ESCUELA POLITÉCNICA DEL EJÉRCITO

DPTO. DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS DE COMPUTACIÓN E INFORMÁTICA

EVALUACIÓN DEL RENDIMIENTO DE REDES IP UTILIZANDO PLATAFORMAS DE VIRTUALIZACIÓN Y MÉTODOS DE SIMULACIÓN

Previa a la obtención del Título de:

INGENIERO DE SISTEMAS DE COMPUTACIÓN E INFORMÁTICA

POR: MARÍA FERNANDA GRIJALVA SUÁREZ Y
DARWIN LEONARDO JÁCOME MORENO

SANGOLQUÍ, 11 de Noviembre del 2009

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por la Srta. María Fernanda Grijalva Suárez y el Sr. Darwin Leonardo Jácome Moreno como requerimiento parcial a la obtención del título de INGENIEROS EN SISTEMAS DE COMPUTACIÓN E INFORMÁTICA

_____

Sangolquí, 11 de Noviembre del 2009

_____

MSc. Walter Marcelo Fuertes Díaz

DEDICATORIA

A mi Dios, a mis padres Marco y Ligia, a mis hermanos, sobrinos y a mi compañero del alma.

MaFer

A Dios, a mis padres, a mi familia, a mi compañera de siempre.

Leo

AGRADECIMIENTOS

Agradezco en primer lugar a mi Dios por ser un ser de luz, omnipotente que me lleva siempre bajo su manto, a mis padres y hermanos, por su infinita ayuda, paciencia, y amor.

Por supuesto agradezco a la Escuela Politécnica del Ejército que me brindo la oportunidad de desarrollarme como profesional, y aportar a la sociedad. A mis queridos compañeros de clases, por los buenos momentos que vivimos aprendiendo y siendo amigos.

A mis apreciados maestros, por sus sabias enseñanzas. A nuestro codirector, por su generosa ayuda, y consejos. Finalmente, quiero agradecer a mi director, que más que nuestro guía, maestro, es y será siempre nuestro gran amigo.

MaFer

En primer lugar debo agradecer a Dios por su infinita sabiduría, por hacer que la gente tenga fé y con ello logre sus objetivos, a nuestra querida politécnica y todos sus docentes y personal por convertirme en un hombre de bien.

A mis padres, mis hermanos que están lejos y los que están cerca, por el apoyo siempre e incondicional, a la familia de mi compañera por el apoyo, paciencia y comprensión en todo el trayecto de esa tesis

Finalmente es mi deseo agradecer a nuestros tutores: el Doctor Ingeniero de Sistemas e Informática Walter Fuertes Díaz por darnos la oportunidad de trabajar juntos, por su valiosa guía y amistad; al Ingeniero en Sistemas e Informática Rodrigo Fonseca por su apoyo valioso en este proceso.

Leo

INDICE DE CONTENIDOS

LISTADO DE TABLAS

LISTADO DE FIGURAS

RESUMEN

Los métodos de simulación y las plataformas de virtualización constituyen dos tecnologías prominentes en el ámbito de la investigación, que son utilizadas para medir el rendimiento de las redes IP. En esta investigación se han diseñado, implementado y puesto en funcionamiento escenarios de red simulados (mediante NS-2) y virtualizados (mediante Xen) con el fin de validar el rendimiento de redes IP. Consecuentemente, se ha realizado varios experimentos utilizando estas dos tecnologías con el propósito de verificar como se va degradando el rendimiento de la red a medida que se incrementan equipos en la misma. Los resultados experimentales iniciales ilustraron que existen diferencias al evaluar el rendimiento de la red a pesar de someter las dos tecnologías con los mismos escenarios y a las mismas pruebas. Esto se debe en el caso de la Simulación a que los parámetros deben ser rigurosamente programados para mejorarlos. En cambio, en el caso de la Virtualización, se deben a la falta de adaptación de otras condiciones operacionales como temporización, mejoramiento de las plataformas de hardware y de virtualización Xen. En este contexto, se realizaron pruebas adicionales incluyendo diversos algoritmos de generación de tráfico, evaluación del ancho de banda (BW) y estimación de paquetes perdidos. Esto permitió finalmente determinar la degradación del rendimiento de la red al evaluar los resultados obtenidos mediante Simulación con NS-2 y Virtualización con Xen.

CAPITULO 1 INTRODUCCION

1.1- Planteamiento del Problema

1.1.1- Conceptualización del Problema

La Escuela Politécnica del Ejército (ESPE), a través del Vicerrectorado de Investigación y Vinculación con la Colectividad, con el objetivo de impulsar y fomentar nuevos talentos profesionales en el desarrollo de la investigación científica y tecnológica, ha convocado a presentar Proyectos de Investigación que se desarrollen como tesis de graduación, para formar parte del portafolio de proyectos del programa de INICIACIÓN CIENTÍFICA de la ESPE-2009.

Del mismo modo, en el Departamento de Ciencias de la Computación de la ESPE existe la línea de investigación de Tecnologías de la Información y Comunicación (TICs) y se tiene la perspectiva de incursionar en la investigación de novedosos campos temáticos. En este contexto, se presenta el plan de tesis, con título: "**EVALUACIÓN DEL RENDIMIENTO DE REDES IP, UTILIZANDO PLATAFORMAS DE VIRTUALIZACIÓN Y METODOS DE SIMULACION.**", como requisito previo a la obtención del título de Ingenieros de Sistemas e Informática.

La investigación y la aplicación de tecnologías computacionales modernas deben ser usadas para la solución a problemas reales. Una de estas tecnologías son las plataformas de virtualización.

Estas plataformas permiten crear escenarios de redes virtuales que emulen equipos interconectados entre sí, utilizando los recursos de un solo computador [1], en donde se pueden levantar y ejecutar aplicaciones de redes, a fin de ser probados o medidos antes de su despliegue, con lo cual se podrían reducir el

riesgo de daño en las redes, el costo de inversión así como el costo de la experimentación [2]. Estas plataformas pueden ser usadas para la validación de software, emulación de prestación de servicios en redes y una variedad de aplicaciones [3].

El ámbito en el cual se desenvolverá este proyecto de iniciación en investigación científica es el dimensionado y evaluación de métricas de rendimiento de redes de datos, utilizando plataformas de virtualización de libre distribución y contrastando estos resultados, con los obtenidos al utilizar métodos de simulación. En concreto, esta investigación pretende contribuir con un estudio de los factores que afectan la penalización del rendimiento producida por la capa de virtualización durante el dimensionado de redes IP, con el fin de mejorar la precisión de resultados de dicha experimentación.

1.1.2- Formulación del Problema

En la actualidad una red de datos empresarial no solo es aquella infraestructura que comunica computadores para compartir datos y recursos, sino, que se ha convertido en el núcleo principal de la prestación de servicios que una empresa puede brindar. Las redes de datos en el Ecuador y en el mundo tienen gran importancia, pues facilitan la comunicación entre las personas, optimizan la distribución de la información, generan la comercialización de servicios y posibilitan compartir los recursos en forma local y remota. Así mismo, se ha incrementado de manera exponencial las aplicaciones informáticas en el Internet. Todo esto implica enfrentar nuevos desafíos de seguridad, disponibilidad y rendimiento óptimo de la red, evitando congestiones o la degradación de dichos servicios.

Para enfrentar estos desafíos, se requieren pruebas de rendimiento previas a la Implantación del servicio en la red. El desconocer con precisión el comportamiento del rendimiento en una red de datos, constituye un generador de problemas para los administradores de red. Por lo tanto, es conveniente investigar e innovar mecanismos para dicho dimensionado. Una alternativa es el dimensionado de redes mediante métodos de simulación, que imitan el funcionamiento de la red, pero que no son capaces de reproducir el funcionamiento total del hardware. Cómo otra alternativa se plantea el uso de plataformas de virtualización. No obstante, en investigaciones relacionadas [4], se ha demostrado que la utilización de plataformas de virtualización en el dimensionado de redes, genera cierta penalización, lo que ocasiona que los resultados obtenidos no tengan la precisión deseada [5].

Para resolver los problemas enunciados, la presente investigación propone evaluar los factores que incrementan la penalización e inciden en el rendimiento cuando se utilizan plataformas de virtualización, con el fin de encontrar soluciones para mejorar la precisión de los resultados obtenidos en dichos entornos.
(Ver Figura 1.1).

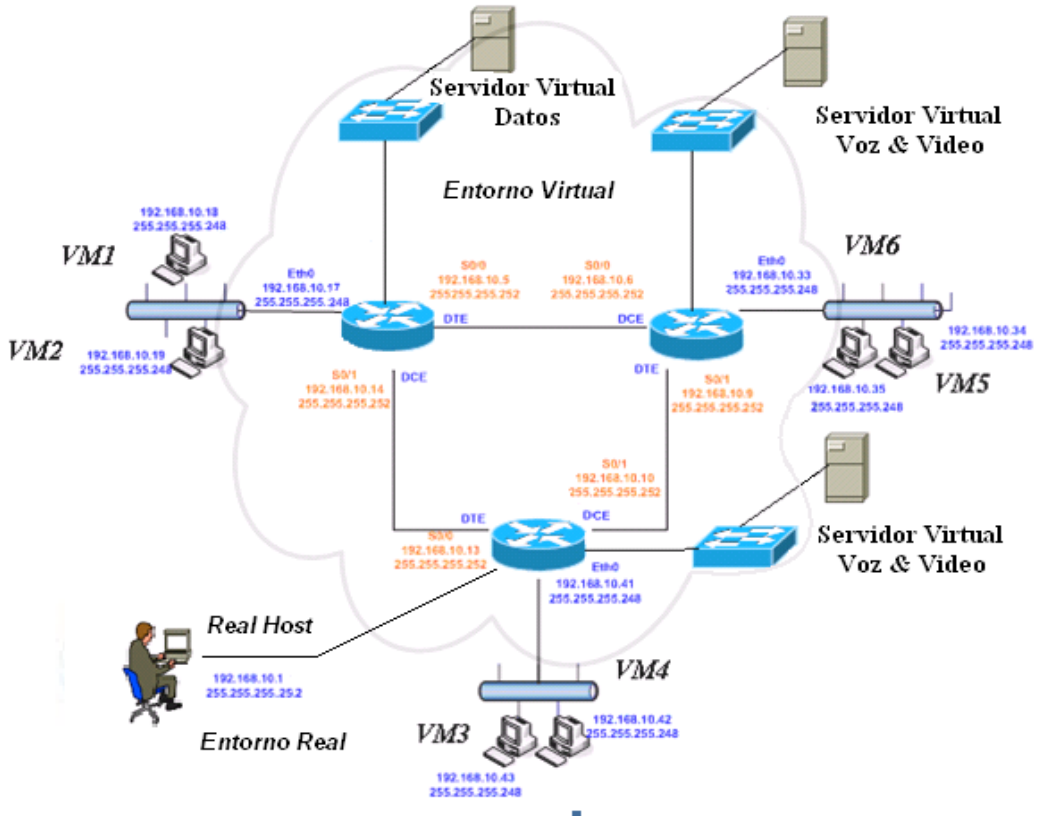

Figura 1.1: Topología arbitraria para el dimensionado de redes, mediante entornos virtualizados

1.1.3- Delimitación Espacial

Este proyecto será realizado en su fase inicial, con equipos propios, utilizando software de libre distribución bajo plataformas Linux. Para pruebas posteriores y validaciones finales se utilizarán los laboratorios especializados del Departamento de Ciencias de la Computación de la ESPE.

1.1.4- Delimitación Temporal

El presente proyecto se enmarca dentro de los lineamientos de la investigación científica y en base a la revisión del estado del arte contemporáneo en el tema de dimensionado de redes utilizando entornos virtualizados [6][7].

1.2- Objetivos

1.2.1- Objetivo General

Realizar una comparación cuantitativa de los resultados obtenidos del dimensionado de servicios de redes IP mediante técnicas de virtualización, versus las obtenidas mediante técnicas de simulación, con el fin de determinar los factores que afectan el rendimiento y optimizar la precisión de las medidas en entornos virtuales

1.2.2- Objetivos Específicos

- Investigar, diseñar e implementar escenarios virtuales de red mediante tecnologías de virtualización de código abierto o de libre distribución.
- Investigar, diseñar y poner en funcionamiento escenarios de red mediante herramientas de simulación de código abierto o de libre distribución
- Analizar y aplicar métodos estadísticos que permitan realizar la comparación de resultados obtenidos mediante entornos virtualizados versus entornos simulados.
- Investigar y analizar los factores que afectan el rendimiento de los entornos virtualizados, con el fin de proponer soluciones para mejorar la precisión del dimensionado de redes IP en estos entornos

1.3- Justificación

Esta propuesta se basa en herramientas de virtualización de libre distribución, que permiten crear entornos de redes virtuales en una determinada topología y realizar el dimensionado de redes. Estos entornos virtuales facilitan la experimentación de redes haciendo que se visualicen como un ambiente real,

cara a hacer pruebas y medir el rendimiento antes de su despliegue, beneficiando a la empresa en el ámbito económico y disminución de riesgos.

Sus principales contribuciones radican tanto en incursionar en las tecnologías de virtualización para experimentación de redes IP, como en la búsqueda de soluciones al problema del overhead (penalización) que se provoca al utilizar estos entornos y que genera resultados sin la precisión deseada.

1.4- Alcance

El desarrollo de este tema de tesis, tiene como alcance realizar experimentos utilizando herramientas de virtualización y simulación de redes, para evaluar y mejorar la precisión de los resultados obtenidos en entonos virtualizados. Se enmarca también dentro de la convocatoria realizada por el Vicerrectorado de Investigación, para presentar proyectos de Investigación de iniciación científica que sirvan como temas de graduación a nivel de pre-grado. Así mismo está conceptualizado dentro del campo de Redes y Sistemas Distribuidos, específicamente en el tema de Virtualización de entornos de red.

1.5- Hipótesis de Trabajo

La evaluación del rendimiento de redes IP utilizando plataformas de virtualización y métodos de simulación con el fin de determinar y ajustar factores que afectan el rendimiento, permitirá aumentar la confianza de la información que proyecten los experimentos de entornos virtuales.

1.6- Metodología

A continuación se describen los pasos a seguir:

- Estudiar el estado del arte de trabajos de investigación relacionados con el mejoramiento de la precisión de resultados experimentales en redes IP utilizando entornos virtuales;
- Investigar las tecnologías de virtualización y en concreto la construcción de escenarios virtuales de red con **Xen** [8][9]. Así mismo, estudiar los factores que producen el overhead (penalización) y que afectan el rendimiento en los entornos virtuales;
- Investigar los fundamentos y la utilidad de las técnicas de simulación y en concreto el simulador **NS-2** [10], en el análisis de eventos en redes de datos;
- Investigar y aplicar métodos de inyección de tráfico **UDP** o **TCP**, tanto en el entorno virtualizado, como en el simulado (**Iperf** [11] o **Netperf** [12]);
- Investigar y aplicar herramientas de limitación y ajuste de ancho de banda, retardo, pérdida de paquetes, entre otros como es el caso del **NetEm** [13];
- Diseñar e implementar un escenario de red en concreto**.** Esta topología debe ser creada y desplegada tanto con la herramienta de virtualización como con la de simulación sobre Linux, a fin de realizar el dimensionado de la red y obtener una comparación cualitativa;
- Tomar medidas en los dos escenarios de experimentación, de diversos parámetros de las redes tales como: ancho de banda consumido, tiempo entre llegada de paquetes y retardo;

- Investigar y aplicar métodos estadísticos que permitan generar estadísticas y gráficas relativas a las mediciones realizadas, calcular la similitud entre los resultados obtenidos entre los dos grupos de datos y realizar ajustes a las configuraciones;
- Evaluar y discutir los resultados experimentales;
- Escribir la memoria.

CAPITULO 2 MARCO TEORICO

2.1- Revisión del Estado del Arte

Virtual machine o máquina virtual es un término que se enunció por primera vez en 1972, cuando IBM publica su Virtual Machine Facility/370 (VM/370) que se compone de 3 diferentes sistemas operativos: Control Program (CP), Conversational Monitor System (CMS), Remote Spooling and Commnications Subsystem (RSCS), juntos estos 3 sistemas producen facilidad de acceso múltiple de propósito general [14], la idea es que un Sistema Operativo en este caso el CP, simule al propio hardware un número de veces determinado, es decir se creen copias exactas del hardware base pero de forma lógica con la posibilidad de ejecutar sobre tales copias diferentes sistemas operativos.

Durante esta investigación se encontró un único trabajo muy relacionado al presente, es el propuesto por Muñoz en [15], quién realiza una comparación entre los resultados obtenidos entre VNUML como herramienta de Virtualización y el simulador OPNET, para analizar la QoS en los servidores Web, cuyos resultados muestran una importante aproximación al medir el rendimiento de un servidor utilizando las dos tecnologías.

Tradicionalmente la comunidad científica viene utilizando estas dos tecnologías pero por separado, sea en un entorno simulado o en un entorno virtualizado. En este contexto, en lo que concierne a la aplicación de NS-2 en dimensionado de redes, el trabajo propuesto por [16], describe un entorno real modelado y simulado con NS-2 para el análisis de QoS en la provisión de recursos en redes de VoIP basado en retardo (on end-to-end delay) y características de pérdida de paquetes (packet loss) utilizando un planificador FIFO. En relación con otros

métodos de simulación, el trabajo propuesto por [17] presenta un estudio comparativo de tres simuladores híbridos de red: OPNET, NS-2 y NCTUns, utilizando una topología experimental y modelada en los tres simuladores. La comparación consistió en desplegar tráfico de varios escenarios combinando patrones de tráfico CBR y una sesión de FTP, con el fin de comparar la precisión de los simuladores y además comprobar la respuesta de la red a nivel de concentrador y enrutador respectivamente a patrones de tráfico conocidos. En [18] son analizadas algunas características de NS-2 en el funcionamiento de los métodos existentes para la estimación de BW disponible. Así mismo en este trabajo se analiza el comportamiento y manejo de parámetros del agente generador de tráfico CBR como una mejor alternativa para la estimación del mismo. Todos estos esfuerzos son una pequeña muestra de la utilidad del NS-2 en el dimensionado de redes, que además han servido como fundamento en el desarrollo de nuestra investigación.

En lo referente al uso de tecnologías de virtualización para medir el rendimiento de las redes, el trabajo propuesto en [3] muestra el desempeño sistemático del rendimiento de la red virtualizada mediante la comparación de varias tecnologías de virtualización Xen, VMWare, OpenVz frente a evaluaciones tipo benchmark para determinar la herramienta que proporciona el mejor rendimiento general. En un enfoque muy parecido, en el trabajo de fin de master propuesto por [19] se evalúa la conveniencia del dimensionado de networking utilizando máquinas virtuales. Aquí se realizó una evaluación de Xen2, Xen3, Vmware, UML y Qemu, utilizando para dichos escenarios la integración de la herramienta NetShaper.

Previo a ello, se definen varios criterios de comparación basados en el tráfico y el throughput. Comparado con el nuestro, ellos utilizaron la virtualización pero no validaron sus resultados mediante métodos de simulación.

Finalmente, en un contexto más cercano al de ésta investigación en [20] se utilizó Xen como plataforma de virtualización para probar la funcionalidad del VoD (Video Bajo demanda), intentando emular el servicio de VoD de ADSL en un entorno virtual con el fin de comparar la precisión de los resultados obtenidos entro lo real y lo virtual. Para tales fines se capturó tráfico de video en una solución ADSL real. Luego, con el análisis del tráfico capturado, se consiguió emular VoD tipo ADSL, utilizando el entorno de virtualización. Posteriormente, el BW, el retardo y el tiempo entre llegada de paquetes fue medido tanto en el entorno real como en el virtualizado. Estos parámetros fueron ajustados para obtener un comportamiento similar entre clientes y servidores en ambos casos.

En contraste con esta investigación, se analiza y se compara los resultados obtenidos en un entorno virtualizado vs el simulado puesto que las dos tecnologías son soluciones mucho mas económicas y adecuadas para la realización de estos experimentos.

2.2- Conceptos básicos, técnicas de virtualización, plataformas de virtualización: UML, VMware, Xen, KVM, VirtualBox, OpenVZ

2.2.1- Definición de virtualización

La virtualización es el particionamiento lógico de un equipo físico en múltiples VM's (máquinas virtuales), que comparten recursos de hardware, como CPU, memoria y dispositivos de entrada y salida [21]. Permite a las VM ejecutarse al

mismo tiempo y que trabajen de manera completamente funcional, como si fueran máquinas reales.

A continuación se enumeran las ventajas de virtualizar:

- Ahorro en: infraestructura, costos de inversión de hardware, tiempo, software, mantenimiento físico y lógico;
- Fácil portabilidad, y migración de las VM en caliente;
- Se incrementa la seguridad, utilizando servidores aislados y dedicados para tareas personalizadas;
- Administración centralizada de las VM's, así como sus recursos de almacenamiento y de red;
- Creación de VM de forma práctica utilizando técnicas de clonación;
- Aprovechar de forma eficiente las capacidades de proceso de los servicios.

Entre las desventajas encontradas en este estudio se encontraron:

- Falta de normalización que evite conflictos de hardware y software;
- Por defecto solo se pueden crear hasta 5 o 7 VM's dependiendo las capacidades del equipo físico, pero la investigación se vio en la necesidad de crear más VM, para ello se logro habilitar más interfaces de red, mediante *loop backs*, que se encargan de crear espacios para las VM;
- La virtualización introduce un nivel de proceso adicional, que traduce las llamadas de la VM al SO anfitrión, produciendo un overhead debido al consumo de recursos [21].

2.2.2- Definición de máquina virtual

Una máquina virtual es una instancia eficiente de un SO, desplegada sobre una plataforma de virtualización, emulando los recursos de una máquina real [14]

como: interfaces de red, disco duro, memoria RAM y dispositivos de red de Entrada y Salida, de manera que su comportamiento sea igual al de la máquina física. Se pueden crear y ejecutar varias VM's independientes unas de otras, las mismas pueden trabajar sin interferencias, y bajo iguales o diferentes cargas de trabajo.

En 1972 IBM fue quien comenzó a utilizar el concepto de máquina virtual, particionando sus ordenadores en máquinas independientes que trabajen bajo un mismo equipo físico, realizando tareas y aplicaciones a igual tiempo. Idea que apareció para aprovechar al máximo la inversión en mainframes que en esa época eran recursos caros [22] .

Las principales características de las VM's son [23]:

- Compatibilidad: las VM's son totalmente compatibles con los sistemas operativos X86, sus aplicaciones y controladores de dispositivos, tal que dentro de la VM se puede ejecutar cualquier software y este comportarse como si estuviera en la máquina host;
- Aislamiento: el comportamiento de las VM's es totalmente aislado de la máquina host, sin embargo pueden compartir los mismos recursos físicos de un ordenador, esta característica hace que la disponibilidad y el desempeño de las aplicaciones de cada VM sean independientes, haciendo que cada una trabaje de manera aislada, protegiendo su información;
- Encapsulación: la encapsulación, es el poseer la capacidad de almacenar todos los recursos de una máquina real. Por ello las VM's se consideran aptas para su portabilidad y además su fácil gestionamiento de datos;

- Independencia del hardware: La VM puede crear cuantos dispositivos virtuales considere necesarios, independientes de cuantos disponga el equipo físico.

Algunas Recomendaciones:

- Las VM's deben ser independientes unas de otras, para no afectar el desempeño individual;
- Dar soporte a cada VM, cada cierto tiempo;
- No sobrecargar al máximo las VM's;
- Es recomendable utilizar equipos que soporten virtualización asistida por hardware.

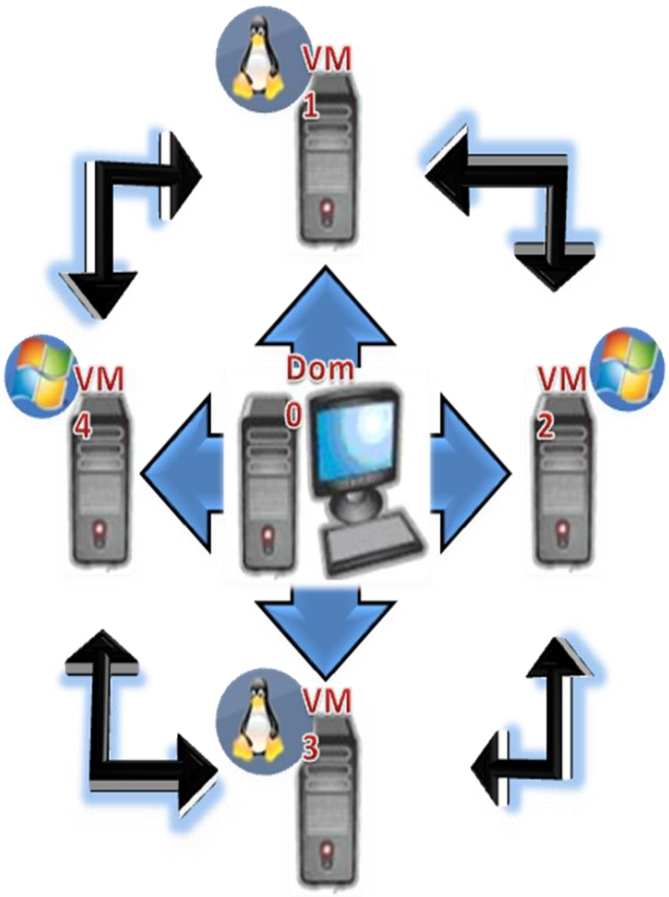

Figura 2.1: Vista simplificada del entorno virtual

2.2.3- Escenario de redes virtuales

Un escenario de red virtual, es el diseño de una topología de red, que incluye conexiones entre VM's y sus diferentes dispositivos de red como: bridges, hubs, routers, dentro de una red LAN o WAN, que luego será implementado bajo una herramienta virtual o simulada, con el objetivo de plasmar un escenario virtual/simulado, que será percibido como uno real.

2.2.4- Definición de Simulación de redes

Es un entorno de experimentación de prueba, repetible en cuantas veces se desee, en la que se diseñan y recrean escenarios de redes, mediante la programación de eventos que permiten configurar una red como si se tratase de una red real.

Simular entonces es recrear un modelo de red, utilizando cálculos matemáticos y lógicos que describen el comportamiento, estructura de la red y sus componentes en un período de tiempo a desplegarse [24]. Según R.E. Shannon [25] es: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema".

2.2.5- Definición de emulación de redes

Es una recreación virtual de algún escenario de red, en el que se trata de desplegar los elementos que integran una red real, tal como: los dispositivos de

red, los diferentes protocolos, agentes de tráfico, todo ello trabajado en tiempo real. Por tanto cada resultado de emulación de red será diferente, lo cual es contrario a la simulación, puesto que ésta, hace cálculos con resultados iguales. Algunas de las aplicaciones de emulación se enumeran a continuación:

- Aplicaciones de trabajo como: emuladores de software, de SO's, de hardware;
- Para ambientes de pruebas: emuladores de escenarios de red, copias y respaldos de servidores, validación de software;
- Aplicaciones para docencia educacional, para investigaciones referentes a redes IP;
- Mantenimiento.

2.2.6- Técnicas de Virtualización

a. Virtualización de hardware

Cuando se crea una aplicación reflejo del hardware completo del equipo físico anfitrión, convirtiéndose en la técnica más compleja. Reutiliza el manejo de procesos; manejo de memoria / memoria virtual, e incluso emula las instrucciones a nivel del CPU. Poniendo la ventaja de evitar la necesidad de poner parches en el SO o de hacer modificaciones que resultan molestosas. El inconveniente debido a esta característica es que la emulación se vuelve lenta. El SO por tanto no se ejecuta sobre el hardware real sino sobre el virtualizado.

La arquitectura x86, incorpora la virtualización asistida por hardware, la cual tiene una extensión en el procesador x86, para facilitar las tareas de

virtualización, con el paso del tiempo esta arquitectura llega a trabajar en dos enfoques, virtualización completa, y para-virtualización.

Ejemplo:

- Mame
- Qemu

b. Virtualización a nivel del Sistema Operativo

La VM es el invitado que comparte el mismo kernel del sistema anfitrión, tiene los mismos recursos lógicos, pero no se virtualiza el hardware. En esta técnica lo que se tiene es uno o varios SO's dentro del equipo anfitrión. La ventaja de esta técnica es que mejora el rendimiento y la eficiencia, pues garantiza el aislamiento y seguridad de recursos en cada VM.

Ejemplos:

- OpenVZ
- Virtuozzo
- UML
- Colinux
- VMWare

c. Paravirtualización

También conocida como Virtualización asistida por el SO, es una técnica que crea instancias de SO's en las VM's, para ello el SO del equipo anfitrión necesariamente tiene que ser modificado para poder trabajar con el sistema huésped, ya que con ello se reemplazan instrucciones x86 no virtualizables.

Debido a ello se crea una capa intermedia llamada Hypervisor entre el hardware del recurso físico y el SO de la VM, éste es el monitor que enviará las llamadas que permitirán manejar los recursos del sistema anfitrión desde las VM's, alcanzando un alto rendimiento.

Esta técnica fue creada para utilizarla como alternativa a la arquitectura x86, es decir cuando no se tiene un equipo con virtualización de hardware [26].

Ejemplos:

- Xen
- UML

d. Virtualización completa (full virtualization)

Permite ejecutar sobre el sistema anfitrión, uno o varios SO's en las VM's huésped, sin sufrir modificaciones, las VM's utilizando el hypervisor, quien es el intermediario que ejecuta las instrucciones entre las capas.

Ejemplos:

- VirtualBox
- VirtualPC
- QEMU
- KVM
- VMware Workstation
- VMware Server
- OpenVZ
- XenServer

En esta técnica se encuentran las siguientes desventajas:

- Costos significativos en complejidad
- Rendimiento en tiempo de ejecución.

2.2.7- Plataformas de virtualización

a. UML

UML por sus siglas en inglés (User Mode Linux), Modo de Usuario de Linux. Es una plataforma de virtualización modificada del núcleo de Linux para que funcione sobre su propia interfaz de llamadas al sistema, permitiendo tener varios SO's de diferentes distribuciones de Linux. Utilizada especialmente para la creación de laboratorios virtuales de investigación experimental.

Algunas de las características de UML [27]:

- Permite la creación de honeypots [28], que son herramientas que prueban la seguridad de las máquinas sin comprometerlas.
- Seguridad de servicios de red.
- Permite ejecutar entornos de pruebas, en donde el software es inestable o sea incompatible con el núcleo del sistema que aloja el UML.
- Desarrollo de nuevos kernel.

b. VMWare

VMWare es una plataforma de virtualización que emula un ambiente con características de hardware determinadas, logrando parecerse a un sistema físico con todos sus recursos. VMWare es similar a VirtualPC, sin embargo difieren debido a que VMWare ejecuta las instrucciones directamente sobre el hardware

físico, mientras que VirtualPC traduce las instrucciones en llamadas al SO para luego ser ejecutadas por el sistema físico [29].

VMWare se lo puede utilizar tanto para sistemas Windows como Linux. Existen varios productos de VMWare como:

- VMWare Player: versión gratuita que permite ejecutar VM creadas mediante otros productos de VMWare, pero que por el mismo no puede;
- VMWare Server: versión actualmente gratuita, posee mejor administración de recursos, con funcionalidad de administración remota, capaz de abarcar múltiples VM en una arquitectura Servidor;
- VMWare Workstation: versión comercial, disponible para Windows y Linux, permite la creación de múltiples VM dentro de un Workstation;
- VMWare ESX Server: versión comercial, complejo de virtualización y de alto rendimiento, pensado para la centralización y virtualización de servidores.

c. Xen

Es un entorno de virtualización de código abierto desarrollado por la Universidad de Cambridge en el año 2003. Se distribuye bajo licencia GPL de GNU. Permite ejecutar múltiples instancias de SO's con todas sus características, pero carece de entorno gráfico. El núcleo de Xen, que administra las VM's, se conoce como hypervisor, que en Xen es el dominio principal (Dom0).

Las VM's son designadas como dominios de nombre genérico DomU, las cuales se ejecutan con diversas cargas de trabajo de forma completamente funcional, proporcionando independencia del sistema anfitrión; administración propia de los recursos, la seguridad de la calidad de los servicios; además tiene la característica de poder migrar máquinas virtuales en caliente y algo muy atractivo es que funciona en máquinas de no tan alto rendimiento [9].

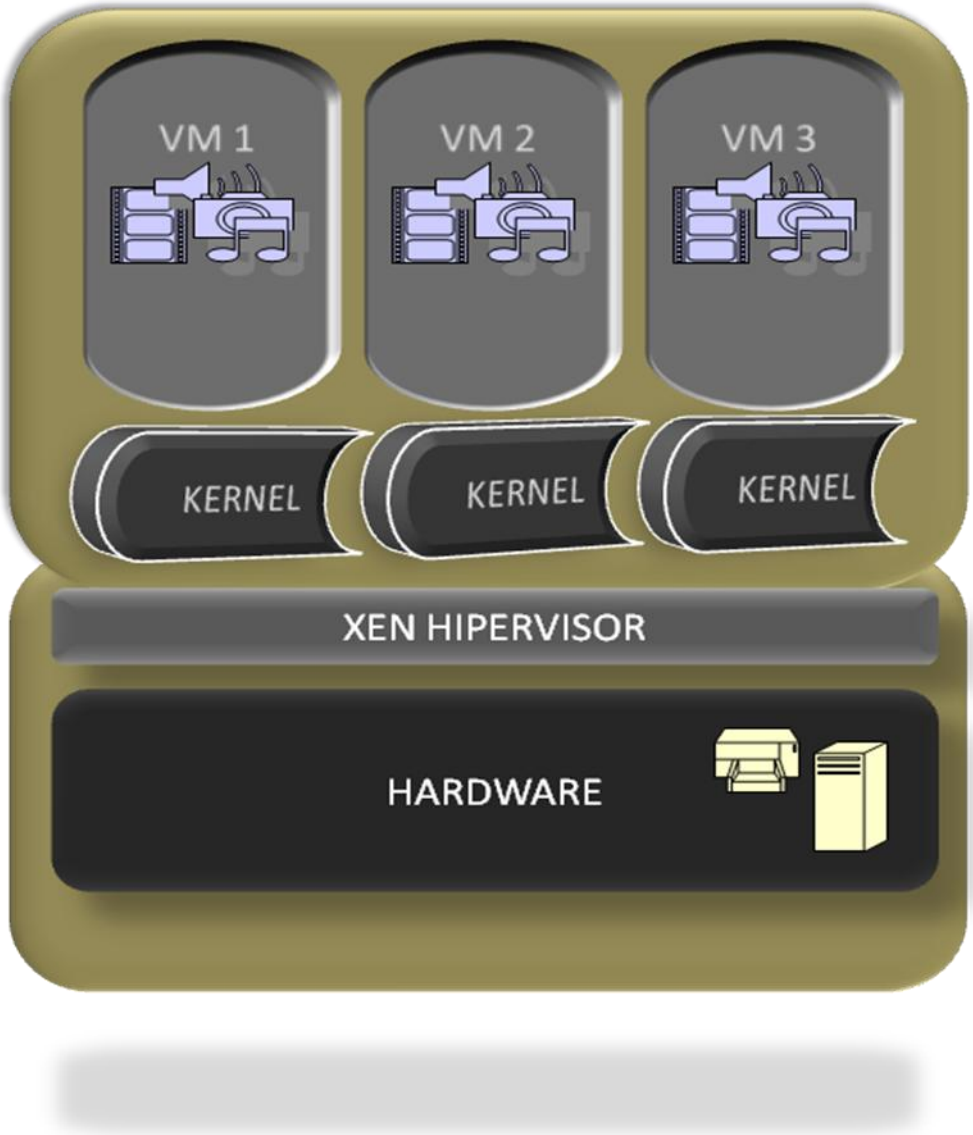

Figura 2.2: Arquitectura de Xen

d. KVM (Kernel-Based Virtual Machine)

En KVM el núcleo de Linux es el hypervisor, que tiene el control de los dispositivos reales, de la planificación de tareas, la gestión de memoria. KVM está dentro de la virtualización completa de libre distribución en Linux sobre la arquitectura hardware x86.

Todo proceso en Linux tiene dos modos de ejecución que son: el Kernel y el modo usuario. KVM incorpora a estos dos modos un tercero, que lo llama: modo invitado, el mismo que tiene los dos modos antes mencionados [21].

Para poder ser ejecutado necesita de una versión modificada de QEMU. Plataforma virtual que ejecuta VM's con imágenes de disco de SO's que no han sido modificados. Haciendo que las VM's tengan un hardware individual virtualizado, como: discos duros, tarjeta de red, tarjeta gráfica, memoria, entre otros.

e. Virtual Box

Desarrollado para ser utilizado sobre plataformas x86, inicialmente construido como software privado, pero que en la actualidad es gratuita o la versión de Virtual Box OSE, sujeta a la licencia GPL.

Ejecuta varios SO's sin modificación incluyendo a: GNU/Linux, Mac OS X, Windows, y Solaris. Utilizado para: fortalecimiento de servidores, con balanceo de carga, ahorro de energía y sobre todo para el ahorro de hardware.

Como principal ventaja Virtual Box tiene una interfaz de usuario amigable, además es versátil y en el que se puede utilizar varias versiones de Linux, además emplea imágenes ISO como unidades virtuales de CD, DVD o floppy [30].

Una de las desventajas de Virtual Box cuando se lo compara con otras plataformas de virtualización como VMware Workstation, Virtual PC, éste no posee algunas funcionalidades de los antes mencionados, sin embargo se destaca por su herramienta RDP (Remote Desktop Protocol), que permite manejar remotamente las VM's.

f. OpenVz

Plataforma de virtualización a nivel de SO, ejecuta múltiples SO independientes, basado en Virtuozzo, que una herramienta de código abierto, comercial. Su núcleo es modificado y proporciona aislamiento, administración de recursos, cada VM se comporta de manera tal, que no afecta el rendimiento o disponibilidad de las demás.

Dentro de las características especiales están [31]:

- Escalabilidad: modelo de kernel único, con capacidad de escalabilidad de Linux 2.6, haciendo posible la creación de hasta 64 VMs, y hasta 64 Gb de RAM;
- Densidad: por la capacidad de alojar cientos de entornos virtuales que dan un tiempo de respuesta aceptable;
- Administración masiva: se puede realizar la administración a todas las VM, con tan solo realizar un script;

- Escenarios de Uso: se lo utiliza en seguridad, consolidación de servidores, hosting, desarrollo y pruebas, y en el entorno educativo.

2.3- Métodos de simulación de redes. El NS-2

Network Simulator es un simulador de código abierto diseñado específicamente para la investigación de redes de computadoras [10]. Desarrollado por el grupo de trabajo Virtual InterNetwork Testbed (VINT) fundado por la *Defense Advanced Research Projects Agency* (DARPA) [32].

Esta herramienta permite la simulación de:

- Protocolos de enrutamiento, multicast e IP, tales como UDP, TCP y RTP sobre redes normales e inalámbricas (locales y satélite) [10];
- Generadores de tráfico (FTP, Telnet, WEB, CBR, VBR);
- Utiliza mecanismos de gestión de colas (droptail, RED, etc.), y algoritmos de enrutamiento;
- Algoritmos de ruteo;
- Comunicación Full Duplex.

Orientado a eventos discretos. Un evento discreto es el conjunto de relaciones lógicas, matemáticas y probabilísticas que integran un modelo computacional que evoluciona en el tiempo mediante cambios instantáneos en las variables de estado.

NS-2 trabaja a nivel de paquetes y ha sido ampliamente utilizado en el ambiente académico y de investigación. Además NS-2 se encuentra implementada en C++ con el objetivo de reducir los tiempos de procesamiento, utilizando un intérprete OTcl con soporte de programación orientada a objetos, así

NS-2 separa la lógica de la trayectoria de datos de las estructuras de control y ejecución de la simulación.

Ofrece la posibilidad de trabajar sobre distintas configuraciones de red, y hacer cuantos cambios se requieran, logrando de esta manera ser una herramienta ideal para probar diferentes escenarios de red, hacer pruebas y comparar sus resultados, reduciendo tiempos y gastos innecesarios, como si se hiciera en redes reales, constituyendo un beneficio para los administradores de red.

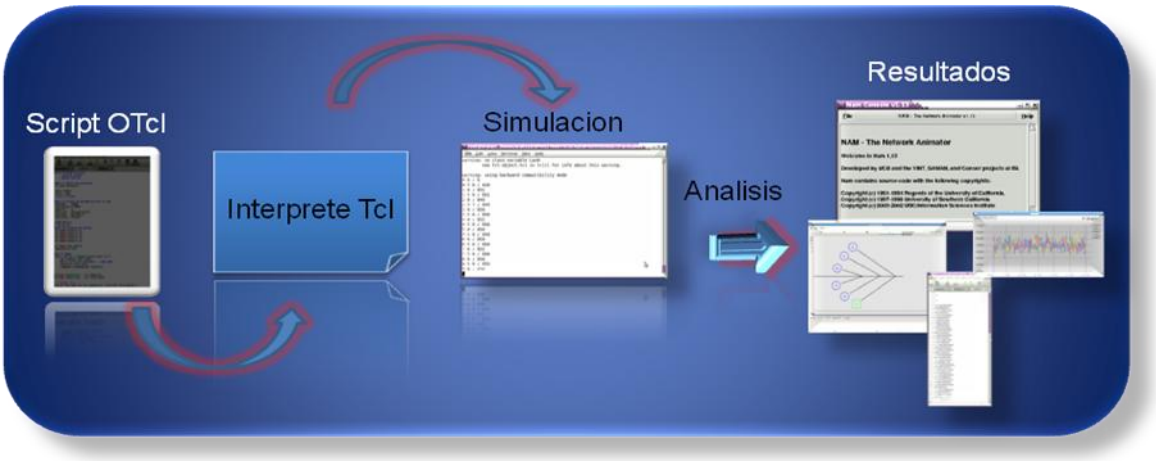

Figura 2.3: Funcionamiento de NS-2

Como se muestra en la Figura 2.3, el único Input para comenzar el análisis en NS-2 es un script en OTcl desarrollado por el usuario, NS-2 se encarga posteriormente de procesar la información ingresada. Los resultados que arroja NS-2 se pueden dar en 2 formas, mediante archivos .tr, o mediante una interfaz gráfica llamada NAM, la cual es mucho más didáctica.

2.4- Mecanismos de inyección o generación aleatoria de paquetes UDP.

2.4.1- Protocolo UDP (Protocolo de Datagrama de usuario)

Protocolo que trabaja a nivel de transporte que, a diferencia de TCP**,** no es orientado a conexión y no es fiable. Se caracteriza por su sencillez, rapidez, y es muy utilizado en el campo de la multimedia, además de aplicaciones en las que se permita la pérdida de datagramas**,** además UDP trabaja como un multiplexor/demultiplexor cuando envía y recibe los datagramas, dichos datagramas se envían con un único datagrama IP y viajan a través de la red utilizando puertos.

2.4.2- Mecanismos de inyección en virtualización

a. Iperf

Funciona para IPV4 o IPV6, tanto en distribuciones de Linux, cuanto en Microsoft Windows y Mac, es una herramienta cliente-servidor para virtualización, de código abierto, que permite:

- Inyectar aleatoriamente paquetes o datagramas UDP y TCP;
- Soporta además multicast y conexiones múltiples simultáneas.

Su utilización en fácil, rápida y dinámica, a través de la línea de comandos. Tanto el cliente como el servidor tienen parámetros propios de configuración en la red:

- Cliente: iperf - c <IP> - u - p <Puerto> - f <Formato> - t <Tiempo> - b <Bandwidth>
- Servidor: iperf - s - u - p <Puerto> - f <Formato> - t <Tiempo> - b <Bandwidth>

Significado de parámetros básicos:

- c: dirección IP del cliente;
- s: servidor ;
- u: se va a utilizar el protocolo UDP.

b. Mgen

Multi-generator [33], de código abierto, fue desarrollado por el Laboratorio de Investigación Naval (NRL). Herramienta que proporciona la capacidad de hacer pruebas del rendimiento de redes IP, especializado en redes de protocolo UDP, actualmente se encuentra en desarrollo para implementar TCP.

Implementa características como:

- Generación de patrones de tráfico en tiempo real, brindando estadísticas del trafico generado;
- Utilizado para proveer estadísticas sobre el rendimiento, tasa de pérdida de paquetes, demora en la red;
- Actualmente se lo encuentra para Unix, plataformas Win32, y MacOS en versiones Mgen 4.x, y Mgen 3.x;
- Cuenta con interfaz gráfica.

c. Netperf

Netperf [34], utilizado para medir varios aspectos del rendimiento de redes IP, en los que incluye:

- La transferencia de datos y solicitud inyección de tráfico UDP, TCP
- Observar la latencia y el throughput generado.

d. Netcat

Netcat (nc), utilidad que ayuda a analizar los datos a través de las conexiones de red, ya sean entrantes o salientes, con TCP o UDP. Con un diseño seguro, que puede ser usado dentro de scripts, o algún otro programa. Su distribución es gratuita bajo la licencia GPL.

Principalmente abarca el campo de las seguridades en redes:

- Depuración, análisis y manipulación de redes;
- Servicios TCP, UDP;
- Utilizada como medio de transferencia de archivos, o inyección de paquetes;
- Tiene la aplicabilidad de chat entre cliente-servidor;

Sintaxis:

- nc [- options] hostname port [s] [ports]
- nc - l –p port [- options] hostname port [s] [ports]

Parámetros:

- l : modo “listen”, que está a la espera de una conexión entrante;
- p: es el puerto local;
- u : se ejecuta en modo UDP;
- e : ejecuta la línea de comando después de conectarse;
- c : ejecuta órdenes de Shell después de conectarse.

2.4.3- Mecanismo de inyección en Simulación

Para la generación de paquetes UDP, en simulación es necesario implementar la creación de un script que incluye un algoritmo de generación de paquetes UDP en NS-2, cuyas características son:

El agente UDP genera los paquetes de tamaño fijo o variable, pero por defecto el tamaño máximo del paquete está fijado en 1000 bytes.

Para generar los paquetes, se añaden a un agente de tráfico, que puede ser de tipo CBR, Exponencial, Pareto, o de acuerdo a la traza de un archivo.

2.5- Mecanismos de limitación y ajuste de ancho de banda, retardo y pérdida de paquetes para el dimensionado y medición de tráfico y QoS.

2.5.1- Mecanismo de limitación en virtualización y ajuste de ancho de banda

a. Iperf, ajuste de ancho de banda

La herramienta Iperf antes mencionada, entre sus diferentes opciones, provee de mecanismos de ajuste del ancho de banda por medio de datos como:

- Protocolo: UDP o TCP;
- Agente generador de tráfico tipo CBR para UDP;
- Tasa de transferencia máxima;
- Tiempo de respuesta de cada cliente en el servidor;
- Puertos de envió.

Reportando datos que servirán para el análisis del rendimiento de la red como:

- Total de segundos de los paquetes transferidos;
- La transferencia de los paquetes-datagramas en Bytes;

- El ancho de banda total generado;
- El Jitter total, variación del retardo;
- Número de paquetes perdidos, y recibidos.

Logrando ser una herramienta para la medición del ancho de banda, reportando el throughput generado, para el análisis del rendimiento de la red entre el cliente y el servidor.

b. Netem

Es una herramienta emulador de red que permite manejar parámetros como el retardo, número de paquetes perdidos, observar duplicación, y reordenamiento. Trabaja con la herramienta **tc** (traffic control), que incluye algunas disciplinas de colas llamadas qdisc para la gestión del ancho de banda. Cuando se implementan estas disciplinas cambiamos la forma en la que los paquetes se envían a través de la red, su forma de trabajar es aceptar los datos, y luego optar por reordenarlos, retrasarlos o descartarlos. Generalmente se las utiliza para restringir el tráfico de una red.

2.5.2- Mecanismo de limitación en simulación

El Script desarrollado en programación OTcl implementa líneas de código que permiten delimitar el ancho de banda a través de un cálculo matemático, en Mbits/s. El retardo se implementa mediante la utilización de un parámetro programado que simula el funcionamiento del mismo, al igual que para el tamaño de los paquetes enviados.

El agente que permite recoger las estadísticas para el reconocimiento de los factores del rendimiento de la red, es el Agente Receptor Loss Monitor, que obtiene datos sobre: paquetes recibidos, paquetes perdidos, bytes recibidos, bytes perdidos.

Para la medición del tráfico, dentro de la programación del escenario de red en NS-2, se requiere implementar un generador de tráfico ya sea FTP, Telnet, Web, CBR o VBR, y añadirlo a un agente TCP/UDP, según sea el caso. Para ello se necesita enviar los parámetros necesarios para su configuración como: rate, packetsize, random (permite aleatoriamente introducir ruido en la señal), etc.

Para utilizar el protocolo UDP, se trabajó con el agente CBR, el mismo que posee las siguientes características:

- Fácil aprendizaje y sencilla aplicación;
- Modela y genera resultados consistentes de generación de tráfico.

Para el gestionamiento de colas NS-2 crea mecanismos que se generan en los routers, tales como Drop Tail, RED, CBQ, algoritmos de Dijkstra, etc.

2.6- Métodos estadísticos para análisis de dos o más variables, para determinar las diferencias y exactitud de los resultados.

2.6.1- Media Aritmética

La media aritmética ayuda a obtener el promedio de un conjunto finito de números, para verificar de forma estadística que los resultados son aceptables.

$$\overline{X} = \frac{x_1 + x_2 + x_3 .... x_n}{n}$$

Desventaja:

- Debido a que la media trabaja con variables continuas, es decir que adquiere cualquier valor dentro de un intervalo especificado, por ejemplo: el ancho de banda en una red; el retardo, etc. Haciendo posible aplicarla solo en aquellos parámetros que generen variables continuas, pero en el caso de, el número de paquetes perdidos, la media no se podría analizar, puesto que son variables discretas.

2.6.2- CDF (Función de Distribución Acumulada)

Si X es una variable aleatoria, para cualquier número real X0, existe la probabilidad P(X ≤ X0) del evento X ≤ X0 (X toma cualquier valor menor o igual a X0).

La probabilidad P(X ≤ X0) que depende de la elección de X0 es la probabilidad acumulada hasta X0 que es la ***función distribución o distribución acumulada*** y se denota por F(X0). [35]

$$F(X_0) = P(X \leq X_0)$$

Características Interesantes:

- Cada función de distribución acumulada es no decreciente
- Para poder identificar la gráfica de una CDF de una variable discreta, se puede observar que siempre va a ser escalonada, lo contrario a una continua.

2.7- Conclusiones

En este capítulo se presento una recopilación del estado del arte, definiciones preliminares de lo que es virtualización, técnicas con sus diferentes características, ventajas y limitaciones, y las principales plataformas de virtualización a modo de conocimiento y comparativa, además el método de simulación, los diferentes mecanismos de inyección de tráfico, medición de ancho de banda, y métodos estadísticos que proporcionan un análisis cuantitativo de validación de la investigación.

Como se pudo observar, la virtualización ayuda a emplear lo que se llama la “tecnología verde”, ya que permite un ahorro de recursos tanto físicos como lógicos, y en cuanto al dimensionado de redes, dejando ver un panorama de prevención, de una mejor visualización de la entrega de paquetes. Analizando las plataformas mencionadas, se eligió como herramienta de experimentación a Xen, por su alta disponibilidad y rendimiento, puesto que trabaja bajo para-virtualización, debido a esta técnica las VM’s corren directamente sobre el procesador, sin realizar emulación de éste.

La simulación constituye la herramienta que facilita la repetición de experimentos, en cuantas veces se desee para una mejor apreciación del escenario de red, antes de incorporarlo al entorno real, como herramienta escogida NS-2, por su amplia información vía web, y aplicabilidad en el entorno de pruebas de red.

Ambas tecnologías son dedicadas y especiales de acuerdo a las características que disponga el administrador de red.

CAPITULO 3 DISEÑO E IMPLEMENTACION DEL ESCENARIO DE RED UTILIZANDO PLATAFORMAS DE VIRTUALIZACIÓN

3.1- Introducción

Existen varias técnicas de virtualización y de la misma forma le corresponden las plataformas disponibles, el uso de una plataforma específica está establecido por las tareas y rendimiento que se esperen del dimensionado, la plataforma de virtualización Xen usa la técnica de paravirtualización para desplegar las VM's.

Xen inició como un proyecto de investigación de la Universidad de Cambridge y junto a su nacimiento se fundó XenSource Inc., que luego sería adquirida por Citrix Systems en Octubre del 2007, lo que dió paso a que los productos de XenSource Inc. sean renombrados bajo la marca de Citrix.

Por un tiempo Xen fue un producto comercial bajo la marca Citrix, sin embargo en Octubre del 2009 se decidió que las versiones de Xen desarrolladas por Citrix serían de ahí en adelante totalmente de código abierto, se espera que Xen crezca aún más como plataforma de virtualización y se consolide como la mejor.

3.2- Análisis, diseño e implementación del escenario virtual

3.2.1- Instalación de Xen como plataforma Virtual

La plataforma de virtualización XenServer está disponible en varios sitios de la Internet, la mayoría de ellas funcionan o están adheridas en distribuciones de GNU/Linux[9], se puede elegir entre distribuciones de la comunidad, comerciales o empresariales, la diferencia subjetiva entre ellas es el soporte técnico.

El método de instalación de XenServer presentado en adelante muestra los pasos necesarios para situar la plataforma virtual en una distribución GNU/LINUX llamada Ubuntu Server, se hace necesario recordar que la plataforma virtual Xen debe ser instalada sobre un SO tipo servidor.

Todos los comandos que necesiten ejecutarse durante la instalación se deben realizar con privilegios de super usuario (root) o en su defecto se debe usar el prefijo *sudo*, que permite tener temporalmente privilegios avanzados.

a. Actualización del kernel y los repositorios del SO

El primer paso consiste en la actualización del kernel y los repositorios globales de software, mediante los siguientes comandos:

- *apt-get update*
- *apt-get upgrade*
- *apt-get install linux-image-server linux-server*

b. Validación de los requisitos previos

Para continuar con la instalación de XenServer es necesario contar con:

- Una distribución instalada de GNU/Linux tipo server, con GRUB bootloader sobre un equipo con hardware aceptable para virtualización.
- El paquete iproute2.
- Las utilidades brigde de GNU/Linux (bridge-utils).
- El sistema de conexiones en caliente (hotplug) de GNU/Linux.

Algunos de los requisitos previos, como el sistema de conexiones en caliente, ya están incluidos en algunas de las distribuciones de GNU/Linux, no obstante si los paquetes no han sido instalados se debe ejecutar el siguiente comando:

- *apt-get install ubuntu-xen-server build-essential libncurses5-dev gawk mercurial*

c. Descarga, configuración e instalación del Kernel Xen

Para continuar con la instalación de XenServer es necesario contar con un kernel de GNU/Linux modificado, así el hypervisor de Xen tendrá más privilegios sobre el SO, esto se conoce como la técnica de paravirtualización.

Los siguientes comandos crean un directorio, descargan y permiten configurar el kernel Xen modificado para la instalación.

- *mkdir -p ~/build/linux-2.6.27-xen*
- *cd /usr/src/*
- *hg clone http://xenbits.xensource.com/ext/linux-2.6.27-xen.hg*
- *cd linux-2.6.27-xen.hg*
- *make O=~/build/linux-2.6.27-xen/ menuconfig*

El menú de configuración del Kernel se debe parecer a los siguientes parámetros:

- General setup - Choose SLAB allocator (SLUB (Unqueued Allocator)) - (X) SLAB
- Processor type and features - Subarchitecture Type (PC-compatible) - (X) Enable Xen compatible kernel

- Bus options (PCI etc.) - [*] PCI support

  [*] Xen PCI Frontend

  [ ] Xen PCI Frontend Debugging (NEW)

- Networking support - Networking options - <*> 802.1d Ethernet Bridging
- Device Drivers - [*] Network device support - [ ] Ethernet (10000 Mbit)
- Device Drivers - XEN –

  [*] Privileged Guest (domain 0)

  <*>Backend driver support (NEW)

  <*>Block-device backend driver (NEW)

  <*>Block-device tap backend driver (NEW)

  <*>Network-device backend driver (NEW)

  (8) Maximum simultaneous transmit requests (as a power of 2) (NEW)

  [ ] Pipelined transmitter (DANGEROUS) (NEW)

  < > Network-device loopback driver (NEW)

  <*> PCI-device backend driver (NEW)

  PCI Backend Mode (Virtual PCI) - [ ] PCI Backend Debugging (NEW)

  < > TPM-device backend driver (NEW)

  <M>SCSI backend driver (NEW)

  <M>Block-device frontend driver

  <M>Network-device frontend driver

  <M>Network-device frontend driver acceleration for Solarflare NICs (NEW)

  <M>SCSI frontend driver (NEW)

  <*> User-space granted page access driver (NEW)

<*> Framebuffer-device frontend driver (NEW)

<*> Keyboard-device frontend driver (NEW)

[*] Disable serial port drivers (NEW)

<*> Export Xen attributes in sysfs (NEW)

(256) Number of guest devices (NEW)

Xen version compatibility (3.0.4 and later)

Después de la configuración se procede a la construcción e instalación del kernel usando los siguientes comandos:

- *make O=~/build/linux-2.6.27-xen/*
- *make O=~/build/linux-2.6.27-xen/ modules_install install*

Para verificar la instalación del kernel modificado se usa un commando de listado de archivos en el directorio */boot/.*

```
-rw-r--r-- 1 root root   504280 2009-01-29 22:23 abi-2.6.27-11-server
-rw-r--r-- 1 root root   503560 2008-11-04 22:22 abi-2.6.27-7-server
-rw-r--r-- 1 root root    85313 2009-01-29 22:23 config-2.6.27-11-server
-rw-r--r-- 1 root root    87256 2009-02-12 20:51 config-2.6.27.5
-rw-r--r-- 1 root root    85319 2008-11-04 22:22 config-2.6.27-7-server
drwxr-xr-x 2 root root     4096 2009-02-12 22:28 grub
-rw-r--r-- 1 root root  8983433 2009-02-12 22:28 initrd.img-2.6.27-11-server
-rw-r--r-- 1 root root  8979323 2009-02-12 22:26 initrd.img-2.6.27-7-server
drwx------ 2 root root    16384 2009-02-12 19:30 lost+found
-rw-r--r-- 1 root root   124152 2008-09-11 22:11 memtest86+.bin
-rw-r--r-- 1 root root  1354638 2009-01-29 22:23 System.map-2.6.27-11-server
-rw-r--r-- 1 root root  1258568 2009-02-12 20:51 System.map-2.6.27.5
-rw-r--r-- 1 root root  1351952 2008-11-04 22:22 System.map-2.6.27-7-server
-rw-r--r-- 1 root root     1130 2009-01-29 22:27 vmcoreinfo-2.6.27-11-server
-rw-r--r-- 1 root root     1129 2008-11-04 22:25 vmcoreinfo-2.6.27-7-server
-rw-r--r-- 1 root root  2341536 2009-01-29 22:23 vmlinuz-2.6.27-11-server
-rw-r--r-- 1 root root  2192827 2009-02-12 20:51 vmlinuz-2.6.27.5
-rw-r--r-- 1 root root  2338976 2008-11-04 22:22 vmlinuz-2.6.27-7-server
-rw-r--r-- 1 root root   470144 2008-10-06 20:15 xen-3.3.gz
```

Figura 3.1: Lista de Kernels instalados sobre una versión servidor de Ubuntu

El resultado será una lista como muestra la Figura 3.1, en la cual se puede apreciar la existencia de un nuevo kernel, el 2.6.27.5, sin embargo carece de ramdisk, por lo que es necesario construir uno usando los siguientes comandos:

- *depmod 2.6.27.5*
- *update-initramfs -c -k 2.6.27.5*
- *update-grub*

d. Finalización de la instalación

Uno de los últimos pasos para la instalación del kernel modificado, es revisar el archivo */etc/modules* a fin de que el parámetro *loop max_loop* sea igua a 64, este paso es necesario cuando se planea crear VM's basadas en imágenes de disco, de lo contrario si se usa VM's basadas en LVM se lo puede pasar por alto. Para finalizar la instalación reiniciamos el equipo, y en el GRUB elegimos la distribución que contenga la palabra Xen.

3.2.2- Diseño, construcción y despliegue del escenario con máquinas virtuales

Para evaluar el rendimiento de redes IP en virtualización se creó un entorno de red LAN, con variaciones en el número de estaciones de trabajo y en los parámetros de funcionamiento de la red.

En concreto, dado un retardo y un ancho de banda específico se midieron a través de herramientas de análisis de red indicadores como: throughput, pérdida de paquetes, jitter. A fin de determinar cuál es el indicador de red que permite analizar mediante técnicas estadísticas el rendimiento de la red virtualizada.

Para realizar una comparación del dimensionado de redes a nivel de tecnologías, se homologaron los experimentos en cuanto a: protocolos de transporte, protocolos de aplicación, mecanismos de inyección de tráfico, mecanismos de restricción de recursos, tiempo de experimentación, procesamiento de resultados.

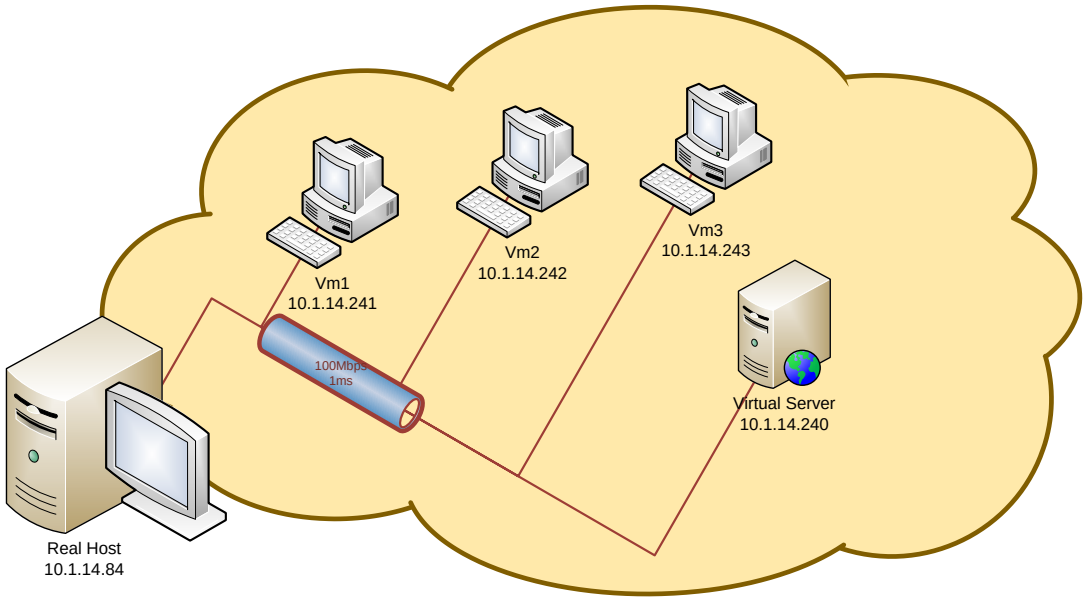

Figura 3.2: Diseño del escenario LAN con 4 VM's

La Figura 3.2, muestra uno de los diseños iniciales de la red LAN, con un escenario compuesto por 4 máquinas virtuales se comprobaría el alto rendimiento de XenServer con respecto a otras plataformas virtuales como lo afirma en [5], consolidando a Xen como la mejor plataforma de virtualización.

Para exigir al máximo a la plataforma de virtualización se diseñaron y desplegaron varios escenarios de red, entre ellos de 4,6,7,10,11 y 16 máquinas virtuales.

La Figura 3.3, muestra uno de los diseños finales de la red LAN, con un escenario de 11 máquinas virtuales, de las cuales 10 hacen las veces de clientes en la red. El objetivo que se persigue con estos diseños es comprobar la degradación del rendimiento de la red a medida de que incrementan las estaciones de trabajo con la condición de que todas las estaciones inyecten tráfico UDP al mismo tiempo.

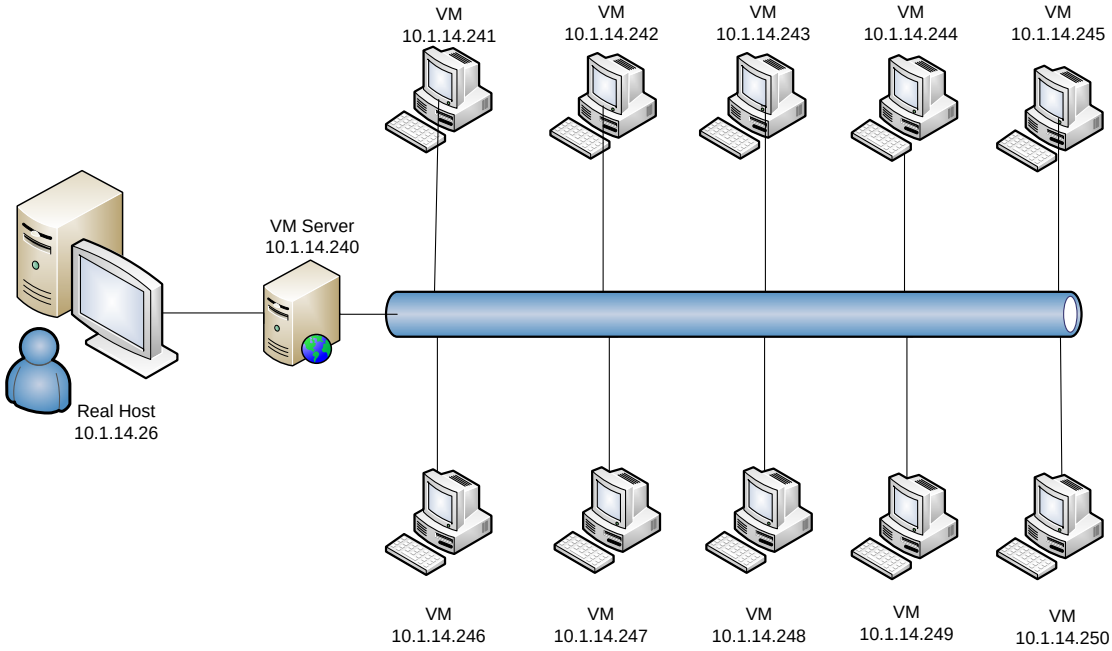

Figura 3.3: Diseño del escenario LAN con 11 VM's

El crear una máquina virtual mediante Xen demanda un conjunto de pasos, en los que haciendo uso de las herramientas de Xen se ejecuten comandos, estos obedecen al tipo de máquina virtual que se desee instalar, en este panorama si es una instalación basada en imágenes de disco se usarán instrucciones especificas para creación de imágenes, de otra forma si es una instalación basada en LVM se usarán instrucciones específicas para el Manejo de Volúmenes Lógicos. Se ha

seguido un conjunto de pasos (ver apéndice A), para crear una máquina virtual mediante Xen basada en imágenes de disco.

Se puede desplegar el escenario de red virtualizado ejecutando comandos de Xen-Tools en diferentes terminales de GNU/Linux o de otra forma se puede crear un script Shell (ver apéndice B), que permita desplegar el escenario automáticamente.

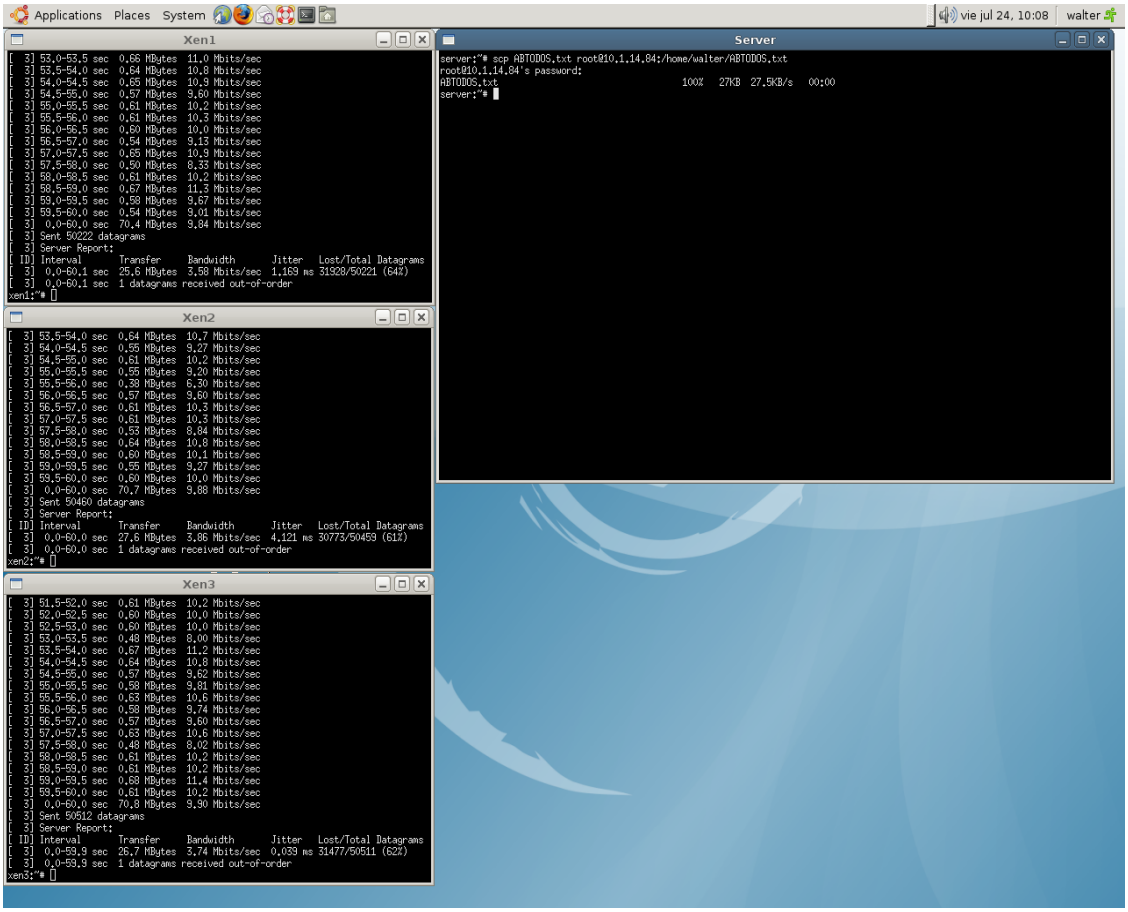

Figura 3.4: Despliegue de 4 VM's, 3 Clientes y 1 Servidor

La Figura 3.4, muestra el escenario de red virtualizado, se puede observar que están presentes 3 estaciones tipo cliente y una estación tipo servidor, desplegadas sobre Ubuntu Server 8.10 con Kernel 2.6.27.5 y Xen 3.3.1.

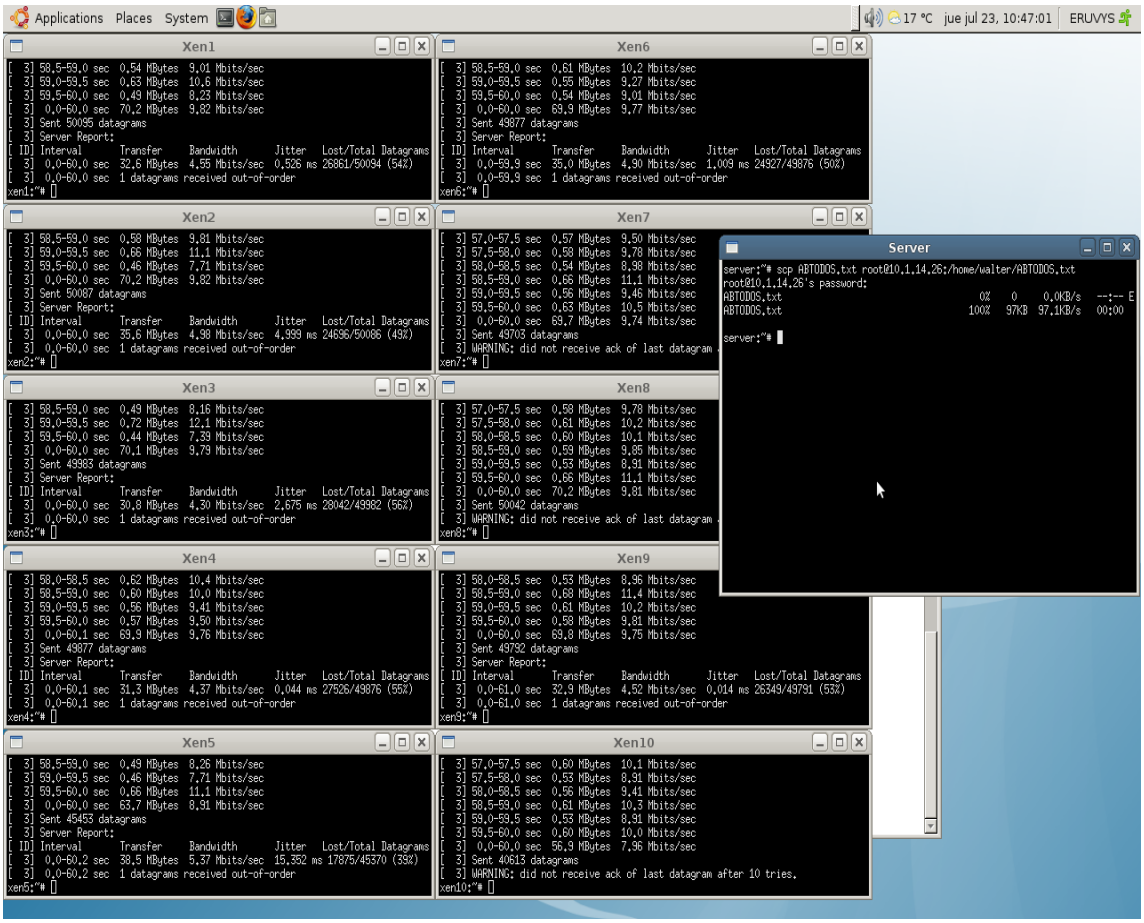

Figura 3.5: Despliegue de 11 VM's 10 Clientes y 1 Servidor

La Figura 3.5, muestra el escenario de red virtualizado cuando están presentes 10 estaciones tipo cliente y una estación tipo servidor, desplegadas sobre Ubuntu Server 8.10 con Kernel 2.6.27.5 y Xen 3.3.1. Este escenario de red en particular sirvió para la comprobación en la degradación del rendimiento red, el principal indicador de red considerado fue la pérdida de paquetes.

3.2.3- Esquema de direccionamiento IP

Para identificar cada una de las VM’s y considerando que el alcance de la red es local, se usaron los siguientes parámetros mostrados en la tabla 3.1.

Tabla 3.1: Esquema de direccionamiento de red para 11 VM's

Máquina Virtual	**Dirección IP**	**Máscara de Subred**	**Puerta de enlace predeterminada**	**DNS preferido y alternativo**
xen.server.com	10.1.14.240	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen1.example.com	10.1.14.241	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen2.example.com	10.1.14.242	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen3.example.com	10.1.14.243	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen4.example.com	10.1.14.244	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen5.example.com	10.1.14.245	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen6.example.com	10.1.14.246	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen7.example.com	10.1.14.247	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen8.example.com	10.1.14.248	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen9.example.com	10.1.14.249	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104
xen10.example.com	10.1.14.250	255.255.255.0	10.1.14.1	10.1.0.101 – 10.1.0.104

3.2.4- Inyección de tráfico

Para inyectar tráfico en la virtualización se usó iperf, que es una herramienta que mide el rendimiento de la red de acuerdo a varios indicadores. Iperf usa el modelo cliente – servidor para realizar las pruebas de rendimiento, principalmente es usada para medir el ancho de banda máximo de un enlace.

La sintaxis para el uso de esta herramienta se presenta a continuación:

- iperf -s [options]
- iperf -c server [options]
- iperf -u -s [options]
- iperf -u -c server [options]

Algunas de las opciones de uso general para iperf son las siguientes:

- -f, --format [kmKM] formato del reporte: Kbits, Mbits, KBytes, MBytes
- -i, --interval n tiempo de intervalo entre reporte
- -l, --len n[KM] asigna el tamaño del buffer de lectura/escritura
- -p, --port n asigna el puerto por el que se escuchará o se conectará
- -u, --udp realiza la prueba con UDP en vez de TCP
- -w, --window n[KM] Tamaño de ventana TCP
- -C, --compatibility para uso con versiones antiguas
- -M, --mss n asigna el máximo tamaño de segmento TCP
- -h, --help imprime un resumen de ayuda

Para medir el rendimiento de red se usaron paquetes UDP, generados mediante CBR, en el lado del servidor se necesitó usar la siguiente instrucción:

- iperf -s -u -i 0.5 -p 5001 -f m

En el lado de los clientes se usó la siguiente instrucción por cada estación tipo cliente presente en el escenario:

- iperf -c 10.1.14.240 -u -i 0.5 -p 5001 -f m -b 10.0M -t 60

3.2.5- Aplicación de Software de medición de tráfico y Evaluación

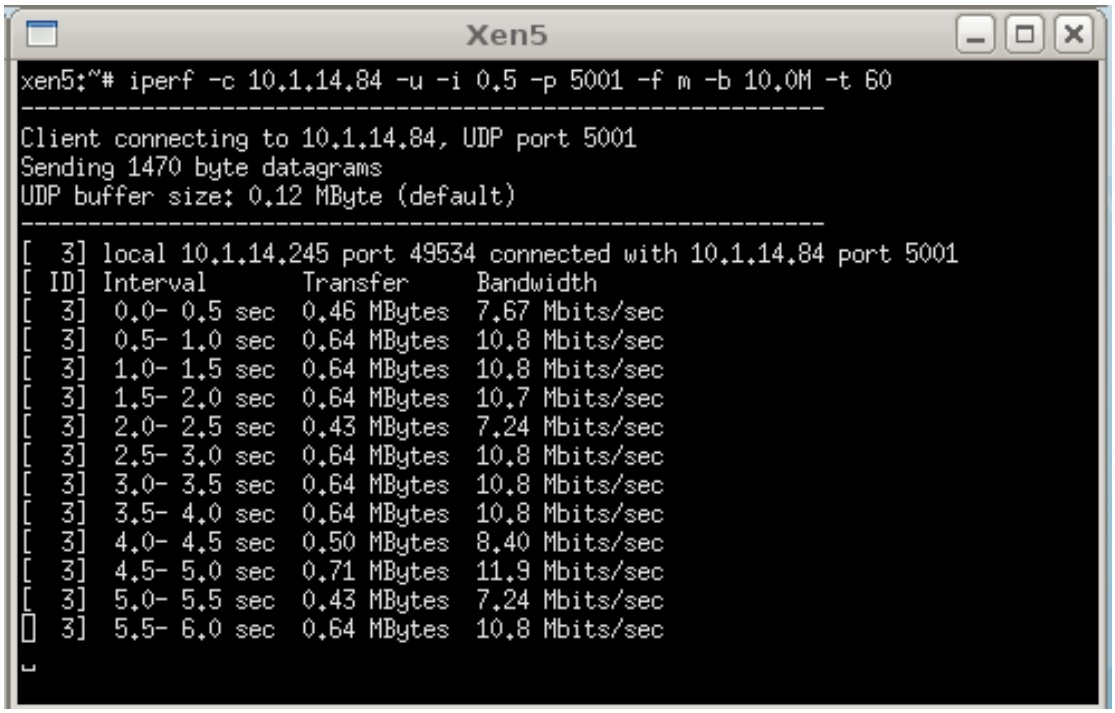

Figura 3.6: Reporte de Iperf cuando se inyecta tráfico CBR/UDP

La Figura 3.6, exhibe el reporte generado por la herramienta de rendimiento de red Iperf, como se puede apreciar se obtienen datos útiles para procesar en la comparación de tecnologías. Iperf reporta datos como: ancho de banda en un intervalo determinado, MTU, buffer de envio/recepción, conexiones activas en el enlace, paquetes perdidos en la prueba, variación del retardo, entre otros.

Iperf ha sido evaluada de acuerdo a la RFC 2544, que describe los requerimientos para que un software de red sea considerado como herramienta de rendimiento, de lo cual según [31] se han obtenido los siguientes resultados:

- **Verificación de paquetes recibidos** – Iperf solo determina paquetes perdidos
- **Tráfico bidireccional** – Iperd soporta completamente esta característica
- **Asignación del tiempo entre tramas** – Iperf no soporta esta característica
- **Formato de paquete definido por el usuario** – Iperf solo puede especificar contenido de datos UDP en la definición de tramas.

3.3- Conclusiones

Gracias a los niveles de privilegios que soportan actualmente los procesadores, se puede usar la paravirtualización, Xen aprovecha esta característica del hardware y se adapta a un esquema de privilegios mediante la modificación del kernel en SO, aumentando su rendimiento de virtualización sobre otras plataformas.

Los discos Ram o Ramdisk son espacios de memoria reservada que permiten a aplicaciones o conjunto de archivos ser accedidos de manera más eficiente en términos de tiempo. Esta técnica es útil sobre todo cuando se necesita que cierto conjunto de archivos sean accedidos de manera rápida y masiva.

El diseñar los escenarios de red para virtualización considerando una escala ascendente de estaciones presentes, permite observar la degradación del rendimiento de red de acuerdo a los indicadores de red.

El número de máquinas virtuales que se pueden desplegar depende del hardware disponible, haciendo de esto una dificultad al momento de experimentar con escenarios amplios de red.

CAPITULO 4 DISEÑO E IMPLEMENTACION DEL ESCENARIO DE RED UTILIZANDO METODOS DE SIMULACIÓN DE REDES

4.1- Introducción

Los Métodos de Simulación son una tecnología prominente en el ámbito de la investigación, son utilizados para medir el rendimiento de las redes IP.

Proveen un ambiente repetible y controlable para imitar el funcionamiento de una red experimental durante un intervalo de tiempo. Estos métodos permiten modelar un conjunto de supuestos que se expresan a través relaciones lógicas y matemáticas que evolucionan en el tiempo. Sin embargo los métodos de simulación no son capaces de reproducir el funcionamiento total del hardware.

Esta investigación tiene como propósitos: diseñar, implementar y poner en funcionamiento escenarios simulados (mediante NS-2) a fin de validar el rendimiento de redes IP; realizar varios experimentos a fin de visualizar como se va degenerando el rendimiento a medida que se incrementan equipos en la red.

4.2- Análisis, diseño e implementación del escenario virtual

4.2.1- Instalación del NS2

La plataforma de simulación Network Simulator 2 o NS-2 para abreviar, está disponible en varios sitios de la Internet, es un paquete de software para simulación de redes de dominio público, con el cual se pueden simular varios protocolos usados para Internet.

Mediante NS-2 se puede simular redes terrestres, inalámbricas y hasta satelitales, lo que hace de este software una herramienta poderosa en el dimensionado y pruebas de red. Además de que es una de las opciones más

populares en la investigación científica, avalada por una comunidad que mantiene y actualiza el software constantemente.

El método de instalación de NS-2 presentado en adelante muestra los pasos necesarios para situar la plataforma de simulación en una distribución GNU/LINUX llamada Ubuntu Server, esta plataforma no requiere necesariamente de un sistema operativo tipo servidor.

Todos los comandos que necesiten ejecutarse durante la instalación se deben realizar con privilegios de super usuario (root) o en su defecto se debe usar el prefijo *sudo*, que permite tener temporalmente privilegios avanzados.

a. Validación de los requisitos previos

Para instalar NS-2 es necesario contar con una copia comprimida del software, la misma que puede ser descargada del siguiente enlace:

- http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/?_test=var2

Es necesario instalar las librerías *libxmu-dev* a través del siguiente comando en la consola de GNU/Linux.

- apt-get install build-essential autoconf automake libxmu-dev

b. Instalación y configuración del software

El primer paso para la instalación es descomprimir el archivo .tar.gz mediante el comando *tar* en un directorio conocido, como por ejemplo /home/usuario.

Dentro del directorio donde se descomprimió el software se ejecuta el comando:

- ./install

A continuación se fijan los parámetros globales para que el software funcione apropiadamente, para lo cual se debe editar el fichero *~/.bashrc*. Al final del archivo añadimos las siguientes líneas, cuidando de cambiar “/mi/directorio” por el directorio en el que fue descomprimido NS-2(en este caso /home/usuario).

 # LD_LIBRARY_PATH

OTCL_LIB=/mi/directorio/ns-allinone-2.33/otcl-1.13

NS2_LIB=/mi/directorio/ns-allinone-2.33/lib

X11_LIB=/usr/X11R6/lib

USR_LOCAL_LIB=/usr/local/lib

export

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$U

SR_LOCAL_LIB

 # TCL_LIBRARY

TCL_LIB=/mi/directorio/ns-allinone-2.33/tcl8.4.18/library

USR_LIB=/usr/lib

export TCL_LIBRARY=$TCL_LIB:$USR_LIB

 # PATH

XGRAPH=/mi/directorio/ns-allinone-2.33/bin:/mi/directorio/ns-allinone-

2.33/tcl8.4.18/unix:/mi/directorio/ns-allinone-2.33/tk8.4.18/unix

NS=/mi/directorio/ns-allinone-2.33/ns-2.33/

NAM=/mi/directorio/ns-allinone-2.33/nam-1.13/

PATH=$PATH:$XGRAPH:$NS:$NAM 1.13/

PATH=$PATH:$XGRAPH:$NS:$NAM

Para finalizar la instalación reiniciamos el equipo, y probamos en la consola de GNU/Linux el comando *ns*, el resultado debería ser un mensaje descriptivo respecto a que faltan argumentos para la simulación.

4.2.2- Diseño y construcción del escenario de red

Para llevar a cabo esta investigación, se diseñaron e implementaron diferentes escenarios de prueba utilizando NS2 en la misma plataforma de hardware y software base. Luego se configuraron aquellos parámetros que se relacionan con el rendimiento de red como BW, latencia y pérdida de paquetes. A continuación se aplicaron métodos de inyección de tráfico UDP mediante diversos algoritmos de generación de tráfico con el nombre de *Agentes de tráfico*. Finalmente se modificaron dichos escenarios con mayor número de equipos en la red para comprobar su comportamiento.

Con miras a comparar el rendimiento de red entre simulación y virtualización se homologaron los diseños de escenarios de red usados en la virtualización, de la misma manera los protocolos y demás parámetros del dimensionado de redes.

OTcl, es una extensión de Tcl orientada a objetos, este lenguaje se utiliza para la creación de scripts entendibles para el compilador de NS-2, y que

permiten desplegar un escenario de red simulado que es animado por una herramienta llamada *nam* o *network animator.*

Se crearon varios scripts OTcl (ver apéndice C), en los que se simularon 4,6,7,10,11 y 16 nodos o equipos de la red, una vez más con el fin de homologar los experimentos realizados en virtualización.

Para visualizar el comportamiento de un escenario simulado es necesario usar el comando *ns* acompañado del nombre del archivo OTcl, en el que se detallan mediante programación las características de la red.

La Figura 4.1, muestra la animación de un escenario de red simulado, con 5 nodos tipo cliente y uno tipo servidor, esta animación es muy útil para visualizar el posible comportamiento que tendrán los nodos de acuerdo a las restricciones y capacidades de la red. Uno de los indicadores de red más notables en NS-2 por su forma de manifestarse en la animación es la perdida de paquetes.

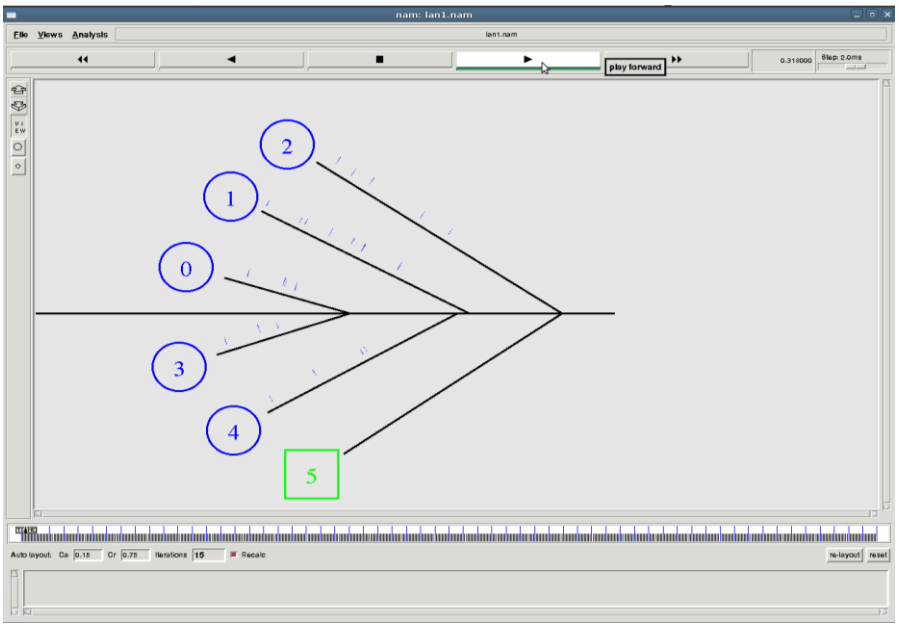

Figura 4.1: Escenario de red simulado mediante NS-2 y animado mediante NAM

4.2.3- Inyección de tráfico

En NS-2 se usan *agentes* cuando se necesita simular cierta aplicación o protocolo. Los *agentes* de NS-2 son clases de programación orientada a objetos en ciertos casos derivadas de una mayor, que permiten mediante programación simular el comportamiento que tendrá en la red un protocolo o aplicación.

Existen varios tipos de agentes generadores de tráfico en NS-2, para determinar el más adecuado a las necesidades se debe considerar cual será el protocolo de transporte, puesto que la combinación errónea de agentes puede incurrir en resultados poco confiables en la simulación.

Entre los agentes generadores de tráfico se puede mencionar a:

Exponential, que genera tráfico de acuerdo a una distribución exponencial con momentos de encendido y momentos de apagado, los paquetes son de tamaño constante al igual que la tasa de envío

Pareto, que genera tráfico de acuerdo a la distribución de Pareto, es idéntica al agente exponencial excepto en que los periodos de encendido y apagado son tomados de una distribución de Pareto.

CBR, que genera tráfico de acuerdo a una tasa de envío constante, los paquetes son de tamaño constante y opcionalmente se puede introducir ruido aleatorio en la señal.

Traffic Trace, que genera tráfico de acuerdo a un archivo de traza, cada registro en el archivo de traza consiste en 2 campos de 32 bits, el primer campo contiene el tiempo en microsegundos que pasan hasta que el siguiente paquete sea generado, y el segundo campo contiene el tamaño en bytes del siguiente paquete.

Un agente generador de tráfico en NS-2 necesita estar adjuntado a otro agente de transporte, en el siguiente conjunto de instrucciones se muestra la forma de adjuntar un agente generador de tráfico a un agente de transporte.

- Se crea un Agente UDP y lo adjunto al nodo

```
set source [new Agent/UDP]
$ns attach-agent $node $source
```

- Se crea un agente de Trafico CBR

```
set traffic [new Application/Traffic/CBR]
$traffic set type_ CBR
$traffic set packet_size_ $size
$traffic set rate_ $rate
$traffic set random_ 1
$traffic set interval_ 0.01
```

4.2.4- Evaluación

En cuanto a facilidad de instalación, NS-2 es una herramienta que no genera mayores complicaciones, está respaldada por una comunidad de usuarios que la mantienen activa y actualizada. Los sitios de descarga en la Internet son abundantes y el soporte es preciso en los foros y blog's creados por la comunidad. Es compatible con versiones de GNU/Linux y se la puede usar en Windows por medio de un emulador de consola GNU/Linux. Las actualizaciones de librerías están disponibles en los repositorios globales de GNU/Linux.

En cuanto a facilidad de uso, NS-2 requiere de un esfuerzo en aprendizaje de sintaxis para la codificación de los archivos tcl, la Internet provee de varios ejemplos y los manuales de la aplicación son precisos y prácticos en su terminología. Para alcanzar resultados confiables en la simulación se debe balancear entre conocimientos de redes y conocimientos de programación, a fin de que las combinaciones de agentes representen lo que se planifica del experimento.

4.3- Conclusiones

NS-2 es una herramienta útil para el dimensionado de redes, sobre todo por la variedad de conceptos de red que es capaz de simular, además de la apertura para modificar sus clases de acuerdo a las necesidades del experimento.

Al igual que la virtualización, los métodos de simulación permiten cuantificar la degradación del rendimiento de red a medida de que aumentan nodos, y estos se encuentren generando tráfico simultáneamente.

Los resultados de un experimento de simulación son más confiables y se pueden mejorar con la homologación detallada de parámetros hacía un entorno de red real. El ajuste es necesario cuando la divergencia con resultados estadísticos medidos en tiempo real es alta.

CAPITULO 5 EVALUACION EXPERIMENTAL, AJUSTE Y DISCUSIÓN DE RESULTADOS

5.1- Aplicación de técnicas estadísticas para determinación de diferencias y similitudes en los resultados obtenidos tanto en virtualización como en simulación.

5.1.1- Técnica Estadística: Promedio

Se realizaron 10 experimentos de cada escenario en virtualización mediante Xen que incluyeron la creación de 3, 5, 10 y hasta 15 máquinas clientes, de la misma forma se implementaron la cantidad en nodos para la simulación mediante NS-2.

Tanto en la virtualización como en la simulación se ejecutaron experimentos en un período de tiempo de 60 segundos.

Con el conjunto de datos recolectados, se decidió utilizar la técnica estadística del promedio, llamada también Media Aritmética, la misma que arrojo los datos que posteriormente utilizamos para graficarlos. Se observó el rendimiento de la red de acuerdo al hardware de base disponible.

a. Validación de resultados en base al hardware disponible

La aplicación de la Media Aritmética sirvió para comprobar cuanto afecta la disponibilidad de los recursos físicos en el buen desempeño de una red virtualizada.

Los primeros experimentos fueron realizados en un PC con menores características al PC definitivo. Las características de los equipos mencionados se detallan en la siguiente Tabla:

Tabla 5.1: Comparación de características técnicas de los equipos base

Item	Descripción	Equipo Antiguo (EA)	Equipo Nuevo (EN)
1	Procesador	Intel Pentium 4 3.2Ghz	Intel Core ™ 2 Quad CPU 2.4 Ghz.
2	Cache size	2048 KB	8192 KB
3	RAM Total	1 GB	4GB
4	Partición HD	108 GB	130 GB
5	Virtualización por Hardware	NO	SI

Teniendo en cuenta los recursos del equipo antiguo (EA de aquí en adelante), y sabiendo que por tener poca disponibilidad de hardware solo se puede llegar a desplegar dos escenarios con 3 y 5 estaciones (VM's y nodos), y a modo de comparación se opto por realizar la misma experimentación en el equipo nuevo (EN de aquí en adelante), que sería posteriormente el quipo de experimentos definitivo.

b. Comparación del rendimiento en base al BW.

La Figura 5.1, ilustra el rendimiento del BW obtenido en un escenario de 3 estaciones y un servidor durante 10 mediciones. La figura sirve para apreciar que la medición del BW no fue la forma más legible de presentar los resultados, puesto que, para que los mismos sean analizados visualmente fue necesario calcular la media aritmética que por ser una medida de tendencia central, puede llegar a producir sesgos en las mediciones.

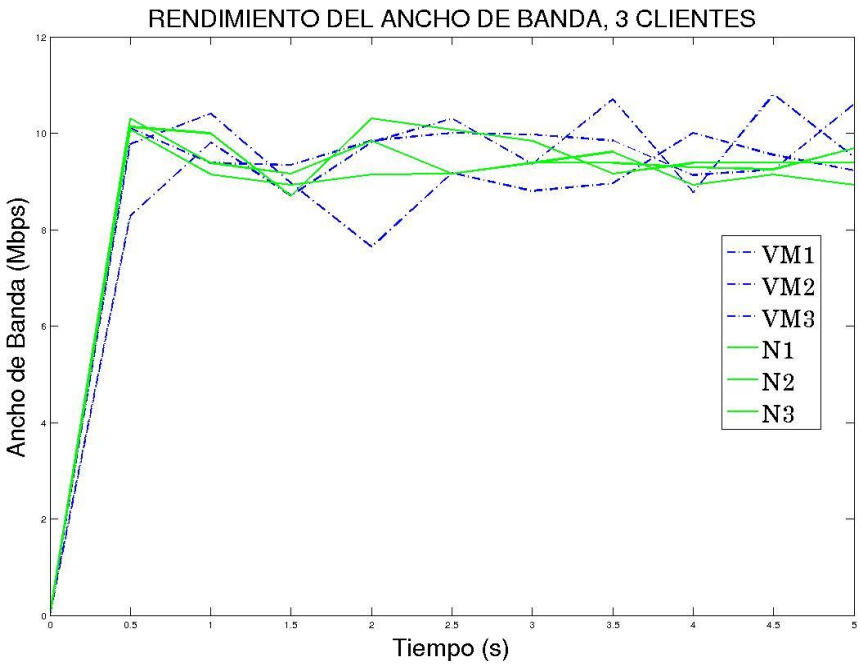

Figura 5.1: Rendimiento del Ancho de Banda, 3 estaciones, entorno Virtualizado vs. Simulado

5.1.2- Técnica Estadística CDF

Como se menciono anteriormente la técnica estadística de la media no ayudo con un resultado que satisficiera la necesidad de observar el rendimiento de una red, de forma consistente. Por ello se decidió analizar la técnica CDF, Función de Distribución Acumulada que permitió asegurar la confiabilidad de los resultados.

a. Comparación entre escenarios de red en base a la pérdida de paquetes

De las 10 mediciones realizadas en Xen, se decidió procesar los datos del experimento que mejor rendimiento alcanzó basados en la pérdida de paquetes

IP. Mientras que, en simulación mediante NS-2 cómo se explicó en párrafos anteriores, los resultados se mantienen constantes.

Las Figuras 5.2 y 5.3, muestran los resultados obtenidos al comparar la pérdida de paquetes entre Xen y NS-2. Como se puede observar las curvas de función de probabilidad acumulada difieren considerablemente cuando están presentes 3 o 5 estaciones.

La diferencia de los resultados en gran medida se debe al *overhead* generado por la capa de virtualización.

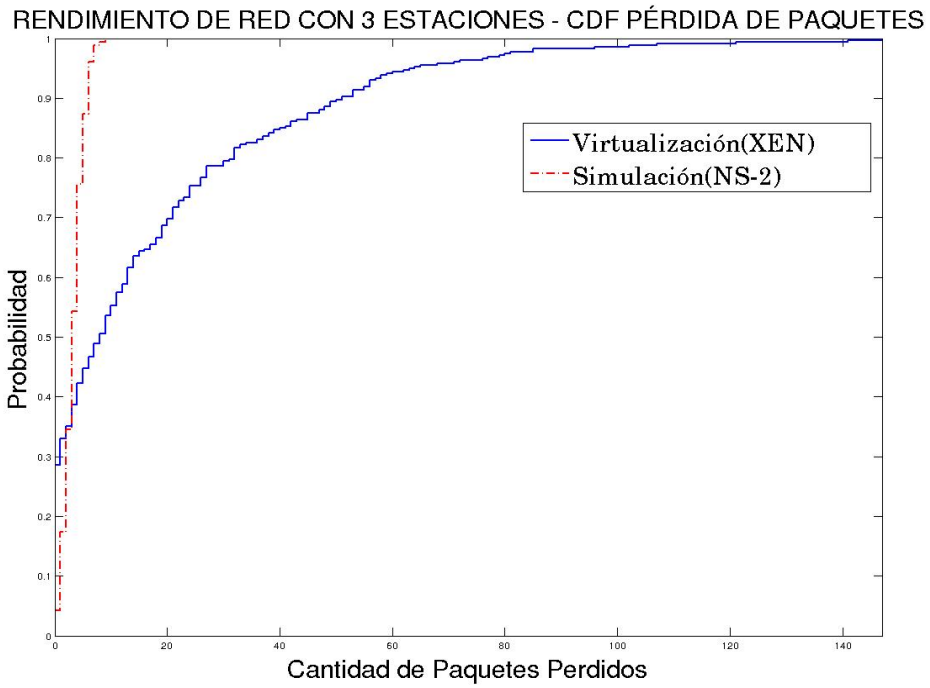

Figura 5.2: Función de distribución de probabilidad acumulada de paquetes perdidos con 3 estaciones

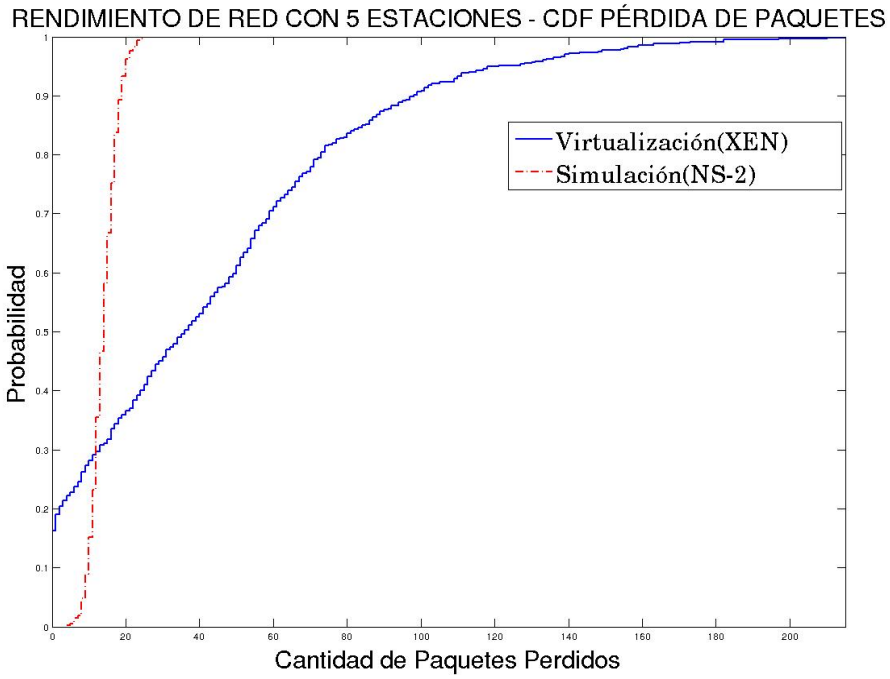

Figura 5.3: Función de distribución de probabilidad acumulada de paquetes perdidos con 5 estaciones

Las Figuras 5.4 y 5.5, muestran la comparación cuando están presentes 10 y 15 estaciones respectivamente. Como se puede apreciar a medida que se sigue incrementando el número de equipos, el rendimiento de red tiende a degradarse utilizando ambos entornos, es decir sigue incrementando el número de paquetes perdidos.

Así por ejemplo, mediante NS-2 se ha perdido la capacidad de que sean recibidos todos los paquetes en un instante determinado.

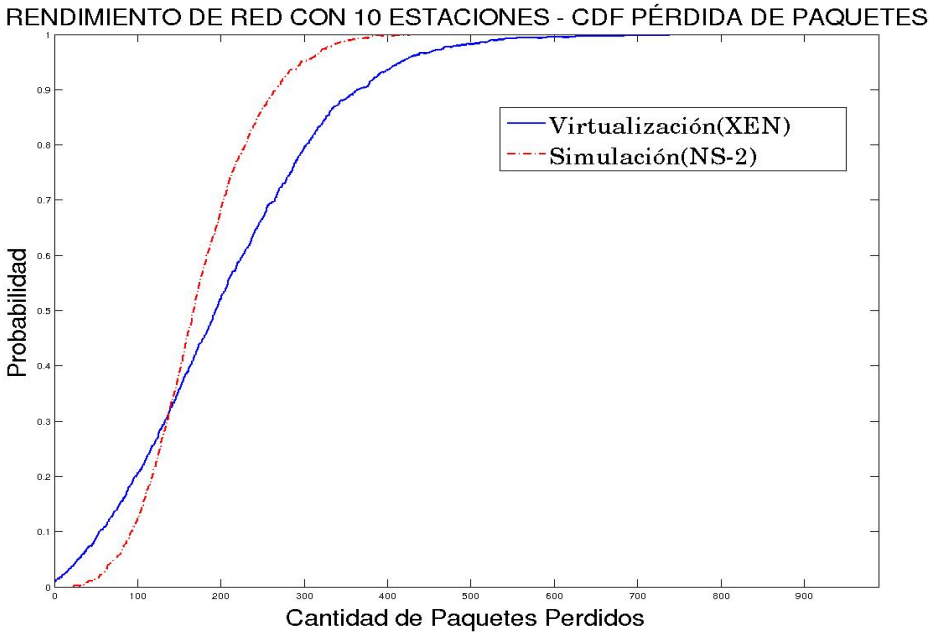

Figura 5.4: Función de distribución de probabilidad acumulada de paquetes perdidos con 10 estaciones

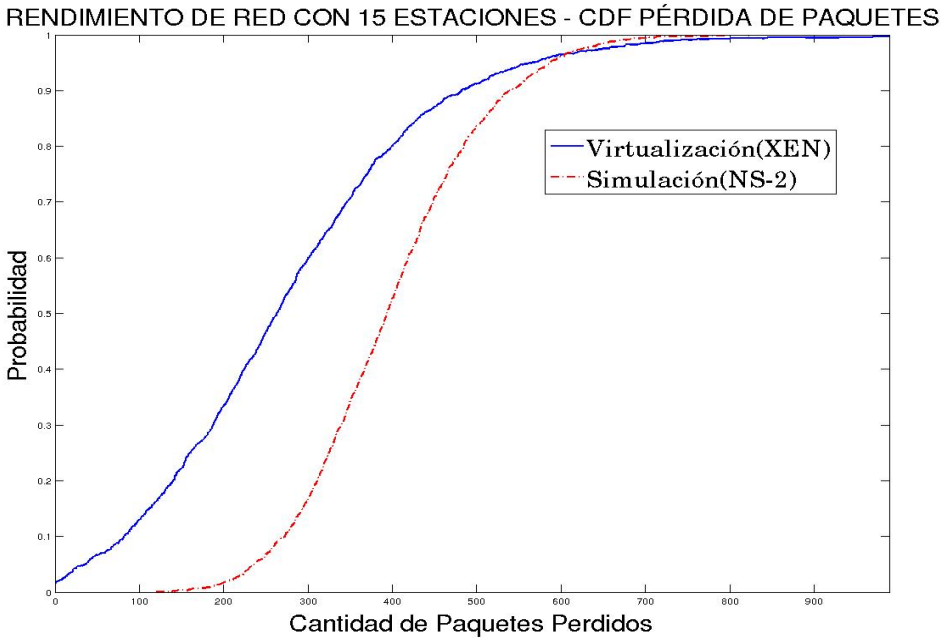

Figura 5.5: Función de distribución de probabilidad acumulada de paquetes perdidos con 15 estaciones

5.2- Análisis de los factores que afectan el rendimiento.

Los factores que contribuyen a la evaluación en el desempeño y rendimiento de una red, son varios, como por ejemplo:

- La capacidad: es el índice máximo de datos enviados;
- La propagación del retardo: cuanto le toma a un paquete viajar a través de la red;
- Longitud de los paquetes: bits transmitidos de los paquetes;
- Número de estaciones existentes;
- Protocolos utilizados;
- Tráfico agregado;
- Velocidad de conexión y conectividad;
- Disponibilidad y fiabilidad: dispuestas en medidas de tiempo, entre otros.

Para el presente proyecto se manejó factores que implicaban una comparación entre plataformas de virtualización y métodos de simulación, debiendo dirigir los esfuerzos sobre aquellos que arrojen Xen y NS-2.

5.2.1- Factor: Generador de Tráfico

Sabiendo que uno de los factores que participan en el desempeño de una red, son los agentes generadores de tráfico, se procedió a determinar el mejor agente generador de tráfico en NS-2 que participe con el protocolo UDP. Se realizaron varias pruebas ajustando el tamaño del paquete IP.

La Figura 5.6, muestra los resultados porcentuales en pérdida de paquetes de los diferentes agentes generadores de tráfico. Como se puede apreciar,

EXPONENTIAL es el agente que mejores resultados alcanza, entregando el 99.82% de los paquetes generados en la prueba.

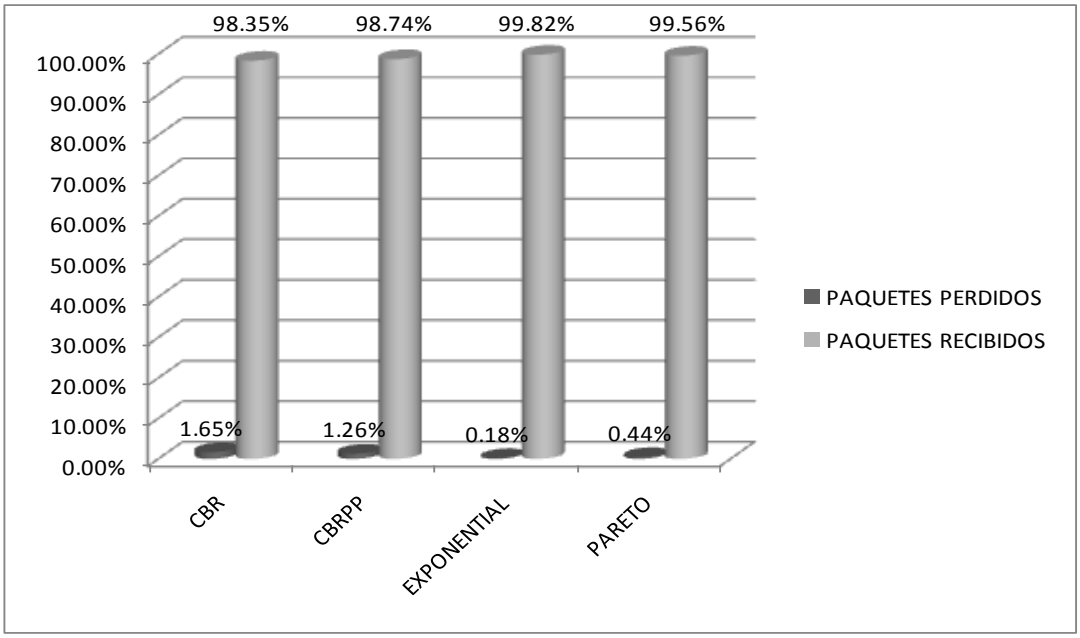

Figura 5.6: Pérdida de paquetes IP por agente generador de tráfico en simulación

A pesar de que *EXPONENTIAL* según los resultados, es el agente generador de tráfico que mejor rendimiento tiene en el entorno simulado, se decidió usar *CBR* para homologar el mismo tipo de agente en el entorno virtualizado, considerando que de acuerdo con [3] *Iperf* es una herramienta de inyección de tráfico CBR.

5.2.2- Factor: hardware base disponible

Las Figuras 5.7 y 5.8, muestran cómo la disponibilidad de los recursos de hardware de los 2 equipos afecta el rendimiento en los experimentos realizados.

En el caso de la virtualización, el rendimiento de la red se degenera en el Equipo EA (incrementa el overhead) y por lo tanto disminuye el rendimiento. En el caso del EN los resultados muestran el rendimiento según el BW establecido. En cuanto a la simulación se demuestra que no existe influencia del hardware base, pues presenta el mismo comportamiento en ambos equipos, es decir sin cambios en el rendimiento de la red. Eso sí, el tiempo de ejecución aumenta o disminuye conforme los recursos de hardware disponibles.

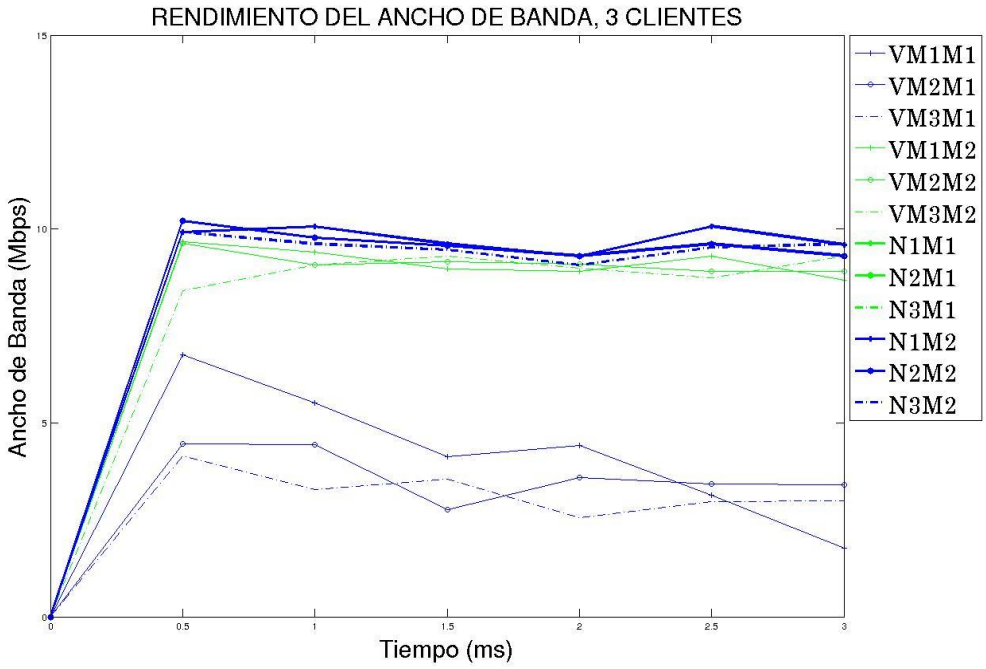

Figura 5.7: Comparación en base al hardware del EA y el EN, con 3 estaciones

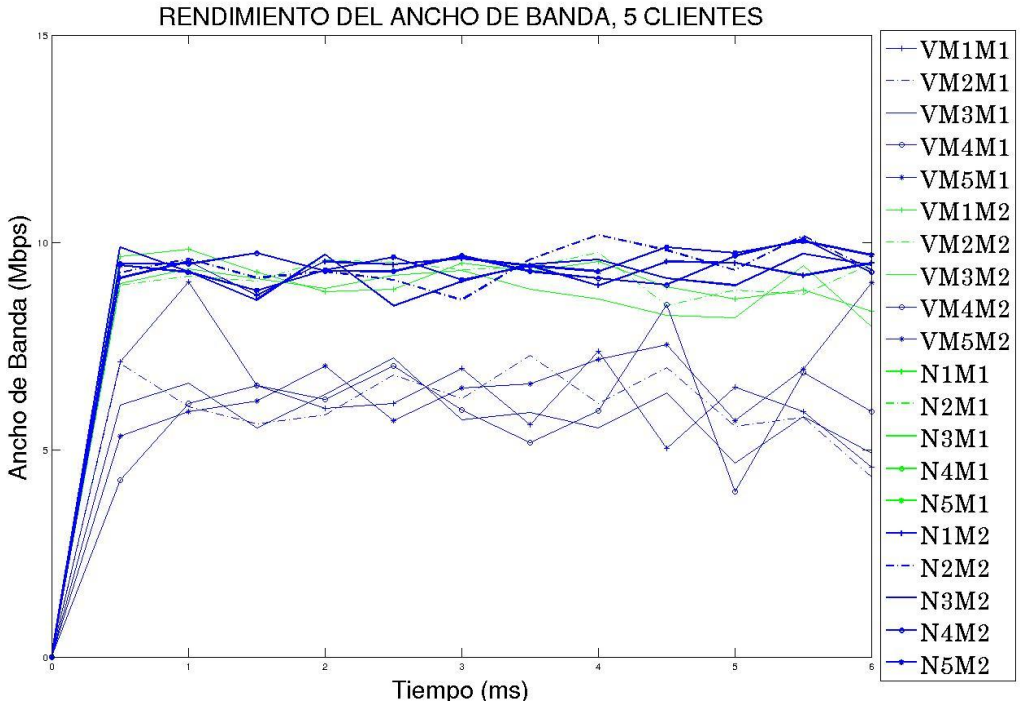

Figura 5. 8: Comparación en base al hardware del EA y el EN, con 5 estaciones

5.2.3- Factor: Paquetes perdidos incrementando número de equipos

La Figura 5.9: muestra la mayor degradación de la red que se produce en el entorno virtualizado cuantificando el número de paquetes perdidos dado el incremento de equipos en la red y en consecuencia en base al incremento del overhead en el hardware base. Los resultados muestran que la diferencia entre Virtualización y Simulación va en el orden de 3.71%, 8.8% y 30.17% respectivamente a 3, 5 y 10 estaciones presentes en el escenario.

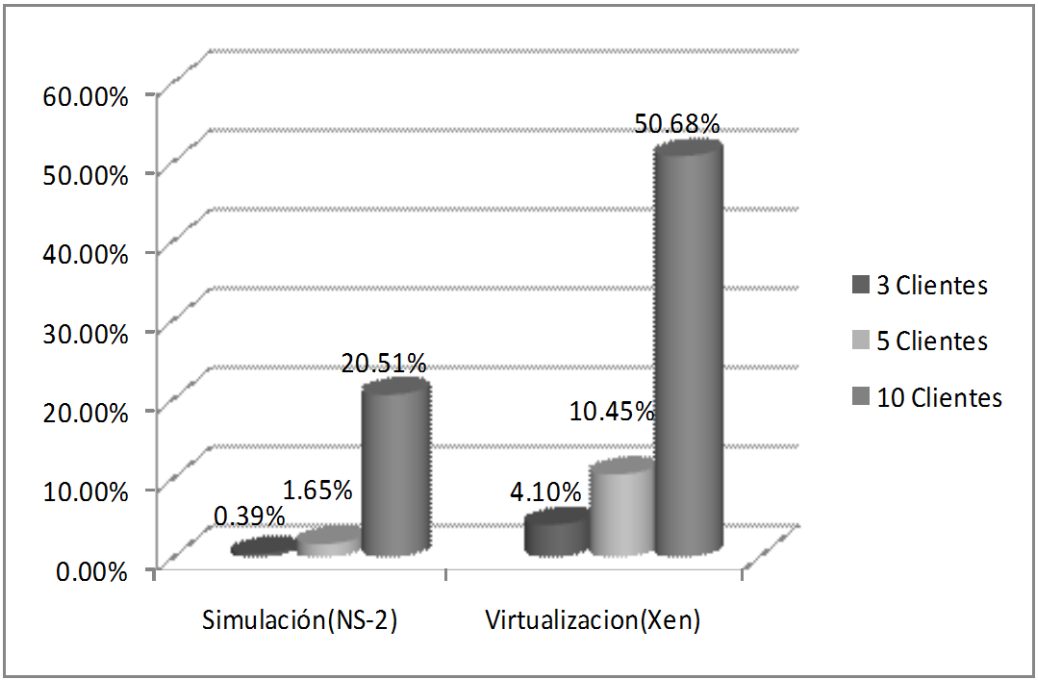

Figura 5.9: Porcentaje de paquetes perdidos sobre el total de paquetes generados

5.3- Ajuste y reconfiguración

Se ha demostrado que en el caso de la Simulación, los parámetros deben ser rigurosamente programados para mejorarlos.

En los experimentos realizados, nos pudimos dar cuenta, de parámetros que influían mucho en el comportamiento de la red, y que a medida que los ajustábamos con respecto a los valores generados en la virtualización, se incrementaba el funcionamiento a niveles buenos. Entre los parámetros están:

- Tamaño de paquete;
- Retardo;
- Ruido aleatorio;
- Intervalo;
- Cálculo del BW.

En el caso de la Virtualización, para ajustar los parámetros se utilizó la herramienta Jperf, que no es más que la misma herramienta Iperf, pero con interfaz gráfica, y con opciones dinámicas de procesamiento de datos. Además se deben adaptar otras condiciones operacionales como servidores dedicados, que incluyan característica de virtualización asistida por hardware, temporización, otras métricas de rendimiento y el mejoramiento del hardware base.

5.4- Obtención de resultados y discusión

La Evaluación Experimental realizada incluyó el diseñó, implementación y puesta en funcionamiento de escenarios simulados mediante NS-2 y virtualizados mediante Xen a fin de validar el rendimiento de redes IP.

Los primeros resultados experimentales ilustraron que existen diferencias al evaluar el rendimiento de la red a pesar de someter las dos tecnologías con los mismos escenarios y a las mismas pruebas. Luego se ajustaron aquellos parámetros que se relacionan con el rendimiento de red como BW, latencia, pérdida de paquetes, etc. A continuación se aplicaron métodos de inyección de tráfico UDP/TCP (para el caso de la Virtualización), así como diversos algoritmos de generación de tráfico (para el caso de la Simulación). Posteriormente, se tomaron varias medidas del rendimiento en los dos escenarios. Para contrastar estos resultados se realizaron algunas pruebas de validación ajustando parámetros.

Finalmente se modificaron dichos escenarios con mayor número de equipos en la red para comprobar cómo se produce la degeneración de la red a medida que se incrementan los equipos.

CAPITULO 6 CONCLUSIONES Y RECOMENDACIONES

6.1- Conclusiones Generales

- Se diseñó y configuró una LAN inicialmente de 3 estaciones dispuestas en una topología física en estrella, mediante el protocolo UDP, que luego se fue incrementando, 5, 10 y 15, conectadas a 1 servidor. Mediante el escenario se pudo poner en funcionamiento la red e interactuar desde todas las máquinas virtuales clientes, hacia el servidor virtual, y desde cada nodo cliente al servidor en simulación. En concreto, dado un retardo y el ancho de banda se midió y analizó el comportamiento de dicho escenario, cuando se realizó la transferencia de paquetes hacia el servidor.
- Las Herramientas escogidas en este proyecto de grado, permitieron trabajar de forma eficiente, y dinámica, logrando ser tecnologías potentes dentro del campo de investigación de redes, ya sea para simular o virtualizar. NS-2 mostró ser un Método flexible, por la posibilidad que brinda de trabajar con scripts en lenguaje OTCL permitiendo agregar cuantos algoritmos sean necesarios para una topología de red más clara y precisa. Y de acuerdo a los datos obtenidos de Xen, se determinó que es la herramienta que mejor se acopla cuando se trata de comparaciones en el rendimiento de una red, cumpliendo con ello los objetivos de diseñar e implementar escenarios de red en estos dos entornos.
- Con la implementación de métodos estadísticos, como la media, en el caso del rendimiento del BW cuando se incrementan equipos basándonos en el factor del hardware Base, y el método CDF utilizado para la pérdida de

paquetes, se logró comparar los resultados obtenidos mediante entornos virtualizados versus entornos simulados y entender que factores producen el overhead y son los que afectan el rendimiento de la red, luego se propuso ajustes que serían necesarios para el mejor desempeño de la misma.

- Los métodos de inyección de tráfico como Iperf, y algoritmos en el NS-2, incrementaron la garantía del análisis, con resultados coherentes, verificando la funcionalidad de los escenarios de red, utilizando el modelo cliente-servidor.
- En general a pesar de mantener constante el ancho de banda en ambas tecnologías, se observo una diferencia notable en el rendimiento de la red, en las tecnologías, debido básicamente a su propia naturaleza, ya que la virtualización se ejecuta en tiempo real, haciendo que cada experimento sea diferente y único, además de introducir una penalización no cuantificada por la capa de virtualización propia, contrario a la simulación, puesto que ésta al manejar algoritmos programados, y cálculos matemáticos, implica que los resultados van a ser siempre los mismos, sin diferenciar estructura física, ni entorno de desarrollo, pero influye la introducción de ruido aleatorio que inyecta el agente de tráfico. Se podría decir, que el aspecto que la hace eficiente a comparación de XEN, es el tiempo de ejecución rápido que le dedica cuando genera el escenario simulado.
- Una principal limitación, es que si se requiere de varias VM, se debe contar con un equipo físico con buenas características en recursos tanto lógicos como físicos.

- Como principales contribuciones de esta investigación se ha provisto de un estudio de las divergencias existentes entre los resultados al medir el rendimiento en las dos tecnologías citadas; y se ha verificado cómo se degenera el rendimiento de la red en ambos entornos al ser sometida a otras condiciones.

6.2- Recomendaciones

- Es importante empaparse del conocimiento sobre ambas tecnologías, Xen y NS-2, para su ocupación, para lograr que en la práctica sea más sencillo de analizar cualquier procedimiento a seguir.
- La correcta instalación de las Herramientas, mediante un buen tutorial, logrará evitar tiempos valiosos en los que se puede ocupar, sino estamos seguros de lo que instalamos.
- Para comenzar a programar en NS-2, se necesitara comenzar por los algoritmos básicos, que poco a poco, se irán incrementando de acuerdo al nivel de dificultad que se quiera implementar en la topología de red.
- Para comenzar a ver el rendimiento de una red, la topología de red, debe ser básica, estaciones, conectadas a un servidor, incluyendo poco a poco los elementos necesarios para armar un escenario de red que permita observar su desempeño, y sobre todo los factores que lo afectan.
- Analizar de manera cuidadosa los agentes generadores de tráfico que se desee implementar, así como el manejo de colas y los protocolos.

6.3- Trabajo Futuro

A continuación se exponen algunas de las líneas de investigación futura que se prende realizar:

- Trabajar en la parte en la que el rendimiento está siendo afectado, de forma que a partir de los resultados se pueda extrapolar cuánto error ha de haber en un experimento que se haga posteriormente. En otras palabras obtener estadísticamente la regresión del comportamiento de la red y cuantificar el comportamiento de sobrecarga.
- Se quiere representar una comunicación entre n nodos, o VM mediante un medio compartido, con conexión TCP.
- Contar con una herramienta de benchmarking para que las comparaciones sean estandarizadas en el rendimiento de la red para virtualización y simulación.

Apéndice A Creación de una máquina virtual mediante Xen basada en imágenes de disco

A.1- Introducción

A continuación se exhibe el conjunto de pasos necesarios para crear una máquina virtual mediante Xen basada en imágenes de disco, cabe comentar que después de crear la primera VM, ésta puede ser clonada para la creación del escenario de red, esto en términos de optimización del tiempo.

A.2- Proceso de creación de la máquina virtual

Para crear VM's en Xen se usan comandos propios llamados *Xen-Tools*, que son una colección de scripts perl para crear, actualizar y eliminar las máquinas virtuales y son instalados junto con el paquete de Ubuntu Xen, además existen otros paquetes como *Xen-Shell*, que permite administrar las VM's y Argo que es una interfaz web que permite controlar las VM's pero con cierto ovehead.

Primero empezamos editando el archive */etc/xen-tools/xen-tools.conf*, que contiene los valores por defecto para crear imágenes, solo es necesario modificar los siguientes parámetros, los demás dentro del archivo pueden permanecer en su estado original.

- dir = /home/xen
- memory = 256Mb
- dist = lenny .
- gateway = 10.1.14.240
- netmask = 255.255.255.0

- broadcast = 10.1.14.255
- passwd = 1
- kernel = /boot/vmlinuz-2.6.27-11-server
- initrd = /boot/initrd.img-2.6.27-11-server
- mirror = http://ftp.de.debian.org/debian/
- serial_device = hvc0
- disk_device = xvda #default

Antes de continuar se debe crear un directorio en donde las imágenes serán guardadas, en este caso el directorio creado es */home/xen*. El comando *xen-create-image* permite crear una imagen de máquina virtual con los parámetros por defecto del archivo *xen-tools.conf*, a menos de que se especifiquen nuevos parámetros como en la siguiente línea:

- xen-create-image --hostname=xen1.example.com --size=4Gb --swap=256Mb --ip=10.1.14.240 --arch=amd64 --role=udev

La ejecución del comando *xen-create-image* puede tomar un tiempo dependiendo del ancho de banda hacia el internet y de las características de hardware del equipo anfitrión. Luego de haberse creado la imagen se genera un archivo de configuración ubicado en */etc/xen* y deberá estar nombrado como sigue *xen1.example.com.cfg*, el mismo que debe parecerse a lo siguiente:

```
#
# Configuration file for the Xen instance xen1.example.com, created
# by xen-tools 3.9 on Wed Aug 12 16:20:14 2009.
#
#
#  Kernel + memory size
#
kernel      = '/boot/vmlinuz-2.6.27-11-server'
ramdisk     = '/boot/initrd.img-2.6.27-11-server'
```

```
memory      = '256'

#
#  Disk device(s).
#
root        = '/dev/xvda2 ro'
disk        = [
          'file:/home/xen/domains/xen1.example.com/swap.img,xvda1,w',
          'file:/home/xen/domains/xen1.example.com/disk.img,xvda2,w',
                ]

#
#  Hostname
#
name        = 'xen1.example.com'
#
#  Networking
#
vif         = [ 'ip=10.1.14.241,mac=00:16:36:2D:C4:A1' ]

#
#  Behaviour
#
on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Antes de iniciar por primera vez la máquina virtual se debe copiar un kernel apropiado para el buen funcionamiento, puesto que el comando *xen-create-image* copia el kernel del *Dom0*, para esto se recurre las siguientes líneas de comando:

- mount -o loop /home/xen/domains/xen1.example.com/disk.img /mnt
- cd /mnt/lib/modules/
- cp -pfr /lib/modules/2.6.27-11-server/ .
- cd
- umount /mnt

Para iniciar la VM que se ha creado se ejecuta *xm create /etc/xen/xen1.example.com.cfg* y para correrla se ejecuta *xm console*

xen1.example.com, se puede verificar el funcionamiento de la VM a través del comando *xm list*, con el que aparecerá el nombre de la VM en una lista.

Apéndice B Script para el despliegue automático del escenario de red virtualizado

B.1- Introducción

A continuación se presenta el script Shell que permite el despliegue automático de máquinas virtuales, el mencionado script ha sido elaborado en términos de optimización de tiempo.

B.2- Script para despliegue automático de VM's

```
#!/bashrc
#Nombre del archivo: progesc.sh
#Descripción: Script Shell para despliegue de máquinas virtuales
#Autores: Walter.Fuertes, Rodrigo Fonseca, MaFer Grijalva, Leo.Jácome
xm create /etc/xen/xen1.example.com.cfg
xm console xen1.example.com
Sleep 15
xm create /etc/xen/xen2.example.com.cfg
xm console xen2.example.com
Sleep 15
xm create /etc/xen/xen3.example.com.cfg
xm console xen3.example.com
Sleep 15
xm create /etc/xen/xen4.example.com.cfg
xm console xen4.example.com
```

```
Sleep 15
xm create /etc/xen/xen5.example.com.cfg
xm console xen5.example.com
Sleep 15
xm create /etc/xen/xen6.example.com.cfg
xm console xen6.example.com
Sleep 15
xm create /etc/xen/xen7.example.com.cfg
xm console xen7.example.com
Sleep 15
xm create /etc/xen/xen8.example.com.cfg
xm console xen8.example.com
Sleep 15
xm create /etc/xen/xen9.example.com.cfg
xm console xen9.example.com
Sleep 15
xm create /etc/xen/xen10.example.com.cfg
xm console xen10.example.com
Sleep 15
xm create /etc/xen/xen.server.com.cfg
xm console xen.server.com
Sleep 15
#Fin del script
```

Apéndice C Creación de Scripts OTcl para el despliegue de un entorno de red simulado

C.1- Introducción

En el caso de la simulación con NS-2 se configuró el escenario de red como sigue: primero se creó un nodo LAN que sería el concentrador de cada nodo cliente en la transferencia de los paquetes hacia el nodo servidor. Luego se incluyó el agente *LossMonitor* que permitió implementar un *sumidero* de tráfico, uno por cada cliente permitiendo obtener estadísticas sobre el número de bytes recibidos, número de paquetes perdidos, número de paquetes recibidos. Así mismo, se estructuraron dos procedimientos, uno que permitió añadir una fuente de tráfico CBR sobre un agente UDP para cada nodo cliente, y el otro que grababa periódicamente el BW calculado por los receptores de tráfico y escribía los resultados sobre archivos de texto plano.

C.2- Script OTcl para entornos de red simulados

```
#Descripción: Script para simulación de red mediante NS-2
#Nombre: lan.tcl
#Autores: MaFer Grijalva, Leonardo Jácome, Walter Fuertes, Rodrigo Fonseca
#se crea un objeto de simulación
set ns [new Simulator]

$ns color 0 Blue
$ns color 1 Red

#se crea un arreglo de parámetros para la LAN
set lan1(bw)  100Mb
set lan1(delay)        0.300ms
```

```
set lan1(ll)    LL
set lan1(ifq)   Queue/DropTail
set lan1(mac)       Mac/802_3
set lan1(chan)      Channel

set numClients 16
#se crean los archivos de salida
set f0 [open oute0.tr w]
set f1 [open oute1.tr w]
set f2 [open oute2.tr w]
set f3 [open oute3.tr w]
set f4 [open oute4.tr w]
set f5 [open oute5.tr w]
set f6 [open oute6.tr w]
set f7 [open oute7.tr w]
set f8 [open oute8.tr w]
set f9 [open oute9.tr w]
set f10 [open oute10.tr w]
set f11 [open oute11.tr w]
set f12 [open oute12.tr w]
set f13 [open oute13.tr w]
set f14 [open oute14.tr w]

set nf [open lan1.nam w]
$ns namtrace-all $nf

#se crean 16 nodos
for {set i 0} {$i < ($numClients)} {incr i} {
        set node($i) [$ns node]
        #puts "Direccion de $node($i) :  node-addr"
        $node($i) color blue
        lappend arrayNodesLan1 $node($i)
}
```

```
$node([expr $numClients - 1]) shape box
$node([expr $numClients - 1]) color green

puts "creando Lan1.."
set miLan1 [$ns make-lan $arrayNodesLan1  $lan1(bw) $lan1(delay) \
                        $lan1(ll) $lan1(ifq) \
                        $lan1(mac) $lan1(chan)]

#$ns duplex-link $node(0) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(1) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(2) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(3) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(4) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(5) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(6) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(7) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(8) $node([expr $numClients - 1]) 100Mb 1ms DropTail
#$ns duplex-link $node(9) $node([expr $numClients - 1]) 100Mb 1ms DropTail

#se crea los sink de tráfico uno por cada cliente y lo adjuntamos al último nodo

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
set sink3 [new Agent/LossMonitor]
set sink4 [new Agent/LossMonitor]
set sink5 [new Agent/LossMonitor]
set sink6 [new Agent/LossMonitor]
set sink7 [new Agent/LossMonitor]
set sink8 [new Agent/LossMonitor]
set sink9 [new Agent/LossMonitor]
set sink10 [new Agent/LossMonitor]
```

```
set sink11 [new Agent/LossMonitor]
set sink12 [new Agent/LossMonitor]
set sink13 [new Agent/LossMonitor]
set sink14 [new Agent/LossMonitor]

$ns attach-agent $node([expr $numClients - 1]) $sink0
$ns attach-agent $node([expr $numClients - 1]) $sink1
$ns attach-agent $node([expr $numClients - 1]) $sink2
$ns attach-agent $node([expr $numClients - 1]) $sink3
$ns attach-agent $node([expr $numClients - 1]) $sink4
$ns attach-agent $node([expr $numClients - 1]) $sink5
$ns attach-agent $node([expr $numClients - 1]) $sink6
$ns attach-agent $node([expr $numClients - 1]) $sink7
$ns attach-agent $node([expr $numClients - 1]) $sink8
$ns attach-agent $node([expr $numClients - 1]) $sink9
$ns attach-agent $node([expr $numClients - 1]) $sink10
$ns attach-agent $node([expr $numClients - 1]) $sink11
$ns attach-agent $node([expr $numClients - 1]) $sink12
$ns attach-agent $node([expr $numClients - 1]) $sink13
$ns attach-agent $node([expr $numClients - 1]) $sink14

#Se define un procedimiento de finalización de la simulación
proc finish {} {
        global  ns f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13 f14 nf
        #Close the output files
        close $f0
        close $f1
        close $f2
        close $f3
        close $f4
        close $f5
        close $f6
        close $f7
```

```
        close $f8
        close $f9
        close $f10
        close $f11
        close $f12
        close $f13
        close $f14

        $ns flush-trace
        close $nf
        #Call xgraph to display the results
      exec nam lan1.nam &
      exec xgraph oute0.tr oute1.tr oute2.tr oute3.tr oute4.tr oute5.tr oute6.tr oute7.tr oute8.tr oute9.tr oute10.tr oute11.tr oute12.tr oute13.tr oute14.tr -geometry 800x400 &
        exit 0
}

#Se define un procedimiento (attach-const-traffic) que adjunta un agente UDP a un nodo y un agente #generador de tráfico CBR al agente de transporte UDP, el procedimiento conecta #el sumidero de tráfico a la fuente y retorna el objeto de tráfico

proc attach-const-traffic { node sink size rate } {

        set ns [Simulator instance]

        #Creo un Agente UDP y lo adjunto al nodo
        set source [new Agent/UDP]
        $ns attach-agent $node $source

        #Creo un agente de Trafico CBR
        set traffic [new Application/Traffic/CBR]
```

```
        $traffic set type_ CBR
        $traffic set packet_size_ $size
        $traffic set rate_ $rate
        $traffic set random_ 1
        #se adjunta la fuente de tráfico al generador de tráfico
       $traffic attach-agent $source

        #se conecta la fuente y el sink
        ns connect $source $sink
        return $traffic
}

#Creamos las fuentes de tráfico UDP
        set source0 [attach-const-traffic $node(0) $sink0 1470 10.0Mb]
        set source1 [attach-const-traffic $node(1) $sink1 1470 10.0Mb]
        set source2 [attach-const-traffic $node(2) $sink2 1470 10.0Mb]
        set source3 [attach-const-traffic $node(3) $sink3 1470 10.0Mb]
        set source4 [attach-const-traffic $node(4) $sink4 1470 10.0Mb]
        set source5 [attach-const-traffic $node(5) $sink5 1470 10.0Mb]
        set source6 [attach-const-traffic $node(6) $sink6 1470 10.0Mb]
        set source7 [attach-const-traffic $node(7) $sink7 1470 10.0Mb]
        set source8 [attach-const-traffic $node(8) $sink8 1470 10.0Mb]
        set source9 [attach-const-traffic $node(9) $sink9 1470 10.0Mb]
        set source10 [attach-const-traffic $node(10) $sink10 1470 10.0Mb]
        set source11 [attach-const-traffic $node(11) $sink11 1470 10.0Mb]
        set source12 [attach-const-traffic $node(12) $sink12 1470 10.0Mb]
        set source13 [attach-const-traffic $node(13) $sink13 1470 10.0Mb]
        set source14 [attach-const-traffic $node(14) $sink14 1470 10.0Mb]

#Se define un procedimiento (record) que periodicamente registra los valores del
#indicador de red que se haya calculado, el cálculo se realiza con datos
#provenientes del sumidero de tráfico
```

```
proc record {} {
        global sink0 sink1 sink2 sink3 sink4 sink5 sink6 sink7 sink8 sink9 sink10
        sink11 sink12 sink13 sink14 f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13 f14

        set ns [Simulator instance]
        #se asigna el tiempo despues del cual el procedimiento será llamado de
        #nuevo
        set time 0.5
        #se asignan los bytes recibidos por los sumideros de tráfico}
        set bw0 [$sink0 set bytes_]
        set bw1 [$sink1 set bytes_]
        set bw2 [$sink2 set bytes_]
        set bw3 [$sink3 set bytes_]
        set bw4 [$sink4 set bytes_]
        set bw5 [$sink5 set bytes_]
        set bw6 [$sink6 set bytes_]
        set bw7 [$sink7 set bytes_]
        set bw8 [$sink8 set bytes_]
        set bw9 [$sink9 set bytes_]
        set bw10 [$sink10 set bytes_]
        set bw11 [$sink11 set bytes_]
        set bw12 [$sink12 set bytes_]
        set bw13 [$sink13 set bytes_]
        set bw14 [$sink14 set bytes_]

        #se asignan el número de paquetes recibidos y el número de paquetes
        #perdidos
        set numP0 [$sink0 set npkts_]
        set numPLost0 [$sink0 set nlost_]
        set numP1 [$sink1 set npkts_]
        set numPLost1 [$sink1 set nlost_]
        set numP2 [$sink2 set npkts_]
        set numPLost2 [$sink2 set nlost_]
```

```
set numP3 [$sink3 set npkts_]
set numPLost3 [$sink3 set nlost_]
set numP4 [$sink4 set npkts_]
set numPLost4 [$sink4 set nlost_]
set numP5 [$sink5 set npkts_]
set numPLost5 [$sink5 set nlost_]
set numP6 [$sink6 set npkts_]
set numPLost6 [$sink6 set nlost_]
set numP7 [$sink7 set npkts_]
set numPLost7 [$sink7 set nlost_]
set numP8 [$sink8 set npkts_]
set numPLost8 [$sink8 set nlost_]
set numP9 [$sink9 set npkts_]
set numPLost9 [$sink9 set nlost_]
set numP10 [$sink10 set npkts_]
set numPLost10 [$sink10 set nlost_]
set numP11 [$sink11 set npkts_]
set numPLost11 [$sink11 set nlost_]
set numP12 [$sink12 set npkts_]
set numPLost12 [$sink12 set nlost_]
set numP13 [$sink13 set npkts_]
set numPLost13 [$sink13 set nlost_]
set numP14 [$sink14 set npkts_]
set numPLost14 [$sink14 set nlost_]
#Se toma el tiempo actual
set now [$ns now]
puts $f0 "$numPLost0 / $numP0"
puts $f1 "$numPLost1 / $numP1"
puts $f2 "$numPLost2 / $numP2"
puts $f3 "$numPLost3 / $numP3"
puts $f4 "$numPLost4 / $numP4"
puts $f5 "$numPLost5 / $numP5"
puts $f6 "$numPLost6 / $numP6"
```

```
puts $f7 "$numPLost7 / $numP7"
puts $f8 "$numPLost8 / $numP8"
puts $f9 "$numPLost9 / $numP9"
puts $f10 "$numPLost10 / $numP10"
puts $f11 "$numPLost11 / $numP11"
puts $f12 "$numPLost12 / $numP12"
puts $f13 "$numPLost13 / $numP13"
puts $f14 "$numPLost14 / $numP14"

#se calcula el ancho de banda en Mbps y se lo escribe en los archivos
#puts $f0 "$now [expr $bw0/$time*8/1048576]"
#puts $f1 "$now [expr $bw1/$time*8/1048576]"
#puts $f2 "$now [expr $bw2/$time*8/1048576]"
#puts $f3 "$now [expr $bw3/$time*8/1048576]"
#puts $f4 "$now [expr $bw4/$time*8/1048576]"
#puts $f5 "$now [expr $bw5/$time*8/1048576]"
#puts $f6 "$now [expr $bw6/$time*8/1048576]"
#puts $f7 "$now [expr $bw7/$time*8/1048576]"
#puts $f8 "$now [expr $bw8/$time*8/1048576]"
#puts $f9 "$now [expr $bw9/$time*8/1048576]"
#puts $f10 "$now [expr $bw10/$time*8/1048576]"
#puts $f11 "$now [expr $bw11/$time*8/1048576]"
#puts $f12 "$now [expr $bw12/$time*8/1048576]"
#puts $f13 "$now [expr $bw13/$time*8/1048576]"
#puts $f14 "$now [expr $bw14/$time*8/1048576]"

#Se enceran las variables que guardan los paquetes generados y los
#paquetes perdidos
$sink0 set nlost_ 0
$sink0 set npkts_ 0
$sink1 set nlost_ 0
$sink1 set npkts_ 0
$sink2 set nlost_ 0
```

```
$sink2 set npkts_ 0
$sink3 set nlost_ 0
$sink3 set npkts_ 0
$sink4 set nlost_ 0
$sink4 set npkts_ 0
$sink5 set nlost_ 0
$sink5 set npkts_ 0
$sink6 set nlost_ 0
$sink6 set npkts_ 0
$sink7 set nlost_ 0
$sink7 set npkts_ 0
$sink8 set nlost_ 0
$sink8 set npkts_ 0
$sink9 set nlost_ 0
$sink9 set npkts_ 0
$sink10 set nlost_ 0
$sink10 set npkts_ 0
$sink11 set nlost_ 0
$sink11 set npkts_ 0
$sink12 set nlost_ 0
$sink12 set npkts_ 0
$sink13 set nlost_ 0
$sink13 set npkts_ 0
$sink14 set nlost_ 0
$sink14 set npkts_ 0

$sink0 set bytes_ 0
$sink1 set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0
$sink4 set bytes_ 0
$sink5 set bytes_ 0
$sink6 set bytes_ 0
```

```
        $sink7 set bytes_ 0
        $sink8 set bytes_ 0
        $sink9 set bytes_ 0
        $sink10 set bytes_ 0
        $sink11 set bytes_ 0
        $sink12 set bytes_ 0
        $sink13 set bytes_ 0
        $sink14 set bytes_ 0


        #Se reprograma el procedimientos
        $ns at [expr $now+$time] "record"
}



$ns at 0.0 "record"
#Se inician las Fuentes de tráfico
        $ns at 00.0 "$source0 start"
        $ns at 00.0 "$source1 start"
        $ns at 00.0 "$source2 start"
        $ns at 00.0 "$source3 start"
        $ns at 00.0 "$source4 start"
        $ns at 00.0 "$source5 start"
        $ns at 00.0 "$source6 start"
        $ns at 00.0 "$source7 start"
        $ns at 00.0 "$source8 start"
        $ns at 00.0 "$source9 start"
        $ns at 00.0 "$source10 start"
        $ns at 00.0 "$source11 start"
        $ns at 00.0 "$source12 start"
        $ns at 00.0 "$source13 start"
        $ns at 00.0 "$source14 start"
```

```
        #Se detienen las Fuentes de tráfico
        $ns at 60.0 "$source0 stop"
        $ns at 60.0 "$source1 stop"
        $ns at 60.0 "$source2 stop"
        $ns at 60.0 "$source3 stop"
        $ns at 60.0 "$source4 stop"
        $ns at 60.0 "$source5 stop"
        $ns at 60.0 "$source6 stop"
        $ns at 60.0 "$source7 stop"
        $ns at 60.0 "$source8 stop"
        $ns at 60.0 "$source9 stop"
        $ns at 60.0 "$source10 stop"
        $ns at 60.0 "$source11 stop"
        $ns at 60.0 "$source12 stop"
        $ns at 60.0 "$source13 stop"
        $ns at 60.0 "$source14 stop"

#Se llama al procedimiento de finalización después de 60 segundos de simulación
$ns at 60.0 "finish"

#Se ejecuta la simulación
$ns run
```

REFERENCIAS

[1] J. N. Matthews, W. Hu, M. Hapuarachchi, T. Deshane, D. Dimatos, G. Hamilton, M. McCabe, J. Owens, “Quantifying the Performance Isolation Properties of Virtualization Systems”. In Proc. of ExpCS’07, 13–14 June, 2007, San Diego, CA.

[2] W. M. Fuertes and J. E. López de Vergara, “A quantitative comparison of virtual network environments based on performance measurements”, in Proceedings of the 14th HP Software University Association Workshop, Garching, Munich, Germany, 8-11 July 2007.

[3] J. Walters, V. Chaudhary, M. Cha, S. Guercio Jr, S. Gallo, “A Comparison of Virtualization Technologies for HPC”. In Proc. of 22nd International Conference on Advanced Information Networking and Applications (AINA) 2008, Okinawa 25-28 March 2008, pp. 861-868

[4] R. Figueiredo, P. Dinda, J. Fortes, “Resource Virtualization Renaissance”, IEEE Computer, Vol. 38, Issue 5, May 2005, pp. 28 – 31.

[5] W. Fuertes, J. E. López de Vergara, “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. Septiembre de 2008.

[6] M. Tim Jones, “An overview of virtualization methods, architectures, and implementations”. Emulex Corp. Longmont, Colorado, 29 December 2006.

[7] SCOPE Alliance, “Virtualization: State of the Art”. Version 1.0, April 3, 2008. Copyright © 2008.

[8] Xen, Web 2009 [Online] http://www.xen.org/

[9] P. Barham, B. Dragovic, K. Fraser, S. H, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield. “Xen and the Art of Virtualization”. In Proc. of 19th ACM symposium on Operating systems principles. Bolton Landing, NY, USA, 2003. pp. 164 – 177.

[10] The Network Simulator ns2, [Online] http://www.isi.edu/nsnam/ns, Last verification, Oct. 2008.

[11] Iperf http://dast.nlanr.net/Projects/Iperf/

[12] Netperf. [Online]http://www.netperf.org/netperf/NetperfPage.html,Web 2009.

[13] S. Hemminger: “Network Emulation with NetEm”. Open Source Development Lab. April 2005.

[14] R. J. Creasy, “The Origin of the VM/370 Time-sharing System”, IBM J. RES. DEVELOP. VOL. 25 NO. 5 SEPTEMBER 1981.

[15] A. Muñoz, “Performance analysis in accessing web services”, [Online] http://det.bi.ehu.es/NQAS/opnet/

[16] V. Rakocevic, R. Stewart, R. Flynn, “VoIP Network Dimensionaing using Delay and Loss Bounds for for Voice and Data Applications”. Techincal Report.

[17] G. Flores, M. Paredes, E. Jammeh, M. Fleury, M. Reed, M. Ghanbari, “Packet by Packet Analysis in Contemporary Network Simulators”. IEEE Latin America Transactions, vol. 4, no. 4, pps: 299-307, June 2006.

[18] K. Lakshminarayanan, V. Padmanabhan, J. Padhye, “Bandwidth Estimation in Broadband Access Networks”, Microsoft Research-Technical Report, MSR-TR-2004-44, IMC’04, Taormina, Sicily, Italy. Copyright 2004 ACM, October 25–27, 2004.

[19] A. Grau, H. Weinschrott, C. Schwarzer: “Evaluating the Scalability of Virtual Machines for Use in Computer Network Emulation”. Stuttgart University, Oct/06.
[20] W. Fuertes and J. E. López de Vergara, “An emulation of VoD services using virtual network environments”. In Proc. GI/ITG Workshop on Overlay and Network Virtualization NVWS'09, Kassel-Germany, March 2009.
[21] W. Fuertes y J. E. López de Vergara, “Evaluación de plataformas Virtuales para realizar experimentos de medidas en redes IP”, Madrid 2007.
[22] David Bello, “Uso de maquinas virtuales en las enseñanzas de las TIC”, Educación y Futuro Digital, 17 de Abril 2009.
[23] Maquina Virtual,[Online]http://www.vmware.com/es/overview/vmachine.html
[24] Jose Miguel Herrera M. “NS2 - Network Simulator”. Valparaíso, 12 de mayo de 2004.
[25] R.E.Shannon,[Online]http://www.uamcav.uat.edu.mx/persona/documentos/2%2Fsimu/Introducci%F3n_Simulacion.pdf
[26] Paravirtualizacion, [Online],http://www.xtech.com.ar/articulos/xen/articulo-xen-coletti-html/x18.html
[27] Dalibor Dobrilović, and Borislav Odadžić, “Virtualization Technology as a Tool for eaching Computer Networks”, World Academy of Science, Engineering and Technology 13 2006.
[28] Honeypot, [Online],http://es.wikipedia.org/wiki/Honeypot, Last modification, 23 sep 2009.
[29] VMWare, [Online], http://es.wikipedia.org/wiki/VMware, Last modification, 29 oct 2009.
[30] VirtualBox, [Online], http://es.wikipedia.org/wiki/VirtualBox, Last modification, 30 oct 2009.
[31] OpenVZ, [Online], http://es.wikipedia.org/wiki/OpenVZ, Last modification, 3 nov 2009
[32] Teerawat Issariyakul and Ekram Hossain. “Introduction to Network Simulator NS-2”. ISBN 987-0-387-71759-3, Springer 2009.
[33] Mgen, [Online], http://cs.itd.nrl.navy.mil/work/mgen/
[34] Netperf, [Online], http://www.netperf.org
[35] CDF Cummulative Distribution Function o Función de Distribución Acumulada,[Online], http://colposfesz.galeon.com/est501/probabi/teo/cap402/cap402.htm

ABREVIATURAS Y ACRÓNIMOS

C

CDF *Cumulative Distributiion Function*, Función de Distribución Acumulada

CPU *Central Processing Unit*, Unidad Central de Procesamiento

G

GNU GPL *General Public License*, Licenciamiento público general

K

KVM *Kernel Based Virtual Machine*, Máquina Virtual Basada en Kernel

L

LAN *Local Area Network*, Red de área local

M

MGEN *Multi Generador Tool Set*, Conjunto de herramientas de Generación Múltiple

N

NAM *Network Animator*, Animador de Red

NS *Network Simulator*, Simulador de Red

S

SO Sistema Operativo.

T

TCP *Transmition Control Protocol*, Protocol de Control de Transmisiones

U

UML *User Mode Linux*, Modo de Usuario Linux

V

VM *Virtual Machines*, Máquinas Virtuales.

W

WAN *Wide Area Network*, Red de Area Extensa

BIOGRAFÍA. María Fernanda Grijalva Suárez

CÉDULA DE IDENTIDAD: 171518136 – 6
EMAIL: grijalvasmf@espe.edu.ec, mafergs2184@gmail.com
TELEFONO DE CONTACTO: 2323789 / 084081613
LUGAR Y FECHA DE NACIMIENTO: Quito, 21 de Agosto de 1984
LUGAR ACTUAL DE RESIDENCIA: Quito - Ecuador

ESTUDIOS:

INSTITUCIÓN: Unidad Educativa Particular “Giovanni Antonio Farina”
TÍTULO: Bachiller en Ciencias, especialidad Físico Matemático
AÑO: 2002

INSTITUCIÓN: Escuela Politécnica del Ejército “ESPE”
TÍTULO: Ingeniera de Sistemas de Computación e Informática
AÑO: 2009
PROYECTO DE GRADO:
“Evaluación del rendimiento de redes IP utilizando plataformas de Virtualización y métodos de Simulación” Proyecto de Iniciación Científica del departamento Ciencias de la Computación de la Escuela Politécnica del Ejército del Ecuador, para realizar la formación de Ingeniero en Sistemas e Informática, duración 1 año.

BIOGRAFÍA. Darwin Leonardo Jácome Moreno

CÉDULA DE IDENTIDAD: 172064794 – 8
EMAIL: jacomemdl@espe.edu.ec, leonardojacome@gmail.com
LUGAR Y FECHA DE NACIMIENTO: Quito, 16 de Septiembre de 1985
LUGAR ACTUAL DE RESIDENCIA: Quito - Ecuador

ESTUDIOS:

INSTITUCIÓN:
TÍTULO: Técnico en Comercio y Administración especialidad Informática
AÑO: 2003

INSTITUCIÓN: Escuela Politécnica del Ejército “ESPE”
TÍTULO: Ingeniera de Sistemas de Computación e Informática
AÑO: 2009
PROYECTO DE GRADO:
“Evaluación del rendimiento de redes IP utilizando plataformas de Virtualización y métodos de Simulación” Proyecto de Iniciación Científica del departamento Ciencias de la Computación de la Escuela Politécnica del Ejército del Ecuador, para realizar la formación de Ingeniero en Sistemas e Informática , duración 1 año.

HOJA DE LEGALIZACION DE FIRMAS

ELABORADA POR

MARÍA FERNANDA GRIJALVA SUÁREZ
Y DARWIN LEONARDO JÁCOME MORENO

COORDINADOR DE LA CARRERA

ING. DANILO MARTINEZ

Lugar y fecha: Sangolquí, 11 de Noviembre del 2009