[::]

IPv6 para operadores de Red

Alejandro Acosta
Santiago Aggio
Guillermo Cicileo
Tomas Lynch
Antonio M. Moreiras
Mariela Rocha
Arturo Servin
Sofía Silva Berenguer

Internet Society
ISOC-AR
Capítulo
Argentina

[::]
IPv6 para operadores de Red

Alejandro Acosta

Santiago Aggio

Guillermo Cicileo

Tomas Lynch

Antonio M. Moreiras

Mariela Rocha

Arturo Servin

Sofía Silva Berenguer

Internet Society | ISOC-AR Capítulo Argentina

IPv6 para Operadores de Red, 1ª Edición. 2014
Ebook

ISBN 978-987-45725-0-9

Suipacha 128 3° piso F
Ciudad de Buenos Aires, Argentina

Diseño Integral: Transversal Branding

Comité Editor: Christian O'Flaherty y Carlos M. Martínez

::

Agradecimientos

A Internet Society (www.isoc.org.) por haber donado los fondos que han permitido la realización de este Proyecto y su constante apoyo para estimular la continuidad y relevancia de los Capítulos.

A LACNIC (www.lacnic.net) por sus aportes al contenido de este libro así como también por las tareas de capacitación orientadas a la toma de conciencia, que en torno a IPv6 vienen desarrollando en Latinoamérica y Caribe.

A todos los autores y colaboradores que han posibilitado con su dedicación y trabajo la concreción de este Proyecto, que tiene por objeto contribuir a la Comunidad de Internet en la adopción e implementación del nuevo Protocolo IPv6.

La Comisión Directiva
ISOC-AR Capítulo Argentina de Internet Society

::

Indice de contenidos

Autores

Alejandro Acosta

Alejandro Acosta estudió Licenciatura en Computación en la Universidad de Nueva Esparta, Venezuela (1995-2001) y luego obtuvo un master en Gestión de Tecnologías de la Información de la misma universidad.
Actualmente Alejandro es Ingeniero I+D de Lacnic. Anteriormente fue miembro de la Comisión Electoral de LACNIC y presidente de LAC-TF (IPv6 Task Force). Coordina el encuentro anual del Foro Latinoamericano de IPv6 y modera la lista de correo de la IPv6 Latin America Task Force. Es profesor de TCP/IP en la Universidad de Nueva Esparta para estudiantes del noveno semestre.
También ha participado en varios encuentros durante los últimos años incluyendo LACNIC, LACNOG, IGF, LACIGF y encuentros de la IETF. Ha obtenido varias certificaciones, entre ellas la IPv6 Sage Certified (Hurricane Electric, 10 de noviembre) y la Novell Certified Linux Administrator (Novell CLA, febrero de 2010).
También ha participado en artículos para revistas tecnologicas.
Ha sido miembro de Lacnic, del Grupo de Usuarios Linux de Venezuela, IPv6VE y miembro del Capitulo ISOC de Venezuela.

Santiago Aggio

Ingeniero Electrónico especializado en redes de datos y cómputo en áreas científicas y académicas. Actualmente implementa tecnologías de Computación de Alto Desempeño (HPC) mediante clusters y máquinas virtuales, sobre redes IPv4 e IPv6. Ha participado en múltiples proyectos del ámbito académico, desarrollando soluciones de QoS y recientemente implementando sistemas de monitoreo sobre enlaces de Internet y de Redes Avanzadas. Actualmente se desempeña en gestión de redes en la Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca y como Profesional en el CCTBB (Centro Científico Tecnológico Bahía Blanca) dependiente del Conicet en Argentina.

.3

Centros de Datos y Virtualización en IPv6

3.1_
Introducción

El contenido al que accedemos mediante el uso de Internet se encuentra almacenado en máquinas que se alojan en Centros de Datos distribuidos en el mundo. El acceso a este contenido es posible mediante aplicaciones que se comunican con procesos que corren en servidores con sistemas operativos y capacidades de hardware diferentes. Esta diversidad sumada a que el hardware de estas máquinas en general se encontraba sobre-dimensionado para correr pocas tareas, a la alta demanda de brindar servicios de Internet alojados en servidores propios, y al avance de la tecnología de integración de los microprocesadores actuales que presentan extensiones de virtualización, han producido un aumento considerable en el despliegue y uso de las máquinas virtuales.

Una característica que distingue a las máquinas virtuales es que hacen un mejor aprovechamiento del hardware al permitir tener múltiples máquinas corriendo simultáneamente, sirviendo requerimientos de manera independiente, aumentando la capacidad de acceso a recursos y servicios de red, lo que otorga una mayor flexibilidad al momento de migrar servicios y mejora la seguridad al aislar la máquina física y su administración respecto del sistema operativo que ejecuta cada una de las máquinas virtuales.

Las máquinas virtuales presentan diferentes modelos de interfaces de red que funcionan de manera similar a una interfaz física conocida, como es una placa de red ethernet. En cuanto al protocolo IP, en particular IPv4, la mayoría de los paquetes de software disponibles que se encuentran operando, funcionan de forma similar a la implementación disponible en cualquier sistema operativo.

Si tenemos en cuenta, por un lado, el agotamiento del pool de direcciones IPv4, y por otro el crecimiento del número de mecanismos de transición para hacer uso de IPv6, además de las diferentes propuestas que han surgido recientemente sobre la implementación de centros de datos que operan solo con IPv6 en el troncal de su red, resulta imprescindible preguntarnos que grado de soporte y de implementación del protocolo IPv6 presentan las máquinas virtuales.

En este capítulo vamos a describir las diferentes técnicas de virtualización, analizaremos los diferentes modelos de interfaces virtuales y por último consideraremos el nivel de soporte y de implementación del protocolo IPv6 que proveen las máquinas virtuales que podemos utilizar hoy. Además presentaremos los comandos y pasos necesarios para configurar IPv6 en máquinas virtuales.

3.2
Soporte de Virtualización en Procesadores

Aunque la mayoría de los procesadores actuales presentan las extensiones para virtualización, es recomendable antes de instalar cualquier paquete de software de virtualización en Linux, verificar si el procesador tiene capacidad de virtualización completa. Para esto podemos usar el siguiente comando:

```
# egrep '(vmx|svm)' --color=always /proc/cpuinfo
```

La respuesta puede contener la sigla vmx (Intel), svm (AMD) o no devolver nada en el caso de no ser compatible para virtualización.

3.3_
Tipos de virtualización

La tecnología de virtualización[1] permite disponer de múltiples máquinas corriendo en paralelo a partir de un único hardware. Estas múltiples máquinas son virtuales y su nivel de virtualización puede establecerse a partir del hardware mismo, mediante un supervisor que trabaja a modo de capa entre el hardware y el sistema operativo, o a nivel del sistema operativo sobre el que corren múltiples servidores virtuales independientes. En este aspecto podemos hacer una analogía entre virtualización y un sistema multitarea, en donde tenemos corriendo varios procesos simultáneamente sobre un único sistema operativo.

Las arquitecturas de virtualización cuentan en general con un componente que media entre el hardware y el sistema operativo huésped que se denomina supervisor o Monitor de Máquina Virtual (MMV) que se encarga de traducir el código binario, controlar la ejecución y administrar el acceso a los dispositivos y a diferentes recursos del hardware.

A modo de resumen, los 4 tipos de arquitecturas de virtualización que se describen a continuación y se muestran en las Figuras 1, 2, 3 y 4, son las disponibles hoy en diferentes implementaciones y bajo diferentes sistemas operativos (SO).

3.3.1. Emulación

- Emulador de hardware.
- Simula el hardware requerido mediante una Máquina Virtual (MV).
- Ejecuta cualquier SO nativo, sin modificación.

- El SO no advierte que usa un hardware ficticio.
- Ej: QEMU, Parallels, Microsoft Virtual Server.

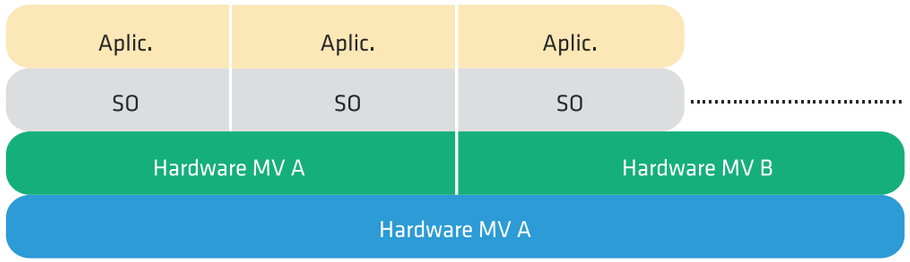

FIGURA 1: EMULACIÓN

3.3.2. Virtualización nativa o completa (Full)

- Máquina Virtual que media entre el SO huésped y el hardware nativo.
- El hardware y los recursos son compartidos y controlados por el supervisor o Monitor de Máquina Virtual (MMV)
- Es mas rápido que emulación, pero su desempeño se ve afectado debido a la intermediación del Monitor
- El SO no requiere ser modificado, pero debe soportar la arquitectura sobre la que corre.
- Ejemplo: VMware, z/VM (IBM), Linux KVM (Kernel Virtual Machine).

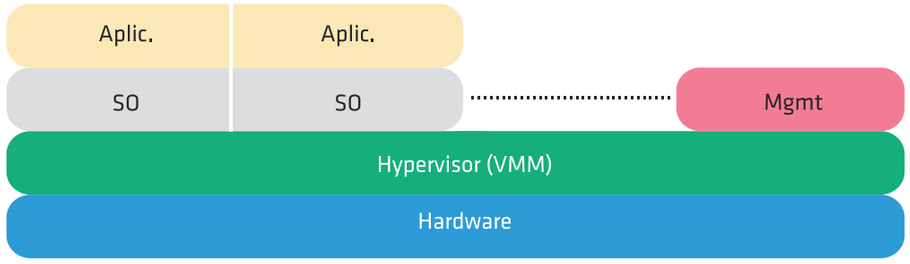

FIGURA 2: VIRTUALIZACIÓN NATIVA O COMPLETA

3.3.3. Para virtualización

- Similar a virtualización completa
- Comparte procesos con el supervisor
- Requiere re-compilar o portar el SO huésped para interactuar con el MMV.
- Soporta múltiples SO simultáneamente.
- Desempeño similar a un sistema no virtualizado.
- Ejemplo: Xen, UML (User Mode Linux)

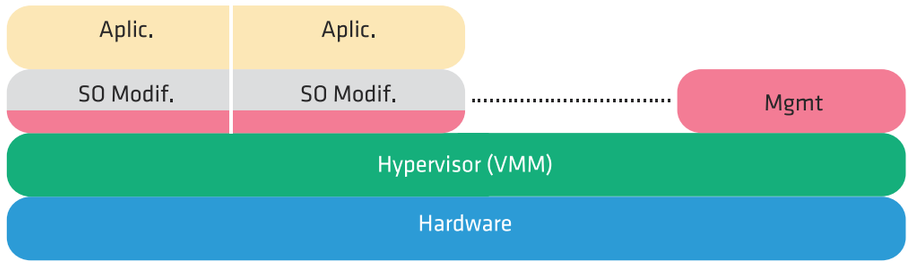

FIGURA 3: PARAVIRTUALIZACIÓN

3.3.4. Virtualización a nivel del sistema operativo

- Virtualiza servidores sobre el kernel de un SO.
- Divide un servidor físico (SF) en múltiples servidores virtuales (SV).
- Cada SV se ve y se comporta como un SF.
- Se pueden ejecutar múltiples copias de un OS (con distintas versiones) sobre un mismo SF.
- Ejemplo: OpenVZ, Virtuozzo, Linux-VServer, Solaris Zones, FreeBSD Jails.

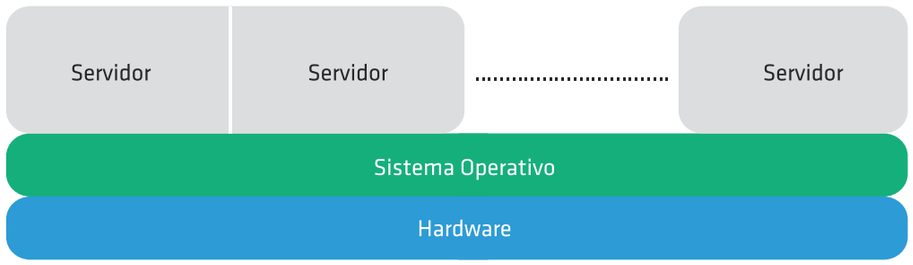

FIGURA 4: VIRTUALIZACIÓN A NIVEL DEL SO

En la Tabla 1. se muestra el soporte que presentan las distribuciones de Linux para con los diferentes paquetes de virtualización de código abierto.

Paquete	Distribución de Linux con soporte
Xen	RedHat 5.x, CentOS 5.x, Arch, Alpine, Debian, Fedora, Finnix, Gentoo, OracleLinux, OpenSuSE, Ubuntu
KVM	RedHat REHL 5.4 o superior, Ubuntu LTS 10.0.4 o superior, openSuSE SLES 11 SP1 o superior
OpenVZ	REHL 6 kernel 2.6.32., REHL 5 kernel 2.6.18. Plantillas oficiales para: CentOS 5 y 6, Fedora 17 y 18, Debian 6.0, Scientific 6, Suse 12.1, 12.2 y 12.3, Ubuntu 8.04, 10.04, 11.10, 12.04 y 12.10

TABLA 1: VIRTUALIZACIÓN SOPORTADA POR DISTRIBUCIÓN DE LINUX

El crecimiento en la adopción de KVM que se observa en la Tabla 1, se debe en parte a que recientemente las empresas HP, Intel, IBM y Red Hat, fundaron la Open Virtualization Alliance (OVA) para establecer un consorcio que actualmente cuenta con cientos de miembros. El objetivo de esta alianza es posicionar a KVM como una opción de virtualización de código abierto que sea rentable y de importancia estratégica para las empresas y proveedores de servicios de computación en la nube. Esta estrategia ha llevado a que, por ejemplo, RedHat se haya volcado a KVM en su distribución versión 6 y haya dejado de incluir a Xen, que fuera adquirido por la empresa Citrix, quién mantiene el proyecto de código abierto junto a versiones comerciales.

3.4_
Modos de virtualización de red

En esta sección describiremos los diferentes modos de virtualización a nivel de red que presentan las implementaciones de Xen, OpenVZ y KVM.

3.4.1. Xen

El principal componente de Xen[2] es el supervisor, quién se aloja entre el hardware y los sistemas operativos huéspedes. El supervisor es responsable de aislar y proteger el sistema, controlando el acceso y la asignación de recursos, además de diagramar la porción de máquina física asignada a cada huésped.

Xen permite albergar múltiples sistemas operativos que reciben el nombre de dominios (Dom). Estos dominios son planificados por el supervisor para hacer uso real de las CPUs físicas disponibles. A su vez cada SO maneja sus propias aplicaciones.

Al iniciar el sistema con el kernel de Xen, este crea el dominio Dom0, que tiene privilegios para el manejo del resto de los dominios y el acceso a los dispositivos virtuales. El resto de los dominios se denominan domU, siendo U un índice de valor 1 a N. Dentro Dom0 el proceso xend es quién se comunica con el supervisor para manejar las máquinas virtuales y permitir el acceso a sus consolas.

Xen puede compartir la interfaz de red física entre múltiples dominios, permitiendo que cada domU pueda tener una o más interfaces de red virtuales[3]. Xen combina los paquetes salientes de cada interfaz de red virtual del respectivo domU sobre la interfaz de red física. Del mismo modo, Xen separa cada paquete que entra por la placa de red física destinado a la interfaz virtual de cada domU activo.

Xen presenta tres modos de configurar las interfaces virtuales permitiendo crear diferentes arquitecturas de red de acuerdo a la cantidad de placas de red disponibles en el servidor, a las direcciones IP y subredes a asignar de forma manual o automática, a los servicios corriendo sobre

la interfaz virtual y que no tienen acceso directo por la interfaz física, etc. De los tres modos que se describen a continuación y son válidos para usar IPv4, solo se tendrán en cuenta los dos primeros en los ejemplos de configuración del protocolo IPv6.

3.4.1.1. Modo bridge

En este modo el tráfico entre interfaces es a nivel de capa 2 teniendo en cuenta solo las direcciones Físicas (MAC) e independizándose de las capas superiores. Este modo es el recomendado para usar en Xen por ser más simple en cuanto a su funcionamiento y configuración. Las direcciones MAC son visibles sobre la interfaz física y en el segmento de red ethernet a la que está conectada.

3.4.1.2. Modo router

En modo router, los paquetes son enviados entre las diferentes IP asignadas a las interfaces físicas y virtuales. Estas direcciones IP asignadas son visibles desde la red ethernet local, no así sus direcciones MAC. Estas direcciones IP son resueltas por ARP a la dirección MAC de la interfaz física, sumando la funcionalidad de proxy-arp para interfaces virtuales.

3.4.1.3. Modo NAT

En modo NAT, Xen funciona de manera similar al modo router, con la diferencia de que sus direcciones IP no son visibles desde el exterior. La diferencia está en la asignación de direcciones IP entre la interfaz del respectivo domU y la dirección IP asignada a la interfaz virtual del Dom0. Dada una clase C, El domU utiliza en su interfaz el rango de IPs desde la .2 a .127, entonces cada interfaz virtual del Dom0 utiliza el rango de .128 a 254 (+127) en concordancia con el domU. Por ejemplo, si configuramos para el domU la subred 10.0.0.2/24 con puerta de enlace 10.0.0.1, la interfaz virtual del Dom0 se auto-asigna la dirección IP 10.0.0.129.

3.4.2. OpenVZ

OpenVZ[3] es un sistema de virtualización a nivel del sistema operativo y presenta un kernel de Linux modificado. Además de la virtualización, OpenVZ presenta funcionalidades que lo destacan, como son la aislación, el manejo de recursos (mediante el uso de cuotas de disco, porciones de tiempo de ejecución en la CPU y contadores de recursos del kernel), y el establecimiento de puntos de control de chequeo, lo que facilita la migración en caliente de un contenedor a otra máquina, guardando previamente el estado completo de la MV.

Cada MV es creada mediante la instalación de plantillas que pueden bajarse de su sitio y están disponibles para diferentes versiones y distribuciones de Linux.

En cuanto a los modos de virtualización de red, OpenVZ presenta dos tipos de interfaces de red bien diferenciadas, que se distinguen por el nivel de seguridad y de acceso a su configuración (por parte del usuario y del administrador). Estas interfaces son:

- Virtual Ethernet device (veth)
- Virtual Network device (venet)

Virtual Ethernet (veth) es un dispositivo que provee funcionalidad en capa 2 y puede utilizarse dentro del contenedor mediante la asignación de una dirección física MAC. Se comporta como un dispositivo ethernet real Virtual Network (venet) es el dispositivo de red que presenta OpenVZ por omisión cuando instalamos un nuevo contenedor. Es un dispositivo que funciona en capa 3 y se comporta como una conexión punto a punto entre el contenedor y el servidor físico. Este dispositivo es el más seguro en cuanto al nivel de aislación, pero presenta algunas limitaciones en cuanto a su funcionalidad y administración.

En resumen, las diferencias entre las interfaces virtuales que provee OpenVZ se muestran en la siguiente tabla:

Características	Veth	Venet
Dirección MAC	Si	No
Broadcast dentro del CT	Si	No
Captura de tráfico	Si	No
Seguridad de red	Baja: Independiente del anfitrión y controlada en cada CT	Alta: controlada por el anfitrión
Utilizada en Bridge	Si	No
Soporta IPv6	Si	Si (no completa)
Desempeño	Rápido	Más rápido y más eficiente

TABLA 2: INTERFACES VIRTUALES EN OPENVZ

En la sección de configuración de IPv6 en OpenVZ utilizaremos la interfaz virtual veth que soporta IPv6 de forma completa.

3.4.3. KVM

KVM[4] es una tecnología que agrega capacidad de virtualización al kernel de Linux. En Linux un proceso tiene 2 modos de ejecución: modo kernel y modo usuario. Lo novedoso de KVM es que agrega un modo más de ejecución que se denomina huésped. La arquitectura que presenta KVM se muestra en la Figura 5.

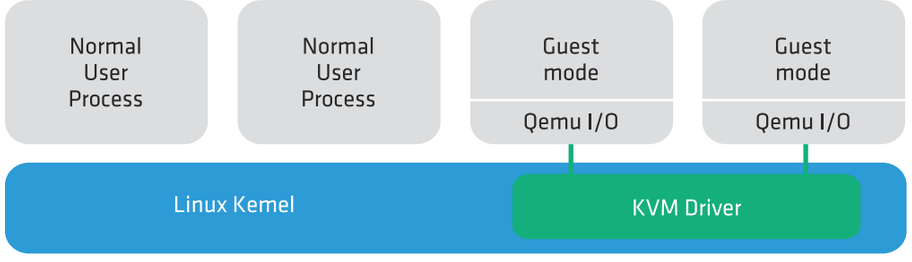

FIGURA 5: ARQUITECTURA Y MODOS DE EJECUCIÓN EN KVM

Las distribuciones de Linux RedHat y CentOS 6 proveen de manera nativa y de base el soporte y las herramientas para usar KVM como medio de virtualización. Esto puede verificarse ejecutando:

```
# yum grouplist | grep -i virt
   Virtualization
   Virtualization Client
   Virtualization Tools
   Virtualization Platform
```

Los paquetes contenidos en estos grupos pueden listarse ejecutando, por ejemplo, para el grupo Virtualization:

```
# yum -q groupinfo “Virtualization”
Group: Virtualization
 Description: Provides an environment for hosting virtualized guests.
 Mandatory Packages:
   qemu-kvm
 Default Packages:
   hypervkvpd
 Optional Packages:
   qemu-guest-agent
   qemu-kvm-toolsyum
```

KVM utiliza dos modos para configurar la red: modo bridge y modo usuario. El modo usuario es el modo por omisión y se basa en un modelo de virtualización de red que utiliza NAT. En la sección 5.3 de configuración de IPv6 en KVM veremos el caso para modo bridge.

3.4.4. Proxmox

Proxmox es un gestor para virtualización de servidores virtuales basado en código abierto. Se basa en OpenVZ y KVM, por lo que permite soportar Windows y Linux. Proxmox es miembro de la Open Virtualization Alliance.

Para su administración presenta un interfaz web muy amigable desde donde se crean, controlan y administran las MV. Estas MV se crean teniendo en cuenta que tecnología de virtualización se selecciona: si funcionan bajo el supervisor en el caso de KVM, se denominan máquinas virtuales (VM) o pueden crearse como huéspedes dentro de OpenVZ y se denominan contenedores (CT). Los módulos del kernel de OpenVZ y KVM se encuentran activos y se verifican ejecutando:


```
# lsmod | grep ^vz
vzethdev                8189  0
vznetdev               19230  5
vzrst                 188071  0
vzcpt                 142549  1 vzrst
vzdquota               56321  4
vzmon                  25335  7 vzcpt,vzrst,vznetdev
vzdev                   2765  6 vzmon,vzdquota,vznetdev,vzethdev
vzevent                 2178  1
lsmod | grep ^kvm
kvm_intel              51799  0
kvm                   321061  1 kvm_intel
```

Las versiones más recientes de Proxmox se basan en Debian GNU/Linux 7.0 sobre un kernel 2.6.32 modificado.

3.4.5. VMware

VMware[5] es el producto comercial de virtualización más difundido y adoptado por los proveedores de contenido y de servicios de computación en la nube. Esta adopción se debe en parte a su interfaz de administración gráfica que resulta muy amigable para quién comienza a experimentar con la tecnología de virtualización y a la robustez que presenta el supervisor a las MV que se ejecutan.

VMware ofrece versiones de algunos de sus productos sin cargo, que pueden bajarse de su sitio y utilizarse luego de ser registrado. Estos productos presentan limitaciones en cuanto a funcionalidad y al número de máquinas virtuales soportadas, pero resultan interesantes para evaluar el producto y para experimentar en ambientes de uso personal o de bajo requerimiento.

3.5_
Implementación de IPv6 en máquinas virtuales

A pesar que el protocolo IPv6 está ampliamente difundido y ya ha sido desplegado en diferentes ámbitos y tipos de redes, los libros de referencia de máquinas virtuales y los manuales de los diferentes paquetes disponibles de código abierto o comerciales, no contienen suficiente información sobre la configuración de IPv6 en máquinas virtuales, comparado con la cobertura que estos presentan para IPv4.

Se podría suponer que el soporte para IPv6 que brindan estos productos aún no ha alcanzado la suficiente madurez o que el requerimiento, de parte de los clientes para con los proveedores de contenido y servicios, sigue siendo mayoritariamente sobre IPv4. El objetivo de las próximas secciones es revertir esta impresión e incentivar el uso y la implementación de IPv6 en ambientes virtualizados sobre Linux.

Antes de comenzar con la configuración de IPv6 en la interfaz de red virtual de una MV, describiremos en esta sección algunas de las limitaciones que pueden presentarse en la conectividad y en el kernel en escenarios simples de conexión, donde tenemos máquinas virtuales, un switch y un router IPv6.

3.5.1. Port Security en Cisco

Los Centros de Datos han incorporado diferentes medidas de seguridad que se aplican a partir del mismo puerto físico al que se conecta un cliente. Una posibilidad es aplicar la facilidad port security que proveen los switches Cisco en sus interfaces, lo que restringe el número de direcciones MAC permitidas sobre el puerto. En este escenario que se muestra en la Figura 6 no es posible el uso de máquinas virtuales en modo bridge, debido a que no son visibles las direcciones MAC de las interfaces virtuales sobre el segmento de red ethernet que conecta al Dom0 con el switch físico. La facilidad port security se configura por puerto y permite, si se especifica, establecer un número máximo de direcciones MAC seguras o especificar cada dirección MAC a conectar a dicho puerto, aunque esta última opción no es la más deseable.

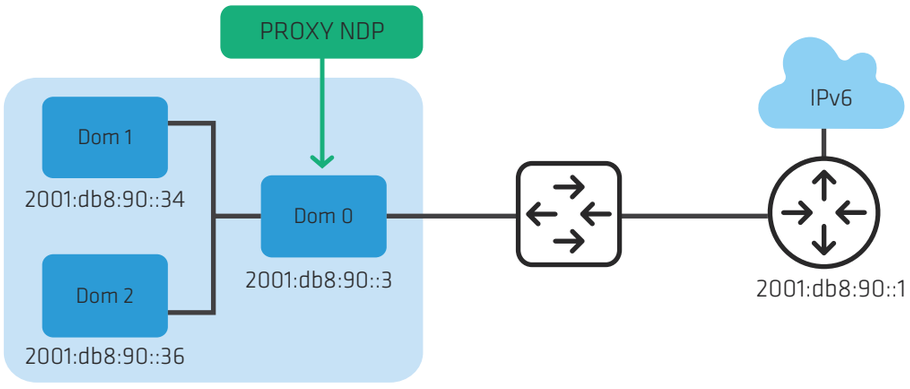

FIGURA 6: ESCENARIO PORT SECURITY

Para implementar IPv6 en ambientes virtualizados con soporte para auto-configuración es recomendable que las MV puedan operar en modo bridge, para lo cual es necesario desactivar la limitación que impone port security. Si esto no es posible, es necesario utilizar el modo router, para lo cual el kernel debe soportar la funcionalidad NDP que se describe a continuación y que facilita la configuración y el uso en IPv6.

3.5.2. Neighbor Discovery Proxy

En la Figura 7 se muestra la arquitectura de conectividad al utilizar Xen en modo router. Al crear las máquinas virtuales bajo este modo, se requiere que el Dom0 actúe para el protocolo IPv6 como un Neighboor Discovery Proxy, cumpliendo con las recomendaciones del RFC4389.

La funcionalidad de Neighbor Discovery Proxies (ND Proxy) descripta en el RFC4389[6] recomienda en primer lugar y de ser posible utilizar la tecnología de bridge a nivel de enlace. Sin embargo, esta tecnología basada en la solución IEEE 802.1D no siempre es aplicable. El RFC 4389 describe dos posibles escenarios de uso, estos son wireless upstream y PPP upstream, pero también es posible aplicarlo en otros escenarios, como es nuestro caso.

Cuando un equipo requiere conectarse a una máquina virtual, este genera paquetes ND tipo multicast (ff02::1) que llegan desde el exterior en primera instancia al dom0. Este debe actuar como un intermediario para el protocolo ND y reenviar estos paquetes dentro de la red virtual que compone el Dom0 y las máquinas virtuales. Las respuestas generadas por la máquina virtual deben a su vez ser transmitidas por el Dom0 sobre la red exterior para que las mismas alcancen al respectivo solicitante.
De los 5 tipos de paquete ICMPv6 que se definen en el RFC 2461[7], existen 2 tipos que nos interesa analizar y se relacionan con la identificación de vecinos:

Solicitud de Vecino (Neighbor Solicitation) – generado por los nodos para determinar la dirección en la capa de enlace de sus vecinos, para verificar que el nodo vecino sigue activo (es alcanzable), y para detectar las direcciones duplicadas. Tipo en paquete ICMPv6 = 135.

Anunciación de Vecino (Neighbor Advertisement) – generado por los nodos como respuesta a la "solicitud de vecino", o para indicar cambios de direcciones en la capa de enlace. Tipo en paquete ICMPv6 = 136.

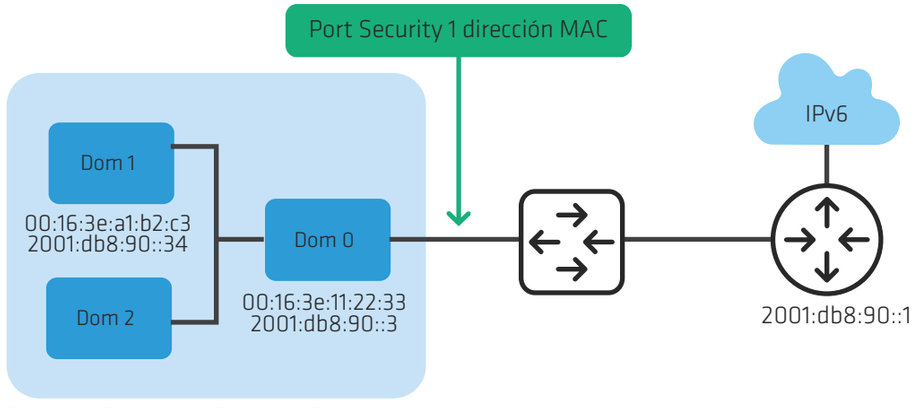

FIGURA 7: ESCENARIO PROXY NDP

A diferencia de la funcionalidad proxy_arp bajo IPv4, que se encuentra disponible en los kernels desde hace tiempo, la funcionalidad proxy_ndp esta disponible a partir de la versión del kernel 2.6.19[8].

Estas funcionalidades que se agregan en el kernel 2.6.19 introducen una Neighbor Cache para cada interfaz donde se almacenan la dirección

IPv6 y su MAC asociada de vecinos dentro del segmento (router, Dom0, domU, etc), y permite además mantener el estado de asignaciones de direcciones IPv6 al utilizar auto-configuración cuando llegan los anuncios de prefijos IPv6 por parte del router. Sin la funcionalidad de proxy_ndp, es necesario agregar en el router la dirección IPv6 y su MAC en la interface, agregar la dirección IPv6 de cada MV y la ruta destino de cada MV en el supervisor. En resumen, este es un ejemplo de los pasos necesarios para la configuración manual antes descripta:

- En el router (2001:db8:90::1)
  ipv6 neighbor 2001:db8:90::34 GigabitEthernet5/1.88 0014.4f8d.e352
- En el Dom0 (2001:db8:90::3)

```
# ip -6 addr add 2001:db8:90::34/64 dev eth0
# ip -6 route add default gw 2001:db8:90::1 dev eth0
```

Para obtener la interfaz virtual de la MV dom1 ejecutamos

```
# xm network-list dom1
```

Luego agregamos la ruta a la MV dom1, su interfaz virtual y la dirección fuente

```
# ip -6 route add 2001:db8:90::34 dev vif11.0 src 2001:db8:90::3
```

- En el dom1 (2001:db8:90::34)

```
# ip -6 neigh add 2001:db8:90::1 lladdr fe:ff:ff:ff:ff:ff dev eth0
# ip -6 neigh add 2001:db8:90::3 lladdr fe:ff:ff:ff:ff:ff dev eth0
# ip -6 neigh show
```

3.5.3. Parámetros del kernel de Linux para IPv6

Cuando utilizamos una tecnología de virtualización en Linux, el kernel resulta la pieza fundamental en el funcionamiento del supervisor o nodo principal y las MV. En la sección 4.2 vimos algunas limitaciones que pueden presentarse por faltantes de soporte de IPv6 en la versión del kernel.

En[9] se enumeran las variables y parámetros del kernel que se configuran para IPv6. Estas variables definen el comportamiento de IPv6 en los diferentes niveles de la capa de red y permiten modificar el estado y el control de diferentes parámetros que afectan su comportamiento, como son la activación del protocolo, la fragmentación y ensamblado de los paquetes IPv6, el reenvío (forwarding) de paquetes entre interfaces, el valor del MTU, la auto-configuración aceptando el prefijo anunciado, la detección de duplicados, entre otros.

Como veremos en los puntos de la Sección 3.3.5. algunas de estas variables influyen en el funcionamiento de las MV en IPv6 y es necesario cambiar su valor para habilitar funcionalidades requeridas y para desactivar otras que lo afectan. Es importante tener en cuenta que algunos de estos cambios en estas variables se realizan automáticamente cuando activamos una MV porque están contemplados en los archivos de configuración cuando ejecutamos un script o activamos una interfaz o servicio, pero otros deberán ser realizados por el administrador del supervisor o nodo principal.

3.5.4. Radvd

Router Advertisement Daemon (radvd) es un demonio que anuncia direcciones y rutas IPv6 sobre la red local y permite asignar direcciones IPv6. Este demonio envía periódicamente mensajes de anuncio definidos en el RFC 2461 ya descriptos en el punto 3.4.2 y recibe los mensajes de solicitud de vecinos para finalmente asignar una dirección IPv6 a un nodo en un modo de configuración automática sin estado.

Radvd resulta interesante de usar en escenarios virtualizados en los cuales no es posible definir un bridge, la funcionalidad de proxy_arp no esta disponible o los mensajes del protocolo ND no llegan a los nodos. En estas situaciones es posible utilizar el nodo principal o el supervisor como un router para anunciar los prefijos y direcciones IPv6 a las MV.

La configuración del demonio radvd se define en Linux en el archivo /etc/radvd.conf. Los parámetros más importantes a definir son la interfaz donde escucha el demonio y sobre el que emite los mensajes de anuncio, el prefijo IPv6, el tiempo de vida del prefijo, y su frecuencia de envío.

Un ejemplo de configuración es el siguiente:

```
interface eth0 {
       AdvSendAdvert on;
        MinRtrAdvInterval 3;
        MaxRtrAdvInterval 10;
        prefix 2001:db8:90::/64 {
               AdvOnLink on;
               AdvAutonomous on;
               AdvRouterAddr on;
        };
};
```

Es importante notar que radvd no anuncia parámetros de configuración como podría ser la dirección IPv6 de servidores de nombre de dominio (DNS), de un servidor WINS o de un servidor TFTP para equipos que requieren transferir archivos de configuración al iniciarse. Para estos casos es necesario implementar un servidor DHCPv6.

3.6_
Configuración de IPv6 en máquinas virtuales

La configuración de IPv6 en una interfaz de cualquier máquina virtual no debería ser diferente a la que aplicamos en una máquina real que corre un sistema operativo que está preparado para usar IPv6.

Por esta razón nos centraremos solo en los modelos de virtualización de red que nos brindan el soporte adecuado para configurar IPv6, y evitaremos las implementaciones que no son nativas o completas.

Para los ejemplos utilizaremos el prefijo de dirección IPv6 2001:db8::/32 reservado para documentación en el RFC 3849 y utilizaremos como referencia el sistema operativo Linux incluido en la distribución CentOS.

La configuración de red en la distribución de CentOS es la misma que presentan las distribuciones Fedora y RedHat, basada en un conjunto de archivos donde se definen variables y parámetros que controlan la asignación de IPv6. El archivo de órdenes (script) utilizado para iniciar el servicio de IPv6, configurar las interfaces y rutas, y reasignar parámetros del kernel es ifup-ipv6[10]. El detalle de archivos y variables se describen en la siguiente tabla:

VARIABLES DEFINIDAS EN EL ARCHIVO /ETC/SYSCONFIG/NETWORK		
Variable	**Descripción**	**Kernel**
IPV6INIT=yes|no	Habilita la configuración de IPv6 para la interfaz.	El módulo ipv6 debe estar cargado en el kernel.
IPV6FORWARDING=yes|no	Controla en reenvio de paquetes	net.ipv6.conf.DEVICE.forwarding=1|0
VARIABLES DEFINIDAS EN EL ARCHIVO /ETC/SYSCONFIG/NETWORK-SCRIPTS/IFCFG-ETHX, CON X=0,1,...N		
Variable	**Descripción**	**Kernel**
IPV6_DEFAULTGW=<dirección IPv6>	Controla la ruta por defecto o puerta de enlace IPv6 (valor opcional)	
IPV6ADDR= <dirección IPv6>[/<longitud del prefijo>]	Especifica la dirección primaria IPv6 de forma estática o manual	
IPV6ADDR_SECONDARIES= "<dirección IPv6>[/<longitud del prefijo>]..."	(valor opcional)	
IPV6_ROUTER=yes|no	Controla la auto-configuración IPv6 (no: interfaz proveedor múltiple sin encaminado)	net.ipv6. conf.DEVICE.forwarding=1|0
IPV6_AUTOCONF=yes|no IPV6_MTU=<MTU for IPv6>	Controla la auto-configuración IPv6 Controla el MTU en IPv6 aplicado a esta interfaz. (valor opcional)	net.ipv6.conf.DEVICE.accept_ra=1|0 redirects=1|0

TABLA 3: CONFIGURACIÓN DE INTERFACES DE RED: VARIABLES Y PARÁMETROS DEL KERNEL

La configuración de IPv6 presenta dos modos de operación por omisión: router y host. Estos modos se definen mediante la combinación de variables que activan diferentes parámetros como vimos en el punto 3.4.3.

Para el modo router, las variables deberían tener los valores:
`IPV6FORWARDING=yes, IPV6_AUTOCONF=no, IPV6_ROUTER=yes`

Para el modo host, se anula el reenvío de paquetes y se habilita la auto-configuración:
`IPV6FORWARDING=no, IPV6_AUTOCONF=yes`

3.6.1. Xen

Como vimos en la sección 3.3.1, Xen presenta 3 modos de virtualización de red: Bridge, Router y NAT. De estos 3 modos, solo describiremos la configuración de IPv6 en los dos primeros, descartando el modo NAT porque es una tecnología orientada al uso de IPv4.

Por cada interfaz virtual de red, Xen crea un par de dispositivos de red. El que se denomina ethN, reside en el dominio huésped y se denomina de forma similar a una interfaz física. Esto significa que para el dominio huésped domU, su configuración es similar a la que usamos sobre una interfaz física ethernet. El segundo dispositivo de red es el que reside en el dominio principal dom0 y se identifica con el nombre vifDOMID.DEVID, donde DOMID es el identificador del dominio huésped y DEVID es el identificador del dispositivo ethernet creado para el domU. Por ejemplo, para el dominio dom5, el identificador es el 5, y los dispositivos creados para Dom0 y el dom5 toman el nombre de vif5.0 y eth0. Si se asignara una segunda interfaz virtual al dom5, esta seria la eth1 y en el Dom0 estaría asociada a la interfaz virtual vif5.1

Entre ambos dispositivos de red se establece un canal de comunicación virtual por el que pasa el tráfico entre el dominio huésped y el Dom0, que finalmente alcanza la interfaz física usando un bridge o router, dependiendo del modo de virtualización seleccionado en la instalación de Xen.

Cada interfaz virtual de red requiere de la asignación de una dirección MAC Ethernet para su funcionamiento. Esta asignación puede definirse en el archivo de configuración del dominio huésped, junto a otros parámetros de red que también pueden asignarse de forma manual.

Xen utiliza 3 formas de hacerlo, con el siguiente orden de preferencia:

- Asignar una dirección MAC de un rango asociado a un Identificador Unico Organizacional (OUI) que sea válido y que quién lo asigna debe controlar dicho rango y ser responsable por esta configuración

- Mediante la generación de una secuencia aleatoria de 6 bytes, con el primer byte siguiendo el patrón de bits xxxxxx10, con cada bit x generado de manera aleatoria, y los restantes 5 bytes también generados al azar.

- Asignando una dirección al azar dentro del espacio 00:16:3e:xx:xx:xx. El proyecto Xen tiene asignado el OUI 00:16:3e y está disponible a los usuarios de Xen para las asignaciones locales.

Estas diferentes formas de asignación de la dirección MAC responden a que esta debe ser única entre todos los dispositivos que se encuentran en el mismo segmento de la red local, sean estos físicos o virtuales. Si no se tiene un OUI propio, es preferible generar la dirección MAC usando la segunda opción ya que aleatoriamente tiene 46 bits frente a los 12 bits de la tercera opción, evitando la duplicidad y mejorando la seguridad cuando somos víctimas de un ataque de barrido de direcciones IPv6.

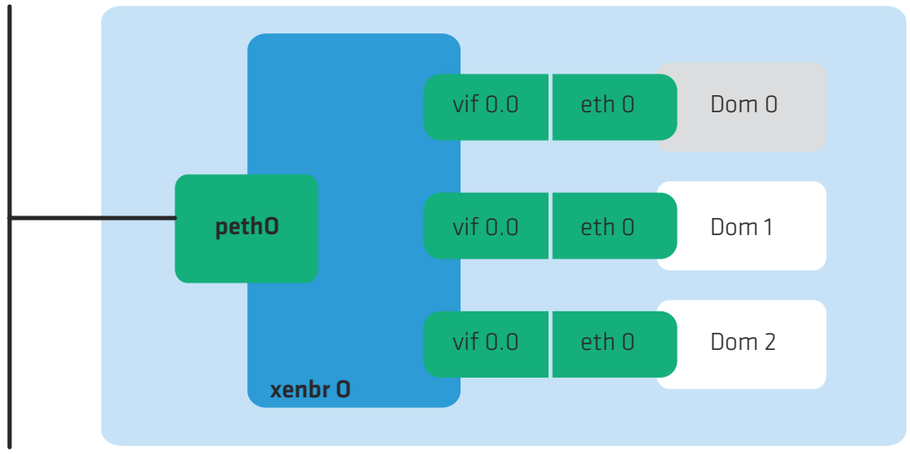

FIGURA 8: XEN EN MODO BRIDGE

El modo bridge en Xen es la configuración de red por defecto y la más común, creando el bridge por software en el Dom0 y permitiendo que todas las domU sean visibles en la red.

En el modo bridge se reasignan los nombres de las interfaces, siendo el dispositivo eth0 renombrado a peth0, el dispositivo físico y se crea un bridge con el nombre xenbr0 como se muestra en la Figura 8.

Cada MV que se crea tiene asociado un archivo de configuración en donde se definen los parámetros como son el número de CPUs, el tamaño de la memoria, los sistemas de archivos asociados al disco, y la interfaz virtual que se denomina vif.

Cuando se ejecuta el demonio xend al iniciar el Dom0 se activan las MV cuyos archivos de configuración se encuentran en el directorio /etc/xen/auto.

Si tenemos 2 o más interfaces físicas es posible definir un bridge para cada interfaz física y luego asociar cada MV a uno de estos bridges dentro del archivo de configuración. En el escenario que se muestra en la Figura 9 tenemos una máquina física que tiene una tarjeta ethernet de 4 puertos en la que se habilitan 2 puertos ethernet y se definen 2 bridges debido a que cada puerto esta conectado a una VLAN diferente.

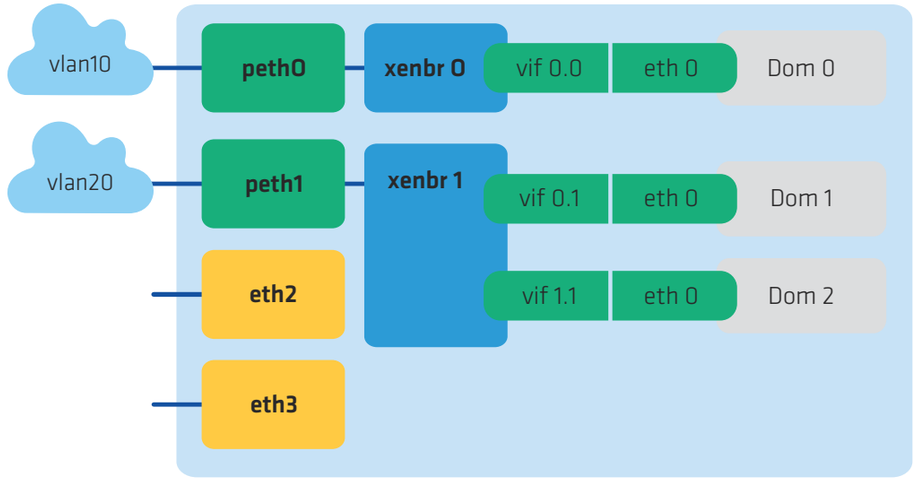

FIGURA 9: EJEMPLO DE ARQUITECTURA BRIDGE EN XEN

Para ver las MV que se están activas y corriendo, usamos el comando xm:

```
# xm list
Name                ID Mem(MiB) VCPUs State   Time(s)
Domain-0             0     5450     8 r----- 103347.5
iris                 1     1024     1 -b---- 1177253.0
ns1                  2     1024     1 -b----  27655.0
vpn                 10      512     1 -b----   7742.6
```

Las 3 MV asociadas a diferentes bridges tienen la siguiente configuración:

```
# cat /etc/xen/auto/* | grep ^vif
iris: vif = [ “mac=00:16:3e:5c:90:25,bridge=xenbr0,script=vif-bridge” ]
ns1:  vif = [ “mac=00:16:3e:20:42:7a,bridge=xenbr1,script=vif-bridge” ]
vpn:  vif = [ “mac=00:16:3e:78:6e:6a,bridge=xenbr0,script=vif-bridge” ]
```

En este caso tenemos 3 MV que utilizan 2 bridges diferentes con los nombres xenbr0 y xenbr1. También podemos observar que las direcciones MAC tienen el OUI 00:16:36 para todos los casos.

```
xenbr0    Link encap:Ethernet  HWaddr FE:FF:FF:FF:FF:FF
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:4480186 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:827157828 (788.8 MiB)  TX bytes:0 (0.0 b)
```

```
xenbr1    Link encap:Ethernet  HWaddr FE:FF:FF:FF:FF:FF
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:4480134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:827893361 (789.5 MiB)  TX bytes:0 (0.0 b)
```

Si observamos la distribución de interfaces virtuales en cada bridge y la interfaz física asociada, tenemos:

```
# brctl show
bridge name    bridge id            STP enabled    interfaces
virbr0         8000.000000000000    yes
xenbr0         8000.fefffffffff     no             vif10.0
                                                   vif1.0
                                                   vif0.0
                                                   peth0
xenbr1         8000.fefffffffff     no             vif2.0
                                                   vif0.1
                                                   peth1
```

Es necesario ahora definir en la MV la configuración IPv6 en su interfaz. A modo de prueba y en forma manual podemos ejecutar los siguientes comandos dentro de vm01:

```
# ip -6 addr add 2001:db8:90::30/64 dev eth0
# ip -6 route add default via 2001:db8:90::1
```

Podemos verificar la configuración IPv6 ejecutando los comandos:

```
# ip -6 addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:90::30/64 scope global dynamic
    inet6 fe80::5054:ff:fe7d:78ed/64 scope link

# ip -6 route show
2001:db8:90::/64 dev eth0  proto kernel  metric 256  expires 0sec mtu
1500 advmss 1440 hoplimit 4294967295
fe80::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
hoplimit 4294967295
default via  2001:db8:90::1 dev eth0  proto kernel  metric 1024
expires 0sec mtu 1500 advmss 1440 hoplimit 64
```


En este momento tendríamos conectividad IPv6. Para verificarlo podemos ejecutar un ping6 destinado al router previamente configurado:

```
# ping6 2001:db8:90::1
PING 2001:db8:90::1(2001:db8:90::1) 56 data bytes
64 bytes from 2001:db8:90::1: icmp_seq=1 ttl=64 time=1.17 ms
64 bytes from 2001:db8:90::1: icmp_seq=2 ttl=64 time=0.457 ms
64 bytes from 2001:db8:90::1: icmp_seq=3 ttl=64 time=0.504 ms
.........
```

3.6.1.1. Modo router en Xen

En modo router Xen requiere actuar como intermediario del protocolo NDP para permitir la auto-configuración IPv6 de las MV. En el Dom0 definimos los parámetros del kernel que habilitan el uso del protocolo NDP agregando las siguientes lineas en el archivo /etc/sysctl.conf

```
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.default.proxy_ndp=1
net.ipv6.conf.all.proxy_ndp = 1
```

El próximo paso es indicar en el archivo de configuración de la MV el script a utilizar en modo router al momento de iniciar la MV:

```
vif=[“mac=00:16:3E:20:42:7A,script=vif-route,ip=192.168.1.30”]
```

Para tener conectividad IPv6 es necesario configurar la MV y podemos hacerlo de forma manual como vimos para Xen en modo bridge en el ejemplo de vm01, agregando estos comandos en el archivo /etc/rc.local para que se ejecuten al iniciar al MV o podemos configurarlo de manera permanente en los archivos que se describieron en la tabla 3.

Si estamos dentro de un centro de datos tipo solo IPV6, necesitamos tener conectividad IPv6 en las MV desde el momento de la creación para tener acceso a las mismas por fuera del supervisor y de la consola que este provee. Xen no ofrece de base esta capacidad en sus scripts, por lo que es necesario proveer una solución que puede provenir de aplicar un parche al script vif-route para la versión de Xen en uso, o bajar un script disponible en Internet que nos provea la solución sin necesidad de modificar nosotros el script vif-route. Algunas de estas soluciones pueden consultarse en sitios similares a[11].

Con el script vif-route ya con soporte para IPv6, tenemos que ajustar la configuración de red de Xen para que funcione en modo router. Esto lo hacemos editando el archivo /etc/xen/scripts/xend-config.sxp y quitando el comentario a las lineas:

```
(network-script network-route)
(vif-script     vif-route)
```

Ahora podemos agregar en la definición de la interfaz virtual la dirección IPv6 a asignar a la MV:

```
vif=[“mac=00:16:3E:20:42:7A,ip=192.168.1.30 2001:db8:90::30/64”]
```

Al iniciar la MV tendremos conectividad IPv6 dentro del segmento de red y sobre el prefijo IPv6 donde se encuentra la MV.

3.6.2. OpenVZ

Para que los contenedores puedan usar IPv6 es necesario previamente configurar IPv6 en la interfaz del nodo principal y activar algunos parámetros del kernel.

Este es un ejemplo para configurar IPv6 para la distribución de Linux CentOS.

Agregar los siguientes parámetros de configuración de red en el archivo /etc/sysconfig/network

```
IPV6INIT=yes
```

Agregar las siguientes lineas en el script de configuración de red de la respectiva interfaz. Para el caso de la interfaz eth0, el archivo a modificar es /etc/sysconfig/network-scripts/ifcfg-eth0

```
NETWORKING_IPV6=yes
IPV6FORWARDING=yes
IPV6_DEFAULTDEV=eth0
IPV6_AUTOCONF=no
IPV6_DEFAULTGW=2001:db8:90:192::1
IPV6ADDR=2001:db8:ab34:90:192::2/64
```

En este ejemplo utilizamos el prefijo 2001:db8:90:192::/64, con la dirección ::1 para identificar a la puerta de enlace y la ::2 para principal.

En el archivo /etc/sysctl.conf es necesario agregar los siguiente parámetros del kernel para permitir el reenvío de paquetes IPv6 y activar la funcionalidad de intermediario del protocolo ND para la auto-configuración de la interfaz virtual del contenedor

```
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.default.proxy_ndp=1
net.ipv6.conf.all.proxy_ndp = 1
```

Finalmente, para que estos valores queden activos en el kernel y se active cuando reiniciamos el contenedor, ejecutamos:

```
# sysctl -p
```


Para una última comprobación de que estos parámetros del kernel están activos con el valor 1, ejecutamos:

```
# sysctl -a | grep net.ipv6.conf
```

Después de configurar el nodo principal, procedemos a configurar la máquina huésped. Es importante primero verificar que el módulo IPv6 del kernel este activo. Podemos verificarlo ejecutando:

```
# lsmod | grep ipv6
```

Y luego agregamos en el archivo /etc/sysconfig/network la variable

```
IPV6INIT=yes
```

Para asegurarnos que la pila IPv6 se activa al reiniciar la MV.

A modo de prueba podemos hacer una configuración manual de IPv6 para la MV. Para ello ejecutamos:

```
# ip -6 addr add 2001:db8:90::28/64 dev eth0
# ip -6 route add ::/0 via 2001:db8:90::1
```

Comprobamos que tenemos conexión IPv6 con el nodo principal y con el router

```
# ping6 -q -c 5 2801:db8:90::2
PING 2001:db8:90::2(2001:db8:90::2) 56 data bytes

--- 2001:db8:90::2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.016/0.024/0.046/0.012 ms

# ping6 -q -c 5 2001:db8:90::1
PING 2001:db8:90::1(2001:db8:90::1) 56 data bytes

       --- 2001:db8:90::1 ping statistics ---
       5 packets transmitted, 5 received, 0% packet loss, time 3999ms
       rtt min/avg/max/mdev = 0.332/0.342/0.352/0.014 ms
```

Si queremos que esta configuración sea permanente definimos los parámetros configurados manualmente en las variables del archivo / etc/sysconfig/network-script/ifcfg-eth0

```
NETWORKING_IPV6=yes
IPV6FORWARDING=no
IPV6_DEFAULTDEV=eth0
IPV6_AUTOCONF=no
IPV6_DEFAULTGW=2001:db8:90::1
IPV6ADDR=2001:db8:90::28/64
```

Si utilizamos direcciones IPv6 públicas, podemos verificar que otros sitios IPv6 son alcanzables. Por ejemplo, google ofrece los servidores de dominio DNS públicos para IPv6 2001:4860:4860::8888 y 2001:4860:4860::8844. Podemos verificar que son alcanzables ejecutando:

```
# ping6 -c 5 2001:4860:4860::8888
PING 2001:4860:4860::8888(2001:4860:4860::8888) 56 data bytes
64 bytes from 2001:4860:4860::8888: icmp_seq=1 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=2 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=3 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=4 ttl=54 time=176 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=5 ttl=54 time=175 ms

--- 2001:4860:4860::8888 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 175.747/176.003/176.745/0.531 ms

# ping6 -c 5 2001:4860:4860::8844
PING 2001:4860:4860::8844(2001:4860:4860::8844) 56 data bytes
64 bytes from 2001:4860:4860::8844: icmp_seq=1 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=2 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=3 ttl=54 time=175 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=4 ttl=54 time=176 ms
64 bytes from 2001:4860:4860::8844: icmp_seq=5 ttl=54 time=178 ms

--- 2001:4860:4860::8844 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 175.670/176.351/178.209/1.047 ms
```

Luego podemos agregar estos servidores DNS para nuestra máquina virtual en el archivo /etc/resolv.conf de la siguiente forma:

```
nameserver 2001:4860:4860::8888
nameserver 2001:4860:4860::8844
```

3.6.3. KVM

En los ejemplos de configuración de IPv6 en KVM que siguen a continuación nos basaremos en el modo bridge de virtualización de red y utilizando la distribución CentOS de Linux

El primer paso es verificar que están instalados los paquetes que requiere KVM para su funcionamiento y administración. Estos paquetes son, en la mayoría de las distribuciones de Linux:

```
bridge-utils, qemu-kvm-tools, qemu-kvm, libvirt, virt-manager
```


Para verificar que están instalados podemos correr el comando rpm con un filtro específico para cada paquete. Por ejemplo, para verificar que están instalados los paquetes qemu-kvm ejecutamos

```
# rpm -qa | grep qemu-kvm
qemu-kvm-tools-0.12.1.2-2.355.0.1.el6.centos.3.x86_64
qemu-kvm-0.12.1.2-2.355.0.1.el6.centos.3.x86_64
```

Luego verificamos que los módulos de KVM están corriendo.

```
# lsmod | grep kvm
kvm_intel              53484  4
kvm                   316602  1 kvm_intel
```

Ahora iniciamos el demonio libvirtd que nos permite gestionar el sistema de virtualización. Las dos formas de hacerlo son:

```
# /etc/init.d/libvirtd start
# service libvirtd start
```

Para que libvirtd se ejecute cuando reiniciamos la máquina física, configuramos:

```
# chkconfig  level 35 libvirtd on
```

KVM presenta un nueva interfaz denominada virbr0, que es un bridge virtual propio que se crea por defecto para brindar aislación y comunicación entre el nodo principal y las futuras máquinas virtuales. Esta interfaz tiene asignada la dirección IP 192.168.122.1/24 y dentro de esta subred pueden asignarse direcciones IPv4 al resto de las MV a crear.

```
# ifconfig virbr0
virbr0    Link encap:Ethernet  HWaddr 52:54:00:B8:20:57
          inet addr:192.168.122.1  Bcast:192.168.122.255 Mask:255.255.255.0
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
             collisions:0 txqueuelen:0
             RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Se observa que la dirección MAC tiene el OUI 52:54:00 que es el correspondiente al proyecto QEMU. Si observamos el estado del cortafuegos para IPv4 y la cadena NAT vemos que el bridge virbr0 es local y que utiliza NAT sobre la IP de la interfaz eth0.

```
# iptables -t nat -L -n
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  tcp  --  192.168.122.0/24    !192.168.122.0/24    masq
ports: 1024-65535
MASQUERADE  udp  --  192.168.122.0/24    !192.168.122.0/24    masq
ports: 1024-65535
MASQUERADE  all  --  192.168.122.0/24    !192.168.122.0/24

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Para usar IPv6 en KVM es necesario configurar un bridge que no tenga las limitaciones de virbr0 antes de comenzar a instalar las nuevas MV. Para esto, en el directorio /etc/sysconfig/network-scripts creamos un nuevo archivo ifcfg-br0 y modificamos el archivo ifcfg-eth0 para incorporarlo al bridge. Los archivos son los siguientes:

Se resaltan las principales diferencias entre los archivos de configuración entre el bridge br0 y la interfaz eth0. La variable DEVICE indica el nombre del dispositivo, TYPE define el tipo de dispositivo (Bridge|Ethernet) y en el caso de la interfaz eth0 es necesario asociarla al bridge mediante el parámetro BRIDGE=br0.

/etc/sysconfig/network-scripts/ifcfg-br0	/etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=br0	**DEVICE=eth0**
BOOTPROTO=static	BOOTPROTO=static
BROADCAST=192.168.1.255	HWADDR=4C:72:B9:B0:E3:D0
DNS1=192.168.1.2	NM_CONTROLLED=no
GATEWAY=192.168.1.1	ONBOOT=yes
IPADDR=192.168.1.32	**TYPE=Ethernet**
IPV6ADDR=2001:db8:90::32/64	**BRIDGE=br0**
IPV6INIT=yes	
IPV6_AUTOCONF=no	
NETMASK=255.255.255.0	
NM_CONTROLLED=no	
ONBOOT=yes	
TYPE=Bridge	

TABLA 4: ARCHIVOS DE CONFIGURACIÓN DE INTERFACES

Otras variables propias de los archivos “ifcfg-X” son BOOTPROTO, que indica si se asigna una dirección IP estática o dinámica (static|dhcp), la variable ONBOOT para configurar la red al iniciar la máquina física y la variable NM_CONTROLLED que indica si el programa Network Manager controla dicho dispositivo.

Ahora necesitamos reiniciar la red para activar el bridge.

```
# /etc/init.d/network restart
```

Si verificamos las interfaces tenemos las siguientes:

```
# ifconfig
br0       Link encap:Ethernet  HWaddr 4C:72:B9:B0:E3:D0
          inet addr:192.168.1.32  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: 2001:db8:90::32/64 Scope:Global
          inet6 addr: fe80::4e72:b9ff:feb0:e3d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:383792 errors:0 dropped:0 overruns:0 frame:0
          TX packets:250606 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:225550607 (215.1 MiB)  TX bytes:39931149 (38.0 MiB)

eth0      Link encap:Ethernet  HWaddr 4C:72:B9:B0:E3:D0
          inet6 addr: fe80::4e72:b9ff:feb0:e3d0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:602636 errors:0 dropped:0 overruns:0 frame:0
          TX packets:378914 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:558122420 (532.2 MiB)  TX bytes:48845029 (46.5 MiB)
          Interrupt:20 Memory:f7c00000-f7c20000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22942 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22942 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16110994 (15.3 MiB)  TX bytes:16110994 (15.3 MiB)

virbr0    Link encap:Ethernet  HWaddr 52:54:00:B8:20:57
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

En este ejemplo el bridge br0 está configurado con direcciones IPv4 e IPv6, y utiliza un modelo de doble pila que permite usar ambos protocolos.

En modo bridge, los parámetros del kernel con valor 0 a verificar son:

```
net.ipv6.conf.all.forwarding = 0
net.bridge.bridge-nf-call-arptables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-ip6tables = 0
```

Aunque no es la finalidad de este capítulo entrar en el detalle de los pasos necesarios para crear una nueva máquina virtual, si nos interesa ver como configurar IPv6 al crear una nueva máquina virtual. En las siguientes imágenes se observa la diferencia en las opciones avanzadas cuando esta previamente configurado un bridge en el supervisor. En la Figura 10 se muestra el caso por defecto, en donde solo se puede configurar la red virtual en modo NAT, lo cual no es compatible para IPv6.

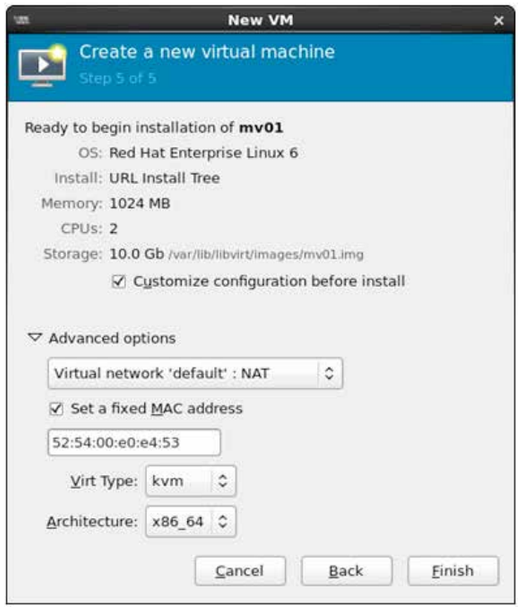

FIGURA 10: CONFIGURACIÓN DE RED VIRTUAL EN MODO NAT

Por otro lado en la Figura 11 se observa que esta disponible el dispositivo eth0 asociado al bridge br0 para la nueva MV. En ambos casos las direcciones MAC pueden fijarse y presentan el OUI 52:54:00 antes mencionado.

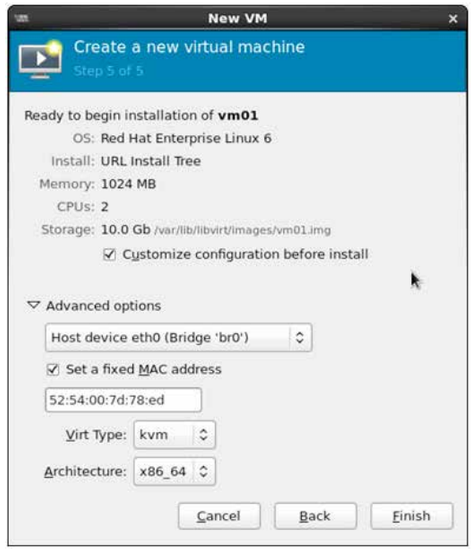

FIGURA 11: CONFIGURACIÓN DE RED VIRTUAL EN MODO BRIDGE

La instalación continúa y se muestra en una consola virtual generada mediante la aplicación VNC. La configuración de la red dentro de la MV no varía de cualquier instalación habitual en la distribución utilizada y es en ese paso en el que debe seleccionarse si la asignación es de forma manual o automática para las direcciones IPv4 e IPv6.

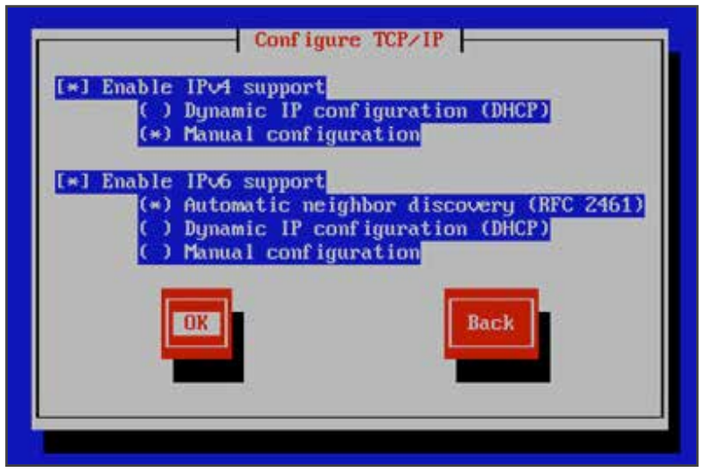

FIGURA 12: CONFIGURACIÓN DE TCP/IP

Si en el paso de configuración que muestra la Figura 12 seleccionamos la opción de configuración manual para ambos protocolos, el proceso de instalación nos ofrece completar los campos de la ventana que se muestra en la Figura 13. Para IPv6 debemos definir el campo de la dirección IPv6 y el del prefijo. Por ejemplo: 2001:db8:90::30/64

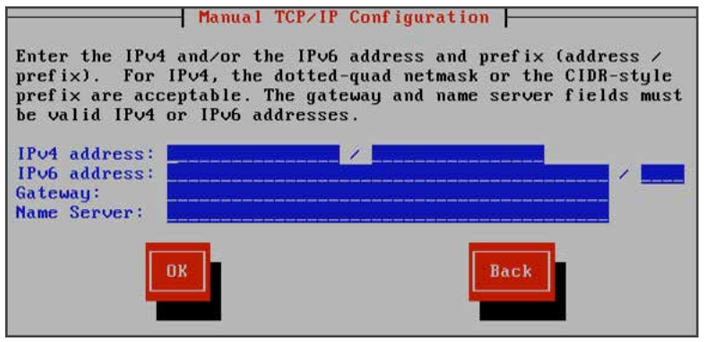

FIGURA 13: CONFIGURACIÓN MANUAL DE TCP/IP

Una vez finalizada la instalación de la MV, podemos conectarnos vía IPv6 y verificar la configuración de la red. En este ejemplo la dirección IPv6 es asignada de forma automática por NDP para un prefijo 2001:db8:90::/64.

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:7D:78:ED
          inet addr:192.168.1.30  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: 2801:0:90::5054:ff:fe7d:78ed/64 Scope:Global
          inet6 addr: fe80::5054:ff:fe7d:78ed/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:125527 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4683 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20534972 (19.5 MiB)  TX bytes:366917 (358.3 KiB)
```

En el nodo principal tenemos también la interfaz vnet0 que es una interfaz tipo tap asociada al proceso KVM y hace la función de capa de enlace con la interfaz de la respectiva MV. Esta asociatividad con la interfaz de la MV se puede observar en su dirección MAC, la cual solo varía en el primer byte FE.

```
vnet0     Link encap:Ethernet  HWaddr FE:54:00:7D:78:ED
          inet6 addr: fe80::fc54:ff:fe7d:78ed/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4624 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108631 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:360021 (351.5 KiB)  TX bytes:18974201 (18.0 MiB)
```


En el bridge podemos ver las interfaces definidas y asociadas al br0:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.4c72b9b0e3d0       no              eth0
                                                        vnet0
virbr0          8000.525400b82057       yes             virbr0-nic
```

Al igual que con Xen y OpenVZ, podemos verificar la configuración y asignaciones de direcciones IPv6 en el supervisor y en las MV utilizando los comandos ip -6 addr|route o ip -6 neigh. Para verificar la conectividad podemos usar ping6, traceroute6 y mtr -6, apuntando a direcciones IP destino dentro del segmento o prefijo de red y fuera de la LAN.

3.7_ Switches virtuales

El número de máquinas virtuales que corren sobre una máquina física ha crecido en función de la mayor capacidad de procesamiento y de virtualización que ofrecen la evolución permanente de los microprocesadores . En los primeros trabajos de investigación este número alcanzaba las 10 MV alojadas en cada máquina física con un desempeño aceptable. En 2010 este número aumento a 40 y llegó hasta 60 MV alojadas, y en la actualidad el número supera las 120 MV corriendo sobre una máquina real.

Este alto número de MVs ha creado el concepto de Centro de Datos Virtualizados, en los cuales el último salto de un paquete que atraviesa un switch sucede dentro del servidor (edge switch) y en la medida que el número de MV aumenta puede transformarse en un cuello de botella.

Por otro lado el uso de tecnologías de red de 10 Gbps en las troncales y en las capas de acceso y de agregación requieren de un alto desempeño por parte de las máquinas virtuales para abastecer el número de conexiones simultaneas, el ancho de banda de cada conexión, los retardos y latencia en la transferencia de los datos requeridos por la aplicación y el uso del procesador, memoria y disco que requieren los diferentes procesos que ejecutan en el requerimiento de un servicio.

El modelo tradicional de virtualización no es escalable cuando pensamos en implementaciones dentro de un Centro de Datos. Este modelo tradicional presenta un supervisor de capa de hardware que provee un modelo de virtualización de red simple, mediante un switch en capa 2 (L2) o a través de un router IP en capa 3 (L3), pero sin independizarse del uso del kernel.

Los switches virtuales presentan una evolución respecto al modelo tradicional, incorporando tecnologías que mejoran el plano de control y de visibilidad en la capa de red, al soportar protocolos de gestión de interfaces como SNMP y con acceso a linea de comandos (CLI) remoto sin necesidad de entrar previamente al supervisor. A esto se agregan otras capacidades como es disponer de más de una interfaz virtual (VIF) por MV, configurar VPN para diferenciar segmentos entre MV, migración de MV entre subredes, y tener acceso a la tabla de reenvío de paquetes (forwarding table) y manejar la salida a uno o más puertos, además de soportar IPV6.

3.7.1. Open vSwitch

Open vSwitch[12] es un switch desarrollado e implementado para ambientes virtualizados y se diferencia del modelo tradicional incluido en los kernels de los sistemas operativos, en que presenta una interfaz de control de reenvío de paquetes (forwarding) de grano fino, lo que permite implementar tecnologías que ofrecen los switches físicos como son la calidad de servicio (QoS), definición de túneles, reglas de filtrado, etc.

Open vSwitch ha sido incorporado en diferentes tecnologías de virtualización basadas en Linux como son Xen/XenServer, KVM, y VirtualBox y su arquitectura se muestra en la Figura 14. La versión 4.3 de Xen incorpora de base el paquete Open vSwitch para el manejo de la red virtual.

Una de las ventajas de Open vSwitch es que la mayor parte del código esta escrito en lenguaje C, lo que hace que sea independiente de la plataforma y es fácilmente trasladable a otros entornos. En cuanto a su funcionalidad como switch, presenta las siguientes características:

- Soporte del estándar de VLAN 802.1Q, con capacidad para definir puertos en modo acceso (access) y modo troncal (trunk).
- Protocolos de monitoreo por flujos: NetFlow y Sflow, y capacidad para espejado de puertos (port mirroring)
- Políticas de QoS (Calidad de Servicio)
- Configuración de Túneles: GRE, GRE sobre IPSEC, VXLAN y LISP
- Manejo de fallas de conectividad mediante el estándar 802.1ag
- Uso de extensiones de OpenFlow 1.0
- Alto desempeño en el reenvío de paquetes al usar un módulo del kernel en Linux.

El módulo es soportado en Linux kernel versión 2.6.18 o superior, siendo desde la versión 2.6.32 la más probadas para Xen con algunos parches sobre CentOS. También presenta soporte para Citrix XenServer y RedHat Enterprise.

Open vSwitch también puede ejecutarse en el espacio del usuario, sin la intervención del módulo del kernel, pero a un mayor costo en el rendimiento.

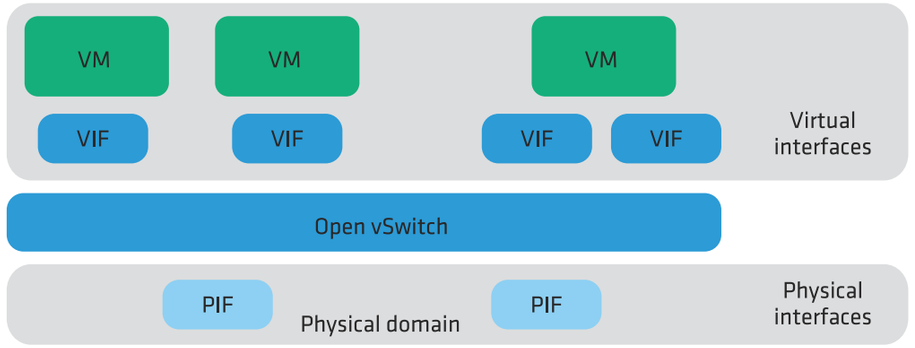

FIGURA 14: OPEN VSWITCH ARQUITECTURA

La especificación de OpenFlow 1.2 incluye un nuevo soporte para identificar flujos IPv6 además de permitir la re-escritura del encabezamiento del paquete IPv6 mediante el uso de estructuras más flexibles en el proceso de identificación de flujos. Esto permite identificar dentro de un flujo IPv6 la dirección de origen, la dirección destino, el número de protocolo, la clase de tráfico, el tipo de paquete ICMPv6, el código de ICMPv6, los campos del encabezamiento del NDP y la etiqueta de flujo (flow label) IPv6.

Open vSwitch presenta una serie de comandos que permiten monitorear y administrar el switch y operar sobre la tabla de flujos mediante el agregado, modificación y borrado de flujos. En el agregado del flujo es necesario especificar las características del flujo a identificar y luego definir una acción sobre ese flujo, como por ejemplo reasignar los puertos de salida, cambiar el identificador de una VLAN, disminuir el límite de saltos del paquete IPv6, etc.

La administración de los flujos IPv6 se realiza mediante el comando ovs-ofctl. El siguiente es un ejemplo de como agregar un flujo IPv6:

```
# ovs-ofctl add-flow br0 in_port=2,dl_type=0x86dd, \
ipv6_src=2001:db8::/32,ipv6_dst=2001:db8::/32,actions=out
put:5
```

Si analizamos la sintaxis del comando ovs-ofctl tenemos 4 partes a analizar:

- add-flow → agrega un flujo.
- br0 → identificador del bridge activo donde agregar el flujo a identificar.

- in_port=2,dl_type=0x86dd,ipv6_src=2001:db8::/32,ipv6_dst=2001:db8::/32 → Campos a comparar para identificar el flujo: puerto de entrada al OpenFlow bridge, flujo IPv6, dirección IPv6 origen, dirección IPv6 destino

- actions=output:5 → Acción a tomar sobre el fujo identificado: el paquete se envía al puerto 5 del OpenFlow bridge.

Para ver el flujo antes agregado, ejecutamos:

```
# ovs-ofctl dump-flows br0
       NXST_FLOW reply (xid=0x4):
       cookie=0x0, duration=3.214s, table=0, n_packets=0, n_bytes=0,
       ipv6,in_port=2,ipv6_src=2001:db8::/32,ipv6_dst=2001:db8::/32
       actions=output:5
```

Si queremos borrar el flujo antes agregado, ejecutamos:

```
# ovs-ofctl del-flows br0 in_port=2,dl_type=0x86dd, \
ipv6_src=2001:db8::/32, ipv6_dst=2001:db8::/32
```

Para más información ver en detalle el manual del comando ovs-ofctl[13].

3.7.2. Switches virtuales comerciales

Hay un número importante de fabricantes que desarrollaron sus propios switches virtuales para dar soporte y nuevas funcionalidades a las tecnologías de virtualización. Estas características los hacen similares al resto de los switches físicos, incorporando nuevos estándares y protocolos que facilitan la instalación y la administración de las máquinas virtuales en los centros de datos.

La mayoría ha sido desarrollado para dar soporte a VMware como tecnología de virtualización. Estos productos proveen diferentes capacidades y características que aumentan la prestación de los switches virtuales tradicionales que se ejecutan en el kernel.

VMware provee su propio producto de software que se llama VMware vNetwork Distributed Switch (VDS)[14]. Con este producto VMware supera a su predecesor vNetwork Standard Switch incorporando nuevas capacidades de administración, monitoreo y provisión a través de una interface centralizada, VLANs privadas y en el manejo de tráfico al agregar limitación en la velocidad de recepción de los puertos.

En colaboración con VMware, Cisco desarrolló el producto Cisco Nexus 1000V[15], un paquete de software que se instala en un hardware convencional para servidor y que agrega nuevas capacidades a las que presenta el propio VDS de VMware, en cuanto a la conectividad (LACP, Virtual Port Channels), al manejo de tráfico, a la Calidad de Servicio (DSCP, ToS), Seguridad (Listas de Acceso, RADIUS, DHCP snooping, ARP inspection), Monitoreo (SNMPv3, NetFlow v9), entre otras.

Otras empresas también han desarrollado productos similares diseñados para soportar el software de VMware. Uno de estos es el IBM Distributed Virtual Switch (DVS) 5000V, el cual también incorpora nuevas tecnologías y protocolos como son el Edge Virtual Bridging (EVB) basado en el estándar IEEE 802.1Qbg que permite la gestión escalable y flexible de la configuración de red y aplicar diferentes políticas por cada máquina virtual, eliminando muchos de los problemas de red introducidos con la virtualización tradicional de servidores.

Algo similar ocurre con HP y su producto FlexFabric Virtual Switch 5900v, el cual se basa en una solución integral de software y un switch físico en el ToR (Top-of-Rack) que mediante la tecnología Virtual Ethernet Port Aggregator (VEPA) permite asignar un puerto virtual de conexión a cada MV. Al igual que otros productos antes mencionados, el FlexFabric Virtual Switch 5900v es una solución de software diseñada para integrarse con el supervisor VMware ESX y como una alternativa al VMware vSwitch.

3.8_
IPv6 en centros de datos

Existen varias formas de introducir y operar IPv6 en Centros de Datos. Una forma es continuar con una operación IPv4 dentro del centro de datos y hacer algún tipo de translación en el borde, una segunda forma es usar la pila doble y una tercera es usar únicamente IPv6. En resumen tenemos:

1) Translación de IPv4 en el borde: En este escenario el centro de datos mantiene su infraestructura interna en IPv4 y hace algún tipo de translación a IPv6 en el borde. Usualmente este mecanismo se aplica sólo a servicios HTTP/HTTPS y se hace uso de proxies reversos.

2) Pila Doble: Aquí encontramos pila doble a través todos los servicios del centro de datos o al menos en los que presentan servicios a usuarios. También puede encontrarse pila doble solo en el borde mientras que las conexiones internas son IPv4 o IPv6 únicamente.

3) Solo IPv6: Esta es generalmente la etapa final de la transición de un centro de datos a IPv6. Aquí encontramos IPv6 en todos los elementos del centro de datos. Para ofrecer servicios a los usuarios legados de IPv4 se utiliza algún tipo de translación en el borde.

El uso de estos escenarios no es necesariamente en la forma secuencial descrita y tampoco ninguno es el mejor, el más correcto o el recomendado. Cada uno ofrece diferentes beneficios y desventajas que deben ser analizados para seleccionar la mejor opción.

3.8.1. Recomendaciones operativas para un centro de datos IPv6

A pesar de que IPv6 tiene más de diez años, aún es muy poca la experiencia operativa como para formular un grupo de mejores prácticas aceptadas universalmente. Sin embargo a continuación presentamos algunas consideraciones operativas a tomarse en cuenta para un centro de datos.

3.8.1.1. Direccionamiento

Existen varias consideraciones importantes en relación al direccionamiento en un centro de datos. La primera es que tipo de direccionamiento se debe usar; esto es Agregado por Proveedor (PA), Proveedor Independiente (PI) o direcciones Unique Local IPv6 (ULAs). En relación de PA vs PI, PI proveé una independencia del ISP y reduce los problemas con renumeración, sin embargo trae consigo el pago por una asignación del RIR y muy posiblemente otros costos extras de administración y operación.

En caso de usar ULAs, éstas deben solo usarse en infraestructura que no requiere acceso al Internet público como servidores de bases de datos, servidores de aplicación e interfaces de administración entre otros. En caso de además usar direccionamiento PA, el uso de ULAs puede disminuir el problema de renumeración.

Otro punto de debate es la longitud de los prefijos en el centro de datos. En general recomendamos el uso de subredes de 64 bits por cada VLAN o segmento de red. El uso de subredes de longitud mayor a 64 bits es aceptable siempre y cuando se conozcan los posibles inconvenientes como el romper SLAAC y tener que usar configuración manual. Finalmente los planes de direccionamiento deben seguir los principios de ser jerárquicos y poder agregar espacio. Se recomienda al menos el uso de un /48 por cada centro de datos.

3.8.1.2. Seguridad

La mayoría de los aspectos de seguridad de IPv6 se aplican a los centros de datos los cuales pueden encontrarse en[16]. Sin embargo un aspecto importante son los ataques a Neighbor Discovery Protocol (NDP). Este ataque es similar a los ataques de ARP de IPv4 y el atacante puede llenar el caché de vecinos y acabarse la memoria del enrutador resultando en la inhabilidad de éste para reenviar paquetes.

A pesar que el espacio de las subredes de 64 bits es muy grande para emplear un escaneo tradicional como en IPv4, existen algunos métodos que permiten reducir el espacio de escaneo. Si el escaneo es una preocupación para el administrador del centro de datos se recomienda no hacer uso de SLAAC y evitar asignar direcciones manualmente usando “low-byte” (i.e. de 0 a 256), direcciones basadas en IPv4 y direcciones que asemejen una palabra (i.e. bebe:cafe).

Aunque los centros de datos son ambientes controlados donde el uso de DHCP no es común y la imperzonalización de RA no es común, se recomienda el uso de herramientas que eviten el secuestro de RA (RFC 6104, RFC 6105) y DHCP[17].

Y sin mayor diferencia que en IPv4, también es necesario tomar todas las precauciones para evitar los ataques de amplificación y aplicar BCP38[18] en filtrado en el ingreso. Al mismo tiempo se debe enfatizar el uso de listas de control de acceso en los puntos de translación.

3.8.1.3. Monitoreo

El monitoreo es una operación crítica para las operaciones de cualquier red y debe ser hecho con el mismo cuidado en IPv6 y en IPv4. En el caso de centros de datos no son diferentes al hecho en cualquier otra red con IPv6. Es sin embargo importante considerar que los equipos de red y el software de monitoreo debe soportar IPv6 en la colección de datos (por ejemplo MIBs) a pesar de que el transporte de estos sea solo IPv4 (por ejemplo es posible recolectar información de IPv6 usando IPFIX a pesar de que los paquetes sean enviados usando transporte IPv4).

3.7.1.4. Sistemas de administración de red y aplicaciones

Los centros de datos pueden usar software para administrar sus operaciones como por ejemplo sistemas de administración de direcciones (IPAM), sistemas de provisionamiento y otra variedad de software de documentación y operación. Es importante que este software este preparado para soportar IPv6 en sus modelos de datos. En general, si IPv6 no ha sido soportado aún por estas aplicaciones los cambios pueden ser más complejos que agregar más espacio en los campos de entrada.

3.8.2. Motivaciones para un centro de datos solo IPv6

Existen varias motivaciones para considerar un centro de datos solamente IPv6. Primeramente tenemos la escasez de direcciones IPv4 puede obligar a tratar de rescatar direcciones donde no son totalmente necesarias. De la misma forma, esta limitación en obtener más direcciones IPv4 limitará el crecimiento de centros de datos en pila doble o en un ambiente IPv4 con translación en el borde.

Otra motivación es el ahorro de costos de administración, operación y mantenimiento que un ambiente de solo IPv6 puede traer en comparación con el manejo de dos pilas de direccionamiento. En principio los administradores de red deben de aprender dos pilas de protocolo, deben aplicar reglas de seguridad en duplicado y deben manejar operaciones en dos protocolos. Todo esto además de agregar un trabajo extra en la administración del centro de datos lo deja propenso a errores de configuración y huecos de seguridad.

Algunos otros factores que incrementan el costo de operación de centros de datos en pila doble son: El desarrollo, prueba y QA (Quality Assurance) de aplicaciones en dos pilas de protocolos; operación y administración de fallas; y administración y monitoreo de la red entre otros.

3.9_
Referencias

[1] M. Tim Jones, Virtual Linux. An overview of virtualization methods, architectures, and implementations. IBM DeveloperWorks article, December 2006

[2] The Xen Project, the powerful open source industry standard for virtualization, http://www.xenproject.org

[3] OpenVZ, http://www.openvz.org

[4] Timo Hirt, KVM - The kernel-based virtual machine, 2010

[5] VMware, http://www.vmware.com

[6] RFC4389, Neighbor Discovery Proxies (ND Proxy), http://tools.ietf.org/rfc/rfc4389.txt

[7] RFC2461, Neighbor Discovery for IP Version 6 (IPv6), http://tools.ietf.org/rfc/rfc2461.txt

[8] IPv6: Updates for net-2.6.19, http://lwn.net/Articles/200018/

[9] sysctl, https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

[10] initscripts-ipv6, http://www.deepspace.net/initscripts-ipv6.html

[11] BenV's notes, Xen and routed IPv6, http://notes.benv.junerules.com/tag/xen/

[12] Open vSwitch, http://www.openvswitch.org

[13] ovs-ofctl - administer OpenFlow switches, http://openvswitch.org/cgi-bin/ovsman.cgi?page=utilities%2Fovs-ofctl.8

[14] Vmware vNetwork Distribution Switch, http://www.vmware.com/products/datacenter-virtualization/vsphere/distributed-switch.html

[15] Cisco Nexus 1000V Series Switches for VMware vSphere, http://www.cisco.com/en/US/products/ps9902/index.html

[16] Operational Security Considerations for IPv6 Networks",draft-ietf-opsec-v6. Chittimaneni, K., Kaeo, M., and E. Vyncke. 2013

[17] Network Reconnaissance in IPv6 Networks", draft-ietf-opsec-ipv6-host-scanning Gont, F. and T. Chown. 2013

[18] BCP38 Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. P. Ferguson, D. Senie. 2000