Modelo Distribuido para la Gestión de Entornos Virtuales de Red simulando Balanceo de Carga

F. Meneses*, W. Fuertes*, L. Guerra*, J. E. López de Vergara†, H. Aules*
*Departamento de Ciencias de la Computación, Escuela Politécnica del Ejército, Sangolquí, Ecuador
*Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Madrid, España

RESUMEN.- Las tecnologías de Virtualización pretenden aprovechar al máximo los recursos físicos disponibles en las empresas, convirtiéndose en una estrategia de compartición de hardware. Sin embargo, a la hora de configurar y desplegar un entorno virtual de red (VNE, Virtual Network Environment), se presentan problemas de interoperabilidad y escalabilidad. De *interoperabilidad*, ya que existen en la industria una diversidad de plataformas de Virtualización, que provocan que un VNE no pueda ser abstraído de manera que represente a todas las tecnologías de Virtualización subyacentes. De *escalabilidad*, debido a que los VNE normalmente son desplegados en un sólo equipo anfitrión (mono host), provocando dificultad en el despliegue de VNEs complejos. Ante este escenario, la presente investigación se enfoca en el despliegue automático de un VNE en un ambiente distribuido, independiente de la plataforma de Virtualización. Para llevarlo a cabo, se han utilizado técnicas de modelado fundamentadas en el Modelo de Información Común (CIM, Common Information Model) del Grupo de Trabajo para Gestión Distribuida (DMTF, Distributed Management Task Force). Para evaluar la viabilidad del modelo, se ha construido un cliente WBem desarrollando un API, el cual permite desplegar VNEs automáticamente. Además este software facilita la distribución física de un VNE en diferentes servidores con diferentes plataformas, simulando un planificador de balanceo de carga, sin perder el control de ninguna máquina virtual ni del escenario al que pertenecen. Los resultados muestran la solución de los problemas antes mencionados.

Palabras clave.- Balanceo de carga, Entorno Virtual de Red**,** modelado de Información, Modelo de Información común, cliente WBem,

ABSTRACT: Virtualization technologies are intended to maximize the available physical resources in enterprises, becoming a strategy of sharing hardware. However, when configuring and deploying a virtual network environment (VNE), certain problems of interoperability and scalability appear. Interoperability, because there are in the industry a variety of virtualization platforms, which cause that an VNE cannot be abstracted based on the characteristics of the underlying virtualization technology. Scalability, because a virtual network environment are usually is deployed in a single host, causing difficulty in the deployment of complex VNE. According to this brief introduction, this research focuses on the automatic deployment of VNE in a distributed environment, independent of the virtualization platform. To accomplish this, modeling techniques have been used based on the Common Information Model (CIM, Common Information Model) Working Group for Distributed Management (DMTF, Distributed Management Task Force). To evaluate the feasibility of the model a WBem client has been constructed by developing an API, which allows us to deploy a VNE automatically. In addition, this API facilitates the physical

distribution of a VNE on different hosts with distinct platforms, simulating a load balancing scheduler, without lose neither control of any virtual machine where they belong to. The results show the solution of the problems mentioned above.

Key words: Load Balancing, Network Virtual Environment, Information Modeling, Common Information Model, WBem client.

Introducción

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 [1]. Es la técnica que permite configurar el despliegue de diversas máquinas virtuales que son interconectadas entre sí en un entorno centralizado, formando un Entorno Virtual de Red (VNE) [2]. Sin embargo, la mayoría de las herramientas de Virtualización tienen un enfoque centralizado, es decir el despliegue y administración de los recursos en los VNE se realiza sólo en un host físico, lo cual ocasiona baja escalabilidad y dificulta la creación de escenarios complejos [3]. Por otra parte, cuando se incrementa el número de máquinas virtuales en un único host físico, se produce un problema de saturación de hardware, motivo por el cual el crecimiento del VNE debe repartirse o distribuirse en varios equipos anfitriones de manera distribuida.

Ante este escenario, el presente trabajo se enfoca en modelar entornos virtuales de red aplicando criterios de computación distribuida, sustentada en los preceptos del Modelo de Información Común *(CIM, Common Information Model)* [4], de la Fuerza de Trabajo de Gestión Distribuida *(DMTF, Distributed Management Task Force)* que ha estandarizado un conjunto de tecnologías y protocolos para modelar y acceder a los recursos de sistemas computacionales y redes de entornos distribuidos. Por consiguiente, el objetivo de este trabajo es resolver la distribución de un VNE en diferentes equipos anfitriones simulando un balanceo de carga, sin perder el control de ninguna máquina virtual.

En este contexto, la comunidad científica ha mantenido el interés en el despliegue automático de entornos virtuales de red. Así por ejemplo, algunos trabajos, han considerado el modelado y su despliegue en su conjunto, como VNUML [5], Net-Kit [6] o MLN [7]. Sin embargo, estas herramientas están muy ligadas a las plataformas de Virtualización específicas. En un enfoque diferente, Galán et al., en [8], 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) [9] para obtener modelos específicos de pruebas (TSMs) utilizado por las herramientas de gestión específicas de cada test-bed. En cuanto a modelos de máquinas virtuales, en [10][11], los autores exploran cómo recuperar información de recursos en Xen y VMware y luego integran servicios de información Grid. El trabajo propuesto por Fahy et al., [12], detalla los requerimientos de especificación tal como un lenguaje de modelo de información para controlar la Virtualización de los recursos de red y servicios. En relación a estándares DMTF-CIM, el trabajo propuesto por Fuertes et al., en [13], definió un modelo genérico para desplegar entornos virtuales de red en un entorno centralizado, independiente de la plataforma de Virtualización. En este mismo contexto, Meneses et al., en [14], ha modelado entornos virtuales de red, enfocándose en la

distribución de los procesos de construcción y despliegue de dichos escenarios, pero sin balanceo de carga.

Para evaluar la viabilidad de este modelo se ha construido un cliente WBem (Web-Based Enterprise Management) [15], que es una arquitectura de unificación que permite modelar sistemas y la implementación de sus atributos y métodos desde diversas tecnologías subyacentes. Es decir, se ha implementado una interfaz de aplicación (API, Application Program Interface), que permite al software construido para este efecto, desplegar entornos virtuales de red automáticamente, independientemente de la plataforma de Virtualización de base utilizada. Este API se ejecuta sobre el sistema gestionado, llamado CIMOM (CIM Object Manager) [16], que es accedido desde un cliente que monitorea y controla el dispositivo gestionado.

Como contribución, en esta investigación se diseñó e implementó un modelo de extensión DMTF-CIM, que ha permitido la distribución física de un VNE, gestionado en diferentes host anfitriones, simulando un planificador de balanceo de carga, sin perder el control de ninguna máquina virtual. A fin de hacer posible la transferencia de archivos y la ejecución remota, se ha incorporado también a la aplicación cliente API los servicios de Jcraft [17], que son librerías que contiene funciones para transferencia de archivos y ejecución remota.

El resto del artículo ha sido organizado de la siguiente manera: en la Sección 2 se describe el marco teórico. La sección 3 presenta el Diseño e Implementación del modelo y del software que lo valida. En esta sección se detalla además el algoritmo de balanceo de carga utilizado. La sección 4 proporciona la Evaluación de Resultados. En la sección 5, se describe los trabajos relacionados. Finalmente, la sección 6 contiene las Conclusiones y el Trabajo Futuro.

MARCO TEÓRICO

Modelos de Información de Infraestructuras de Red y Sistemas Computacionales

Con el propósito de facilitar la gestión de datos y recursos entre diferentes sistemas computacionales, la DMTF 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 [4]. Estas tecnologías están agrupadas en el concepto de WBem y se basan en el uso del Modelo de Información Común (CIM) 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 [15]. 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, el cual 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 el CIM [16].

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 [4]. El valor de CIM se deriva de su orientación a objetos [15]: 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 [18]: *i)* **Meta esquema**, 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, 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 [19]. 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 pérdida de la información.

Enfoques de modelado de infraestructuras virtualizadas

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 [19]. Sin embargo, las especificaciones de CIM para las redes, servicios, y recursos no consideran el despliegue automático de entornos virtuales de red en su conjunto.

En este contexto, en la industria existen algunos enfoques de modelado, tales como: ***i)*** Perfil de Virtualización DMTF-CIM [20], que define cómo utilizar CIM para la representación de los sistemas virtuales y sus recursos definidos en el Perfil del Sistema Virtual [21]; ***ii)*** Formato Abierto de Virtualización (OVF, *Open Virtualization Format*) [22], que es un estándar abierto para el envasado y distribución de aplicaciones virtuales; y ***iii)*** VMware-CIM [23], 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 [24], que es un conjunto de bibliotecas para la gestión de MVs utilizando el modelo de Virtualización de la DMTF; y, ***ii)*** Xen-CIM [24], para la

definición de modelos de sistema y recursos virtualizados utilizando Xen como plataforma de Virtualización.

El análisis de estos enfoques ha sido la base para proponer el diseño e implementación de un esquema de extensión DMTF-CIM, para el despliegue de un VNE, que será expuesto a continuación:

DISEÑO E IMPLEMENTACIÓN

Requisitos de diseño

Este modelo requiere ser lo suficientemente flexible para responder a los procesos asociados con la construcción automática, despliegue, gestión y publicación de un entorno virtual de red en un ambiente distribuido. Este modelo también debe ser compatible con múltiples plataformas de Virtualización. Para lograrlo, en esta investigación se ha considerado la reutilización del enfoque DMTF-CIM, en lugar de diseñar una jerarquía de clases partiendo desde cero. Parte de este diseño se detallan en nuestros trabajos previos en [13] y [14]. Así mismo, tal como se ha indicado en las secciones anteriores, CIM contiene conceptos comunes y los modelos típicos de los sistemas informáticos y elementos de red que pueden ser reutilizados. A continuación se resumen los principales requerimientos:

- Administración de los entornos virtuales, de una manera autónoma a través del repositorio CIMOM, el cual debe trabajar centralizadamente en un servidor; y las plataformas de Virtualización en otros, los cuales deben estar interconectados, posibilitando que estos funcionen en servidores que pueden estar ubicados en cualquier lugar;
- Asignación automática de direcciones IP para cada máquina virtual;
- El modelo debe ser escalable de manera que cuando se tengan necesidades de nuevas máquinas virtuales, bajo la misma plataforma de Virtualización se pueda desplegar en otros servidores, que tengan recursos disponibles;
- Con el fin de estandarizar nombres y de conseguir una independencia entre la aplicación API-WBem & Jcraft y el repositorio CIMOM, se debe establecer en este último una parametrización determinada;
- Para distribuir físicamente un escenario virtual de red, en diferentes host anfitriones, se debe simular un planificador de balanceo de carga, sin perder el control de ninguna máquina virtual.

Arquitectura básica

La Fig. 1 muestra la arquitectura abstracta en la que se basa el modelo. Se compone de redes con hosts anfitriones interconectados a través de enlaces lógicos y enlaces físicos. En particular, dentro de cada host anfitrión es posible crear e implementar un VNE utilizando una plataforma de Virtualización dada (por ejemplo, Xen o VMware Server). En cada uno de estos VNEs se muestra cómo las máquinas virtuales están interconectadas a través de conexiones virtuales. Cada máquina virtual tiene su propia tarjeta virtual de red (NIC, Network Interface Card) y la dirección IP establecida. Adicionalmente, esta arquitectura permite que uno o más VNEs puedan ser desplegados en la misma máquina cliente. Cabe

señalar que cada máquina virtual tiene una dependencia en el respectivo host dentro del mismo VNE. En contraste, obsérvese el host físico 3, en el cual se ha distribuido un mismo escenario virtual de red en dos host físicos. Para lograrlo, manteniendo el modelo se ha agregado el algoritmo simulando balanceo de carga, mismo que será explicado posteriormente.

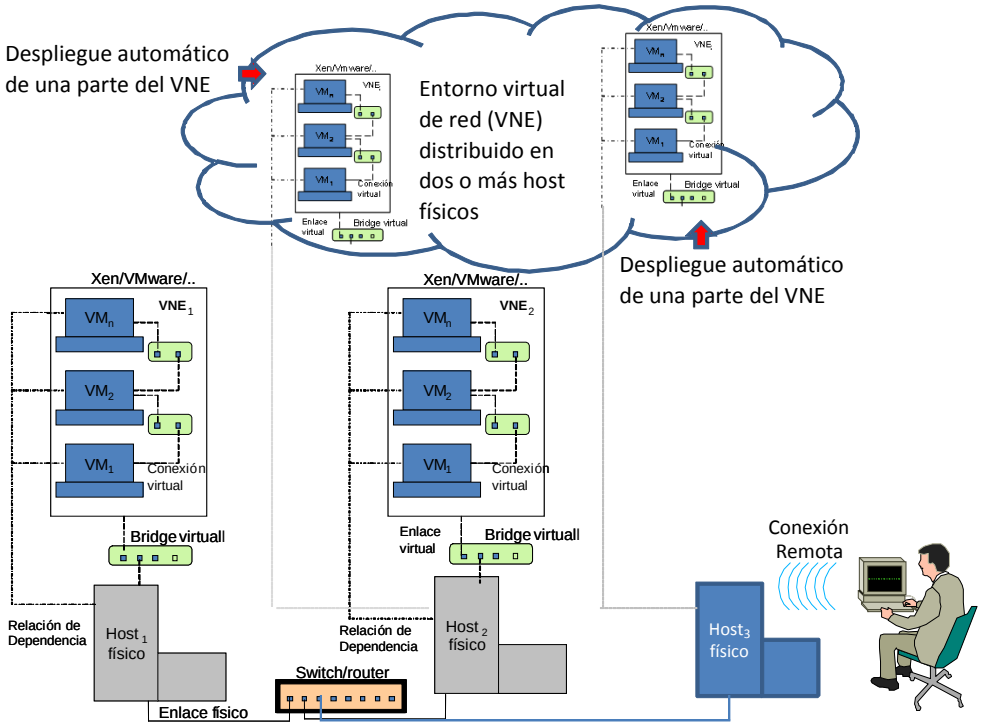

Figura 1. Arquitectura básica para desplegar un VNE en un ambiente distribuido.

Diseño Conceptual para el Modelo Distribuido.

Con el fin de cumplir con los requisitos de diseño y con la arquitectura básica mencionada, la Fig. 2 muestra la solución propuesta para obtener un modelo distribuido que permita gestionar el despliegue automático de un VNE, independientemente de la plataforma de Virtualización subyacente.

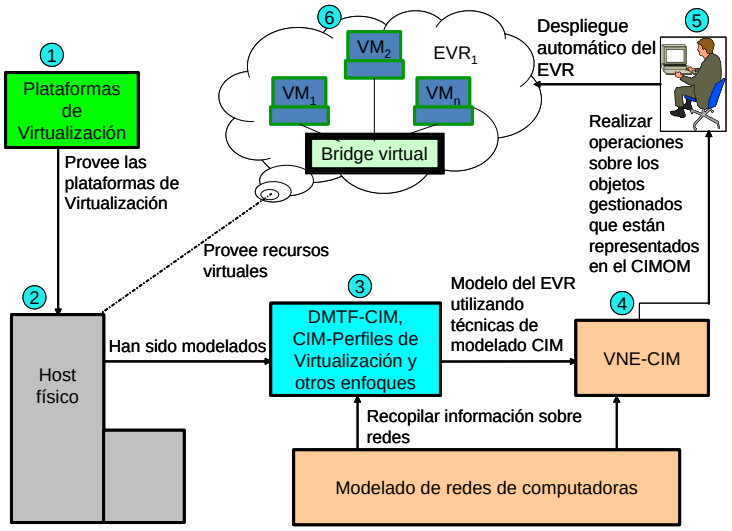

Figura 2. Marco conceptual para desplegar un entorno virtual en un ambiente distribuido

A partir del diseño conceptual propuesto en la Fig. 2, corresponde caracterizarlo en una arquitectura computacional cara a diseñar e implementar el modelo. Como se puede observar, la Fig. 3 es un esquema del modelo genérico propuesto, que debe ser modelado utilizando técnicas UML-CIM que se traducen en un diagrama de clases UML (véase Fig.4). Este diagrama se ha expresado en una descripción formal de nuevas clases del esquema CIM, luego estas fueron compiladas para hacer una validación sintáctica y en seguida fueron almacenadas en el repositorio CIMOM. El diagrama de clases UML se presenta en la Fig. 4.

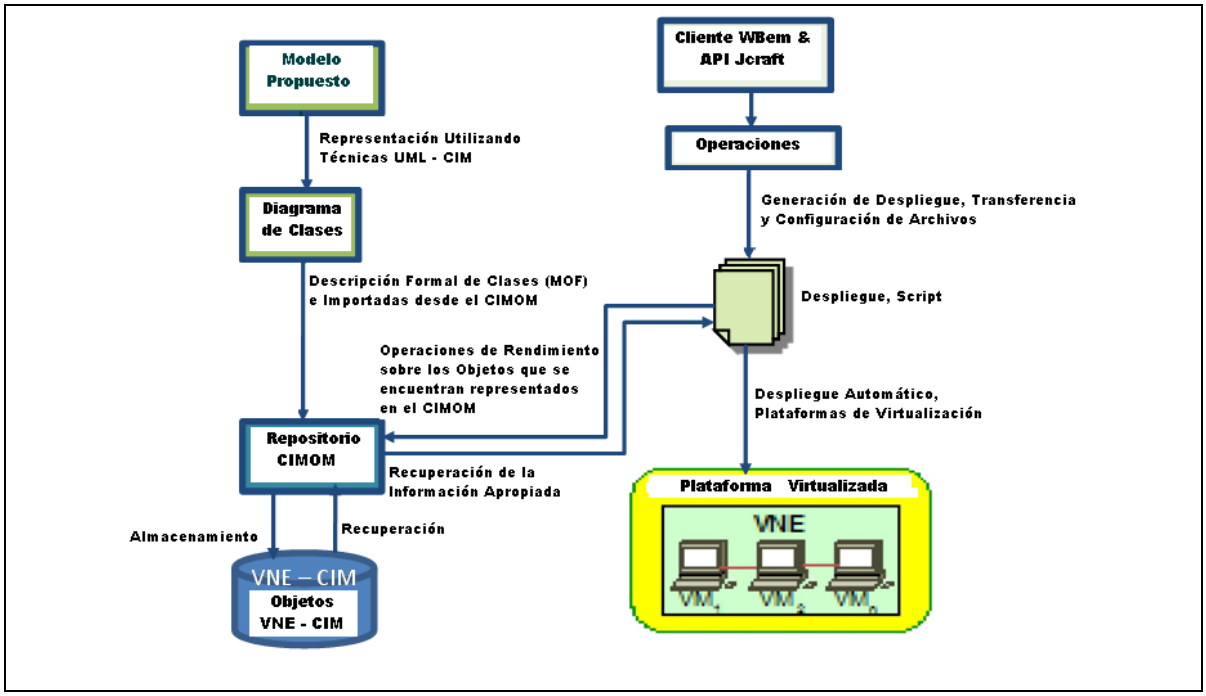

Figura 3. Arquitectura para el diseño e implementación del Modelo Propuesto**.**

Para poner a prueba este modelo, se ha construido una Aplicación Cliente API- WBem, que proporciona una interfaz para acceder y administrar clases y objetos, para realizar operaciones en los objetos administrados que se almacenan en el CIMOM & Jcraft, y para generar de forma automática entornos virtuales de red para diferentes plataformas de Virtualización. Una vez que el VNE se ha importado como un conjunto de instancias de MOF de las clases VNE-CIM, el administrador del API ejecuta una operación de VNE (de clases de Java) para acceder al CIMOM y para generar y transferir los archivos script a los respectivos host físicos, con el fin de producir la operación deseada (por ejemplo, construir y desplegar el VNE). Por lo tanto, la cuestión principal es recuperar las instancias de confirmación de la extensión del modelo (que se describe en la siguiente sección) almacenados en el CIMOM. Para ello, se ha integrado una aplicación de código abierto llamado WBem Services [19], para interactuar con el servidor CIMOM.

Finalmente, para la transferencia de archivos, el despliegue automático de las máquinas virtuales en los diferentes host físicos, su configuración, y la ejecución remota, se incorporan a la aplicación los servicios de Jcraft [20].

En relación a la extensión del esquema CIM resultante del modelo, la Fig. 4 muestra el Diagrama de Clases. Para diferenciarlo se la ha antepuesto el prefijo VNE-CIM:

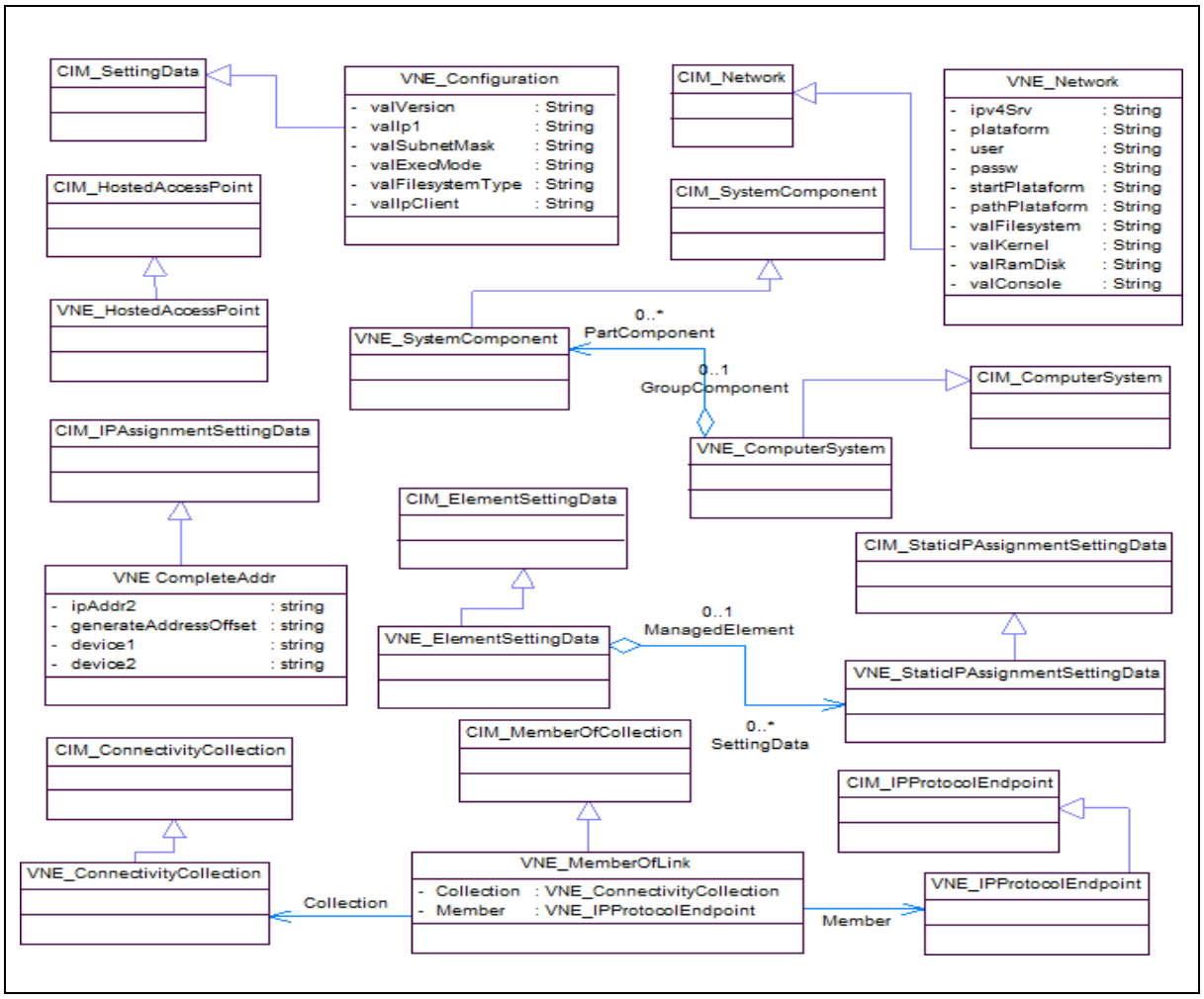

Figura 4. Diagrama de Clases VNE - CIM

Implementación Cliente API WBem & Jcraft

En relación a la implementación de la API-WBem & Jcraft, ésta se compone de las siguientes clases principales (se ha utilizado una vez más el prefijo VNE_):

- Clase *Main*, abre el cuadro de diálogo para que ingrese el usuario al sistema;
- Clase *loginDialog*, se deriva de la clase javax.swing.JDialog; presenta al usuario el cuadro de diálogo para que ingrese la identificación y contraseña; además verifica que se encuentren estos datos en el sistema. Si existen los VNEs, los recupera y activa un objeto de la clase VNE_Man_Start;
- Clase *VNE_Man_LoadBalance*, se conecta al CIMOM, activa objetos de la clase *VNE_ManV2* y accede a las características de las máquinas virtuales de los servidores en Xen; confronta esta información con las de los hipervisores respectivos a fin de determinar las máquinas virtuales disponibles de cada servidor; asigna el escenario de una máquina virtual del servidor que dispone la mayor cantidad de máquinas virtuales; simulando de esta manera el balanceo de carga;

- Clase *VNE_Man_Start*, se conecta al CIMOM, activa objetos de las clases *VNE_ManV2* y *VNE_Man_RemoteExec* y levanta remotamente el VNE;
- Clase *VNE_ManV2*, activa objetos de la clase *VNE_Man_Transfer*, extrae del CIMOM la información pertinente y genera con ella los archivos que levantan las máquinas virtuales y de configuración, seguidamente transfiere estos archivos al servidor que tiene la respectiva plataforma de Virtualización;
- Clase *VNE_Man_Transfer* activa un objeto *VNE_Man_UserInfo*, crea una sesión y un canal de comunicación; se conecta al servidor destino remoto; transfiere el archivo del host fuente al destino; desconecta el canal y la sesión;
- Clase *VNE_Man_RemoteExec*, activa un objeto *VNE_Man_UserInfo*; se conecta al servidor remoto, abre un canal de comunicación, ejecuta el comando remotamente y desconecta el canal y la sesión;
- Clase *VNE_Man_UserInfo*, autentifica al usuario del servidor de Virtualización;
- Clase *VNE_Man_GetEfficiency*, permite medir tiempos de ejecución de algún proceso.

La Fig. 5 representa el diagrama de secuencia de la API WBem - Jcraft, es decir, la interacción entre el usuario y el programa principal, con el objetivo de validar el modelo. El funcionamiento general es el siguiente:

En primer lugar, se solicita una operación de gestión (por ejemplo, el despliegue) (1). Este programa crea un objeto Main de clase Java para abrir un cuadro de diálogo para ingresar al sistema (2). El usuario registra el ID y la contraseña (3). El objeto LoginDialog valida los datos ingresados; si se encuentra autentificado, recupera de la base de datos el escenario que se encuentra autorizado acceder (4). Si el usuario se encuentra autorizado para simular el balanceo de carga (5) (ver figura 6), realiza el proceso respectivo y continua para levantar una máquina virtual en Xen (6). Activa un objeto de la clase VNE_Man_Start (7). Mediante un ciclo para barrer los escenarios recuperados se conecta al repositorio CIMOM, y obtiene la información para crear los archivos SH y de configuración (8, 9, 10, 11, 15 y 18). Luego, transfiere los archivos SH y de configuración desde la aplicación API, hacia el respectivo servidor de Virtualización (13). Ejecuta remotamente los scripts para levantar el entorno virtual (16). Despliega el entorno virtual en la estación cliente (19).

En este punto, conviene mencionar que el programa principal es responsable de crear todos los scripts de implementación y archivos de configuración. También es responsable de llevar a cabo la transferencia de estos archivos, la ejecución remota y el despliegue automático de cada VNE. Esto ha sido probado para Xen y las plataformas de VMware Server. Se ha utilizado tres host físicos, los cuales se describen en la siguiente sección. Para obtener más información, o realizar una prueba de concepto, por favor acceda a los programas fuente en http://dcc.espe.edu.ec/wfuertesd.

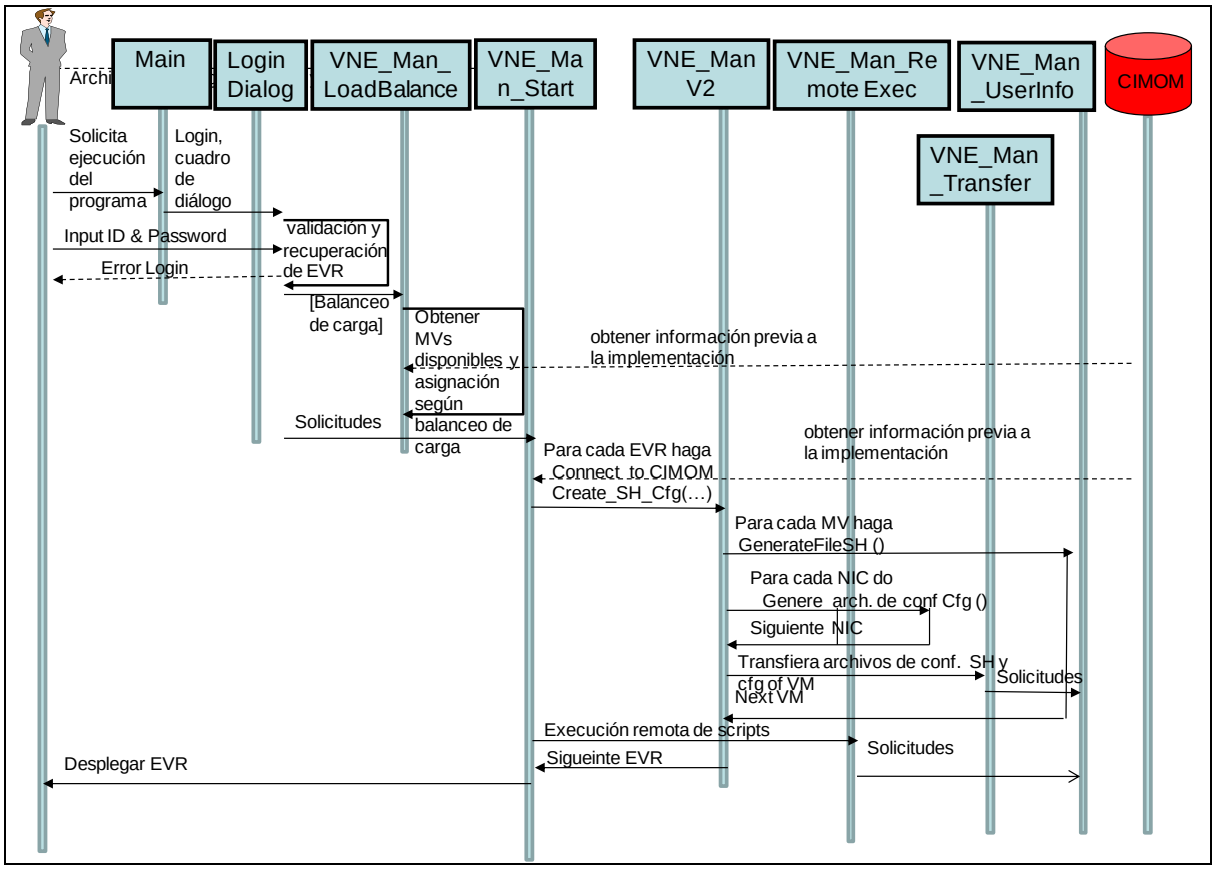

Figura 5 Diagrama de Secuencia de la API WBem - Jcraft

La Fig. 6 representa el diagrama de secuencia del balanceo de carga, es decir, la interacción entre el usuario y el programa principal, con el objetivo de optimizar la asignación de recursos para el despliegue de máquinas virtuales. El funcionamiento general es el siguiente:

En primer lugar, se solicita una operación de gestión (por ejemplo, el despliegue) (1). Este programa crea un objeto Main clase Java para abrir un cuadro de diálogo para ingresar al sistema (2). El usuario registra el ID y la contraseña (3). El objeto LoginDialog valida los datos ingresados; si se encuentra autentificado, recupera de la base de datos la condición de que se encuentra autorizado acceder a balanceo de carga (4). Si el usuario se encuentra autorizado para simular el balanceo de carga (5) , a través de un ciclo para barrer los escenarios que tienen la mayor cantidad de máquinas virtuales en Xen, se conecta al CIMOM y accede a las características de las máquinas virtuales de los respectivos servidores; transfiere el estado de cada hipervisor Xen desde el servidor hacia el cliente API; verifica que la máquina virtual respectiva no se encuentre en el hipervisor a fin de ponerla en la lista de máquinas virtuales disponibles (6); concluido el ciclo, asigna el escenario de una máquina virtual del servidor que dispone la mayor cantidad de máquinas virtuales y continua con el proceso para levantar el escenario virtual correspondiente (7).

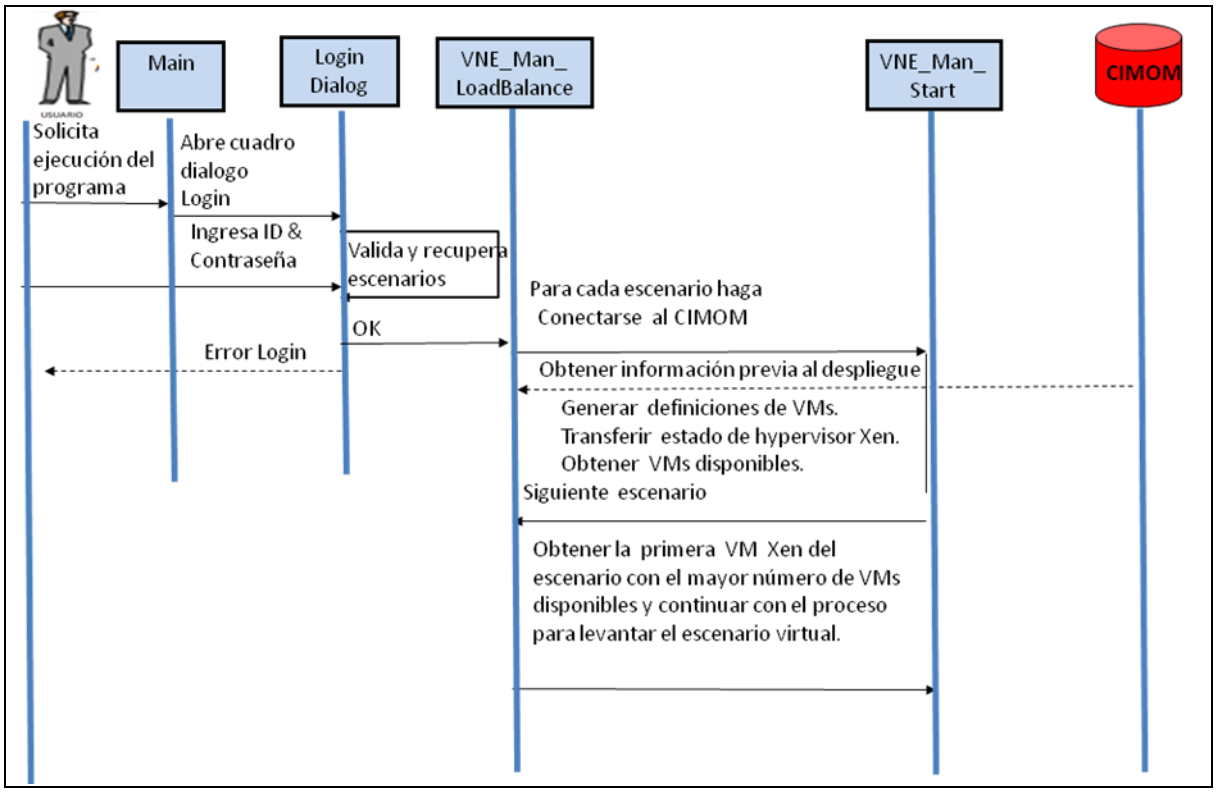

Figura 6 Diagrama de Secuencia del algoritmo de Balanceo de Carga

EVALUACIÓN DE RESULTADOS

Todas las pruebas se llevaron a cabo en tres host físicos con las siguientes características: El primer host físico con procesador Intel ® Core (TM)2 Quad CPU de 2.39 GHz y 3.23 GB de RAM donde se instaló Xen v.3.3 sobre Ubuntu 8.10. El segundo host físico con procesador Intel ® Core (TM)2 Quad CPU de 2.39 GHz y 3.23 GB de RAM donde se instaló VMware v.3.0.0, sobre Ubuntu 9.04. El tercer host físico con procesador Intel ® Pentium Dual a 1,86 GHz y 4GB de RAM, donde se instaló el CIMOM sobre Ubuntu 8.10.

La Tabla 1 muestra los resultados obtenidos al ejecutar el cliente API WBem & JCraft. El *tiempo de ejecución* es el tiempo que se tarda en encontrar la información en el CIMOM, generar el script de implementación, transferir los archivos a los respectivos hosts y ejecutar los scripts para arrancar los VNEs. Mientras que, el *tiempo de despliegue* es el tiempo necesario para ejecutar la secuencia de comandos para desplegar un VNE. Esta tabla muestra que el tiempo de ejecución en VMware (vmw1, vmw2, vmw3) es el es alrededor del 0.2 % en milisegundos en comparación del tiempo de ejecución con máquinas virtuales Xen (vm1, vm2, vm3). En contraste, el tiempo de despliegue con VMware es 8 veces mayor comparado con el tiempo de despliegue de un VNE en Xen. Es importante señalar que el escenario Xen se levanta remotamente en servidores distintos, mientras que VMware en el mismo host. Se puede notar además que las MVs con VMware consumen 10 veces más la CPU y la memoria que las MVs con Xen.

USUARIO	HOST	MAQUINA VIRTUAL	TIEMPO EJECUCION API (ms)	TIEMPO DESPLIEGUE VNE (ms)	CONSUMO CPU	CONSUMO MEMORIA
user 1	10.1.18.71	vm3	16248	5091	0%	0.60%
	10.1.18.60	vmw3	27	40049	11.30%	11.10%
user 2	10.1.18.71	vm1	16277	5091	0%	1.20%
		vm2	17330	5091		
		vm3	17300	5091		
	10.1.18.60	vmw1	23	40049	10.20%	24.00%
		vmw2	49	40049		
		vmw3	163	40049		
user 3	10.1.18.71	vm1	16352	5091	0%	1.20%
		vm2	16361	5091		
		vm3	16373	5091		
	10.1.18.59	vm1	16196	5091	0%	0.10%
		vm2	16247	5091		
	10.1.18.60	vmw1	27	40049	13.60%	29.40%
		vmw2	34	40049		
		vmw3	35	40049		

Tabla 1. Resultados de las pruebas para escenarios distribuidos**.**

Se realizaron también pruebas para simular el balanceo de carga, cuyos resultados se indican en las Tablas 2 y 3. En este caso se ha incluido el *tiempo de transferencia*, que es tiempo que tarda el hipervisor en asignar los recursos necesarios para levantar un VNE, en este caso son Xen.

USUARIO	HOST ASIGNADO	MAQUINA VIRTUAL ASIGNADA	TIEMPO TRANSFERENCIA ESTADO HIPERVISOR XEN (ms)	TIEMPO EJECUCION API (ms)	TIEMPO DESPLIEGUE VNE (ms)	CONSUMO CPU
uxen1	10.1.18.71	vm1	2789	16287	5091	0%
uxen2	10.1.18.59	vm1	1438	16287	5091	0,50%
uxen3	10.1.18.71	vm2	1602	16321	5091	0%
uxen4	10.1.18.59	vm2	1499	21207	5091	1,30%
uxen5	10.1.18.71	vm3	3050	16364	5091	0,00%
uxen6	10.1.18.59	vm3	1482	20240	5091	0,60%

Tabla 2. Resultados de las pruebas para balanceo de carga entre dos servidores**.**

La Tabla 3 muestra los resultados cuando se ha incluido una MV con VMware, donde se demuestra que el tiempo de transferencia es cero, mientras que cuando se usa Xen en un equipo remoto se demora un promedio de 1700 ms en el tiempo de ejecución.

USUARIO	HOST ASIGNADO	MAQUINA VIRTUAL ASIGNADA	TIEMPO TRANSFERENCIA ESTADO HIPERVISOR XEN (ms)	TIEMPO EJECUCION API (ms)	TIEMPO DESPLIEGUE VNE (ms)	CONSUMO CPU
user 1	10.1.18.71	vm3		16468	5091	0,20%
	10.1.18.60	vmw3		19	40049	13%
uxen1	10.1.18.59	vm1	1442	16292	5091	0,85%
uxen2	10.1.18.71	vm1	2390	16360	5091	0%
uxen3	10.1.18.59	vm2	1512	18025	5091	1,24%
uxen4	10.1.18.71	vm2	1540	16494	5091	0%
uxen5	10.1.18.59	vm3	1620	20120	5091	0,80%

Tabla 3. Resultados de las pruebas combinando balanceo de carga y escenarios distribuidos en Xen y VMWare**.**

En este punto conviene resaltar que cuando se produce un pico de demanda de los recursos de las máquinas virtuales, las peticiones totales de recursos pueden exceder los recursos disponibles en un host. Es en este momento que mediante un algoritmo de balanceo de carga que proporciona funcionalidad de redundancia activa – pasiva, se logre que automáticamente se recoloquen las máquinas virtuales en hosts en los que los recursos están disponibles balanceando continuamente la capacidad y asegurando que cada máquina virtual tenga acceso a los recursos apropiados en tiempo real. El balanceo de carga se mantiene gracias al algoritmo que divide de la manera más equitativa el despliegue de las máquinas virtuales, para evitar la saturación del equipo centralizado en el que se distribuye el VNE. Las Figuras 7 y 8 muestran a continuación los resultados obtenidos durante las pruebas realizadas.

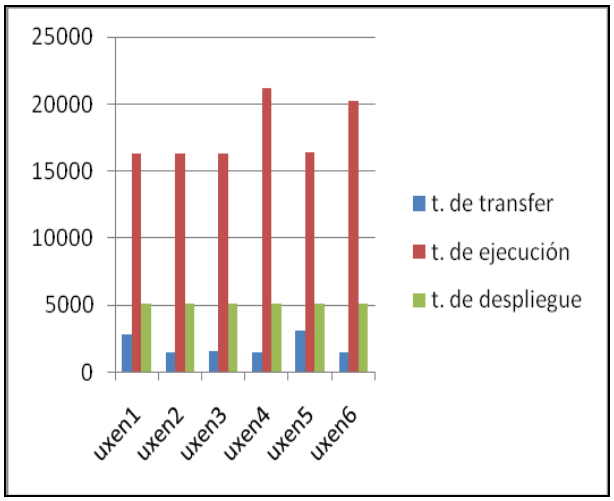

Figura 7. Tiempo de transferencia, ejecución y despliegue de los VNEs

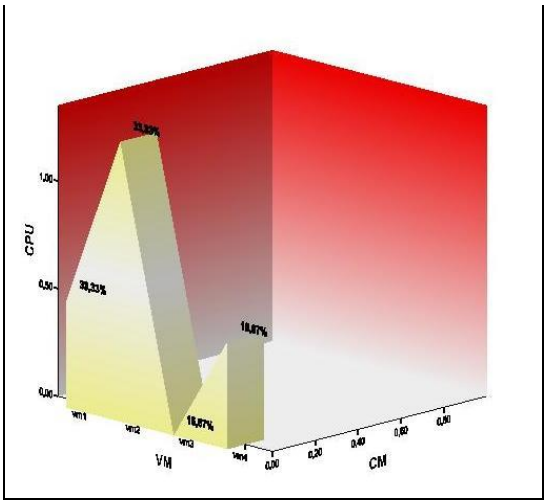

Figura 8. Consumo de CPU durante el despliegue

TRABAJOS RELACIONADOS

Existen algunos trabajos, que han considerado el modelado de un VNE y su despliegue en su conjunto, como VNUML [5], Net-Kit [6] o MLN [7]. 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, VNUML considera 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 [8], describe una metodología de un modelo genérico para especificar escenarios de red (equivalente al concepto VNE) 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) [9] 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 cuanto a modelos de máquinas virtuales, en [10][11], los autores exploran cómo recuperar información de recursos en Xen y VMware y luego integran 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 para recuperar información de MVs y su almacenamiento asociado, que puede ser utilizado en servicios Grid.

Una tercera investigación comparable ha sido descrita por Fahy et al., [12]. 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. 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.

Trabajos más cercanos que consistieron la base de la presente investigación, son los propuestos en primer lugar por Fuertes et al., en [13]. Aquí los autores diseñan e implementan un modelo genérico de extensión CIM, el mismo que ha sido almacenado en un CIMOM y utilizado para el despliegue de escenarios utilizando distintas plataformas de Virtualización, pero sobre un mismo host físico. En segundo lugar, por Meneses et al., en [14], quienes se enfocan en el despliegue automático de un VNE, independiente de la plataforma de Virtualización pero en un ambiente distribuido, mejorando la investigación anterior. Esta última propuesta mantiene el problema de que al crecer el VNE se saturan los host físicos. Comparado con nuestra solución, ninguno de ellos ha incorporado un algoritmo que simule un balanceo de carga cuando el VNE crece en su complejidad.

CONCLUSIONES Y TRABAJO FUTURO

En la presente investigación se ha diseñado un esquema de extensión DMTF-CIM que ha permitido caracterizar un modelo distribuido para desplegar automáticamente entornos virtuales de red, independientemente de la plataforma de Virtualización subyacente, utilizando técnicas de modelado de información fundamentadas en DMTF-CIM e incluyendo un algoritmo que simule balanceo de carga. Para evaluar la viabilidad del modelo, se ha construido un cliente WBem desarrollando un API, el cual permite desplegar VNEs automáticamente. Este software recupera los objetos del repositorio CIMOM, genera los scripts, transfiere los archivos a los respectivos hosts y los ejecuta remotamente; además facilita la distribución física de un VNE en diferentes servidores con diferentes plataformas. Los resultados de las pruebas (evaluadas con Xen y VMware) muestran las soluciones de los problemas mencionados al inicio de esta investigación. En consecuencia, como ventajas de este trabajo alcanzadas se puede mencionar que reduce la complejidad en la construcción y despliegue de escenarios virtuales de red utilizando diferentes plataformas de Virtualización y además aprovecha eficientemente los recursos con el algoritmo de balanceo de carga. Además el cliente WBem implementado sobre la base de este modelo, parcialmente toma el control de los recursos disponibles y permite el despliegue de escenarios virtuales de red en diferentes hosts físicos, independiente de la plataforma de Virtualización y de una manera automática a través de perfiles de usuario.

Como trabajo futuro se incorporará a la aplicación cliente WBem-Jcraft, pistas de auditoría, incorporando varias capas a fin de liberar al cliente del consumo de recursos en el servidor.

AGRADECIMIENTOS

Los autores agradecen al Grupo de Investigación de Sistemas Distribuidos del Dpto. de Ciencias de la Computación de la Escuela Politécnica del Ejército del Ecuador, por la provisión del laboratorio dónde se realizó la experimentación de esta investigación.

REFERENCIAS BIBLIOGRAFICAS

[1] J. Humphreys and T. Grieser, "Mainstreaming Server Virtualization. The Intel Approach". White paper. Sponsored by Intel. June, 2006

[2] W. Fuertes and J. E. López de Vergara, "An emulation of VoD services using virtual network environments", Published in GI/ITG Workshop on Overlay and Network Virtualization NVWS'09, Kassel-Germany, March-2009. Date of acceptance: 09-12-2008. Published in Electronic Communications of the EASST. Volume 17. ISSN 1863-2122.

[3] F. Galán, and D. Fernández, "Distributed Virtualization Scenarios Using VNUML", Proc. Systems and Virtualization Management 2007, Toulouse, France, October 2007.

[4] DMTF, "Common Information Model (CIM) Core Model". Document Number DSP0111. Version 2.4, August 2000.

[5] F. Galán, D. Fernández, W. Fuertes, M. Gómez and J. E. López de Vergara, "Scenario-based virtual network infrastructure management in research and educational test beds with VNUML", Annals of Telecommunications, vol. 64(5), pp. 305-323, May 2009.

[6] M. Pizzonia and M. Rimondini, "NetKit: Easy Emulation of Complex Networks on Inexpensive Hardware", In Proc of TridentCom 2008, Innsbruck (Austria), March 2008.

[7] 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.

[8] F. Galán, J. E. López de Vergara, D. Fernández, R. Muñoz, “A Model-driven Configuration Management Methodology for Networking Test beds Infrastructures”. Proceedings of the Eleventh IEEE/IFIP Network Operations and Management Symposium (NOMS 08), pp. 747-750, April 2008, Salvador, Bahía, Brazil.

[9] Fermín Galán, Jorge E. López de Vergara, David Fernández, Raül Muñoz, “Scenario-based Configuration Management for Flexible Experimentation Infrastructures”, Proc. TridentCom 2009, Washington DC, USA, April 2009.

[10] M. Kunce, L. Wang, “Information Service of Virtual Machine Pool Grid Computing”, Proceedings of Euro-Par 2007 Workshops: Parallel Processing, LNCS 4854, pp. 174-184, Berlin 2008.

[11] L. Wang, M. Kunzue, J. Tao, “From CIM to GLUE: Translate Resource Information of Virtual Machines to Computational Grids”. GI/TG, KuVS FG Virtualiserung. February 2008

[12] C. Fahy et al., “Towars Information Model That Supports Service-Aware, Self-managing Virtual Resources”. MACE 2008, LNCS 5276, pp. 102-107. Berlin 2008.

[13] W. Fuertes, J. E. López de Vergara, F. Meneses, F. Galán, “A Generic Model for the Management of Virtual Network Environments”, In Proc. Of 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan, 19-23 April 2010.

[14] F. Meneses, W. Fuertes, L. Guerra, J. E. López de Vergara y H. Aules, “Modelo Distribuido para la Gestión de Entornos Virtuales de Red”, Publicado en las memorias del VI Congreso de Ciencia y Tecnología ESPE 2011, realizado en Sangolquí, Ecuador del 8 al 10 de junio de 2011. ISSN 1390-4663.

[15] C. Hobbs. “A Practical Approach to WBem/CIM Management”, Published by CRC Press, ISBN 9780203500132, Nov 2, 2004.

[16] WBem Services, Java™ Web Based Enterprise Management, [Online:] http://wbemservices.sourceforge.net/.

[17] Jcraft, [en línea] http://www.jcraft.com/jsch/.

[18] J. E. López de Vergara Méndez, 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.

[19] DMTF, “CIM System Virtualization Model White Paper”. Document Number: DSP2013. Version 1.0.0, November 2007.

[20] DMTF, “CIM System Virtualization Profile”. Document Number: DSP1042. Version: 1.0.0a, August 2007.

[21] DMTF, “CIM Virtual System Profile”. Document Number: DSP1057. Version: 1. May 2007.

[22] DMTF, “Open Virtualization Format Specification”, Document Number: DSP0243, Version: 1.0.0, February 2009.

[23] VMware Inc. “VMware-CIM SDK, Programming Guide”, Version 1.0, http://www.VMware.com/pdf/CIM_SDK_Prog_Guide_esx30.pdf.

[24] Libvirt-CIM provider [Online] http://www.libvirt.org/CIM/.

[25] The Xen CIM Project [Online] http://wiki.xensource.com/xenwiki/XenCim.