Universidad Autónoma de Madrid

Escuela Politécnica Superior

Programa de Doctorado en Ingeniería en Informática y de Telecomunicación

TESIS DOCTORAL

DEFINICIÓN DE UN MODELO GENÉRICO PARA LA CARACTERIZACIÓN DE ESCENARIOS VIRTUALES DE REDES IP

Autor

Walter Marcelo Fuertes Díaz

MSc. Ingeniero de Sistemas e Informática.

Director

Jorge Enrique López de Vergara Méndez

Doctor Ingeniero de Telecomunicación

Madrid, 2010

Título: Definición de un Modelo Genérico para la Caracterización de Escenarios Virtuales de Redes IP

Autor: Walter Marcelo Fuertes Díaz

Director: Jorge Enrique López de Vergara Méndez

TRIBUNAL CALIFICADOR

Presidente:

☐

Vocales:

☐

☐

☐

Secretario:

☐

Realizado el acto de defensa y lectura de Tesis en Madrid, el día de de 2010.

CALIFICACIÓN:

El presidente **Los Vocales**

El secretario

"Con mucha razón se sostiene que el desarrollo de una sociedad se encuentra íntimamente ligado al accionar de sus universidades; siendo esto así, los centros de educación superior tienen una gran responsabilidad, lo que obliga a los mismos, entre otros, para alcanzar sus objetivos, a contar con una planta docente con un alto nivel de preparación académica y científica, promoviendo así la generación de conocimiento y el desarrollo de tecnologías. Para ello y en el entendido de que la docencia universitaria, es una dignidad, el docente debe investigar y prepararse todo el tiempo, situación ésta que conlleva una gran vocación y concomitantemente con ello, perseverancia constante, haciendo del sacrifico que demanda esta noble actividad, una gran realización y satisfacción personal; solo así se alcanzarán las metas previamente establecidas, lo cual redundará en beneficio de la sociedad a la que nos debemos."

Dr. Juan Carlos Orbe
Secretario General de la ESPE.

Agradecimientos

En primer lugar, debo dar gracias a Dios por su bondad, a la **ESCUELA POLITÉCNICA DEL EJÉRCITO DEL ECUADOR (ESPE)** por brindarme la oportunidad de buscar la calificación como Ph.D. y a la **UNIVERSIDAD AUTÓNOMA DE MADRID-ESPAÑA (UAM)**, por ser el templo del saber, en dónde desarrollé sólidos conocimientos para obtener este tan anhelado grado.

En segundo lugar, quiero agradecer al director de mi Tesis Doctoral, al Doctor Jorge Enrique López de Vergara Méndez, por su gran calidad profesional y humana, por su encomiable guía, orientación, impulso, valiosa y responsable colaboración y su apoyo determinante en la consecución de los objetivos, que ha hecho posible la cristalización de esta investigación.

Deseo además expresar mi gratitud a los miembros del **Grupo de Investigación de Redes de la UAM**, Doctores Javier Aracíl, José A. Hernández, Sergio López-Buedo, Juan González, Víctor López, Bass Huiszoon, Angel de Castro, por su ejemplo, apoyo y su amistad. Así mismo debo agradecer a los doctorandos José L. García, Felipe Mata, José L. Añamuro, Carlos Fadón, Fernando Herrero y Alfredo Salvador, con quienes compartí momentos de franca armonía y constante sacrificio. De igual forma a los investigadores del **Lab-B-209**, de las hermanas repúblicas de Argentina, Colombia y España, por sus importantes consejos durante mi estadía en la UAM, y porque ese laboratorio fue el espacio de concentración, inspiración, abnegación y aprendizaje.

Cúmpleme expresar mi reconocimiento a los Doctores: Fermín Galán y David Fernández, quienes me brindaron la oportunidad de participar y concretar algunas publicaciones y quienes aportaron en mi formación como investigador. Quiero incluir además a los Ings. Fausto Meneses y Hernán Aules por su importantísimo apoyo al final del desarrollo de mi Tesis Doctoral.

Finalmente, es importante reconocer el apoyo y confianza que en su día las autoridades de la **ESPE** tuvieron en mí, como el señor Crnl. José Núñez M. (Ex Rector), Tcrn. Marco Quintana (Ex Decano FISI), Ing. Lauro López (Director de Evaluación), Dr. Juan C. Orbe (Secretario General), Dr. Alfredo Suquilanda (Director de Relaciones Internacionales), Ing. Lourdes de la Cruz (Director UGI). Así mismo vaya mi reconocimiento a las autoridades actuales: señor Gral. Rubén Navia L. (Rector), Crnl. Carlos Rodríguez (Vicerrector Académico), Crnl. Rodolfo Salazar (Vicerrector de Investigaciones) y Crnl. Wilson Sánchez, Director DECC. A todos ustedes vaya mi profundo agradecimiento por esta gentil deferencia y por ese respaldo total para esta consecución.

Con el mayor afecto por siempre al pueblo hermano de España quien nos acogió a los míos y a mí durante el tiempo de permanencia en la Madre Patria.

Resumen

La presente Tesis Doctoral aborda la aplicación de las Tecnologías de Virtualización como un medio de investigación en redes IP. El principal objetivo es el diseño, creación y validación de un modelo genérico que caracterice los entornos virtuales de red, independientemente de la plataforma de virtualización; toda vez que el diseño y puesta en funcionamiento de estos entornos es un proceso complejo. Este objetivo ha sido materializado a través de tres enfoques: el modelo analítico, el modelo de infraestructura de despliegue y el modelo de gestión de la configuración.

Desde el punto de vista analítico, se ha realizado una evaluación cuantitativa de las plataformas de virtualización cara a determinar qué plataforma provee mejores prestaciones y un rendimiento cercano al nativo. Con esta evaluación, se ha emulado el servicio de Video bajo Demanda utilizando la plataforma seleccionada. Además, puesto que la capa de virtualización genera una penalización, se ha medido el overhead del rendimiento en estos entornos, con la decisión de predecir dicha penalización en un experimento dado a posteriori. Así mismo se han aplicado técnicas estadísticas para determinar qué variables afectan el rendimiento, incrementando el overhead.

Desde el punto de vista del modelo de infraestructura del despliegue, se han establecido algunas propuestas a fin de definir una arquitectura base que permita el despliegue automático de escenarios virtuales de red en entornos distribuidos, infraestructura que posteriormente va a ser modelada. En este propósito se implementó una interfaz de Servicios Web para el despliegue de entornos virtuales y posteriormente se ha modelado el despliegue de un escenario virtual basado en Tecnologías Grid.

Desde el punto de vista de gestión de la configuración, la Tesis se ha enfocado en especificar un entorno virtual de red, mediante técnicas de modelado de información a fin de resolver los problemas asociados con la interoperabilidad y la gestión de recursos virtuales. Con este propósito, se analizaron los enfoques existentes para modelar las infraestructuras virtualizadas. Basándose en este análisis, se diseñó y construyó un modelo genérico basado en el Modelo de Información Común de la DMTF. Para evaluar la viabilidad de este modelo, se implementó un sistema de gestión, que permitió desplegar entornos de red virtuales de forma automática. Los resultados de las pruebas demostraron la eficacia de esta investigación, que fue evaluada con Xen y VMware Server.

Palabras clave: Análisis de Varianza, despliegue, emulación, escenario de red, Grid, herramientas de virtualización, máquina virtual, modelo analítico, modelado de información, modelado de infraestructuras virtualizadas, overhead, perfil de virtualización, prestación de servicios, rendimiento, Regresión Lineal, Servicios Web, simulación, sistema de gestión.

Abstract

This Ph.D. Thesis addresses the application of virtualization technologies as a means of experimentation and research in IP networks. The main objective is to design, develop and validate a generic model that characterizes the virtual network environments, independently of the virtualization platform, given its difficulty of design and complex operation. This objective has been materialized through three viewpoints: the analytical model, the deployment infrastructure model, and the management configuration model.

From the analytic point of view, we carried out a quantitative evaluation of virtualization platforms with the aim to determine which platform provides a better performance very close to native. With this initial assessment, a set of experiments has been carried out to emulate a Video on Demand service using the virtualization platform selected. Furthermore, since the virtualization layer creates an overhead, we have measured the performance overhead in virtualized environments in order to predict this overhead in a given experiments. We also applied statistical techniques to determine which variables affect the performance, increasing the overhead.

From the deployment infrastructure model point of view, we established some proposals to define a basic architecture to enable the automatic deployment of virtual network topologies in distributed environments, infrastructure that is going to be modeled later. With this purpose, we set up a Web Services interface for the deployment of virtual network environment. Finally, the deployment of a virtual network environment based on Grid technologies has been modeled.

From the management configuration point of view, the thesis focused on the use of information modeling techniques to model a virtual network environment aiming to solve the problems associated with interoperability and management of virtual resources. With this purpose, first, we analyzed existing approaches for modeling virtualized infrastructures. Based on this analysis, we designed and implemented a generic model for the characterization and management of virtual network environments based on DMTF's Common Information Model. To assess the feasibility of this model, we implemented management system, which makes viable the deployment of virtual network environments automatically, regardless of the used virtualization platform. The results of tests showed the effectiveness of this application, which was assessed with Xen and VMware Server.

Keywords: analytical model, Analysis of Variance, deployment, emulation, Grid, network environment, information modeling, Linear Regression, modeling of virtualized infrastructure, overhead, profile of virtualization, service delivery, performance, simulation, management system, virtualization tools, virtual machine, Web services.

A mi amada esposa Silvia,

A mis pequeños hijos Walter y Mateo por ser el motivo de inspiración.

A mis padres y hermanos

A la inmensidad del amor de Dios por otorgarme luz, entendimiento, perseverancia, salud, mi familia y la vida.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción y objetivos

1.1 Introducción y motivación

De acuerdo con Gartner Inc. [Gartner08], la Tecnología de Virtualización es actualmente una de las diez tendencias disruptivas, que está modificando el funcionamiento de los centros de procesamiento de datos, redefinirá esquemas tecnológicos, departamentos de TI y rediseñará la industria de micros y servidores. Estas tecnologías pueden ser usadas para crear entornos de validación y pruebas de software [Mathews07]. Además, estas plataformas pueden ser utilizadas para la consolidación de servidores, pruebas de balanceo de carga y dimensionado de servicios de red [Jones06].

La virtualización en esencia es una estrategia para compartir recursos. Puede ser usada para particionar un equipo físico y soportar múltiples máquinas virtuales (MVs, *virtual machines*), interconectadas entre sí para asignar recursos de hardware como CPU, memoria y dispositivos de entrada y salida [Hart-Sears07]. Es una técnica que permite, vía hardware, disponer de varios sistemas operativos (OS, *Operating System*) hospedados del mismo o diverso tipo en cada MV.

En la industria han sido diseñadas numerosas plataformas de virtualización que tienen sus características específicas en función de la técnica de virtualización a la que pertenecen [Scope08]. Así, unas requieren de hardware actualizado (*virtualización nativa*). Otras apuntan a la compatibilidad consumiendo masivamente los recursos computacionales, deteriorando el rendimiento del equipo anfitrión (*virtualización completa*). Algunas sacrifican originalidad y compatibilidad (*paravirtualización*), pero se acercan al rendimiento computacional nativo. Otras trabajan como procesos del OS, pero ante la falla de una MV, afectan a toda la topología arriesgando la continuidad del experimento (*virtualización a nivel de OS*). En cualquier caso para asegurar el rendimiento aceptable y una plataforma confiable, se debe proveer el hardware robusto necesario y la tecnología de virtualización adecuada, para permitir que las MVs operen eficientemente [Fernández06].

En la presente Tesis Doctoral, y como cuestionamiento científico, se ha direccionado la aplicación de las Tecnologías de Virtualización como un medio de experimentación e investigación en redes IP. Luego de algunos resultados, se puede corroborar que estas plataformas son una alternativa de despliegue de redes IP mediante la implementación de Entornos Virtuales de Red EVR o VNE, *Virtual Network Environment* (su acrónimo en inglés).

En el ámbito de esta investigación, un EVR es definido como un conjunto de dispositivos virtuales (equipos de computo, servidores, enrutadores y puentes) conectados colectivamente en

una topología dada, que puede ser desplegada en uno o varios equipos anfitriones (hosts), que emula un sistema equivalente y que se percibe como si fuera un entorno real [Fuertes09a].

Sin embargo, las Tecnologías de Virtualización generan una penalización del rendimiento causada por la capa de virtualización (*overhead* de aquí en adelante) [Deshane06] [Cherkasova05] [Wood08], lo que reduce la precisión de los resultados experimentales y por lo tanto limita la credibilidad en su aplicación. Por otro lado existen varios desafíos en torno a la aplicación de estas tecnologías, como son la interoperabilidad, la complejidad en el despliegue y la gestión de recursos virtuales [Grau06] [Vyatta07] [Wlodarz07].

Para dar respuesta a los problemas planteados, la presente Tesis Doctoral plantea la definición de un modelo genérico que caracterice escenarios virtuales de redes IP desde tres puntos de vista: el modelo analítico, el modelo de infraestructura de despliegue y el modelo de gestión de la configuración. En síntesis, a continuación se describe el procedimiento:

Inicialmente, en esta investigación se estudiaron los fundamentos de la Virtualización, sus técnicas y plataformas. Dada su diversidad, se realizó una evaluación cuantitativa de las mismas, mediante la medición del consumo de CPU y memoria. Luego y como comprobación de la utilidad de los EVR, se realizó la emulación de servicios multimedia en redes IP. De estos experimentos se obtuvo las primeras conclusiones relacionadas con las plataformas que tienen un rendimiento más cercano al nativo, identificándose así mismo las diferentes dificultades al intentar reproducir el funcionamiento de un entorno real y la complejidad en desplegar automáticamente un EVR.

Como otro mecanismo de verificación de los resultados obtenidos en los EVR, se efectúo una evaluación del rendimiento de redes IP utilizando Plataformas de Virtualización comparándola con los resultados entregados mediante métodos de simulación. Puesto que el *overhead* continuaba afectando estos resultados, se procedió a aplicar diversos métodos estadísticos para cuantificar la penalización del rendimiento (*overhead*) en EVR, con el fin de predecir dicha penalización. En este esfuerzo se consiguió obtener por un lado una expresión analítica y por otro un mecanismo que permita establecer que variables afectan más al rendimiento en un EVR dado.

Continuando con la Tesis, se establecieron algunas propuestas para implementar una infraestructura base (infraestructura que posteriormente es modelada) que permita el despliegue automático de escenarios virtuales de red en entornos distribuidos. Primero se implementó una interfaz de Servicios Web para el despliegue de EVR, y posteriormente se modeló una interfaz basada en Servicios Grid [Foster05], integrando Grid y WSRF [Wsrf08].

Con la infraestructura base, y para resolver los problemas relacionados con la interoperabilidad y gestión de recursos asociados a las plataformas de virtualización, se realizó un estudio sobre las técnicas de modelado de información cara a modelar EVR. Basados en este conocimiento, se inició con el diseño y construcción de un modelo genérico para desplegar EVR, que sería validado a

través de la implementación de un sistema de gestión consiguiendo gestionar los recursos virtuales de un EVR, independientemente de la plataforma de virtualización.

1.2 Contexto de la tesis y planteamiento del problema

La tesis doctoral se ha enmarcado en una línea de investigación cuyos temas predominantes son el despliegue de redes de comunicaciones utilizando plataformas de virtualización, cuya línea central converge con la gestión de EVR. En este contexto, a continuación se explican los problemas específicos que han motivado esta investigación:

En **primer lugar**, se ha determinado que existe cierta complejidad tanto en la construcción de la topología de red, como en el despliegue de los EVR. Esto obedece al amplio espectro de plataformas de virtualización en la industria, como son: Imunes [Imunes], KMV [KMV09] , Netkit [Netkit], OpenVz [Open09], Qemu [Qemu], User Mode Linux (UML) [UML], VirtualBox [Vbox10], Virtual Server [Vserv10], VMware [VMware], VNUML [Galán09], y Xen [Barham03], entre las mas conocidas, donde predomina la particularización de su propio código e instrucciones por cada herramienta, es decir la carencia de especificaciones y procedimientos estandarizadas.

En **segundo lugar**, en diversas investigaciones, [Cherkasova05] [Menon05] [Deshane06] [Fernandez06] [Wood08], se ha establecido que la capa de Virtualización genera una sobrecarga u *overhead*, sobre todo cuando se trata de someter un entorno virtual a cargas de trabajo muy grandes o a escenarios de red complejos [Galán07], lo cual implica una degradación del rendimiento de la red virtualizada.

En **tercer lugar**, en investigaciones preliminares [Humphreys06] [Grau06] [Barham03], se ha comprobado que aún existen algunos desafíos, como son la interoperabilidad, la complejidad de automatización y la soportabilidad usando diversas plataformas de virtualización (multiplataforma) e incorporando o combinando ciertos componentes de diversas plataformas (plataformas híbridas).

En **cuarto lugar**, la mayoría de las herramientas de virtualización citadas tienen un enfoque centralizado, es decir el despliegue y administración de los servicios en los escenarios virtuales se realiza solo en un equipo físico, lo cual ocasiona baja escalabilidad y dificulta la creación de escenarios complejos [Galán07] [Vyatta07] [Wlodarz07] en entornos distribuidos.

Para contribuir con soluciones a estos problemas expuestos, esta tesis doctoral presenta cuatro líneas de investigación: **Primera**: Modelado del comportamiento de red para definir un modelo genérico que caracterice escenarios virtuales en redes IP. **Segunda**: Medición del *overhead* en entornos de virtualización. **Tercera**: Definición del comportamiento de los enlaces mediante emulación de redes para optimizar la precisión de las medidas en entornos virtuales. Y **Cuarta**: Definición del comportamiento de servicios dinámicos en entornos distribuidos, para la

formalización de una interfaz genérica que permita desplegar escenarios virtualizados. La Figura 1-1 muestra el contexto de esta Tesis doctoral.

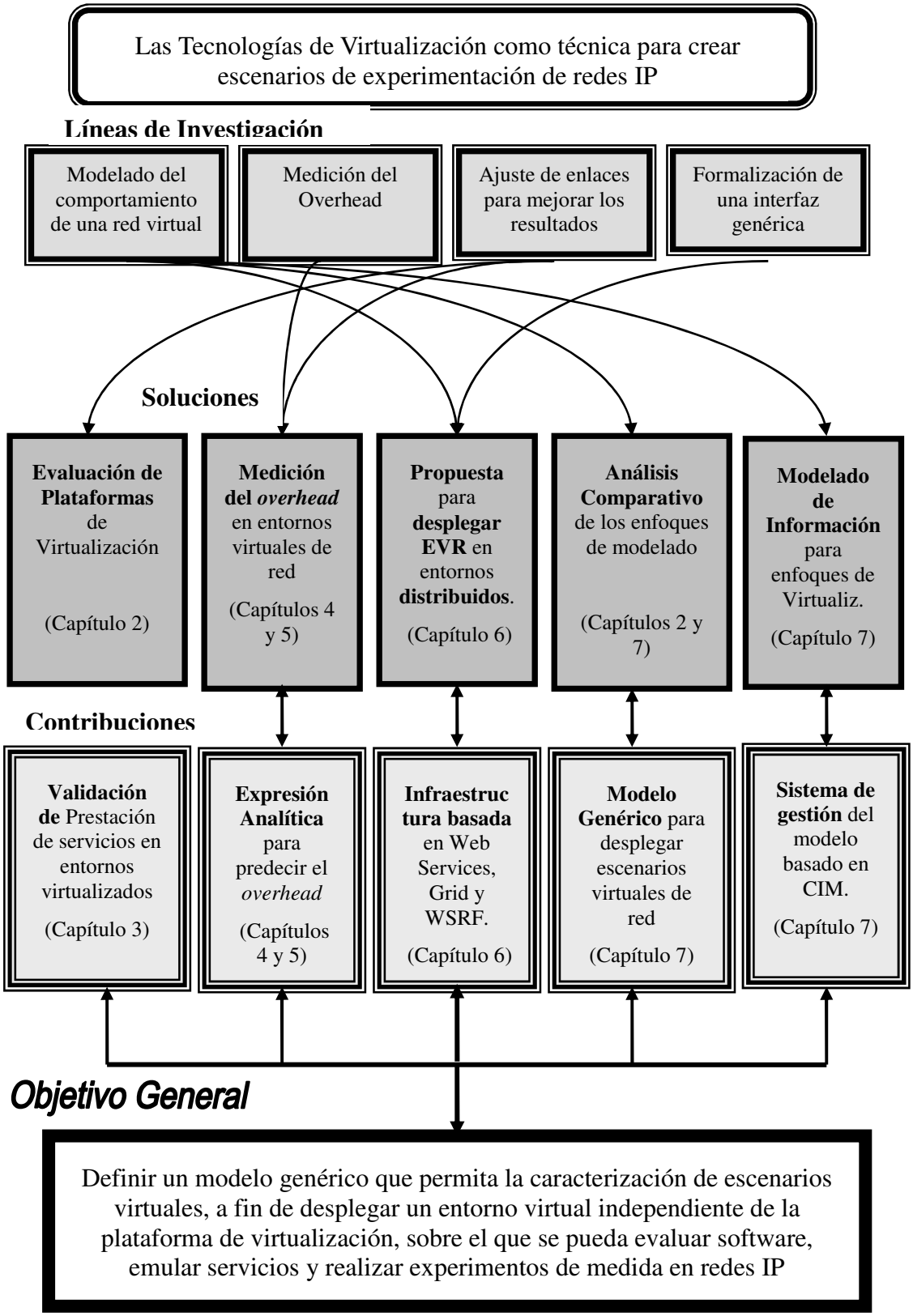

Figura 1-1 Contexto de la Tesis Doctoral.

1.3 Objetivo de la Tesis Doctoral e Hipótesis

El objetivo principal de la Tesis es definir un Modelo Genérico que permita la caracterización de escenarios virtuales, a fin de construir y desplegar un entorno virtual independiente de la plataforma de Virtualización, sobre el que se pueda evaluar software, emular servicios, interactuar con servicios dinámicos en redes distribuidas y realizar experimentos de medida en redes IP, teniendo en consideración los límites y condiciones de la virtualización.

Este objetivo ha sido materializado a través de tres enfoques: el modelo analítico, el modelo de infraestructura de despliegue y el modelo de gestión de la configuración. La Fig. 1-2 que ha sido derivada de las soluciones y contribuciones delimitadas en el contexto de la Tesis (véase Fig. 1-1), ilustra cómo fue abordado cada enfoque durante el desarrollo de la investigación.

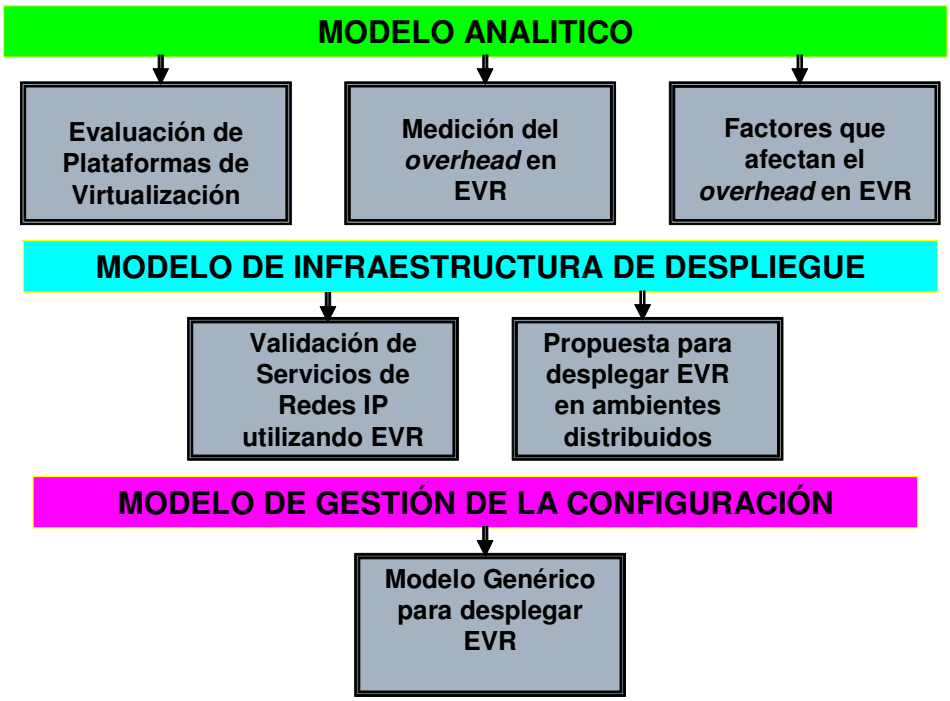

Figura 1-2 Esquema por enfoques para la materializacion del Objetivo General.

Para llevar a cabo este objetivo, cada enfoque ha dado respuesta al Planteamiento del Problema (véase apartado 1.2) que ha originado las siguientes hipótesis de trabajo:

1.3.1 Primera Hipótesis

"Un entorno de red virtualizado puede ser utilizado para emular servicios reales de redes IP".

Esta hipótesis motiva a definir una descripción genérica de la funcionalidad de las redes de computación. Además conduce a verificar y describir el comportamiento de los emuladores de red con la posibilidad de mejorar, e integrar algunas soluciones para conseguir medidas con mayor precisión en entornos virtuales. Esto se pretende cubrir con los siguientes objetivos concretos:

i.) Definir la funcionalidad de la red (topología, número de nodos, tipos de enlace, protocolos, direccionamiento IP, conmutación, enrutamiento, seguridad, etc.) y su integración en un modelo genérico.
ii.) Definir un modelo genérico para caracterizar la funcionalidad de un entorno virtual, independientemente de la herramienta de virtualización.
iii.) Definir el comportamiento de los enlaces simulados mediante herramientas de emulación tales como NetEm, realizar una adaptación para mejorar resultados y su integración en el modelo genérico.

1.3.2 Segunda Hipótesis

"Si las técnicas de Virtualización en un escenario virtual añaden una sobrecarga, entonces los resultados de la experimentación en redes carecen de precisión".

Esta hipótesis conduce a analizar el *overhead* de los escenarios de virtualización y en consecuencia el caracterizar el rendimiento de una red virtual, con el fin de dilucidar bajo qué condiciones y parámetros se presenta el *overhead*. De igual manera conduce a analizar cuáles son los límites de la virtualización y con qué técnicas se podría disminuir este efecto. Esto se pretende cubrir con los siguientes objetivos concretos:

i.) Analizar la forma de asignación de recursos por parte del Sistema Operativo en el entorno virtual, con el fin de describir dicha funcionalidad.
ii.) Realizar el estudio de los *overheads* de los escenarios de virtualización con el fin de identificar los diversos factores que afectan el rendimiento.

1.3.3 Tercera Hipótesis

"Si en un entorno virtualizado se distribuye en diferentes equipos y con diverso hardware virtualizado, entonces debería existir soportabilidad e interoperabilidad entre plataformas de virtualización."

Está hipótesis estimula a buscar un modelo genérico para lograr que los escenarios virtuales sean reutilizables o migrables con las mínimas adaptaciones pertinentes a distintos contextos y plataformas. Esto se pretende cubrir con los siguientes objetivos concretos:

i.) Analizar y evaluar herramientas para la creación dinámica y despliegue automático de escenarios virtuales, con el fin de determinar la soportabilidad, interoperatibilidad, integración y distribución de los recursos del escenario virtual.
ii.) Definir un modelo genérico para caracterizar el rendimiento de un entorno virtual, según la técnica o herramienta de virtualización que se utilice.

1.3.4 Cuarta Hipótesis

"Si los entornos virtualizados operan en redes distribuidas, entonces se puede interactuar coordinando la distribución de recursos entre servicios dinámicos, con múltiples dominios colectivos o individuales."

Esta hipótesis induce a definir una interfaz genérica para gestión de entornos virtuales, buscando la interoperabilidad de la interacción máquina a máquina a través de una red distribuida. Esto se pretende cubrir con los siguientes objetivos concretos:

i.) Estudiar y evaluar las técnicas y estándares de despliegue de servicios dinámicos en entornos distribuidos.

ii.) Definir una interfaz genérica que interactúe con los servicios dinámicos en entornos distribuidos y permita desplegar escenarios virtualizados.

1.4 Estructura de la memoria

El resto de esta Tesis ha sido estructurada de la siguiente manera:

El Capítulo 2 presenta una síntesis bibliográfica sobre el Estado del Arte que sustenta la presente investigación y que ha sido organizada como sigue: Inicia con una sinopsis de los conceptos básicos de la Virtualización, técnicas y plataformas. Continúa con una evaluación comparativa de las mismas. Posteriormente realiza un estudio sobre las técnicas de modelado de información cara a modelar entornos virtualizados. Continúa con un breve bosquejo de Modelado de Información con un Análisis comparativo de los enfoques de modelado de infraestructuras de virtualización y algunas implementaciones. Finaliza con el analisis de diversas alternativas, cara a definir una arquitectura de despliegue de EVR en entornos distribuidos

El Capítulo 3 se centra en la comprobación de la utilidad de los escenarios de virtualización. Por tanto este capítulo aporta una Emulación de Servicios Multimedia utilizando entornos virtuales de red. Luego explica una evaluación del Rendimiento de Redes IP utilizando Plataformas de Virtualización y Métodos de Simulación.

El Capítulo 4 define un mecanismo para medir la penalización (*overhead*) producido en la capa de virtualización. Se ha aplicado un modelo de Regresión lineal para obtener una expresión analítica que permita predecir el *overhead* de un EVR dado, utilizando la plataforma de virtualización Xen.

El Capítulo 5 define una propuesta para determinar los factores que incrementan la penalización del rendimiento en un experimento utilizando un EVR. Aquí se ha utilizado la técnica estadística denominada ANOVA para establecer qué variables en el interior de un EVR afectan considerablemente al *overhead* y en qué medida.

El Capítulo 6 presenta una propuesta para desplegar escenarios virtuales de red en entornos

distribuidos, con el fin de establecer una arquitectura base sobre la cual se puedan desplegar EVR. En un primer enfoque, se presenta como punto de partida un modelo basado en escenarios distribuidos con VNUML. Luego se implementa una interfaz de Servicios Web para el despliegue de escenarios virtuales dinámicos. Finalmente se modela una interfaz de Servicios Grid, integrando Grid como plataforma y WSRF como un conjunto de especificaciones para interactuar con el estado de los recursos, que en este contexto han sido identificados como EVR.

El Capítulo 7 se enfoca en el diseño y construcción de un modelo genérico para caracterizar EVR basado en DMTF-CIM. Continúa, como método de validación, con la implementación de un sistema de gestión que active las operaciones y el acceso de clases y objetos del Modelo Genérico. Este sistema fue desarrollado mediante WBEM Services y Java. Finaliza con la evaluación de los resultados experimentales y su discusión.

El Capítulo 8 expone el resumen de las conclusiones en base a cada hipótesis. Luego se realiza una valoración y análisis del trabajo realizado. Continúa con una descripción de las contribuciones. Finaliza con las líneas futuras de investigación y Trabajos futuros.

El Apéndice A describe el procedimiento y los resultados obtenidos al realizar la evaluación del rendimiento de redes IP utilizando plataformas de virtualización Xen comparándola con los resultados obtenidos utilizando métodos de simulación como el NS-2.

El Apéndice B despliega parte del esquema de análisis y consecuencias al aplicar ANOVA. Así mismo describe el algoritmo en MatLab para validar el método analítico que permite obtener los coeficientes de Regresión Lineal Múltiple y aplicar el test ANOVA.

El Apéndice C contiene el código correspondiente al modelo de extensión MOF, VNE-CIM con sus respectivas clases e instancias añadidas gestionadas por el CIMOM.

Capítulo 2 Revisión del Estado del Arte

2.1 Introducción

Este capítulo presenta una síntesis bibliográfica sobre el Estado del Arte que sustenta la presente investigación y que ha sido organizada de la siguiente manera: Inicia con una sinopsis de los conceptos básicos de la Virtualización, técnicas, plataformas, herramientas para la generación automática de EVR, herramientas de benchmarking y herramientas para bancos de pruebas (test-beds) orientadas a EVR (véase apartado 2.2). Continúa con una evaluación comparativa de las mismas, cara a identificar aquella plataforma cuyo rendimiento es más cercano al nativo (véase apartado 2.3). Posteriormente realiza un estudio de las técnicas de modelado de información utilizadas en virtualización, puesto que para el desarrollo de esta Tesis es necesario modelar escenarios virtualizados (véase apartado 2.4). Continúa con un Análisis comparativo de los enfoques de modelado de infraestructuras de virtualización y algunas de sus implementaciones (véase apartado 2.5). Finaliza con el análisis de diversas alternativas, cara a definir una arquitectura de despliegue de EVR en entornos distribuidos (véase apartado 2.6).

2.2 Tecnologías de Virtualización

2.2.1 Definiciones preliminares

La *Virtualización* es la forma de particionamiento lógico de un equipo físico en múltiples máquinas virtuales, para compartir recursos de hardware, como CPU, memoria y dispositivos de entrada y salida [Humphreys06]. El concepto de *máquina virtual* surgió originalmente con el sistema MV/370 de IBM en 1972. Una MV es un duplicado de una máquina real, eficiente y aislado [Popek74]. *Duplicado*, en razón de que la MV se debería comportar de forma idéntica a la máquina real, aún con menos recursos disponibles y con las diferencias de temporización al tratar con dispositivos. *Aislado*: Se pueden ejecutar varias maquinas virtuales sin interferencias y con diversas cargas de trabajo. *Eficiente*: Debería ejecutarse a una velocidad cercana a la del hardware real [Adams06].

La virtualización está siendo usada para diversas *aplicaciones* en las que se incluyen: consolidación de servidores para utilizar los recursos desaprovechados, balanceo de carga, entornos de pruebas y validación de software, emulación de servicios de redes, entre los más importantes. Se

debe precisar que estas dos últimas aplicaciones son los propósitos principales de esta investigación. Las principales *ventajas* de la virtualización son: Ahorro de costos de inversión de hardware, simplificación de la gestión dada la administración centralizada de todas las MVs, portabilidad entre servidores físicos, facilidad de técnicas de recuperación de desastres. Algunas *desventajas* son: falta de estandarización; dificultad en el acceso a la red de información del host desde las MVs y la introducción de una penalización u *overhead* debido al consumo de recursos reduciendo el rendimiento [Fernández06].

En la industria se han desarrollado las siguientes *técnicas* de virtualización [Scope08]: ***i)*** *Virtualización completa,* que intenta reproducir el funcionamiento de un ordenador origen en otro destino, sin realizar modificaciones en el sistema operativo; ***ii)*** *Paravirtualización,* que reduce los problemas de rendimiento con cierta modificación en el sistema operativo. En esta técnica se utiliza un componente denominado *hypervisor*, que es una capa de virtualización intermedia entre el hardware y los sistemas operativos hospedados, y que hace de árbitro para el acceso de éstos a los recursos del computador de forma organizada; ***iii)*** *Virtualización a nivel de sistema operativo,* que agrupan procesos y recursos en contenedores especializados pero que tienen un kernel común, por lo que un fallo en él, compromete a todas las MVs. Finalmente; y ***iv)*** *Virtualización por hardware*, que es un nuevo enfoque que usa los avances del hardware de procesadores Intel-VT (*Virtualization Technology*) y AMD-Pacifica *(Advanced Micro Devices)* para eliminar la necesidad de parches en el sistema operativo. Entre 2005 y 2006 Intel y AMD añadieron extensiones independientes a la arquitectura x86 para facilitar las tareas de virtualización.

2.2.2 Plataformas de Virtualización

La presente Tesis se ha enfocado en las principales herramientas de código abierto o de libre distribución, con la excepción de VMware Server (que es gratuito, pero no de código abierto), debido a su generalizada utilización en plataformas Windows y Linux. A continuación se presenta una breve descripción de las herramientas más importantes:

UML (*User Mode Linux*) [UML09], fue creada por Jeff Dike. Permite ejecutar múltiples instancias de diferentes distribuciones de Linux. UML requiere de dos componentes principales un kernel y un sistema de archivos (*root file system*). El primero, es una versión modificada del kernel, mientras que el segundo, es una imagen, que a la vez es un único archivo emulando un sistema de archivos real. Las aplicaciones corren como procesos de usuario final sobre el kernel propio de la MV. UML ofrece opciones para conectividad y mecanismos para la interconexión. UML es una herramienta potente pero compleja si se pretende construir escenarios que incluyan muchas MVs y topologías complejas de red [Fernández06]. Para evitar dicha complejidad y como mejoras, se han desarrollado algunas herramientas de código abierto para la creación de escenarios virtuales basadas en UML, como son VNUML (*Virtual Network User Mode Linux*) [Galán08] y Netkit

[Pizzonia08] [Rimondini07].

VMware [MVwa10], es una plataforma basada en virtualización completa y la mayor parte de las instrucciones se ejecutan directamente sobre el hardware físico. Otros productos incluyen VMware Workstation que es de pago, y los gratuitos VMware Server y VMware Player.

Virtual Box [Vbox10], también está basada en virtualización completa. Se distribuye bajo licencia GNU LGPL (*Lesser* GPL). Dispone de una interfaz gráfica denominada Virtual Box Manage, que permite crear MVs con Windows o Linux y su respectiva configuración de red. VirtualBox ofrece un mecanismo de acceso remoto a las MVs mediante RDP (*Remote Desktop Protocol*).

Qemu [Qem08] , es un emulador genérico de procesadores. Ejecuta MVs en Linux o Windows. Está licenciado bajo licencias LGPL y GPL. Ha liberado el módulo de aceleración de Qemu llamado *kqemu*, que es un controlador que permite ejecutar directamente por el procesador aquellas instrucciones que no sea necesario emular, lo que implica mayor eficiencia.

Xen [Barham03], 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 sistemas operativos con todas sus características, pero carece de entorno gráfico. En el caso de requerirlo, se convierte en una herramienta de uso comercial. El núcleo de Xen, que administra las MVs, se conoce como *hypervisor.*

KMV [KMV09], es una herramienta de libre distribución, que emplea la técnica de virtualización completa, usando las extensiones de virtualización por hardware Intel VT o AMD, para crear MVs que ejecutan distribuciones de Linux. Además requiere una versión modificada de Qemu para completar el entorno virtual.

OpenVZ [Open09], utiliza una técnica de virtualización a nivel de sistema operativo. Es un proyecto de código abierto basado en *Virtuozzo (*software comercial), que trabaja bajo distribuciones Linux, donde la compañía SWsoft ha puesto su código bajo la licencia GNU GPL. OpenVZ carece de las propiedades de *Virtuozzo*, pero ofrece un punto de partida para probarlo y modificarlo.

2.2.3 Plataformas para el despliegue de máquinas virtuales

De acuerdo con [Galán10], en la industria existen algunas plataformas de virtualización que proporcionan gestión basada en EVR, que ocultan los detalles de bajo nivel y que soportan EVR complejos definidos por el usuario. En el siguiente apartado como parte del análisis del Estado del Arte se describirán algunas plataformas de virtualización que incluyen su propia arquitectura de despliegue.

SmartDomian, según [Grehant08] [SmartDom] es una herramienta que permite el despliegue automático de MVs Xen en entornos distribuidos. SmartDomains utiliza una representación

completa de la especificación de recursos distribuidos, incluyendo información acerca de cómo organiza su creación y eliminación. Ha sido probado inclusive en entornos Grid.

VNUML [Vnuml] [Fernández07] es una plataforma de propósito general, basada en UML que ha sido desarrollada para la emulación de EVR en un equipo físico. El objetivo de la misma es facilitar al usuario un mecanismo de simulación de redes, mediante el cual el usuario diseña un escenario, constituido por equipos GNU/Linux e interconectarlos mediante distintas redes. VNUML ha sido utilizado para probar aplicaciones, despliegue de laboratorios de networking, redes señuelo para determinar amenazas en la red y hosting. Sin embargo, por basarse en UML, el uso de VNUML implica solo MVs en Linux y es "claramente no utilizable en aquellos casos en los que se realicen medidas de eficiencia de aplicaciones, ya que el *overhead* introducido por la virtualización falsearía estas medidas" [Galán04]

NetKit, [Netkit], es una plataforma para emular redes basadas en UML. Ha sido concebida como un ambiente para experimentos de networking [Rimondini07]. Permite crear varios dispositivos virtuales de red (routers, switches y Pcs), que se puedan interconectar para formar una red en una única PC. El propósito de este proyecto es solucionar muchas de las dificultades al usar UML para Networking. Para emular una red con Netkit, se debe escribir un shell script o directamente mediante línea de comandos. El script inicia los dispositivos de la red y los interconecta como sea requerido.

MLN, [Bouabid09] MLN (Manage Large Networks) es una plataforma de gestión de MVs diseñado para generar y ejecutar EVR basadas en Xen, VMware Server y UML. Es ideal para la creación de laboratorios virtuales de red para la educación, la verificación, para el alojamiento o simplemente para experimentar con MVs.

vBET, [Jiang03] (VM-Based Emulation) es un emulador de testbeds utilizando MVs. Crea un entorno virtual distribuido con redes y equipos finales.

VMware VirtualCenter [VirtualCenter07], VMware VirtualCenter ofrece gestión centralizada, automatización operacional, optimización de recursos de alta disponibilidad para entornos de IT. Se caracteriza por ser un entorno basado en servicios distribuidos para equipar un centro de datos utilizando como plataforma de virtualizaciòn VMware. Entre sus principales beneficios se puede citar la migración en vivo de MVs entre servidores físicos completamente separados y su sistema de monitorización. VirtualCenter esta compuesta por los siguientes componentes: Control central de gestión, base de datos, infraestructura virtual del cliente, agentes e infraestructura de acceso Web. Es un producto comercial que actualmente ha sido incluido en la plataforma VMware Infraestructure 3, que integra productos de tales como VMware VMotion (para la migración en vivo de MVs), VMwareHA (para ejecutar aplicaciones en MVs) y VMware DRS (para la asignación de recursos virtuales), los cuales se venden por separado.

XenServer [XenServer] es una plataforma de virtualización de servidores basada en Xen que

suministra características de migración de MVs, soporte de almacenamiento compartido y gestión centralizada de multi-servidores. XenServer utiliza el hipervisor Xen para virtualizar cada servidor en el que está instalado, permitiendo a cada uno alojar múltiples MVs de forma simultánea. XenServer permite a los administradores de TI crear múltiples clusters de recursos y cómo gestionar desde un único punto de control. Entre los elementos principales de Xen Server se pueden citar: el Hipervisor Xen, instaladores para XenServer, XenExpress, XenEnterprice o XenCenter, la interfaz de línea de comandos y las plantillas para instalación sobre Windows, o algunas distribuciones de Linux.

Enomalism, [Enomalism] es un gestor gráfico de código abierto basada en el Web que ha sido desarrollado por Enomaly (consultora canadiense) para la gestión de MVs Xen. Enomalism proporciona una interfaz de gestión de múltiples servidores independientes de forma concurrente usando el entorno Enomalism Virtualized Grid y la plataforma Elastic Computing. Entre los elementos principales de Enomalism se pueden citar: el asistente de creación de servidores virtuales, las plantillas que facilitan la configuración de servidores virtuales, un mecanismo de despliegue de aplicaciones, la integración con herramientas de terceros mediante Web Services y la gestión centralizada de usuarios a través de LDAP.

ConVirt (Controlling Virtual Systems) [Convirt10], anteriormente conocido como *XenMan* es un proyecto de código abierto para la gestión de plataformas de virtualización para un centro de datos que utilizan Xen y KVM. ConVirt es una herramienta gráfica que permite centralizar la gestión de todo el entorno virtualizado desde una única consola. Las tareas más comunes son: Gestión de múltiples servidores centralizada, comunicaciones seguras vía SSH, almacenamiento de imágenes y administración remota. Actualmente existen dos versiones, ConVirt 2.0 de código abierto y ConVirt 2.0 Enterprise.

2.2.4 Herramientas de benchmarking

Las herramientas de benchamarking sirven para evaluar y comparar el rendimiento de recursos en entornos reales y virtuales. El término “benchmark” designa una escala de comparación de rendimiento de la utilización del CPU, los procesos de comunicación, acceso al disco y rendimiento de la red entre otros. A continuación se describen algunas herramientas que han sido investigadas en el desarrollo de la Tesis:

SPEC [SPEC], es una corporación no lucrativa formada para establecer, mantener y endosar un sistema estandarizado de benchmark relevante que se pueda aplicar a la nueva generación de computadores. SPEC desarrolla benchmarks y realiza revisiones, publicando resultados sometidos de organizaciones miembros y de otros concesionarios. En el ámbito de la virtualización, SPEC ha formado desde noviembre de 2006, un subcomité para desarrollar métodos y estándares de comparación del funcionamiento de la virtualización para los servidores de centros de datos. Este

grupo está investigando el balanceo de cargas de trabajo que se separan a través de múltiples MVs en un solo servidor, y los métodos y métricas usadas son tomados de la base de trabajos como SPECjAppServer2004 y [SPECweb2005]. Este último, por ejemplo, emula peticiones de navegador sobre conexiones de banda ancha del Internet a un Web Server.

Unixbench [UnixBench], contiene una colección de procedimientos de prueba para evaluar múltiples aspectos del rendimiento de un servidor, mediante una comparación del uso del CPU, procesos de copia, lectura, escritura y un análisis de los procesos de inter comunicación. Incluye pruebas de tipo aritmético (overhead), de sistema (creación de procesos, número de iteraciones) y de compilación.

Imbench [Imbench], dispone de varias pruebas con medidas en microsegundos y representan los retardos de ida y vuelta para diversos formatos de inter procesos de comunicación. Existen pruebas como lectura, escritura, de estado, de prueba abierta y cercana, de cancelación, de bifurcación, de comunicación entre procesos, de comprobación de TCP, RPC/TCP, UDP y RPC/UDP, cada uno con su funcionalidad específica.

Autobench [Autobench], es un script de Perl para automatizar los procesos de benchmarking del rendimiento de un Web Server o para conducir una prueba de comparación de dos Web servers diferentes. El script funciona con *httperf*, que como se observará en el apartado 4.4, es una herramienta para medir el rendimiento de un Web Server, generando varias cargas de trabajo durante su ejecución. El Autobench conecta varias instancias de httperf, incrementando el número de peticiones de conexión por segundo en cada iteración, que le permite extraer datos significativos en formato que puede ser importado directamente en hojas electrónicas para análisis y graficación de resultados.

Ubench [Ubench], proporciona una única medida de rendimiento para las máquinas que ejecutan varias distribuciones del sistema operativo Unix. Se ha desarrollado para medir CPU y memoria. En el futuro se prevé pruebas agregadas para el disco y el TCP/IP. Ubench realiza cálculos matemáticos con números enteros o de punto flotante, durante tres minutos de varios procesos que se están ejecutando concurrentemente, así como una serie de copias de datos en localizaciones de memoria. Sus resultados son benchmarks de CPU y memoria.

VMmark, de acuerdo con [VMmark06] [Markhija07], proporciona medidas exactas y confiables del rendimiento de entornos virtualizados. Es el primer benchmark creado con el propósito de evaluar entornos virtualizados para la industria de computadoras x86. VMmark es una herramienta valiosa para fabricantes de equipo, vendedores de software, integradores del sistema y otras organizaciones que desean adoptar decisiones apropiadas en la infraestructura virtual y determinar el rendimiento de diferente hardware con diferentes plataformas de Virtualización.

2.2.5 Herramientas para test-beds que incluyen a máquinas virtuales

Emulab, [Emulab], es un testbed que proporciona el acceso integrado a entornos experimentales de redes. Provee a los investigadores una amplia gama de ambientes para emularlos, eliminar errores, evaluar redes, escenarios virtuales y entornos distribuidos. Existen varias instalaciones del software de Emulab que proveen una interfaz común a más de una docena de sitios alrededor del mundo, con centenares de nodos. También es usado para impartir clases en esos campos y es de uso público, con esquemas de autorización y políticas de uso. Algunos de sus ambientes experimentales son: emulación, experimentación de Internet en vivo, redes inalámbricas, procesamiento de señales de radio definidas por software, sensores de red, redes móviles y simulación.

Modelnet, [ModelNet] es un emulador de red que permite evaluar sistemas distribuidos y en línea como el Internet, enfocados en la escalabilidad. Permite evaluación de prototipos que se ejecuten en SOs no modificados en varios escenarios de red. ModelNet combina la capacidad de repetición de la simulación con el realismo del despliegue vivo de la emulación. Es utilizado para el diseño y evaluación de sistemas peer to peer, protocolos de capa de transporte, conmutación basada en contenido, sistemas de archivos distribuidos y herramientas de medición de red. Cada instancia de una aplicación se ejecuta en un nodo virtual. ModelNet multiplexa nodos virtuales a través de un conjunto de máquinas físicas llamadas nodos de borde. El sistema configura los nodos del borde para enrutar sus paquetes a través del núcleo de Modelnet, con el objeto de limitar el retardo de cada paquete, el ancho de banda y la pérdida de paquetes en una topología especificada. Para usar ModelNet se debe especificar la topología, configurar el direccionamiento IP de cada nodo virtual y de borde y realizar el despliegue mediante scripts. Puede ser usado en combinación con NS para escenarios de redes complejas [Guruprasad05].

PlanetLab [PlanetLab] es un laboratorio computacional para poner a disposición de la comunidad científica y de investigación recursos computacionales para la experimentación de nuevas tecnologías y protocolos que requieran una red para su ejecución. PlanetLab es un conjunto de servidores distribuidos a través de las redes académicas del mundo, que a su vez forman un gran laboratorio a gran escala. Actualmente está compuesto por sobre los 800 servidores repartidos en más de 400 sitios en el mundo. PlanetLab soporta virtualización en entornos distribuidos donde cada servicio se ejecuta sobre un tramo global de recursos. PlanetLab soporta plataformas Linux-VServer sobre múltiples servidores donde se ejecutan concurrentemente.

Hasta aquí se ha conseguido realizar un análisis de las principales plataformas y herramientas de generación, gestión y despliegue de EVR. Sin embargo, es conveniente realizar una comparación cuantitativa de las plataformas de virtualización que permiten crear EVR, a fin de verificar mediante la experimentación dicha funcionalidad. El apartado 2.3 muestra esta comparación:

2.3 Evaluación Comparativa de las plataformas de Virtualización

Dada la variedad de herramientas de virtualización en la industria, en este apartado se plantea una evaluación comparativa de las mismas, basada en el despliegue de un EVR, como se indica a continuación:

2.3.1 Diseño del escenario

Para proceder a la evaluación, se ha diseñado el escenario de la Fig. 2.1. Es un entorno Cliente/Servidor virtual integrado con la parte real (host anfitrión). Todas las MVs son Linux (Virtual Web Server, MV1, MV2, Router 1, Router2 y Router3). Además, en el servidor denominado Virtual Web Server se instaló Apache 2 como servicio Web. Este escenario ha sido diseñado porque reúne el número de equipos y componentes suficientes para hacer pruebas de red, conmutación, pruebas de conectividad, ejecución de aplicaciones, enrutamiento, etc. Este escenario puede ser modificado de acuerdo a las necesidades que se tengan, añadiendo o disminuyendo saltos en la red, uniendo sistemas autónomos, realizando filtrado de paquetes, NAT (*Network Address Translation*), etc.

El experimento para evaluar las herramientas consistió en medir la carga del CPU y memoria durante el arranque del escenario y ejecución de una descarga de archivos dentro del escenario virtual con la mayoría de las herramientas de virtualización comentadas en la apartado 2.2. Concretamente, se midió el rendimiento de la red virtual, así como el del equipo físico que actuaba como anfitrión.

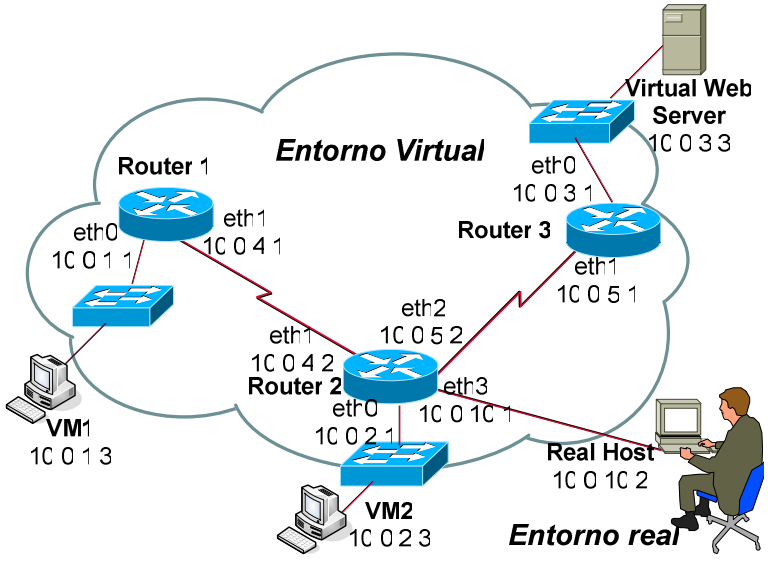

Figura 2-1 Diseño Lógico y físico del escenario virtual.

2.3.2 Implementación

Todas las pruebas se desarrollaron sobre Linux Debian 4.0-testing-i386 Kernel 2.6.18-4-686, en un computador Pentium D, 2.80 GHz, RAM de 1GByte y una partición Ext3 de 60 GB en disco duro. En todas las MVs se instaló el mismo sistema de ficheros y el mismo kernel 2.6.18-3-686-GNU/Linux Debian, independientemente de la herramienta de virtualización, a fin de obtener una comparación más justa (véase Tabla 2-1). La única herramienta que no lo permitió fue Netkit, por lo que su evaluación no ha sido considerada en el análisis realizado posteriormente.

Tabla 2-1 Herramientas de virtualización evaluadas.

	Versión	Kernel	Complementos
VNUML	1.7.2	2.6.18.3-686-GNU/Linux Debian	
Netkit	2.4	2.6.11.7-i686-GNU Linux	
QEMU	0.9.0	2.6.18.3-686-GNU/Linux Debian	Kqemu 1.3.0pre11
Xen	3.0	2.6.18.4-xen-686-GNU/Linux Debian	
VMware	1.0.1	2.6.18.3-686-GNU/Linux Debian	VMware-player 2.0.0
VirtualBox	1.3.8	2.6.18.3-686-GNU/Linux Debian	

El procedimiento utilizado para implementar el experimento consistió en los siguientes pasos: instalación de cada herramienta de virtualización, creación de MVs, clonación, direccionamiento IP, configuración de servicios, creación y aplicación de algoritmos para arranque automático en *shell script*, sincronización de reloj con NTP *(Network Time Protocol),* construcción y aplicación del algoritmo para medición. Finalmente la aplicación de software de medición de tráfico.

2.3.3 Evaluación Experimental

La Tabla 2-2 muestra la media (μ) obtenida de los datos recolectados cuando arrancan los escenarios, calculados de once medidas realizadas con cada herramienta de virtualización con el comando *top* de Linux (comando que provee información del uso de CPU de usuario, kernel, etc.). Estos resultados muestran que VNUML es la herramienta que consume menos CPU durante el arranque con un 31.73 %. Le sigue Xen, con 35.14%.

Tabla 2-2 Consumo de CPU en el arranque.

	user	system	nice	Idle	Wa	hw int	sw int	steal time	Summa-Idle
VNUML	10.78	20.75	0.00	60.29	0.13	0.05	0.03	0.00	31.73
Netkit	16.03	31.60	0.00	51.84	0.34	0.04	0.15	0.00	48.16
QEMU	63.10	23.95	0.00	10.91	0.18	0.06	0.22	0.00	87.51
Xen	2.04	4.17	0.00	58.86	22.01	0.03	0.13	6.76	35.14
VMware	13.29	68.72	0.00	14.34	3.23	0.11	0.14	0.00	85.49
VirtualBox	3.46	38.00	34.26	17.03	6.97	0.09	0.10	0.00	82.88

Como puede observarse en la Figura 2-2, cada campo del consumo de CPU se comporta

diferente. Sin embargo, su sumatoria claramente indica cuál herramienta de virtualización consume el mayor o menor porcentaje de CPU, cuando se arranca el escenario de red. Qemu es la herramienta que mayor consumo de CPU ha demostrado en estas medidas, aunque con valores próximos a VMware y VirtualBox.

La Tabla 2-3 muestra el tiempo esperado cuando arranca el escenario de red con cada herramienta de virtualización, y lo compara con el consumo de CPU registrado. La mejor solución es aquella que tiene menor valor en ambas columnas. En este caso, las mejores herramientas de virtualización son aquellas basadas en UML y Xen.

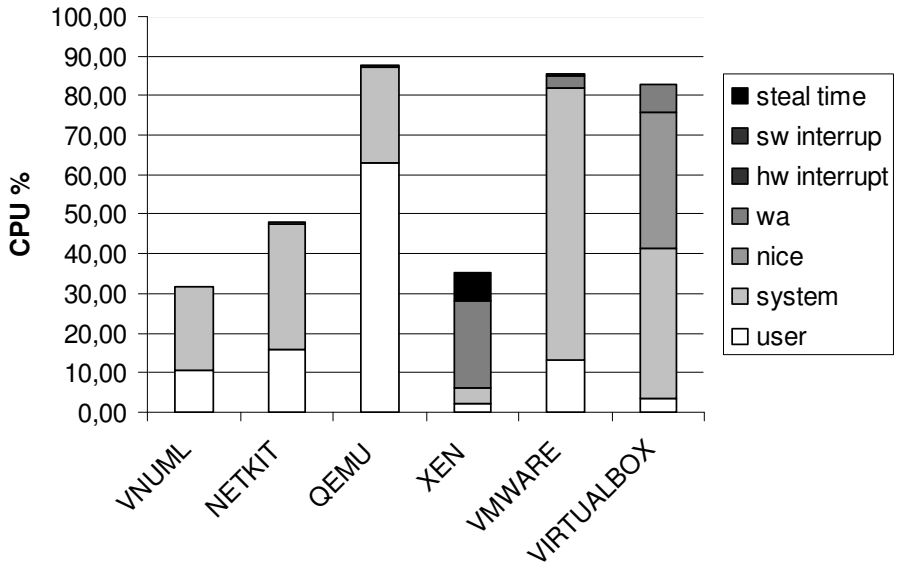

Figura 2-2 Consumo de CPU durante el arranque.

Tabla 2-3 Tiempo transcurrido durante el arranque.

	Time (s)	CPU %
VNUML	80	31.73
Netkit	58	48.16
QEMU	120	87.51
Xen	109	35.14
VMware	105	85.49
VirtualBox	87	82.88

La tabla 2-4 provee el consumo de memoria en el arranque. Se observa que Xen consume 204.90 MB. Esto se debe probablemente a que Xen tiene la habilidad de redimensionar la memoria dinámicamente [Barham03], tanto del Dominio 0 (host anfitrión), como de las MVs, reduciendo la cantidad de memoria requerida por el host anfitrión.

La Figura 2-3, finalmente, muestra una comparación entre el consumo del CPU y memoria durante el arranque con la ejecución de la descarga de archivos. Se puede observar que ambos

valores varían considerablemente en la mayoría de las herramientas de virtualización, excepto Xen, la cual mantiene valores estables.

Tabla 2-4 Memoria consumida durante el arranque

	Consumed	Buffers	Cache	Total (KB)	Total (MB)
VNUML	613,652.98	89,257.40	247,900.77	276,494.81	270.01
Netkit	466,763.82	34,243.32	242,244.45	190.276.05	185.82
QEMU	717,314.60	18,740.75	451,260.44	247,313.41	241.52
Xen	345,967.69	11,539.29	124,614.41	209,814.00	204.90
VMware	643,106.22	8,102.36	341,359.51	293,644.34	286.76
VirtualBox	861,175.28	3,175.58	151,525.76	706,473.94	689.92

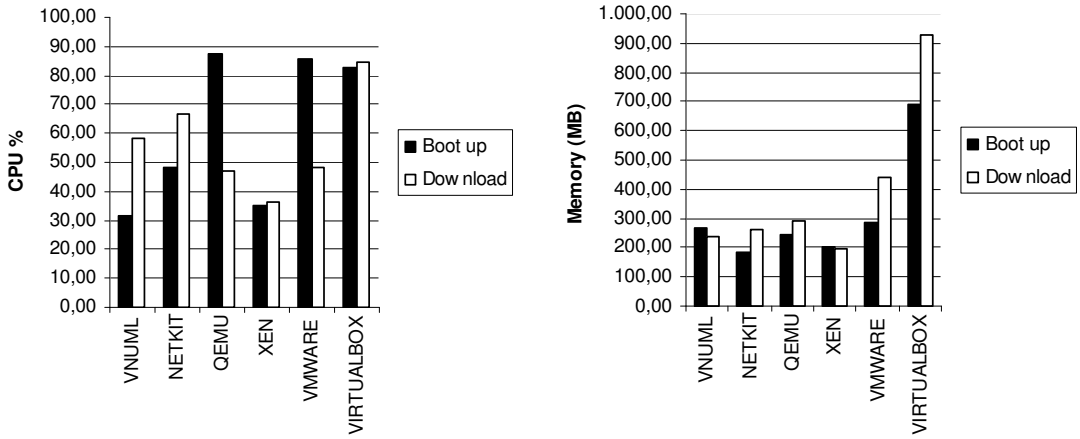

Figura 2-3 Comparación en el arranque y ejecución de descargas. CPU (izq.) y memoria (derecha).

Al finalizar esta evaluación comparativa, se han identificado además algunas diferencias significativas entre las plataformas de virtualización a la hora de desplegar un escenario virtual. Por ello surgió la necesidad de estudiar las técnicas de modelado de información para infraestructuras virtualizadas con el fin de modelar un escenario virtual independiente de la plataforma de virtualización. En el siguiente apartado se explica las técnicas de modelado de información cara a modelar un EVR.

2.3.4 Trabajos relacionados

Los resultados presentados anteriormente pueden ser comparados y complementados con otros trabajos. Aquí se han incluido los más relevantes, que se han encontrado durante la investigación:

El trabajo presentado en [Matheus07] describe el diseño de una benchmark que cuantifica el impacto del rendimiento en VMs. Dicho experimento incluye seis diferentes pruebas intensivas: CPU, memoria, disco, networking, creación de procesos y forkbomb. Ellos describen el diseño de un benchmarking aplicando específicamente el SPECWeb2005 con diversos servicios instalados en varias VMs con diferentes SOs en la misma maquina física, comparando tres técnicas de

virtualización, virtualización completa con VMware Workstation, que es la distribución comercial de VMware, Paravirtualización con Xen y a nivel de OS mediante Solaris Containers y OpenVZ. Finalmente presentan los resultados obtenidos.

El trabajo que nos ocupa difiere en razón de que se ha evaluado la mayor cantidad de herramientas de virtualización y porque se ha aplicado una única prueba patrón (benchmark propia) usando comandos de Linux Debían.

En [Deandrade07], existe un trabajo que realiza un estudio comparativo de dos plataforma de virtualización, XenExpress 3.1 y VMWare Server 1.02, aplicando para el efecto herramientas de benchmarking de libre distribución como Ubench, UnixBench, Netperf, Sysbnech, entre otras. Para su medición propone dos escenarios, el uno en ambiente nativo y el segundo con solo dos VMs ejecutándose al mismo tiempo, con las herramientas mencionadas. Luego aplica las pruebas embebidas en cada software de benchmark y presenta los resultados obtenidos. En nuestra investigación, el diseño del escenario es diferente y sobre todo el número de Plataformas de virtualización comparadas es mayor.

En [Markhija06], se presenta un benchmark escalable para entornos virtualizados, llamado VMMark [VMMark06], que consiste en medir varias cargas de trabajo ejecutándose simultáneamente en MVs separadas, utilizando VMWare ESX Server 3.0. Este benchmark consistió en un conjunto de diversas cargas de trabajo, comúnmente encontradas en los centros de datos, incluyendo Mail Server, Java Server, Standby Server, Web Server, Database Server y File Server, realizando con cada uno de ellos y en diferentes plataformas un sin número de simulaciones y pruebas mediante, Microsoft Exchange, LoadSim, SPECjbb2005, SPECWeb2005, Swingbench, Dbench, entre otras. Posteriormente se presentó la metodología de validación y normalización.

Todas estas pruebas se realizaron utilizando VMware, ESX Server 3.0 y en el cliente Microsft Windows 2003. La generación de carga del cliente se ejecutó utilizando 4GB de memoria y un enlace de 100 Mbps, lo cual podría significar que en equipos de sobremesa para experimentación sería imposible crear VMs con esos requerimientos. Por otro lado, dicho benchmark no tiene en cuenta el comportamiento de la red, que es vital en el caso de nuestra investigación.

En [Romondini07], se muestra una matriz de comparación de productos de virtualización orientados a red. Luego es configurada y emulada una topología de red compleja usando Netkit. En este documento se describe todo el método de configuración y prueba. En nuestro caso, también incluimos Netkit, pero se agregó la posibilidad de cuantificar el consumo de recursos.

El trabajo descrito en [Fernández06], provee una memoria descriptiva de VNUML, explicando el uso de técnicas de virtualización para la creación de complejos EVR. Estos escenarios pueden ser usados para enseñanza de redes de comunicación, y para varios proyectos de experimentación de redes. Allí se considera como principales ventajas de la virtualización la reducción de costos en la inversión de hardware y la facilidad para la configuración de escenarios de MVs y su conexión

con el host. En nuestra investigación, también se ha incluido a VNUML, como un frontend de UML. Por ello, se ha desarrollado nuestro propio escenario con XML, basado en su Lenguaje de referencia. Luego se ha intentado compararlo con otras plataformas de virtualización.

In [Grau06], se evalúa la conveniencia de la emulación de networking en maquinas virtuales. Inicialmente se realiza una breve descripción de varias Plataformas de virtualización, y luego de un proceso de selección realiza una evaluación de Xen2, Xen3, Vmware, UML y Qemu, utilizando para dichos escenarios la integración de la herramienta NetShaper[1]. Previo a ello, se definen varios criterios de comparación basados en el tráfico y el throughput. Finalmente se miden y evalúan sus resultados. En nuestra investigación, adicionalmente a las plataformas de virtualización descritas se ha incluido a VirtualBox, Netkit y VNUML y en lugar de realizar medidas de caracterización del tráfico, se ha medido cuantitativamente el rendimiento de un sistema virtual, tomando varias muestras del consumo de memoria y CPU.

El trabajo presentado en [Brastaad06] examina el uso de la virtualización en la administración de servicios de alta disponibilidad usando Xen. Este trabajo esta basado en el Software de Herbeat[2], para tener la capacidad de simular la migración de VMs entre nodos físicos. En este caso se realiza un test práctico con la integración entre Xen y Herbeath. El trabajo que nos ocupa, difiere porque el nos hemos enfocado a la implementación de EVR.

En [Seetharaman06] la virtualización es introducida desde una perspectiva de pruebas. Allí se resumen experiencias y se compara diversos enfoques como técnicas de libre distribución o comercialización, velocidad de operación y principales ventajas y desventajas. Sin embargo ellos no proveen ningún resultado experimental. En nuestra investigación, se presenta el escenario de prueba, han sido configurados las redes y servicios y luego han sido tomadas las medidas con las principales Plataformas de virtualización.

En [Vanderham04], fueron discutidos algunos proyectos para configurar redes virtuales mediante User Mode Linux, tal como MLN, VNUML y SNB, explorando sus fortalezas y debilidades y definiendo restricciones y requerimientos para la configuración de una red virtual ideal, en razón de realizar experimentos de protocolos de redes en el ámbito educativo. En nuestra investigación se ha incluido otras técnicas de virtualización y por lo tanto otras plataformas de virtualización.

En [Hibler04], existen algunos experimentos utilizando Emulab. Allí se describe técnicas que permiten emulación de ambientes de red para virtualizar recursos como equipos, routers y redes combinándolas con simuladores como el NS y Jail de FreeBSD. Ellos obtienen experiencias con

[1] http://net.informatik.uni-stuttgart.de/software/index.html

[2] http://www.linux-ha.org/HeartbeatProgram

varios nodos y miden el ancho de banda. Nuestra investigación intenta comparar diversas técnicas de virtualización para justificar que la plataforma de experimentación obtenida, es aquella que consume menos recursos de CPU y memoria.

2.4 Modelado de Información

Desde el punto de vista de gestión de la configuración, la Tesis se ha enfocado en especificar un EVR, mediante técnicas de modelado de información a fin de resolver los problemas asociados con la interoperabilidad y la gestión de recursos virtuales. Este apartado tiene como propósito analizar qué lenguajes de definición, modelos y técnicas informáticas estandarizadas en la industria son viables de aplicar para la representación de la información, cara a modelar la generación automática o dinámica de EVR (véase apartado 2.4). En este mismo apartado se abordó el análisis de enfoques existentes para Modelado de infraestructuras de Virtualización (véase apartado 2.5).

2.4.1 Modelo de Información Común (CIM)

Con el fin de facilitar la gestión de datos y recursos entre diferentes sistemas de software, la Fuerza de Tareas de Gestión Distribuida (DMTF, *Distributed Management Task Force*), ha definido un conjunto de tecnologías y protocolos para modelar y acceder a los recursos de sistemas computacionales y redes de una manera estándar abstrayéndose de las tecnologías y proveedores. Estas tecnologías están agrupadas en el concepto de (WBEM, *Web-Based Enterprise Management*) y se basan en el uso del modelo de Información común (CIM, *Common Information Model*) para modelar sistemas, sus atributos y métodos.

WBEM es un conjunto de tecnologías utilizadas para escribir programas e interfaces para la gestión de un entorno empresarial. WBEM se basa en CIM, que es un modelo conceptual de información para describir entidades gestionadas, su composición, y sus relaciones. Utiliza el formato de objetos gestionados (MOF, *Managed Object Format*) para definir esta descripción de objetos gestionados de una manera formal. El principal elemento de una arquitectura de WBEM es el (CIMOM, *CIM Object Management*), que básicamente es un administrador de base de datos para las instancias de clases CIM y representa el punto central de acceso a recursos gestionados. Los clientes CIM implementan el acceso normalizado a los datos y a las operaciones de gestión de los recursos que están representados en el CIMOM. Por último, los proveedores de la CIM, actúan como intermediarios entre CIMOM y los dispositivos gestionados, implementando los métodos definidos por la CIM.

CIM es un enfoque para la gestión de sistemas, software, usuarios, redes y más, que proporciona una definición coherente y la estructura de datos utilizando técnicas orientadas a objetos [DMTFDSP0111]. El valor de CIM se deriva de su orientación a objetos [Hobbs04]: abstracción y

clasificación, para reducir la complejidad del dominio del problema; herencia de objetos para heredar métodos y propiedades; y la capacidad para describir dependencias, asociaciones y relaciones. CIM se estructura de la siguiente manera [López03]:

i.) **Metaesquema**, para la definición formal del modelo, que define las construcciones básicas de CIM como son clases, propiedades, métodos, etc;

ii.) **Modelo nuclear**, que define un esquema aplicable para todas las áreas de gestión;

iii.) **Modelos comunes,** que definen esquemas para un área de gestión en particular; y

iv.) **Modelo de Extensión**, que a partir de un modelo común, se crean para un esquema específico o concreto o un determinado fabricante.

CIM tiene una relación formal con el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) que fue adoptado por el Grupo de Gestión de Objetos (OMG, *Object Management Group*) en noviembre de 1997 [OMG], y que se ha convertido en un estándar para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente de diseño. Para formalizar la relación existente entre CIM y UML se ha preparado el documento denominado "Perfil UML para CIM", descrito en [DMTFDSP2013]. En su alcance se definen las especificaciones que permiten la conversión automática entre el formato de archivos CIM-MOF u otra representación equivalente de un modelo CIM y el modelo UML. Esta conversión puede ser realizada en ambas direcciones sin perdida de la información. En los siguientes apartados se describirán UML, CIM-MOF y MOF.

2.4.2 El Lenguaje de Modelado Unificado (UML)

De acuerdo con [Booch97] el Lenguaje de Modelado Unificado puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML se ha convertido en un estándar para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente de diseño. UML tiene una notación gráfica muy expresiva que permite representar todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

Un objetivo importante de UML es que sus modelos sean independientes del lenguaje de implementación, de tal forma que los diseños se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML, de manera particular los lenguajes orientados a objetos.

UML es un método formal de modelado, lo cual aporta las siguientes ventajas: mayor rigor en la especificación, permite realizar una verificación y validación del modelo realizado y finalmente se pueden automatizar determinados procesos permitiendo generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos)

A pesar de las ventajas expresadas, UML presenta los siguientes problemas: UML permite hacer

especificaciones más claras, pero no permite automatizar su validación, UML mantiene alta ambigüedad de ciertos diagramas (casos de uso). UML está orientado a los ingenieros y técnicos, y mantiene difícil comunicación con los usuarios y clientes, UML es relativamente complejo y finalmente, UML no se puede ejecutar.

UML incluye los siguientes diagramas: **Diagrama de clases** que **m**uestra un conjunto de clases, interfaces, así como sus relaciones; **Diagrama de objetos** que muestra un conjunto de objetos y sus relaciones; **Diagrama de casos de uso** que muestra un conjunto de casos de uso y actores y sus relaciones; **Diagrama de secuencia** que **e**s un diagrama de interacciones que resalta la ordenación temporal de los mensajes; **Diagrama de colaboración** que es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes; **Diagrama de estados** que cubren la vista dinámica de un sistema y el comportamiento de una interfaz, una clase o una colaboración; **Diagrama de actividades** que muestra el flujo de actividades dentro de un sistema.; **Diagrama de componentes** que muestra la organización y las dependencias entre un conjunto de componentes; **Diagrama de despliegue** que **m**uestra la configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

2.4.3 CIM-Managed Object Format (CIM-MOF)

La especificación CIM define un lenguaje basado en el lenguaje de definición de interfaz (IDL) llamado (MOF, Managed Object Format), para representar gestión de la información [CIM-Tutorial]. La sintaxis de MOF es una manera de describir las definiciones de objetos en forma textual. Los principales componentes de una especificación MOF son descripciones textuales de las clases, asociaciones, propiedades, referencias, los métodos y las declaraciones de instancia y de los calificadores de asociados.

2.4.4 Meta-Object Facility

Es una definición formal de UML, que se consigue mediante un metamodelo expresado en un metalenguaje denominado (MOF, *Meta-Object Facility*). Se trata de un estándar OMG que define un lenguaje común y abstracto para definir lenguajes de modelado para acceder e intercambiar modelos expresados en dichos lenguajes. MOF usa cinco construcciones básicas para definir un lenguaje de modelado: clases (usadas para definir tipos de elementos), generalización (define herencia entre clases), atributos (define propiedades de los elementos del modelo), asociaciones (definen relaciones entre clases) y operaciones (define operaciones dentro del ámbito de una clase).

MOF extiende UML para que este sea aplicado en el modelado de diferentes sistemas de información. MOF define diversos metamodelos, esencialmente abstrayendo la forma y la estructura que describe los metamodelos [Caramazana04]. MOF define los elementos esenciales,

sintaxis y estructuras de metamodelos que son utilizados para construir modelos. Finalmente MOF se utiliza para definir un modelo de infomación para un dominio de interés en particular. Esta definición es posteriormente usada para conducir el diseño e implementación subsecuentes de software conectado con el modelo de información.

2.5 Enfoques de modelado de infraestructuras virtualizadas.

Este apartado muestra un análisis comparativo de los principales enfoques de modelado de virtualización encontrados en esta investigación y algunas de sus implementaciones. Este análisis es muy importante porque revela las fortalezas y limitaciones de cada enfoque y los aspectos que deben ser mejoradas como aportación de la Tesis. En primer lugar, este apartado presenta un resumen de las características relevantes de cada enfoque de modelado y de algunas de sus implementaciones. Luego se describen los criterios de evaluación y se procede a verificar los resultados de la evaluación.

2.5.1 Introducción

El modelado de sistemas virtualizados ha sido una creciente preocupación para los organismos de normalización, dada la evolución que las tecnologías de virtualización han tenido en los últimos años. De hecho, la DMTF ha definido desde finales de 2007 un conjunto de extensiones para CIM, que incluye un modelo para diferentes tipos de plataformas de virtualización [DMTFDSP2013]. Sin embargo, las especificaciones de CIM para las redes, servicios, y recursos no consideran el despliegue automático un EVR en su conjunto.

En este contexto, en la industria existen algunos enfoques de modelado, tales como: ***i)*** Perfil de Virtualización DMTF-CIM [DMTFDSP1042], que define cómo utilizar CIM para la representación de los sistemas virtuales y sus recursos definidos en el Perfil del Sistema Virtual [DMTFDSP1057]; ***ii)*** Formato Abierto de virtualización (OVF, *Open Virtualization Format*) [DMTFDSP2043], que es un estándar abierto para el envasado y distribución de aplicaciones virtuales; y ***iii)*** VMware-CIM [VMware-CIM] que proporciona un modelo de objetos coherente para las máquinas virtuales y sus dispositivos de almacenamiento asociados.

Además, existen implementaciones de los enfoques de modelado mencionados en el párrafo anterior tales como: ***i)*** Libvirt-CIM [Libvirt-CIM], que es un conjunto de bibliotecas para la gestión de MVs utilizando el modelo de virtualización de la DMTF; y, ***ii)*** Xen-CIM [Xen-CIM] para la definición de modelos de sistema y recursos virtualizados utilizando Xen como plataforma.

2.5.2 Perfiles de Virtualización DMTF-CIM

Con respecto los perfiles de modelado de virtualización DMTF-CIM, desde 2007, el Grupo de

Trabajo (SVPC, *System Virtualization, Partitioning and Clustering Working Group*), ha trabajado en la creación de perfiles para la gestión de los sistemas computacionales virtualizados, y la creación de especificaciones para la gestión de virtualización inter-operable. En el ámbito de esta Tesis, los perfiles más sustanciales son el perfil de Virtualización del Sistema [DMTFDSP1042] y el perfil del Sistema Virtual [DMTFDSP1057]. El primero se enfoca en el modelado de la relación entre un host físico y sus recursos virtuales. También se ocupa de tareas específicas de virtualización como la creación o modificación de los sistemas virtuales y sus configuraciones. En contraste, el segundo define un modelo de objetos mínimo dirigido a verificar la inspección y el funcionamiento básico de un sistema virtual (es decir, MV), sus componentes, y su ciclo de vida.

2.5.3 Formato de Virtualización abierta (OVF)

Otro enfoque reciente para modelar los entornos virtuales (también definido por el DTMF SVPC) es el Formato de Virtualización Abierta (OVF, *Open Virtualization Format*) [DMTFDSP2043], cuyo objetivo es la especificación formal de dispositivos virtuales (VA, *virtual appliances*) intentando conseguir la neutralidad de plataformas de virtualización. Un VA puede ser definido como una pila de software pre-configurado que comprende una o más máquinas virtuales entre sí en una topología dada para proporcionar servicios auto contenidos, siendo de esta manera similar al concepto de EVR. El modelado de un VA se realiza a través de un descriptor OVF basado en XML el cual especifica diferentes piezas de información (es decir, los requisitos de hardware, discos virtuales, redes, etc.). OVF también considera los mecanismos de seguridad para garantizar la integridad de un VA. Cabe mencionar que OVF se centra en el empaquetamiento y los pasos de distribución del ciclo de vida, pero la operación de gestión de las MVs, una vez que se han desplegado está fuera de su ámbito de aplicación.

2.5.4 VMware-CIM

Otro enfoque importante es el modelado de VMware basado en CIM, llamado VMware-CIM. VMware-CIM es específicamente un esquema de extensión CIM, es decir, un modelo de la estructura lógica de las plataformas de VMware-ESX, sus MVs, y de sus recursos asignados [VMware-CIM]. Las clases de este esquema se han organizado en tres grupos principales: Modelo de dominio virtual, Modelo de subsistemas de almacenamiento virtual y modelo de múltiples-rutas del servidor VMware. La versión liberada en mayo 2009 llamado VMware CIM API 4.0 ha sido desarrollada para las siguientes aplicaciones: ***i)*** el vSphere CIM SDK de la plataforma VMware, que ofrece un conjunto de herramientas para desarrolladores de software para crear aplicaciones, y ***ii)*** la Arquitectura de Sistema de Gestión de hardware del servidor (SMASH, *System Management Architecture for Server Hardware*) [VMware-SMASH], que contiene sugerencias para utilizando

las clases CIM se pueda llevar a cabo casos de uso comunes para administrar los servidores ESX.

Para concluir, cabe mencionar que en septiembre de 2008, la DMTF puso en marcha la Iniciativa de Gestión de Virtualización (MVAN, *Virtualization Management Initiative*) [MVAN], que agrupa a OVF y algunos perfiles de virtualización DMTF descrito anteriormente (recientemente, la incubadora de nube de computación (*cloud computing incubator*) ha sido también incluida), para enfocar sus esfuerzos en la estandarización de virtualización y afines.

2.5.5 Implementaciones

2.5.5.1 Libvirt-CIM

Libvirt-CIM es un proveedor CMPI, (*Common Manageability Programming Interface*), es decir, se puede utilizar con cualquier CIMOM que soporte la interfaz de CMPI, que tolera el modelo de virtualización DMTF-SVPC. Según [Libvirt-CIM], el objetivo de este enfoque es soportar a la mayoría de las características exportados por Libvirt-CIM, que permite la gestión de múltiples plataformas con un solo proveedor. Por lo tanto, Libvirt-CIM es un conjunto de herramientas para gestionar las siguientes plataforma de virtualización: Xen, QEMU, KMV, OpenVZ y LXC. Libvirt-CIM también proporciona acceso remoto a MVs a través de conexiones cifradas y autentificadas.

La idea básica detrás de Libvirt-CIM es combinar la capacidad de gestión de plataformas de virtualización heterogéneos que proporciona Libvirt (utilizando una serie de ficheros XML para describirlos) con el enfoque de gestión estandarizado provisto por CIM. Por lo tanto, Libvirt-CIM ofrece un envoltorio para obtener toda la potencia de Libvirt en la gestión de plataformas heterogéneas utilizando clientes CIM convencionales.

A pesar de lo expuesto en el párrafo anterior, durante esta investigación, se ha observado que no todas las características anunciadas han sido implementadas por Libvirt-CIM. Además, puesto que su objetivo es aplicar el modelo basado en la virtualización disponible en el DMTF-CIM, no ha existido ninguna contribución adicional relacionada con el modelo. Por otra parte, no fue diseñado para implementar un EVR completo, sino más bien funciona mediante MVs individuales.

2.5.5.2 VMware-CIM SDK

Por otra parte, existe una implementación del modelo de extensión VMware-CIM, llamado VMware-CIM SDK, que proporciona una interfaz CIM a los desarrolladores para que construyan aplicaciones de gestión. Según [VMware-CIM], los desarrolladores pueden utilizar las aplicaciones estándar compatible con CIM para administrar los servidores ESX, MVs y sus dispositivos de almacenamiento. VMware-CIM SDK también incluye un CIMOM llamado *Pegasus* instalado junto con la infraestructura de gestión de VMware. Con VMware-CIM SDK, es posible explorar las MVs en la máquina del servidor ESX y examinar el almacenamiento físico asignado a cada

MV. Además, es posible explorar los entornos VMware de forma remota mediante consultas a través de *Pegasus*.

2.5.5.3 Xen-CIM

Otra implementación trascendente se conoce como Xen-CIM. El proyecto Xen-CIM trabaja como una implementación de código abierto de un proveedor para los perfiles DMTF- CIM SVPC para administrar Xen, que puede ser utilizado con *Pegasus, SLIM* u *OpenWBEM* u otro CIMOM). Según [Griffyn06], el primer modelo inicial de Xen-CIM no definía claramente la gestión de los recursos virtuales más allá de la memoria y la CPU. Este modelo carecía de la capacidad de manejar la asignación de recursos virtuales. Sin embargo, las últimas versiones del modelo Xen-CIM [Mellor08] muestran que los recursos son agregados en grupos (pools) facilitando su gestión. Sin embargo, este modelo carece de generalidad, dado que se centra en el entorno Xen. Otra cuestión importante en relación con el proyecto Xen-CIM es decidir cuál de los APIs subyacentes se debe usar, ya sea directamente con el API de C o con el de Libvirt-CIM. El proyecto Xen CIM tiene previsto utilizar Libvirt-CIM, puesto que se trata de un API estándar para la gestión de varias tecnologías de virtualización en la que está incluida Xen. Sin embargo Libvirt-CIM introduce otra capa de código lo cual posiblemente no sea lo más funcional.

2.5.6 Análisis comparativo

Para formalizar la evaluación de los diferentes enfoques de modelado y de algunas implementaciones analizadas hasta aquí, se ha considerado definir los siguientes criterios como aspectos fundamentales de la gestión de las redes y puntos de vista del modelado de información. Una breve descripción se proporciona a continuación:

- ***Gestión del Ciclo de Vida***: El enfoque de modelado o la implementación gestiona el ciclo de vida de las MVs: crear, destruir, suspender, reanudar, y funciones de configuración de MVs;
- ***Asignación de recursos***: El enfoque de modelado o la implementación establece el patrón básico de asignación de recursos para las agrupaciones de recursos, las asignaciones, y los datos de ajuste;
- ***Monitoreo de Red Virtual***: El enfoque de modelado o la implementación tiene una interfaz de gestión de virtualización para monitorear el tráfico o el rendimiento de la red;
- ***Monitorización de MV***: El enfoque de modelado o la implementación tiene capacidad para controlar la disponibilidad de: RAM, los ciclos de la CPU y espacio en disco duro;
- ***Dispositivos de almacenamiento de disco y Herramientas de Gestión***: El enfoque de modelado o la implementación incluye capacidades para gestionar los recursos de

almacenamiento basados en estándares abiertos;

- ***Gestión de Acceso Remoto***: El enfoque de modelado o la implementación dispone de capacidades para acceder remotamente a MVs;
- ***Interoperabilidad***: El enfoque de modelado o la implementación tiene la capacidad de soportar diferentes plataformas de virtualización;
- ***Seguridad***: El enfoque de modelado o la implementación tiene la capacidad de trabajar con técnicas de cifrado o autenticación;
- ***Despliegue automático de un EVR***: El enfoque de modelado o la implementación incluye la posibilidad de desplegar automáticamente un EVR completo. Este criterio no debe ser confundido con el relacionado a la gestión del ciclo de vida de MVs individuales. Por el contrario, está relacionado con la gestión del despliegue de EVR en conjunto (que se compone de varias MVs interconectados en topologías arbitrarias).

La tabla 2-5 muestra los resultados obtenidos de esta evaluación. Los perfiles de modelado de virtualización DMTF-CIM son los enfoques de modelado más completos, por lo que ha sido considerado base de esta Tesis. En el caso de las implementaciones, Libvirt-CIM es la solución más completa para administrar MVs, redes virtuales y el almacenamiento. Sin embargo, como se muestra en esta evaluación, ninguno de los enfoques tratados en este documento proporciona funcionalidad en relación con el despliegue automático de EVR en su conjunto. Por lo tanto, en concreto, esta Tesis ha sido direccionada para contribuir a la industria en la solución de esta limitación en particular.

Tabla 2-5 Evaluación de enfoques de modelado de infraestructuras de virtualización

PARÁMETRO	DMTF-CIM	OVF	VMWARE-CIM	LIBVIRT-CIM	VMWARE-CIM-SDK	XEN-CIM
Gestión del ciclo de vida	☑	☑	☑	☑	☑	☑
Asignación de recursos	☑	☑	☑	☑	☑	⟐
Monitoreo de Red Virtual	☒	☒	☒	☐	☒	☒
Monitorización de MVs	☑	☑	☑	☑	☑	☑
Dispositivos de almacenamiento	☑	☑	☑	☑	☑	☑
Gestión de acceso remoto	☒	☒	☑	☑	☑	☒
Interoperabilidad	☑	☑	Solo ESX	☑	Solo ESX	Solo Xen
Seguridad	☒	☑	☒	☑	☒	☐
Despliegue Autom. de un EVR	☒	☒	☒	☒	☒	☒

⟐: El modelo no está claro en la especificación del tipo de los recursos
☐: En pruebas o en desarrollo
☑: El modelo o la aplicación disponen de la característica.
☒: El modelo o la aplicación carecen de la característica.

2.6 Técnicas de despliegue de EVR en entornos distribuidos

Este apartado describe el analisis del Estado del Arte desde el punto de vista del modelo de infraestructura del despliegue. Esto es importante puesto que se requiere definir una arquitectura base que permita el despliegue automático de EVR en entornos distribuidos. Por tanto, en este analisis se ha incluido la Computación Grid y las Organizaciones Virtuales que se crean. Luego una capa de Virtualización, en el sentido de crear entornos de MVs sobre los recursos que gestiona el Grid.. Luego se incluye WSRF para implementar servicios Grid [Kumar] y finalmente RM-ODP como marco conceptual.

2.6.1 Computacion Grid

De acuerdo a Foster en [Foster08], Grid es cualquier entorno en el que se cumplan las siguientes condiciones: *coordinación de recursos que no estén sujetos a control centralizado*, lo que implica la integración de recursos y usuarios que pertenecen a diferentes dominios de administración; *uso de estándares abiertos y protocolos de propósito general*, que se encargan de las tareas básicas de autenticación, autorización y descubrimiento de recursos y acceso a los mismos; y, por último, la *obtención de calidades de servicio no triviales*, relacionadas con el tiempo de respuesta, disponibilidad, seguridad y asignación de recursos.

Dentro de Grid se incluyen formas de colaboración entre personas u organizaciones geográficamente distribuidas que se agrupan y asocian para compartir recursos comunes (aplicaciones, procesos, dispositivos y equipos), dando lugar a organizaciones virtuales. En este punto conviene diferenciarlas de los escenarios de red virtuales, en los que se aplican conceptos de partición de hardware y aislamiento, proporcionados por las tecnologías de virtualización.

La arquitectura Grid aporta los mecanismos para interactuar con los recursos, implementando por una parte, *mecanismos de búsqueda* (que permitan el descubrimiento del estado de los recursos), y por otra, *mecanismos de gestión de recursos* (que permitan la provisión de un determinado nivel de calidad de servicio y su control) [Foster01]. Adicionalmente, en cuanto a seguridad, Grid define un núcleo de comunicaciones y protocolos de autentificación requeridos en transacciones de red y proveen mecanismos criptográficos seguros para verificar la identidad de usuarios y recursos [García06].

2.6.2 Tecnologías de Virtualización y su integración con Grid

La virtualización provee una capa de abstracción que puede ser aplicada en entornos de computación distribuida [Figuereido2003]: Las tecnologías de Virtualización permiten particionar

un equipo físico en múltiples máquinas virtuales (con un sistema operativo hospedado en cada una de ellas), compartiendo los recursos hardware del equipo anfitrión, tales como CPU, memoria y dispositivos de I/O (entrada y salida).

2.6.3 Web Service Resource Framework (WSRF)

WSRF es un conjunto de especificaciones diseñadas para fusionar aplicaciones Grid y tecnologías de servicios Web en el marco conceptual de OGSA (*Open Grid Service Architecture)*, definiendo los detalles técnicos de los Servicios Grid.

WSRF fue aprobado como estándar de OASIS (*Organization for the Advancement of Structured Information Standards)* en abril de 2006 [WSRF08]. Define un marco para modelado y acceso al estado de los recursos utilizando servicios Web. WSRF proporciona los mecanismos para la transformación hacia un servicio Web con estado (WS-Resource), el cómo manipular dicho servicio a fin de que pueda crear y administrar recursos, cómo anunciar esos recursos al mundo exterior y cómo utilizarlos para lograr más funcionalidad.

Un *WS-Resource* [Czajkowski04], se define en WSRF como la composición de un servicio Web más el estado de un recurso *(Stateful Resource)* que: ***i)*** es un conjunto específico de valores expresable en un documento XML (*eXtended Markup Language*); ***ii)*** tiene un ciclo de vida bien definido; y ***iii)*** tiene identificación, por lo que puede ser objeto de actuación de uno o más servicios Web. En concreto un *WS-Resource* tiene varias propiedades y los valores de estas propiedades definen el estado de los recursos.

WSRF está definido por las siguientes especificaciones: *WS-Resource Lifetime*, que proporciona mecanismos para administrar el ciclo de vida de un recurso; *WS-Resource Properties* que especifica como se define el acceso a las propiedades; *WS-RenewableReferences*, que renueva el *WS-Addressing* cuando la referencia actual se convierte en inválida; *WS-Service Group*, que define una vía para crear una agrupación de servicios Web así como el registro y disponibilidad de los mismos; y *WS-Base Fault,* que define un mecanismo para indicar errores.

Adicionalmente, aunque no formen parte de WSRF, existen dos especificaciones relacionadas: *WS-Notification*, que define un servicio Web para publicar las notificaciones y *WS-Addressing*, que proporciona mecanismos para direccionar servicios Web.

WSRF ha permitido modelar el ciclo de vida del recurso "EVR", caracterizándolo como un *WS-Resource*, y ha facilitado los detalles para implementar una interfaz usando un lenguaje de programación orientado a objetos.

2.6.4 RM-ODP

Es un marco de referencia para el desarrollo de aplicaciones distribuidas, para soportar de forma integrada aspectos tales como la distribución, interoperabilidad y transparencia de los componentes, objetos o recursos, requeridos en nuestra solución. De acuerdo con [Foster05], el núcleo central del modelo de referencia RM-ODP está recogido en cuatro normas básicas: visión de conjunto, fundamentos, arquitectura, y semántica arquitectural. Estas reglas establecen algunos conceptos fundamentales: ***i)*** la especificación de un sistema en términos de diferentes *puntos de vista*, que están interrelacionados entre sí; ***ii)*** el uso de un *modelo de objetos común* para la especificación del sistema desde cada uno de los puntos de vista; ***iii)*** la definición de una infraestructura que permita ocultar ciertas complejidades inherentes a los sistemas distribuidos (*transparencia*); y ***iv)*** la definición de un conjunto de *funciones comunes* que son de utilidad para la construcción de sistemas abiertos y distribuidos.

El papel clave de RM-ODP es proveer una referencia sólida para la especificación de la arquitectura en la que el soporte de transparencia de distribución, interconexión y portabilidad pueda ser integrado, facilitando el diseño e implementación de un sistema distribuido.

2.7 Conclusiones

En este capítulo, se ha abordado a manera de síntesis el marco teórico que sustenta esta Tesis Doctoral. Para su comprobación, se ha implementado una plataforma de experimentación virtual (EVR) para el despliegue de redes, medidas de rendimiento y prestación de servicios. Se ha evaluado cuantitativamente algunas plataformas de virtualización, midiendo el consumo de recursos de CPU y memoria. Como resultado se determinó que Xen sería la mejor herramienta de virtualización para implementar tal escenario de red. Una mención importante se ha dado a VNUML y Netkit que proporcionan un lenguaje de definición para crear y desplegar EVR, reduciendo los costos de implementación y experimentación. Se concluye además que todas las plataformas de virtualización evaluadas proveen facilidades para la funcionalidad de red, por lo que en gran medida su rendimiento dependerá del diseño del EVR a implementar. Estos resultados han sido publicados en [Fuertes07].

Por otro lado, se ha analizado y evaluado varios enfoques de la infraestructura de modelado y de las implementaciones de virtualización. En esta evaluación se ha determinado que CIM del DMTF es el enfoque primordial, aunque en realidad no cubre el modelado y el despliegue de EVR en su conjunto. Así mismo, tal como se muestra en el apartado 2-4, se han desarrollado enormes esfuerzos por la comunidad científica para resolver varios problemas inherentes a la gestión de las tecnologías de virtualización utilizando modelos de información, que corroboran la pertinencia y oportunidad de nuestra propuesta. Esta comparación ha sido publicada en [Fuertes0410]

Adicionalmente se han analizado diversas alternativas de despliegue de EVR, cara a definir una arquitectura de despliegue en entornos distribuidos. Se ha analizado una infraestructura de Computación Grid y su integración con entornos virtualizados, asi como WSRF y RMODP.

Entre los principales problemas detectados en este análisis se aprecia que existe una gran cantidad de herramientas de virtualización en el mercado, algunas de ellas que presentan mayor dificultad en la configuración y despliegue de los EVR. Además no todas ellas (excepto las aquí evaluadas) disponen de funcionalidades de conectividad y networking.

En el siguiente capítulo será verificada la funcionalidad de los EVR y la validación de la capacidad de prestación de servicios de red en estos entornos.

Capítulo 3 Validación de la prestación de servicios en entornos virtualizados.

3.1 Introducción

En este capítulo, como aplicación y verificación de la utilidad de los EVR, se presenta la emulación del servicio de video streaming bajo demanda (VoD) en una conexión ADSL (*Asymmetric Digital Subscriber Line*), pero ejecutándose sobre la plataforma de virtualización Xen, herramienta que en la evaluación realizada en la Tesis (véase apartado 2.3) consumió menos CPU y memoria. Este servicio ha sido elegido ya que ofrece prestaciones relacionadas con conceptos parametrizables y medibles en redes, como son ancho de banda, cantidad de transmisión, retardo, variabilidad en el retardo, calidad de servicio (QoS), etc., que son factibles de cuantificar en EVR. Para tales fines se conceptualizó un método incremental para aumentar la precisión de los resultados obtenidos en un EVR. Para tales propósitos se capturo tráfico de video en la conexión ADSL y se realizaron algunos experimentos descritos en los siguientes apartados. Finalmente con los resultados de estos experimentos y el análisis del tráfico capturado, se intentó ajustar dichos parámetros utilizando diversas herramientas de código abierto y de libre distribución (véase apartado 3.2 al 3.4).

Puesto que en estos experimentos se evidencia la presencia de la penalización causada por la capa de virtualización y con el fin de determinar otro método de validación de estos experimentos, en este capítulo se presenta una evaluación del rendimiento de redes IP utilizando plataformas de virtualización comparándola con una herramienta de simulación como es NS-2 (véase apartado 3.5)

3.2 Declaración del problema y motivación

Las plataformas de Virtualización son una tecnología potencial para reproducir una topología de red real utilizando un escenario virtual. Estas plataformas permiten ejecutar y probar múltiples entornos de validación de software [Mathews07], y brindan facilidades para realizar el dimensionado de prestación de servicios de redes.

Hasta la fecha existían al menos dos formas para realizar el dimensionado de redes a la hora de desplegar nuevos servicios y aplicaciones: Una alternativa consistía en montar otra infraestructura de red en paralelo, pero esto exigiría contar con nuevos equipos y dispositivos de hardware, lo cual encarecería la solución. La otra alternativa consistía en utilizar herramientas de simulación, como el NS (*Network Simulator*) [NS2-08] u otros, que son usados para evaluar el rendimiento de las

redes. No obstante, los simuladores intentan reproducir el funcionamiento de un sistema real (retardo, paquetes perdidos, etc.) utilizando software, pero no son capaces de reproducir con precisión la funcionalidad del hardware [Pizzonia08] de un sistema real (nuevos dispositivos, configuraciones, arquitecturas).

Frente a estas dos alternativas, este trabajo propone la utilización de plataformas de virtualización para evaluar la prestación de servicios de red. Según los resultados obtenidos en [Muñoz08], utilizando las tecnologías de virtualización son muy aproximados a los derivados en un escenario real. Sin embargo, la tecnología de virtualización genera una sobrecarga de rendimiento causada por la capa de virtualización [Barham03]. Esto significaría que la virtualización todavía no ha madurado para asegurar resultados lo suficientemente cercanos a la prestación de servicios de red en entornos reales.

En este contexto, en este capítulo se propone aplicar un método para mejorar los resultados obtenidos en los EVR, tratando de parecerse a los obtenidos en entornos reales. En particular, nuestro objetivo principal es obtener un resultado lo más cercano posible a un caso real. Estas mejoras proporcionan estrategias para interactuar con aplicaciones de tiempo real y para garantizar su prestación de servicios en condiciones similares antes de su puesta en producción. Además, estas mejoras podrían mejorar la percepción de aceptación por parte de los usuarios.

Para llevar a cabo, en primer lugar, se ha diseñado e implementado las topologías de prueba requeridas para hacer las comparaciones entre los entornos virtuales y reales. En segundo lugar, se ha ajustado los parámetros del EVR para emular los aspectos operacionales del entorno real. Con estos resultados, se ha encontrado nuevas directrices para ajustar los parámetros de emulación en los EVR.

Antes de describir el método incremental, es importante describir algunas razones por las que se ha elegido VoD como servicio y el VLC como interfaz de video. Primero, el video streaming es un método para transferir datos digitales en tiempo real. Este proceso convierte el vídeo y audio en un formato digital comprimido, y luego distribuye los datos a través de las redes informáticas. Luego, el vídeo puede ser pre-almacenado, (vídeo bajo demanda), o puede ser codificado en tiempo real (videoconferencia) [Apostolopoulos02]. Además, el VoD es un servicio ampliamente utilizado. Finalmente, se ha elegido VoD con el objetivo de tratar de medir sus parámetros de red y emular a este servicio utilizando los entornos virtuales de redes IP.

En lo referente a la interfaz de video, se ha seleccionado la solución VideoLAN [Vlc], porque es software libre, tanto para la transmisión de vídeo bajo demanda como para vídeo en vivo. Además, el formato de vídeo puede ser cambiado. VideoLAN es también un reproductor multimedia portátil que funciona tanto para Microsoft Windows y Linux y se puede utilizar diversos medios de almacenamiento y transmisión.

En esta investigación la solución VideoLAN, básicamente, tiene dos funciones principales: ***i)***

como servidor para transmitir archivos de vídeo de diferentes formatos al cliente, y ***ii)*** como un cliente, para controlar la comunicación entre el cliente y el servidor, e implementar las funciones de video clásicas (play, pause, next, etc.).

3.3 Experimentos de emulación

Esta sección describe el método desarrollado para mejorar los resultados obtenidos en los EVR. Este método se ha dividido en cinco pasos, como se muestra en el siguiente diagrama de flujo (véase Fig. 3-1). Su diseño, detalles de implementación, configuración y evaluación son la base de esta investigación y se explica en los párrafos siguientes:

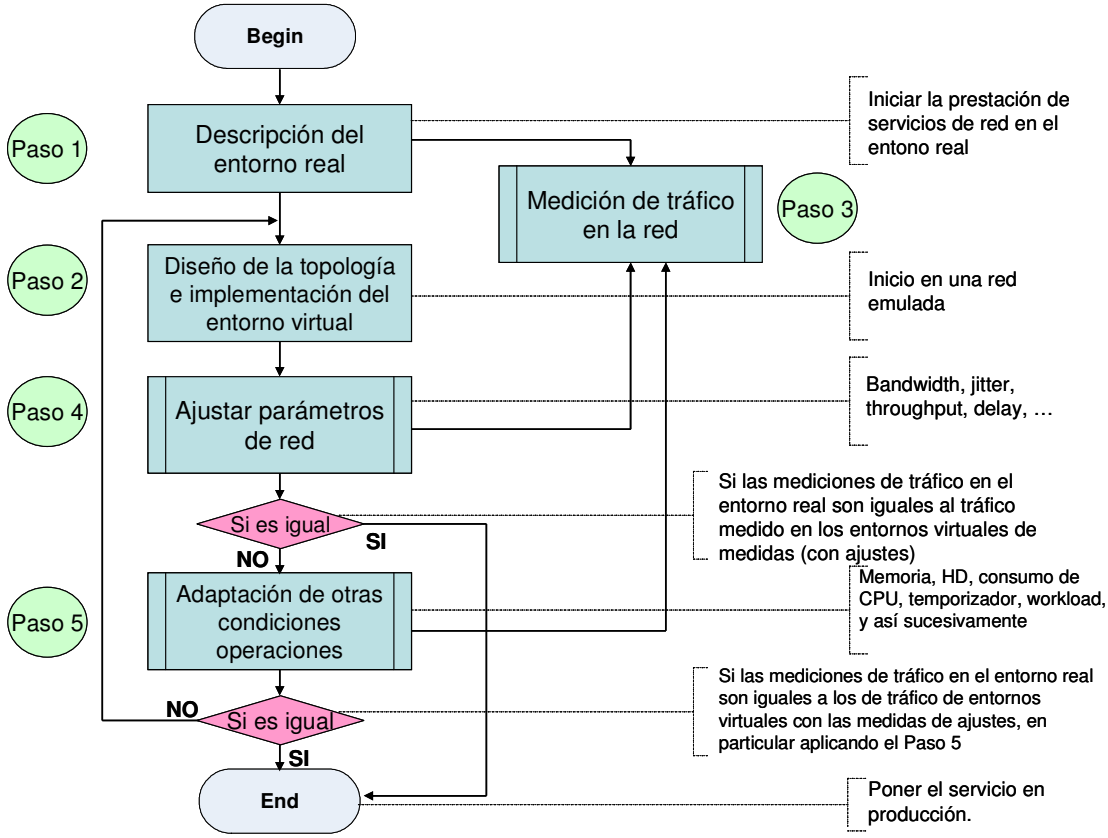

Figura 3-1 Diagrama de flujo para implementar el método para mejorar los resultados obtenidos en los entornos virtuales de red

3.3.1 Paso 1: Descripción del entorno real

Tal como se indica en el diagrama de flujo, el primer paso es describir el entorno real y describir el experimento (véase Fig. 3-2). En esta investigación se ha seleccionado el VoD en una conexión ADSL. ADSL fue elegido porque ha sido específicamente diseñado para explotar la naturaleza bidireccional de la mayoría de las comunicaciones multimedia. A continuación se probó la emulación de este servicio frente a una conexión ADSL clásica de 1,2 Mbps.

Antes de proceder al paso 2, es importante mencionar que en esta investigación no consideró la

emulación de diferentes características de transmisión de las conexiones ADSL, concernientes a la última milla (por ejemplo, la longitud del cable, la atenuación de la señal, o de usuarios concurrentes). Nos se ha centrado en el ancho de banda y el retardo, y se supone que otras propiedades ya han sido cubiertas por estas. Teniendo en cuenta los resultados obtenidos, este supuesto puede considerarse aceptable (véase apartado 3.4).

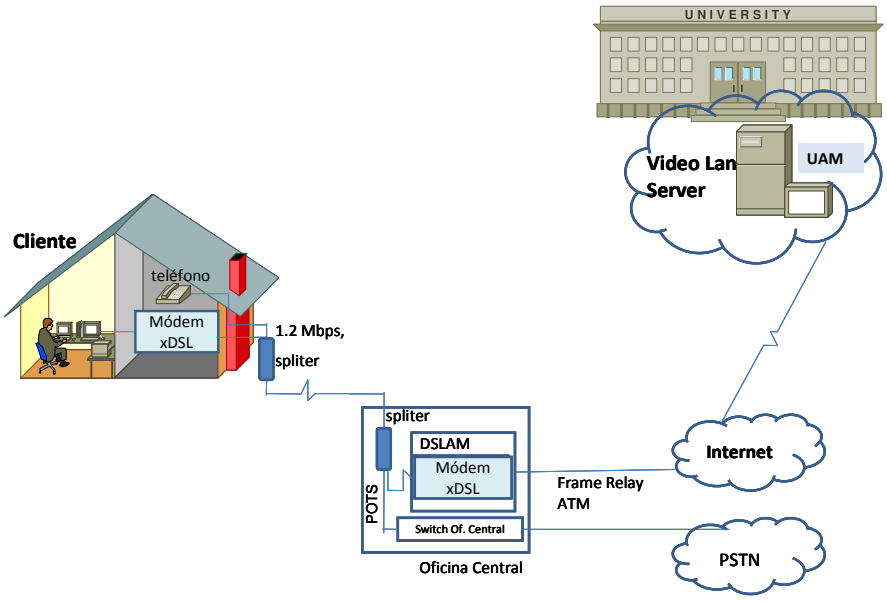

Figura 3-2 Entorno de pruebas de una conexión ADSL

3.3.2 Paso 2 Diseño e implementación de un entorno virtual de red

Como segundo paso en el método se ha diseñado e implementado el entorno virtual de red que se muestra en la Fig. 3-3. En este entorno, en el mismo equipo ubicado en el Laboratorio de la UAM, por un lado, se ha instalado y configurado el servidor de VideoLAN en una MV. Por otro, se instaló el reproductor multimedia VideoLAN como cliente (en el mismo host anfitrión), para mostrar el vídeo, especialmente teniendo en cuenta la necesidad de manejar un entorno gráfico y considerando que Xen tiene limitaciones en tales requisitos. Luego, se ajustó el enlace entre el Router2 y Router1 para que sea similar al caso de una conexión ADSL.

El siguiente método ha sido utilizado para implementar el diseño propuesto en un EVR (véase Fig. 3-3): En primer lugar se ha creado la primera MV e instalado el OS huésped Linux Debian. Entonces, se clonó de la primera MV a las MVs restantes, con el fin de reducir el tiempo de instalación. En este punto, cabe señalar que los routers que se muestran en la Fig. 3-3 son MVs a las que se le asignó la funcionalidad de los dispositivos de enrutamiento. Después de esto, se ha añadido interfaces virtuales, configurado direcciones IP y se han levantado servicios. Luego, se ha sincronizado el reloj de cada MV con el protocolo NTP (*Network Time Protocol*) en el interior del host real. Esto permitió que las MVs estén sincronizadas con la hora del sistema. Esto es muy

necesario porque las MVs (routers y en este caso) trabajan mediante la compartición del tiempo del hardware físico del host, puesto que una MV no puede duplicar exactamente el comportamiento de tiempo de una máquina física. Por último, se ha creado y ejecutado los scripts respectivos que automáticamente construyen y despliegan el entorno de experimentación. Como punto final, se ha instalado el software para el monitoreo de tráfico, tanto en el entorno de red virtual como en el host. También vale la pena mencionar que todos los experimentos se realizaron utilizando el software de código abierto y de libre distribución.

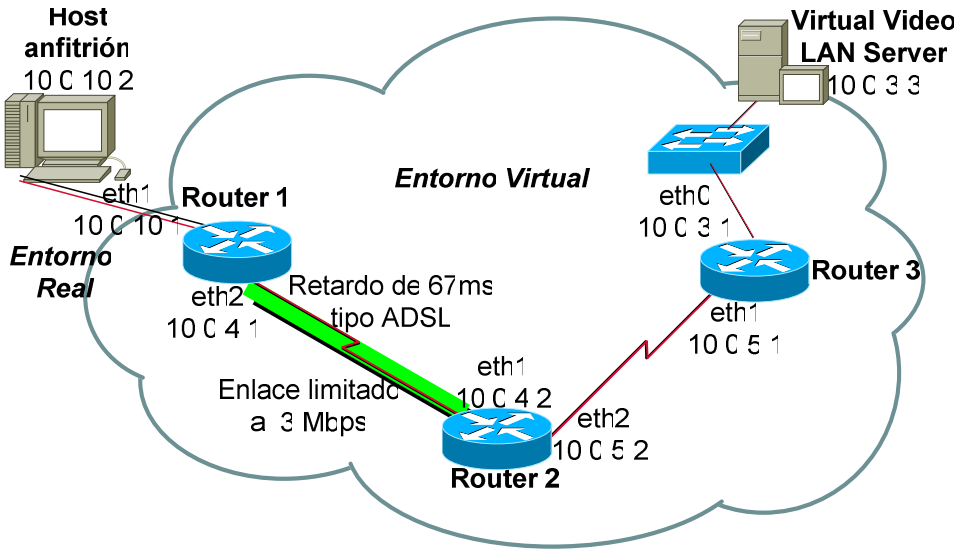

Figura 3-3 Entorno de red virtual utilizado para emular VoD a través de una conexión ADSL

3.3.3 Paso 3: Medición del Tráfico de la Red

Una vez que fueron implementados ambos entornos, se tomaron las mediciones de tráfico de red apropiadas de acuerdo con el paso 3 de este método. Han sido transmitidos archivos de vídeo, tanto en el entorno real (véase Fig. 3-2) como en el virtual (véase la Fig. 3-3) en dirección descendente (descarga) desde el servidor de VideoLAN hacia el cliente. Se han realizado varias pruebas para confirmar que la captura de tráfico de datos sea libre de errores.

Para capturar el tráfico de los dos experimentos que se describen a continuación, se utilizó Tcpdump [Jacobson] y se ha deshabilitado el modo promiscuo en las interfaces correspondientes. Tcpdump es una herramienta de línea de comandos cuya utilidad principal es analizar el tráfico que pasa a través de la red. Los registros obtenidos fueron visualizados con Wireshark [Wireshark], que es un analizador de protocolos de red. A continuación, se aplicaron filtros para crear archivos de texto plano que fueron procesados con una secuencia de comandos para obtener la función de distribución acumulativa (CDF, *Cumulative Distribution Function*) del tiempo entre llegada de los paquetes de vídeo. Estas distribuciones de probabilidad se han utilizado para contrastar los resultados, siguiendo la metodología presentada en [García07] para comparar el rendimiento de los

servicios multimedia en redes IP.

a) Experimento en el entorno ADSL Real

El primer experimento consistió en la captura de vídeo en tiempo real, de trafico RTP *(Real Time Protocol)* en el entorno de ADSL real, durante la transferencia de vídeo (véase Fig. 3-2). Esta captura se realizó de forma simultánea tanto en el cliente en casa y en el servidor ubicado en el laboratorio de la UAM, para obtener la CDF del tiempo entre llegada de los paquetes de video.

Para obtener un valor apropiado del retardo entre el servidor VideoLAN y el cliente ADSL se aplicó el comando ping con paquetes de tamaño fijo de 1370 bytes. Ping mide el tiempo de ida y vuelta RTT (*Round-Trip Time Delay*) utilizando el protocolo de control de mensajes de Internet (ICMP). Con esto, sobre la base de 4300 paquetes que se envían durante más de 45 minutos se obtuvo una media (μ) de 135,6 ms, con una correlación r (0) 40% y una desviación estándar (σ) de 17ms para el RTT. Cabe señalar que la pérdida de paquetes era insignificante en la medición de tráfico.

Se utilizó RTT en lugar de OWD (*One way delay*) que podría ser una mejor opción para ADSL, dado que OWD debe ser calculado entre dos nodos sincronizados, que no era posible establecer en el entorno real en el experimento que se llevó a cabo

b) Experimento en el entorno virtualizado

El segundo experimento consistió en la captura de tráfico de RTP de vídeo en el EVR (véase la Fig. 3-3). Todas las pruebas fueron llevadas a cabo en un solo host (Pentium D, 2,80 GHz, 1 GB de RAM) con Linux Debian 4.0, y una de las particiones ext3 con 120 GB. En todas las máquinas virtuales fueron instalados los mismos sistemas de ficheros y los mismos kernel (2.6.18-xen-686-GNU/Linux Debian).

3.3.4 Paso 4: Ajuste de parámetros de entorno virtual de red

De acuerdo con el paso 4 de este procedimiento, se ajustaron los valores de los parámetros obtenidos en el experimento antes de transferir el vídeo, para calcular la CDF del tiempo entre llegadas de paquetes en este entorno y contrastar los resultados. A partir de estos valores el retardo se configuro a 67,8 ms (debido a la asimetría del ADSL, se ha suponiendo una aproximación simétrica end-to-end y se ha tomado la mitad de la RTT obtenido para el caso real). Además, se ha limitado el ancho de banda en el enlace desde el router 2 al router 1 a 1,2 Mbps. También era posible emular el enlace ascendente ADSL con un menor ancho de banda, pero el experimento no era necesario porque el servicio de VoD es unidireccional (es decir, desde el servidor de VideoLAN hacia los clientes).

Para emular los parámetros de red (ancho de banda y retardo de extremo a extremo) en la

interfaz eth1 de Router1 1 y Router2 del EVR se utilizaron las siguientes utilidades de control de tráfico Linux: i) *tc traffic control* [Brown06] que abarca el conjunto de mecanismos y operaciones por las que los paquetes están en la cola de transmisión / recepción en una interfaz de red, ii) el emulador de red (NetEm) [Hemminger06], que es una mejora de las facilidades de control de tráfico de Linux que permite añadir retardo, pérdida de paquetes y otros parámetros. En este punto se debe indicar que se ha utilizado esta herramienta para emular el retardo de extremo a extremo similar al obtenido en un entorno de ADSL, y, iii) HTB (*Hierarchical Token Bucket*), que es una disciplina de la cola para limitar la velocidad de ancho de banda. Como se muestra en la Fig. 3-2, la conexión de ultimo-salto ADSL a un ancho de banda clásico de 1,2 Mbps, lo que debe ser ajustada en el EVR. Una vez realizados estos cambios, el tráfico de vídeo fue capturado en el EVR, a fin de realizar las comparaciones descritas en el apartado 3.3.3.

3.3.5 Paso 5: Adaptación de otras condiciones de funcionamiento

Como último paso del método propuesto deben ser analizadas las siguientes condiciones de operación del sistema donde se ejecuta el experimento de emulación. Esto permitirá identificar los mecanismos que pueden ayudar a mejorar el método.

a) Servidor dedicado

Los experimentos anteriores se llevaron a cabo en un servidor dedicado sólo para el servicio de VoD. Cuando se ha añadido otro servicio, tal como la transferencia de archivos en el EVR, se incrementó la degradación del rendimiento. En resumen, la degradación en el tiempo de respuesta puede ser explicado porque los recursos de la CPU son compartidas en el sistema. Entre otras soluciones, este problema puede ser resuelto mediante la adición de la cantidad apropiada de los recursos [Fernando06]. Pero la mejor solución debe ser equilibrar las cargas de trabajo en máquinas virtuales.

b) Resolución del temporizador

El sistema operativo usa temporizadores para proporcionar una multitud de servicios. En el caso del EVR, puesto que hay varias MVs ejecutándose en una plataforma única, existen una variedad de enfoques para asignar el reloj virtual en el reloj de la plataforma física, y cualquiera de estos puede causar desplazamiento de reloj. Además, cualquier cambio en el comportamiento del reloj podría dar lugar a errores en el cálculo del rendimiento medido. Debido a problemas de reloj, las mediciones de la utilización en EVR no son fiables [Fernando06]. En cualquier caso, para mejorar las mediciones, el EVR debe ser capaz de obtener acceso a los temporizadores de alta resolución. Los temporizadores de mayor resolución son necesarios para que el sistema procese los datos a intervalos más precisos. Una manera de aplicar es mediante la adición de parches para introducir un

nuevo subsistema, por ejemplo, contadores de tiempo del núcleo de alta resolución *hrtimers Linux (*resolution kernel timers).

Por otra parte, ya que las MVs deben compartir los recursos de la misma CPU (siempre por el anfitrión real) y, sobre todo debido a la complejidad de la emulación de hardware virtual, tales características son muy sensibles al tiempo, dando lugar a imprecisiones en el tiempo el sistema en las máquinas virtuales. Este hecho ha sido bien documentado en [Marshal06] [VMware08]. Por otra parte, la instalación de software específico para plataformas de virtualización, denominado *Adiciones* (Additions, como en el caso de VirtualBox), es una mejor solución para mejorar este sincronismo [Marshal06].

c) Otros parámetros de rendimiento

Además de las mediciones tradicionales que ya han sido probadas, la medición de otros parámetros de rendimiento también deben ser considerados, tales como: el número de usuarios, el número de máquinas virtuales, las cargas de trabajo el tiempo de respuesta, el consumo de CPU y memoria, y así sucesivamente. Además, como última sugerencia, el número de CPUs virtuales debe ser igual al número de CPUs de núcleo físico en el entorno real, y, la cantidad de memoria física asignada a la máquina virtual debe ser igual a la memoria física en el entorno real [Casazza06]. Esto crearía una comparación mas justa.

3.4 Resultados experimentales y discusión

3.4.1 Comparación entre el servidor de ADSL y el servidor virtual

La Fig. 3-4 muestra la CDF del tiempo entre llegada de paquetes el servidor ADSL real y el servidor virtual. Es evidente que las distribuciones de probabilidad son similares, aunque existe cierta disparidad debido a la sobrecarga producida por la capa de virtualización.

3.4.2 Comparación entre el cliente ADSL y el cliente de EVR.

La Fig. 3-5 muestra las CDF del tiempo entre llegada de paquetes para el cliente de ADSL, el cliente virtual sin ajustes, y el cliente virtual con ajustes. Esta figura ilustra que el cliente virtual sin ajuste es completamente diferente de los resultados experimentales del cliente de red virtual con ajustes. Una vez más las trazas de distribución de probabilidad de los dos entornos son visualmente similares Esto pone de manifiesto el logro de nuestro método. Sin embargo, como se observa en la Fig. 3-4, las CDFs no son exactamente iguales en el lado del servidor, y con mayor dificultad en el lado del cliente.

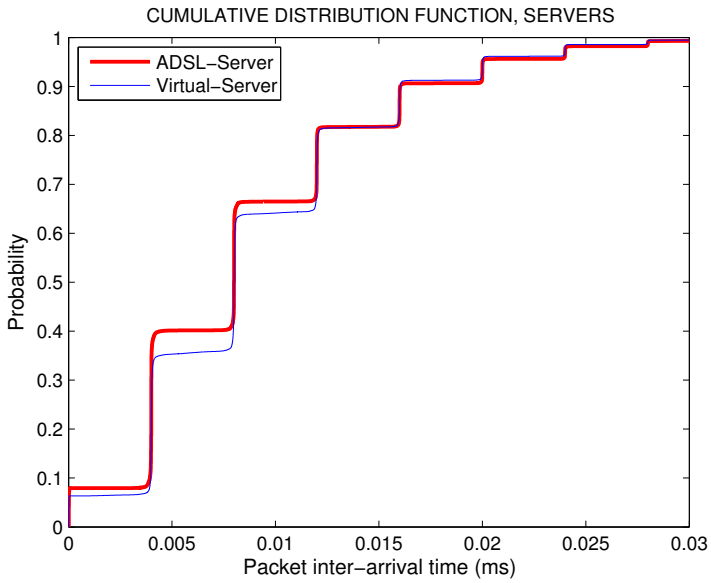

Figura 3-4 Diferencias entre las CDFs del tiempo entre llegada de paquetes entre el servidor ADSL y servidor virtual

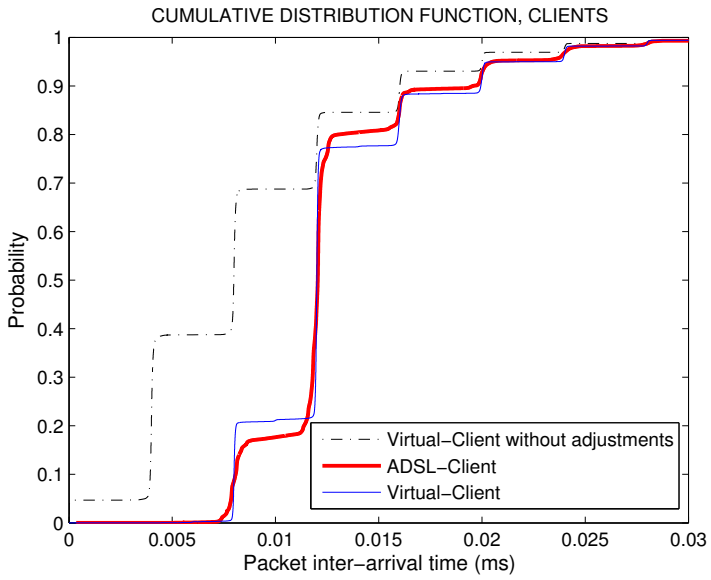

Figura 3-5 Diferencias entre las CDFs del tiempo de llegada de paquetes de video entre el cliente de ADSL, el cliente virtual con ajustes, y el cliente virtual sin ajustes.

3.4.3 Divergencia Kullback-Leibler

Para comparar los resultados obtenidos en nuestros experimentos, se ha calculado la divergencia de Kullback Leibler [Kullback51], que mide la diferencia entre una CDF real *r* y una CDF arbitraria *v*, tal como se define en la ecuación (1):

$$Divergence(R \parallel V) = \int_{-\alpha}^{\alpha} r(x) \log \frac{r(x)}{v(x)} dx \qquad (1)$$

Donde r(x) y v(x) expresan las CDFs de R(cliente de ADSL real) y de V (cliente virtual) del tiempo entre llegadas de de paquetes de variables aleatorias.

La evaluación de la ecuación (1) para los archivos CDF de los clientes ADSL real y virtual con el cliente ajustado da una divergencia=0.0473. Además, si el CDF de los clientes ADSL real se compara con el CDF del cliente virtual sin ajustes da un resultado de divergencia = 1,9972. Como se esperaba, estos resultados muestran que la disparidad es mayor cuando los parámetros no se ajustan en el cliente virtual.

3.5 Trabajos Relacionados

Existen muy pocos trabajos que analizan los resultados de emulación de servicios que utilizan entornos de virtualización. Una primera investigación similar ha sido descrita por Fernando en [Fernando06]. Aquí, el autor explora los beneficios de la virtualización y comenta las dificultades de medir los resultados de forma análoga al mundo real. Este trabajo mide la utilización de recursos tanto del núcleo del monitor de máquina virtual (VMM) como de las máquinas virtuales. Nuestro objetivo, sin embargo, se centra en la búsqueda de pautas para lograr mejores resultados en las mediciones de desempeño en el ERV.

Una segunda investigación similar ha sido descrita por Casazza et al. [Casazza06]. Aquí, los investigadores explican una metodología para caracterizar el rendimiento de la carga de trabajo de las tecnologías de virtualización de servidores para consolidar múltiples servidores físicos. Se presenta un ejemplo de referencia usando el servidor Web, servidor de correo electrónico y el servidor de base de datos. Nuestra investigación, sin embargo, se centra en proporcionar una infraestructura de red para llevar a cabo experimentos de dimensionamiento. Además, se ha basado en los parámetros de rendimiento tradicional. Por último, se hizo la comparación directa con los datos obtenidos entre los dos entornos real y virtual.

Un tercer trabajo comparable tercero ha sido explicado por Song et al. [Song07]. En este trabajo los autores estudian la interacción de VoD y otros servicios empresariales en un contexto típico de consolidación de servidores. Ellos comparan sus resultados usando puntos de referencia. En comparación con este esfuerzo, nuestro objetivo es proporcionar datos cualitativos asociados a cómo los servicios de red trabajan en entornos de virtualización, cómo los EVR reproducen este comportamiento y cómo imitar los resultados lo más cerca posible al caso real.

En cuanto a la emulación de servicios de red a través de EVR, en [Maier07] se provee bancos de pruebas para proporcionar un entorno de red virtual configurable para las mediciones de rendimiento comparativo de implementaciones reales. En el mismo contexto en [Galán06] direccionan la emulación de servicios a través de técnicas de virtualización de un banco de pruebas de un subsistema multimedia IP (IMS) destinadas a la validación funcional de los servicios.

Además, la investigación de Raj et al., en [Raj07] presenta el marco de trabajo *Vmedia* (Virtualización de multimedia), para compartir dispositivos multimedia entre múltiples máquinas virtuales. En comparación con nuestra investigación, estos tienen como meta principal la implementación de topologías complejas, siendo el rendimiento un objetivo secundario, a diferencia de nuestra investigación, cuyo objetivo es encontrar un método para mejorar el rendimiento en los resultados obtenidos en EVR.

Por último, otra investigación similar se puede encontrar en [Baumgartner03], donde se evalúa el rendimiento de routers virtuales para la investigación de redes. Sin embargo, nuestra investigación tiene algunas diferencias. Además de la utilización de las MVs Xen, se ha comparado los resultados del EVR con los obtenidos en un entorno real, obteniéndose valores similares al aplicar nuestro método para mejorar los resultados

3.6 Evaluación del Rendimiento de Redes IP utilizando Plataformas de Virtualización y Métodos de Simulación

Este apartado ha sido incluido puesto que describe un mecanismo adicional para contrastar los resultados obtenidos al realizar experimentos con plataformas de virtualización comparándolas con los métodos de simulación.

Las plataformas de Virtualización y los Métodos de Simulación constituyen dos tecnologías prominentes en el ámbito de la investigación, que son utilizadas para medir el rendimiento de las redes IP. Por una parte, las Plataformas de Virtualización permiten crear escenarios de redes virtuales que emulen equipos interconectados entre sí, que son usados para pruebas de software, emulación de prestación de servicios en redes y una variedad de aplicaciones [Fuertes09b]. Sin embargo su principal limitación es la penalización generada por la capa de Virtualización que modifica la precisión de los resultados experimentales [Fuertes07]. Por otra parte están los métodos de simulación que proveen un entorno repetible y controlable para imitar el funcionamiento de una red experimental durante un intervalo de tiempo. Los métodos de simulación 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 tuvo dos propósitos: Primero, diseñar, implementar y poner en funcionamiento escenarios simulados (mediante NS-2 [NS2-08]) y virtualizados (mediante Xen [Barham03]) a fin de validar el rendimiento de redes IP. Segundo, realizar varios experimentos utilizando estas dos tecnologías a fin de verificar como se va degenerando el rendimiento a medida que se incrementan equipos en la red. Dentro de este contexto, no se han identificado suficientes evidencias

relacionadas, sin embargo Muñoz en [Muñoz06], utilizó OPNET y VNUML [Galán09] para analizar la calidad de servicio en los servidores Web, cuyos resultados muestran una importante aproximación. Cabe mencionar que existen varios trabajos de simulación [Flores06] y virtualización [Mathews07] pero de manera aislada que no contemplan la realización en conjunto del dimensionado de redes [Falcon07].

Para llevar a cabo esta investigación, se diseñó e implementó diferentes escenarios de prueba utilizando NS2 y Xen en la misma plataforma de hardware y software base. Luego se configuraron aquellos parámetros que se relacionan con el rendimiento de red como ancho de banda, latencia y pérdida de paquetes. 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 su comportamiento.

Por tanto las principales contribuciones de esta investigación fueron: ***i)*** proveer de un estudio de las divergencias existentes entre los resultados al medir el rendimiento en las dos tecnologías citadas; y ***ii)*** la verificación de cómo se degenera el rendimiento de la red en ambos entornos al ser sometida a otras condiciones de forma que a partir de los resultados se pueda extrapolar cuánto error ha de haber en un experimento que se haga posteriormente. 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. En este sentido, se ha detectado que en el caso de la Simulación, los parámetros deben ser rigurosamente programados para mejorarlos. En el caso de la Virtualización, se deben adaptar otras condiciones operacionales como servidores dedicados, temporización [Fuertes09a] la mejora del hardware base.

En el caso de existir algún interés en los fundamentos teóricos, la configuración del experimento, y los resultados obtenidos, favor véase el **Apéndice A**.

3.7 Conclusiones

En este capítulo, se ha implementado un método para mejorar los resultados obtenidos en EVR, en contraste con los obtenidos en entornos reales. Se ha realizado una serie de experimentos emulado un servicio de VoD real de ADSL en un entorno virtual de red con Xen. Los resultados experimentales han mostrado cierta similitud en el tiempo de llegada de paquetes entre los dos entornos en el lado del servidor y del lado del cliente. Sin embargo, las distribuciones de probabilidad no son precisamente las mismas, ya que el entorno de red virtual presenta una sobrecarga no cuantificada y porque el retardo ADSL emulado es una aproximación. En cualquier

caso, los resultados del experimento han proporcionado datos cualitativos en relación a cómo funcionan los servicios, la calidad percibida del servicio de VoD, las configuraciones necesarias, y así sucesivamente. Para concluir, se ha presentado un procedimiento para emular los servicios de red en entornos de red virtual, haciendo hincapié en los factores que afectan a los resultados experimentales. Este procedimiento y sus resultados han sido publicados en [Fuertes08a] y [Fuertes09a]

Como complemento en este mismo capítulo, se diseñó, implementó y puso en funcionamiento 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 ancho de banda, 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 como se produce la degeneración de la red a medida que incrementan los equipos. 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. Finalmente, se ha observado que en el caso de la Simulación, los parámetros deben ser rigurosamente programados para mejorarlos. En el caso de la Virtualización, se deben adaptar otras condiciones operacionales como servidores dedicados, temporización, otras métricas de rendimiento y el mejoramiento del hardware base. Estos resultados han sido publicados en [Fuertes09c]

Capítulo 4 Propuesta para predecir el *overhead* producido en la capa de virtualización

4.1 Introducción

En este capítulo se ha tratado de cuantificar la sobrecarga introducida por la capa de virtualización para una topología de prueba dada. Para llevar a cabo esta investigación, se ha diseñado e implementado varios experimentos con diferentes topologías de prueba utilizando dos plataformas de virtualización típicas como VMware Server 3.0.0 [VMware] y Xen 3.0.3 [Barham03]. VMware ha sido utilizada en razón de ser una de las plataformas de mayor consumo de CPU y memoria en un EVR, mientras que Xen es la plataforma que obtuvo un consumo muy cercano al de un equipo real (véase Capítulo 2, apartado 2.3). En particular, se ha trabajado en la parte, en el que el rendimiento se ve afectado, por lo que a partir de estos resultados pueda ser extrapolado cuánto error existirá en un experimento posterior.

Para formalizar estos resultados, se ha aplicado el método de *regresión lineal* para modelar la relación entre una o más variables que afectan el rendimiento. Asi mismo, la regresión lineal proporciona un mecanismo para establecer una ley que puede derivar una expresión analítica que, en este caso, podría caracterizar el *overhead*. En este contexto, como una contribución, esta investigación presenta una expresión analítica, que podría predecir el comportamiento de *overhead* en un EVR dado.

4.2 Declaración del problema y motivación

Las tecnologías de virtualización generan una sobrecarga en el rendimiento causada por la capa de virtualización, lo que reduce la precisión de los resultados experimentales y por lo tanto limita la credibilidad en su aplicación. La necesidad de cuantificar el *overhead* se ha estudiado durante varios años. Por ejemplo, Cherkasova, [Cherkasova05], evaluaron la sobrecarga causada por el procesamiento de operaciones de I/O utilizando Xen [Barham03]. Además, en [Menon05], los autores analizaron la penalización del rendimiento efectuado por las aplicaciones de red que se ejecutan en MVs de Xen. Otros trabajos de investigación [Wood08] [MVMark06] [Padala07] [Zibitsker09] [Friedman07] estiman las necesidades adicionales de recursos efectuados por el *overhead* de virtualización con métodos estadísticos y desde varios puntos de referencia (micro benchmarks). En estos casos, han intentado medir la sobrecarga de rendimiento, pero aún no han sido capaces de predecir la sobrecarga causada por la capa de virtualización en un entorno de red

virtual (EVR).

La falta de conocimiento sobre los detalles específicos de la penalización del rendimiento es un problema crítico, ya que los resultados obtenidos en un EVR no son absolutamente confiables. Por lo tanto, es adecuado analizar los factores que aumentan esta sobrecarga. Con este fin, se ha tratado de cuantificar la sobrecarga introducida por la virtualización. Pero, ¿cómo se traducen estas mediciones para predecir el *overhead* de virtualización para una topología de prueba dada? Teniendo en cuenta estos problemas, esta sección propone medir el *overhead* en EVR. .

4.3 Fundamentos Teóricos

4.3.1 Medición del rendimiento computacional versus la medición del rendimiento de un EVR.

De acuerdo con Jain [Jain91], para evaluar el rendimiento de un sistema informático existen algunos indicadores comunes, tales como: tiempo de falla, baja variabilidad, no redundancia, capacidad nominal, tiempo de respuesta, capacidad utilizable, tiempo de entrega, el factor de estiramiento, throughtput, eficiencia, consumo de CPU y memoria, fiabilidad, probabilidad de errores, disponibilidad y el *overhead*. Aunque todas estas medidas son útiles para un sistema computacional, no todas pueden ser utilizadas para medir el rendimiento de un EVR. Esto es razonable ya que el rendimiento de la red se refiere a la calidad de servicio de un producto de telecomunicaciones y debe ser medida desde el punto vista de la satisfacción del cliente. En este caso, las mediciones deben hacerse en términos de latencia y el rendimiento.

Por otro lado, antes de tomar cualquiera de las medidas descritas en el párrafo anterior, es necesario identificar los factores que afectan el rendimiento en un EVR. Entre los más importantes que se pueden mencionar son: ***i)*** Las políticas de planificación, gestión de procesos y gestión de la memoria de los Sistemas Operativos, ***ii)*** Los componentes de hardware del sistema, tales como la velocidad de reloj, la arquitectura y la tecnología de fabricación, y el tamaño de la memoria caché de nivel 2; ***iii)*** El número de núcleos de procesador, y finalmente, ***iv)*** La distribución de la carga de trabajo (balanceo de carga).

Finalmente, es importante identificar las técnicas que están disponibles en la medición de rendimiento. Entre los más conocidos podemos incluir: monitoreo del sistema; referencias de comparación (benchmarking), modelado con sistemas reales, modelado de rendimiento y la simulación de sistemas.

Con todos estos argumentos, en el ámbito de esta investigación, para medir el *overhead* en un EVR, se evaluará la utilización de la CPU durante la ejecución de la transferencia de archivos. Para ello, se ha considerado dos factores que pueden afectar al rendimiento: la carga de trabajo entre un cliente y un servidor y diferentes anchos de banda en las conexiones.

4.3.2 *Overhead* de rendimiento en la virtualización

Las tecnologías de virtualización proporcionan una abstracción de los recursos de hardware permitiendo múltiples instancias de Sistemas Operativos para ejecutarse de forma simultánea en un único servidor físico. Esta particularidad introduce un nivel adicional de abstracción y, por consiguiente, *overhead* adicional. Técnicamente, el *overhead*, significa la cantidad de tiempo de procesamiento utilizados por el software del sistema disminuyendo el rendimiento. En el contexto de esta investigación, el *overhead* se entiende como la diferencia de costes entre ejecutar una aplicación dentro de un EVR y ejecutar la misma aplicación en un entorno real.

Según Casazza [Casazza06], algunas de las cuestiones clave del *overhead* de virtualización incluyen: ***i)*** un SO invitado puede no informar en tiempo exactos; ***ii)*** las herramientas de monitoreo de rendimientos son muy limitadas, ***iii)*** multitud de configuraciones, ***iv)*** contención de recursos con otros equipos virtuales (MV), y ***v)*** la coherencia de los resultados. Otros enfoques consideran varios factores que aumentan el *overhead*. Por ejemplo, algún *overhead* puede ser causado por dispositivos de I /O como tarjetas de interfaz de red, actividad de la unidad de disco duro [IntelTR08]. Otros parámetros claves pueden estar asociados con el número de MVs e incluso de su respectiva configuración individual (número de procesadores y cantidad de memoria). Por otra parte, el *overhead* de virtualización depende del número de MVs y el perfil de carga de trabajo, ya que cada MV implica una sobrecarga importante para el servidor físico en función del OS invitado. Por último, el *overhead* de virtualización puede ser causado por operaciones que no puedan ser ejecutados directamente en el hardware y las asignaciones adicionales que se utilizan para proporcionar la MV con un entorno normal.

Finalmente, de acuerdo con nuestras investigaciones anteriores [Fuertes07], la selección de una específica tecnología de virtualización, puede afectar el *overhead* de rendimiento (véase apartado 2.3). Además, cualquier diferencia en el hardware del sistema puede afectar el rendimiento en virtualización. En consecuencia, la sobrecarga podría implicar que la sobrecarga en una MV puede degradar otra MV.

4.4 Configuración experimental

Esta sección describe la arquitectura de prueba implementada con el fin de medir el *overhead* de rendimiento producido en un EVR. Básicamente, esta arquitectura consiste en el despliegue de dos EVR, el primero mediante Xen, y el segundo mediante VMware, en el que se ha llevado a cabo varios experimentos. El experimento consistió en diseñar e implementar una topología de prueba con dos máquinas virtuales conectadas por un puente Ethernet. Como se muestra en la Fig. 4-1 (a), un entorno Web cliente/servidor ha sido emulado utilizando Xen. La primera MV es el cliente y la segunda MV es el servidor Web virtual. La misma topología fue probada en otra máquina física

usando VMware Server, como se muestra en la Fig. 4-1 (b).

El objetivo principal es medir el *overhead* del rendimiento de la CPU producida por la capa de virtualización. Para validar estas medidas también se ha realizado la misma prueba, pero en este momento, la topología se basa en equipos reales (véase Fig. 4-1 (c)). Todos estos experimentos fueron realizados usando los mismos parámetros experimentales tales como el número de conexiones, diferentes cargas de trabajo, y diferentes anchos de banda. Cada MV se ha configurado con dos procesadores, 96 MB en RAM y 4 GB en un disco duro. Además, varios parámetros experimentales fueron configurados en el experimento de base (véase Fig. 4-1 (d)).

Para emular el servidor Web se ha instalado y configurado el servidor Web Apache para ejecutar en el servidor Web virtual, el cual responde a las peticiones requeridas por los clientes de cada uno de los EVR. Luego, se ha utilizado la herramienta *httperf* [Mosberger98], que permite generar múltiples solicitudes de cargas de trabajo desde los clientes *HTTP* al servidor Web (es decir, en este caso una MV de cada EVR) proporcionando un mecanismo flexible para medir el desempeño. Httperf informa detalles del porcentaje de uso de la CPU, el rendimiento del sistema y de usuario de la red, y las estadísticas de solicitudes y respuestas durante la descarga.

Para garantizar la fiabilidad de los resultados, 5000 solicitudes simultáneas se hicieron desde cada MV hacia el servidor Web virtual, tanto con Xen como con VMware configurado desde *httperf*. El tamaño de los archivos de carga de trabajo varía de 1KB, 10KB, 20KB, 30KB, 50KB, 70KB y 100KB. La tasa de transmisión cambia incrementalmente desde 20kbps a 100 Kbps. Entonces, se ha implementado un único algoritmo en shell script, que nos permitió medir el consumo de la CPU y el rendimiento durante la ejecución de la transferencia. Como se muestra en las Figs. 4-1 (a), (b) y (c), una vez desplegado el entorno, se ha iniciado la transferencia de archivos para recoger las medidas en la fase de ejecución, utilizando una herramienta de *httperf*, ajustando algunas opciones para evitar tiempos de inactividad, el trabajo en segundo plano y la actualización de cada segundo, durante la transferencia.

Todos los experimentos se realizaron en dos máquinas de escritorio con procesador Intel Core ™ 2 Quad CPU de 2,4 GHz., 3,23 GB de RAM, 100/1000 interfaz PCI tarjeta de red, y 100 GB a 7200 RPM de disco IDE. En estas mediciones, se utilizó Ubuntu Server 8.10, Kernel 2.6.27.5 en una partición ext3 con 60 GB. Ambos equipos de escritorio se utiliza para instalar Xen en un lado y VMware en el otro. Así mismo, cada computadora de escritorio se utilizó para instalar el cliente real y el servidor Web real.

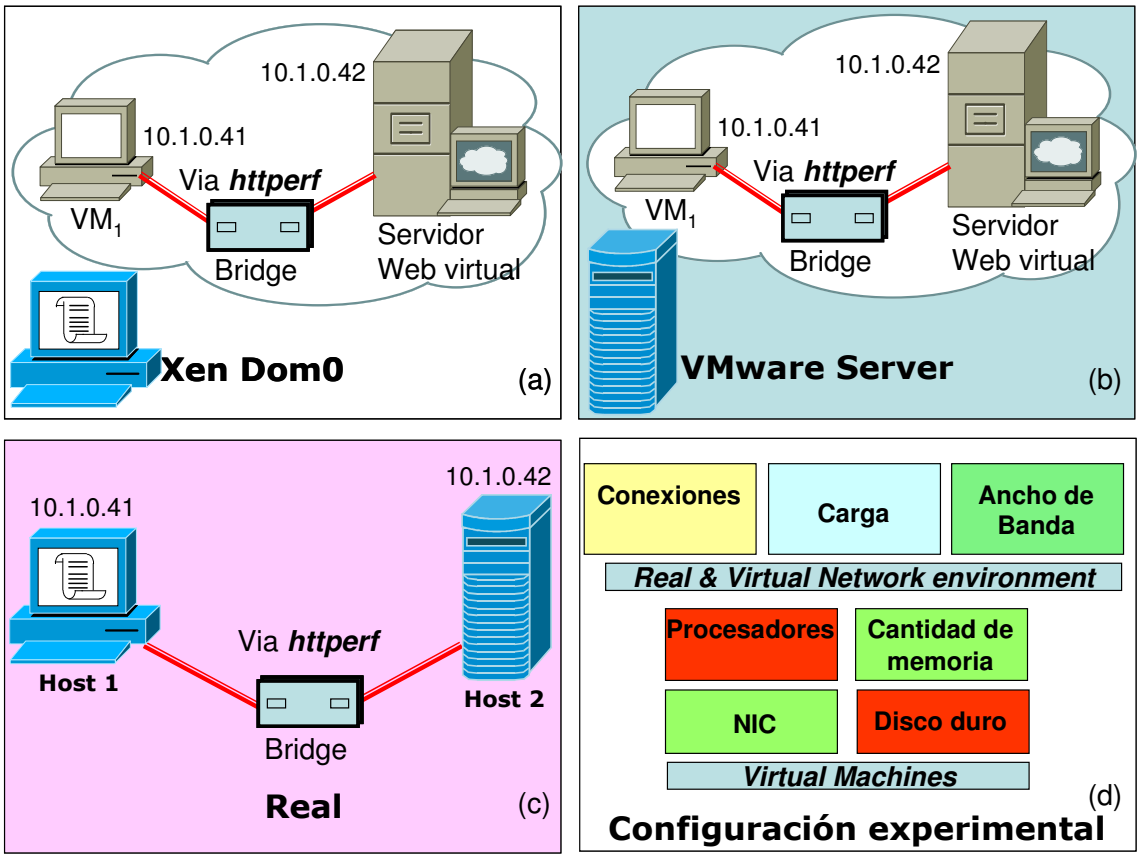

Figura 4-1 (a) Topología de prueba para emular un Web Server utilizando Xen; (b) Web Server utilizando VMware; (c) Web Server con equipos reales; (d) configuraciones y ajustes para la evaluación del experimento.

4.5 Resultados Experimentales

La Fig. 4-2 muestra el grafico de dispersión de la media de los datos recogidos, calculado de once pruebas realizadas con cada herramienta de virtualización y la topología real, cuando la carga de trabajo ha sido un archivo de 50 KB. Estos resultados muestran que Xen tiene un consumo de CPU que es, aproximadamente cercano al derivado del rendimiento del caso real. Por el contrario, VMware consume mucho más CPU en comparación con el caso real, como consecuencia, el *overhead* es mayor.

La Fig. 4-3 muestra el consumo de CPU de las tres topologías de prueba, cuando fueron sometidos tanto a diferentes cargas *HTTP* desde las MVs hacia el servidor Web virtual; como con diferentes velocidades de transmisión (ancho de banda). Como se puede observar, los comportamientos en el entorno Xen son aproximadamente cercanos a los resultados derivados del rendimiento real, por el contrario VMware tiene más diferencias en relación con el entorno real.

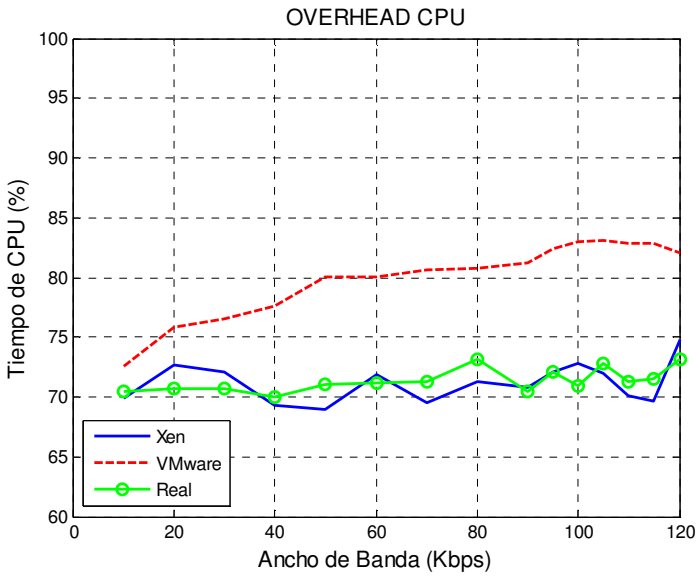

Figura 4-2 El consumo de CPU de las tres topologías de prueba durante la transferencia de la carga de trabajo con un archivo de 50 KB

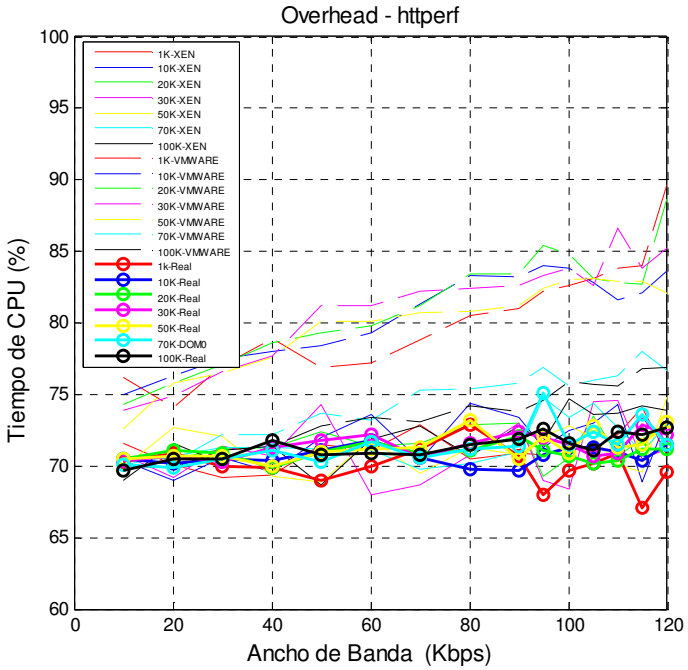

Figura 4-3 El consumo de CPU de las tres topologías de prueba durante la transferencia, con diferentes cargas de trabajo y diferentes anchos de banda

4.6 Modelado Estadístico y Verificación

El siguiente procedimiento se ha utilizado para extrapolar los resultados obtenidos a partir de las topologías de prueba anteriores (Real, Xen y VMware):

- Se tomaron varias muestras experimentales estadísticas en los tres escenarios descritos anteriormente;

- Se han definido claramente las variables de respuesta: En primer lugar, 1, 10, 20, 30, 50, 70 y 100 KB, que es la carga de trabajo generado por cada MV en cada experimento, y en segundo lugar, las variables independientes, de 20, 40, 60, 80, 100 y 120 Kbps, que constituyen el ancho de banda. Luego, se ha procesado esta información utilizando MatLab y el Scientific Workplace 5.5, por su facilidad de cálculo;
- Se ha aplicado el método de *Regresión Lineal* para modelar la relación entre las variables. La *regresión lineal* proporciona un mecanismo para establecer una ley de la que se puede derivar una expresión analítica que en este caso va a caracterizar el *overhead*. La *regresión lineal* se ha utilizado para ajustar estas variables, generando el análisis matemático utilizando la curva de mejor ajuste y verificando estos resultados por el método de mínimos cuadrados;
- Posteriormente, las medidas de tendencia central y de dispersión fueron analizadas para proporcionar un mayor argumento de verificación matemática.

Este procedimiento ha sido explicado en términos generales. Ahora, particularizaremos este procedimiento para esta investigación de la siguiente manera: Cuando se realiza un primer tratamiento de datos de la topología de prueba real, con una carga de trabajo de 1KB, esto corresponde al siguiente modelo de *regresión lineal*, como puede observarse en la Fig. 4-4.

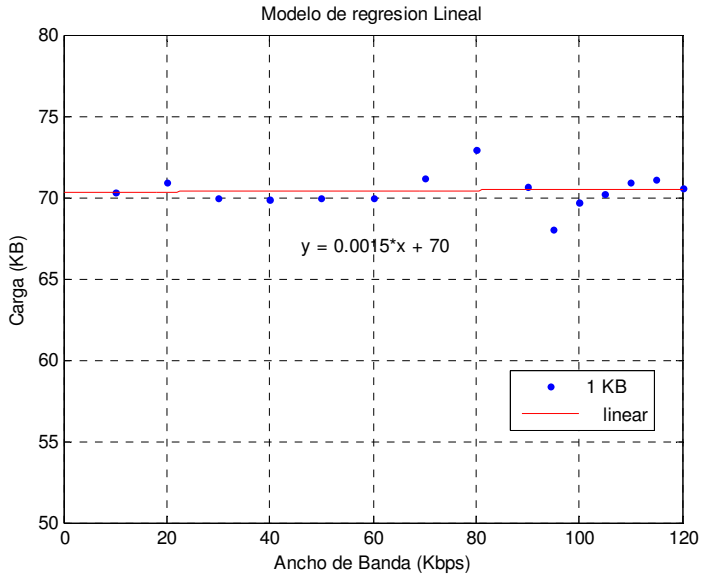

Figura 4-4 Modelo de Regresión lineal propuesto

En este punto, cabe señalar que, dado que no todos los puntos de dispersión de la Fig. 4-4 siguen un comportamiento lineal, fue necesario verificar contra modelos matemáticos de mejor ajuste hasta el polinomio de quinto grado de precisión y la variabilidad. Aquí se ha determinado que las diferencias no son significativas (sólo unas décimas), por lo que se determinó que la aplicación de la *regresión lineal* es una gran opción para el presente estudio.

La expresión analítica lineal es:

$$y = \beta_o + \beta_1 x$$

El modelo anterior no tiene en cuenta el error. Por lo tanto, se debe incorporar el modelo lineal de probabilidad para variables aleatorias. Por lo tanto, una mejor propuesta de este estudio es la siguiente:

$$y = \beta_o + \beta_1 x + \varepsilon$$

Cuyos elementos son:

y: Es la variable a modelar o es la variable dependiente (carga de la CPU);
x: Es la variable que representa el ancho de banda y es la variable independiente;
ε : Es el componente aleatorio de error;
β_o : Es el punto de intersección donde se cruza la recta con el eje y (cargas);
β_1 : Representa la pendiente de la línea.

Con las implicaciones anteriores se asume que el error aleatorio tiene una distribución normal con media=0 y varianza $=\sigma^2$. La variable independiente, (ancho de banda) es el predictor y la variable dependiente (carga del CPU de los equipos con tamaño de archivo diferentes) se conoce como la variable de respuesta. En el análisis de regresión lineal es necesario tener en cuenta los pasos anteriores descritos en este apartado, para estimar un buen modelo matemático que se ajuste a los resultados de los experimentos en los entornos virtuales y real. Por analogía se ha realizado el mismo procedimiento para todas las cargas, obteniendo un modelo de regresión lineal para el caso real que se ilustra en la Fig. 4-5.

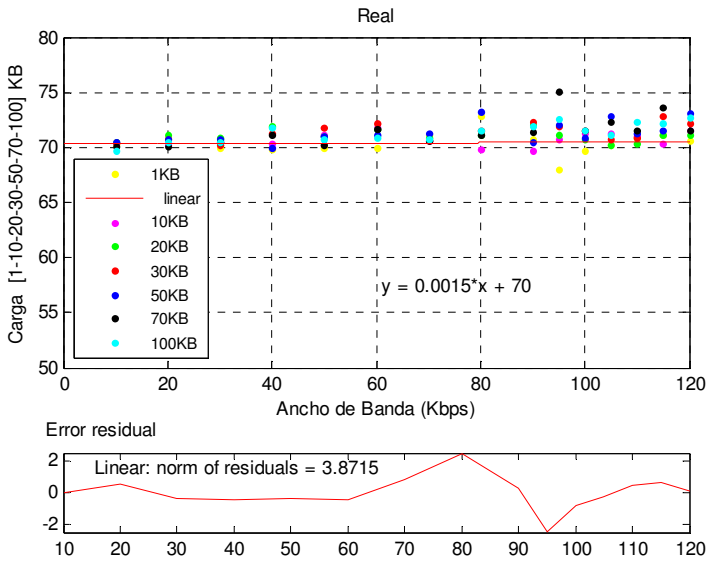

Figura 4-5 Modelo de regresión lineal para el entorno real y su error residual

Ahora se debe repetir el mismo procedimiento para Xen y VMware (véase Fig. 4-6 y 4-7).

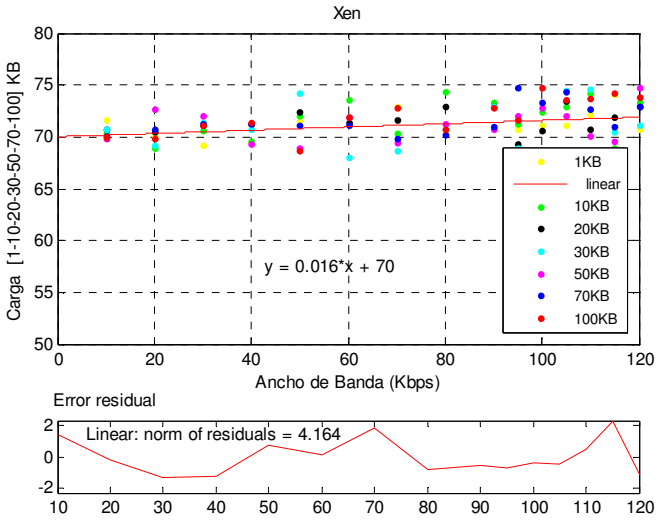

Figura 4-6 Modelo de regresión lineal para el entorno virtual con Xen y su error residual

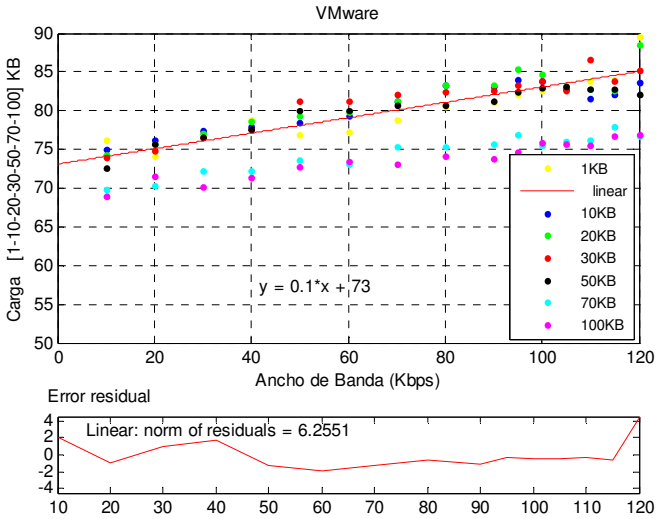

Figura 4-7 Modelo de regresión lineal para el entorno virtual con VMware y su error residual

La Tabla 4-1 muestra el resumen de los errores residuales, así como otros elementos para establecer las comparaciones estadísticas entre los diferentes entornos.

Tabla 4-1 Estimación del error residual

Entorno	y = mx + b	Error residual
Real	y=0.0015x+70	3.8715
Xen	y=0.016x+70	4.164
VMware	y=0.1x+73	6.2551

Como se muestra en la Tabla 4-2, en el entorno real existe un error residual de 3.8715. En el EVR utilizando Xen hay un error residual de 4.164. En el EVR utilizando VMware presenta un error residual de 6.2551. Por lo tanto, podemos deducir estadísticamente que en el caso de Xen, es un pronóstico excelente para un entorno real.

La Tabla 4-2 muestra las medidas de tendencia central y medidas de dispersión para complementar nuestro análisis:

Tabla 4-2 Medidas de tendencia central y de dispersión.

Estadígrafo	Ancho de Banda	Real	Xen	VMware
min	20	67.1	69.2	74.1
max	120	72.9	74.1	89.6
mean	73	70.03	71.17	80.39
median	80	70	71.1	80.5
Standard Deviation	36.19	1.356	1.251	3.974
Range	110	5.8	4.9	15.5
Residual	0,00	3.8715	4.164	6.2551
Coefficient of Var.	0.496	0.019	0.018	0.049

Como se muestra en la Tabla 4-2, al comparar el coeficiente de variación (CV) se observa que el entorno virtual utilizando Xen es muy cercano al entorno real, es decir: Real-CV=0.019 y Xen-CV=0.018. Por el contrario, VMware-CV=0.049, es decir es peor en comparación con el entorno real. Esto indica que el entorno virtual utilizando Xen ha producido menos errores que el entorno virtual utilizando VMware. En cuanto al análisis de la desviación estándar entre el entorno real=1.356, el entorno virtual utilizando Xen=1.251, y el entorno virtual de VMware=3.974, de forma análoga al procedimiento anterior, se puede deducir que el entorno virtual Xen es mejor que el entorno virtual en VMware relación con el entorno real. Como resultado de lo anterior, la expresión analítica de mejor predicción es la siguiente:

$$y=0.016x+70 \quad (1)$$

El modelo lineal obtenido es una expresión analítica aplicable para predecir la carga de trabajo que se producirían en un experimento determinado, utilizando EVR con Xen. Por ejemplo, usando la misma configuración establecida en apartado 4.4, pero a una velocidad de transferencia no incluida en el experimento, ¿cuál sería la carga de trabajo si el ancho de banda es de 15 Kbps?

Reemplazando *x* en la ecuación (1), se tiene:

$$y=0.016*(15)+70;$$
$$y=70.24 \text{ KB}.$$

Como punto final, es importante señalar que en este modelo lineal se ha incluido el error residual Epsilon ε, por lo tanto, se infiere que el modelo de regresión lineal es aceptable para nuestro estudio. Por último, también vale la pena mencionar que las expresiones analíticas obtenidos pueden aplicarse para predecir el *overhead* calculados en la evaluación de un servicio en un EVR utilizando Xen.

4.7 Trabajos relacionados

Hay muy pocos documentos que discuten sobre la penalización producida por los EVR. Aquí se han incluido algunas de las investigaciones más importantes relacionadas en la literatura, que fueron el punto de inicio de este estudio.

La investigación propuesta por Cherkasova et al., en [Cherkasova05], presenta un sistema de monitoreo para medir la sobrecarga del CPU causado por el procesamiento I/O usando Xen. Este estudio intenta cuantificar el rendimiento y analizar esta sobrecarga para un conjunto de cargas de trabajo intensivas de I/O. En el mismo contexto Menon et al., en [Menon05], analizaron la penalización de rendimiento efectuados por aplicaciones de red ejecutándose en MVs Xen. Comparado con nuestra propuesta, estos trabajos no intentan predecir la sobrecarga causada por la capa de virtualización.

Una importante investigación similar fue propuesta por Wood et al., [Wood08]. Aquí, los autores intentan estimar los requerimientos de recursos adicionales incurridos por la penalización de la virtualización utilizando un modelo de regresión lineal. Este modelo puede ser usado para estimar las necesidades de recursos de cualquier aplicación a ser virtualizada en una plataforma determinada. En comparación con nuestra investigación, su enfoque fue en base a un conjunto seleccionado de micro-puntos de referencia (micro-benchmarks) el cual puso a prueba los recursos del sistema en diferentes tasas de tráfico I/O y luego empleó estos perfiles de uso para predecir la variable de consumo del CPU en entornos virtualizados.

En cuanto a la medición del rendimiento virtualizado, hay algunas referencias disponibles. Por ejemplo, las obras propuestas por [VMarck] [Padala07], han utilizado benchmarking para evaluar el rendimiento de los entornos virtualizados. Estos benchmarks intentaron proporcionar una base para la comparación de hardware y plataformas de virtualización en ejercicios de consolidación de servidores. Sin embargo, ambos carecen de la capacidad de caracterizar sobrecarga de virtualización comparada con una plataforma nativa. Algunos trabajos más recientes [Zibitsker09] [Friedman07], tratan de medir el impacto del rendimiento en la consolidación de servidores y la aplicación de la virtualización en el rendimiento de las aplicaciones. En estos casos, el papel de la modelización es evaluar cómo las decisiones específicas pueden afectar al rendimiento y la escalabilidad de las cargas de trabajo individuales. Su trabajo tiene en cuenta cómo el tiempo de servicio del CPU para cada carga de trabajo se verán afectados por la sobrecarga de CPU del hipervisor, y cómo el tiempo de respuesta de I/O para cada carga de trabajo se extenderá debido a los retrasos causados por el aumento del tiempo del planificador I / O en un hipervisor.

4.8 Conclusiones

En este capítulo, se ha diseñado e implementado tres topologías de prueba utilizando dos plataformas de virtualización típicas como son VMware y Xen. Luego, se han comparado sus resultados frente un entorno de red real. Se ha evaluado los experimentos con diferentes cargas de trabajo y diferentes anchos de banda. Se ha trabajado en la parte en que el rendimiento ha sido afectado. Se ha aplicado el modelo de regresión lineal para modelar la relación entre las variables. Se ha utilizado estadígrafos tales como el error residual, coeficiente de variación y la desviación estándar para obtener un modelo matemático de regresión lineal. Por lo tanto, este procedimiento ha permitido establecer una ley que podría derivar una expresión analítica que caracterice al *overhead*, el cual predice el comportamiento de la carga u *overhead* en un EVR. También vale la pena mencionar que las expresiones analíticas encontradas pueden aplicarse para predecir la penalización calculada en la evaluación de un servicio en un EVR de Xen. Este modelo fue aceptado para publicación en [Fuertes10b]

Sin embargo, mediante este método no ha sido posible establecer que variables influyen determinantemente en la degradación del rendimiento (*overhead*) ni en qué medida. En consecuencia, en el siguiente capítulo se tratará de verificar en qué medida el incremento de variables incrementan o mejoran el rendimiento u *overhead* en un EVR, utilizando otras técnicas estadísticas como el ANOVA (Analisys of Variance).

Capítulo 5 Propuesta para determinar los factores que afectan el rendimiento en un escenario virtual.

5.1 Introducción

Como una contribución adicional en la Tesis doctoral, se ha considerado imprescindible determinar los factores que incrementan la penalización u *overhead* producida por la capa de virtualización en un Entorno Virtual de Red (EVR). En este capítulo, se tratará de responder a los siguientes cuestionamientos:

- ¿Cuáles son las variables que afectan (influyen) el rendimiento en un EVR?
- ¿En qué medida el incremento de variables aumentan el *overhead* o mejoran el rendimiento o en un EVR?
- ¿Al aumentar el número de CPUs en el interior de las MVs mejora o empeora el rendimiento?
- ¿Al limitar el ancho de banda, qué sucede con el rendimiento del EVR?
- ¿Cómo empeora o mejora el ancho de banda por causa del *overhead*?

Por lo tanto, este nuevo aporte se diferencia del anterior capítulo, puesto que ha sido enfocado en probar cuánto influyen cada una de las variables o elementos que intervienen en el despliegue de un EVR. Para este propósito se ha implementado una topología de prueba mediante Xen, a la cual se le va incrementando diversas variables. En estos experimentos ha sido seleccionada la plataforma Xen puesto que en mediciones anteriores, el conusmo de CPU y memoria de un EVR con Xen es muy cercano al de un equipo real (véase capítulo 2, apartado 2.3).

Para resolver este planteamiento fue desarrollado un programa en MatLab, para automatizar el método analítico que permita calcular los coeficientes de Regresión Lineal Múltiple y el test ANOVA (Analysis of Variance). Sus resultados fueron posteriormente comprobados con los resultados obtenidos al procesar los datos con el software estadístico SPSS.

ANOVA fue aplicado sobre la medida del rendimiento de un EVR, con el fin de determinar cómo influyen y en qué medida variables como el CPU, memoria, retardo, paquetes perdidos, throughput y tiempo de duración de la transferencia.

5.2 Configuración experimental

Tal como muestra la Fig. 5-1, se ha diseñado y puesto en funcionamiento un EVR en el cual el usuario desde la MV_1 realiza peticiones de transferencia de cargas a un Virtual Web Server, con el fin de evaluar el performance de la red así como la penalización del rendimiento.

Este experimento consistió en ir incrementando progresivamente el número de MVs, el número de CPUs en cada MV, y el número de bridges, hasta alcanzar tres bridges que generan tres segmentos de conexión con nueve MVs.

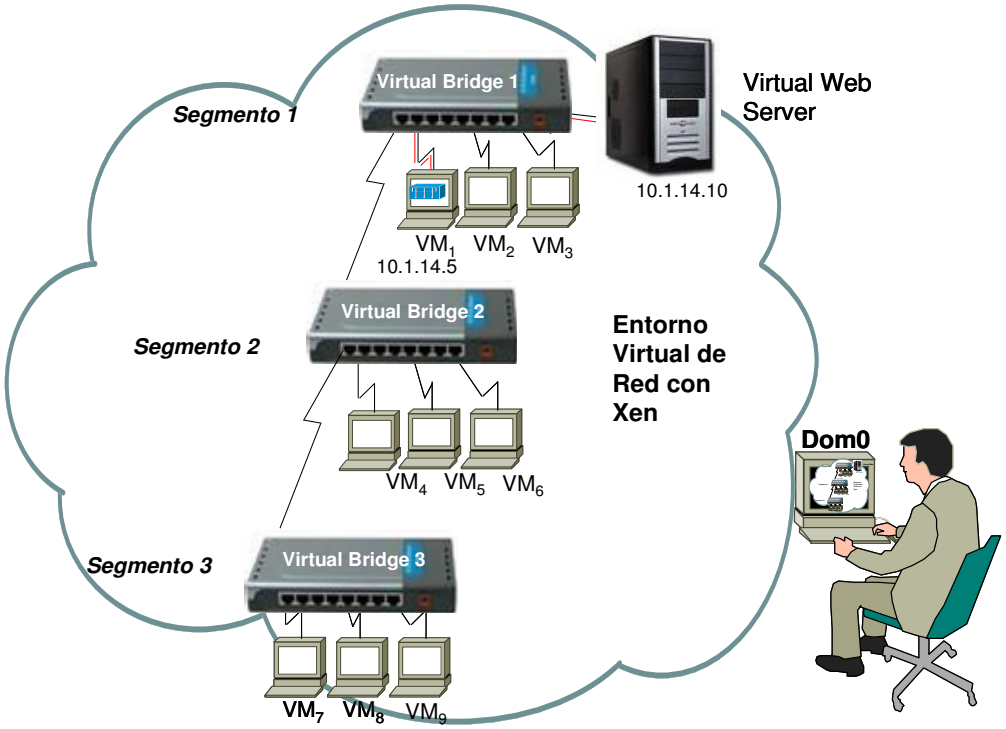

Figura 5-1 Topología de Prueba

El objetivo principal fue medir cuánto influye en la degradación del rendimiento del equipo anfitrión las variables que intervienen en el EVR, tales como el número de CPUs, memoria, retardo, número de paquetes perdidos, throughput y el tiempo de duración de la transferencia, a medida en que se van incrementando VMs y el numero de CPUs en cada una de las mismas. La configuración del experimento realizado se describe a continuación:

- En todo el experimento solamente la MV_1 realizó peticiones contra el Virtual Web Server a una velocidad constate (rate) de 10 peticiones/s, a un carga de transferencia constante de 10.000KB y emulando un número de 10 conexiones concurrentes hacia el Virtual Web Server.

- A cada segmento se iba incrementando de una en una hasta tres MVs, conectándolas al respectivo virtual bridge Ethernet, para obtener las medidas de dichos parámetros. Las MVs que se iban añadiendo, no recargaban al Virtual Web Server. Así mismo, las MVs de cada segmento enviaban y recibían paquetes entre ellas a través del comando ping, sin producir tráfico interferente.
- Cada MV de cada segmento fue evaluada con 1, 2, 3 y 4 CPUs, con fines de provocar variabilidad.
- Cada experimento fue repetido 11 veces para determinar la media a fin de que no se produzcan sesgos en las mediciones.

Se efectuaron dos experimentos utilizando el mismo EVR: En el **primero** los datos fueron medidos a lo natural sin ningún tipo de ajustes. En el **segundo** se ajustó el ancho de banda limitándolo a 10 Mbps, tanto en la MV1 como el Virtual Web Server con el fin de disponer de una primera observación de cómo ha sido afectado el throughput.

Todos los experimentos se realizaron en un solo host anfitrión que tiene un procesador Intel Core ™ 2 Quad CPU de 2,4 GHz., 3,23 GB de RAM, un interfaz PCI 100/1000, disco IDE de 100 GB a 7200 RPM. El sistema operativo instalado tanto en el equipo anfitrión como en cada MV fue Ubuntu Server 8.10, Kernel 2.6.27.5, con una partición ext3.

El software utilizado para nuestro propósito fue el siguiente:

- ***Httperf*** que es una herramienta para medir el rendimiento del servidor Web. Proporciona un mecanismo flexible para la generación de diferentes cargas de trabajo de http desde los clientes hacia el servidor Web.
- ***SAR*** (*System Activity Report*), que es un software para recopilar y mostrar información sobre el rendimiento del sistema ejecutado desde el Dom0, el cual producirá informes sobre el uso (consumo) del CPU, memoria, paquetes perdidos, retardo (rtt) y throughput.
- ***HTB*** (*Hierarchical token bucket*), que es un algoritmo incluido en los programas TC de Linux, que sirve para controlar el uso del ancho de banda de salida en un enlace determinado. HTB permite utilizar un enlace físico para simular múltiples enlaces y enviar distintos tipos de tráfico en diferentes enlaces simulados

5.3 Evaluación de Resultados sin limitar el Ancho de Banda

Luego de diseñar e implementar la topología de prueba de la Fig. 5-1, e ir aumentando progresivamente las variables, se realizaron 54 experimentos (cada uno validado 11 veces) tanto en el EVR sin ajustes así como en el EVR limitando su ancho de banda a 10Mbps. Los resultados fueron los siguientes:

La Fig. 5-2 muestra cómo mientras se incrementa el número de CPUs en las MVs se incrementa

el consumo del CPU.

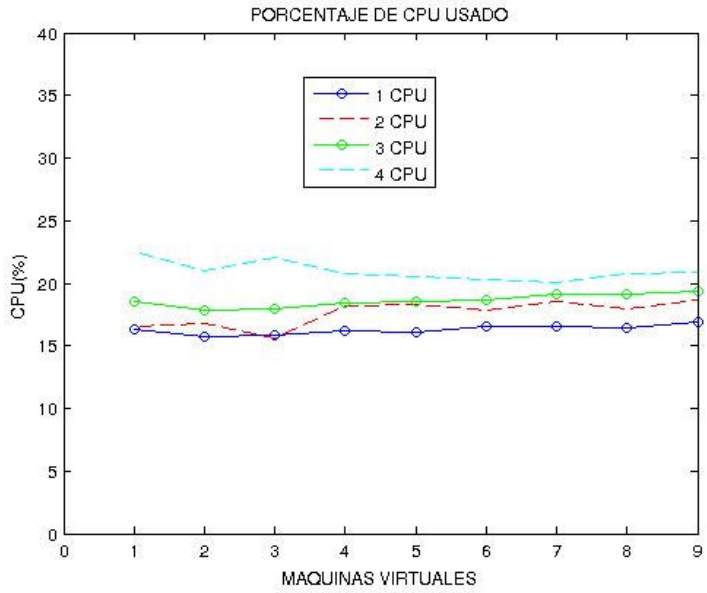

Figura 5-2 Consumo de CPU del Dom0 durante la transferencia

La Fig. 5-3 muestra cómo mientras se incrementa el número de CPUs en las MVs se incrementa el consumo de la memoria.

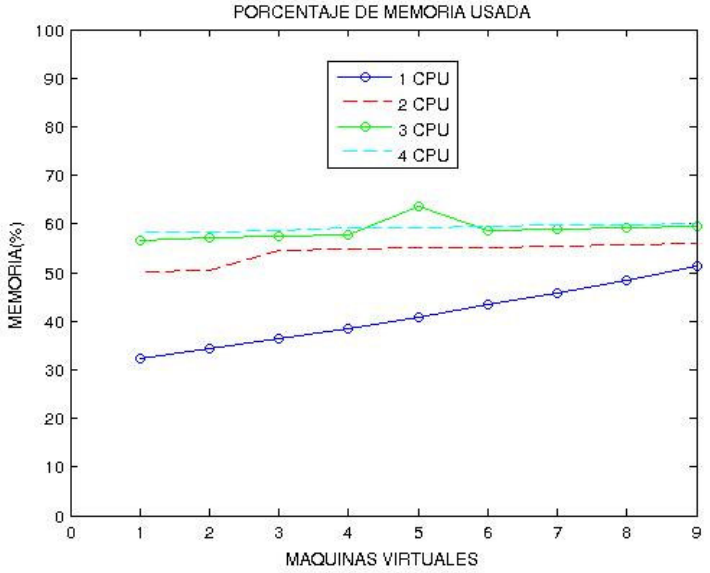

Figura 5-3 Consumo de memoria del Dom0 durante la transferencia

La Fig. 5-4 muestra que el ancho de banda se mantiene constante, a pesar de ir incrementando el número de CPUs en cada MVs.

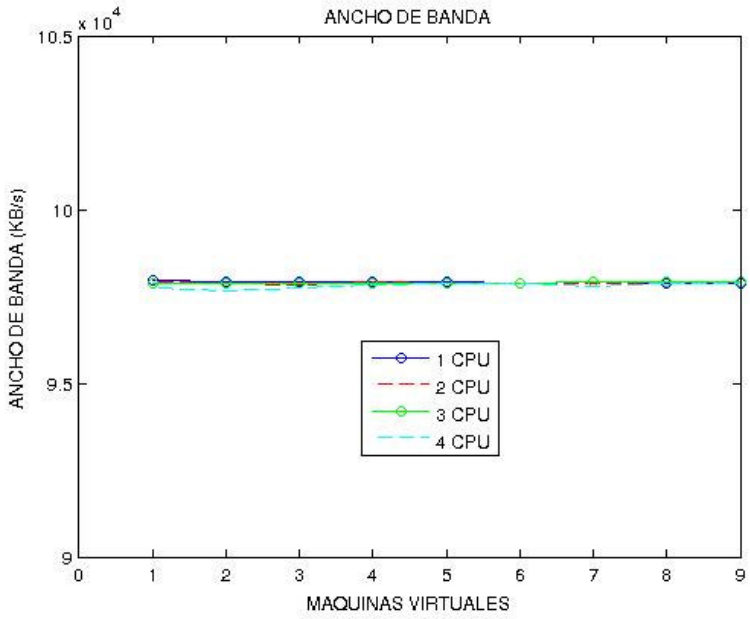

Figura 5-4 Throughput reportado durante la transferencia

La Fig. 5-5 muestra cómo se incrementa el número de paquetes perdidos a medida que se aumenta el número de CPUs en cada MVs.

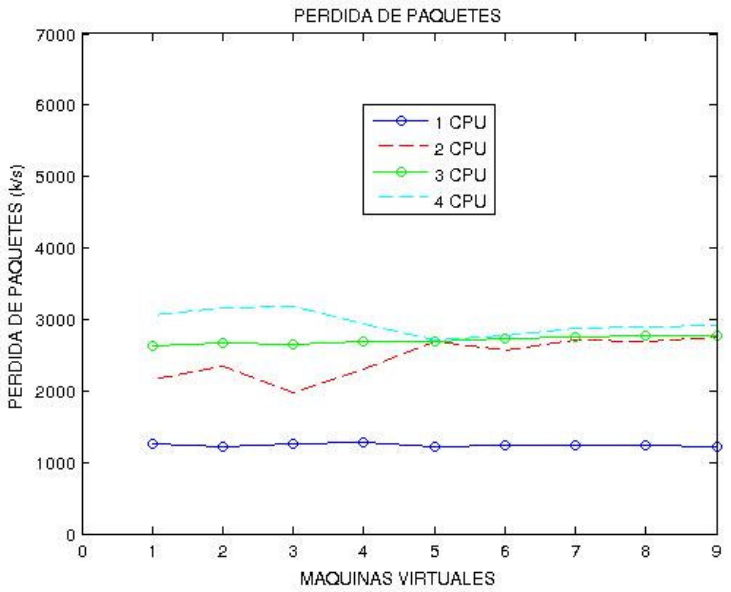

Figura 5-5 Pérdida de paquetes reportados durante la transferencia

La Fig. 5-6 muestra cómo permanece constante el retardo a pesar de que se incrementa el número de CPUs en cada MVs.

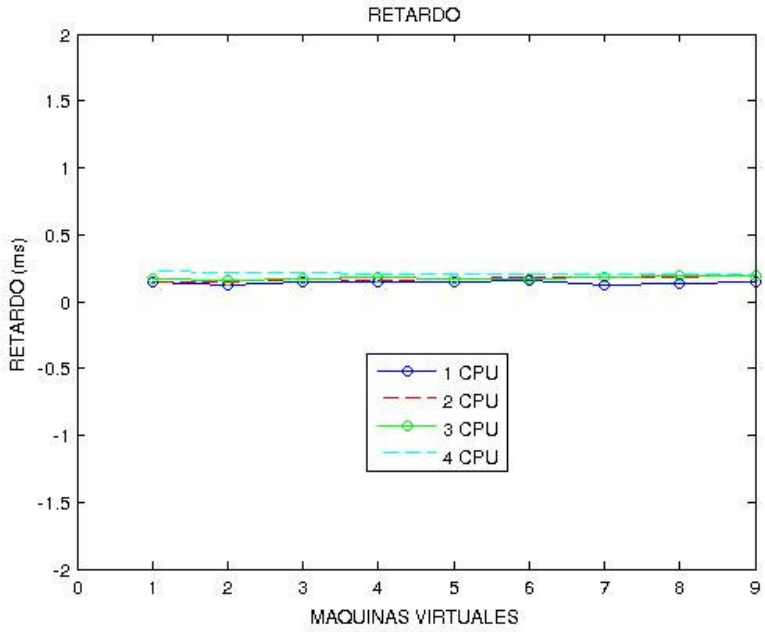

Figura 5-6 Retardo reportado durante la transferencia

La Fig. 5-7 muestra cómo al incluirse un segundo CPU, existe una mínima variación en el tiempo de duración de la transferencia a pesar de que se incrementa el número de CPUs en cada MVs. Así mismo se puede apreciar que el resto permanece constante.

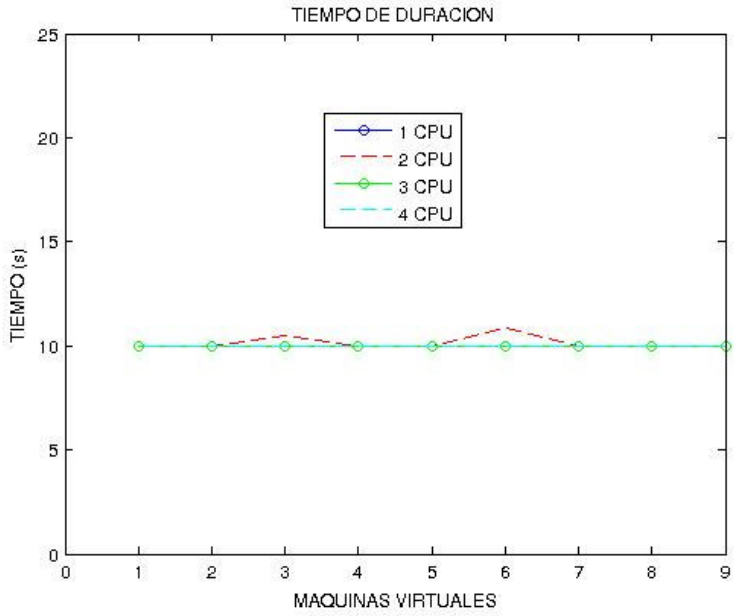

Figura 5-7 Tiempo de duración de la transferencia

5.4 Comparación de resultados entre el EVR sin ajustes y el EVR Limitando el Ancho de Banda

Con el fin de medir como sería afectado el throughput en el EVR, se ha ajustado el ancho de banda utilizando HTB. HTB es muy útil para limitar el ancho de banda de descarga y de subida de un cliente. Este algoritmo ha sido utilizado puesto que su aplicación ha sido probada en otros experimentos de mayor complejidad realizados.

Para limitar el ancho de banda mediante HTB, tanto en el cliente como el virtual Web Server se ejecutaron las siguientes instrucciones en la línea de comandos de cada MV, obteniéndose el resultado deseado como se puede apreciar en la Fig. 5-8:

tc qdisc add dev eth0 root handle 100: htb

tc class add dev eth0 parent 100: classid 100:1 htb rate 1250kbps

tc -s -d class show dev eth0

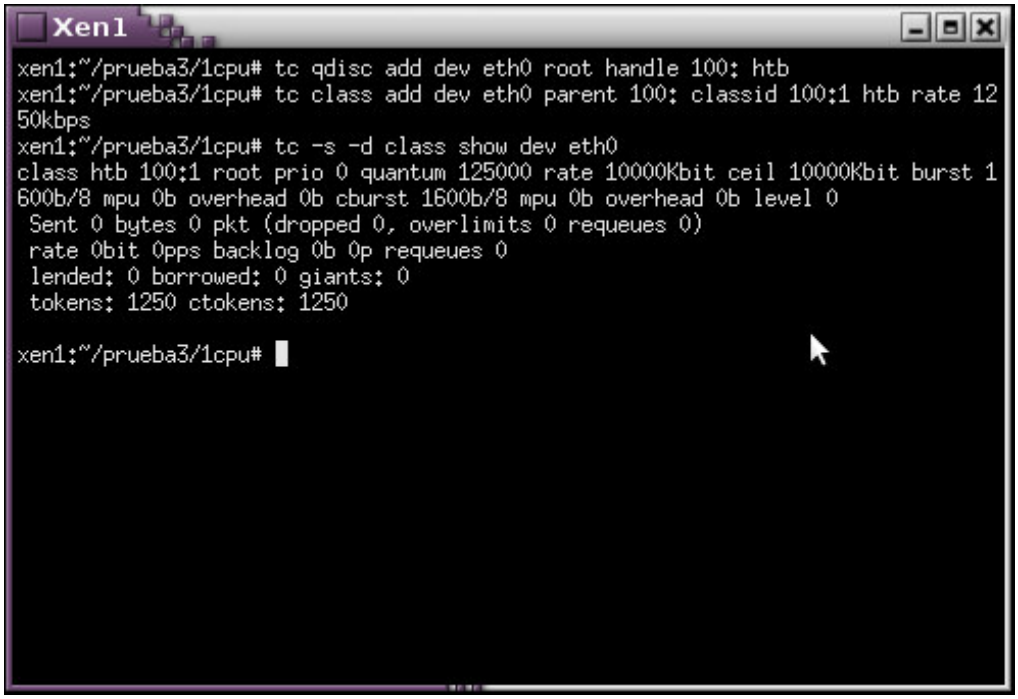

Figura 5-8 Limitación del ancho de banda a 10Mbps mediante HTB

Una vez que se ha realizado la evaluación de los resultados ajustando el ancho de banda, con un procedimiento similar al del apartado 5-3, se ha considerado útil graficar y comparar las diferencias entre los dos experimentos mencionados. De esta manera, desde las Figs. 5-9 a la 5-14 se muestra el comportamiento de los diferentes factores, notándose visualmente que existe un leve incremento

en la minoría de los mismos, manteniéndose constante el "throughput". En consecuencia, el ajuste del ancho de banda en el EVR no ha afectado determinantemente.

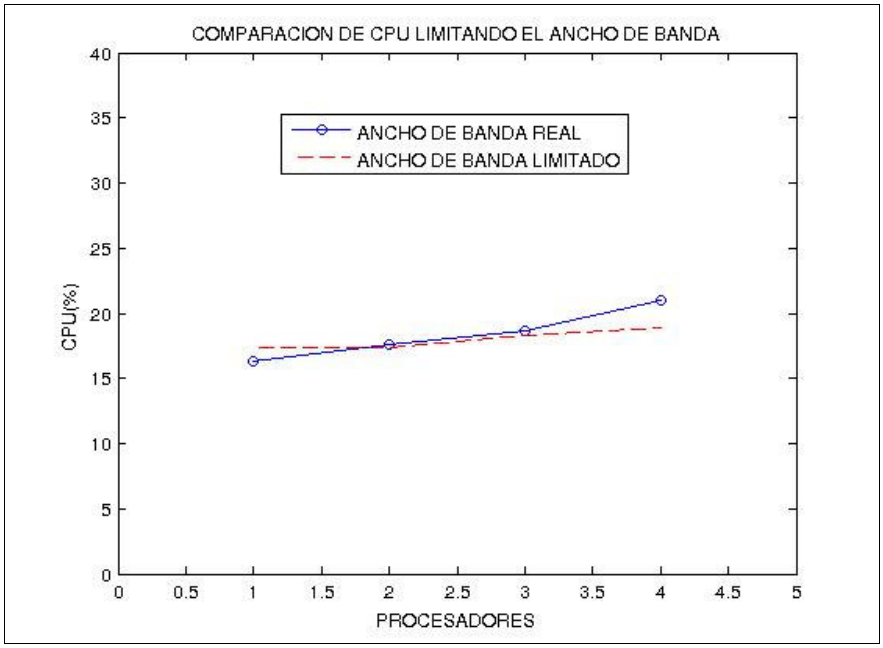

Figura 5-9 Comparación de los resultados del consumo de CPU limitando el AB a 10 Mbps versus el EVR sin ajustes

Como se pude apreciar el consumo de CPU ha sido afectado levemente, con seguridad por la aplicación de HTB en el experimento.

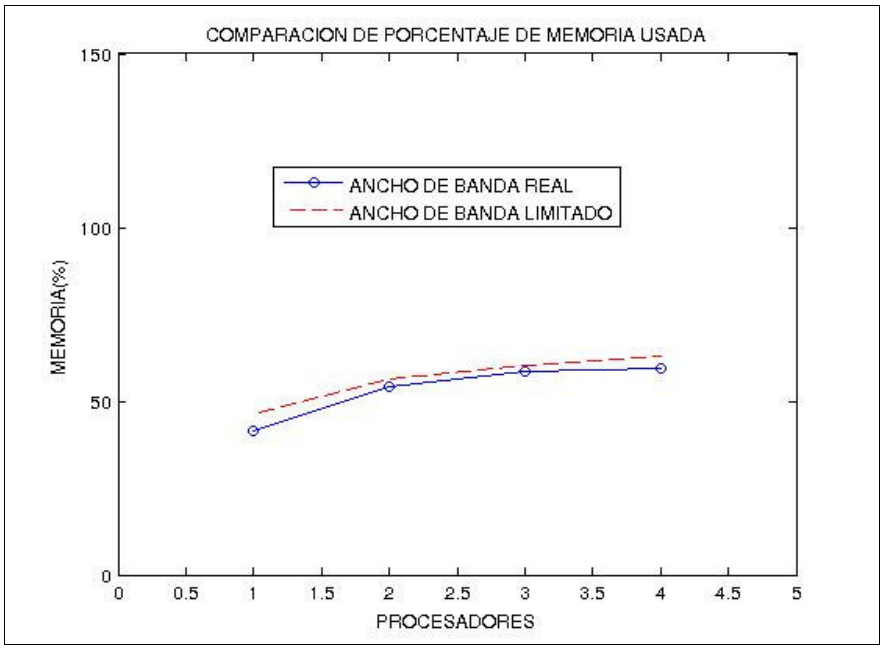

Figura 5-10 Comparación de los resultados del consumo de Memoria limitando el AB a 10 Mbps versus el EVR sin ajustes

Como se pude apreciar la memoria ha sido afectada levemente, con seguridad por la aplicación de HTB en el experimento.

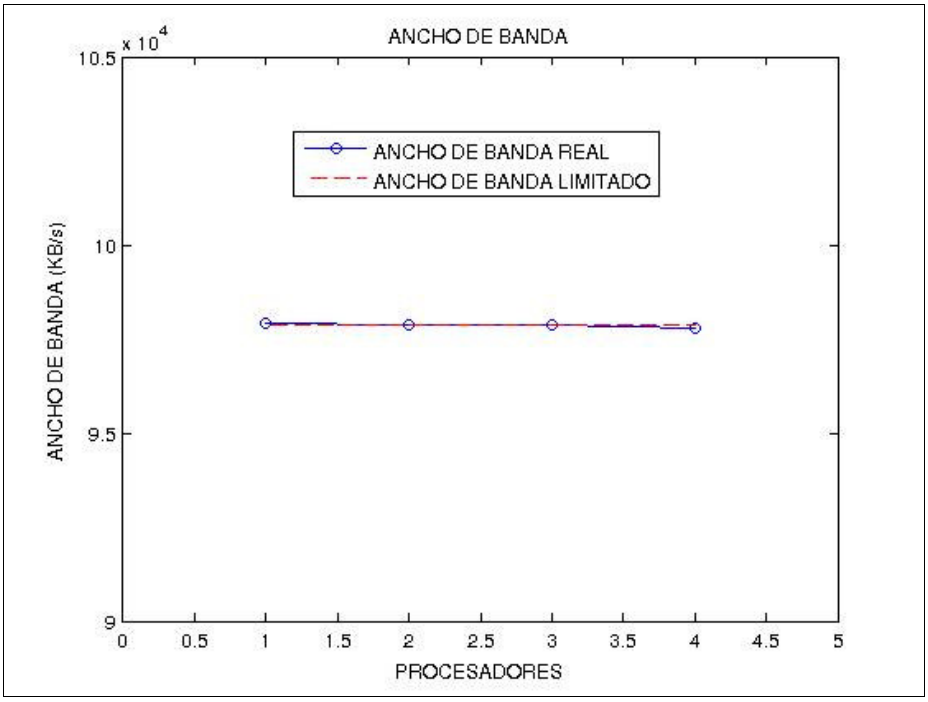

Figura 5-11 "Comparación del throughput limitando el AB a 10 Mbps versus el EVR sin ajustes"

Como se puede apreciar el throughput no ha sido afectado, a pesar de los ajustes y de la aplicación de HTB en el experimento.

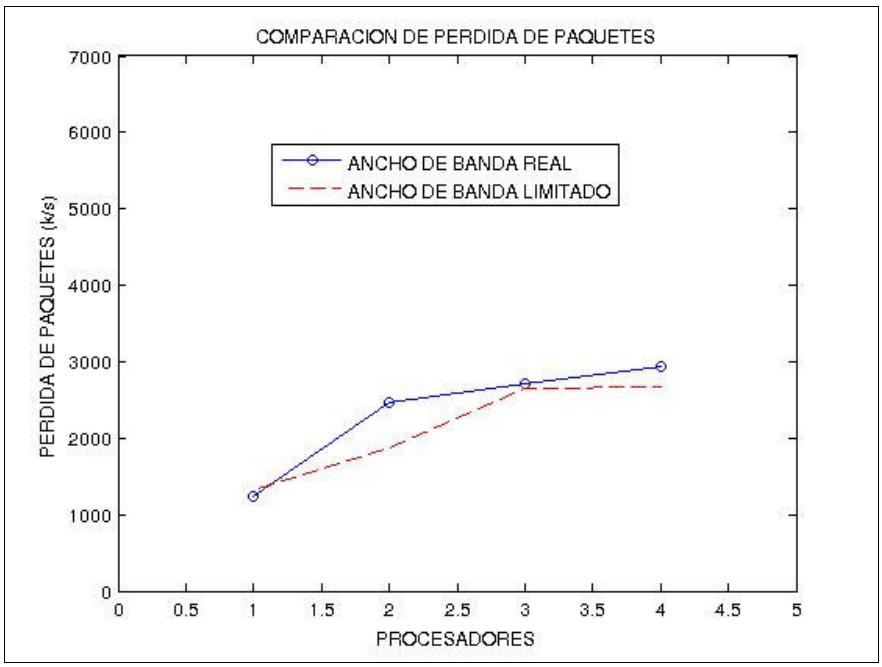

Figura 5-12 "Comparación de los resultados de la Perdida de Paquetes limitando el AB a 10 Mbps versus el EVR sin ajustes"

Como se puede apreciar la perdida de paquetes ha sido afectada levemente, presumiblemente por la aplicación de HTB en el experimento.

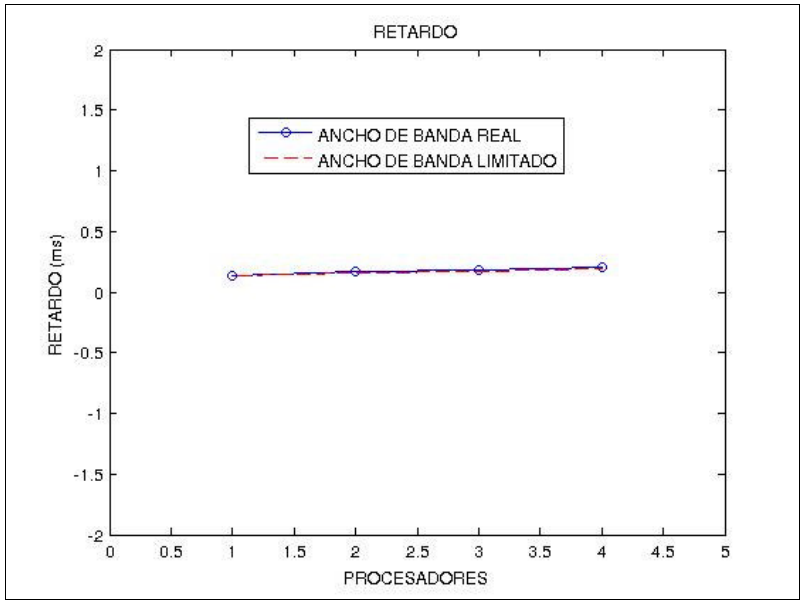

Figura 5-13 "Gráfico de dispersión que compara los resultados del Retardo limitando el AB a 10 Mbps versus el AB real"

Como se puede apreciar el retardo no ha sido afectado, a pesar de los ajustes y de la aplicación de HTB en el experimento.

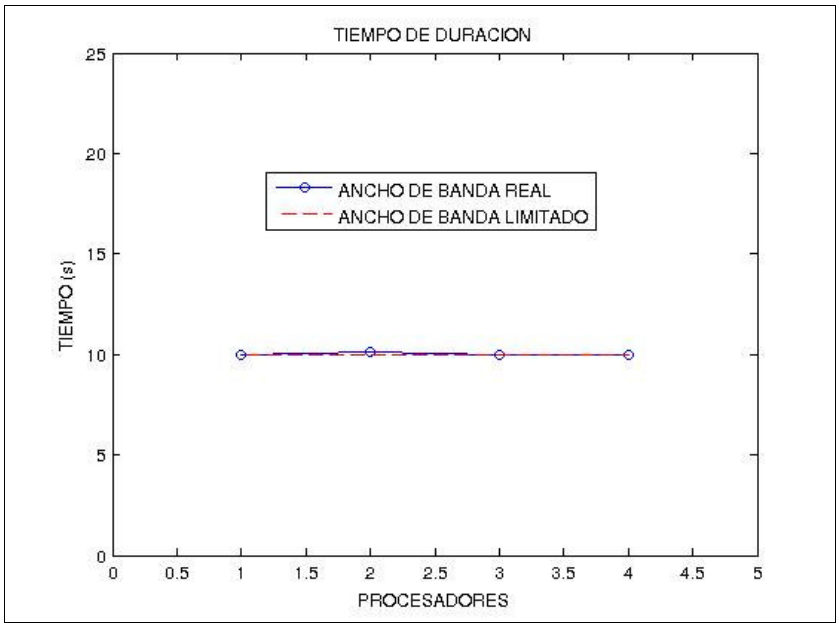

Figura 5-14 Comparación los resultados del Tiempo de Duración de la transferencia, limitando el AB a 10 Mbps versus el EVR sin ajustes

Como se puede apreciar el tiempo de duración de la transferencia no ha sido afectado, a pesar de los ajustes y de la aplicación de HTB en el experimento.

Puesto que al comparar los resultados obtenidos entre un EVR sin ajustes, versus los resultados obtenidos al limitar el ancho de banda, se observa que el 50% de los factores no han sido afectados, pero sobre todo el "throughput", en consecuencia, se ha decidido aplicar el análisis de varianza al conjunto de datos obtenidos al EVR sin ajustes.

En el siguiente apartado se detalla el procedimiento de aplicación del ANOVA.

5.5 Modelado Estadístico y Verificación

5.5.1 ANOVA

El análisis de la Varianza (ANOVA), es una técnica estadística que se aplica para comprobar si son iguales las medias de dos o más poblaciones independientes mediante la comprobación de varianzas insesgadas de muestras de diversas fuentes, para ello se utiliza la prueba de distribución F "Fisher" [Córdova02] [Córdova06]. El ANOVA trabaja muestras pequeñas, por esta razón se planifican adecuadamente la recolección de datos, en otras palabras los experimentos deben ser diseñados a medida. Esta técnica de análisis de la varianza ha sido aplicada en casi todas las disciplinas científicas, siendo en la actualidad un tema muy amplio de estudio.

Para el cálculo del ANOVA, se debe identificar las variables dependientes e independientes que intervienen en el experimento.

Entonces, sea $y = f(x_i)$ (1)

En donde:

- ***y*** es la variable dependiente que representa el rendimiento del equipo anfitrión calculada por el despliegue de un EVR y por la carga en la transferencia durante el experimento.
- x_i, representa a cada variable independiente (llamada también predictora, factor o respuesta). Las variables independientes identificadas en el experimento son, consumo de CPUs, consumo de la memoria, retardo, paquetes perdidos, throughput y por último al tiempo de duración de la transferencia de datos. Cada x_i ha sido sometida a diversos *k* tratamientos o niveles.

En base al primer planteamiento (ecuación (1)), el Modelo de ANOVA para cada una de las variables de respuesta como un único factor, con varios niveles de tratamiento está dado por:

$$x_{ij} = \mu_i + \varepsilon_{ij} \quad (2)$$

En donde:

- x_{ij} Representa cada observación $x_{ij}(i = 1,2,...,k, \quad j = 1,2,...,n_i)$ de la muestra
- ε_{ij} Mide la desviación o error del dato observado x_{ij} con respecto a la media μ_i. Esta desviación se denomina también "error o residuo". Dado que las variables aleatorias x_{ij}

son independientes y tienen cada una distribución normal $N(\mu_i, \sigma^2)$, las ε_{ij} son, entonces, variables aleatorias independientes y tienen distribución normal $N(0, \sigma^2)$.

- Por otro lado, cada media μ_i se desvía de la media total μ una cantidad $\alpha_i = \mu_i - \mu$. Este desvío es denominado, "efecto del i-ésimo tratamiento".

  Observe que:

$$\sum_{i \to 1}^{k} \alpha_i = 0 \qquad (3)$$

En resumen, el modelo de clasificación simple o de un factor completamente aleatorizado, es la ecuación:

$$x_{ij} = \mu_i + \varepsilon_{ij} = \mu + \alpha_i + \varepsilon_{ij} \qquad (4)$$

En donde,

- $i = 1, 2, ..., k, \quad j = 1, 2, ...n_i, \quad \sum n_i = n$.
- Las variables aleatorias x_{ij} son independientes y normales $N(\mu_i, \sigma^2)$.
- Las variables aleatorias ε_{ij} son independientes y normales $N(0, \sigma^2)$.
- μ es la media total, y $\alpha_i = \mu_i - \mu$ es el efecto del tratamiento i .

Mediante el ANOVA, se trata de probar si el efecto de un factor o tratamiento (variable independiente, por ejemplo throughput) en la respuesta de un proceso o sistema (el overhead) es "significativo", al realizar experimentos variando los niveles de ese factor (CPU_1, CPU_2, CPU_3 y CPU4) en cada VM del EVR, a fin de conseguir variabilidad en los tratamientos para obtener el valor de significación esperado.

Cada diseño experimental o método de análisis de varianza se puede representar mediante un *Modelo de Regresión Lineal Múltiple*, para analizar el comportamiento general de cada una de las variables independientes. En este punto se puede diferenciar del Modelo de Regresión Lineal Simple, que fue utilizado en el capítulo 4, apartado 4.6, en la que se relaciona la variable dependiente identificada como carga de trabajo del CPU con una sola variable independiente que representaba el ancho de banda.

Además, El ANOVA se puede clasificar según el número de variables independientes (o vías), si es de un factor se denomina de clasificación simple (o de una vía) como es el caso de nuestro estudio en el que ha sido aplicado el ANOVA en forma individual para cada variable independiente.

En el procedimiento de una vía el modelo es completamente aleatorio, generando además un proceso de aleatoriedad por tratamientos. Así por ejemplo al ir incorporando experimentalmente

MVs y el número de CPUs en el interior de cada VM, por esta razón adicional ha sido factible considerar esta técnica en nuestra investigación.

5.5.2 Estimación de los factores o parámetros

Para la estimación de los parámetros se aplica el método de los mínimos cuadrados, que consiste en minimizar el cuadrado de la suma de los errores, los mismos que son causados por las estimaciones con respecto a las observaciones.

Si suponemos que se dispone de $n > k$ observaciones, además se denota como x_{ij} al valor de la i-ésima observación de la variable x_j, como se describe a continuación en la Tabla 5-1:

Tabla 5-1 Niveles de Factor

	x_1	x_2		x_k
y_1	x_{11}	x_{12}		x_{1k}
y_2	x_{21}	x_{22}		x_{2k}
⋮	⋮	⋮	⋮	⋮
y_n	x_{n1}	x_{n2}		x_{nk}

Para la investigación durante la experimentación se obtuvo los siguientes datos para cada variable independiente (Ver tablas desde la 5-2 hasta la 5-7):

Tabla 5-2 Datos procesados del consumo de CPU

MV	CPU_1	CPU_2	CPU_3	CPU_4
1	16,36	16,59	18,49	22,47
2	15,76	16,83	17,83	21,05
3	15,79	15,60	18,01	22,03
4	16,23	18,20	18,44	20,75
5	16,14	18,25	18,52	20,54
6	16,54	17,84	18,69	20,30
7	16,57	18,50	19,14	20,04
8	16,39	17,99	19,08	20,71
9	16,91	18,60	19,41	20,90

Tabla 5-3 Datos procesados del consumo de la memoria

MV	Memory 1	Memory 2	Memory 3	Memory 4
1	32,30	50,14	56,50	58,23
2	34,22	50,33	57,11	58,47
3	36,32	54,52	57,42	58,75
4	38,44	54,74	57,84	59,08
5	40,91	55,04	63,51	59,33
6	43,42	55,26	58,52	59,48
7	45,88	55,48	58,82	59,65
8	48,48	55,78	59,17	59,89
9	51,29	56,05	59,48	60,14

Tabla 5-4 Datos procesados del retardo

MV	rtt 1	rtt 2	rtt 3	rtt 4
1	0,15	0,15	0,17	0,23
2	0,12	0,15	0,16	0,22
3	0,15	0,17	0,17	0,22
4	0,15	0,16	0,18	0,21
5	0,15	0,17	0,17	0,20
6	0,16	0,18	0,17	0,20
7	0,12	0,18	0,18	0,20
8	0,13	0,18	0,19	0,20
9	0,15	0,20	0,19	0,20

Tabla 5-5 Datos procesados de paquetes perdidos

MV	Perd 1	Perd 2	Perd 3	Perd 4
1	1,257.49	2,168.90	2,630.91	3,059.51
2	1,218.15	2,340.99	2,668.36	3,167.12
3	1,266.08	1,966.97	2,659.29	3,190.55
4	1,280.61	2,310.73	2,694.18	2,938.08
5	1,227.92	2,689.76	2,700.28	2,721.00
6	1,230.24	2,565.62	2,729.14	2,774.75
7	1,243.95	2,713.68	2,749.77	2,869.06
8	1,241.65	2,685.87	2,768.54	2,896.58
9	1,213.02	2,745.17	2,780.73	2,907.70

Tabla 5-6 Datos procesados del throughput

MV	throug 1	throug 2	throug 3	throug 4
1	97939,07	97900,54	97866,84	97725,72
2	97912,47	97892,74	97886,29	97672,82
3	97896,26	97829,88	97872,46	97762,83
4	97897,76	97903,56	97889,26	97834,6
5	97912,17	97890,76	97893,62	97880,28
6	97889,95	97892,45	97868,57	97873,99
7	97916,88	97852,87	97898,22	97788,34
8	97894,05	97871,42	97897,11	97861,34
9	97885,44	97891,46	7899,29	97857,81

Tabla 5-7 Datos procesados del tiempo de duración de la transferencia

MV	time 1	time 2	time 3	time 4
1	9,97	9,98	9,98	9,99
2	9,97	9,98	9,98	10
3	9,98	10,46	9,98	9,99
4	9,98	9,98	9,98	9,98
5	9,97	9,98	9,98	9,98
6	9,98	10,88	9,98	9,98
7	9,97	9,98	9,98	9,99
8	9,98	9,98	9,98	9,98
9	9,98	9,98	9,98	9,98

5.5.3 Procedimiento de cálculo de los coeficientes de Regresión y ANOVA

Con la misma consideración con que se trabajó en el modelo de *Regresión Lineal Simple* del capitulo 4, el modelo de *Regresión lineal múltiple* es factible ponerlo en la forma matricial, porque resulta más versátil su aplicación, además debido a la cantidad de tratamientos que están involucradas, es más sencillo para manipularlo algebraicamente.

A continuación se explica el procedimiento para el cálculo de los coeficientes de *Regresión Lineal Múltiple,* tomando como ejemplo el consumo del CPU (véase la tabla 5-2). Es importante señalar, que este procedimiento se ha validado programado un método analítico utilizando *MatLab* (véase Apéndice B-3) el mismo que ha permitido calcular los coeficientes de Regresión Lineal Múltiple y la aplicación del test ANOVA. Por eficiencia y simplicidad para el resto de variables se empleó el software estadístico *SPSS*, que también fue utilizado para calcular los coeficientes de regresión y el test ANOVA. Esto ha permitido comprobar los resultados, que se obtuvieron al emplear el método analítico con MatLab (véase Tablas desde 5-8 a 5-12).

Entonces sea la matriz Y que representa las MVs (n=9 que corresponde al número de MVs), y sea la Matriz X (la primera columna siempre es 1) que corresponde a la variable independiente CPU, con k=4 tratamientos. El procedimiento de cálculo fue el siguiente:

Tabla 5-8 Formulación Matricial para validar el procedimiento de calculo ANOVA

Matriz y		Matriz X				
1		1	16.36	16.59	18.49	22.47
2		1	15.76	16.83	17.83	21.05
3		1	15.79	15.6	18.01	22.03
4		1	16.23	18.2	18.44	20.75
5		1	16.14	18.25	18.52	20.54
6		1	16.54	17.84	18.69	20.3
7		1	16.57	18.5	19.14	20.04
8		1	16.39	17.99	19.08	20.71
9		1	16.91	18.6	19.41	20.9

i.) Calcular la transpuesta de la matriz X, esta es X'

Tabla 5-9 Matriz transpuesta

1	1	1	1	1	1	1	1	1
16.36	15.76	15.79	16.23	16.14	16.54	16.57	16.39	16.91
16.59	16.83	15.60	18.20	18.25	17.84	18.5	17.99	18.60
18.49	17.83	18.01	18.44	18.52	18.69	19.14	19.08	19.41
22.47	21.05	22.03	20.75	20.54	20.30	20.04	20.71	20.90

ii.) Calcular la matriz X'.X, esta es:

Tabla 5-10 Matriz X´.X Factor: 1.0e+003 *

0.0090	0.1467	0.1584	0.1676	0.1888
0.1467	2.3920	2.5839	2.7333	3.0762
0.1584	2.5839	2.7963	2.9532	3.3174
0.1676	2.7333	2.9532	3.1236	3.5144
0.1888	3.0762	3.3174	3.5144	3.9652

iii.) Calcular la matriz inversa de X'.X

Tabla 5-11 Matriz $(X'.X)^{-1}$

664.9590	-10.8232	-11.2634	3.4965	-16.9389
-10.8232	7.9747	-0.7406	-4.7703	-0.8239
-11.2634	-0.7406	0.8471	-0.3026	0.6703
3.4965	-4.7703	-0.3026	4.1562	0.1038
-16.9389	-0.8239	0.6703	0.1038	0.7931

iv.) Calcular el producto de la matriz transpuesta de X por la matriz Y. Esta permitirá determinar el valor de $b = (X'.X)^{-1}.(X'.Y)$. En concreto esta matriz representa los valores de los coeficientes del modelo de Regresión Lineal Múltiple (véase Tabla 5-12):

Tabla 5-12 Matriz b

-17.0851
-2.7845
-0.9170
6.5313
-1.8128

Entonces el modelo de regresión es:

$$\bar{y} = -17.0851 - 2.7845x_1 - 0.9170x_2 - 6.5313x_3 - 1.8128x_4$$

v.) Calcular la suma de los cuadrados de los errores, SCE representados por los modelos:

$$\mathbf{SCE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \varepsilon'\varepsilon$$

O también este modelo, el cual vamos a utilizar en nuestro estudio:

$$\mathbf{SCE} = Y'.Y - b'X'X$$

Por lo tanto SCE = 5.9639

$$SCE = 5.9639$$

vi.) Calcular la suma de los cuadrados de regresión, SCR:

$$SCR = b'.X'.X - \frac{(\sum_{i=1}^{n} y_i)^2}{n}; SCR = 54.0361$$

vii.) Calcular la suma de los cuadrados total, SCyy:

$$SC_{yy} = Y'.Y - \frac{(\sum_{i=1}^{n} y_i)^2}{n}; SC_{yy} = 60.0000$$

viii.) Calcular el error cuadrado medio, que es un estimador insesgado de la varianza σ^2, la misma que se calcula así:

$$MCE = s^2 = \frac{SCE}{n-k-1}, MCE = \frac{5.9639}{9-4-1}, MCE = 1.4910$$

ix.) Calcular el cuadrado medio de la siguiente manera:

$$MCR = \frac{SCR}{k}, MCR = \frac{54.0361}{4}, MCR = 13.5090$$

.

x.) Determinar los grados de libertad de la investigación, así:

$$k = 4$$

$$n - k - 1 = 9 - 4 - 1 = 4$$

$$n - 1 = 8$$

xi.) Calcular la razón F calculada:

$$F_{abs} = \frac{MCR}{MCE} = \frac{13.5090}{1.4910} = 9.0605$$

xii.) Calcular el factor de significación (p-value)

La prueba de ANOVA provee un valor probable (p-value) el cual determina si la hipótesis nula H_o debe ser aceptada o no, de acuerdo con un nivel de significación pre-definido (típicamente $\alpha = 0.05$). Este valor es calculado en base a la razón F_{abs} calculada, combinándola con los grados de libertad tanto del numerador como del denominador.

En el siguiente apartado se describe un cuadro resumen del análisis de la Varianza (ANOVA) de los datos calculados en el apartado 5.5.3, y la forma de comprobación mediante el software estadístico SPSS (véase Tablas desde la 5-13 hasta 5-15):

5.5.4 Análisis para el consumo de CPU

Tabla 5-13 Tabla de Análisis de la Varianza

Fuente de Variación	Suma de Cuadrados	Grados de Libertad	Cuadrado medio	F
Regresión	SCR	k	MCR	$F_{abs} = \frac{MCR}{MCE}$
Error o residual	SCE	$n - k - 1$	$s^2 = MCE$	
Total	SC_{yy}	$n - 1$		

La Tabla 5-13 muestra los parámetros estimados con el test ANOVA según [Galindo99]. Al reemplazar en esta tabla los datos calculados de los hechos observados en base al modelo analítico descrito, se obtiene la Tabla 5-14.

Tabla 5-14 Tabla de Análisis de la Varianza

Fuente de Variación	Suma de Cuadrados	Grados de Libertad	Cuadrado medio	F
Regresión	54.0361	4	13.5090	9.0605
Error o residual	5.9639	4	1.4910	
Total	60.0000	8		

Tal como se ha señalado en párrafos anteriores, los resultados encontrados en la Tabla 5-14 obtenidos mediante este método analítico, pueden ser calculados directamente desde el software estadístico *SPSS*. Esto permite validar la exactitud del método analítico. SPSS además proporciona el valor de significación de la investigación (p-value, Sig) directamente:

Tabla 5-15 Tabla de Análisis de la Varianza mediante SPSS ANOVA(b)-CPU

Modelo		Suma de cuadrados	gl	Media cuadrática	F	Sig.
1	Regresión	54.036	4	13.509	9.060	.028(a)
	Residual	5.964	4	1.491		
	Total	60.000	8			

a Variables predictoras: (Constante), CPU 4, CPU 1, CPU 2, CPU 3
b Variable dependiente: MV

5.5.5 Comprobación de la Hipótesis

La **hipótesis nula** H_0 consiste en afirmar que las medias de las k poblaciones (o tratamientos) son iguales, (o las k muestras provienen de la misma población). En el caso del consumo del CPU, sería:

Hipótesis Nula; $H_0: \beta_0 = \beta_1 = \beta_2 = \cdots = \beta_k = 0$

Hipótesis Alternativa $: H_1: \beta_k \neq 0$

Estadístico de Prueba; $F_{abs} = 9.060$;

Región de Rechazo; El estadístico de prueba $F_{abs} = 9.060$; $F_{abs} = 9.060$ debe ser comparado con el valor obtenido en la tabla de distribución de probabilidad Fisher que es: $F_{0.05}(4,4) = 6.39$ resulta que:

$F_{abs} > F_{\alpha}(k, n-k-1)$; $p - value = 0.028$; 0.028>0.05

Básicamente si $p-value > \alpha$ $F_{obs} > F_{\alpha}(k, n-k-1)$ entonces H_o es aceptada (el factor es no significativo). Como 0.028<0.05, la H_o es rechazada. Por tanto, hay que rechazar la hipótesis nula, por lo que se concluye que las MVs están siendo influenciadas por los CPUs.

La misma comprobación fue realizada para el resto de variables independientes tales como: consumo de la memoria, el retardo, el incremento de paquetes perdidos, throughput y por último para el tiempo de duración de la transferencia de los datos, para ello todo el proceso lo calcularemos mediante SPSS.

5.5.6 Coeficiente de determinación múltiple

Además del test ANOVA, se debe analizar el *Coeficiente de Determinación Múltiple*, este informa sobre la relación existente entre las variables independientes y la dependiente y sirve además para verificar la dispersión de los datos. Se define el coeficiente de determinación múltiple R^2 como:

$$R^2 = \frac{SCR}{SC_{yy}} = 1 - \frac{SCE}{SC_{yy}} \tag{5}$$

Reemplazando en (5): $R^2 = \frac{54.0361}{60.0000} = 0.900602$

El rango de variación de R^2 es de $0 \leq R^2 \leq 1$, si el valor de R^2 es cercano a 1, determina que el modelo es bueno, en cambio si se acerca a 0, el ajuste es malo; en cambio si $R^2 = 0$ indica que falta por completo el ajuste del modelo a los datos, y si $R^2 = 1$ se obtiene un ajuste perfecto. Como en este caso $R^2 = 0{,}901$ esto indica que el ajuste es muy bueno. Sin embargo R^2 tiende a sobre estimar el valor de la correlación entre las variables involucradas, por tanto, es necesario emplear el coeficiente de determinación ajustado R_a^2 este está diseñado para compensar el sesgo optimista determinado por R^2, su modelo es:

$$R_a^2 = R^2 - \frac{k(1-R^2)}{n-k-1} \tag{6}$$

Reemplazando en (6):

$$R_a^2 = 0{,}900602 \frac{4(1-0{,}900602)}{9-4-1} = 0.801204$$

El cálculo de R^2 que corresponde a R cuadrado y a R_a^2 que corresponde a R cuadrado ajustada puede ser obtenida a través de SPSS (véase Tabla 5-16)

Tabla 5-16 Resumen del modelo - CPU

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación
1	.949(a)	.901	.801	1.22106

a Variables predictoras: (Constante), CPU 4, CPU 1, CPU 2, CPU 3

5.5.7 Tabla de Coeficientes de Regresión y de determinación múltiple

La tabla 5-17 resume los valores de los coeficientes del modelo de *Regresión Lineal Múltiple*, obtenidos mediante *SPSS*, mismos que validan el método analítico descrito en el apartado 5.5.3, numeral del i al iv).

Tabla 5-17 Coeficientes(a)-CPU

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	-17.085	31.487		-.543	.616
	CPU 1	-2.784	3.448	-.376	-.808	.465
	CPU 2	-.917	1.124	-.343	-.816	.460
	CPU 3	6.531	2.489	1.241	2.624	.059
	CPU 4	-1.813	1.087	-.523	-1.667	.171

a Variable dependiente: MV

A partir de este momento para optimizar nuestro análisis para el resto de variables independientes, se ha utilizado el software estadístico SPSS, que tal como ha sido demostrado, sigue el mismo proceso analítico descrito para la CPU en el apartado 5.5.3 (véase Tablas desde la 5-18 hasta la 5-32).

5.5.8 Análisis para el consumo de Memoria

Tabla 5-18 ANOVA(b)-Memoria

Modelo		Suma de cuadrados	gl	Media cuadrática	F	Sig.
1	Regresión	59.969	4	14.992	1966.211	.000(a)
	Residual	.031	4	.008		
	Total	60.000	8			

a Variables predictoras: (Constante), Memory 4, Memory 3, Memory 2, Memory 1
b Variable dependiente: VM

Tabla 5-19 Resumen del modelo - Memoria

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación	Estadísticos de cambio				
					Cambio en R cuadrado	Cambio en F	gl1	gl2	Sig. del cambio en F
1	1.000(a)	.999	.999	.08732	.999	1966.211	4	4	.000

a Variables predictoras: (Constante), Memory 4, Memory 3, Memory 2, Memory 1

Tabla 5-20 Coeficientes(a)-Memoria

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	-89.612	40.038		-2.238	.089
	Memory 1	.277	.063	.661	4.390	.012
	Memory 2	.023	.045	.019	.509	.638
	Memory 3	-.014	.024	-.011	-.598	.582
	Memory 4	1.398	.768	.330	1.821	.143

a Variable dependiente: VM

5.5.9 Análisis para el Retardo

Tabla 5-21 ANOVA(b) -Retardo

Modelo		Suma de cuadrados	Gl	Media cuadrática	F	Sig.
1	Regresión	59.520	4	14.880	124.008	.000(a)
	Residual	.480	4	.120		
	Total	60.000	8			

a Variables predictoras: (Constante), rtt 4, rtt 1, rtt 3, rtt 2
b Variable dependiente: VM

Tabla 5-22 Resumen del modelo - Retardo

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación	Estadísticos de cambio				
					Cambio en R cuadrado	Cambio en F	gl1	gl2	Sig. del cambio en F
1	.996(a)	.992	.984	.34640	.992	124.008	4	4	.000

a Variables predictoras: (Constante), rtt 4, rtt 1, rtt 3, rtt 2

Tabla 5-23 Coeficientes(a)-Retardo

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	1.890	5.681		.333	.756
	rtt 1	-22.108	8.675	-.120	-2.549	.063
	rtt 2	77.790	15.248	.459	5.102	.007
	rtt 3	68.679	17.673	.254	3.886	.018
	rtt 4	-91.500	17.470	-.390	-5.238	.006

a Variable dependiente: VM

5.5.10 Análisis para el incremento de paquetes perdidos

Tabla 5-24 ANOVA(b)-Paquetes perdidos

Modelo		Suma de cuadrados	gl	Media cuadrática	F	Sig.
1	Regresión	59.014	4	14.754	59.860	.001(a)
	Residual	.986	4	.246		
	Total	60.000	8			

a Variables predictoras: (Constante), perdida 4, perdida 1, perdida 3, perdida 2
b Variable dependiente: VM

Tabla 5-25 Resumen del modelo- Paquetes perdidos

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación	Estadísticos de cambio				
					Cambio en R cuadrado	Cambio en F	gl1	gl2	Sig. del cambio en F
1	.992(a)	.984	.967	.49646	.984	59.860	4	4	.001

a Variables predictoras: (Constante), perdida 4, perdida 1, perdida 3, perdida 2

Tabla 5-26 Coeficientes(a)- Paquetes perdidos

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	-131.384	21.762		-6.037	.004
	perdida 1	-.005	.013	-.045	-.437	.685
	perdida 2	-.003	.002	-.281	-1.145	.316
	perdida 3	.059	.008	1.103	7.786	.001
	perdida 4	-.003	.002	-.176	-1.306	.262

a Variable dependiente: VM

5.5.11 Análisis para throghput

Tabla 5-27 ANOVA(b)-Throughput

Modelo		Suma de cuadrados	gl	Media cuadrática	F	Sig.
1	Regresión	44.938	4	11.234	2.983	.157(a)
	Residual	15.062	4	3.766		
	Total	60.000	8			

a Variables predictoras: (Constante), throughput 4, throughput 2, throughput 3, throughput 1
b Variable dependiente: MV

Tabla 5-28 Resumen del modelo – throughput

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación	Estadísticos de cambio				
					Cambio en R cuadrado	Cambio en F	gl1	gl2	Sig. del cambio en F
1	.865(a)	.749	.498	1.94051	.749	2.983	4	4	.157

a Variables predictoras: (Constante), throughput 4, throughput 2, throughput 3, throughput 1

Tabla 5-29 Coeficientes(a) – throughput

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	1336.892	6052.980		.221	.836
	throughput 1	-.009	.058	-.054	-.151	.887
	throughput 2	-.028	.030	-.254	-.934	.403
	throughput 3	.000	.000	-.401	-1.389	.237
	throughput 4	.024	.012	.630	1.945	.124

a Variable dependiente: MV

5.5.12 Análisis para el Tiempo

Tabla 5-30 ANOVA(b)-Tiempo

Modelo		Suma de cuadrados	gl	Media cuadrática	F	Sig.
1	Regresión	40.078	4	10.019	2.012	.258(a)
	Residual	19.922	4	4.981		
	Total	60.000	8			

a Variables predictoras: (Constante), Time 4, Time 3, Time 2, Time 1

b Variable dependiente: VM

Tabla 5-31 Resumen del modelo-Tiempo

Modelo	R	R cuadrado	R cuadrado corregida	Error típ. de la estimación	Estadísticos de cambio				
					Cambio en R cuadrado	Cambio en F	gl1	gl2	Sig. del cambio en F
1	.817(a)	.668	.336	2.23172	.668	2.012	4	4	.258

a Variables predictoras: (Constante), Time 4, Time 3, Time 2, Time 1

Tabla 5-32 Coeficientes(a)-Tiempo

Modelo		Coeficientes no estandarizados		Coeficientes estandarizados	t	Sig.
		B	Error típ.	Beta		
1	(Constante)	-5261.845	4765.586		-1.104	.331
	Time 1	307.236	237.667	.591	1.293	.266
	Time 2	-.800	2.815	-.094	-.284	.790
	Time 3	334.425	196.607	.611	1.701	.164
	Time 4	-112.909	144.914	-.300	-.779	.479

a Variable dependiente: VM

Para un mejor análisis, previo la aceptación o no de H_o, la Tabla 5-33 muestra un resumen de los *Coeficientes de Determinación Múltiple* y de *F- Probabilidad Fisher* calculada en base a: $F_{0.05}(4,4)=6.39$.

Del mismo modo, la Tabla 5-34 muestra los coeficientes de *Regresión Lineal Múltiple*.

Finalmente, la Tabla 5-35 muestra el resumen final incluido el valor de significación (p-value) para aceptar o rechazar H_o en base a $F_{0.05}(4,4) = 6.39$

Tabla 5-33 Resumen de Coeficientes(a) de todas las variables independientes

Variables Factor	**Coeficiente R^2**	**Coeficiente Ajustado**	**Prob. F F_{obs}**	**Condición de la Hipótesis H_o**
CPUs	0.901	0.801	9.060	9.060 > 6.39 Rechaza
Memoria	0.999	0.999	1966.211	1966.211 > 6,39 Rechaza
Retardo	0.992	0.984	124.008	124.008 > 6.39 Rechaza
Paquetes Perdidos	0.984	0.967	59.860	59.860 > 6.39 Rechaza
Throughput	0.742	0.498	2.983	2.983 < 6.39 No se Rechaza
Tiempo	0.668	0.336	2.012	2.012 < 6.39 No se Rechaza

Tabla 5-34 Resumen de Coeficientes de los polinomios de Regresión de todas las variables independientes

Variables Dependientes	x^0	x^1	x^2	x^3	x^4
CPUs	-17.085	-2.784	-0.9170	6.531	-1813
Memoria	-89.812	0.277	0.02269	-0.0141	1-398
Retardo	1.890	-22.108	77.790	68.679	-91-500
Paquetes Perdidos	-131.384	-0.00547	-0.00276	0.05858	-0.00296
Throughput	1336.892	-0.00879	-0.0283	-0.0000366	0.02351
Tiempo	-5261.845	307.236	-.800	334.425	-112.909

Tabla 5-35 Resumen final incluido el valor de significación (p-value)

Fuente	Suma de cuadrados sin error	Grados de Lib.	Cuadrado medio sin error	F_{abs}	*p*-value	H_o
CPU	54.036	4	13.509	9.060	0.028	*Se rechaza*
Memoria	59.969	4	14.992	1966.211	0.000	*Se rechaza*
Retardo	59.520	4	14.880	124,008	0.000	*Se rechaza*
Paquetes Perdidos	59.014	4	14.754	59.860	0.001	*Se rechaza*
Thoughtput	44.938	4	11.234	2.983	0.157	*Se acepta*
Tiempo de transm.	*40.019*	*4*	*10.019*	*2.012*	*0,258*	*Se acepta*

5.5.13 Interpretación de los resultados

En orden de prelación o significación las variables **memoria**, **retardo**, **pérdida de paquetes** y **CPU** son las variables que influyen en el rendimiento de un EVR. Todas estas variables rechazan H_o.

Las variables **Throughput** con p-value=0.157 y **tiempo de duración** con p-value=0.258, no

influyen en el rendimiento de un EVR, por tanto son aquellas que aceptan H_o

Las variables de mayor significación (p-value=0.000) son la **memoria** y el **retardo**, que son afectadas en función del incremento de CPUs y MVs**.** La variable **pérdida de paquetes** es la que mas rechaza H_o debido a que p-value=0.001, es decir se acerca más a 0.000 y por tanto influye en el rendimiento. En cambio la de menor rechazo es la variable **CPU** con p-value=0.028, que es influyente en el rendimiento de un EVR, pero en menor significación que la memoria, el retardo y la pérdida de paquetes.

Esto además se puede verificar al aplicar la Probabilidad de distribución calculada en el ANOVA, que se compara con **F** "Probabilidad de distribución de Fisher" en las tablas estadísticas, cuyos valores demuestran que la **memoria** con F_{abs}= 1966.211, el **retardo** con F_{abs}= 124.008 y los **paquetes perdidos** con F_{abs}= 59.860 tienen mayor influencia que el CPU con F_{abs}= 9.060.

Por otro lado cada modelo de regresión polinomial múltiple obtenido (véase Tabla 5-34), puede servir para predecir cuántos paquetes se van a perder o qué retardo va a haber como consecuencia de incrementar el número de VMs o de CPUs. La fundamentación responde al siguiente modelo matemático:

$$y = \beta_o + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \varepsilon_0 \quad (7)$$

Donde:

β_o Representa la intersección con el rendimiento del EVR;

$\beta_1, \beta_2, \beta_3, \beta_4$ Constituyen las pendientes de la variable pérdida de paquetes;

x^0, x^1, x^2, x^3, x^4 Constituyen los valores de pérdida de paquetes;

ε_0 Corresponde al error típico.

Este modelo responde a un polinomio de regresión múltiple de cuarto grado en razón de los k tratamientos (k=4).

Tomando de la Tabla 5-34 los coeficientes de regresión polinómica $\beta_1, \beta_2, \beta_3, \beta_4$, y reemplazando en la ecuación (7) el modelo al cual representa la variable pérdida de paquetes es el siguiente:

$$y = -131.384 - 0.005x - 0{,}003x^2 + 0.059x^3 - 0.003x^4 + 21.764 \quad (8)$$

Para un valor aleatorio de x que representa el valor observado de la variable pérdida de paquetes, se utilizará los siguientes paquetes perdidos: 1257.49, 1218.15, 1266.08, 1280.61 extraídos de la Tabla 5-5 que corresponden a las MV_1, MV_2, MV_3, MV_4 respectivamente. Estos valores son reemplazados en el modelo de la ecuación (8) generando el valor de rendimiento de un EVR igual a $-7.384*10^9$, el mismo que se mantendrá constante. En consecuencia se trata de interpolar con el rendimiento de un EVR obtenido, al recalcular algorítmicamente

$\varepsilon_0, \varepsilon_1, \varepsilon_2, \varepsilon_3, ...\varepsilon_n$, errores que se generarían al aumentarse máquinas virtuales, como se observa a continuación:

$\varepsilon_1 = -1.3883x10^{10}$;

$\varepsilon_2 = -1.4973x10^{10}$;

$\varepsilon_3 = -1.5329x10^{10}$.

Al realizar el análisis comparativo entre los errores calculados se obtiene una aproximación del $\pm 10\%$ de error típico en la variable pérdida de paquetes cuando se incrementa una MV. Esto quiere decir que al incrementar una MV se incrementará el $\pm 10\%$ de paquetes perdidos en el mismo EVR. Finalmente se puede observar que el error típico ε sigue incrementando progresivamente.

5.6 Conclusiones

En esta investigación se ha determinado los factores que afectan el rendimiento al realizar un experimento en un EVR. Para este fin se ha diseñado una topología de prueba utilizando Xen como plataforma de virtualización. En este experimento se iban incrementando una a una las MVs, así como el número de CPUs en el interior de cada una de las MVs (EVR), de tal forma que se fue calculando como su incorporación iba afectando el rendimiento del EVR.

Como herramienta de validación fue desarrollado un programa en MatLab, para automatizar el método analítico que permita calcular los coeficientes de Regresión lineal múltiple y el test ANOVA. Sus resultados fueron posteriormente comprobados con los resultados obtenidos al procesar los datos con el software estadístico SPSS. En nuestro caso se ha empleado el test ANOVA para realizar pruebas sobre los factores estimados de los EVR para conocer su influencia en el rendimiento. Su fundamento principal fue realizar la comprobación de una hipótesis nula, la misma que puede ser aceptada o rechazada. Para cumplimentar este método además se calculó el Coeficiente de Determinación, el mismo que da mayor fuerza de interpretación a la relación entre las variables.

En orden de prelación o significación las variables memoria, retardo, perdida de paquetes y CPU son las variables que influyen en el rendimiento de un EVR. Todas estas variables rechazan Ho. Mientras que las variables Throughput y tiempo de duración, no influyen en el rendimiento de un EVR, por tanto son aquellas que aceptan Ho.

En relación al por qué el número de MVs y CPUs influye en el **retardo** y **pérdida de paquetes**, pero no en el **throughput** se puede indicar dos razones: Primera, porque los cambios de contexto entre MVs afectan al retardo; y Segundo porque es conocido que Xen presenta normalmente

pérdida de paquetes en mediciones de EVR, y lógicamente, perderá más cuanto más máquinas virtuales haya. En este mismo ámbito, se debe reconocer que el **ancho de banda** que tiene disponible el procesador es tan grande, que para el número de máquinas virtuales desplegadas no es apreciable la dependencia.

Capítulo 6 Propuesta para desplegar escenarios virtuales de red en entornos distribuidos

6.1 Introducción

Dada la importancia de contar con una infraestructura base que permita desplegar automáticamente EVR independientemente de la plataforma de virtualización, y desde el punto de vista del modelo de infraestructura de despliegue (véase Fig. 1-2), este capítulo presenta algunas propuestas de posibles infraestructuras para conseguirlo.

Para llevarlo a cabo se ha seguido una metodología incremental, partiendo de un modelo basado en el despliegue dinámico de escenarios virtuales distribuidos con VNUML (*Virtual Network User Mode Linux*) [Vnuml] (vease apartado 6.4). Tras esto se ha realizado una implementación basada en el desarrollo de una interfaz de servicios Web sin estado y por último un modelo de interfaz de servicios Web con estado, integrando Grid y WSRF. Como marco conceptual se ha utilizado RM-ODP (*Reference Model for Open Distributed Processing*) [RMODP08], que integra aspectos relacionados con la transparencia, distribución, interoperabilidad y portabilidad de sistemas distribuidos.

6.2 Declaración del problema y motivación

Las tecnologías de virtualización abarcan una variedad de mecanismos y técnicas que hacen frente a problemas computacionales como la seguridad, el rendimiento y la disponibilidad de los recursos de hardware y software [Figuereido]. En los últimos años estas tecnologías han facilitado la prestación de servicios en entornos distribuidos. En este capítulo se proponen distintas soluciones para el despliegue de escenarios con MVs en entornos distribuidos, aprovechando como plataforma de experimentación el Proyecto PASITO (Plataforma de Análisis de Servicios de Telecomunicaciones).

PASITO es una infraestructura pública construida sobre la red académica española RedIRIS, basada en la interconexión de grupos telemáticos de investigación, que ofrece un laboratorio de pruebas distribuido para construir, depurar y evaluar escenarios de servicios de telecomunicaciones. Aunque esta plataforma tiene definidos varios experimentos, uno de los más

relevantes es probar las técnicas de virtualización para analizar las posibilidades de esta tecnología en las redes de comunicaciones.

La creación de varios escenarios virtuales desplegados en diversos dominios de administración incrementa su complejidad, por lo que se requiere de un método de procesamiento computacional *(Computación Distribuida)* y de una estrategia (*Computación Grid*) que proporcione una plataforma de ejecución segura, haciendo posible la coordinación de individuos, instituciones, recursos físicos y virtuales en topologías distribuidas [Foster01].

Esta necesidad de integrar tecnologías de virtualización y entornos Grid ha sido analizada por [Figuereido03] [Sundararaj04] y [Wang06] para desplegar EVR en plataformas distribuidas. Incluso en [Garbacki07] se exponen las ventajas de este tipo de virtualización de equipos en el contexto de los sistemas Grid, como son: la posibilidad de compartir recursos en una forma controlada, la capacidad de migración y la facultad de controlar la ejecución de recursos virtuales. Contrariamente a esto, nuestra visión es aprovechar las ventajas del Grid para mejorar la distribución de escenarios de redes virtuales. Desde este punto de vista, algunos aspectos aún no han sido resueltos. Por ejemplo, el despliegue automático y la distribución de un único EVR en diferentes servidores, el control de los recursos virtuales a través de interfaces de servicios Web, el acceso seguro a los recursos, etc.

En este contexto, por un lado se propone el uso de servicios Web para la gestión de escenarios virtuales distribuidos, y por otro, incorporar WSRF (*Web Service Resource Framework*) [WSRF08], con el fin de utilizar Grid para realizar dicha distribución de escenarios.

6.3 Definición de la Infraestructura base y requisitos

Para estos experimentos se ha considerado conveniente aprovechar una plataforma de red física existente y útil para la experimentación de servicios de telecomunicación denominada PASITO. Su infraestructura es distribuida y se ha creado sobre la red académica nacional RedIRIS con la colaboración de varios grupos de investigación especializados de España. RedIRIS aporta la red troncal de comunicaciones, parte de la infraestructura de prueba y el middleware de soporte para facilitar el uso de la plataforma. Los grupos de investigación aportan otra parte de los recursos, que junto con los desplegados por el proyecto completan la plataforma distribuida de servicios propuesta.

De los distintos experimentos que se realizan en PASITO, algunos se centran en aplicar tecnologías de virtualización que ayudarán a aprovechar la plataforma y hacer uso eficiente de sus recursos. Dentro de esta iniciativa, se consideran los siguientes requisitos mínimos para formalizar una infraestructura base:

- Configuración de varios EVR para pruebas, en diferentes equipos o nodos, para análisis de

prestación de servicios de red;

- Implementación de técnicas de seguridad, para el control de acceso a sistemas distribuidos, tanto para los escenarios de prueba como para la arquitectura en sí. Es decir: probar y mantener el control de los usuarios y recursos autorizados a utilizar la plataforma;
- Planificación y control para la ejecución simultánea de varios procesos y la posibilidad de que varios usuarios accedan simultáneamente a un recurso (CPU, memoria, disco). Por tanto, es necesario que exista un planificador que procese las peticiones de recursos que pueden pertenecer al mismo o a diferentes dominios, obteniendo información actualizada del grado de utilización y de la disponibilidad de los mismos.

La topología de PASITO a alto nivel se ilustra en la Fig. 6-1. Está formada por una red de área extensa, compuesta por diferentes nodos interconectados, en diversos dominios de administración (el nodo *n* constituye una representación genérica). Cada nodo esta conectado a la red principal mediante una interfaz del encaminador, que provee las rutas. En un mismo nodo pueden existir varias subredes y varias redes de área local virtuales VLAN (*Virtual Local Area Networks)*. A éstas se conectan uno o más servidores físicos. Internamente, es deseable que cada servidor pueda desplegar uno o más escenarios virtuales compuestos por puentes, encaminadores, MVs, que están conectados mediante TCP/IP.

Por su parte, los clientes, ubicados en cualquier nodo, son capaces de desplegar escenarios de red virtuales configurados previamente, ya sea en sus servidores o en servidores de otros nodos, e interactuar con dichos escenarios.

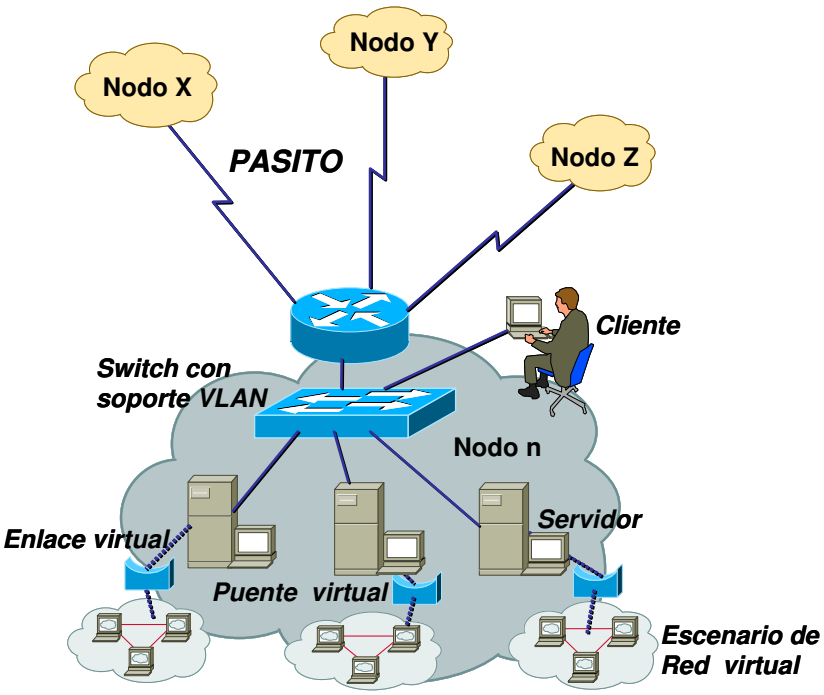

Figura 6-1 Escenario de interconexión de PASITO

Para cumplir estos requisitos, a continuación se describen los métodos, técnicas y

especificaciones que serán usadas en las soluciones propuestas en este capítulo (véase Fig. 6-2).

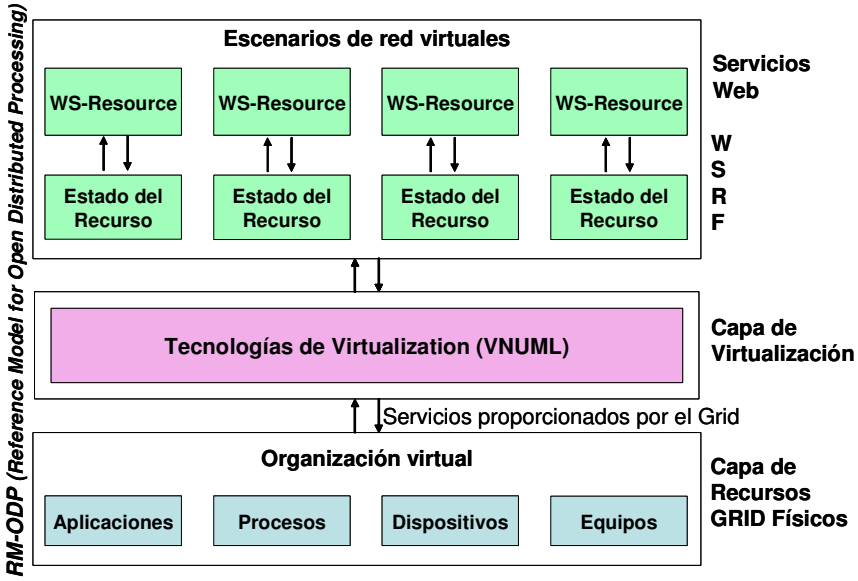

Figura 6-2 Marco conceptual y arquitectura de capas.

6.4 Soluciones Propuestas

En esta sección se describe primeramente VNUML y el sistema EDIV de gestión distribuida de escenarios (apartado 6.4.1) para posteriormente describir las mejoras al modelo basadas en servicios Web y Grid (apartado 6.4.2 y 6.4.3) y que constituyen la contribución principal de este artículo.

6.4.1 Modelo basado en escenarios distribuidos con VNUML

La herramienta VNUML [Vnuml] permite la gestión automatizada de escenarios de red virtuales, en los que un conjunto de MVs se interconectan formando topologías arbitrariamente complejas y extensas.

El ciclo habitual de trabajo con VNUML comienza con el diseño del escenario deseado mediante un lenguaje basado en XML. Los elementos principales de dicho lenguaje son las etiquetas <*MV*> (con las que se definen las MVs y sus atributos: sistema operativo, interfaces de red, direccionamiento IP, rutas, etc.) y <*net*> (con las que se definen las redes que interconectan las interfaces de las MVs). Una referencia completa del lenguaje puede encontrarse en [Vnuml].

La principal característica del lenguaje VNUML es su naturaleza descriptiva y de alto nivel. Por un lado, es sencillo e intuitivo, en contraposición con las aproximaciones procedurales (ej. el lenguaje del simulador de red NS-2 [NS2-08]). Por otro lado, el usuario se concentra en la especificación del escenario deseado. Es decir, no necesita conocer los detalles de bajo nivel de la

tecnología de virtualización subyacente utilizada para crear redes y nodos (ya que esto lo hace automáticamente la herramienta VNUML, como se describe a continuación), simplificando enormemente su trabajo y maximizando su productividad. Más aún, existe una interfaz gráfica [VnumlGUI08] que permite diseñar escenarios visualmente, sin tener que editar el texto XML de la especificación.

Una vez que el escenario ha sido especificado en un fichero, el intérprete de VNUML lo toma como entrada, y lo implementa en un equipo físico anfitrión (*host*) mediante MVs UML [Dike06] y redes virtuales (emuladas mediante procesos en espacio de usuario o puentes virtuales implementados a nivel de sistema operativo). Una vez creado el escenario, el usuario interactúa con él, pudiendo utilizar de nuevo VNUML para automatizar la ejecución de secuencias de comandos en las MVs. Finalmente, VNUML permite la eliminación de las máquinas y redes virtuales que conforman el escenario, una vez se ha finalizado su uso, liberando los recursos en el equipo anfitrión. Creación, ejecución de comandos y eliminación son las tres operaciones de gestión que VNUML realiza sobre los escenarios.

Tradicionalmente, VNUML ha adoptado un enfoque mono-host (es decir, despliegue de todo el escenario en el mismo host). Si bien era posible la integración de escenarios en hosts distintos (además de equipos reales externos, ej. *routers* Cisco), cada uno había de ser gestionados independientemente, sin tener una visión integrada. No obstante, el proyecto EDIV (Escenarios Distribuidos con VNUML) entre el DIT de la UPM y Telefónica I+D ha desarrollado un recubrimiento de VNUML para permitir gestión de escenarios distribuidos transparentemente, implementando la arquitectura descrita en [Galán07].

La arquitectura del EDIV (véase la Fig. 6-3) se basa en un conjunto de servidores interconectados localmente (típicamente, con *switches*) y un controlador de despliegue. Dicho controlador procesa especificaciones de escenario VNUML y se encarga de invocar un módulo segmentador para realizar la asignación de cada máquina virtual a uno de los servidores de despliegue concretos. Se utiliza una aproximación modular, de forma que el algoritmo de asignación puede ser desarrollado independientemente. En el primer prototipo, se han considerando tres casos: *round robin* simple, *round robin* ponderado (ej., usando la carga como métrica de ponderación) y asignación explícita (el usuario especifica explícitamente que MVs se asignan a cada host).

Basándose en el resultado del segmentador, el controlador coordina las operaciones necesarias para implementar y gestionar el escenario en modo distribuido. En concreto, se encarga de dividir la especificación global en sub-especificaciones para cada host (cada una de las cuales contiene un "fragmento" que es procesado por la instancia local de VNUML en dicho host) y de preparar la interconexión de MVs en distintos hosts cuando es preciso por necesidades del escenario (configurando convenientemente redes de área local virtuales 802.1q [Vlan01] en los *switches* de

interconexión de los hosts). La interfaz de operación entre el controlador de despliegue y los hosts se basa en comandos (ej., SSH).

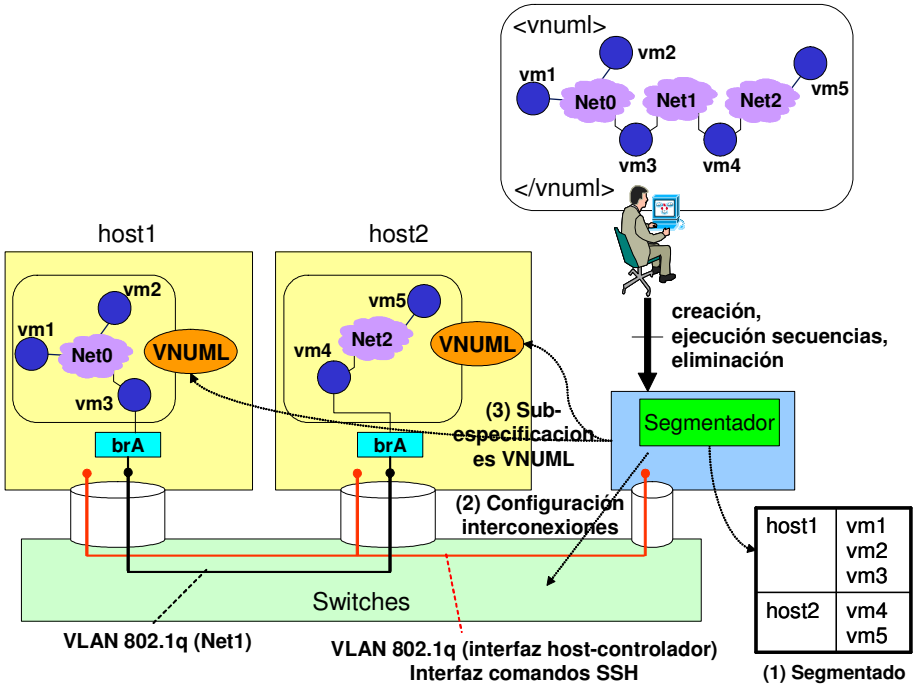

Figura 6-3 Escenario virtual distribuido con VNMUL

Es de destacar como EDIV cumple el objetivo de transparencia para el usuario. La interfaz que ofrece el controlador de despliegue en múltiples servidores es la misma que ofrece VNUML clásico en contexto de un único servidor (lenguaje de especificación VNUML y los tres modos de gestión: creación, ejecución de secuencias y eliminación). Por tanto, desde el punto de vista del usuario, no hay diferencia de uso y, de hecho, ni siquiera tiene por qué conocer el detalle de cómo se han asignado las MVs a servidores específicos (el controlador de despliegue es el que maneja esta correspondencia).

6.4.2 Modelo basado en interfaz de Servicios Web

Aprovechando las capacidades de VNUML, expuestas en el apartado anterior (6.4.1) y como otra solución, a continuación se explica la implementación de una interfaz para desplegar EVR utilizando servicios Web.

Considerando el escenario de la Fig. 6-4 el cliente (desde un nodo de PASITO) invoca un servicio de despliegue de escenarios virtuales previamente configurados, a través de una interfaz específica en el equipo servidor.

Para esta implementación se ha definido el algoritmo que se muestra en la Fig. 6-5. En el lado del cliente, se ha desarrollado un programa Java, que toma por línea de comandos el URL *(Uniform Resource Locator)*, el nombre del archivo XML que contiene la descripción del escenario virtual de red VNUML (preparada previamente) y la operación a realizar (despliegue, repliegue). Dicho

cliente incluye un conjunto de clases generadas a partir de la descripción de la interfaz del servicio en WSDL (*Web Service Description Language*) para acceder al servicio requerido.

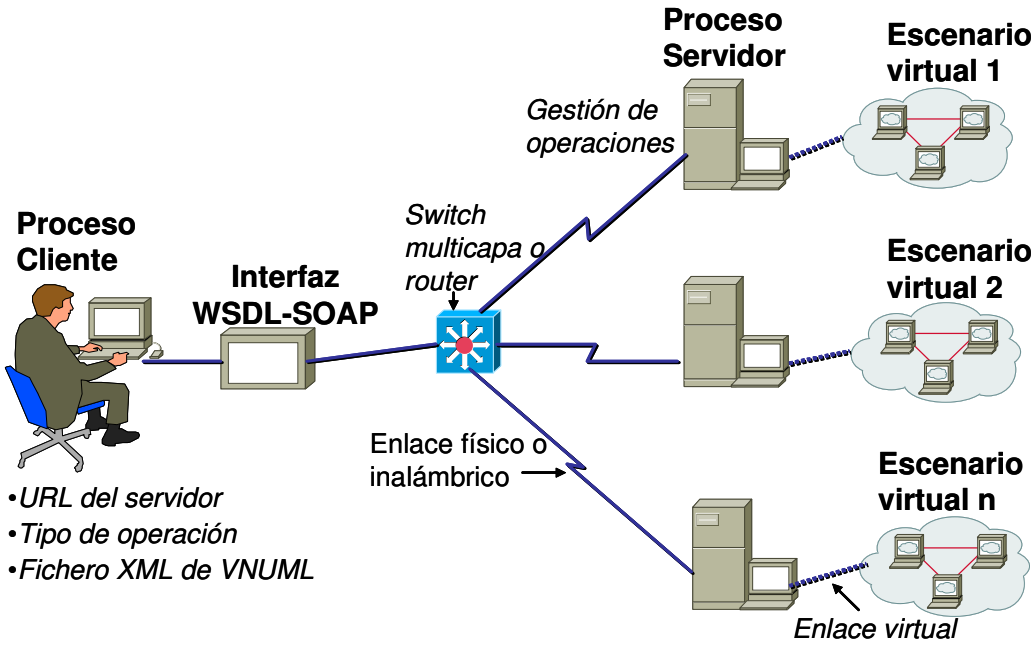

Figura 6-4 Diseño lógico – físico de un nodo de PASITO de Servicios. Web.

En el lado del servidor, en primer lugar se inicia el servicio en los servidores donde se debe desplegar el escenario virtual. Una vez que un servidor recibe una solicitud invoca la operación contenida en los argumentos recibidos como parámetros. Luego, en tiempo de ejecución interactúa con VNUML para que se despliegue el escenario de virtualización contenido en el fichero de configuración XML, que fue enviado como parámetro especificado. Finalmente devuelve el valor de la operación al cliente y el control del programa.

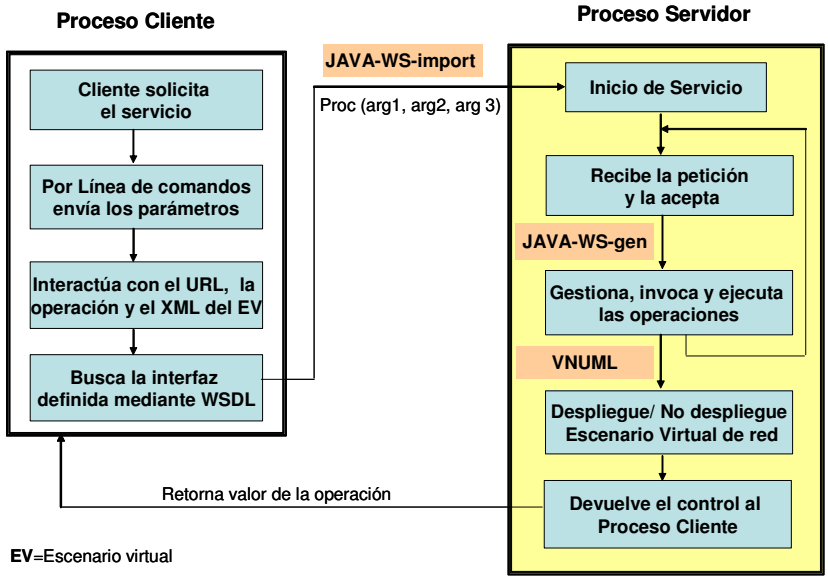

Figura 6-5 Componentes, flujo de ejecución e invocación de operaciones

Como se puede observar, respecto al modelo descrito en el apartado 6.4.1, esta solución aporta la implementación de un servicio Web, en el que se ha definido una interfaz WSDL, mediante la cual se describen las operaciones del servicio y se detallan los protocolos y los formatos de los mensajes para interactuar con los escenarios de virtualización. La definición de la interfaz en WSDL permite la ejecución de métodos entre diferentes plataformas, por lo que se puede tener diferentes procesos clientes programados en diferentes lenguajes y bajo cualquier plataforma accediendo a diferentes servidores.

Sin embargo, este modelo no facilita el control de los recursos, sobre todo cuando han sido desplegados varios escenarios virtuales. Esto se debe a utilizar servicios Web sin estado (*stateless)*, donde no existe un registro de estado entre las llamadas al servicio Web. Es decir, no se conserva el valor de los registros después de cada invocación, por lo que resulta complejo gestionar los recursos utilizados. Este problema ha motivado la solución que se expone en el siguiente apartado.

6.4.3 Modelo Basado en interfaz Grid y WSRF

Este diseño adopta la perspectiva de servicios Web con estado (*stateful)*, lo que significa que se podrá mantener la información de los recursos en cada invocación o subsiguiente ejecución del servicio (WS-Resource). Esto redunda en un mejor control de los recursos "escenarios virtuales". Para su realización, en primer lugar requiere de una plataforma Grid que distribuya el procesamiento y la capacidad de cómputo. Además debe dimensionar los recursos disponibles y los nuevos requisitos para atender las peticiones. En segundo lugar se ha precisado de los conceptos fundamentales de RM-ODP relacionados con la *transparencia de localización* que permite acceder a un objeto o servicio sin ser consciente de la localización del mismo. En tercer lugar, requiere de las especificaciones WSRF, que permiten tratar los mensajes entre servicios Grid de forma abstracta para que los recursos puedan interactuar unos con otros.

Inicialmente, se debe modelar el ciclo de vida de un WS-Resource, cómo es el servicio y cómo interactúa con el recurso "EVR", sus operaciones y propiedades. Para ello se sigue el método indicado en [Foster05]. Como parte de un Grid, el ciclo de vida es el tiempo de duración de un WS-Resource definido por el periodo entre su creación y su destrucción, (Fig. 6-6). Cada WS-Resource "EVR" se crea a través de la Factoría de Servicios y es tratado de forma independiente, pues se le asigna una única identificación y asociación con el servicio Web. Múltiples instancias de escenarios virtuales pueden ser creadas o destruidas vía servicios Web, lo que permite controlar los valores de las propiedades de cada escenario de forma independiente mediante implementaciones específicas.

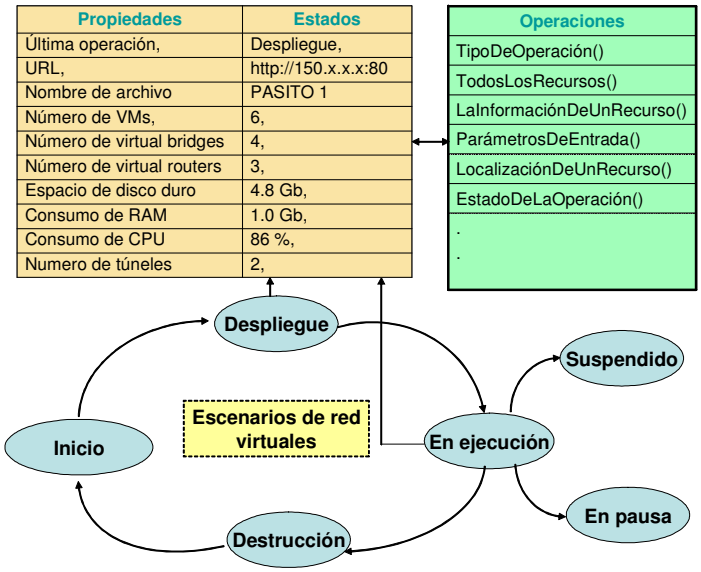

Figura 6-6 Ciclo de Vida del recurso "EVR"

A continuación, se define el procedimiento y la arquitectura necesaria que permita: ***i)*** procesar las peticiones de recursos en el Grid, ***ii)*** proveer las capas de virtualización y ***iii)*** modificar el estado de los recursos. Concretamente, se modela la interfaz que soporte el diseño expuesto en la Fig. 6-1 y el ciclo de vida, Fig. 6-6. Para una mayor claridad, se describen los componentes y el funcionamiento en la Fig. 6-7.

El middleware consiste en la definición de una interfaz de servicios con WSDL+WSRF, un demonio del servicio Web que se está ejecutando y una aplicación de despliegue de servicios. El middleware, es una implementación específica de software (similar al Middleware del Grid que cumple funciones de integración de todos los recursos que participarán en el Grid). Esta implementación interactúa con el sistema de virtualización (invoca la capa de virtualización para conseguir el entorno de ejecución) y es la encargada de interactuar con el Planificador del Grid para que procese las peticiones de los recursos.

Este middleware además interactúa con la Factoría de Servicios, la misma que crea el Contenedor de Recursos (*Resource Home*) y las instancias de Servicio de acceso a recursos lógicos. Luego mediante la Instancia de servicios entrega la identificación de los WS-Resources "EVR" asignados por la Factoría para ser gestionados por el Contenedor de Recursos. Acto seguido, este contenedor hace el descubrimiento y monitoriza los recursos virtuales en base al archivo XML que contiene la descripción del escenario virtual que fue enviado desde el proceso cliente como parámetro. VNUML los activa creando el entorno virtual y dicha información es registrada por el Grid en el documento XML de propiedades del recurso (*Resource Properties*) [Czajkowski04]0 [Foster05] en donde residen los valores o estados de cada escenario virtual, su identificador y una asociación al servicio Web. Finalmente, la Factoría devuelve el URI *(Uniform Resource Identifier*) del nuevo servicio al cliente que interactúa con el servidor como resultado de la llamada inicial.

Para cada nueva instancia de WS-Resource (es decir, para cada nuevo servicio Web que invoque el despliegue del recurso "EVR"), la Factoría de servicios la creará, asignará un nuevo identificador y creará la nueva asociación respectiva, de manera similar a como se explica en [García06] [Foster05] Esto permite acceder a uno o más escenarios virtuales por un cliente o a un mismo escenario virtual por varios clientes. Todo esto ha sido posible gracias a las diversas especificaciones de WSRF. La forma de implementación está detallada en [Sotomayor].

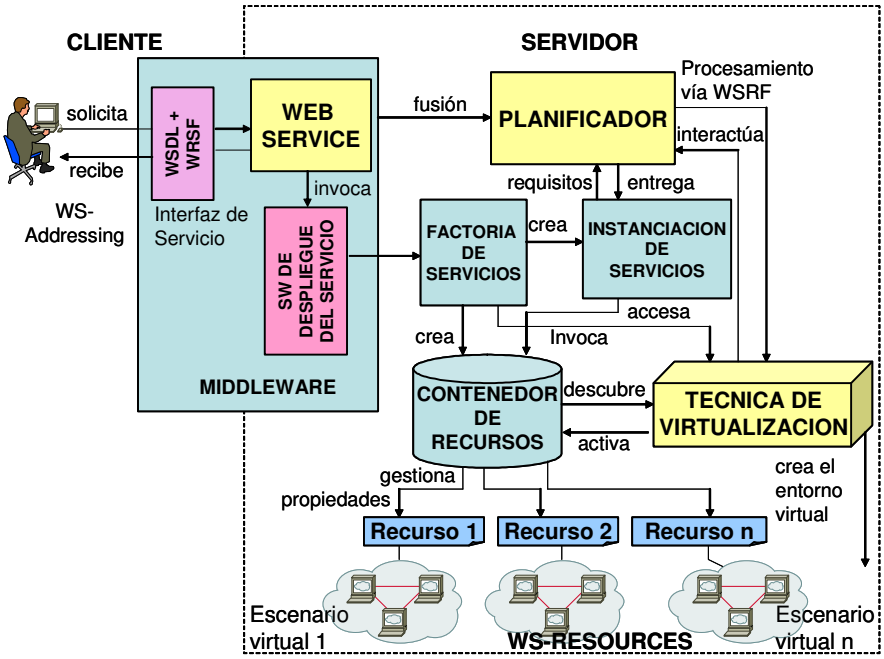

Figura 6-7 Procedimiento y arquitectura del funcionamiento de la interfaz

Por otra parte, el cliente no puede manipular directamente instancias de los recursos; lo hace a través de las interacciones con el servicio que cumple las especificaciones WSRF. WSRF pasa la identificación del recurso cuando ocurre una interacción de mensajes entre el cliente y el WS-Resource. El cliente usa *WS-Addressing* para referenciar el servicio (*EndPoint Reference),* e identificar la dirección de WS-Resource desplegado en un punto de la red dado. A continuación se invoca las operaciones establecidas en el ciclo de vida del WS-Resource, que devolverán los valores de las propiedades de los recursos "escenarios virtuales". Las imágenes de las MVs incluidas en el fichero de descripción del escenario XML de VNUML estarán disponibles en los recursos remotos y se desplegarán, siempre que se haya cumplido el proceso con éxito.

6.5 Discusión

EDIV proporciona una solución al problema de la gestión de escenarios virtuales distribuidos (desplegados en un conjunto de servidores interconectados localmente) en los que luego puedan realizarse pruebas y experimentos, de forma integrada y transparente para el usuario. Sin embargo,

en el contexto de PASITO, donde los servidores de despliegue no se encuentran interconectados localmente sino a través de un troncal de red de RedIRIS, serían necesarias adaptaciones. En concreto, pasar de utilizar redes 802.1q al uso de túneles IP para la interconexión de MVs en distintos servidores. Otra de las limitaciones de EDIV (heredada de VNUML) es que la fuerte orientación a escenarios restringe las posibilidades de realizar operaciones de gestión sobre MVs individualmente (ej., añadir una nueva máquina virtual a una de las redes del escenario sin tener que re-desplegarlo entero). Finalmente, la interfaz controlador-servidor está basada en comandos, lo que, si bien es una aproximación directa y sencilla, implica poco formalismo desde el punto de vista de la gestión.

En el modelo de servicios Web sin estado, la interacción de los procesos sigue un patrón de petición–respuesta. Funciona bien para desplegar los escenarios virtuales desde el proceso cliente a varios servidores. Su fortaleza reside en la definición de su interfaz WSDL, que permite que cualquier cliente Web pueda invocarlo sin tener que conocer nada acerca de los detalles de la implementación del servicio, o sobre que plataforma está funcionando. Sin embargo, tiene la restricción de que en cada invocación no se mantiene el estado o las propiedades de los recursos, con lo cual se dificulta el control, planificación y dimensionado de los recursos. Otro problema no resuelto es el tema de seguridad de acceso a los recursos.

En el modelo de servicio Web basado en Grid, se han agregado las especificaciones WSRF a la definición WSDL, con lo cual se puede interactuar con los WS-Resources con las mismas ventajas de interoperabilidad de un servicio Web definido en WSDL. Otra ventaja es permitir que todos los recursos puedan comunicarse con independencia de su localización. Este modelo utiliza la Factoría de Servicios del Grid, de forma que en lugar de tener un único servicio compartido por todos los clientes se tiene una Factoría que crea instancias de *WS-Resource* individuales. Cuando el cliente invoca a una operación se accede a la Instancia de Servicios y no a la Factoría, con lo cual se puede crear un WS-Resource por cliente, o varios por cliente o uno para varios clientes, lo cual es una gran ventaja comparado con los dos modelos anteriores. Resuelve además cuestiones de planificación de recursos y seguridad de acceso. Por contra, existe mayor complejidad en su diseño, es muy laborioso en su implementación y requiere un mayor número de componentes de hardware y software. La Tabla 6-1 resume el cumplimiento de los requisitos establecidos en el apartado 6.3:

Tabla 6-1. Cumplimiento de los requisitos

Requisito	EDIV	Web Service	Grid+WSRF
1	SI	SI	SI
2	NO	NO	SI
3	NO	NO	SI

6.6 Trabajos relacionados

Es necesario mencionar la existencia de otras herramientas de gestión de infraestructura virtualizada además de VNUML. Herramientas como VirtualCenter [VMwareVC] o Enomalism [Enomalism] son bastante avanzadas en muchos aspectos (consolidación, balanceo de carga en los hosts, etc.) pero carecen de flexibilidad para la creación de topologías arbitrarias. Más parecidas a VNUML son NetKit [Pizzonia08] y MLN [Begnum06], también orientadas a gestión de escenarios, pero sin considerar despliegues multi-hosts como en el caso de EDIV.

En cuanto a la integración de Grid y virtualización, VIOLIN [Ruth05] es un prototipo sobre PlanetLab para ejecutar aplicaciones distribuidas. IN-VIGO [Matsunaga05] es un middleware para recuperación automática de fallos. En [Zhao06] se describe un middleware para gestión de servicios de VMs. Estos trabajos son diferentes al descrito en el presente capitulo, al no incorporar las especificaciones WSRF.

A continuación se mencionan los enfoques que consideran la aplicación de WSRF. En [Keahey05] se introducen los conceptos de espacio de trabajo virtual (virtual workspace) para desplegar automáticamente VMs en la arquitectura Grid, usando Xen y VMware Workstation. Una extensión de este trabajo se describe en [Kecskemeti08], donde se define una solución de despliegue de servicio automático de dispositivos (appliances) virtuales para servicios Grid. Nuestra investigación se diferencia de estas propuestas, ya que distribuye los escenarios virtuales utilizando VNUML, que utiliza un lenguaje de especificación de alto nivel. Por otra parte, ellos no determinan en tiempo real si la distribución y despliegue del escenario virtual será en el mismo servidor o en diferentes servidores.

Respecto a trabajos con enfoques similares, en [Zhao05] y [Zhao06] se analiza la creación y gestión de sesiones de sistemas de archivos con VMs VMware basadas en sistemas Grid y WSRF. Nuestra investigación se diferencia de estos esfuerzos, pues distribuimos escenarios virtuales con instanciación de VMs mediante un lenguaje de descripción de escenarios, manipulando el estado de los recursos vía WSRF.

En cuanto a la distribución de tareas, en [Rubio07] se presenta el despliegue de VMs en un Grid GT4. Su aplicación consiste en el encapsulado del espacio virtual (*virtual workspace*) en una tarea (*job*) Grid, incorporando GridWay, que gestiona la interacción con los Servicios GRAM y GridFtp e interactúa con VMs implementadas con Xen [Barham03]. En este trabajo no se han utilizado los beneficios de las especificaciones WSRF pero su concepción ha sido útil en nuestra investigación. En [Turjanski05] se configura un laboratorio Grid basado en VMs utilizando GT4 y una implementación WSRF. No modelan el ciclo de vida del "EVR". Finalmente, en [Libvirt] se describe la creación de entornos de ejecución desplegados dinámicamente utilizando VMs con WSRF. Lo que en esencia hace es arrancar VMs para que en ellas se ejecute un job del Grid. En

nuestro caso, la tarea *(job)* a entregar al Grid es el despliegue de un escenario virtual.

Todos estos enfoques plantean soluciones parciales al problema de distribuir escenarios virtuales previamente configurados. No le asignan al Grid la tarea de desplegar escenarios virtuales y no han modelado el ciclo de vida de un escenario virtual con WSRF.

6.7 Conclusiones

En este trabajo se han presentado soluciones para distribuir escenarios virtuales, como entornos de experimentación para la plataforma PASITO. Partiendo de EDIV se han ampliado sus soluciones recubriendo VNUML con un conjunto de capas para proporcionar un servicio estándar, seguro e inter-operable de virtualización distribuida, incorporando un nivel de abstracción de tecnologías de virtualización entre el Grid y WSRF. Como apoyo conceptual, se ha adoptado RM-ODP como modelo de referencia para establecer transparencia de localización, acceso y prestaciones. Se ha modelado el procedimiento y la infraestructura para el despliegue y distribución de escenarios virtuales de red mediante servicios Web y después mediante servicios Grid, utilizando VNUML como lenguaje de especificación de dichos escenarios. La discusión realizada ha descrito las ventajas y desventajas de cada solución propuesta, que han sido orientadas para mejorar el control, uso eficiente de los recursos y la seguridad de acceso de redes virtuales ejecutadas en entornos distribuidos.

Tal como se muestra en este capítulo, cada una de estas propuestas no resuelve los principales problemas definidos en el apartado 6.2. Sin embargo al intentar definir una de las propuestas, se ha establecido como una arquitectura base el diseño lógico y físico de un nodo de PASITO similar al que se ilustró en la Fig. 6-4, del apartado 6.4.2, a fin de desplegar un EVR independientemente de la plataforma de virtualización subyacente. Estas propuestas han sido publicadas en [Fuertes08b] y [Galán09].

El siguiente capítulo parte de esta infraestructura base y ha sido enfocado en caracterizar un EVR mediante un modelo que permita normalizar su despliegue automático, independiente de la plataforma de virualizacion a fin de resolver el problema de la interoperabilidad y del despliegue automático.

Capítulo 7 Modelo genérico para caracterizar entornos virtuales de red

7.1 Introducción

Con los aportes obtenidos en los capítulos anteriores, pero sobre todo en la infraestructura base establecida en el capítulo 6, los siguientes apartados exponen el diseño y materialización de la caracterización de dicha infraestructura mediante un modelo genérico, esto, desde el punto de vista del modelo de gestión de configuración (véase Fig. 1-2).

Este capítulo parte del análisis realizado en el Capítulo 2, apartado 2.4 de los enfoques existentes para modelar sistemas de virtualización y algunas de sus implementaciones, enfatizando que es lo que ofrecen y que limitaciones tienen. Con estos resultados, se ha diseñado una propuesta para modelar un EVR basado en CIM (véase apartado 7.3). Así mismo, con el fin de comprobar su viabilidad, se ha implementado un sistema de gestión que incluye una interfaz de aplicación (API, *Application Program Interfaz*) en el cliente (véase apartado 7.4) que invoca a un administrador de objetos CIM (CIMOM, *Common Information Model Object Manager*) para recuperar los datos apropiados, y que, independientemente de la plataforma de virtualización, permita desplegar automáticamente EVR.

7.2 Declaración del Problema y Motivación

Tal como se ha establecido en el apartado 1.2, del Capítulo 1 de esta Tesis Doctoral, existen varios problemas relacionados con la gestión de un EVR. En primer lugar, existe la complejidad en la construcción y el despliegue de una topología de red virtual. Esto es debido a la diversidad de las tecnologías de virtualización disponibles, cada uno con sus propias peculiaridades y herramientas de gestión. Como consecuencia de ello, cada plataforma requiere un modelo específico, que podría ser ineficaz y un claro inconveniente. En segundo lugar, el despliegue y la liberación de los recursos virtuales en un EVR es un paso crítico, especialmente en entornos complejos que requieren una rápida re-configuración entre los experimentos, con el fin de maximizar su uso. En tercer lugar, la mayoría de las herramientas de virtualización tienen un enfoque centralizado (es decir, mono-host) [Galán08b]. Esto significa que el despliegue y gestión de cada EVR tienen que hacerse en un equipo físico, que resulta en una baja escalabilidad que impide la creación de grandes y complejos EVR.

Con el fin de ofrecer soluciones a estos problemas, el trabajo propuesto en este capítulo se centra

en la representación de un EVR buscando un modelo genérico que la caracterice y que sea independiente de la plataforma de virtualización en el que están desplegados. Por lo tanto, la cuestión a resolver es el diseño y la construcción de un modelo genérico para describir el mínimo EVR a desplegarse en cualquier plataforma de virtualización subyacente. Para habilitar la gestión de interoperabilidad de un EVR, este modelo pertenece a la serie de especificaciones derivadas de modelos normalizados aceptados por la DMTF-CIM (véase apartado 2.4) y otros. Además, este modelo considera la extensibilidad a futuro para evitar la obsolescencia, ya que pueden surgir las nuevas exigencias en el futuro.

Como consecuencia de la diversidad de plataformas de virtualización, un EVR normalmente no puede ser abstraído desde las características específicas de la tecnología de virtualización utilizada, lo cual es un problema desde el punto de vista de la interoperabilidad. Aunque existen algunas herramientas de virtualización (como por ejemplo, VNUML [Galán09], Netkit [Pizzonia08], MLN [Begnum06], etc descritas en el capítulo 2, apartados 2.2.3 y 2.2.5) que modelan un EVR en su conjunto y ponen en práctica los procedimientos para su creación y despliegue, que están acoplados a un tecnología específica, sin tener en cuenta la independencia entre el EVR y la plataforma de virtualización. Por lo tanto, como una solución, el principal objetivo de esta tesis doctoral y su contribución principal es definir un modelo genérico que permite la caracterización de una EVR, independientemente de la plataforma de virtualización.

7.3 Diseño y construcción del modelo genérico para entornos virtuales de red basado en DMTF-CIM

7.3.1 Requisitos de Diseño

Este modelo requiere un mayor nivel de abstracción, y por lo tanto debe ser lo suficientemente flexible para responder a los procesos asociados con la construcción automática, despliegue, gestión y publicación de un completo EVR. Este modelo también debe ser compatible con múltiples plataformas de virtualización.

En este contexto, para el diseño de este modelo genérico, se considera la reutilización del enfoque DMTF-CIM, en lugar de diseñar una jerarquía de clases partiendo desde cero. Como se discutió en las secciones anteriores (véase apartado 2.4), CIM contiene conceptos comunes y los modelos comunes de los sistemas informáticos y elementos de red que pueden ser reutilizados. CIM también es extensible a fin de permitir extensiones específicas a la definición común de sus elementos administrados [DMTFDSP0111].

7.3.2 Arquitectura Básica

La Fig. 7-1 muestra la abstracción de la arquitectura en la que se basa el modelo, que es coincidente con la infraestructura base definida en la Fig. 6-4. En un nivel de abstracción mas particularizado, esta se compone de computadores físicos conectados en una red, interconectadas a través de enlaces lógicos (VPN, *Virtual Private Networks*), (VLAN, *Virtual Local Area Network*), etc o enlaces físicos (generalmente, en el caso de LAN). En particular, dentro de cada host físico es posible crear y desplegar una o más redes de topología arbitraria, llamada EVR, utilizando una plataforma de virtualización dada (por ejemplo, Xen o VMware Server). En cada uno de estos EVR, la figura también muestra cómo las MVs están interconectadas a través de conexiones virtuales. Cada MV tiene su propia tarjeta virtual de interfaz de red (VNIC, V*irtual Network Interfaz Card*) y configurada su respectiva la dirección IP. Además, esta arquitectura permite que uno o más EVR puedan ser desplegados en el mismo host o en diferentes hosts, aunque actualmente un mismo EVR debe ser desplegado en el mismo host). Por último, cabe señalar que cada MV tiene una dependencia en un solo host dentro del mismo EVR (es decir, MV no se pueden migrar desde un host a otro).

Como se puede deducir de la Fig. 7-1, la cuestión principal es cómo modelar las MVs, la asignación de sus recursos y sus enlaces para la conectividad. En los siguientes apartados, se muestra cómo este modelo se ha definido y cómo CIM y las implementaciones anteriores han sido reutilizadas.

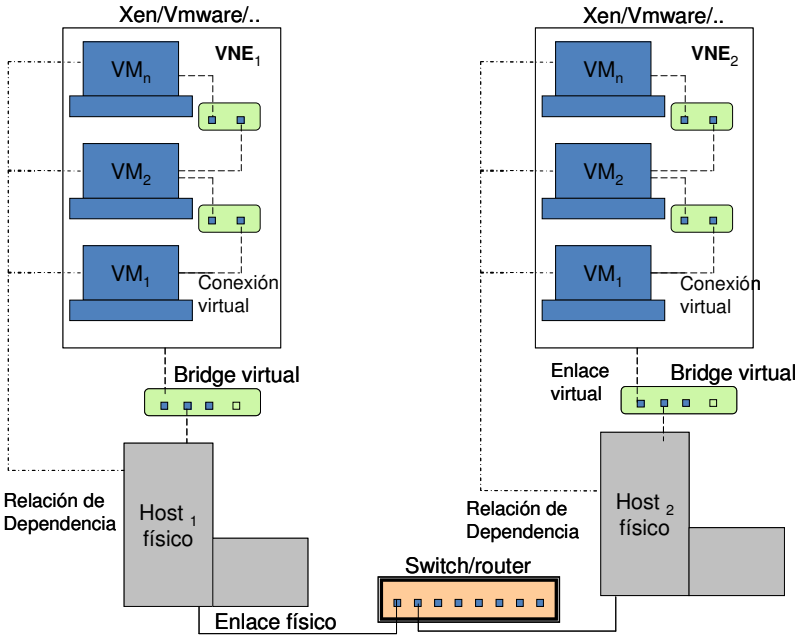

Figura 7-1 Arquitectura Básica

7.3.3 Marco conceptual para el modelo genérico

A fin de cumplir los requisitos de diseño que se explicó tanto en el apartado 7.2 así como en la mencionada Arquitectura base, la Fig. 7-2 muestra la solución propuesta para obtener un modelo genérico que permita gestionar el despliegue automático de un EVR independientemente de la plataforma de virtualización subyacente.

La Fig. 7-2 muestra un esquema conceptual que describe los conceptos del modelo genérico. De hecho, hay varias plataformas de virtualización (1), que ofrecen software basado en soluciones de virtualización dentro de un equipo anfitrión (host) (2). Ambos han sido modelados por los perfiles de virtualización DMTF CIM y otros enfoques (3). Así, el EVR es modelado utilizando CIM y el esquema CIM de Modelado de Red [DMTFDSP0152], que describe y administra los sistemas de red, servicios de interconexión lógica, protocolos y Calidad de Servicio (QoS) (4). Para lograr la interoperabilidad, el usuario debe realizar las operaciones y solicitar propiedades y métodos almacenados en el modelo de objetos genéricos (5). Con esta información, el usuario es capaz de desplegar automáticamente un EVR, para múltiples plataformas de virtualización en el mismo host o en diferentes hosts, que comparte los recursos que son virtualizados (6). Por lo tanto, este modelo genérico, ha sido conceptualizado como una extensión del esquema CIM. CIM fue elegido para el modelado de la red debido a la orientación a objetos, a las estructuras de red bien definidas y la independencia de la plataforma.

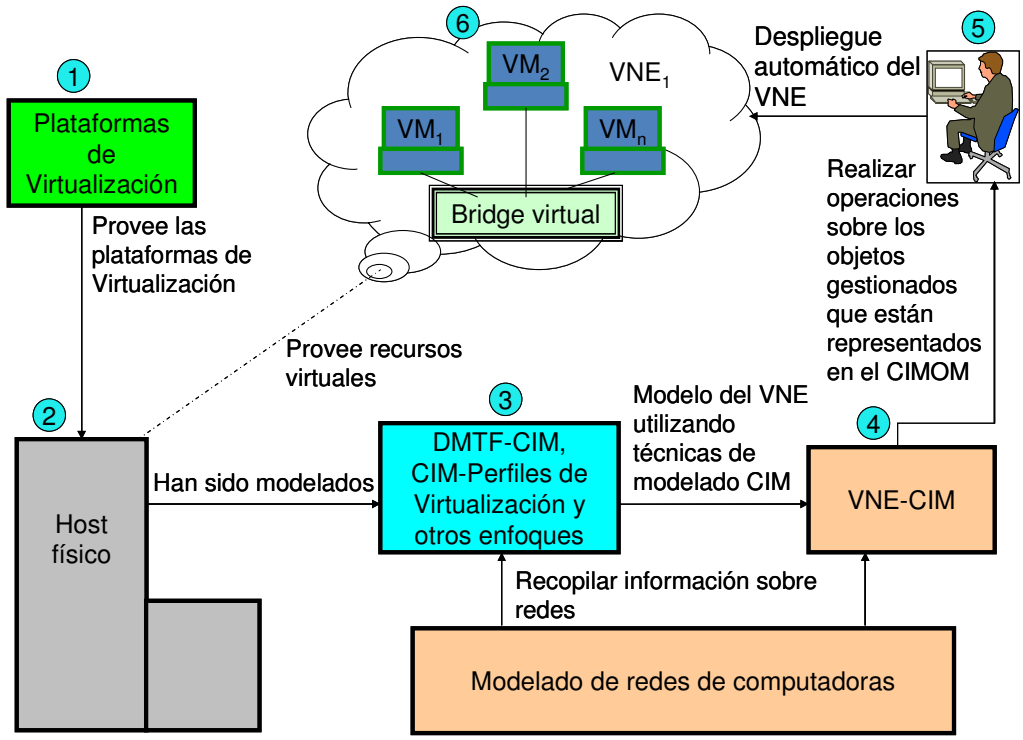

Figura 7-2 Modelo propuesto para el despliegue de un EVR

7.3.4 Diseño del Modelo genérico de un EVR basado en CIM

Teniendo en cuenta el marco conceptual descrito en el apartado 7.3.3, este apartado explica la metodología y los detalles de la implementación del cliente API-CIM y las acciones necesarias relacionadas con gestión del CIMOM. La Fig. 7-3 muestra el enfoque arquitectónico que se ha seguido.

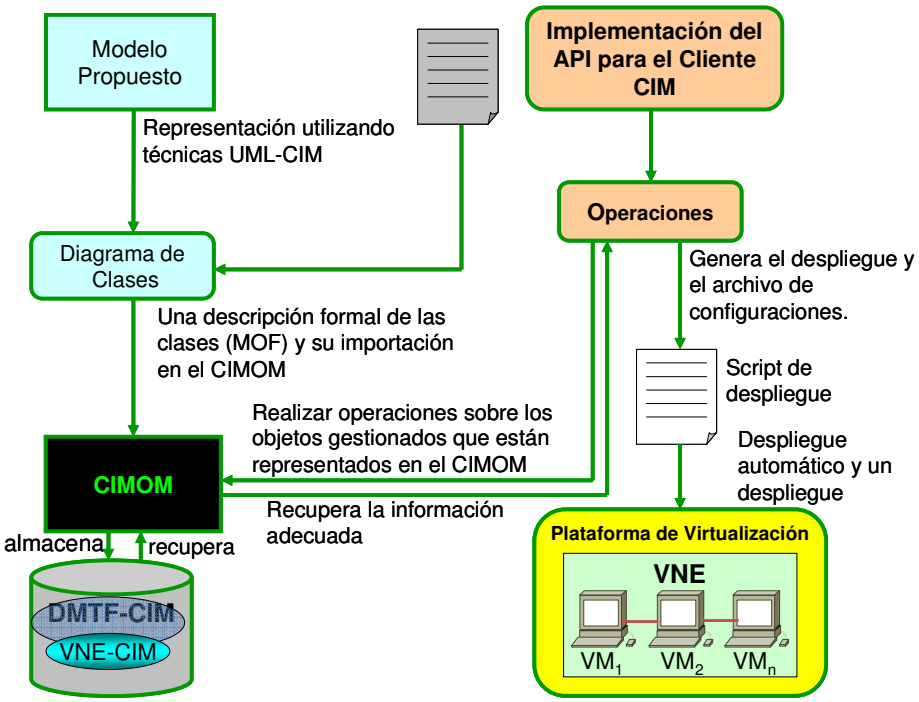

Figura 7-3 Enfoque arquitectónico para el diseño e implementación del modelo propuesto

Arrancando con el modelo propuesto, este modelo se ha representado usando técnicas UML-CIM que se traducen en un diagrama de clases UML (véase fig. 7-4). Según este marco de trabajo, se ha expresado en MOF una descripción formal de nuevas clases del esquema CIM (heredado de la CIM), luego estas fueron compiladas para hacer una validación sintáctica y luego fueron almacenadas en el repositorio CIMOM. Esto puede hacerse en el mismo paso, puesto que la mayoría de las implementaciones CIMOM incluyen también un compilador MOF para la lectura de los archivos MOF y la adición de las clases en el repositorio CIMOM.

La extensión del esquema CIM correspondiente se la ha denominado VNE-CIM la misma que consta de las siguientes clases (el prefijo VNE_ ha sido utilizado para identificar las clases agregadas):

- Clase *VNE_Configuration*, derivada de CIM_SettingData par modelar las propiedades relacionados con la plataforma de virtualización. Cada plataforma de virtualización nueva está modelada como una instancia de esta clase, utilizando las siguientes propiedades: nombre de la plataforma de virtualización, kernel, versión, tipo de consola, la ruta donde los archivos de

configuración debe ser generados, el método de ejecución, el tipo de sistema de archivos y sistema de archivos. Esta clase es accedida por el gestor de aplicaciones del cliente cuando se conecta al repositorio CIMOM, consultando la información que este contiene;

- Clase *VNE_Network*, derivada de CIM_Network: no introduce propiedades adicionales, sino que se utiliza para agregar una instancia de red por cada plataforma de virtualización;
- Clase *VNE_ComputerSystem*, derivada de CIM_ComputerSystem: no introduce propiedades extras, sino que se utiliza para agregar una instancia de MV para cada red y para cada plataforma de virtualización;
- Clase de asociación *VNE_SystemComponent*, derivada de CIM_SystemComponent: no introduce propiedades adicionales, sino que se utiliza para agregar una instancia de la relación entre la MV y la red a la que pertenece. Esta clase de asociación es accedida cuando el gestor de aplicación de cliente explora el repositorio CIMOM para obtener la lista de MVs y su información asociada desplegada en la plataforma de virtualización;
- Clase *VNE_CompleteAddr*, derivada de CIM_IPAssignmmentSettingData: Esta clase se utiliza para modelar propiedades tales como MAC e IP (combinado en completeAddr) y los dispositivos de la MV, usados para disco duro y BIOS (device1, dispositivo2, dispositivo3, etc);
- Clase *VNE_IPProtocolEndpoint*, derivada de CIM_IPProtocolEndpoint: no introduce propiedades adicionales, sino que se utiliza para agregar una instancia de la relación entre la MV y la NIC respectivo por cada nodo;
- Clase *VNE_IPStaticIPAssigmentSettingData*, derivada de CIM_IPStaticIPAssigmentSettingData. Esta clase se utiliza para modelar atributos tales como: InstanceID, IPv4address, y SubnetMask para asignar una dirección IP estática para cada MV;
- Clase *VNE_ElementSettingData*, derivada de CIM_ElementSettingData: no introduce propiedades adicionales, sino que se utiliza para agregar una instancia de la clase asociación entre la clase VNE_CompleteAddr;
- Clase de asociación *VNE_MemberOfLink*, derivada de CIM_MemberOFCollection. Es utilizada para modelar las asociaciones entre un enlace (Colección Reference) y la NIC de la MV que se conecta (Member references).

Para concluir, se podría argumentar por qué algunas clases que no introducen nuevas propiedades son creadas, en lugar de utilizar directamente a sus clases padre. Las razones para hacerlo son las siguientes: En primer lugar, para diferenciar instancias de estas nuevas clases de las clases padre. En segundo lugar, el trabajo futuro puede requerir la adición de nuevas propiedades. En tercer lugar, para perfeccionar la información desde el punto de vista semántico. Como observación final en concordancia con las relaciones de herencia, este modelo tiene sólo dos niveles jerárquicos: las clases de base CIM y las clases derivadas VNE. Con el propósito de ilustrar

la recuperación de los datos por el gestor de aplicaciones del cliente, sólo dos relaciones de agregación han sido introducidas en este diagrama: *VNE_SystemComponent* (para modelar la totalización de los componentes pertenecientes a un *VNE_ComputerSystem*) y *VNE_ElementSettingData* (para modelar la agregación de la configuración *VNE_StaticAssignmentSettingData*).

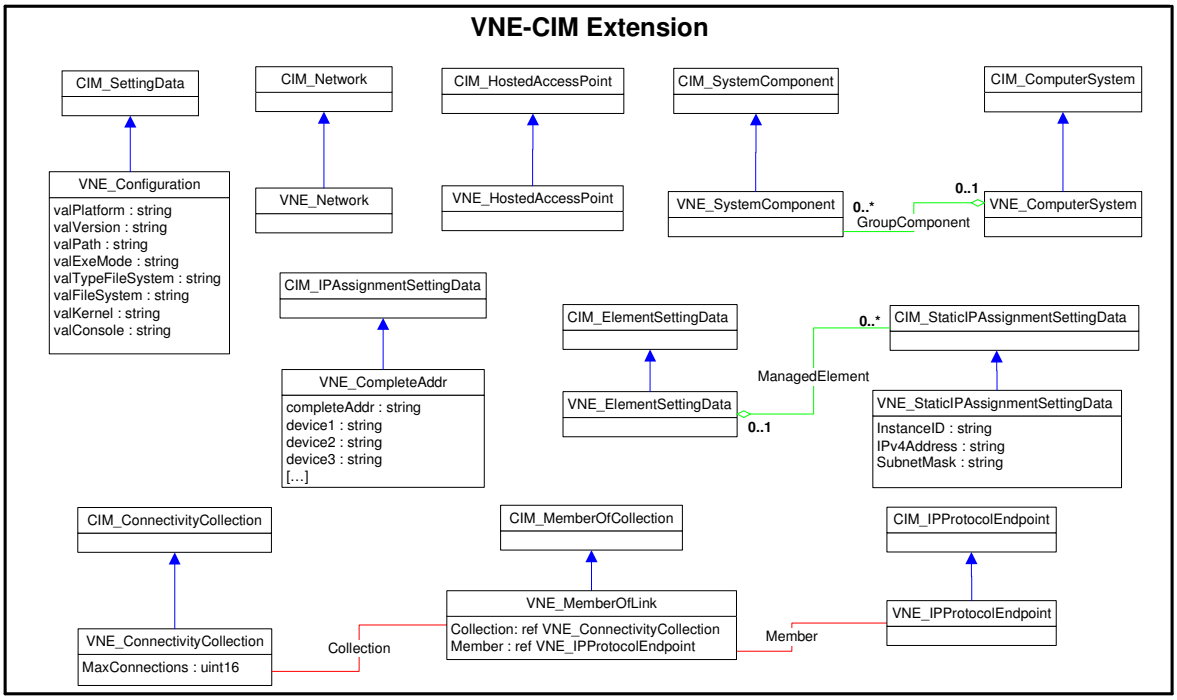

Figura 7-4 Diagrama de Clases del EVR

7.4 Implementación del API-CIM del cliente

Con el fin de probar nuestro modelo, se ha implementado una aplicación cliente CIM, que proporciona una interfaz para acceder y manejar clases y objetos, para realizar operaciones sobre los objetos gestionados que están almacenados en el CIMOM, a fin de generar automáticamente EVR y para desplegar EVR en diferentes plataformas de virtualización. Una vez que el EVR ha sido importado como un conjunto de instancias MOF de las clases CIM-EVR descritos en el apartado anterior, el administrador usando la implementación de la API ejecuta una operación de EVR (a partir de clases de Java) para acceder al CIMOM, generando los archivos de comandos (scripts) con el fin de producir la operación deseada (por ejemplo, desplegar el EVR). Así pues, la cuestión principal es recuperar las instancias que confirman la extensión del modelo (que se describe en el apartado 7-4) almacenados en el CIMOM. Para ello, se ha integrado una aplicación de código abierto WBEM llamado WBEM Services [WbemServices], para interactuar con el servidor de CIMOM. Los servicios de WBEM han sido elegidos para la implementación de la API de la CIM, porque es más comprensiva y estable, de acuerdo a la documentación.

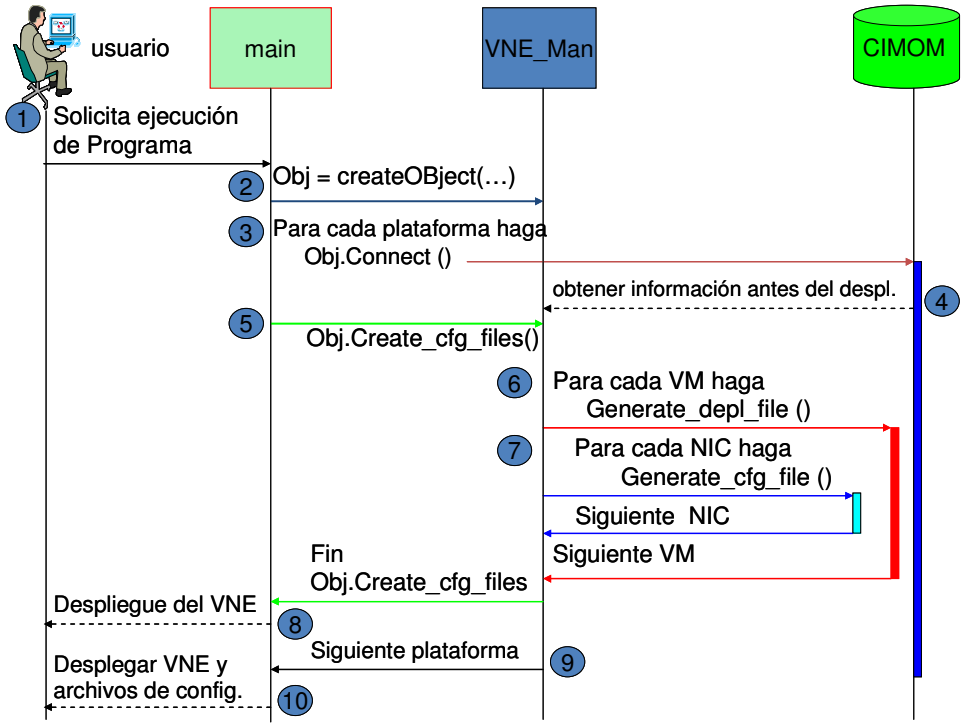

Figura 7-5 Diagrama de Secuencia en el Cliente

La Fig. 7-5 muestra el diagrama de secuencia del cliente, el cual representa la interacción entre el usuario y el programa principal, con el objetivo de probar el modelo. En primer lugar, el usuario solicita una operación de gestión (por ejemplo, el despliegue) (1). Este programa crea un objeto de clase de Java VNE_Man para manejar la conexión con el repositorio de CIMOM (2). Para cada plataforma de virtualización se verifica la relación con la CIMOM (3). Esta conexión nos permite obtener la información apropiada (4). Entonces, una función es llamada a generar un script de despliegue y los archivos de configuración (5). A su vez, el programa principal interactúa con la clase VNE_Man para crear secuencias de comandos (scripts) de despliegue y para arrancar cada máquina virtual en el EVR (6), y con los archivos de configuración para crear una o más tarjetas de red virtual hasta completar el número necesario de MVs (7). Es importante señalar que dentro de la clase VNE_Man, hay un proceso para leer el repositorio CIMOM, los ajustes de datos de las MVs y las direcciones extendidas correspondientes. Después, el programa principal despliega un EVR en una plataforma de virtualización específica subyacente (8). Entonces, el proceso puede repetirse para otras plataformas de virtualización (9), si es necesario. Al mismo tiempo, los scripts de implementación y archivos de configuración se guardan en el disco duro del usuario (10). Vale la pena mencionar que el programa principal es responsable de crear todos los scripts de implementación y archivos de configuración. También es responsable de realizar el despliegue automático de cada EVR. Esto ha sido probado para las plataformas Xen y VMware Server. Para

obtener más información, favor de acceder a los programas fuente y el modelo de objetos almacenados en el CIMOM [1], o en el **Apéndice C**.

7.5 Análisis y Evaluación de resultados

7.5.1 Topología de Prueba

Todas las pruebas fueron llevadas a cabo en un host físico, Intel Pentium ® Dual, 1,86 GHz, 4GB de RAM. En este host físico se han instalado Xen y VMware Server. El mismo sistema de archivos y el mismo kernel (2.6.27.5) se utiliza para todos las MVs. El lado del cliente automáticamente despliega un EVR formado por tres MVs interconectadas, tanto en Xen como en VMware. En esta topología de prueba, un único puente virtual fue utilizado para conectar todos los EVR a la interfaz de red física del hipervisor. En el caso de Xen, la creación de redes virtuales se ha configurado por defecto al utilizar la red-puente (network-bridge). En el caso de VMware Server, se configura como un puente de red (bridged-networking). En ambos casos, se trata de un tipo de conexión de red entre una MV y la red física externa del host. Bajo un puente de red (bridged-networking), una máquina virtual se comporta como una computadora adicional de en la misma red Ethernet física. Por lo tanto, este EVR es similar a los mostrados en la Fig. 7-1 y en consecuencia estos resultados validan la arquitectura de base descrito en el apartado 6.3.2.

7.5.2 Resultados de Prueba

La Tabla 7-1 muestra los resultados obtenidos al ejecutar la aplicación cliente API-CIM. El tiempo de ejecución de aplicación es el tiempo que esta toma en encontrar la información en el CIMOM y generar el script de implementación. El tiempo esperado de despliegue de un EVR es el tiempo que tarda en ejecutar la secuencia de comandos para desplegar un EVR. Este cuadro muestra que el tiempo de ejecución de aplicación es muy pequeño (alrededor del 3%) en comparación con el tiempo transcurrido de despliegue de un EVR. Es importante señalar que la descripción contenida en el EVR CIMOM es similar para ambos casos (es decir, para Xen y VMware Server).

Tabla 7-1. Resultados de Prueba

Plataforma de virtualización	Tiempo de ejecución de la aplicación	Tiempo de Espera para desplegar un EVR	Consumo de CPU	Consumo de Memoria
Xen	2 s	64.5 s	21.4%	122MB
VMware Server	2 s	63.0 s	51.2%	172MB

[1] http://arantxa.ii.uam.es/~jlopezv/noms2010

7.5.3 Discusión

Se ha utilizado el repositorio de clases MOF y mediante la generalización y la agregación de las relaciones se ha añadido nuevas clases en el repositorio VNE-CIM. Como consecuencia, el diseño es independiente de la plataforma de virtualización. Como resultado de ello, no es necesario modificar las clases del VNE-CIM, en el caso de cambiar la plataforma de virtualización. Para agregar una nueva plataforma de virtualización, la única operación que se necesita es añadir una instancia de la clase VNE_Configuration (descrito en el apartado 7.3.4).

Con respecto a las fortalezas de la aplicación cliente CIM-API, se ha implementado una única clase la cual puede generar scripts para todas las plataformas de virtualización utilizadas. Además, genera automáticamente los archivos de todas las plataformas de virtualización en el que el usuario quiere desplegar sus EVR. Cuando se introduce una nueva plataforma de virtualización, es necesario modificar en una proporción menor de la clase VNE_Man dado que cada una plataforma de virtualización tiene sus propios parámetros.

En cuanto a otras cuestiones descritas en la descripción del planteamiento del problema (apartado 7.2), la aplicación de cliente API-CIM parcialmente toma el control de los recursos disponibles, su distribución y la liberación de los recursos del EVR. Además, esta aplicación permite el despliegue de EVR en varios hosts físicos con cambios menores en el programa principal. Por último, vale la pena mencionar que nuestra aplicación se puede personalizar. Esta característica nos permite incluir otras plataformas de virtualización con pequeños ajustes, manteniendo la misma información sobre el modelo deL EVR en el CIMOM.

Por último, con respecto a la adición de más funcionalidad, este modelo es razonablemente compacto y coherente en relación a las propuestas descritas en el apartado 6.4.2, ya que utilizando muy pocas clases, no sólo soporta conectividad entre MVs, sino también múltiples plataformas de virtualización.

7.6 Trabajos relacionados

Existen algunos trabajos, que han considerado el modelado un EVR y su despliegue en su conjunto, como VNUML [Galán09], Net-Kit [Pizzonia08] o MLN [Begnum06]. Sin embargo, nuestro enfoque intenta superarlas en dos áreas. En primer lugar, estas herramientas están muy ligadas a las plataformas de virtualización específicas (por ejemplo, considera VNUML User Mode Linux exclusivamente), mientras que nuestro enfoque es genérico y requiere de una plataforma neutral. En segundo lugar, los trabajos mencionados se basan en modelos no estándar definidos por la herramienta de los ejecutores (normalmente basado en XML), pero no en las normas abiertas estandarizadas como el modelo CIM.

Basado en estándares y muy cercano a nuestro enfoque, el trabajo propuesto por Galán et al., en

[Galán08a], describe una metodología impulsadora de un modelo genérico para especificar escenarios de red (equivalente al concepto EVR) a ser desplegados en infraestructuras de pruebas (no necesariamente virtualizado). Se basa en una arquitectura de modelo de dos capas: una de alto nivel llamada modelo independiente del banco de pruebas (test-bed) (basado en CIM) y un procedimiento de transformación automática (basado en el Model Driven Architecture [OMG03]) para obtener modelos específicos de pruebas (TSMs) utilizado por las herramientas de gestión específicas de cada test-bed. El enfoque utilizado en nuestra investigación es muy centrado en entornos de virtualización. En realidad, el modelo descrito en el apartado 7.4 podría ser considerado un "pseudo-TIM" para EVR y los scripts de despliegue de diferentes TSMs.

En cuanto a modelos de VMs, en [Kunce08] [Wang08], los autores exploran cómo recuperar información de recursos de MVs Xen y VMware y luego lo integra servicios de información Grid. Estos estudios enfatizan la transformación de la información basadas en CIM en un esquema Grid-Glue. En comparación con nuestra investigación, ellos utilizan VMware CIM-SDK y Pegasus para implementar un servicio de información para recuperar información de VMs y su almacenamiento asociado, que puede ser utilizado en servicios de información Grid. La Computación Grid está fuera del alcance de este capítulo.

Una tercera investigación comparable ha sido descrita por Fahy et al., [Fahy08]. Allí, los investigadores detallan los requerimientos de especificación tal como un lenguaje de modelo de información para controlar la virtualización de los recursos de red y servicios. Sin embargo, han elegido el modelo DEN-ng en lugar del modelo CIM. DEN-ng proporciona múltiples puntos de vista de una red de comunicación donde los objetivos de negocio gobiernan todas las entidades gestionadas (se debe tener en cuenta que las especificaciones del modelo de red CIM no consideran la asociación de objetivos de negocios). Nuestro enfoque en su lugar, se basa en una extensión del esquema CIM y ha sido validada a través de la implementación de un sistema de gestión CIM.

7.7 Conclusiones

El principal problema en nuestra tesis doctoral fue diseñar un esquema de extensión de CIM para caracterizar un modelo genérico para EVR, incluido el despliegue del EVR en su conjunto. El resultado del modelado muestra que se ha logrado un modelo genérico que es independiente de la plataforma de virtualización. Para validar el modelo genérico, se ha implementado un sistema de gestión CIM-API que recupera los objetos desde el modelo del VNE-CIM almacenado en un repositorio CIMOM. Los resultados de las pruebas (evaluado con Xen y VMware Server), han demostrado la eficacia de esta aplicación. En consecuencia, la ventaja de este trabajo es que reduce la complejidad en la construcción y el despliegue de EVR utilizando diferentes plataformas de virtualización. Además, la aplicación del cliente API-CIM basado en este modelo, toma

parcialmente el control de los recursos disponibles y con cambios menores, permite el despliegue de EVR en diferentes equipos físicos, cada uno desplegado en una plataforma de virtualización diferente.

Por último, dado que las implementaciones principales tratados en esta investigación se basan en DMTF-CIM, consideramos que el uso de la CIM, como un mecanismo de extensión en nuestra investigación ha sido una acertada decisión. Los resultados de este trabajo han sido publicados en [Fuertes10a].

Capítulo 8 Conclusiones y Trabajos Futuros

8.1 Resumen

En este capítulo se describe un resumen de los resultados obtenidos durante la investigación presentada en capítulos anteriores. Aquí se incluyen las contribuciones de esta Tesis Doctoral en el ámbito de la aplicación de las tecnologías de virtualización en la experimentación y dimensionado de Redes IP, así como en otros campos relacionados con el modelado de información de infraestructuras de virtualización. Posteriormente, se indican las aportaciones y las publicaciones que se han derivado durante el desarrollo de la presente Tesis. Por último se definen las líneas de investigación futuras que se derivan del presente trabajo y que facilitarán la transferencia de conocimientos en este campo temático.

La motivación fundamental de esta Tesis ha sido la evaluación de varias plataformas de experimentación para realizar experimentos de EVR IP, mediante la aplicación de herramientas de virtualización, sobre las que se puedan realizar medidas de rendimiento, prestación de servicios de redes, consumo de recursos y de calidad de servicio.

Luego de haber analizado el Estado del Arte se determinó que existen algunas técnicas de virtualización que tienen como estrategia principal el compartir recursos de hardware, utilizando un único host anfitrión. Igualmente se estableció la existencia de varias herramientas de virtualización, que están siendo usadas principalmente para consolidación de servidores, y pruebas de validación de software, reduciendo los costos de inversión y experimentación.

Puesto que las hipótesis son el fundamento que dan coherencia a la Tesis, a continuación se valorará las conclusiones derivadas de la presente investigación, contrastando con cada hipótesis:

8.1.1 Conclusiones de Primera Hipótesis.

"Un entorno de red virtualizado puede ser utilizado para emular servicios reales de redes IP". Para emular un servicio de red real en un entorno virtual, fue necesario contar con una plataforma de experimentación. Para establecer esta plataforma, resultaba indispensable, realizar previamente una comparación cuantitativa de las plataformas de virtualización, en lo relacionado al rendimiento (consumo de recursos de CPU y memoria) (véase el apartado 2.3). Para ello se instalaron y configuraron seis plataformas de virtualización diferentes y se diseñó un escenario virtual para dicha evaluación. Es posible que este escenario no corresponda exactamente a un entorno de red

real, sin embargo, el despliegue de esta infraestructura permitió hacer diversos experimentos o pruebas asociados a las redes.

Como resultado de esta evaluación se determinó que Xen sería la mejor herramienta de virtualización para implementar tal EVR. Una mención importante se ha dado a VNUML y Netkit que proporcionan una herramienta unificada para crear y desplegar escenarios de red virtuales, reduciendo los costos de implementación.

Como complemento se intentó emular un servicio VoD real de ADSL en un EVR con Xen (véase el apartado 3.3). En esta emulación, se ha implementado un método para mejorar los resultados obtenidos en los EVR, en contraste con los obtenidos en entornos reales. Los resultados experimentales han mostrado cierta similitud en el tiempo de llegada de paquetes entre los dos entornos en el lado del servidor y del lado del cliente. Sin embargo, las distribuciones de probabilidad no son precisamente las mismas, ya que el EVR presenta una sobrecarga no cuantificada y porque el retardo ADSL emulado es una aproximación. En cualquier caso, los resultados del experimento han proporcionado datos cualitativos en relación a cómo funcionan los servicios, la calidad percibida del servicio de VoD y las configuraciones necesarias. Este trabajo además ha permitido verificar la funcionalidad de los entornos virtuales demostrándose que la Virtualización si permite realizar pruebas de validación de software y de emulación de servicios. Finalmente esta investigación ha permitido disponer de un bosquejo inicial de los factores que afectan a los resultados experimentales utilizando plataformas de virtualización.

8.1.2 Conclusiones de la Segunda Hipótesis.

Si las técnicas de Virtualización en un escenario virtual añaden una sobrecarga, entonces los resultados de la experimentación en redes carecen de precisión.

Las tecnologías de virtualización generan una sobrecarga en el rendimiento (*overhead*) causada por la capa de virtualización, lo que reduce la precisión de los resultados experimentales y por lo tanto limita la credibilidad en su aplicación. Esto fue contrastado en la primera hipótesis. Sin embargo como un mecanismo adicional de medición y contraste de esta hipótesis, se desarrollaron los experimentos siguientes:

El **primero**, mediante la comparación de los resultados obtenidos al evaluar varias topologías de prueba utilizando **plataformas de virtualización y métodos de simulación** (véase el apartado 3.6 y el Apéndice A). En esta investigación se diseñó, implementó y puso en funcionamiento 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 a los mismos escenarios y a las mismas pruebas. Luego se ajustaron aquellos parámetros que se relacionan con el rendimiento de red como ancho de banda, 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 como se produce la degeneración de la red a medida que incrementan los equipos. Finalmente, se ha demostrado que en el caso de la Simulación, los parámetros deben ser rigurosamente programados para mejorarlos. En el caso de la Virtualización, se deben adaptar otras condiciones operacionales como servidores dedicados, temporización, otras métricas de rendimiento, el mejoramiento del hardware base y el ajuste del software que se utilizó para inyectar tráfico en la red virtualizada.

El **segundo**, mediante la **medición del *overhead* producido por la capa de virtualización** (véase el apartado 4.1). Para cumplir este propósito, se ha diseñado e implementado tres topologías de prueba utilizando dos plataformas de virtualización típicas como son VMware y Xen. Luego, se han comparado sus resultados frente un entorno de red real. Se ha evaluado los experimentos con diferentes cargas de trabajo y diferentes anchos de banda. Se ha trabajado en la parte en que el rendimiento ha sido afectado. Se ha aplicado el modelo de regresión lineal para modelar la relación entre las variables. Se ha utilizado estadígrafos tales como el error residual, coeficiente de variación y la desviación estándar para obtener un modelo matemático de regresión lineal. Por lo tanto, este procedimiento ha permitido establecer una ley que podría derivar una expresión analítica que caracterice al *overhead*, el cual predice el comportamiento de la carga u *overhead* en un EVR. También conviene mencionar que las expresiones analíticas encontradas pueden aplicarse para predecir la penalización calculada en la evaluación de un servicio en un EVR Xen.

El **tercero**, mediante la creación de una propuesta que intenta determinar cuáles son los factores que afectan el rendimiento en un experimento realizado en un EVR (véase capítulo 5). Para este fin se ha realizado una topología de prueba en la que una a una se van incrementando las MVs, así como el número de CPUs en el interior de cada una de ellas, de tal forma que se iba calculando como su incorporación iba afectando el rendimiento del EVR. Como herramienta de validación se utilizo el análisis de la varianza denominado ANOVA. Esta es una metodología estadística que permite comprobar si son iguales las medias de más de dos poblaciones independientes mediante la comparación de varianzas insesgadas de muestras de diversas fuentes, utilizando para el efecto, la prueba de probabilidad F Fisher. Los resultados finales muestran que variables que intervienen en un EVR son más aceptadas y que variables son rechazadas, es decir cuales de las variables afectan o inciden en la penalización del rendimiento de un EVR.

8.1.3 Conclusiones de la Tercera Hipótesis.

"Si en un entorno virtualizado se distribuye en diferentes equipos y con diverso hardware virtualizado, entonces debería existir soportabilidad e interoperabilidad entre plataformas de virtualización."

En la formulación del problema de la presente Tesis Doctoral (véase apartado 1.2) se han definido tres problemas fundamentales: El primero, que existe cierta complejidad tanto en la construcción de la topología de red virtual, como en el despliegue de los entornos virtuales de red. Esto obedecía al amplio espectro de plataformas de virtualización en la industria. El segundo, se identificó que el despliegue y la liberación de los recursos virtuales en un EVR es un paso crítico, especialmente en entornos complejos que requieren una rápida re-configuración. El tercero, la mayoría de las herramientas de virtualización tienen un enfoque centralizado. Esto significa que el despliegue y gestión de cada EVR tienen que hacerse en un equipo físico, que resulta en una baja escalabilidad que impide la creación de grandes y complejos EVR.

Para dar respuesta a estos cuestionamientos y con el fin de contrastar la tercera hipótesis, el Capítulo 7 de esta Tesis Doctoral (véase apartados 7.1 al 7.5), ha sido enfocado en la representación de un EVR buscando un modelo genérico que lo caracterice y que sea independiente de la plataforma de virtualización. Por lo tanto, la primera cuestión fue estudiar los enfoques de modelado de información para infraestructuras virtualizadas, exploración que consta en el capítulo 2, apartado 2.4. Luego, la segunda cuestión fue resolver el diseño y la construcción de un modelo genérico para describir el mínimo EVR a desplegarse en cualquier plataforma de virtualización subyacente. Para habilitar la gestión de interoperabilidad de un EVR, este modelo fue creado como un modelo de extensión de la serie de especificaciones derivadas de modelos normalizados aceptados por la DMTF-CIM (véase apartado 2.4) y otros. Además, este modelo debía considerar la extensibilidad a futuro para evitar la obsolescencia, ya que pueden surgir las nuevas exigencias en el futuro.

El resultado del modelado (véase apartado 7.3) muestra que se ha logrado un modelo genérico que es independiente de la plataforma de virtualización.

Como última cuestión y como confirmación experimental, se ha validado el modelo genérico, implementando un sistema de gestión CIM (véase apartado 7.4) que recupera los objetos desde el modelo del VNE-CIM almacenado en un repositorio CIMOM. Los resultados de las pruebas (evaluado con Xen y VMware Server), han demostrado la eficacia de esta aplicación.

En consecuencia, entre las ventajas de esta investigación se observan que reduce la complejidad en la construcción y el despliegue de un EVR pudiendo utilizar diferentes plataformas de virtualización (esto resuelve los problemas 1 y 2). Además, la aplicación del cliente API basado en CIM, toma parcialmente el control de los recursos disponibles y con cambios menores, permite el

despliegue de EVR en diferentes equipos físicos, cada uno desplegado en una plataforma de virtualización diferente (esto resuelve el problema 3).

8.1.4 Conclusiones de la Cuarta Hipótesis.

"Si los entornos virtualizados operan en redes distribuidas, entonces se puede interactuar coordinando la distribución de recursos entre servicios dinámicos, con múltiples dominios colectivos o individuales."

Como se ha corroborado durante la investigación de esta Tesis Doctoral, la creación de varios escenarios virtuales desplegados en diversos dominios de administración incrementa su complejidad. Para contrastar esta hipótesis en el Capítulo 6 (véase apartados del 6.1 al 6.6) se ha investigado un método de procesamiento computacional *(Computación Distribuida)* y de una estrategia (*Computación Grid*) que proporcione una plataforma de ejecución segura en topologías distribuidas. Para llevarlo a cabo se ha seguido una metodología incremental, partiendo de un modelo basado en el despliegue dinámico de escenarios virtuales distribuidos con VNUML. Tras esto se ha realizado una implementación basada en el desarrollo de una interfaz de servicios Web sin estado y por último un modelo de interfaz de servicios Web con estado, integrando Grid y WSRF. Como caso de estudio se ha utilizado el Proyecto PASITO. PASITO es una plataforma de red física distribuida para la experimentación de servicios de telecomunicación creada sobre la red académica nacional RedIRIS de España (véase apartado 6.3).

Como conclusiones fundamentales se ha propuesto un modelo de virtualización distribuida, incorporando un nivel de abstracción de tecnologías de virtualización entre el Grid y WSRF. Como apoyo conceptual, se ha adoptado RM-ODP como modelo de referencia para establecer transparencia de localización, acceso y prestaciones. Se ha modelado el procedimiento y la infraestructura para el despliegue y distribución de escenarios virtuales de red mediante servicios Web y después mediante servicios Grid, utilizando VNUML como lenguaje de especificación de dichos escenarios. La discusión realizada (véase apartado 6.5) ha descrito las ventajas y desventajas de cada solución propuesta, que han sido orientadas para mejorar el control, uso eficiente de los recursos y la seguridad de acceso de redes virtuales ejecutadas en entornos distribuidos.

En este punto es preciso hacer notar que esta hipótesis aportó la infraestructura básica para desplegar automáticamente EVR independientemente de la plataforma, la cual posteriormente fue modelada en el capítulo 7. Por tanto, la propuesta para levantar escenarios en entornos distribuidos fue la definida en la Tercera hipótesis (apartado 7.3), mediante el diseño y validación de un modelo genérico independiente de la plataforma de virtualización.

8.2 Valoración y análisis del trabajo realizado (Contribuciones)

En la Tesis doctoral, se ha experimentado con algunas herramientas de virtualización de la industria, tales como VNUML, Netkit, Qemu, Imunes, KMV, VMware Server, VirtualBox y Xen. Se han utilizado diversas técnicas estadísticas tales como Divergencia de Kullback, diferencia Euclidia, Chi cuadrado, Regresión Lineal, Análisis de Varianza y varias medidas de tendencia central. Se han aplicado varias herramientas de software para inyección de tráfico como iperf, netperf [netperf], httperf. Se han manipulado varias herramientas de análisis de tráfico como Wireshark, Tcpdump, SAR. Se han empleado diversas herramientas de emulación de redes como netem, tc, TBF y HTB.

Se ha obtenido una plataforma de experimentación virtual, que permite levantar servicios de red, probar la continuidad, conmutación y enrutamiento, usando varias herramientas de virtualización, tanto de código abierto como VMware Server.

Como **primera contribución** se ha realizado una comparación cuantitativa de las herramientas de virtualización, en base al consumo de CPU y memoria, desarrollando algoritmos en shell script como un procedimiento inédito para tomar decisiones a la hora de seleccionar la herramienta más apropiada (véase apartado 2.3). En este punto se ha identificado la posibilidad de comparar con herramientas de benchmarking existentes en la Web [Spec], para realizar comparaciones estandarizadas del rendimiento de los recursos virtualizados entre plataformas de virtualización.

Se ha emulado y puesto en funcionamiento VoD bajo EVR, particularmente en Xen, notándose cierta divergencia no despreciable en el lado del servidor así como en el lado del Cliente. Por tanto, como **segunda contribución**, se ha implementado un método para mejorar los resultados obtenidos en los entornos virtuales de red, para contrastarlos con los resultados obtenidos en entornos reales.

Se ha diseñado y puesto en funcionamiento varias topologías arbitrarias tanto en Xen como en NS-2, intentando verificar cuál es el grado de similitud o divergencia entre estas dos tecnologías. Como **tercera contribución** 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 diversas condiciones.

Se ha medido el *overhead* producido en un escenario de red virtual, tanto en Xen como con VMware Server. Como **cuarta contribución**, el procedimiento usado ha permitido establecer una expresión analítica que caracterice al *overhead*, el cual predice el comportamiento de la carga u *overhead* en un EVR con Xen. En este mismo contexto como aporte adicional, se ha logrado determinar qué variables afectan el rendimiento y en qué medida, mediante la implementación de un método analítico para automatizar el cálculo de coeficientes de regresión lineal múltiple y la aplicación de ANOVA, validados posteriormente mediante el software estadístico SPSS.

Se ha conseguido caracterizar un modelo genérico que permita desplegar los escenarios virtuales de red. Como **quinta y más importante contribución** se ha logrado diseñar e implementar un modelo de extensión DMTF-CIM, el mismo que ha sido formalmente almacenado en un CIMOM y que puede ser accedido desde un sistema de gestión. Como resultados relevantes de esta investigación, se ha conseguido automatizar el despliegue de un escenario virtual de red, basado en un único modelo de clases orientado a objetos, independientemente de la plataforma de virtualización, sea en el mismo host o en diferentes equipos consiguiendo distribuir los escenarios y gestionar los recursos virtuales de red con mayor eficiencia.

Finalmente, en la búsqueda de distribuir escenarios virtuales de red se realizaron algunas propuestas basadas en EDIV, interfaz de Servicios Web e interfaz de Servicios Web y WSRF. Por tanto, como **sexta contribución** se ha modelado el procedimiento y la infraestructura para le despliegue y distribución de EVR mediante servicios Web y después mediante servicios Grid, utilizando VNUML como lenguaje de especificación de dichos escenarios.

8.3 Publicaciones en revistas y congresos

A continuación se enumeran los artículos y ponencias que se derivaron por el desarrollo de la presente Tesis Doctoral. Se incluyen revistas y congresos nacionales e internacionales así como el ISBN o ISSN si procede. Además, se identifica qué apartados o secciones de la Tesis fueron documentadas mediante estas publicaciones:

AUTORES:	W. Fuertes, Jorge E. López de Vergara.
TITULO:	"**A quantitative comparison of virtual network environments based on performance measurements**"
CONGRESO:	14th HP Software University Association Workshop
PUBLICACIÓN	Proceedings: ISBN-13: 978-3-00-021690-9
LUGAR Y FECHA	Munich, Alemania, 8-11 Julio de 2007.
APARTADOS	Capítulo 2, apartado 2.3

AUTORES:	W. Fuertes, J. E. López de Vergara, F. Galán, D. Fernández.
TITULO:	"**Propuesta para el Despliegue de Escenarios de Red Virtuales en Entornos Distribuidos**"
CONGRESO:	VII Jornadas de Ingeniería Telemática, Jitel'2008
PUBLICACIÓN	Actas: ISBN: 978-84-612-5474-3, págs. 126-133.
LUGAR Y FECHA	Alcalá de Henares, Madrid, 16-18 septiembre de 2008.
APARTADOS	Capítulo 6, apartados del 6.2 al 6.5

AUTORES:	W. Fuertes, J. E. López de Vergara
TITULO:	"**Evaluación de Plataformas de virtualización para experimentación de servicios multimedia en redes IP**"
REVISTA:	Ciencia y Tecnología de la Escuela Politécnica del Ejército.
PUBLICACIÓN	Volumen 1, pp: 35-46, ISSN: 1390-4612.
LUGAR Y FECHA	Sangolquí, Ecuador, el 11-Sep-2008.
APARTADOS	Capítulo 3, apartado 3.2.

AUTORES:	F. Galán, D. Fernández, W. Fuertes, M. Gómez, J.E. López de Vergara
TITULO:	"**Scenario-based Virtual Network Infrastructure Management in Research and Educational Testbeds with VNUML: Application Cases and Current Challenges**"
REVISTA:	Annals of Telecommunications, Special issue on Virtualization.
PUBLICACIÓN	Volume 64, Numbers 5-6, pp: 305-323, ISSN: 0003-4347.
LUGAR Y FECHA	Springer Paris, Junio 11 de 2009.
APARTADOS	Capítulo 6, apartado 6.2 al 6.5

AUTORES:	W. Fuertes and J. E. López de Vergara
TITULO:	"**An emulation of VoD services using virtual network environments**"
CONGRESO:	GI/ITG Workshop on Overlay and Network Virtualization NVWS'09.
REVISTA:	Electronic Communications of the European Association of Software Science and Technology EASST.
PUBLICACIÓN:	Volume 17, ISSN 1863-2122
LUGAR Y FECHA	Kassel-Alemania, Marzo 23-28 de 2009.
APARTADOS:	Capítulo 3, apartado 3.2 al 3.5

AUTORES:	W. Fuertes, L. Jácome, M. Grijalva, J. E. López de Vergara, R. Fonseca
TITULO:	"**Evaluación del Rendimiento de Redes IP utilizando Plataformas de Virtualización y Métodos de Simulación**"
REVISTA:	DECC Report-Revista técnica del Departamento de Ciencias de la Computación de la Escuela Politécnica del Ejército.
PUBLICACIÓN:	Volumen 1, No. 1, pp. 34-43. ISSN 1390-5236
LUGAR Y FECHA	Sangolquí, Ecuador, Diciembre 8, de 2009.
APARTADOS:	Capítulo 3, apartado 3.6 y Apéndice A.

AUTORES:	W. Fuertes, J. E. López de Vergara and Fausto Meneses
TITULO:	"**Educational Platform using Virtualization Technologies: Teaching-Learning Applications and Research Uses Cases**"
CONGRESO:	II ACE Seminar: Knowledge Construction in Online Collaborative Communities.
PUBLICACIÓN:	Proccedings, ISBN 978-0-9842912-0-5
LUGAR Y FECHA	Albuquerque, Nuevo México – USA, Octubre 20-25 de 2009.
APARTADOS:	Capítulos 2, apartados 2.3 y 3.3. Además capítulos 3 al 7.

AUTORES:	W. Fuertes, J. E. López de Vergara, F. Meneses, F. Galán
TITULO:	"**A Generic Model for the Management of Virtual Network Environments**"
CONGRESO:	12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010).
PUBLICACIÓN:	Proccedings
LUGAR Y FECHA	Osaka, Japón, 19-23 Abril de 2010.
APARTADOS:	Capítulo 7, apartados del 7.2 al 7.5.

AUTORES:	W. Fuertes, J. E. López de Vergara, J. Pincha, H. Aules, L. Jacome.
TITULO:	"**Analytical Expression to Predict the Overhead Produced by the VMware and Xen Virtualization Tools**"
CONGRESO:	5th Science and Technology Congress (ESPE-2010)
PUBLICACIÓN	Accepted for its publication in Proceedings: ISSN 1390-4663
LUGAR Y FECHA	Sangolquí Ecuador, Junio 16-18 de 2010
APARTADOS:	Capítulo 4, apartado 4.1- 4.5

8.4 Líneas futuras de investigación y trabajo futuro

Puesto que la investigación científica finaliza cuando han sido difundidos sus resultados y ha existido transferencia de tecnología en el área específica [Fuertes09b], a continuación se revelan las líneas futuras de investigación. Previamente es importante afirmar que se está tratando de proveerle continuidad en la temática del trabajo de investigación de la Tesis con el fin de consolidar madurez científica y temática. Con estos antecedentes se planea realizar el siguiente trabajo futuro:

Con el fin de obtener una topología de experimentación de mayor envergadura, se explorará cómo crear un entorno de colaboración basado en tecnologías de virtualización entre las universidades hermanas de Ecuador, España, USA, Brasil y otros países que deseen generar redes

de investigación universitaria en esta temática.

Se explorará como acceder remotamente a la gestión de un CIMOM y como proveer mayores niveles de seguridad en el despliegue de un EVR en un entorno distribuido. Concretamente se buscará la forma de integrar SWRF y CIM.

En relación a la medición del *overhead*, se planea desarrollar un marco de trabajo (framework) que permita predecir el *overhead* en un EVR arbitrario, independiente de la plataforma de virtualización en la que es desplegado.

En este mismo ámbito se investigará otras técnicas estadísticas, para a partir de los datos medidos ser capaz de obtener las curvas de funcionamiento de un EVR.

Finalmente como línea futura de investigación se considera la definición de un modelo matemático para calcular la complejidad de un EVR, contemplando todas las variables implicadas, de tal forma que se pueda extraer cuánta pérdida de rendimiento o error va a incurrir, el tiempo de despliegue, el retardo obtenido, etc.

En relación a trabajos futuros de otras áreas relacionadas, se planea construir una arquitectura basada en Plataformas de Virtualización para acceder remotamente a cuentas de usuario por un canal seguro, con el fin de facilitar el acceso a máquinas virtuales alojadas en el servidor ubicado en un recinto universitario. Así mismo, crear una plataforma informática de experimentación, en ambiente Linux, que permita emular ataques a redes IP mediante la aplicación de tecnologías de Virtualización con el fin de implementar mecanismos de seguridad para poder contrarrestarlos. Por último, crear laboratorios multiplataforma, mediante la aplicación de tecnologías de virtualización, con el fin de facilitar la administración de la demanda de prestación de servicios y equipos en los laboratorios generales de computación universitarios.

Apendice A Evaluación de Redes IP utilizando plataformas de virtualización y métodos de simulación

A.1 Herramientas

A.1.1 Método de Simulación NS-2

Network Simulator es un simulador de código abierto diseñado específicamente para la investigación de redes de computadoras [Issariyakul09], desarrollada por el grupo de trabajo VINT (*Virtual InterNetwork Testbed*) fundado por la DARPA (*Defense Advanced Research Projects Agency*). 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). NS-2, es un simulador de redes 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. Aunque NS-2 se encuentra implementada en C++ con el objetivo de reducir los tiempos de procesamiento. Utiliza un intérprete OTcl con soporte de programación orientada a objetos. Permite la aplicación de protocolos de red como TCP y UDP y permite simular el comportamiento del tráfico de varios servicios de redes como FTP, Telnet, Web, CBR y VBR. Utiliza mecanismos de gestión de colas, y algoritmos de enrutamiento.

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. Por las razones expuestas, en la presente investigación ha sido la herramienta de simulación escogida.

A.1.2 Iperf

Iperf [Iperf], es una herramienta de código abierto, diseñada para medir el BW de una red. Soporta TCP, UDP entre dos equipos, Servidor y cliente, permitiendo la configuración de varios parámetros de medición tales como: retardo, paquetes perdidos, paquetes generados, etc. Iperf funciona para IPV4 o IPV6, tanto en distribuciones de Linux, cuanto en Microsoft Windows y Mac. Soporta además multicast y conexiones múltiples simultáneas. Iperf reporta medida del BW, throughput, retardo, variación del retardo y pérdida de paquetes o datagramas. Se utiliza a través de la línea de comandos, es del estilo similar a *Mgen*. Iperf dispone de opciones importantes, que combinadamente pueden caracterizar el rendimiento de la red.

A.1.3 Constant Bit Rate (CBR)

Tasa de bits constante, es un algoritmo útil en las mediciones de la calidad de servicio (QoS). En esta investigación ha sido utilizada como mecanismo de inyección de tráfico, el cual envía datos con una tasa de transferencia constante. Por defecto en la herramienta *Iperf,* CBR trabaja con el protocolo UDP, por lo que no hace falta especificarlo ni ajustarlo. En cambio en la programación del escenario de red con NS-2, se requiere implementarlo, y añadirlo a un agente UDP. Para ello se precisó enviar los parámetros necesarios para su configuración como: tasa (rate), tamaño del paquete (packetsize), ruido (random, permite randómicamente introducir ruido en la señal). Se escogió dicha fuente de tráfico pues de acuerdo con [Issariyakul09], el desempeño de CBR como inyector de tráfico de NS-2 genera resultados consistentes haciéndolo el adecuado para esta investigación.

A.2 Configuración del experimento

A.2.1 Diseño y configuración del escenario

Con el objetivo de poder comparar el rendimiento de la red tanto en un entorno Simulado como en uno Virtualizado se diseñó una topología de prueba (véase Fig. 3-6) tratando de configurar parámetros similares durante la medición tales como: enlaces bi-direccionales (half-duplex); BW variable; retardo constante; tamaño de los paquetes constante en un principio y variable posteriormente; CBR como algoritmo de generación de tráfico y, Drop Tail cómo tipo de gestor de colas. Este último para descartar data gramas entrantes nuevos en el caso de superar la máxima capacidad de la cola. La Fig. A-1 ilustra el escenario LAN propuesto en una topología física en estrella y la Tabla A-1 resume los parámetros de configuración descritos:

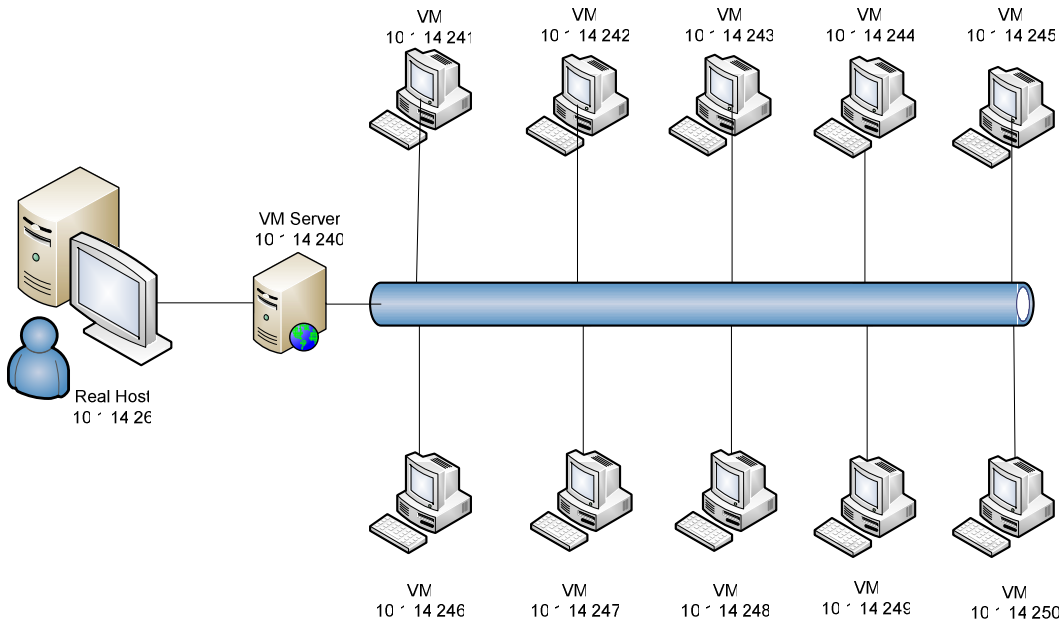

Figura A-1 Diseño de la topología de prueba.

Tabla A-1 Parámetros de Configuración en Xen y NS-2

Parámetros		*Herramienta*	*XEN*		*Herramienta*	*NS-2*
Retardo	*Virtualización*	Netem	delay 0.3 ms	*Simulación*	Script TCL	delay 0.3 ms
Protocolo		Iperf	- u		Script TCL	Agent UDP
Agente de Tráfico		Iperf	UDP/CBR		Script TCL	CBR
Rate		Iperf	-b 10 Mb		Script TCL	10 Mb
Intervalo		Iperf	-i 0.5		Script TCL	0.5
Time		Iperf	-t 60		Script TCL	60

A.2.2 Implementación

El diseño de la topología de prueba fue traducido a un entorno implementado tanto en Xen (véase Fig. A-2), como en NS-2 (véase Fig. A-3). En el primer caso se puso en funcionamiento la red que interactuaba desde cada máquina virtual (estaciones) hacia el servidor virtual. En el segundo caso desde cada nodo cliente hacia su servidor en la simulación. En ambos entornos se midió y analizó el comportamiento de dicho escenario, dado un retardo y el BW, mientras se realizaba la transferencia de paquetes hacia el servidor.

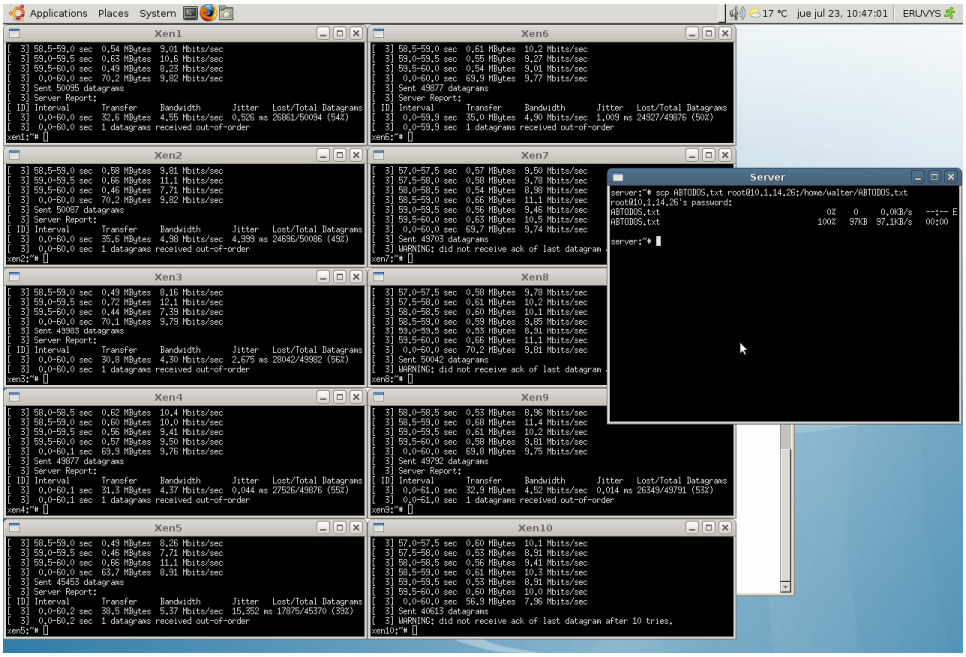

Figura A-2 Escenario de prueba virtualizado implementado.

Todas las pruebas se realizaron sobre Ubuntu Server 8.10. Kernel 2.6.27.5. En el caso de la Virtualización con la plataforma Xen, inicialmente se crearon y configuraron las MVs, cada una con su respectiva dirección IP. Para agilizar el proceso se creó un shell script que permitió levantar la topología de una forma automática. Con ello se procedió a establecer los respectivos parámetros para cada una: el retardo se añadió utilizando *Netem* [Hemminger06], que en un inicio fue variable hasta encontrar el valor adecuado para incluirlo en la simulación. Para delimitar el BW se uso Iperf, en donde se estableció el protocolo UDP, en combinación con el agente generador de tráfico CBR, en razón de ser ideal para soportar aplicaciones en tiempo real con pequeñas variaciones de retardo y útil para los flujos de datos en canales de capacidad limitada.

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 *sink* 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.

Se realizaron 10 experimentos de cada escenario en virtualización mediante Xen, incluyeron la creación de 3, 5, 10 y hasta 15 máquinas y un servidor, de la misma forma se implementaron la cantidad en nodos para la simulación mediante NS-2. Tanto en la virtualización como para la simulación se ejecutaron experimentos en un período de tiempo de 60 segundos.

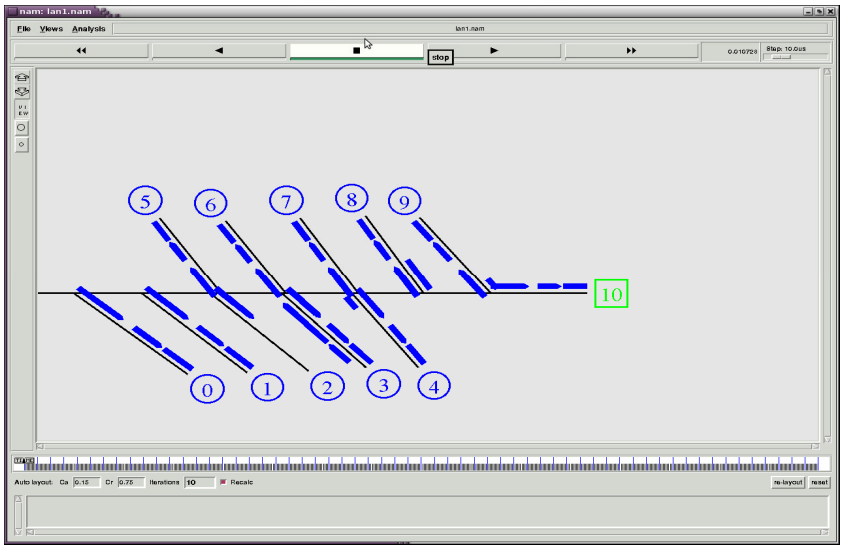

Figura A-3 Escenario de prueba simulado con NS-2.

A.3 Evaluación de resultados y discusión

A.3.1 Validación de resultados en base al hardware base disponible

Para verificar la influencia del hardware base en el rendimiento de la red, los primeros experimentos fueron realizados en un PC con menores características al PC definitivo. Las características de los equipos mencionados se detallan en la Tabla A-2:

Tabla A-2 Comparación de características técnicas del equipo anfitrión

Ítem	Descripción	Equipo Antiguo (M1)	Equipo Nuevo (M2)
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

Debido a los recursos disponibles en el equipo inicial (M1), se desplegaron dos escenarios menores con 3 y 5 estaciones (MVs y nodos), de la misma forma en el equipo definitivo (M2). Las Figs. A-4 y A-5, 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 M1 (incrementa el *overhead*) y por lo tanto disminuye el rendimiento. En el caso de M2 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.

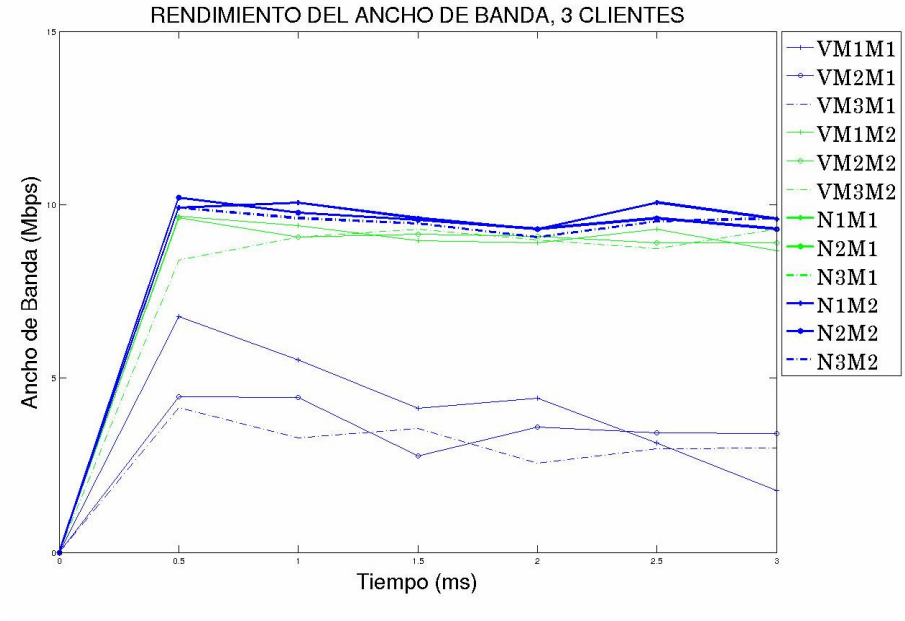

Figura A-4 Comparación de resultados en base al hardware disponible con 3 estaciones.

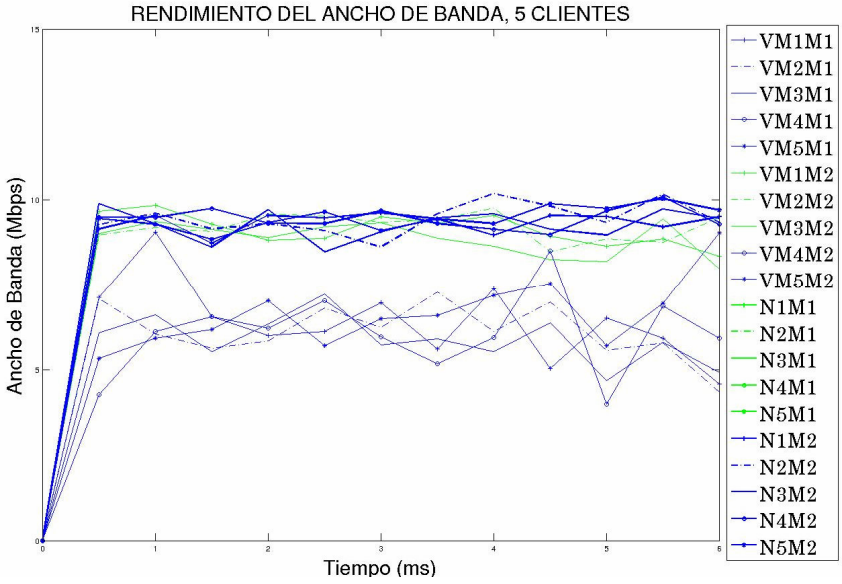

Figura A-5 Comparación de resultados en base al hardware disponible con 5 estaciones.

A.3.2 Validación de agentes generadores de tráfico

Con el fin de determinar el mejor agente generador de tráfico en NS-2, se realizaron varias pruebas ajustando el tamaño del paquete IP. La Fig. A-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.

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 [Issariyakul09], *Iperf* es una herramienta de inyección de tráfico CBR.

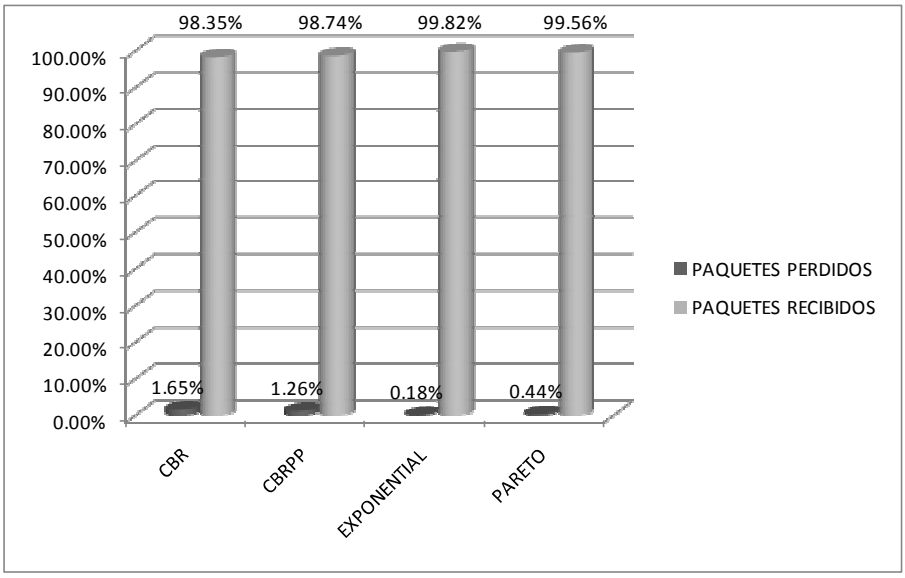

Figura A-6 Pérdida de paquetes IP por agente generador de tráfico en simulación

A.3.3 Comparación del rendimiento en base al BW.

La Figura A-7, ilustra la media del rendimiento del BW obtenido en un escenario de 3 estaciones y un servidor durante 10 mediciones. Cómo se puede apreciar, la este indicador 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 es una medida de tendencia central que puede producir sesgos en las mediciones.

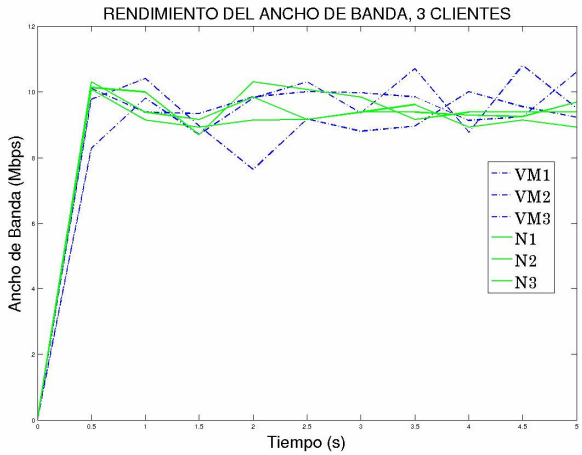

Figura A-7. Rendimiento del BW con 3 estaciones, entorno Virtualizado vs. Simulado.

A.3.4 Comparación entre escenarios de red en base a la pérdida de paquetes

Para asegurar la confiabilidad de los resultados en virtualización mediante 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 el apartado 3.3.2 los resultados se mantienen constantes. Las Figs. A-8 y A-9, 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.

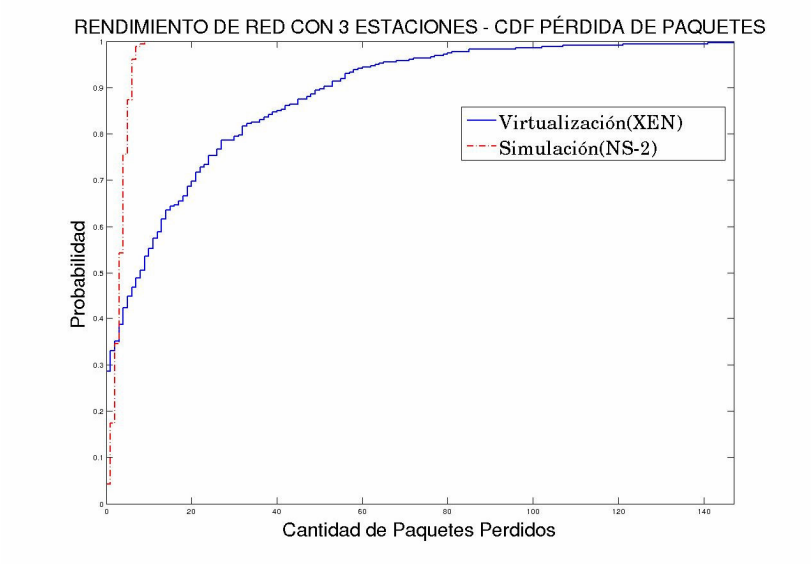

Figura A-8 Función de distribución de probabilidad acumulada de paquetes perdidos con 3 estaciones

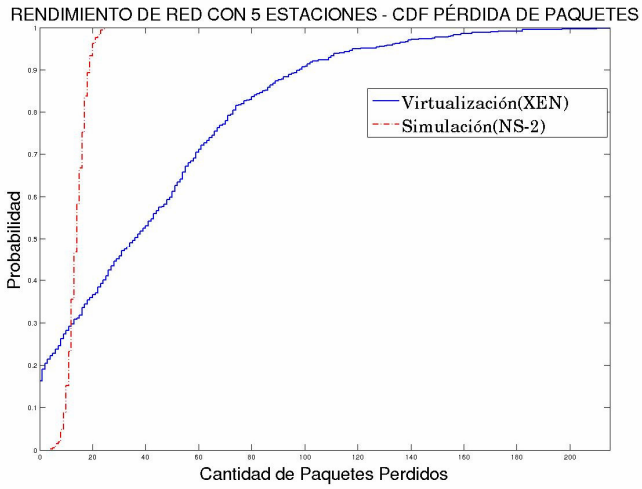

Figura A-9 Función de distribución de probabilidad acumulada de paquetes perdidos con 5 estaciones

Las Figuras A-10 y A-11, 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.

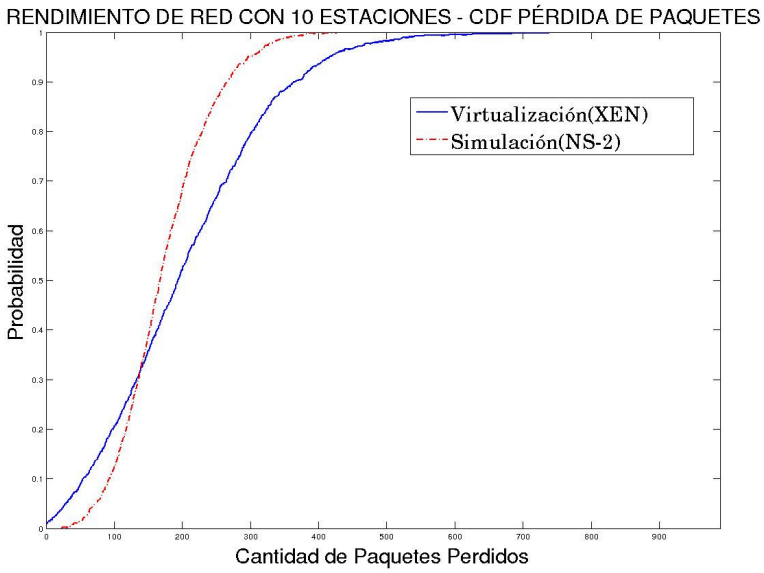

Figura A-10 Función de distribución de probabilidad acumulada de paquetes perdidos con 10 estaciones

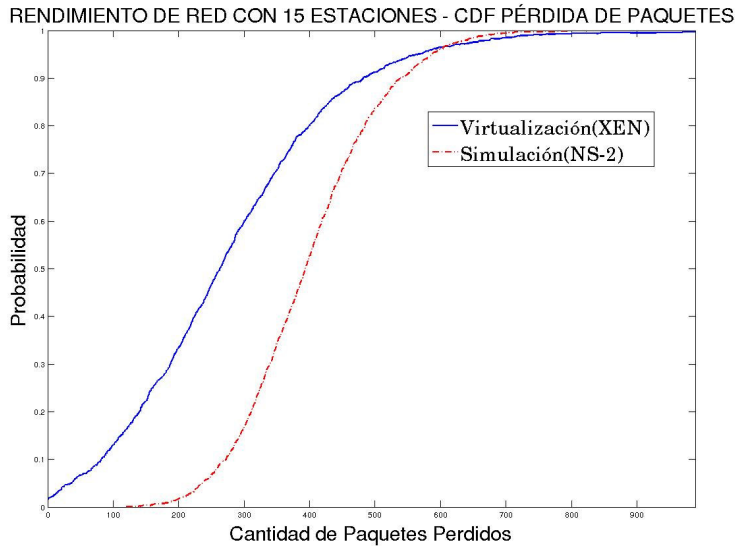

Figura A-11 Función de distribución de probabilidad acumulada de paquetes perdidos con 15 estaciones

La Fig. 3-12 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.

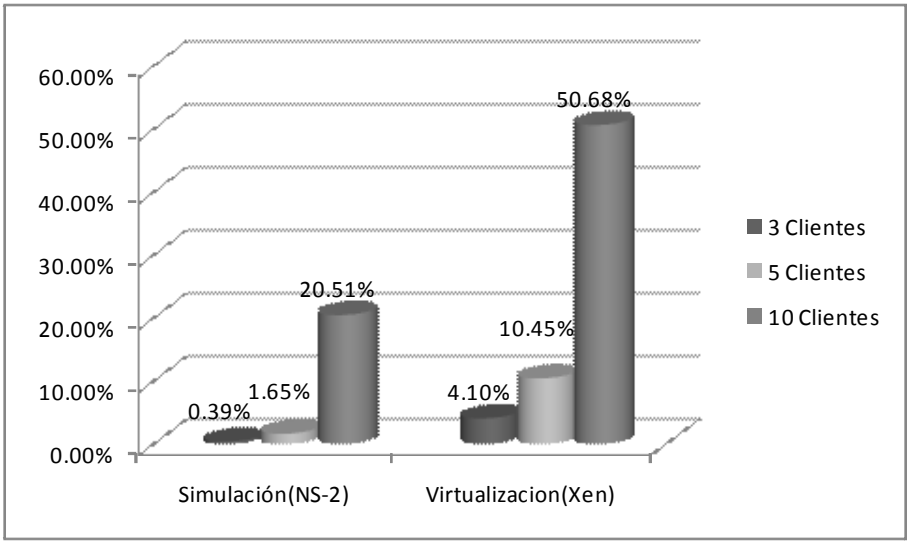

Figura A-12 Porcentaje de paquetes perdidos sobre el total de paquetes generados.

Apendice B Análisis y consecuencias del ANOVA.

B.1 El análisis

La prueba de la hipótesis H_0 contra H_1 se basa en dos estimaciones independientes de la varianza poblacional común σ^2. Estas estimaciones se obtienen particionando la suma de cuadrados total (SCT): $\sum\sum(x_{ij} - x_{..})^2$ en dos componentes.

En efecto, de la relación: $x_{ij} - \bar{x}_{..} = x_{ij} - \bar{x}_{i.} + \bar{x}_{i.} - \bar{x}_{..}$ se obtiene la siguiente identidad de suma de cuadrados:

$$\sum_{i\to1}^{k}\sum_{j\to1}^{n_i}(x_{ij} - \bar{x}_{..})^2 = \sum_{i\to1}^{k}\sum_{j\to1}^{n_i}(x_{ij} - \bar{x}_{i.})^2 + \sum_{i\to1}^{k}\sum_{j\to1}^{n_i}(\bar{x}_{i.} - \bar{x}_{..})^2$$

Es conveniente simbolizar esta partición de suma de cuadrados por:

$$SCT = SCE + SCC \qquad (3)$$

Donde,

- SCE es la suma de cuadrados del error (o dentro de los tratamientos).
- SCC es la suma de cuadrados de las columnas (o entre los tratamientos).

Así mismo, se verifican las siguientes esperanzas de sumas de cuadrados.

$$E(SCE) = (n-k)\sigma^2$$

$$E(SCC) = (k-1)\sigma^2 + \sum_{i\to1}^{k} n_i\alpha_i^2$$

$$E(SCT) = (n-1)\sigma^2 + \sum_{i\to1}^{k} n_i\alpha_i^2$$

B.2 Consecuencias:

De la primera esperanza, resulta que $\frac{SCE}{n-k}$ es una estimación insesgada de la varianza σ^2, independientemente de que la hipótesis nula sea verdadera o falsa.

Si la hipótesis nula, $H_0 : \alpha_i = 0, \quad (i = 1,2,...,k),$ se supone verdadera, entonces, de la segunda y tercera esperanza resultan respectivamente que:

$\frac{SCC}{k-1}$ Es una estimación insesgada de σ^2, y

$\frac{SCT}{n-1}$ Es una estimación insesgada de σ^2.

Las tres estimaciones insesgadas de la varianza común σ^2, se denominan cuadrados medios y son denotados respectivamente para el error, para las columnas y para el total, por: CME, CMC, y CMT.

"Es de esperar, entonces que el cociente $\frac{CMC}{CME}$ sea cercano a uno si la hipótesis nula es verdadera. Pero si la hipótesis nula no es verdadera, CME no cambia, mientras, que CMC será mayor. Esto implica que el cociente será mayor que la unidad. Si se invierte este razonamiento, se concluye que si $\frac{CMC}{CME}$ es significativamente grande se puede concluir que las medias de las poblaciones son distintas".

Además:

La variable aleatoria, $\frac{SCC}{\sigma^2}$ se distribuye como chi-cuadrado con $k-1$ grados de libertad $(x^2_{k-1}).$

La variable aleatoria, $\frac{SCE}{\sigma^2}$ se distribuye como chi-cuadrado con $n-k$ grados de libertad $(x^2_{n-k}).$

En consecuencia, si la hipótesis nula es verdadera, el cociente:

$$F = \frac{\frac{SCC}{\left[\sigma^2(K-1)\right]}}{\frac{SCE}{\left[\sigma^2(N-K)\right]}} = \frac{CMC}{CME}$$ Se distribuye según $F(k-1, n-k).$

Esto es, la variable aleatoria F tiene distribución F con $k-1$ y $n-k$ grados de libertad.

Además dado el nivel de significación α, para los grados de libertad $k-1$ y $n-k$, en la tabla F se encuentra el valor crítico $c = F_{1-\alpha, k-1, n-k.}$

La región crítica o de rechazo de H_0 de la prueba es el intervalo $]c, +\infty[$ a partir de los datos observados se calcula: $F_{cal} = \frac{CMC}{CME}.$

La regla de decisión es: Rechazar la hipótesis nula H_0 si $F_{cal} > c.$ En caso contrario no rechazar $H_0.$

En el modo P, si $P = P[F > F_{cal}]$, se rechazará la hipótesis nula H_0 si $P > \alpha$. En caso contrario no se debe rechazar H_0.

Las sumas de cuadrados del total de las columnas y del error se calculan utilizando las siguientes equivalencias:

$$SCT = \sum_{i \to 1}^{k} \sum_{j \to 1}^{n_i} (x_{ij} - \overline{x}_{..})^2 = \sum_{i \to 1}^{k} \sum_{j \to 1}^{n_i} x_{ij}^2 - C \text{ , donde } C = \frac{T_{..}^2}{n}$$

$$Scc = \sum_{i \to 1}^{k} \sum_{j \to 1}^{n_i} (\overline{x}_{i.} - \overline{x}_{..})^2 = \sum_{i \to 1}^{k} n_i (\overline{x}_{i.} - \overline{x}_{..})^2 = \sum_{i \to 1}^{k} \frac{T_{i.}^2}{n_i} - C$$

$$SCE = SCT - SCC$$

B.3 Algoritmo para automatizar el método analítico.

Este programa fue desarrollado para automatizar el método analítico que permita calcular los coeficientes de Regresión lineal múltiple y el test ANOVA utilizando MatLab. Sus resultados fueron posteriormente comprobados con los resultados obtenidos al procesar los datos con el software estadístico SPSS

```
clc
x=[1 16.36 16.59 18.49 22.47; 1 15.76 16.83 17.83 21.05;1 15.79 15.60 18.01 22.03;1 16.23 18.20
18.44 20.75;1 16.14 18.25 18.52 20.54;1 16.54 17.84 18.69 20.30;1 16.57 18.50 19.14 20.04;1
16.39 17.99 19.08 20.71;1 16.91 18.60 19.41 20.90]
y=[1; 2; 3; 4; 5; 6; 7; 8; 9;]
n=9
k=4
c=x'
d=c*x
e=inv(d)
b=e*(c*y)
SCE=(y'*y)-(b'*(x'*y))
s=((1+2+3+4+5+6+7+8+9)^2)/9
SCR=(b'*x'*y)-s
SCyy=SCE+SCR
k1=k
k2=n-k1-1
k3=n-1
MCE=SCE/k2
MCR=SCR/k1
F=MCR/MCE
Var=MCE
pvalue=(1-fcdf(F,k1,k2))
save AnovaCPU
```

Apendice C Modelo de extensión VNE-CIM

Este apéndice contiene el código correspondiente al modelo de extensión MOF, VNE-CIM con sus respectivas clases e instancias añadidas.

```
// main.mof, v1
//
// Copyright (C) 2009/JUL Walter Fuertes - Fausto Meneses.
// Main classes
[Version("1.0"),
 Description(
   "VNE_Network models the scenario itself, as a specialization of CIM_Network." )]
class VNE_Network : CIM_Network {
};

[Version("1.0"),
 Description(
   "VNE_ComputerSystem , as a specialization of CIM_ComputerSystem." )]
class VNE_ComputerSystem : CIM_ComputerSystem {
};

[Version("1.0"),
 Description(
   "VNE_SystemComponent , as a specialization of CIM_SystemComponent." )]
class VNE_SystemComponent : CIM_SystemComponent {
};

[Version("1.0"),
 Description(
   "VNE_HostedAccessPoint , as a specialization of CIM_HostedAccessPoint." )]
class VNE_HostedAccessPoint : CIM_HostedAccessPoint {
};

[Version("1.0"),
 Description(
   "VNE_IPProtocolEndpoint as a specialization of CIM_IPProtocolEndpoint." )]
class VNE_IPProtocolEndpoint : CIM_IPProtocolEndpoint {
```

```
};

[Version("1.0"),
 Description(
    "VNE_StaticIPAssignmentSettingData as a specialization of
CIM_StaticIPAssignmentSettingData." )]
class VNE_StaticIPAssignmentSettingData : CIM_StaticIPAssignmentSettingData {
};



[Version("1.0"),
 Description(
    "VNE_ElementSettingData , as a specialization of CIM_ElementSettingData." )]
class VNE_ElementSettingData : CIM_ElementSettingData {

};

[Version("1.0"),
 Description(
    "VNE_ConnectivityCollection models the scenario links (i.e. node"
    "interconnections)." )]
class VNE_ConnectivityCollection : CIM_ConnectivityCollection {

    [Description(
        "The max number of connection allowed in the VNE_ConnectivityCollection. "
        "For example, MaxConnections=2 implies a point-to-point connection." )]
    uint16 MaxConnections;

};

[Version("1.0"),
 Description(
    "VNE_CompleteAddr objects model network interface"
    "IPv6 addresses. It is needed due to the existing VNE_StaticIPAssignmentSettingData" )]
class VNE_CompleteAddr : CIM_IPAssignmentSettingData {

    [Description ("The complete address that will be assigned to the ProtocolEndpoint.")]
    string completeAddr;
    string generatedAddressOffset;
```

```
    string device1;
    string device2;
    string device3;

};

// association classes

[Association, Version("1.0"),
 Description(
    "VNE_MemberOfCollection associate a VNELinkConnectivityConnection to"
    "the VNE_IPProtocolEndpoints belonging to it." )]
class VNE_MemberOfLink : CIM_MemberOfCollection {

    [Description("The LinkConnectivityCollection modeling the link." ),
     Override ("Collection")]
    VNE_ConnectivityCollection REF Collection;

    [Description("The IPProtocolEndPoint member of the link."),
     Override ("Member")]
    VNE_IPProtocolEndpoint REF Member;
};

// configuracion.mof, v1
//
// Copyright (C) 2009/JUL Walter Fuertes - Fausto Meneses


[Version("1.0"),
 Description(
    "VNE_Configuration describe los parÃ¡metros base." )]
class VNE_Configuration : CIM_SettingData {
            string valPlataform;
            string valVersion;
            string valPath;
            string valExecMode;
            string valFilesystemType;
            string valFilesystem;
            string valKernel;
```

```
        string valConsole;
};

// valConfiguration.mof, v1
//
// Copyright (C) 2009/JUL Walter Fuertes - Fausto Meneses.

instance of VNE_Configuration {

      InstanceID = "vne-xenvalues";
        valPlataform = "xen";
        valVersion = "3.3.1";
      valPath = "/etc/xen/";
        valExecMode = "wfconsole";
        valFilesystemType = "ext3";
        valFilesystem = "/def/sda";
        valKernel = "/boot/vmlinuz-2.6.27-14-server";
        valConsole = "xterm";
};

instance of VNE_Configuration {

   InstanceID = "vne-vmwarevalues";
        valPlataform = "vmware";
        valVersion = "3.3.1";
      valPath = "/var/lib/vmware/Virtual";
        valExecMode = "wfconsole";
        valFilesystemType = "ext3";
        valFilesystem = "/var/lib/vmware/Virtual Machines/";
        valKernel = "otherlinux";
        valConsole = "pc1";
};

// xenImplem.mof, v1
//
// Copyright (C) 2009/JUL Walter Fuertes - Fausto Meneses

// Network (1)
instance of VNE_Network as $ENVIRONMENT {
```

```
        // key properties
        CreationClassName = "VNE_Network";
        Name = "xenImplem";

};

// the virtual machine (3)

instance of VNE_ComputerSystem as $VM1 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "vm1";

};

instance of VNE_ComputerSystem as $VM2 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "vm2";

};

instance of VNE_ComputerSystem as $VM3 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "vm3";

};

// NICS (3)

instance of VNE_IPProtocolEndpoint as $VM1_NIC1 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
```

```
        Name = "vm1-nic1";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "vm1";

};

instance of VNE_IPProtocolEndpoint as $VM2_NIC2 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
        Name = "vm2-nic2";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "vm2";

};

instance of VNE_IPProtocolEndpoint as $VM3_NIC3 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
        Name = "vm3-nic3";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "vm3";

};

// IPV4 (3):

instance of VNE_StaticIPAssignmentSettingData as $VM1_IPV4_1 {

        // key properties
        InstanceID = "vm1-ip1";
        IPv4Address = "10.1.14.241";
        SubnetMask = "255.255.255.0";

};

instance of VNE_StaticIPAssignmentSettingData as $VM2_IPV4_1 {
```

```
        // key properties
        InstanceID = "vm2-ip1";
        IPv4Address = "10.1.14.242";
        SubnetMask = "255.255.255.0";

};

instance of VNE_StaticIPAssignmentSettingData as $VM3_IPV4_1 {

        // key properties
        InstanceID = "vm3-ip1";
        IPv4Address = "10.1.14.243";
        SubnetMask = "255.255.255.0";

};

// Complete Address (3):

instance of VNE_CompleteAddr as $VM1_CPAD_1 {

        // key properties
        InstanceID = "vm1-ip2";

        completeAddr = "ip=10.1.14.241,mac=00:16:3E:5C:6D:87";
    generatedAddressOffset = "0";
    device1 = "file:/home/xen/domains/xen1.example.com/swap.img,xvda1,w";
    device2 = "file:/home/xen/domains/xen1.example.com/disk.img,xvda2,w";
    device3 = "";

};

instance of VNE_CompleteAddr as $VM2_CPAD_1 {

        // key properties
        InstanceID = "vm2-ip2";

        completeAddr = "ip=10.1.14.242,mac=00:16:3E:5C:6D:88";
    generatedAddressOffset = "0";
    device1 = "file:/home/xen/domains/xen2.example.com/swap.img,xvda1,w";
```

```
        device2 = "file:/home/xen/domains/xen2.example.com/disk.img,xvda2,w";
        device3 = "";

};

instance of VNE_CompleteAddr as $VM3_CPAD_1 {

            // key properties
            InstanceID = "vm3-ip2";

            completeAddr = "ip=10.1.14.243,mac=00:16:3E:5C:6D:89";
        generatedAddressOffset = "0";
        device1 = "file:/home/xen/domains/xen3.example.com/swap.img,xvda1,w";
        device2 = "file:/home/xen/domains/xen3.example.com/disk.img,xvda2,w";
        device3 = "";

};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET0 {

            // key properties
            InstanceID = "net0";

};

// interfaces associations to networks vm (3):

instance of VNE_SystemComponent {
            GroupComponent = $ENVIRONMENT;
            PartComponent = $VM1;

};

instance of VNE_SystemComponent {
            GroupComponent = $ENVIRONMENT;
            PartComponent = $VM2;
```

```
};

instance of VNE_SystemComponent {
        GroupComponent = $ENVIRONMENT;
        PartComponent = $VM3;

};

// interfaces associations to networks (3):

instance of VNE_MemberOfLink {
        Collection = $NET0;
        Member = $VM1_NIC1;

};

instance of VNE_MemberOfLink {
        Collection = $NET0;
        Member = $VM2_NIC2;

};

instance of VNE_MemberOfLink {
        Collection = $NET0;
        Member = $VM3_NIC3;

};

// interfaces associations to nodes (3):

instance of VNE_HostedAccessPoint {
        Antecedent = $VM1;
        Dependent = $VM1_NIC1;
};

instance of VNE_HostedAccessPoint {
        Antecedent = $VM2;
        Dependent = $VM2_NIC2;
};
```

```
instance of VNE_HostedAccessPoint {
        Antecedent = $VM3;
        Dependent = $VM3_NIC3;
};

// IPv4 associations to interfaces (3):

instance of VNE_ElementSettingData {
        ManagedElement = $VM1_NIC1;
        SettingData = $VM1_IPV4_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $VM2_NIC2;
        SettingData = $VM2_IPV4_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $VM3_NIC3;
        SettingData = $VM3_IPV4_1;
};

// Complete address associations to interfaces (3):

instance of VNE_ElementSettingData {
        ManagedElement = $VM1_NIC1;
        SettingData = $VM1_CPAD_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $VM2_NIC2;
        SettingData = $VM2_CPAD_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $VM3_NIC3;
        SettingData = $VM3_CPAD_1;
};
```

```
// vmwareImplem.mof, v1
//
// Copyright (C) 2009/JUL Walter Fuertes - Fausto Meneses.

// Network (1)
instance of VNE_Network as $ENVIRONMENT {

        // key properties
        CreationClassName = "VNE_Network";
        Name = "vmwareImplem";

};

// the virtual machine (3)

instance of VNE_ComputerSystem as $DEBIAN1 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "debian1";

};

instance of VNE_ComputerSystem as $DEBIAN2 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "debian2";

};

instance of VNE_ComputerSystem as $DEBIAN3 {

        // key properties
        CreationClassName = "VNE_ComputerSystem";
        Name = "debian3";

};
```

```
// NICS (3)

instance of VNE_IPProtocolEndpoint as $DEBIAN1_NIC1 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
        Name = "debian1-nic1";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "debian1";

};

instance of VNE_IPProtocolEndpoint as $DEBIAN2_NIC2 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
        Name = "debian2-nic2";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "debian2";

};

instance of VNE_IPProtocolEndpoint as $DEBIAN3_NIC3 {

        // key properties
        CreationClassName = "VNE_IPProtocolEndpoint";
        Name = "debian3-nic3";
        SystemCreationClassName = "VNE_ComputerSystem";
        SystemName = "debian3";

};

// IPV4 (3):

instance of VNE_StaticIPAssignmentSettingData as $DEBIAN1_IPV4_1 {

        // key properties
        InstanceID = "debian1-ip1";
```

```
        IPv4Address = "10.1.14.241";
        SubnetMask = "255.255.255.0";

};

instance of VNE_StaticIPAssignmentSettingData as $DEBIAN2_IPV4_1 {

        // key properties
        InstanceID = "debian2-ip1";
        IPv4Address = "10.1.14.242";
        SubnetMask = "255.255.255.0";

};

instance of VNE_StaticIPAssignmentSettingData as $DEBIAN3_IPV4_1 {

        // key properties
        InstanceID = "debian3-ip1";
        IPv4Address = "10.1.14.243";
        SubnetMask = "255.255.255.0";

};

// Complete Address (3):

instance of VNE_CompleteAddr as $DEBIAN1_CPAD_1 {

        // key properties
        InstanceID = "debian1-ip2";

        completeAddr = "00:0c:29:cf:37:fa";
    generatedAddressOffset = "0";
    device1 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device2 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device3 = "";

};

instance of VNE_CompleteAddr as $DEBIAN2_CPAD_1 {
```

```
        // key properties
        InstanceID = "debian2-ip2";

        completeAddr = "00:0c:29:cf:38:fa";
    generatedAddressOffset = "0";
    device1 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device2 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device3 = "";

};

instance of VNE_CompleteAddr as $DEBIAN3_CPAD_1 {

        // key properties
        InstanceID = "debian3-ip2";

        completeAddr = "00:0c:29:cf:39:fa";
    generatedAddressOffset = "0";
    device1 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device2 = "56 4d 0c c0 58 cf b5 c4-df 3a 29 a0 82 cf 37 fa";
    device3 = "";

};

// the interconnection networks (1)

instance of VNE_ConnectivityCollection as $NET1 {

        // key properties
        InstanceID = "net1";

};

// interfaces associations to networks vm (3):

instance of VNE_SystemComponent {
        GroupComponent = $ENVIRONMENT;
        PartComponent = $DEBIAN1;
```

```
};

instance of VNE_SystemComponent {
        GroupComponent = $ENVIRONMENT;
        PartComponent = $DEBIAN2;

};

instance of VNE_SystemComponent {
        GroupComponent = $ENVIRONMENT;
        PartComponent = $DEBIAN3;

};

// interfaces associations to networks (3):

instance of VNE_MemberOfLink {
        Collection = $NET1;
        Member = $DEBIAN1_NIC1;

};

instance of VNE_MemberOfLink {
        Collection = $NET1;
        Member = $DEBIAN2_NIC2;

};

instance of VNE_MemberOfLink {
        Collection = $NET1;
        Member = $DEBIAN3_NIC3;

};

// interfaces associations to nodes (3):

instance of VNE_HostedAccessPoint {
        Antecedent = $DEBIAN1;
```

```
        Dependent = $DEBIAN1_NIC1;
};

instance of VNE_HostedAccessPoint {
        Antecedent = $DEBIAN2;
        Dependent = $DEBIAN2_NIC2;
};

instance of VNE_HostedAccessPoint {
        Antecedent = $DEBIAN3;
        Dependent = $DEBIAN3_NIC3;
};

// IPv4 associations to interfaces (3):

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN1_NIC1;
        SettingData = $DEBIAN1_IPV4_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN2_NIC2;
        SettingData = $DEBIAN2_IPV4_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN3_NIC3;
        SettingData = $DEBIAN3_IPV4_1;
};

// Complete address associations to interfaces (3):

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN1_NIC1;
        SettingData = $DEBIAN1_CPAD_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN2_NIC2;
```

```
        SettingData = $DEBIAN2_CPAD_1;
};

instance of VNE_ElementSettingData {
        ManagedElement = $DEBIAN3_NIC3;
        SettingData = $DEBIAN3_CPAD_1;
};
```

Referencias

A

[Adams06] K. Adams, O. Agesen: "***A Comparison of Software and Hardware Techniques for x86 Virtualization***". Proc. 12th Int. Conf. on Architectural Support for Programming Languages and Operating Systems. San Jose, California, USA (2006).

[Autobench] Autobench. http://www.xenoclast.org/autobench/

[Apostolopoulos02] J. G. Apostolopoulos, W. Tan, and S. J. Wee, "***Video Streaming: Concepts, Algorithms, and Systems***". Mobile and Media Systems Laboratory. HP Tech. Report 2002-260, Sep. 18th, 2002.

B

[Barham03] 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.

[Baumgartner03] F. Baumgartner, T. Braun, E. Kurt, A. Weyl, "***Virtual routers: A tool for networking research and education***". ACM SIGCOMM Computer Communication. Vol. 33, pp: 127–135. NY, July 2003.

[Begnum06] K. Begnum, ***"Managing Larg Networks of Virtual Machines"***, In Proc of 20th Large Installation System Administration Conf (LISA'06), Washington DC (USA), pp. 205-214, December 2006.

[Bouabid09] A. Bouabid, P. Vidal, J. Broisin, ***"Integrating Learning Management Systems and Practical Learning Activities: The Case of Computer and Network Experiments"*** icalt, pp.398-402, 2009 Ninth IEEE International Conference on Advanced Learning Technologies, 2009

[Brastaad06] E. Braastad: "***Management of high availability services using virtualization***". Master Thesis. Department of Informatics, Oslo University College (May 2006).

[Brown06] M. A. Brown. "***Traffic Control how to***". Version 1.0.2. Oct, 2006.

[Booch97] Grady Booch, Jim Rumbaugh, and Ivar Jacobson, "***Unified Modeling Language User Guide***", ISBN: 0-201-57168-4, Addison Wesley, est. publication December 1997.

C

[Casazza06] J. Casazza, M. Greenfield, K. Shi, ***"Redefining Server Performance Characterization for Virtualization Benchmarking"***. Intel® Technology Journal. Vol. 10, Issue 03. Published: August 10, 2006 ISSN 1535-864X J.

[Canavos88] G. Canavos, "***Probabilidad y Estadística: aplicaciones y métodos***", McGraw-Hill/Interamericana, Mexico, 1988.

[Caramazana04] A. Caramazana Cárcamo. "***Tecnologías MDA (Model Driven Architecture) para el Desarrollo de Software***". Universdad Pontificia de Salmanca, Madrid, España, 2004.

[CIM-Tutorial] DMTF-CIM Tutorial, Availaible at: http://www.wbemsolutions.com/tutorials/CIM/cim-core.html

[Cherkasova05] L. Cherkasova, R. Gardner, "***Measuring CPU overhead for I/O processing in the Xen virtual machine monitor***". In Proccedings of USENIX Annual Technical Conference, p. 24-32, Anaheim, CA, April 10-15, 2005.

[Convirt10] Convirture blog, "***Introducing ConVirt 2.0***", Feb 28, 2010. Available at: http://www.convirture.com/blog/2010/announcements/introducing-convirt-2-0/

[Córdova02] Manuel Córdova Zamora, "***Estadística Inferencial***", Editorial Moshera SLR, ISBN 9972-813-15-0, 2002.

[Córdova06] Manuel Córdova Zamora, "***Estadística Aplicada***", Editorial Moshera SLR, 2006.

[Czajkowski04] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, "***The WS-Resource Framework***", 3 May 2004, white paper.

D

[Deandrade07] Marcos Tadeu de Andrade: "***Un estudio comparativo sobre las principales herramientas de Virtualización".*** Trabajo de Graduación. Universidad Federal de Pernambuco. Centro de Informática. Departamento de Ciencias de la computación. 2007

[Deshane06] T. Deshane, D. Dimatos, G. Hamilton, M. Hapuarachchi, W. Hu, Michael McCabe, J. N. Matthews, "***Performance nsolation of a Misbehaving Virtual Machines with Xen, VMware and Solaris Contaier***s". Clarkson University White Paper. [Available on]: http://people.clarkson.edu/~jnm/publications/isolationOfMisbehavingMVs.pdf . Feb 2006.

[Dike06] J. Dike, "***User Mode Linux***", Prentice Hall, 2006

[DMTFDSP2013] DMTF, ***"CIM System Virtualization Model White Paper"***. Document Number: DSP2013. Version 1.0.0, November 2007.

[DMTFDSP1042] DMTF, ***"CIM System Virtualization Profile"***. Document Number: DSP1042. Version: 1.0.0a, August 2007.

[DMTFDSP1057] DMTF, ***"CIM Virtual System Profile"***. Document Number: DSP1057. Version: 1.0.0a, May 2007.

[DMTFDSP2043] DMTF, ***"Open Virtualization Format Specification"***, Docuument Number: DSP0243, Version: 1.0.0, February 2009.

[DMTFDSP0111] DMTF, ***"Common Information Model (CIM) Core Model"***. Document Number DSP0111. Version 2.4, August 2000.

[DMTFDSP0152] DMTF, "***CIM Network Model Whitepaper***", Documentt Number DSP0152, Version. 1.1, December 2003.

[DMTFDSP2019] DMTF, "***UML profile for CIM***", version 1.0, DMTF Document, DSP0219, June 2007.

E

[Emulab] Emulab, disponible en http://www.emulab.net/doc.php3

[Enomalism] "Enomalism", http://www.enomalism.com (última comprobación, 26 de mayo 2008).

F

[Falcon07] A. Falcon, P. Faraboschi, D. Ortega, "***Combining Simulation and Virtualization through Dynamic Sampling***". In Proc of IEEE International Symposium on Performance Analysis of Systems & Software, 2007. ISPASS 2007. pps: 72-83, San Jose, CA, April 2007.

[Fahy08] C. Fahy et al., "***Towars an Information Model That Suppoerts Service-Aware, Self-managing Virtual Resources***". MACE 2008, LNCS 5276, pp. 102-107. Berlín 2008.

[Figueiredo05] Renato Figueiredo, Peter A. Dinda, Jose Fortes: "***Resource Virtualization Renaissance***". IEEE Computer, Vol. 38, Issue 5, May 2005, pp. 28 - 31.

[Figuereido03] Renato Figueiredo, Peter A. Dinda, Jose Fortes, "***A Case For Grid Computing On Virtual Machines***", in Proc. 23rd International Conference on Distributed Computing Systems, May 2003, pp. 550-559.

[Fernando06] G. Fernando, "***To V or not to V: A practical guide to virtualization***". White paper. BMC Software, Inc. January 2006.

[Fernández06] David Fernández y Fermín Galán. "***Virtual Network User Mode Linux: Herramienta de creación de Escenarios de red basada en Virtualización para la Internet de Nueva Generación***", Memoria Descriptiva. 6º Edición del Premio de Nuevas Aplicaciones de Internet, November 2006.

[Foster01] I. Foster, C. Kesselman, S. Tuecke, "***The Anatomy of the Grid: Enabling Scalable Virtual Organizations***", in Proc. International Journal of High Performance Computing Applications 2001, Volume 15, issue (3), pp. 200–222.

[Foster05] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, "***Modeling Sateteful Resources with Web Services***", Ver 1.1, Mar 2005.

[Foster08] I. Foster, "***What is the Grid? A Three Point Checklist***", http://www.gridtoday.com/02/0722/100136.html (última comprobación, 26 de mayo 2008).

[Flores06] 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.

[Fuertes07] 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.

[Fuertes08a] W. Fuertes, J. E. López de Vergara, "***Evaluación de Plataformas de virtualización para experimentación de servicios multimedia en redes IP***". Publicado en la Revista de Ciencia y Tecnología de la Escuela Politécnica del Ejército, Volumen 1, pps: 35-46, ISSN: 1390-4612, Sangolquí, Ecuador, el 11-Ago-2008.

[Fuertes08b] W. Fuertes, J. E. López de Vergara, F. Galán, D. Fernández, "***Propuesta para el Despliegue de Escenarios de Red Virtuales en Entornos Distribuidos***, publicado en las Actas de las VII Jornadas de Ingeniería Telemática, Jitel'2008, Alcalá de Henares, Madrid, 16-18 septiembre 2008. ISBN: 978-84-612-5474-3, págs. 126-133.

[Fuertes09a] 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. Date of acceptance: 09-12-2008. In addition, pulished in Electronic Communications of the EASST, Volume 17, ISSN 1863-2122.

[Fuertes09b] W. Fuertes, J. E. López de Vergara, F. Meneses, "***Educational Platform using Virtualization Technologies: Teaching-Learning Applications and Research Uses Cases***". in Proc pf II ACE Seminar: Knowledge Construction in Online Collaborative Communities, Albuquerque, NM – USA, October 2009.

[Fuertes09c] W. Fuertes, L. Jácome, M. Grijalva, J. E. López de Vergara, R. Fonseca, "***Evaluación del Rendimiento de Redes IP utilizando Plataformas de Virtualización y Métodos de Simulación"***, DECC Report-Revista técnica del Departamento de Ciencias de la Computación de la Escuela Politécnica del Ejército, Volumen 1, Sangolquí, Ecuador, Diciembre de 2009, pp. 34-43. ISSN 1390-5236

[Fuertes10a] Walter M. Fuertes, Jorge E. López de Vergara, Fausto H. Meneses, Fermín Galán, "***A Generic Model for the Management of Virtual Network Environments***", accepted for its publication in Proc. 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan, 19-23 April 2010

[Fuertes10b] W. Fuertes, J. E. López de Vergara, J. Pincha, H. Aules, L. Jácome, M. Grijalva, "***Analytical Expression to Predict the Overhead Produced by the VMware and Xen Virtualization Tools***". Aceptado para publicación en el 5to. Congreso de Ciencia y Tecnología ESPE-2010, ISSN 1390-4663, Sangolquí-Ecuador, 15-18 de junio de 2010.

[Friedman07] M. Friedman, S. Marksamer, ***"A Realistic Assessment of the Performance of Windows Guest Virtual Machines"***, Computer Measurement Group White paper. March 2007.

G

[Galán04] Fermín Galán Márquez. "***Tecnología Xml en la herramienta de simulación de redes vnuml***". DIT. Universidad Politécnica de Madrid, Marzo 2004.

[Galán06] F. Galán, E. García, C. Chávarri, D. Fernández, M. Gómez, "***Design and Implementation of an IP Multimedia Subsystem (IMS) Emulator Using Virtualization Techniques***". In Proc. of the 13th HP OpenView University Association Workshop (HP-OVUA 2006), pp: 213-224, France, May 2006.

[Galán07] Fermín Galán, and David Fernández, "***Distributed Virtualization Scenarios Using VNUML***", Proc. Systems and Virtualization Management 2007, Toulouse, France, October 2007.

[Galán08a] Galán, F., López de Vergara, J., Fernández, D., Muñoz, R. "***A Model-driven Configuration Management Methodology for Testbed Infrastructures"***, IEEE/IFIP Network Operations and Management Symposium (NOMS 08), Salvador da Bahia (Brazil), pp. (747-750), 7-11 April 2008.

[Galán08b] F. Galán, D. Fernández, M. Ferrer, F. J. Martín, ***"Scenario-based Distributed Virtualization Management Architecture for Multi-host Environments"***, System and Virtualization Management Workshop (SMV 2008), CCIS 18, pp. 49-60, Munich (Germany), October 2008.

[Galán09] F. Galán, D. Fernández, W. Fuertes, M. Gómez and J. E. López de Vergara, "***Scenario-based virtual network infrastructure management in research and educationnal testbeds with VNUML***", Annals of Telecommunications, vol. 64(5), pp. 305-323, May 2009.

[Galán10] Fermín Galán M, "***On Scenario-based Model-driven Configuration Management for Flexible Networking Experimentation Infrastructures***, PhD. Thesis, Director: David Fernandez C., Universidad Politecnica de Madrid, España, Abril de 2010.

[Galindo99] Edwin Galindo, "**Estadística para la Administración y la Ingeniería**", Editorial Mediavilla Hnos, ISBN 9978-82-991-1, 1999

[Garbacki07] P. Garbacki, K. Vijay, "***Efficient Resource Virtualization and Sharing Strategies for Heterogeneous Grid Environments***", in Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich Germany, May 2007, pp. 40 – 49.

[García06] J. García Monroy, ***"Globus Toolkit"***, E.T.S.I Telecomunicación, Departamento de Ingeniería de Sistemas Telemáticos, Madrid, España 2006.

[Garcia07] J.L. García-Dorado et al., "***A quality of service assessment technique for large-scale management of multimedia flows***". In Proc. of 10th IFIP/IEEE (MMNS'2007), San José, California, October 31 - November 2, 2007. Lecture Notes in Computer Science, Vol 4787, pp: 173-176.

[Gartner08] Gartner Inc. ***"Top 10 disruptive technologies according to Gartner",*** Posted on May 31, 2008, UK [Online] http://www.nevillehobson.com/2008/05/31/top-10-disruptive-technologies-according-to-gartner/

[Grau06] Andreas Grau, Harald Weinschrott, Christopher Schwarzer: "***Evaluating the Scalability of Virtual Machines for Use in Computer Network Emulation***". Stuttgart University. Oct/06.

[Grehant08] X. Grehant, O. Pernet, S. Jarp, I. Demeure, and P. Toft, ***Xen Management with SmartFrog On-Demand Supply of Heterogeneous, Synchronized Execution Environments***, In Proc of Euro-Par 2007 Workshops: Parallel Processing HPPC 2007, Volume 4854/2008, pp: 206-213, March, 2008

[Griffyn06] C. Griffyn, ***"Managing Xen DMTF Standards and CIM Provider Work"***. Novel White Paper. January 2006.

H

[Hart-Sears07] T. Hart-Sears and J. Lofton: "***Server Virtualization: The New Future for Midrange Implementation***". Technology Partners International, Inc. July 2007 Copyright 2007 – All Rights Reserved 2070407

[Hemminger05] Stephen Hemminger: "***Network Emulation with NetEm***". Open Source Development Lab. April 2005.

[Hobbs04] C. Hobbs. "***A Practical Approach to WBEM/CIM Management***", Published by CRC Press, ISBN 9780203500132, Nov 2, 2004.

[Humphreys06] John Humphreys and Tim Grieser, ***"Mainstreaming Server Virtualization. The Intel Aproach"***. White paper. Sponsored by Intel. June, 2006.

I

[Issariyakul09] Teerawat Issariyakul and Ekram Hossain. "***Introduction to Network Simulator NS-2***". ISBN 987-0-387-71759-3, Springer 2009.

[IntelTR08] Intel Software Network ***"Measuring Performance of Applications on Virtualized Systems Under Test (SUTs)"*** Technical Report. October 2008.

[Imunes] ***Imunes***: Faculty of Electrical Engineering and Computing, University of Zagreb. Department of Telecommunications. [OnLine] http://www.tel.fer.hr/imunes

[Iperf] Iperf. Internet performance testing. En http://dast.nlanr.net/Projects/Iperf/

J

[Jacobson] V. Jacobson, C. Leres, and S. McCanne: "***Tcpdump***". Available at anonymous@ftp.ee.lbl.gov.

[Jain91] R. Jain, ***"The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,"*** Wiley- Interscience, New York, NY, April 1991, ISBN: 0471503361.

[Jiang03] X. Jiang and D. Xu, ***"vBET: a VM-based emulation testbed***", In Proc. of the ACM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools'03), ACM Association, August 2003.

[Jones06] M. Tim Jones: "***An overview of virtualization methods, architectures, and implementations***". Consultant Engineer, Emulex.Virtual Linux. Dec, 2006.

K

[Keahey05] K. Keahey, I. Foster, T. Freeman, T., Zhang, X. Galron, "***Virtual Workspaces in the Grid***", in Proc. Parallel Processing 11th International Euro-Par Conference,, Lisbon, Portugal. September, 2005

[Kecskemeti08] G. Kecskemeti, P. Kacsuk, G. Terstyanszky, T. Kiss, T. Delaitre, "***Automatic Service Deployment Using Virtualisation***", in Proc. 6th

Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), pp. 628-635.

[Kullback51] S. Kullback and R.A. Leibler, "***On information and sufficiency***". Annals of Mathematical Statistics, 22(1), pp:79-86, March 1951.

[Kunce08] M. Kunce, L. Wang, "***Information Service of Virtual Machine Pool Grid Computing***", Proc. Euro-Par 2007 Workshops: Parallel Processing, *LNCS* 4854, pp. 174-184, Berlin August, 2008.

[KVM] KMV. home page: http://kMV.qumranet.com/kMVwiki

L

[Libvirt-CIM] Libvirt-CIM provider [Online] http://www.libvirt.org/CIM/.

[López03] Jorge Enrique López de Vergara Méndez, Director: Víctor Abraham Villagrá González, ***Especificación de Modelos de Información de Gestión de Red Integrada Mediante el Uso de Ontologías y Técnicas de Representación del Conocimiento***, Tesis Doctoral (*Ph. D. Thesis*), Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid. 2003.

M

[Markhija06] Vikram Makhija, Bruce Rendón, Lisa Roderick, Eric Zamost, Jennifer Andresdon: "***VMmark A Scalable benchMArk for Virtualized Systems***". Technical Report. Vmware-TR´2006-002. Sept, 2006.

[Marshal06] D. Marshall, W. Reynolds and D. McCrory, "***Advanced Server Virtualization, VMware and Microsoft Platforms in the virtual Center***,". In Averbach Publications. ISBN 849339316. May 2006.

[Maier2005] S. Maier, D. Herrscher, K. Rothermel, "***Experiences with node virtualization for scalable network emulation***". Computer Communications. Volume 30, Issue 5, pp: 943-956. March 2007.

[Matsunaga05] A. Matsunaga, M. Tsugawa, M. Zhao, L. Zhu, V. Sanjeepan, S. Adabala, R. Figueiredo, H. Lam, J. Fortes, "***On the Use of Virtualization and Service Technologies to Enable Grid-Computing***", in Proc. Parallel Processing 11th International Euro-Par Conference, Lisbon, September 2005.

[Mellor08] E. Mellor, R. Sharp, D. Scott, ***"Xen Management API"***. API Revision 1.0.6, Copyright XenSource, Inc. Xen, July 2008.

[Menon05] A. Menon, J. Santos, Y. Turner, G. Janakiraman, W. Zwaenepoel, ***"Diagnosing Performance Overheads in the Xen Virtual Machine Environment"***, In Proc. of the 1st ACM/USENIX international conference on Virtual execution environments, Pages: 13 – 23, Chicago USA 2005.

[Mendenhall94] W. Mendenhall, D. Wackerly, and R. Scheaffer, "***Estadística Matemática con Aplicaciones***", Grupo Editorial Iberoamericana, Mexico, 1994.

[Modelnet] Modelnet, http://modelnet.ucsd.edu/howto.html

[Mosberger98] D. Mosberger, T. Jin, "***Httperf-A Tool for Measuring Web Server Performance***". In Proc. of Workshop on Internet Server Performance, 1998.

[Muñoz06] Alex Muñoz: ***Academic OPNET Research and Educational Projects***, University of Basque Country, Departamento de Electrónica y Comunicaciones. http://det.bi.ehu.es/NQAS/opnet/

N

[Netkit] NetKit: "***The poor man's system to experiment computer networking***". http://www.netkit.org/

[Netperf] Netperf. Network rendimiento [Online]:

http://www.netperf.org/netperf/NetperfPage.html.

[NS2-08] "***The Network Simulator NS2***", http://www.isi.edu/nsnam/ns (última comprobación, 26 de mayo 2008).

O

[OpenVZ] OpenVz. Home page: http://www.openvz.org

[OMG03] Object Management Group, "***MDA guide version 1.0.1***", OMG Document omg/03-06-01, June 2003.

[OMG] OMG Unified Modeling Language, home page: http://www.uml.org/

[OVF07] VMware Inc, XenSource. "**The Open Virtual Machine Format**". Whitepaper for OVF Specification version 0.9, Sep/2007.

P

[Padala07] P. Padala, X. Zhu, Z. Wang, S. Singhal, K. Shin. "***Performance Evaluation of Virtualization Technologies for Server Consolidation***". HP Labs Tech Report HPL- 2007-59, 2007.

[Pizzonia08] Pizzonia, M., and Rimondini, M. ***"Netkit: Easy Emulation of Complex Networks on Inexpensive Hardware"***. In Proc. of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom 2008), Innsbruck, Austria, 2008.

[Popek74] Gerald J. Popek and Robert P. Goldberg. "***Formal requirements for virtualizable third generation architectures"***. CACM, 17(7):413–421, 1974.

[PlanetLab] PlanteLab Home Page. Available at http://www.planet-lab.org/

Q

[Qemu] Qemu Open source proccesor emulator: http://fabrice.bellard.free.fr/qemu

[Quintero07] J. Quintero, R. Anaya "***MDA y el papel de los modelos en el proceso de desarrollo de software***". Revista EIA, ISSN 1794-1237 Número 8, p. 131-146. Diciembre 2007 Escuela de Ingeniería de Antioquia, Medellín (Colombia).

R

[Raj07] H. Raj, B. Seshasayee, K. Schwan, "***VMedia: Enhanced Multimedia Services in Virtualized Systems***". Published by Georgia Institute of Technology. Report No.: GIT-CERCS-07-19. 2007.

[Ramachandran09] K. M. Ramachandran, C. P. Tsokos. ***"Mathematical statistics with applications"***, Elsevier Academic Press, ISBN 978-012-3748485, 2009.

[Rimondini07] M. Rimondini: "***Emulation of Computer Networks with Netkit***". Department of Computer science and Automatization, University of Rome, Italy. January 2007

[Rmodp08] "***RM-ODP: The Reference Model for Open Distributed Processing***", http://www.rm-odp.net/ (última comprobación, 26 de mayo 2008).

[Rubio07] A. Rubio-Montero, E. Huedo, R. Montero, I. Llorente, "***Management of Virtual Machines on Globus Grids Using GridWay***", in Proc. IEEE International Parallel and Distributed Processing Symposium, 2007, IPDPS 2007, March 2007.

[Ruth05] P. Ruth, X. Jiang, D. Xu, "***VIOLIN: Virtual Internetworking on OverLay Infrastructure***", IEEE Computer Soc., Vol. 38, 5, May 2005, pp. 63-69.

S

[Seetharaman06] S. Seetharaman and K. Murthy: "***Test Optimization Using Software Virtualization***". IEEE Software, Vol. 23, No. 5, Sept., 2006, pps: 66-69.

[Spec] SPEC, Standard Performance Evaluation Corporation. [Disponible en:] http://www.spec.org/

[Sotomayor07] B. Sotomayor, ***"The Globus Toolkit 4 Programmer's Tutorial"***, University of Chicago, Department of Computer Science.

[Song07] Y. Song, H. Wang, Y. Li, Y. Sun, Y. Zeng, "***Can VoD streaming service co-exist with other services on a VM-based virtualized computing platform?***". In Proc. of the 2007 Asian technology information China. HPC-'2007, pp: 95-103.

[Scope08] SCOPE Alliance "***Virtualization: State of the Art"***. Version 1.0, April 3, 2008.

[SmartDom] SmartDomains tutorial [Availaible on]: http://smartdomains.sourceforge.net/launchsimpledesc.html

[Sundararaj04] A. Sundararaj, P. Dinda, ***"Towards virtual networks for virtual machine Grid computing"***, in Proc. 3rd USENIX Virtual Machine Research And Technology Symposium, MV 2004.

T

[Tcpdump] Tcpdump & libpcap http://www.tcpdump.org/

[Turjanski05] P. Turjanski, D. Fernández, J.P. Suárez, A. Panelli, A. Soba, G. Marshal, "***Computación de alto rendimiento utilizando Grid computing en un***

entorno multiplataforma", MECOM 2005: VIII Congreso Argentino de Mecánica Computacional, Volume 24, pp. 78-92 – Nov. 2005.

U

[Ubench] Ubench 0.32. http://www.phystech.com/download/

[UML] The User-mode Linux Kernel Home Page.

http://user-mode-linux.sourceforge.net/

[UnixBench] UnixBench. https://www.nsa.gov/selinux/papers/freenix01/node15.html

V

[Vanderham04] Jeroen Van Der Ham and Gert Jan Verhoog: "***Virtual environments for networking experiments***". Analytical Network Project, Masters System and Network Administration, University of Amsterdam, the Netherlands. July 1, 2004.

[VirtualBox] Virtual Box, home page: http://www.virtualbox.org/

[Vyatta07] Vyatta, Inc. ***The top Five Virtualization Mistakes***. White Paper. Belmont USA, 2007.

[Vlc] Video Lan Streaming Solution, en http://www.videolan.org/

[Vlan01] "Virtual Local Area Networks", IEEE Standard 802.1q, 2001.

[VMMark06] MVMark. ***A Standard Virtualization Benchamrk***. Home site http://www.VMware.com/products/MVmark/

[VMAN] DMTF, "***Standards for Virtualization Management***". MVAN Initiative for Managing Virtualized Systems, [Online] http://www.dmtf.org/initiatives/MVan_initiative/.

[VMwareVC] VMware VirtualCenter", http://www.VMware.com/products/vi/vc (última comprobación, 26 de mayo 2008).

[VMwareGuide06] VMware, Inc: "***Administration Guide, VMware Server 1.0.***" Revision: 20060706. 2006.

[VMware08] VMware, Inc. "***Timekeeping in VMware virtual machines***". White papers. http://www.VMware.com/pdf/VMware_timekeeping.pdf. Latest revision: 12 August 2008.

[VMware-CIM] VMware Inc. "***VMware-CIM SDK, Programing Guide***", Versión 1.0, http://www.VMware.com/pdf/CIM_SDK_Prog_Guide_esx30.pdf

[VMware-SMASH] VMware-CIM SMASH API Programming Guide, ESX Server 3i version 3.5, VMware Inc., Palo Alto, December, 2007.

[VirtualCenter07] VMware, Inc, "***VMware VirtualCenter***", White paper 2007. Available at: http://www.vmware.com/pdf/vc_datasheet.pdf

[Vnuml] NUML Virtual Network User Mode Linux, http://jungla.dit.upm.es/~vnuml/

[VnumlGUI08] "***VNUML Graphic User Interface***", Disponible en: http://pagesperso.erasme.org/michel/vnumlgui (última comprobación, 26 de mayo 2008).

W

[Wang06] T. Wang, C. Wang, F. Lau, "***An architecture to support scalable distributed virtual environment systems on Grid***", The Journal of Supercomputing, Vol. 36, Issue 3, pp. 249 – 264, June 2006.

[Wang08] L. Wang, M. Kunzue, J. Tao, "***From CIM to GLUE: Translate Resource Information of Virtual Machines to Computacional Grids***". GI/TG, KuVS FG Virtualiserung. February 2008.

[WbemServices] WbemServices, [Online] http://wbemservices.sourceforge.net/#Download

[Wirewshark] Wireshark. http://www.wireshark.org/

[Wood08] T. Wood, L. Cherkasova, K. Ozonat, P. Shenoy: ***"Profiling and Modeling Resource Usage of Virtualized Applications"***. In Proc of the 9th ACM/IFIP/USENIX International Conference on Middleware. Pages: 366-387, Leuven, Belgium, September 2008.

[Wlodarz07] Joachim J. Wlodarz, "***Virtualization: A double-edged sword***". Available on: http://arxiv.org/PS_cache/arxiv/pdf/0705/0705.2786v1.pdf. May 2007.

[WSRF08] "***The WS-Resource Framework***", http://www.globus.org/wsrf/ (última comprobación, 26 de mayo 2008).

X

[Xen-CIM] The Xen CIM Project [Online] http://wiki.xensource.com/xenwiki/XenCim.

[XenServer] XenSource, "XenServer Installation Guide", Release 4.0.1, 2007. Available at: http://docs.vmd.citrix.com/XenServer/4.0.1/installation/index.html

Z

[Zamora06] M. C. Zamora, "***Estadística Aplicada***", Editorial Moshera S.R.L, Perú, 2006

[Zhao05] M. Zhao, V. Chadha, R. Figueiredo, "***Supporting application-tailored Grid file system sessions with WSRF-based services***", in Proc. 14th IEEE International Symposium on High Performance Distributed Computing, HPDC-14, , 24-27, July 2005, pp. 24 – 33.

[Zhao06] M. Zhao, J. Zhang, R.J. Figueiredo, "***Distributed File System Virtualization Techniques Supporting On-Demand Virtual Machine Environments for Grid Computing***", Cluster Computing, Volume 9, Issue 1, January 2006, pp. 45 – 56

[Zibitsker09] B, Zibitsker, G. Sigalov, A. Lupersolsky, ***"Modeling and Optimization in Virtualized Multi-Tier Oracle Environments.*** http://bezsys.blogspot.com/2009/03/modeling-and-optimization-in.html. March 21, 2009.

Abreviaturas y acrónimos

A

ADSL *Asymmetric Digital Subscriber Line*, Línea de Abonado digital Asimétrico

AMD *Advanced Micro Devices*, Dispositivos micro avanzados.

ANOVA *Analisys of Variance,* Analisis de la Varianza

API *Application Program Interface,* Interfaz de programa de aplicación.

B

BER *Bit Error Rate,* Tasa de error de bit

C

CBR *Constant Bit Rate, Tasa de Bit Constante.*

CDF *Comulative Distribution Function***,** Función de Distribución Acumulada

CV *Coefficient of Variation,* Coeficiente de Variación

CPU *Central Proccesing Unit***,** Unidad Central de Procesamiento

CIM *Comun Information Model***,** Modelo de Información común

CIMOM *CIM Object Manager*, Gestionador de Objetos CIM

D

DARPA *Defense Advanced Research Projects Agency*, Agencia de Defensa de Proyectos de Investigación Avanzada

DMTF *Distributed Management Task Force,* Fuerza de Tareas de Gestión Distribuida

E

EDIV *Entornos Distribuidos con VNUML*

F

FTP *File Trasfer Protocol,* Protocolo de trasnferencia de Ficheros.

G

GIF *Graphics Interchange Format*, Formato para intercambio gráfico.

GNU GPL *General Public License*, Licenciamiento público general

H

HTTP *HiperText Transfer Protocolo, Protocolo de Transferencia de Hipertexto.*

HTB *Hierarchical Token Bucket,* Jerarquía de cubeta con testigo

I

IETF *Internet Engeneering Task Force*, Fuerza de tares de Ingeniería de Internet

IPPM *IP Performance Metrics*, Métricas del Rendimiento de IP

ITU *Internaional Telecomunicaction Union*, Unión Internacional de Telecomunicaciones**.**

K

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

L

LAN *Local Area Network*, Red de área local

M

MAC *Media Access Control,* Control de Acceso al Medio

MVAN *Virtualization Management Initiative*, Iniciativa de Gestión de Virtualización

N

NAT *Network Address Traslator*, Traductor de direcciones de red.

NIC *Network Interface Card*, Tarjeta interface de red.

NS *Network Simulator*, Simulador de red

NTP *Network Time Protocol*, Protocolo de Temporización de Red

O

OVF *Open virtualization Format,* Formato abierto de virtualización

OWD *One way Delay,* Retardo de una via.

OGSA *Open Grid Services Architecture*, Arquitectura Abierta de Servicios Grid

OASIS *Organization for the Advancement of Structured Information Standards,* Organización para el Avance de las Normas de Información Estructurada

P

PASITO *Plataforma de Analsis de Servicios de Telecomunicaciones*

PCA *Principal component análisis***,** Análisis de Componentes Principales

PCI *Peripheral Component Interconnect,* Interconexión de Componentes Periféricos

Q

QoS *Quality of Service*, Calidad de Servicio

R

RDP *Remote Desktop Protocolo,* Protocolo de Escritorio Remoto

RFC *Request For Comments*, Comentarios para consultas.

RIP *Routing Information Protocol*, Protocolo de Información de Enrutamiento

RM-ODP *Reference Model for Open Distributed Processing,* Modelo de Referencia para Procesamiento Distribuido Abierto.

RTP *Real-time Transport Protocol*, Protocolo de Transporte de Tiempo Real

RTT *Round Trip Delay time,* Tiempo de Retardo de ida y vuelta

S

SAR *System Activity Report,* Reporte de Actividad del Sistema

SCE *Es la suma de cuadrados del error*

SCC *Es la suma de cuadrados de las columnas*

SCT *Es la suma de cuadrados total*

SLA *Service Level Agreement*, Acuerdo de Nivel de Servicio

SNMP *Simple Network Management Protocol*, Protocolo de Administración Simple de Red

SPEC *Standard Performance Evaluation Corporation*, Corporación para evaluación de Estándares de Rendimiento

SPSS *Statistical Package for the Social Sciences*, Programa estadístico informático para Ciencias Sociales

SVPC *System Virtualization, Partitioning and Clustering Working Group,* Grupo de Trabajo de la virtualización del sistema, particionamiento y clústeres.

T

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

TBF *Token Bucket Filter,* Filtro de cubeta con testigos

U

UDP *User Datagram Protocol*, Protocolo de Usuario

UML *User Mode Linux*, Modo de usuario de Linux

UML *Unified Modeling Lenguaje*, Lenguaje unificado de modelamiento

URL *Uniform Resource Locator,* Localizador uniforme de recursos

V

VA *Virtual Appliance,* Aplicación virtual

VBR *Variable Bit Rate***,** Tasa de Bit variable

VCR *Video Cassete Recorder,* Grabador de video casete.

VIF *Virtual Network Interface.* Interface de Red Virtual.

VLS *Video Lan Server*, Servidor de video de Red de área Local

VLC *Video Lan Client*, Cliente de video de red de área local

VNE *Virtual Network Environment*, Entorno Virtual de Red

VoD *Video on Demand*, Video bajo demanda

VPN *Virtual Private Network,* Red Privada Virtual

W

WAN *Wide Area Network*, Red de Area Extensa**.**

WBEM *Web-Based Enterprise Management,* Gestión de Empresa basado en Web

WSDL *Web Service Description Languaje,* Lenguaje de Descripcion de Servicios Web

WSRF *Web Service Resource Framework*, Marco de Trabajo de Recursos de Servcios Web

X

XML *Extended Markup Language*, Lenguaje de Marcación Extendido