede funcionar como cliente o como servidor y puede medir el rendimiento entre los dos extremos de la comunicación, unidireccional o bidireccionalmente.

Varias compañías ofrecen analizadores de red como el Network General. Hewlett-Packard, Spider System y Novell. Watchdog. El Network General, es un ejemplo del software (y hardware) que puede emplear un administrador de red para garantizar que esta funcione con eficiencia.

El programa Watchdog ofrece información estadística detallada sobre los patrones de tráfico de la red, que puede desplegarse gráficamente de manera que el administrador pueda ver qué estaciones de trabajo generan más tráfico. Watchdog puede reportar las estaciones de trabajo que están generando más paquetes con errores (un indicador de que la red puede tener una tarjeta de interfaz de red defectuosa).

Hoy en día existen muchos artículos relacionados al estudio del rendimiento de un entorno virtualizado de red con diferentes aplicaciones o servicios. (Quintero y Silva, 2013) implementan un ambiente virtualizado para un servidor web y realiza un monitoreo de los indicadores de rendimiento para evaluar una comparación entre el ambiente real y virtual.(Álvarez y Mise, 2013) en este proyecto los autores nombran diferentes herramientas para monitorear y evaluar los indicadores de rendimiento al virtualizar servidores de correo, las herramientas que describen son: free, top, vmstat, Sysstat y SAR. (Peñafiel y Mendoza, 2012) realizan la virtualización de dos servidores, dirigido especialmente al monitoreo del tráfico de red y del rendimiento. Para esta

implementación usan “VMWARE ESX”, ya que les permite realizar un monitoreo más potente y completo. (Mendoza y Cordovilla, 2012) analizan una solución de almacenamiento de datos de servidores que trabajan en un ambiente virtualizado y usan herramientas de monitoreo como el “Performance Monitor” de Windows server 2008, para analizar los indicadores de rendimiento enfocados al hardware como son el disco físico, memoria y red. (Fuertes y de Vergara, 2008) proponen evaluar e identificar cuál de las plataformas de virtualización consume menos CPU y memoria y por tanto es la más adecuada para realizar experimentos de medida de prestaciones y servicios de redes.

CAPITULO III METODOLOGÍA DE LA INVESTIGACIÓN

3.1 Ubicación geográfica del proyecto de investigación

El proyecto de investigación se desarrolló en el edificio de Posgrados en el Campus de la Universidad de las Fuerzas Armadas del Ecuador, ubicado en la Av. Gral. Rumiñahui s/n en el cantón Sangolquí. Su área de influencia estuvo enmarcada en las Redes de Información y sistemas distribuidos.

3.2 Método de investigación

Este proyecto de tesis inició con una investigación bibliográfica. Luego se enfocó a una metodología de investigación científica empírica, analítica y cuantitativa; debido a que es un sistema auto correctivo y progresivo, que utiliza la observación y la experimentación. Se aplicó también la metodología de investigación descriptiva ya que se llegó a conocer el comportamiento del experimento mediante la recolección de datos y el análisis estadístico. De la misma manera esta investigación se encontró dentro de la metodología exploratoria con el propósito de destacar los aspectos fundamentales de la problemática planteada. Esto favoreció a que se pueda encontrar los procedimientos adecuados para llegar a la solución.

3.3 Diseño de la Topología experimental de red

Para evaluar el rendimiento de una infraestructura virtualizada, se creó un entorno de red híbrido (real+virtualizado), utilizando la plataforma de virtualización XEN.

Se instaló XEN 4.1-i386 sobre DEBIAN 3.2.0.4-686-pae, ya que como se describió anteriormente XEN permite alcanzar la virtualización con menor pérdida de rendimiento, sin un soporte especial de hardware, es decir, se reduce el overhead aunque podría disminuir su funcionalidad (tecnología paravirtualizada vs completa).

La Fig.1 muestra el host anfitrión virtualizado en un entorno de red, el mismo que tiene los siguientes elementos: el host anfitrión, un router de borde, un firewall, dos servidores (Web y FTP), y dos PC's.

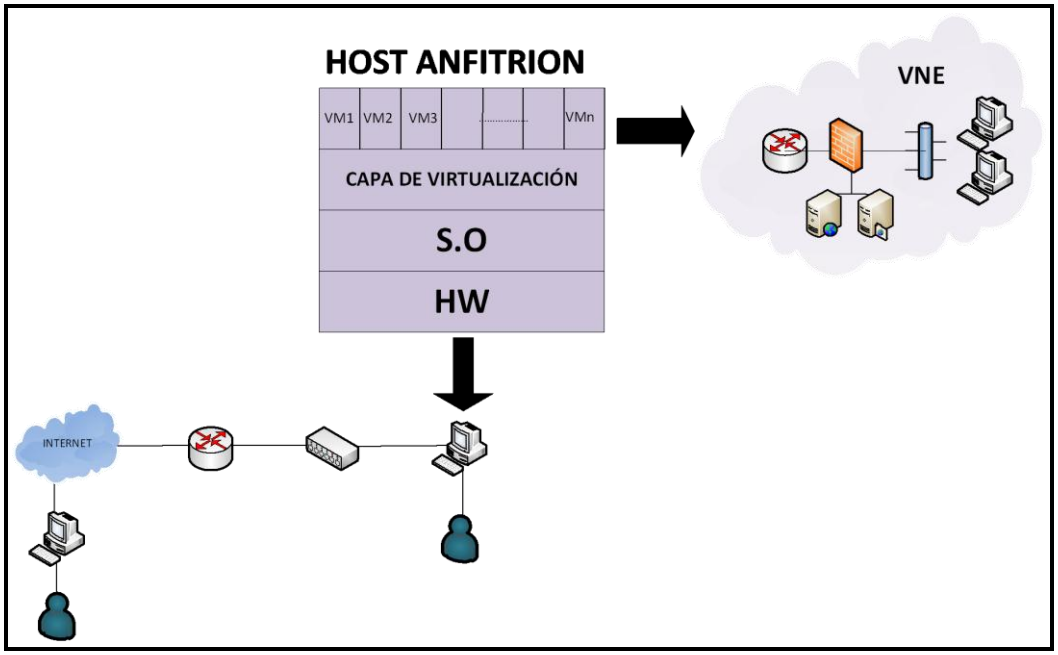


Figura 1: Host anfitrión virtualizado en un entorno de red

Para lograr conectividad, se instaló y configuró un Servidor Web, un servidor de Transferencia de Archivos (FTP) y se asignó a cada una de las máquinas virtuales (MV's) el siguiente esquema de direccionamiento.

Tabla 1

Direccionamiento IP

MAQUINA VIRTUAL	DIRECCIÓN IP	MÁSCARA DE SUBRED	GATEWAY	SERVIDOR DE DOMINIO (DNS)
Router	eth0 192.168.1.11 eth0:1 172.16.30.17	eth0 255.255.255.0 eth0:1 255.255.255.252	192.168.1.13	192.168.1.1
Firewall	eth0 192.168.30.1 eth0:1 172.16.30.1 eth0:2 172.16.30.18	eth0 255.255.255.224 eth0:1 255.255.255.240 eth0:2 255.255.255.252	172.16.30.17	192.168.1.1
Servidor WEB	eth0 172.16.30.2	255.255.255.240	172.16.30.1	192.168.1.1
Servidor FTP	eth0 172.16.30.3	255.255.255.240	172.16.30.1	192.168.1.1
Pc 1	eth0 192.168.30.2	255.255.255.224	192.168.30.1	192.168.1.1
Pc 2	eth0 192.168.30.3	255.255.255.224	192.168.30.1	192.168.1.1

3.4 Implementación de la topología experimental

Para la implementación de la topología experimental se utilizó un CPU con las siguientes características principales: Core i5, 500 GB de HD, 8 GB de RAM.

Como se mencionó anteriormente para la configuración del entorno de red virtual, se utilizó XEN empleando la técnica de paravirtualización sobre DEBIAN como sistema operativo anfitrión.

Se consideró que cada una de las máquinas virtuales sean configuradas como volúmenes lógicos, es por eso, que se realizó la partición del disco duro de la siguiente forma:

- /boot : sda1
- / : sda5
- swap : sda6
- LVM : sda7

Para la instalación de XEN se utilizó el siguiente comando:

```
# apt-get -p install xen-linux-system-686-pae
```

En primero lugar se creó un volumen físico y un grupo de volumen llamado vg0 con los siguientes comandos:

- `# pvcreate /dev/sda7`
- `# vgcreate vg0 /dev/sda7`

Posteriormente para la creación de cada máquina virtual se utilizó el siguiente comando:

```
# xen-create-image   --hostname=router   --memory= 512mb    --vcpus=2 \
   --lvm=vg0   --dhcp   --pygrub  --dist=wheezy
```

Donde:

- xen-create-image: comando de instrucción para crear el dominio invitado.
- --hostname=router: nombre de la máquina virtual.
- -- memory= 512mb: memoria RAM de la máquina virtual.
- --vcpus=2: número de cpu's o procesadores que se le asigna a la máquina virtual.
- --lvm=vg0: usa el almacenamiento del grupo de volumen creado vg0.
- --dhcp: método dhcp para la interfaz de red.
- --pygrub: permite que pueda arrancar domU's de Linux con un kernel dentro del domU en lugar de usar el sistema de archivos del dom0. Esto simplifica la administración, ya que cada domU gestiona su propio kernel e initrd.
- --dist=wheezy: especifica la distribución a ser instalada.

El comando anterior genera automáticamente 4GB de disco duro a la máquina virtual creada y 128 MB de memoria de intercambio (swap).

Los volúmenes lógicos creados de cada máquina virtual se ubican en la siguiente dirección o path:

```
# ls /dev/vg0
```

Las distribuciones que se instalaron en cada una de las máquinas virtuales se muestran a continuación en la Tabla 2:

Tabla 2

Distribuciones Instaladas

MÁQUINA VIRTUAL	DISTRIBUCIÓN
Router	Wheezy
Firewall	Squeezy
Servidor WEB	Wheezy
Servidor FTP	Wheezy
Pc 1	Wheezy
Pc 2	Wheezy

Los archivos de configuración de cada máquina virtual (.cfg) se ubican en la siguiente dirección o path:

```
# ls /etc/xen/
```

Se debe tomar en cuenta que el kernel predeterminado para las distribuciones de DEBIAN usa la consola hvc0.

3.5 Configuración de la topología experimental

Para la configuración de la topología experimental como se muestra en la Fig. 2 se levantó las máquinas virtuales en una terminal virtual por cada una, mediante el comando:

```
# xm  create /etc/xen/router.cfg -c
```

Así, se pudo configurar cada máquina virtual con los parámetros correspondientes tanto a nivel de red como a nivel de aplicaciones.

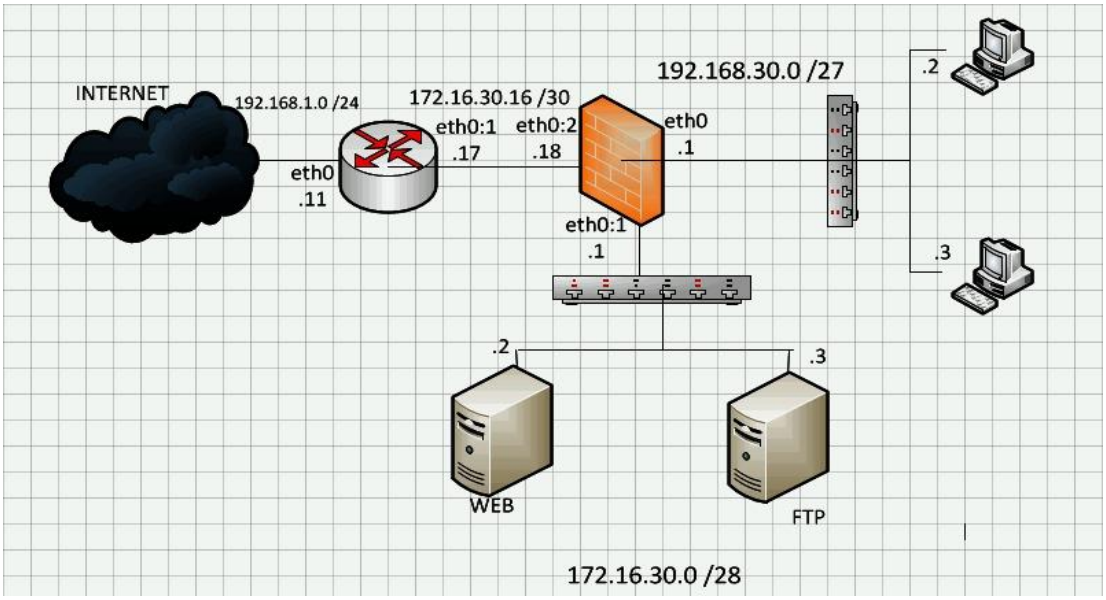


Figura 2: Topología de red experimental.

Tanto en el router de borde como en el firewall se configuró sub-interfaces de red, ya que se necesitaba más de una para tener conectividad en todo el escenario virtual.

En el router de borde se configuró un conjunto de IPtables que permitió la salida hacia el Internet de la LAN virtualizada, las cuales habilitan el DNS, el ICMP, tráfico TCP y UDP y el bit de forwarding que es imprescindible.

Del mismo modo en la máquina virtual número 2 se configuró un conjunto de reglas que le permitió actuar como un firewall, estas reglas permitieron realizar lo siguiente:

- Forward de paquetes, para que otras máquinas puedan salir a través del firewall.
- Se permitió el acceso de la DMZ a una pc en la red LAN.
- Se permitió acceder el terminal server de la DMZ desde la LAN.
- Se cerró el acceso de la DMZ a la LAN.
- Se cerró el acceso de la DMZ al firewall.
- Se cerró los accesos indeseados del exterior.
- Se cerró un puerto de gestión: webmin.

En la máquina virtual número 3 se instaló un servidor HTTP Apache2, que es un servidor Web de código abierto y es usado principalmente para enviar páginas Web estáticas y dinámicas en la World Wide Web.

En la máquina número 4 se instaló un servidor de transferencia de archivos, específicamente vsftp.

3.6 Pruebas línea base

Para medir el consumo de la CPU y la memoria se utilizó el paquete sysstat que provee varias herramientas para monitorear la performance del sistema y el uso de recursos. La herramienta que se utilizó para obtener los datos de la línea base es SAR, la cual reportó información de actividad del sistema principalmente del CPU y memoria,

que son los dos parámetros que fueron de interés medir. A continuación se detalla algunos de los comandos que se utilizó para capturar datos.

- Para capturar datos de consumo del CPU cada 1 segundo y esta información guardar en un nuevo archivo.

```
# sar -u 1-o archivo-sar
```

- Para visualizar los datos en el archivo generado.

```
# sar -f archivo-sar
```

- Para visualizar los datos del archivo-sar generado en un formato fácilmente exportable como cvs, xml, etc.

```
# sadf -d archivo-sar
```

ó

```
# sadf -x archivo-sar
```

- Finalmente la salida del comando sadf se direccionó hacia un nuevo archivo, el cual permitió ser leído y exportado fácilmente por herramientas estadísticas y gráficas como excel, gnu pspp, gnuplot, etc.

```
# sadf -d archivo-sar > nuevo-archivo
```

Las figuras realizadas desde los archivos generados, se los efectuó con el programa gnuplot, ya que permite obtener una gráfica desde cualquier tipo de datos.

Gnuplot puede producir sus resultados directamente en pantalla, así como en multitud de formatos de imagen, como PNG, EPS, SVG, JPEG, etc. Se puede usar interactivamente o en modo por lotes (*batch*), usando scripts. Este programa tiene una base de usuarios y está convenientemente mantenido por sus desarrolladores.

Los primeros resultados que se obtuvo luego de virtualizar el escenario de red propuesto, es el de consumo de CPU y memoria al momento de arrancar cada una de las máquinas virtuales (MV´s), como se observa en la Fig. 3, el porcentaje de sobrecarga de la utilización del CPU comienza a reducir hasta llegar a un 39% de consumo del CPU, si el valor del porcentaje se aproxima a cero quiere decir que el CPU está sobrecargado. Posteriormente el porcentaje de consumo se mantiene en un 50 % aproximadamente, hasta que finalmente luego de 10 segundos trascurridos el CPU queda liberado de las aplicaciones y procesos tanto a nivel de usuario como del sistema o Kernel.

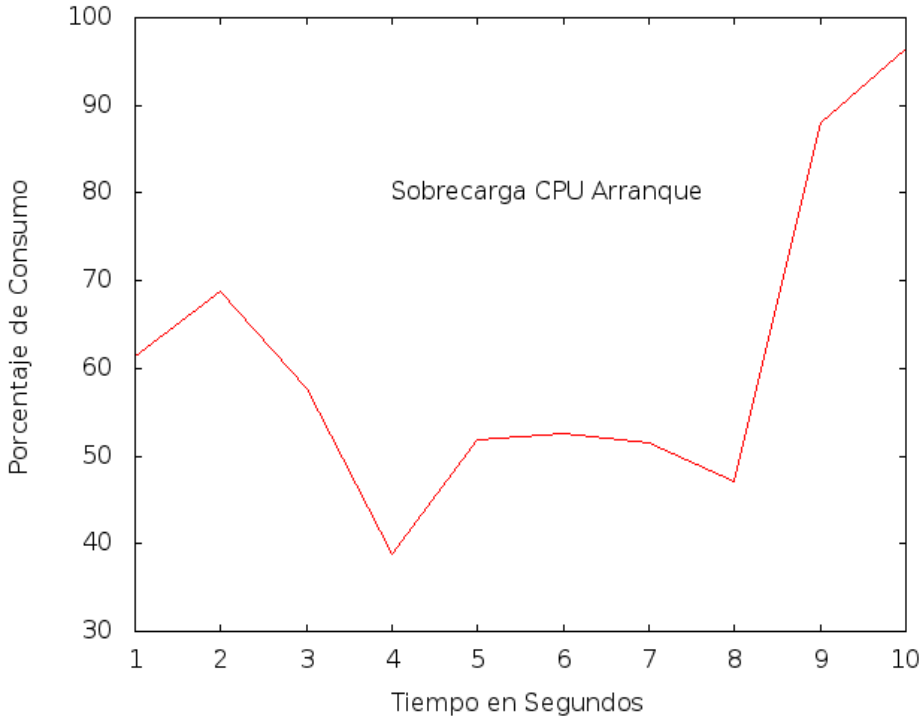


Figura 3: Porcentaje de consumo del CPU al arrancar las MV.

En la Fig.4 podemos observar la variación del consumo del CPU al arrancar las máquinas virtuales, el mismo muestra los intervalos de saturación del procesador, donde existe una pico de consumo entre el 50% y 70%, dado que a medida que el porcentaje de consumo del procesador llega a cero, quiere decir que el CPU está siendo saturado, lo que significa, que en este caso tenemos un consumo máximo del 50% y un mínimo del 30%, con un promedio de saturación del 40% del procesador.

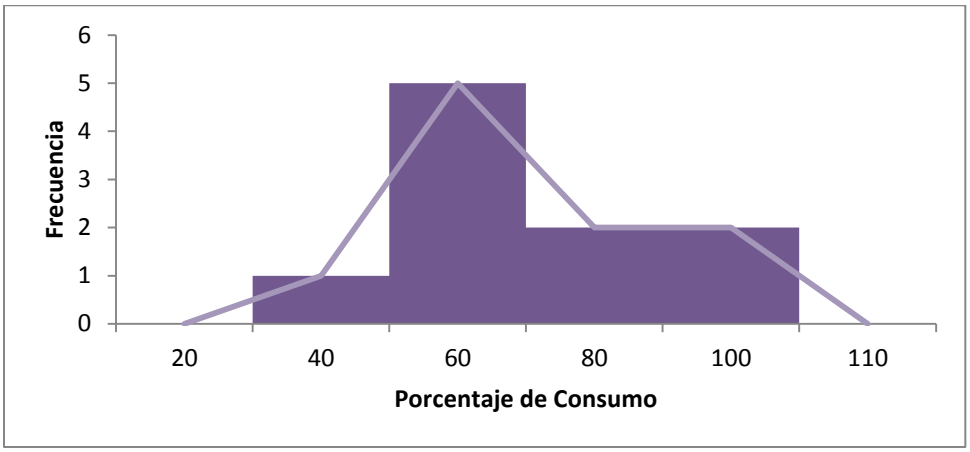


Figura 4: Variación del consumo del CPU al arrancar las MV.

Del mismo modo se observa en el Fig.5 que existe una variación en el consumo de memoria RAM en el instante que se encienden las máquinas virtuales (MV). Además se observa que se incrementa hasta un máximo del 3%. Más tarde se mantiene en un promedio de consumo de memoria del 2.7%, hasta que luego de trascurrido 10 segundos comienza a decrecer dicho consumo.

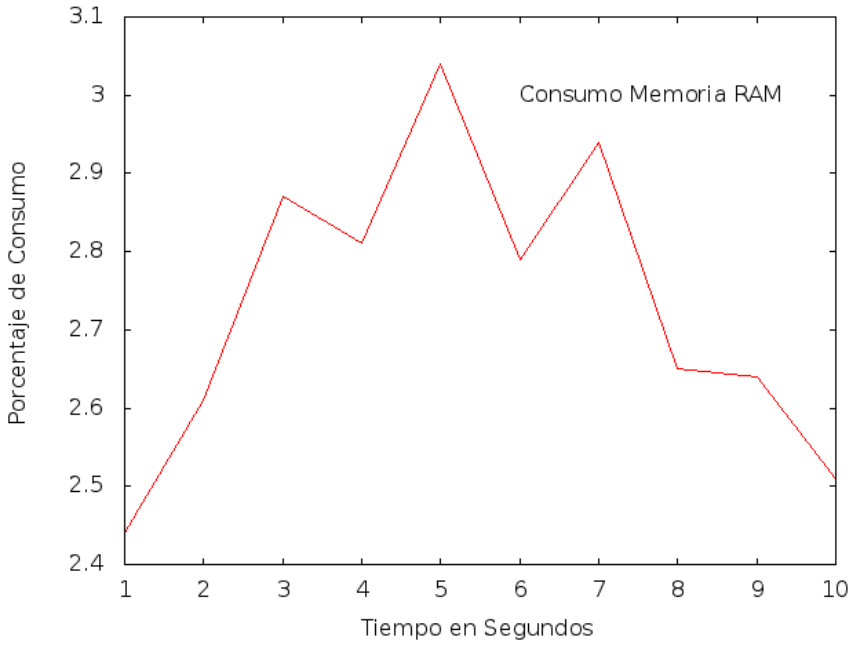


Figura 5: Porcentaje de consumo de memoria RAM al arrancar las MV.

En la Fig.6 podemos observar la variación de consumo de la memoria RAM al arrancar las máquinas virtuales, donde se muestra una diversificación de consumo de memoria entre el 2.75% y 3.25%. En este caso no existe una diferenciación de saturación de memoria importante.

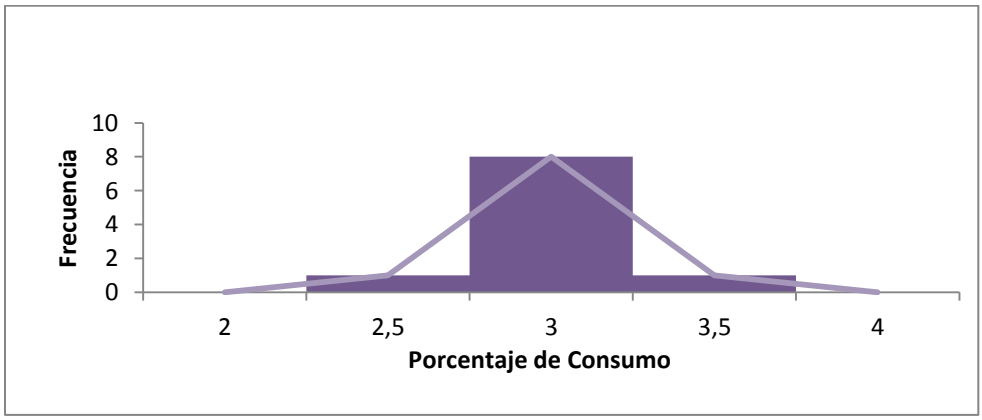


Figura 6: Variación de consumo de memoria RAM al arrancar las MV.

Para realizar la medición se desarrolló pruebas de estrés hacia el servidor web, desde una PC en la red interna como también desde una PC externa, con la ayuda de las herramientas siege y ab respectivamente.

Siege, es una herramienta desarrollada por Jeffrey Fulmer, diseñada para realizar pruebas de estrés. Es utilizada por desarrolladores web para medir el rendimiento de su código bajo coacción, mediante la creación de un número de visitas simultáneas, por un periodo de tiempo.

Esta prueba se la realizó desde el PC 1 mediante el siguiente comando:

```
# siege -c350 -t4M -d1 172.16.30.2
```

Dónde:

- -c350 : es el número de usuarios para simular las conexiones simultáneas.
- -t4M : es el tiempo de ejecución (4 minutos).
- -d1 : es el retardo (delay) en segundos que cada usuario simulado va a tener.

En este caso se creó 350 conexiones simultáneas durante 4 minutos.

Al mismo tiempo se realizó desde una laptop externa otra prueba de estrés, en este caso con la utilización de apache benchamarking tool (ab).

AB es una herramienta de evaluación comparativa de Apache. Está diseñado para dar una impresión de cómo una instalación de Apache funciona. Este en especial muestra cómo la instalación es capaz de servir muchas peticiones por segundo.

Para esta prueba se utilizó un script (Anexo 1) que permitió observar cada una de las peticiones realizadas al servidor web y como resultado se generó un archivo donde se muestra, las conexiones fallidas, total de conexiones establecidas, peticiones por segundo, velocidad de transmisión, etc.

Esta prueba se ejecutó con el siguiente comando:

```
# ./web.sh 450 1000 100 http://172.16.30.2/
```

Dónde:

- ./web.sh : ejecuta el script web.sh
- El parámetro 450 significa el número de veces que se repite el test.
- El parámetro 1000 significa el número total de peticiones por corrida.
- El parámetro 100 significa, cuantas peticiones se realizan a la vez.
- http://172.16.30.2/ es la dirección del servidor web.

Lo anterior envió 1000 solicitudes, 100 a la vez, para 172.16.30.2. Esta prueba se repitió por 450 veces.

Otra de las pruebas que se realizó en forma simultánea, es la descarga de archivos desde el servidor ftp hacia la PC 2, con la finalidad de saturar aún más el CPU y la memoria.

Se realizaron las siguientes instrucciones para la descarga de archivos en la PC 2:

```
# ftp 172.16.30.3
ftp> mget archivo1.....archivo n
```

Finalmente se graficaron los datos obtenidos, luego de las pruebas de estrés realizadas, obteniendo los siguientes resultados.

En lo que se refiere a la sobrecarga del CPU, se observa en la Fig.7, que el consumo de los recursos del procesador es irregular, ya que una vez realizadas las pruebas de estrés, en el primer segundo desciende a un 77%, es decir, el CPU se ha sobrecargado en un 23%, este descenso continua hasta llegar al 47% en el segundo 22, es decir, el CPU llega a un 53% de sobrecarga. Posteriormente se observa que existe una mejora en el rendimiento del CPU, alcanzando a un promedio del 85% durante 54 segundos. Finalmente durante 224 segundos se observa que el CPU mantiene un porcentaje de consumo alto, llegando hasta un 43% en el segundo 218, es decir el CPU se sobrecargó en un 57% como valor máximo y manteniéndose en un promedio del 56.6% hasta terminar las pruebas de estrés.

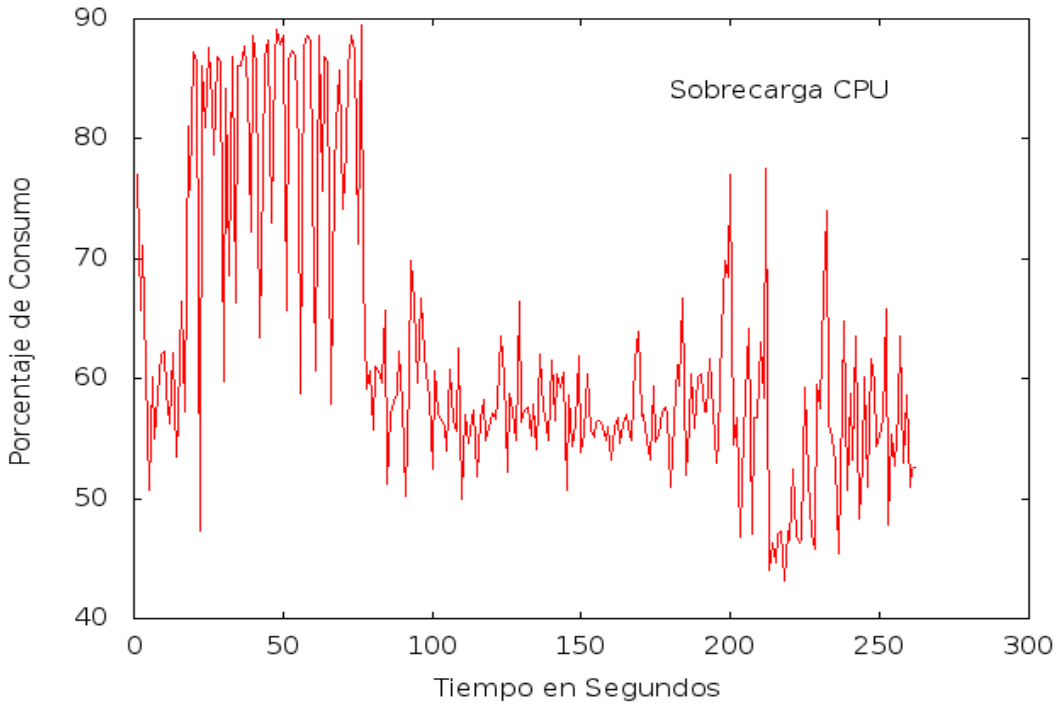


Figura 7: Porcentaje de sobrecarga del CPU

En la Fig.8 se observa la variación de sobrecarga del CPU, donde existe un consumo permanente del procesador entre el 55% y el 65%; es decir, existe una saturación del 45% y 35% por un intervalo mayor de tiempo, del mismo modo se puede visualizar que existe un consumo máximo de aproximadamente el 55%.

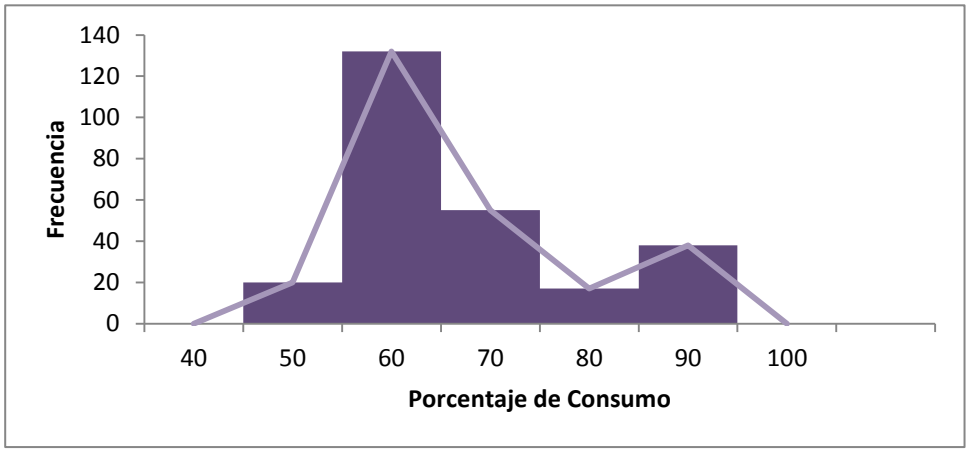


Figura 8: Variación de sobrecarga del CPU

Por otro lado en la Fig.9 se observa que los resultados que generó la herramienta SAR, referente a la memoria son, a medida que se incrementa el porcentaje la memoria está siendo saturada.

En este caso se observa que en los primeros 210 segundos no existe una saturación de memoria considerable, ya que alcanza a un promedio de 36.59% de consumo. Posteriormente se observa que existe un incremento en el consumo de memoria llegando hasta un máximo del 98% en el segundo 262. Este cambio se debe principalmente al tiempo que se encuentra trabajando la memoria, ya que el tiempo de prueba para medir tanto el rendimiento del CPU como el consumo de memoria es de 4 minutos y con pruebas de estrés forzadas.

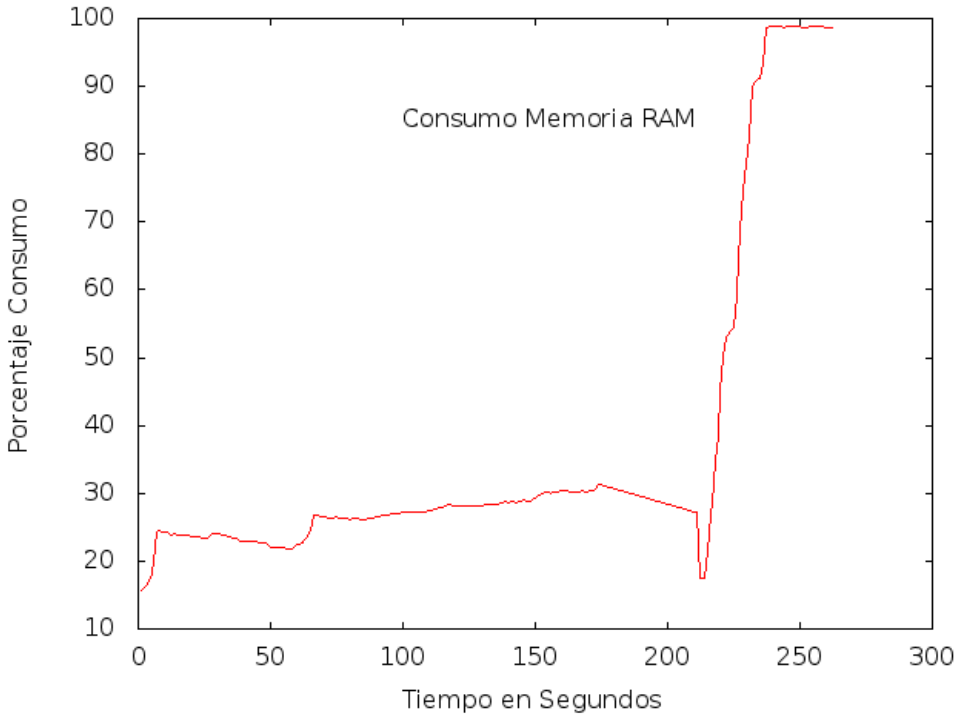


Figura 9: Porcentaje de sobrecarga de Memoria RAM

En la Fig.10 se muestra la variación de sobrecarga de la memoria RAM, donde se observa que existe un intervalo mayor de consumo del 30%, posteriormente se alcanza una saturación de memoria de hasta aproximadamente el 100%.

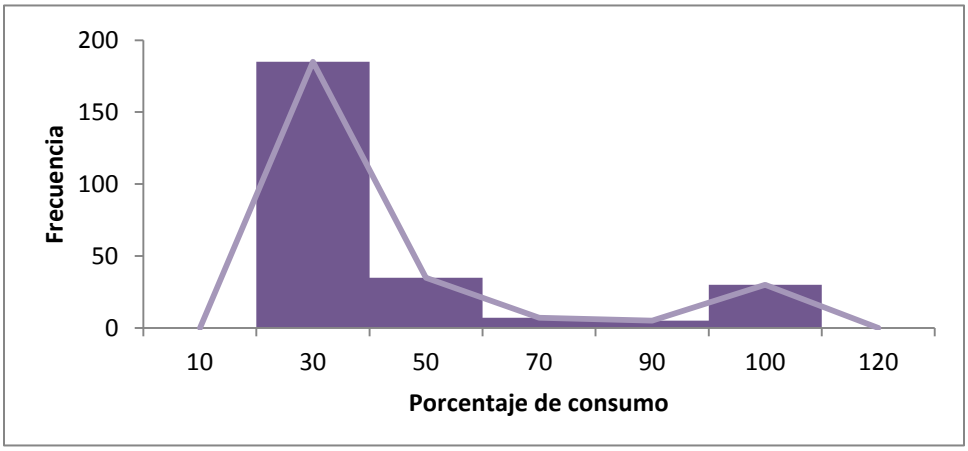


Figura 10: Variación de sobrecarga de Memoria RAM

CAPITULO IV PROPUESTA

4.1 Factores que afectan el desempeño de entornos virtualizados

Según lo analizado en el capítulo III, se determinó que los factores que afectan el desempeño de los entornos virtualizados de red son: la penalización por la utilización de la tecnología de virtualización (overhead), así como las limitaciones de hardware y software del host anfitrión.

Entre las características físicas se tomó en cuenta el tipo de procesador, la capacidad y velocidad de acceso de disco duro y la capacidad de memoria RAM. Dentro de las características lógicas de un CPU, los factores que afectan su rendimiento son: los procesos pesados, el algoritmo de planificación utilizado por el CPU, la técnica de gestión de memoria por defecto, los programas zombis y los parámetros de red mal configurados.

En sistemas operativos basados en UNIX y Linux, un proceso zombie, es un proceso que ha completado su ejecución, pero que todavía tiene una entrada en la tabla de procesos. Cuando un proceso finaliza, toda la memoria y recursos asociados con este son desasignados para que puedan ser usados por otros procesos. De todas maneras, la entrada del proceso permanece en la tabla de procesos. El proceso padre puede leer el estado de salida del proceso hijo ejecutando la llamada del sistema wait. Luego el estado de zombie es eliminado.

En cuanto a los algoritmos de planificación, se ha investigado que los sistemas operativos Linux utilizan algoritmos en tiempo real, como son el FIFO y el Round Robin.

En lo que se refiere a la técnica de gestión de memoria, como se describe en el Capítulo II, Linux gestiona los procesos entre la memoria principal y secundaria, es

decir, utiliza las técnicas de intercambio y paginación de memoria. En la técnica de intercambio utiliza el enfoque basado en particiones de tamaño variable, donde los bloques de memoria son gestionados mediante tres algoritmos: buddy, slab allocator y vmalloc. Por otro lado, la técnica de paginación, divide la memoria física en bloques de tamaños fijos, consumiendo más recursos de memoria y produciendo fragmentación interna.

Los sistemas operativos así como los sistemas bases de ejecución (virtualizador Xen), mantienen procesos inactivos en un segundo plano que se forman luego de ejecutarse, pero que conservan su identificador de registro en la tabla de procesos del sistema. Esto es un aspecto más que impide que el rendimiento de un CPU sea óptimo.

Por otra parte, las plataformas de virtualización permiten crear entornos de redes virtuales, sin embargo, la penalización generada por la capa de virtualización, hace que el rendimiento de la red se degenere.

Finalmente, en el rendimiento del procesador en plataformas virtualizadas o no, los sistemas operativos incrementan en un 70% el rendimiento del CPU, cuando en lugar de procesos pesados se implementan procesos ligeros (hilos). Esto se demuestra también en el tiempo de ejecución, determinándose un 90% menos cuando se ejecutan hilos.

4.2 Activación de mecanismos para incrementar el desempeño de entornos virtualizados.

Como se describió en el numeral 4.1, se ha identificado cinco factores que afectan el rendimiento de un ambiente de red virtualizado, estos son: i) Los procesos pesados; ii) Los algoritmos de planificación del CPU (FIFO y RR); iii) Las técnicas de gestión de memoria (Paginación); iv) Los programas zombis; y, v) Los parámetros de red sobredimensionados.

El propósito de esta investigación es disminuir la penalización que se produce al usar la tecnología de virtualización, por lo que, se pretende activar mecanismos técnicos de gestión de recursos computacionales para incrementar el desempeño en infraestructuras virtualizadas.

En la Tabla 3, se muestra el esquema de solución en forma general donde se plantea varias estrategias que se ha identificado para disminuir la penalización.

Tabla 3

Esquema de solución

PROBLEMA	SOLUCIÒN
Procesos pesados	Utilización de hilos
Métricas de desempeño de red sobredimensionadas.	Utilización de un emulador de red, para configurar parámetros.
Programas zombies	Detección y Eliminación de programas zombies

La solución que se propone para combatir a los procesos pesados, son los hilos; que es un concepto relativamente más eficiente de los S.O.

El término hilo se refiere sintáctica y semánticamente a hilos de ejecución. El término multi-hilo hace referencia a la capacidad de un SO para mantener varios hilos de ejecución dentro del mismo proceso. Un hilo (proceso ligero) es una unidad básica de utilización de la CPU, y consiste en un contador de programa, un juego de registros y un espacio de pila. Los hilos permiten la ejecución concurrente de varias secuencias de instrucciones asociadas a diferentes funciones dentro de un mismo proceso,

compartiendo un mismo espacio de direcciones y las mismas estructuras de datos del núcleo.

En relación a las métricas de desempeño sobredimensionadas, se utiliza un emulador de red llamado Netem con el propósito de configurar parámetros en el entorno virtualizado. En este caso se puede configurar el ancho de banda según la capacidad que se requiera, teniendo en cuenta el ambiente virtualizado, así como la cantidad de información que fluirá a través de ella.

Por otro lado, para combatir los procesos zombis, bastará realizar una secuencia de comandos apropiados que detecten y limpien la tabla de procesos.

Todas las soluciones nombradas anteriormente, son estrategias que se han identificado para disminuir la penalización generada por la capa de virtualización y así aumentar el rendimiento de la red virtualizada.

4.3 Diagrama de propuesta de la solución

La Fig.11, ilustra los mecanismos técnicos de gestión de recursos computacionales que se han propuesto con el objetivo de optimizar el rendimiento del host anfitrión. Se utiliza la combinación de mapas conceptuales que es usado para la representación gráfica del conocimiento, y el diagrama de transición que sirve para ilustrar los estados y las transiciones de los procesos de la solución planteada. En consecuencia, se requiere programar la algorítmica de estos mecanismos, probar y medir en el mismo escenario de prueba (ver Fig. 2), con las mismas condiciones de experimentación de la línea base y compararlas.

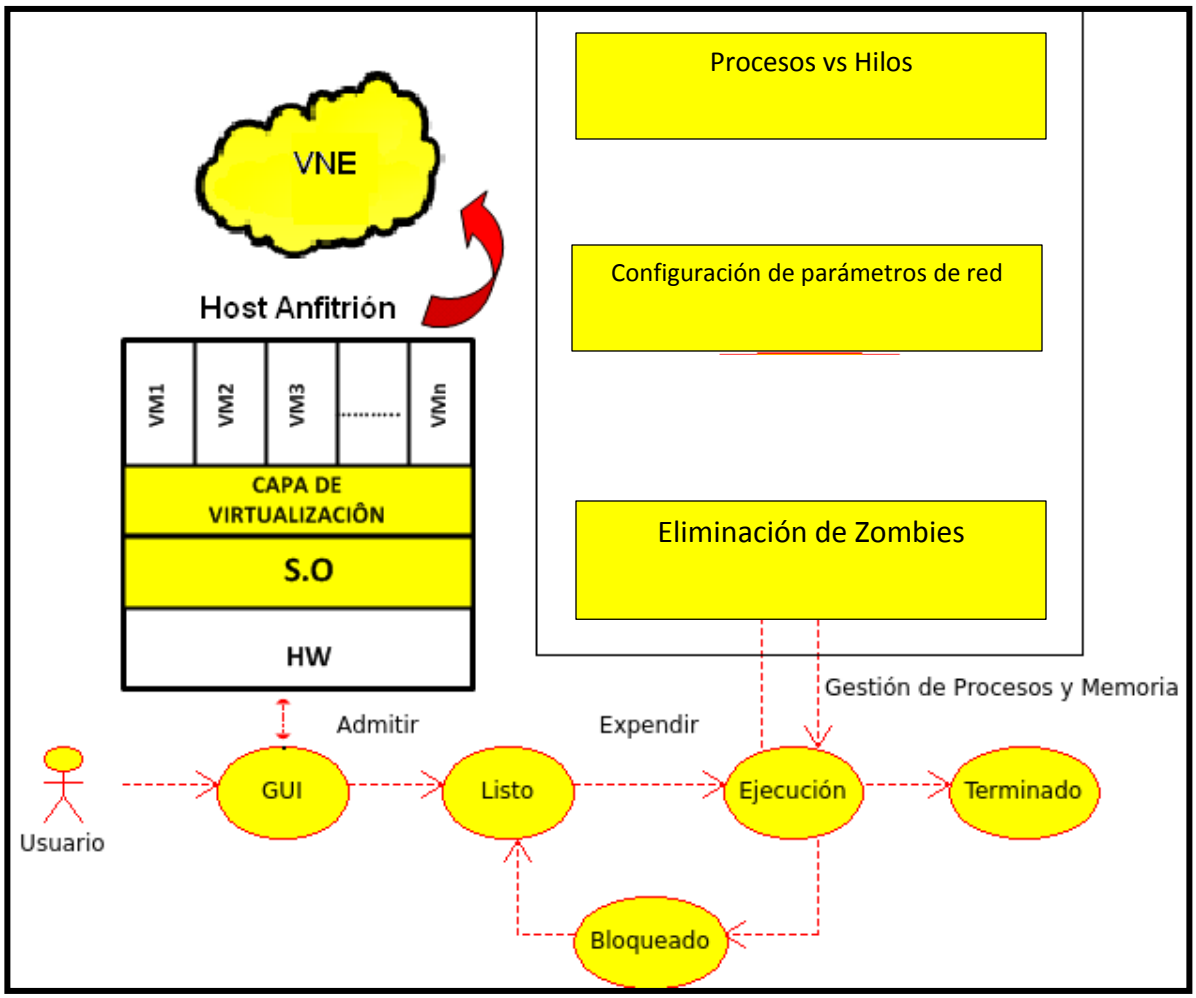


Figura 11: Diagrama de escenario

4.4 Evaluación de resultados

Como se mencionó anteriormente los mecanismos de gestión computacionales que se han identificado para mejorar el rendimiento de la red virtualizada, se los configuraron y se los probaron bajo las mismas condiciones y parámetros de la línea base.

3.4.1 Procesos vs Hilos

Con respecto a la utilización de hilos, se realizó un programa (Anexo 2) que genera el uso de hilos con la finalidad de hacer más rápida la comunicación entre procesos, y medir el consumo del CPU y memoria RAM en las mismas condiciones de la línea base.

En la Fig.12 se puede observar la mejora que existe en el rendimiento del CPU; dado que, mientras el porcentaje de consumo se acerca a cero, el procesador se encuentra saturado. La figura muestra que el rendimiento llega a un pico máximo del 78%, es decir, hay una saturación del 22%. Manteniéndose en un promedio de saturación del 17% durante el tiempo de pruebas, con mejoraras considerables de más del 95%; es decir, el procesador en varios intervalos de tiempo tan solo se ha sobrecargado un 5%.

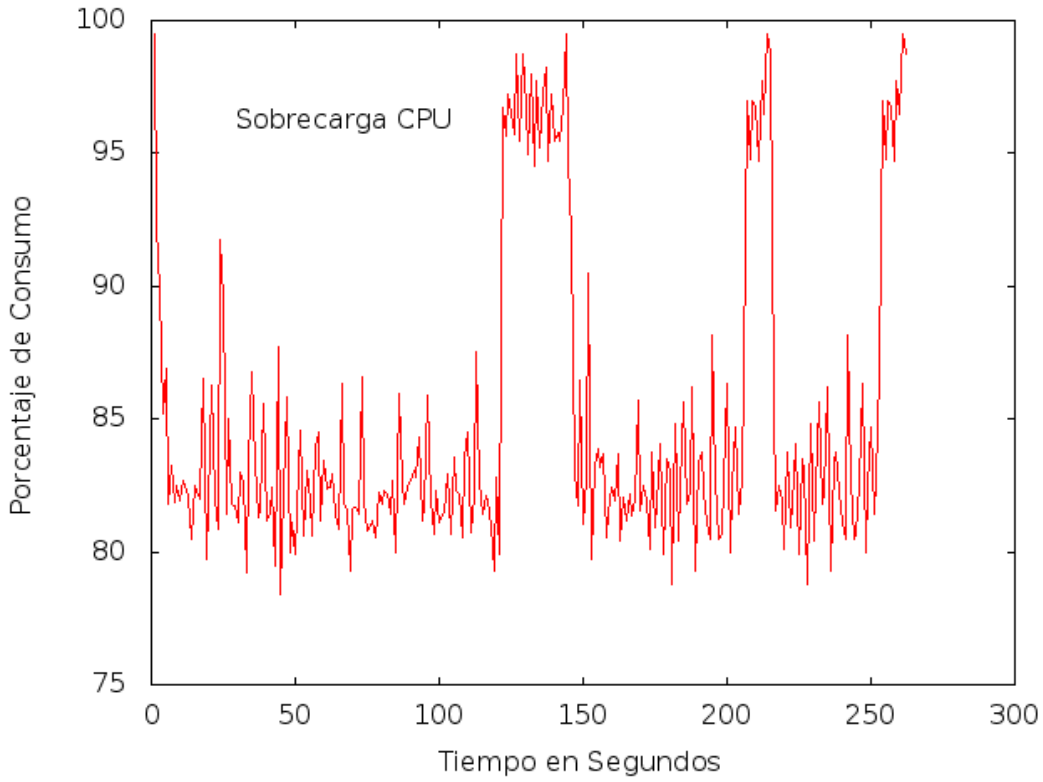


Figura 12: Porcentaje de sobrecarga del CPU con hilos.

En la Fig.13 se puede comparar el porcentaje promedio de consumo del CPU que se midió en la línea base y en la utilización de hilos; probando que, el manejo de hilos optimiza considerablemente el rendimiento del procesador.

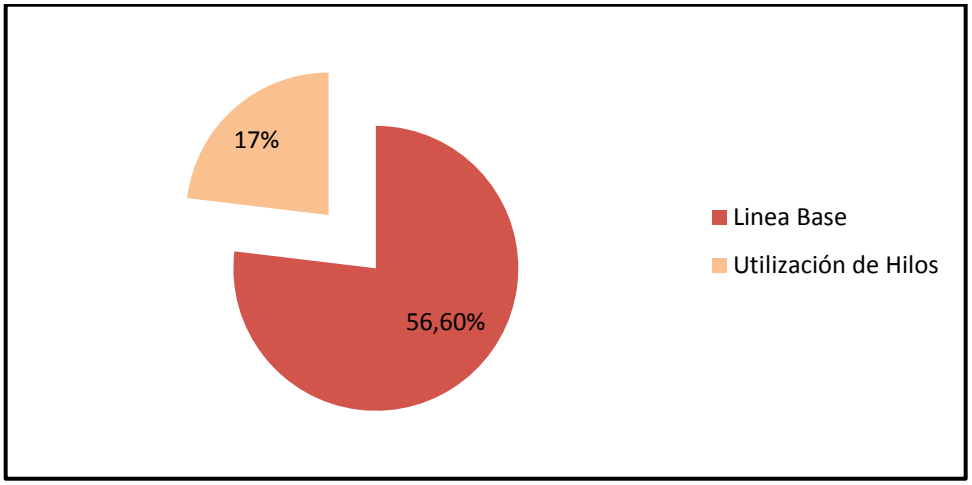


Figura 13: Promedio de Consumo del CPU

En la Fig.14 se puede observar que la memoria RAM alcanza un consumo máximo del 25% y un promedio del 22% durante todo el tiempo de prueba; por lo que, se puede concluir que la memoria RAM mejoró notablemente su rendimiento.

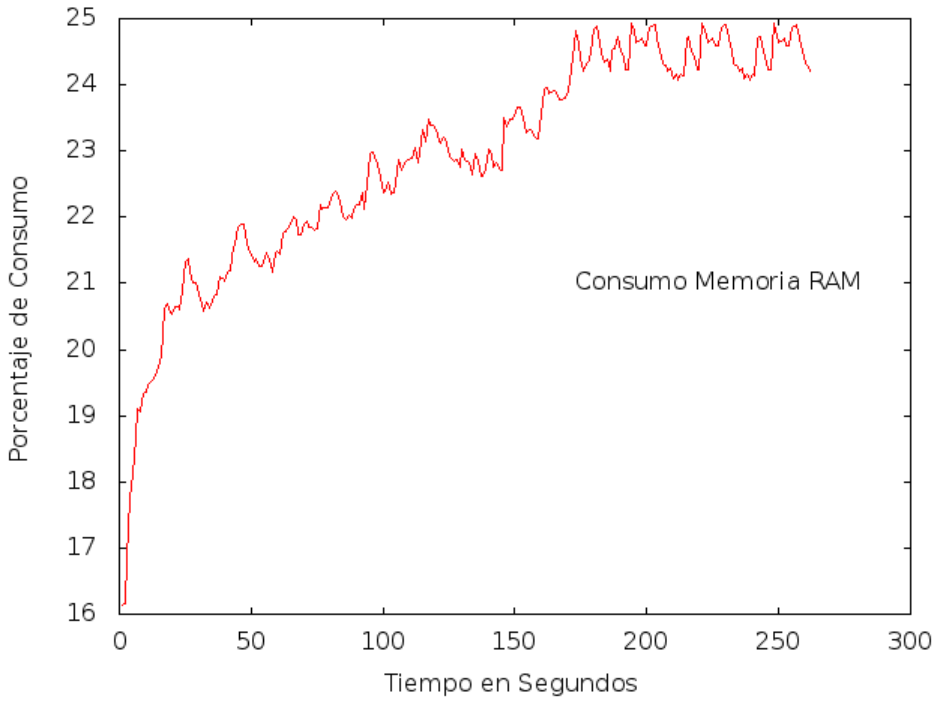


Figura 14: Porcentaje de sobrecarga de Memoria RAM con hilos.

La Fig.15 evidencia la mejora que existe en el consumo de memoria RAM cuando se utiliza hilos, ya que los hilos pertenecientes a un mismo proceso comparten la memoria, datos y los archivos de este, pudiéndose comunicar entre ellos sin necesidad de invocar al sistema operativo.

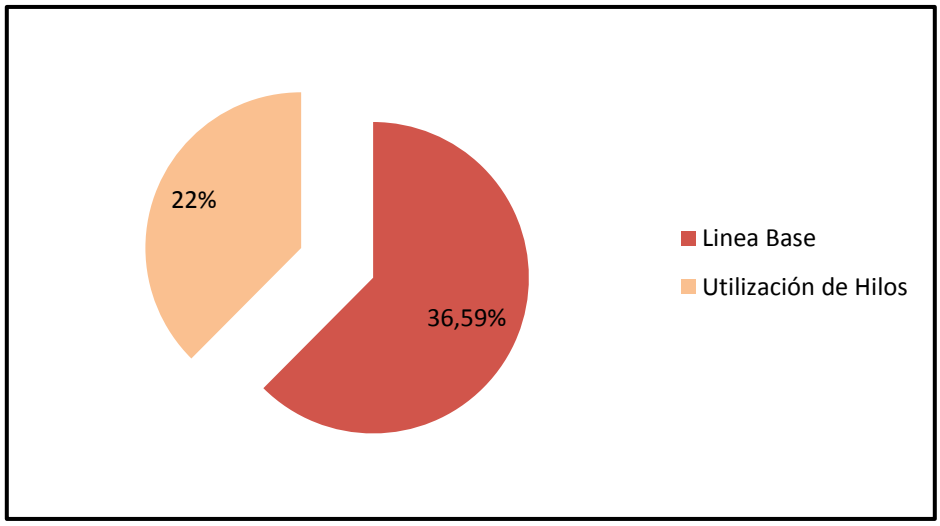


Figura 15: Promedio de Consumo Memoria RAM

3.4.2 Configuración de parámetros de red

Otro de los mecanismos computacionales que se identificó para mejora el rendimiento de la red virtulizada, es la configuración de parámetros de red por medio de un emulador. Para esta prueba se utilizó NetEm que permitió establecer medidas de acuerdo a la capacidad de la red virtualizada. Se redujo el ancho de banda según la necesidad de demanda de los servicios que ofrece la DMZ, sin afectar a los usuarios internos y externos. El emulador de red (NetEm), es una mejora de las facilidades de control de tráfico de Linux que permite añadir retardo, pérdida de paquetes y otros parámetros. En este punto se debe indicar que se ha utilizado esta herramienta para emular el retardo de extremo a extremo similar al obtenido en un entorno real.

En la Fig.16 se puede confirmar que el rendimiento del CPU ha mejorado notablemente cuando se redimensiona los parámetros de red. Llega a un máximo del 67%; es decir, llega a un 33% de saturación, con un promedio del 25.05% de consumo durante todo el tiempo de pruebas.

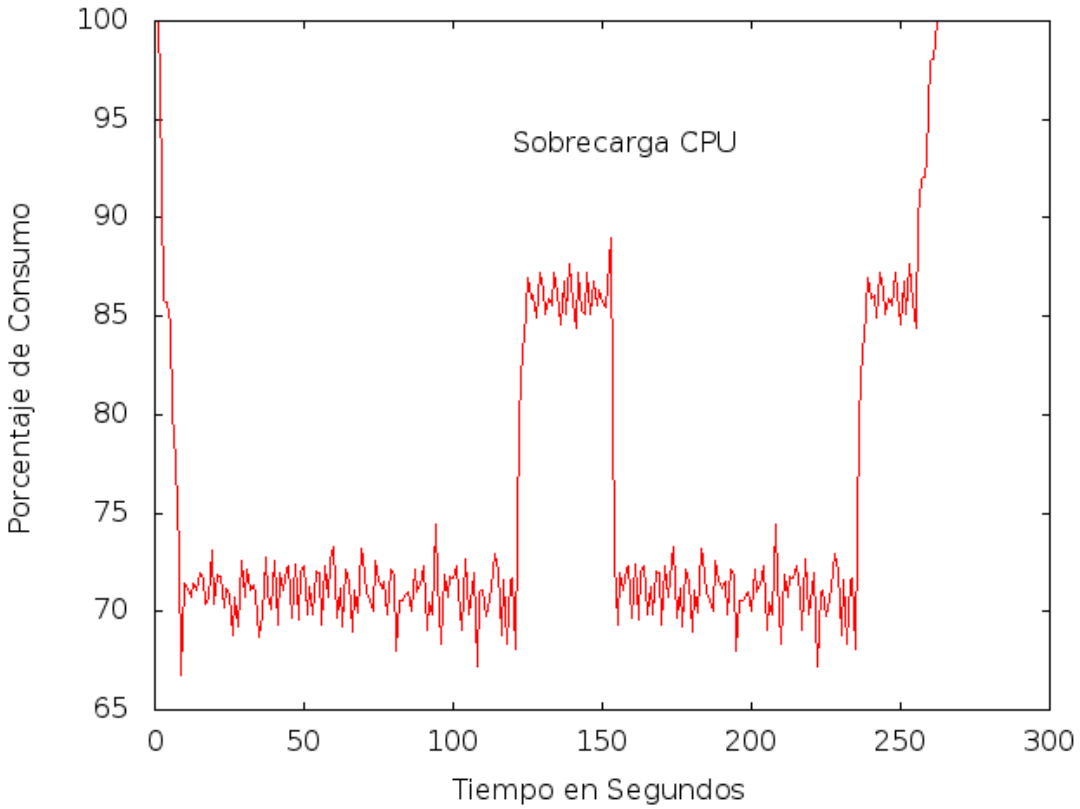


Figura 16: Porcentaje de sobrecarga del CPU al utilizar un emulador de red

En la Fig.17 se observa el promedio de consumo del CPU, comprobando que al configurar parámetros de entorno de red en un ambiente virtualizado se puede mejorar el rendimiento de dicha red.

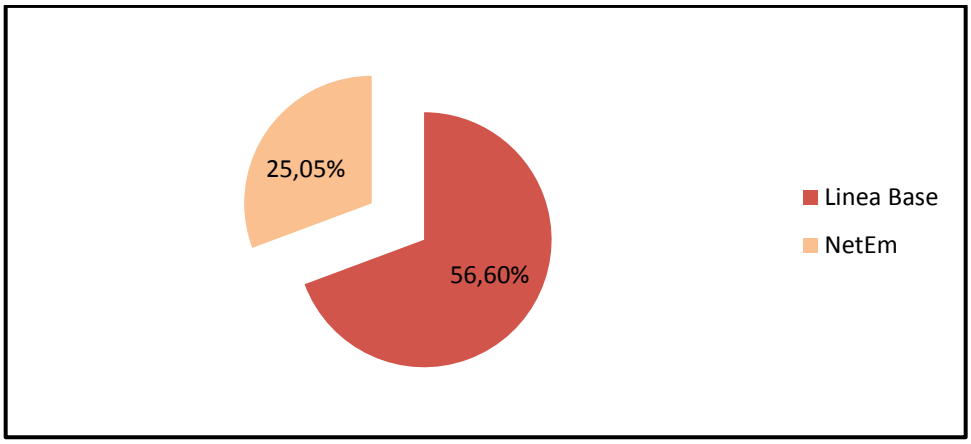


Figura 17: Promedio de Consumo del CPU

En la Fig.18, se puede observar el consumo de memoria RAM, después de haber configurado los parámetros de red, verificando que existe un mejor consumo de la memoria. Llega a un máximo de consumo del 29%, manteniéndose en un promedio del 26% durante todo el tiempo de prueba.

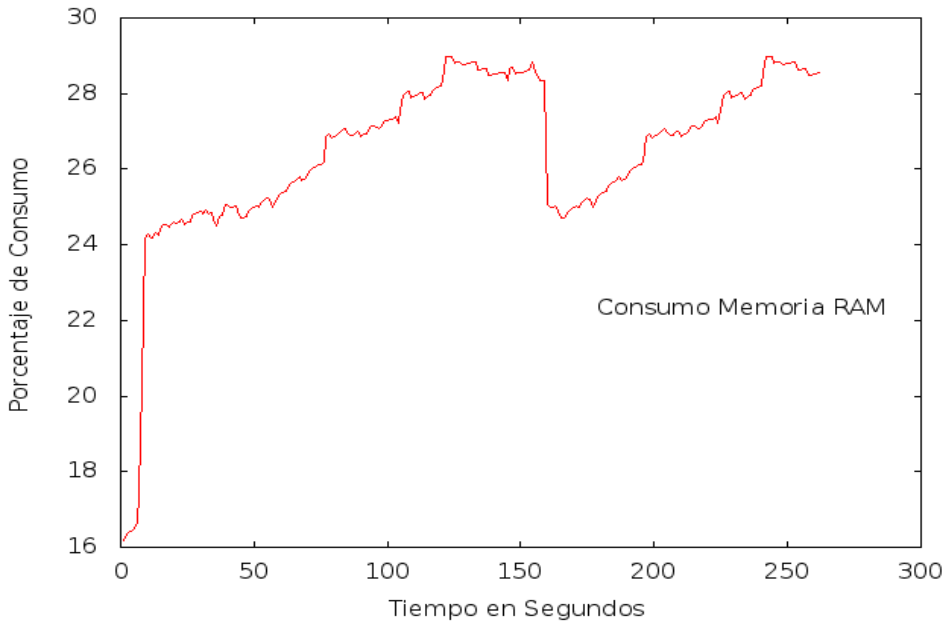


Figura 18: Porcentaje de sobrecarga de Memoria RAM al utilizar un emulador de red

En la Fig.19 se puede verificar que la utilización de un emulador para modificar los parámetros de red según la demanda del entorno de red virtualizado, mejora el consumo de memoria RAM.

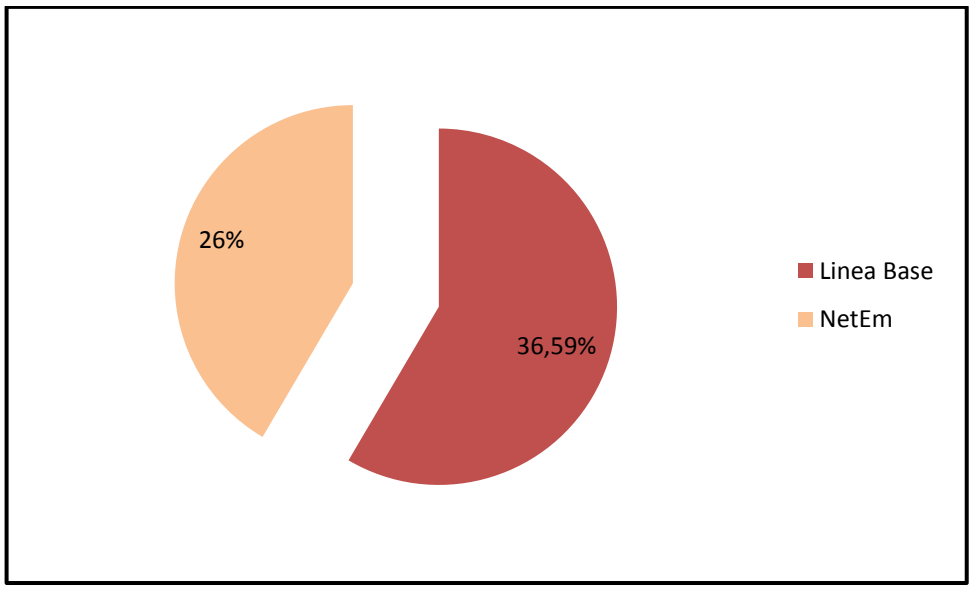


Figura 19: Promedio de Consumo Memoria RAM

3.4.3 Detección y eliminación de procesos zombis

Los procesos zombis son síntomas de un sistema lento o que provoca problemas; ya que, a pesar que su ejecución ha terminado siguen apareciendo en la tabla de procesos, consumiendo recursos computacionales innecesariamente. Para observar que dichos procesos causan problemas en el rendimiento del procesador y la memoria se realizó un programa (Anexo 3) para generar procesos zombis.

En la Fig.20 se observa claramente que al generar procesos zombis el rendimiento del procesador disminuye, de tal forma que llega a un máximo de consumo del 40.2 %; es

decir el procesador llega a saturarse en un 59.8%, manteniéndose en un promedio del 40% durante el tiempo de prueba. Por otro lado se puede identificar que el instante que el procesador queda liberado de los procesos zombis, se llega a un consumo del 47.3%; es decir, existe una saturación del 52.7% como pico máximo, manteniéndose en un promedio del 24.8 % hasta el final de la prueba.

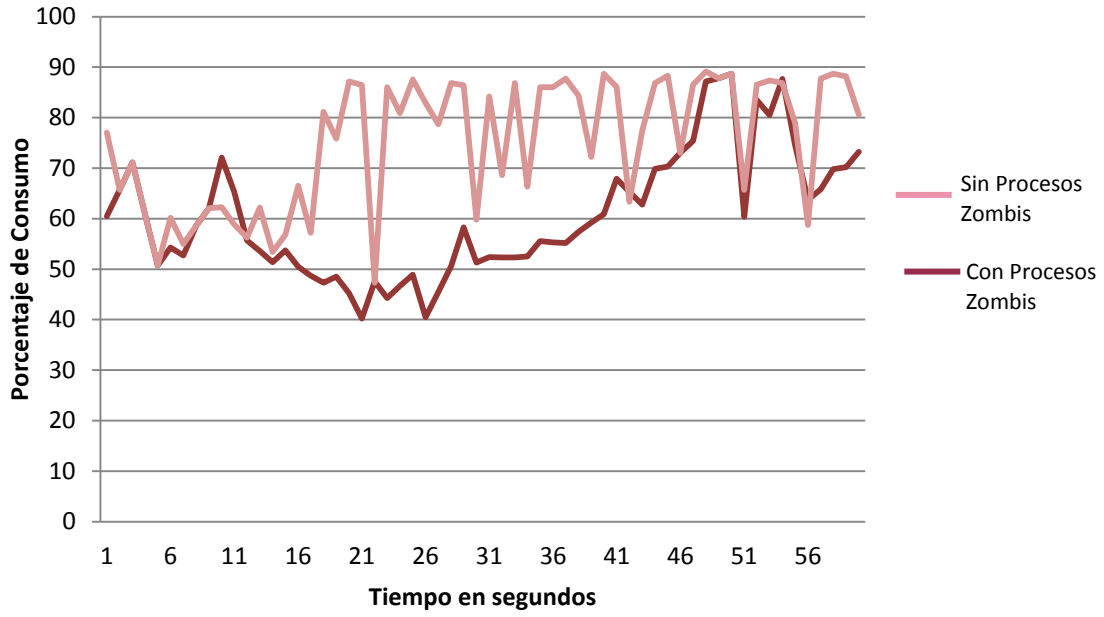


Figura 20: Comportamiento del CPU con procesos zombis y sin procesos zombis.

En la Fig.21 se detalla el consumo promedio del CPU cuando se genera procesos zombis y cuando se elimina los mismos.

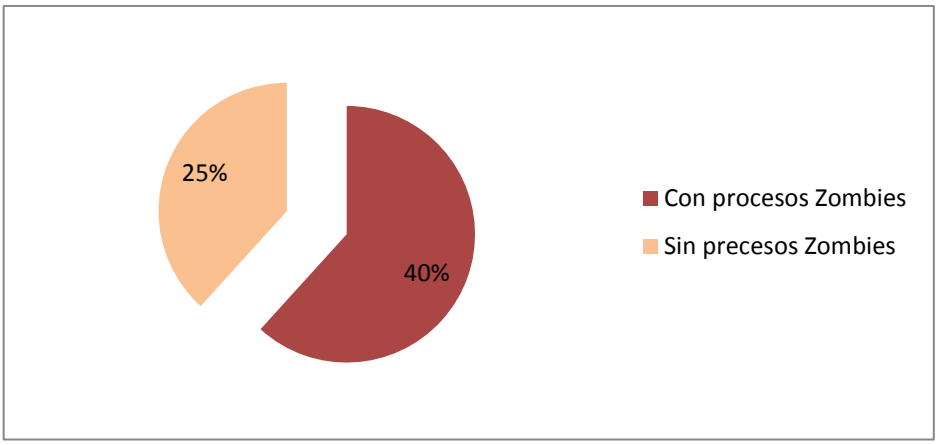


Figura 21: Promedio de consumo del CPU con procesos zombis y sin procesos zombis.

En la Fig.22, se muestra el consumo de memoria RAM cuando se genera procesos zombis, obteniendo un consumo máximo del 28.3%, con un promedio durante todo el tiempo de prueba del 26.5%. Por otro lado, los resultados luego de que la tabla de procesos ha quedado liberada, muestran que existe una mejora en el rendimiento de la memoria, llegando a un consumo máximo del 24% y un promedio del 22.5% durante el tiempo de prueba.

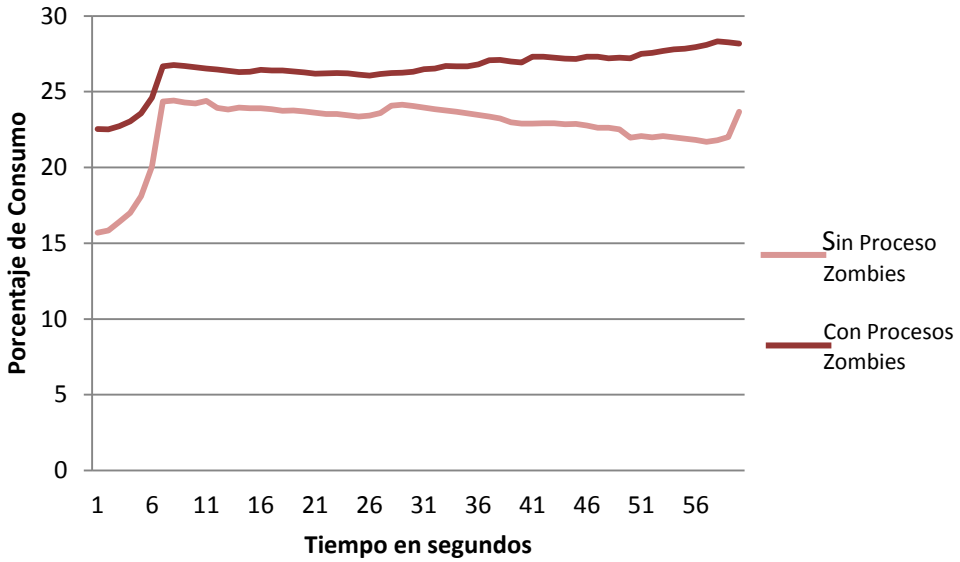


Figura 22: Comportamiento de la memoria RAM con procesos zombis y sin procesos zombis

En la Fig.23 muestra el promedio de consumo de la memoria RAM, al ejecutar y liberar los procesos zombis, demostrando que dichos procesos entorpecen el buen funcionamiento de la memoria.

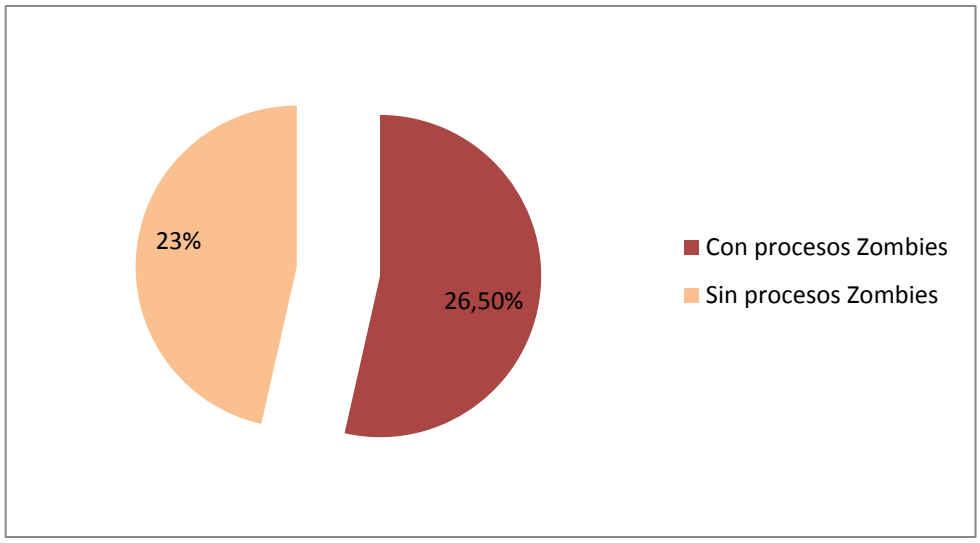


Figura 23: Promedio de consumo de la memoria RAM con procesos zombis y sin procesos zombis.

En la Fig.24 se muestra la comparación de las diferentes técnicas de gestión de recursos computacionales que se utilizó en este proyecto para optimizar el rendimiento del CPU o procesador, comprobando que la utilización de hilos fue la mejor técnica para lograr aquello.

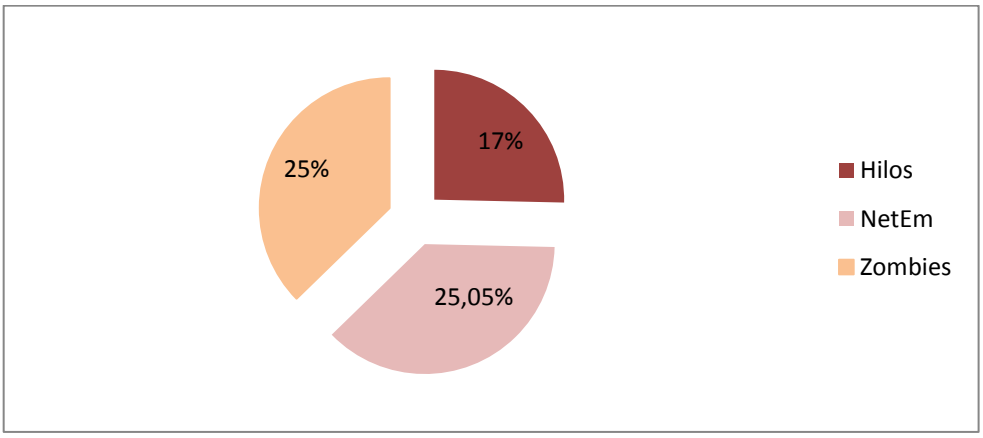


Figura 24: Comparación de las técnicas de gestión de recursos computacionales en el CPU.

En la Fig.25, se observa que la utilización de hilos es la técnica que mejor optimiza el rendimiento de la memoria RAM en un porcentaje mínimo en comparación a las demás técnicas.

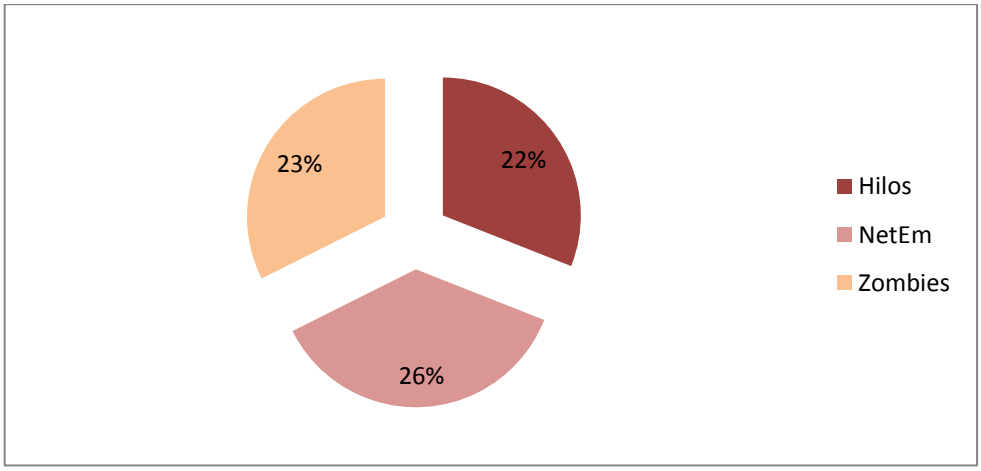


Figura 25: Comparación de las técnicas de gestión de recursos computacionales en la memoria RAM.

CAPITULO V CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

En este trabajo se demostró que los ambientes virtualizados es una estrategia de compartición de hardware y software, son la combinación de los recursos de red del hardware con los recursos de red del software en una única unidad administrativa. Sin embargo, no fue suficiente solo con potenciar físicamente el CPU que se utilizó para este experimento, ni bastó con utilizar un monitor de máquina virtual como XEN para lograr un buen desempeño del procesador y la memoria RAM, ya que como se ha verificado, la capa de virtualización provocó una penalización conocida como overhead.

En este contexto, se diseñó una topología experimental de red híbrida utilizando la plataforma de virtualización XEN bajo la técnica de paravirtualización, la cual permitió alcanzar alto rendimiento.

El ambiente virtualizado de red estuvo conformada por 6 máquinas virtuales, las mismas que fueron configuradas adecuadamente con el propósito de tener conectividad en toda la red.

Al implementar la topología experimental, se consideró que cada una de las máquinas virtuales sean configuradas como volúmenes lógicos; con lo cual se obtuvo una administración totalmente trasparente al usuario. En lo que se refiere a la configuración de la topología experimental, fue necesario la configuración de un conjunto de reglas en el router de borde virtualizado; las cuales habilitaron el DNS, el ICMP, tráfico TCP y UDP y principalmente el bit de forwarding, con el fin de obtener conectividad hacia internet.

Del mismo modo en el firewall virtualizado se configuraron un conjunto de reglas que permitieron principalmente el forward de paquetes para que las pc´s virtualizadas puedan utilizar los servicios de la DMZ y la salida hacia el internet.

Para evaluar el rendimiento de la infraestructura virtualizada diseñada, se realizaron pruebas de estrés hacia los servidores con el fin de obtener una línea base del rendimiento del CPU y la memoria RAM.

La implementación de mecanismos técnicos de gestión de recursos computacionales, permitieron incrementar el rendimiento del ambiente de red virtualizado; pudiendo constatar dicho mejoramiento al comparar los resultados obtenidos de la línea base con los resultados alcanzados luego de la implementación de dichos mecanismos.

La utilización de hilos o procesos ligeros, permitió observar un incremento de mejora considerable en el rendimiento del procesador y la memoria RAM; debido a que, cuando se cambia de un proceso a otro, tiene que intervenir el núcleo del sistema operativo para que haya protección; mientras que, cuando se cambia de un hilo a otro, puesto que la asignación de recursos es la misma, no hace falta que intervenga el sistema operativo.

La utilización de un emulador de red como NetEm para configurar parámetros de red, permitió controlar el desperdicio de recursos de red, como el ancho de banda que se utilizó para realizar las pruebas de línea base, con el fin de evitar la saturación de los servidores virtulizados y obtener una mejor respuesta de rendimiento del procesador y la memoria.

Con la eliminación de los procesos zombis, se observó de igual forma un incremento en el rendimiento del procesador y la memoria.

De las técnicas de gestión de recursos computacionales que se utilizaron, el manejo de hilos fue la que mejor repuesta obtuvo en el cumplimiento del objetivo, al optimizar en un mayor porcentaje el rendimiento del procesador y la memoria RAM que las otras técnicas utilizadas.

5.2 RECOMENDACIONES

La tecnología de virtualización, si bien es cierto ha permitido la separación del hardware y el software, lo cual posibilita a su vez que múltiples sistemas operativos, aplicaciones o plataformas de cómputo se ejecuten simultáneamente en un solo servidor; ha generado en los administradores de red la preocupación por tener un entorno de red virtualizado con alto rendimiento, que permita generar los mismos beneficios en la utilización de los servicios virtualizados que los implementados en un entorno físico.

Varios investigadores han propuesto algunas técnicas para mejorar el rendimiento de un entorno de red virtualizado. Sin embargo, no existe una solución unificada como un segmentador que ofrezca varias soluciones de optimización de forma dinámica. Además las técnicas desarrolladas con el fin de optimizar el rendimiento de los entornos virtualizados deberán ser probadas en un ambiente real, con el fin de verificar la eficiencia en entornos virtuales de alto rendimiento.

BIBLIOGRAFIA

- Al Jabry, H., Liu, L., Zhu, Y., & Panneerselvam, J. (2014, August). A critical evaluation of the performance of virtualization technologies. In Communications and Networking in China (CHINACOM), 2014 9th International Conference on (pp. 606-611). IEEE.
- Fuertes, W., de Vergara, J. L., Pincha, J., Aules, H., Jácome, L., & Grijalva, M. Analytical Expression to Predict the Overhead Produced by the VMware and Xen Virtualization Tools.
- Fuertes, W. M., & de Vergara, J. E. L. (2007, July). A quantitative comparison of virtual network environments based on performance measurements. InProceedings of the 14th HP Software University Association Workshop.
- Fuertes, W. (2009). An emulation of VoD services using virtual network environments. Electronic Communications of the EASST, 17.
- Fuertes, W., Lopez de Vergara, J. E., Meneses, F., & Galan, F. (2010, April). A generic model for the management of virtual network environments. InNetwork Operations and Management Symposium (NOMS), 2010 IEEE (pp. 813-816). IEEE.
- Fuertes, W., Jácome, L., Grijalva, M., de Vergara, J. L., & Fonseca, R. (2009). Evaluación del Rendimiento de Redes IP utilizando Plataformas de Virtualización y Métodos de Simulación. ESPE-DECC "DECC Report Tendencias en Computación", 34.
- Fuertes, W., Vilac, L., & Gallo, D. (2012). Laboratorios de computación multiplataforma aplicando tecnologías de virtualización.
- Herramientas de monitoreo en Linux. Disponible en: http://oracleexperiences.wikispaces.com/file/view/monitorizar_sistemas_linux.pdf. Última comprobación 10/10/2013.
- Quintero Aspiazu, H., & Silva Naranjo, L. (2013). Medición del desempeño y monitoreo de una solución con servicios web transaccional.

- Gad, R., Kappes, M., Mueller-Bady, R., & Ritter, I. (2011, December). Network performance in virtualized environments. In Networks (ICON), 2011 17th IEEE International Conference on (pp. 275-280). IEEE.
- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., & Warfield, A. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177.
- Li, S., Hao, Q., Xiao, L., & Xu, Q. (2009, December). Optimizing network virtualization in kernel-based virtual machine. In Information Science and Engineering (ICISE), 2009 1st International Conference on (pp. 282-285). IEEE.
- Menon, A., Cox, A. L., & Zwaenepoel, W. (2006, May). Optimizing network virtualization in Xen. In USENIX Annual Technical Conference (pp. 15-28).
- Regola, N., & Ducom, J. C. (2010, November). Recommendations for virtualization technologies in high performance computing. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on (pp. 409-416). IEEE.
- Chen, H., Shou, G., Hu, Y., & Guo, Z. (2012, September). Virtual network mapping algorithm with robustness in network virtualization. In Network Infrastructure and Digital Content (IC-NIDC), 3rd IEEE International Conference on (pp. 314-318). IEEE.
- García Calahorro, A. (2009). Estudio de desempeño y funcionalidad sobre diferentes soluciones de virtualización.
- Belén Sotaminga, M., Guerrero Valarezo, C., Abad Eras, A., & Astudillo, I. G. (2011). Implementación de un ambiente de Virtualización para el manejo de múltiples servidores de VoIP sobre una plataforma común de hardware.
- Martínez, S. G. (2008). Evaluación de plataformas virtuales: Estudio comparativo.
- Fuertes, W., & de Vergara, J. L. (2008). Evaluación de Plataformas de Virtualización para Experimentación de Servicios Multimedia en Redes IP. aceptado para su

publicación en la Revista de Ciencia de la Escuela Politécnica del Ejército, Sangolquí, Ecuador.

- Meneses, F., Fuertes, W., Guerra, L., de Vergara, J. L., & Aules, H. (2011). Modelo Distribuido para la Gestión de Entornos Virtuales de Red. In Publicado en las memorias del VI Congreso de Ciencia y Tecnología ESPE.
- IP Performance Metrics (ippm). Disponible en: http://datatracker.ietf.org/wg/ippm/charter/.
  Última comprobación 10/10/2013.
- IPPM Metrics for Measuring Connectivity. Disponible en: http://www.ietf.org/rfc/rfc2678.txt. . Última comprobación 10/10/2013.
- A One-way Delay Metric for IPPM. Disponible en: http://www.rfc-editor.org/rfc/pdfrfc/rfc2679.txt.pdf. Última comprobación 10/10/2013.
- A One-way Packet Loss métricas para IPPM. Disponible en: http://www.rfc-editor.org/rfc/pdfrfc/rfc2680.txt.pdf.
  Última comprobación 29/10/2013.
- A Round-trip Delay Metric for IPPM. Disponible en: http://www.rfc-editor.org/rfc/pdfrfc/rfc2681.txt.pdf. Última comprobación 10/10/2013.
- Round-Trip Packet Loss Metrics. Disponible en: http://www.rfc-editor.org/rfc/pdfrfc/rfc6673.txt.pdf. . Última comprobación 10/10/2013.
- Álvarez Moreno, M., Gimeno, L., & Padellano Avilés, C. (2010). Análisis de desempeño y fiabilidad de sistemas informáticos.
- Santos, J., Quesada A., Santana F. Sistemas Operativos Tema 6. Planificación de Procesos. [diapositiva]. Universidad de las Palmas de Gran Canaria.
- Rivas, M. A. (2002). Planificación de Tareas en Sistemas Operativos de Tiempo Real Estricto para Aplicaciones Empotradas. Memoria presentada para obtener el grado de Doctor en Ciencias Físicas, Universidad de Cantabria.

- Sistemas Operativos. Algoritmo FIFO. Disponible en: http://cnarea21.blogspot.com/2012/02/algoritmo-fifo.html. Última comprobación 11/10/2013.
- Sistemas Operativos. Algoritmo Round Robin. Disponible en: http://cnarea21.blogspot.com/2012/02/algoritmo-round-robin.html. Última comprobación 11/10/2013.
- Sistemas Operativos. Disponible en: http://es.wikiversity.org/wiki/Sistemas_operativos. Última comprobación 11/10/2013.
- Planificación por prioridad. Disponible en: http://algoritmosplanificacion.blogspot.com/2012/08/planificacion-por-prioridad-sjf-short.html.Última comprobación 11/10/2013.
- Romero D. Gestión de memoria. Disponible en: http://www.monografias.com/trabajos13/gesme/gesme.shtml. Última comprobación 11/10/2013.
- Gestión de memoria. Disponible en: http://www.arcos.inf.uc3m.es/~ssoo-va/ssoo-va/libro/pdf/cap04.pdf. Última comprobación 11/10/2013.
- Gust.A. Sistemas operativos – Gestión de memoria. Disponible en: http://es.scribd.com/doc/19063531/003-Sistemas-Operativos-Gestion-de-Memoria. Última comprobación 12/10/2013.
- Agramon. M. (2011). Segmentación De Memoria Y Paginacion De Memoria. Disponible en: http://www.buenastareas.com/ensayos/Segmentacion-De-Memoria-y-Paginacion-De/1884300.html. Última comprobación 12/10/2013.
- Prieto, A., Lloris, A., & Torres, J. C. (1995). Introducción a la Informática. McGraw-Hill.

- Romero D. Gestión de memoria. Disponible en: http://www.monografias.com/trabajos13/gesme/gesme.shtml. Última comprobación 12/10/2013.
- Segmentación de memoria. Disponible en: http://www.dc.fi.udc.es/~so-grado/SO-Memoria.pdf. Última comprobación 12/10/2013.
- Romero D. Gestión de memoria. Disponible en: http://www.monografias.com/trabajos13/gesme/gesme.shtml. Última comprobación 12/10/2013.
- La Red Martínez D. Sistemas Operativos. Disponible en: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/MEMVIR02.htm.Última comprobación 12/10/2013.
- Gestión de Memoria. Disponible en: http://www.ual.es/~acorral/DSO/Tema_3.pdf. Última comprobación 13/10/2013.
- Hemminger, S. (2005, April). Network emulation with NetEm. In Linux Conf Au (pp. 18-23).
- García Fernández, V. N. (2011). Scalability study of Guifi. net and mesh networks.
- Saldaña, J. M., Murillo, J., Fernández-Navajas, J., Ruiz-Mas, J., Viruete, E. A., & Aznar, J. I. Emulación de escenarios de red mediante un testbed.
- Álvarez Moreno, M., Gimeno, L., & Padellano Avilés, C. (2010). Análisis de desempeño y fiabilidad de sistemas informáticos.
- Httperf. Disponible en: https://admin.fedoraproject.org/pkgdb/acls/name/httperf. Última comprobación 13/10/2013.
- Tcpdump. Disponible en: http://www.tcpdump.org/. Última comprobación 13/10/2013.
- Wireshark. Disponible en: http://www.wireshark.org/. Última comprobación 13/10/2013.
- IPERF. Disponible en: http://www.iperf.fr/. Última comprobación 13/10/2013.

- Medición y desempeño de una Red. Disponible en: http://www.geocities.ws/jcredesii/REDES2-44.htm. Última comprobación 13/10/2013
- Quintero Aspiazu, H. E. N. R. Y., & Silva Naranjo, L. E. N. I. N. (2013). Medición del desempeño y monitoreo de una solución con servicios web transaccional (Doctoral dissertation).
- Álvarez Camacho, C. R., & Mise Luzardo, j. l. (2013). Diseño e implementación de un ambiente virtualizado usando varias plataformas de correo electrónico e integrándose entre ellos (doctoral dissertation).
- Peñafiel Escalante, D. A., & Mendoza Bustamante, J. S. (2012).Implementación y medición de un ambiente virtualizado de servidores que se comunican por medio de enlaces remoto (Doctoral dissertation).
- Mendoza Marchan, c. j., & Cordovilla Salinas, M. Y. (2012).Diseño e implementación de un almacenamiento virtualizado de los respaldos de servidores virtualizados (doctoral dissertation).
- GNUPLOT. Disponible en: http://www.gnuplot.info/. Última comprobación 20/10/2013.
- Taringa. Anonymous Ataque DDOS y VPN anonimato. Disponible en: http://www.taringa.net/posts/info/13764466/Anonymous-Ataque-DDOS-y-VPN-anonimato.html.Última comprobación 20/10/2013.
- TechnoBlog. Como medir la performance de Apache (Comando ab). Disponible en: http://www.technoblog.com.ar/index.php/2010/01/como-medir-la-performance-de-apache/.
  Última comprobación 20/10/2013.
- Procesos vs Hilos. Disponible en: http://systope.blogspot.com/2012/05/procesos-e-hilos.html. Última comprobación 15/11/2013.
- Stephen Hemminger: "**Network Emulation with NetEm**". Open Source Development Lab. April 2005.

- Procesos Zombies en Ubuntu. Disponible en: http://xombra.com/index.php?do/articulos/nota/202/op/5/t/matar-procesos-zombie-ubu. Última comprobación 15/01/2015.

ANEXOS

Anexo 1: El siguiente script permitió automatizar la prueba de estrés hacia el servidor web que se genera con lo herramienta ab (Apache Bench). En primer lugar se muestra el pseudocódigo que permite entender fácilmente lo que realiza el programa desarrollado en bash.

PSEUDOCÓDIGO

Si [web.sh no corre]; entonces
  escribir "ERROR: script requiere apache bench"
  escribir "Para Debian o sus derivados instalar 'apt-get install apache2-utils'"
  escribir "Si ya lo tiene instalado, posiblemente usted no tiene permisos para ejecutar esto, trate sudo $(basename $0)"
Salida
Fi

Si [web.sh no tiene argumentos]; entonces
  escribir "ERROR: script necesita 4 argumentos, Donde:"
  escribir "1. Número de veces que se repite el test (e.g. 15)"
  escribir "2. Número total de peticiones por corrida (e.g. 150)"
  escribir "3. Cuantas repeticiones hace a la vez (e.g.50)"
  escribir "4. URL del sitio de prueba (e.g. http://172.16.30.2/)"
  escribir "Ejemplo:"
  escribir " $(basename $0) 15 100 50 http://172.16.30.2/"
  escribir "El ejemplo enviará 150 repeticiones (50 a la vez) hacia http://172.16.30.2. La prueba se repetirá 15 veces."
Salida
Fi

Caso contrario

Número de corridas (runs=$1)
Número de peticiones (numero=$2)
Número de repeticiones a la vez (concurrencia=$3)
Sitio web (sitio=$4)

log=ab.log Genera un archivo ab.log

Si [archivo $log existe]
  escribir removiendo el archivo $log
  remueve el $log
Fi
  escribir ¨Resultados¨
  escribir ¨sitio web……$sitio¨
  escribir ¨peticiones…….$numero¨
  escribir ¨concurrencia….$concurrencia ¨

En la variable run del bucle for (crea una secuencia que empieza en 1 y termina en el valor de la varible $runs); hace
  la salida de ab con la variables $concurrencia, $numero y $sitio >> se adjunte en el archivo log
  escribe ¨Peticiones por segundo¨
Hecho

avg= realiza un promedio de peticiones por segundo
escribe ¨promedio…….$avg   peticiones/sec¨
escribe ¨ ver $log para más detalles¨

SCRIPT EN BASH

```
#!/bin/bash

if ! [ -x "$(type -P ab)" ]; then
  echo "ERROR: script requiere apache bench"
  echo "Para  Debian o sus derivados  instalar 'apt-get install apache2-utils'"
  echo "Si ya lo tiene instalado, posiblemente usted no tiene permisos  para ejecutar esto, trate sudo $(basename $0)'"
  exit 1

fi

if [ "$#" -ne "4" ]; then
  echo "ERROR: script necesita 4 argumentos, Donde:"
  echo
  echo "1. Número de veces que se repite el test (e.g. 15)"
  echo "2. Número total de peticiones por  corrida (e.g. 150)"
  echo "3. Cuantas repeticiones hace a la vez (e.g.50)"
  echo "4. URL del sitio de prueba (e.g. http://172.16.30.2/)"
  echo
  echo "Ejemplo:"
```

```
  echo "  $(basename $0) 15 100 50 http://172.16.30.2/"
  echo
  echo "El ejemplo enviará  150 repeticiones (50 a la vez) hacia http://172.16.30.2. La
prueba se repetirá 15 veces."
  exit 1

else

  runs=$1
  number=$2
  concurrency=$3
  site=$4

fi

log=ab.$(echo $site | sed -r 's|https?://||;s|/$||;s|/|_|g;').log
if [ -f $log ]; then
  echo removing $log
  rm $log

fi

echo
"============================================================="
echo " Resultados"
echo
"============================================================="
echo " sitio web ..... $site"
echo " peticiones..... $number"
echo " concurrencia... $concurrency"
echo "-------------------------------------------------------------"

for run in $(seq 1 $runs); do
  ab -c$concurrency -n$number $site >> $log
  echo -e " run $run: \t $(grep "^Repeticiones por segundo" $log | tail -1 | awk
'{print$4}') reqs/sec"

done

avg=$(awk -v runs=$runs '/^Repeticiones por segundo/ {sum+=$4; avg=sum/runs}
END {print avg}' $log)
echo "-------------------------------------------------------------"
echo " promedio ..... $avg requests/sec"
```

```
echo
echo "ver $log para más detalles"
```

Anexo 2: Script en c para crear procesos e hilos.

Creación de Procesos: En este programa en c crea 1000 procesos con la ayuda de un for.

```
#include <unistd.h>
#include ¨stdio.h¨
void main()
{
        pid_t pid;
        int j;
        int n=10;
        printf(¨\n¨);
        for(j=0;j<1000;i++)
        {
                pid=fork();
                if(pid==0)
                 break;
        }
        printf(¨El proceso padre de %d es: %d\n¨, gerpid(),getppid());
}
```

Creación de hilos: En este programa en c se crean 1000 hilos, aquí se incluye la librería phtread.

```
#include ¨pthread.h¨
#include ¨stdio.h¨
void *func(void *temp)

{
        int b=pthread_self();
        printf(¨Thread %d\n¨,a);
        pthread_exit(0);
        }
        int main()
        {
        pthread_t hilo;
        int j;
        for(j=0;i<=1000;j++)
        {
        pthread_crate(&hilo,NULL,&func,NULL);
        }
}
```

Anexo 3: Programa en c para crear programas zombis. Este código crea procesos zombis por 60 segundos.

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
  Printf (¨Generación de procesos zombis¨);

  pid_t child_pid;

  child_pid = fork ();
  if (child_pid == 0)
 {
    sleep (60);
  }
  else
  {
    exit (0);
  }
  return 0;
}
```

Adicionalmente se generaron procesos zombis cuando se realizaron las pruebas de línea base y se pudo identificar mediante el comando ***top.***

Para conocer que procesos zombis existieron se escribió en consola:

```
$ ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]'
```

Dónde:

ps -A solicita listado de todos los procesos mostrando el **ostat** (estado), el **ppid** (proceso padre), el**pid** (proceso), el **cmd** (como se lanzó el proceso) y mediante una tubería (pipe) (|) creamos el filtro apropiado **grep -e** de los procesos **Zz** (zombi)

Para eliminar todos los procesos zombis se usó el siguiente comando:

```
$sudo kill -HUP `ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]' | awk '{print $2}'`
```

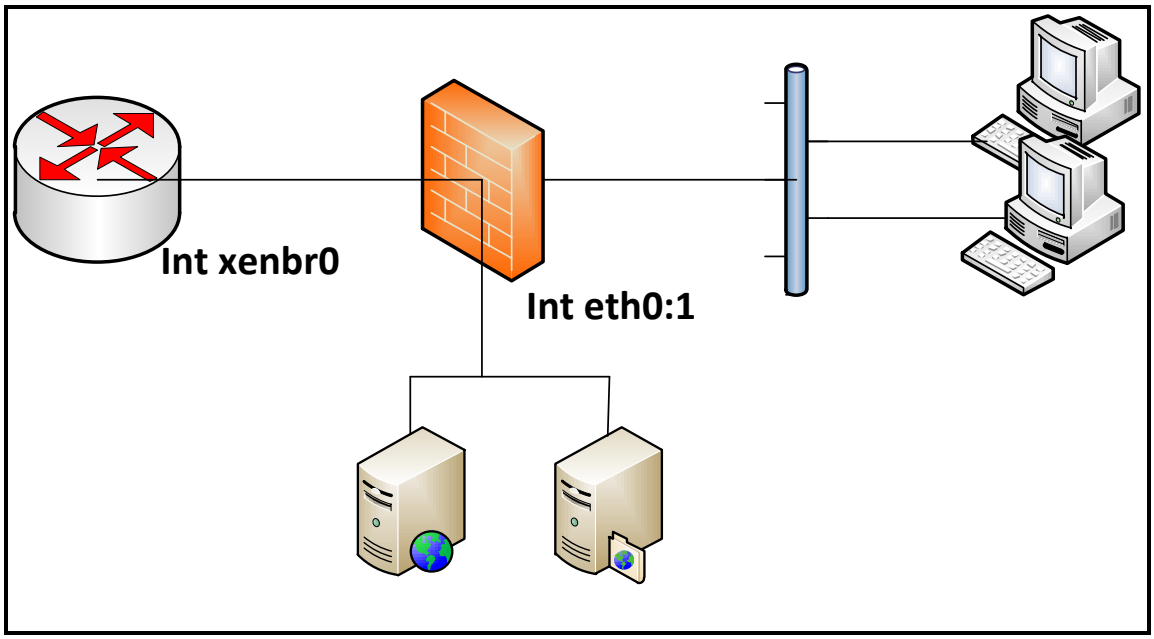
Dónde:

- ps -A -ostat,ppid,pid,cmd | grep -e '^[Zz]': solicita un listado con filtro sobre los procesos [Zz]ombi.

- awk '{print $2}': Filtra por la columna 2 del filtrado anterior. Corresponde al PrentPID (PPID).

- Kill –HUP; HUP (Hang UP), obliga al proceso a reiniciarse examinando sus archivos de configuración.

Anexo 4: Grafica que muestra sobre que interfaces se configuraron los parámetros con netem.



Los qdisc (queuing discipline) se configuró en la interfaz virtual de xen y en lasub-interfaz eth0:1 del firewall. Tomando en cuenta que el qdisc asignado a las interfaces afectará solo al tráfico que salga por dichas interfaces y no al entrante.

El ancho de banda se redujo a la mitad tanto para los usuarios externos como para los usuarios internos.

El comando ***tc (Traffic control)*** permitió configurar ancho de banda de acuerdo a las necesidades de la red.

Para verificar si existen qdisc con figuradas, bastó con el siguiente comando

```
$ tc qdisc show
```

EL siguiente comando permitió configurara el ancho de banda en la Interfaz Virtual

```
$ tc qdisc add dev xenbr0 root handle 1:0 tbf  rate 4000kbit burst 2048 latency 50ms
```

EL siguiente comando permitió configurara el ancho de banda en la Interfaz en Firewall

```
$ tc qdisc add dev eth0:1 root handle 1:0 tbf  rate 81920kbit burst 2048 latency 50ms
```

FECHA DE ENTREGA DE LA TESIS

El presente documento fue entregado en la Dirección de Postrgrado, reposando en la Universidad de las Fuerzas Armadas ESPE, desde:

Sangolquí, 11 de Septiembre de 2015

Ing, Maritza Alexandra Núñez Solís.

AUTOR

Ing. Darwin Alulema, Mgs

COORDINADOR DE LA MAESTRÍA EN REDES DE INFORMACIÓN Y CONECTIVIDAD