Universidad Nacional Autónoma de Nicaragua.
Unan-León

Facultad de Ciencias y Tecnología
Departamento de Computación
Ingeniería en Telemática

Monografía para optar al título de Ingeniero en Telemática:

Vyatta: una alternativa para el uso en los laboratorios del área de Redes de Computadores.

Integrantes:

Br. Lester Ramón Acevedo Hernández.

Br. José Miguel Bárcenas Flores.

Tutor:

M.Sc. Aldo René Martínez Delgadillo.

León, Nicaragua 02 de Abril del 2013.

AGRADECIMIENTO

En primer lugar damos gracias a Dios por darnos la fortaleza espiritual y física para ser capaces de enfrentar cada dificultad que se nos presentó durante la realización de dicho trabajo monográfico.

A nuestros Padres por habernos apoyado tanto sentimental como económicamente a lo largo de nuestras vidas.

A nuestro tutor M.Sc. Aldo Rene Martínez Delgadillo, ya que con sus recomendaciones y el tiempo que nos brindó logramos finalizar nuestra tesis de forma exitosa.

Y finalmente a nuestros Maestros por habernos guiados durante estos cuatro años para llegar a ser los profesionales que hoy somos.

DEDICATORIA

Este trabajo lo dedicamos a nuestros padres que con mucho esfuerzo

Y dedicación nos han apoyado a lo largo de nuestros estudios

Y nuestras vidas al poner toda su confianza en nosotros

Y lograr así que seamos unos excelentes profesionales.

_____ _____

Br. Lester Ramón Acevedo Hernández Br. José Miguel Bárcenas Flores

TABLA DE CONTENIDO:

TABLA DE FIGURAS.

Vyatta: una alternativa para el uso en los laboratorios del área de Redes de Computadores.

Capítulo I: Introducción.

1. Antecedentes.

Hasta el momento en el Departamento de Computación, no se ha realizado ningún trabajo monográfico relacionado con el uso de programas **Open Source Routing**.

El trabajo más similar al nuestro, es la monografía titulada ***"Prácticas de Laboratorio para la Asignatura de Redes de Ordenadores II"*** empleando el simulador Packet Tracer en el año 2008.

El área de Ingeniería Telemática, cuenta con 4 equipos físicos (Router y Switches) hasta el momento. Este es un enorme problema que tiene el área, debido a que no se logran realizar pruebas de laboratorio reales. Lo que utiliza son simuladores como PacketTracer, que son de gran ayuda, pero no permiten interacción entre el software y el equipo físico. Este tipo de simuladores son herramientas interactivas de aprendizaje y simulación de redes, y tienen el propósito de ser usadas como un producto educativo que brinden nociones de configurar dispositivos de Cisco. Packet Tracer, por ejemplo, brinda soporte para protocolos sencillos, pero no trae incorporados protocolos más complejos como BGP. Otra herramienta usada es GNS3, pero este emulador consume demasiados recursos de hardware, ya que se emula uno o más routers reales dentro de la computadora.

Como alternativa, se implementan distribuciones de Linux, con soporte de protocolos básicos y servicios de red, que ayuden a implementar prácticas más reales. Ejemplo de estas distribuciones son Pfsense, Zeroshell y Monowall. Estas herramientas tienen usos diversos, y pueden ser usados como Firewall, Router o NAT y no consumen muchos recursos de hardware.

2. Definición del problema.

¿Es viable utilizar herramientas de software **Open Source Routing** para realizar las prácticas de laboratorios de Redes de Computadoras?

El Departamento de Computación cuenta con 4 laboratorios ampliamente equipados con PCs, sin embargo, a como se ha comentado antes, no se cuenta con suficientes equipos de redes como: Routers y Switches, necesarios para realizar prácticas de laboratorios reales.

Por tal razón, se utilizan simuladores y emuladores software, combinados con distribuciones de Linux pensadas para hacer muchas funciones de quipos reales.

Aún con todo, el uso estas herramientas, no terminan de dar una noción totalmente real al estudiante de la configuración de equipos físicos. Como consecuencia de esto, se ha investigado diferentes herramientas de libre distribución que sean lo más cercana posible a los equipos reales.

3. Justificación.

En base a los problemas antes planteados se hace necesario evaluar alternativas de programas **Open Source Routing,** que ayuden a realizar prácticas de laboratorios más reales en la asignatura de Redes de Computadoras.

Por ende, primero se hará una descripción de diferentes herramientas de **Open Source Routing** entre ellas Vyatta, Zebra, XORP, Quagga y BIRD. Además se realiza una breve explicación de qué es cada herramienta, cómo surgió, qué servicios ofrece y si está constantemente en desarrollo.

Por último, se realizan algunas prácticas de Laboratorios con el sistema Vyatta, enfocadas a la asignatura de Redes de Computadoras demostrando que el soporte de protocolos de red de esta herramienta, son prácticamente iguales que muchos enrutadores y Switches reales de gran envergadura.

4. Objetivos.

4.1 General:

Fundamentar que el programa Vyatta es una alternativa para el uso en los laboratorios del área de Redes de Computadores.

4.2 Específicos:

- Describir las principales características de algunas herramientas de **Open Source Routing** existentes y detallar los requerimientos necesarios para su instalación y configuración.

- Demostrar empíricamente que Vyatta es el programa que posee las características que mejor se adecuan a los laboratorios de la asignatura de Redes de Computadores.

- Desarrollar prácticas de laboratorios con Vyatta, en las que se demuestren las capacidades y funcionalidades de esta herramienta en las distintas capas del protocolo TCP/IP.

Capítulo II: Marco Conceptual

1. Presentación

En el siguiente capítulo se desarrollan todos los aspectos teóricos que conforman la base y el fundamento para la elaboración del trabajo.

1.1 Vyatta:

Vyatta ha creado una de las mejores soluciones de red, la cual está optimizada para ofrecer seguridad integrada y funcionalidad de enrutamiento para entornos físicos y virtuales. El sistema Vyatta se diseñó con el propósito de reemplazar las versiones comerciales de Cisco, desde la serie 1800 hasta la 7200, apoyándose en el bajo coste y flexibilidad de las soluciones open source, y que además al estar basado en Linux funcionaría en la mayoría de sistemas basados en la tecnología x86.

En la versión VC3, Vyatta empieza a utilizar XORP, en concreto su modelo de pila de enrutamiento, y es patrocinado por éste durante 2 años. Más adelante, en la versión VC4, el equipo de Vyatta toma la decisión de ir en otra dirección. A partir de ahí ya no deciden utilizar XORP, porque quieren dotar a Vyatta de más funciones (firewall, VPN, DHCP, VLAN, etc.) las cuales XORP no dispone.

A principios de 2007, Vyatta tenía un gran número de clientes, los cuales comienzan a utilizar el sistema en redes de proveedores de servicios, en los que había que implementar el protocolo BGP. Como el despliegue de redes era tan grande, se encontraron con una serie de limitaciones en el código fuente de XORP relacionados con la escalabilidad y el rendimiento. Para solucionar esto, los ingenieros de Vyatta hicieron muchas reformas en el código fuente de XORP, y al final se determinó que los costes eran mayores que si lo desarrollaban ellos mismos, por lo que se inició el trabajo para realizar un nuevo diseño de pila de enrutamiento. Así nació la versión VC4 con la colaboración de Quagga.

Vyatta ofrece dos opciones de uso, los dispositivos hardware o el software Vyatta Network OS. Los aparatos hardware, integran el software Vyatta Network OS y software de seguridad. Estos dispositivos están basados en las plataformas estándar de hardware x86, y eliminan cualquier dependencia de hardware propietario. En concreto, ofrecen 4 tipos de dispositivos hardware de diferentes características técnicas, que varían según las necesidades del comprador.

Vyatta se puede instalar tanto sobre un servidor físico como en una máquina virtual, permitiendo disponer de servicios de seguridad de red en entornos virtualizados y entornos en nube. Vyatta ofrece a los proveedores de los servicios de nube la opción de asegurar y gestionar sus complejas redes ofreciendo mucho más que un simple firewall, en concreto, VPN con IPsec, SSL en OpenVPN, prevención de intrusiones de red, seguridad de filtrado web y mucho más.

El sistema Vyatta ofrece las siguientes funcionalidades:
Servicios IP [6]:

- ✓ SSH.
- ✓ Telnet.
- ✓ DHCP.
- ✓ DNS.
- ✓ NAT.
- ✓ Web caching.

Enrutamiento básico:

- ✓ Políticas de enrutamiento.
- ✓ RIP.
- ✓ BGP.
- ✓ OSPF.

Servicios de seguridad:

- ✓ Firewall: IPv4, IPv6.
- ✓ Detección de intrusiones (snort) y filtrado web.
- ✓ Servicios de Red Privada Virtual (VPN):
- ✓ VPN Lan-to-Lan con IPSec.
- ✓ Acceso remoto con PPTP.
- ✓ Acceso remoto con L2TP e IPSec.
- ✓ VPN Lan-to-Lan y acceso remoto con OpenVPN.

QoS (Quality of Services):

- ✓ Traffic shape: permite controlar y restringir la anchura de banda (bandwith) de las conexiones salientes.
- ✓ Traffic límite: permite controlar y restringir la anchura de banda (bandwith) de las conexiones entrantes.

Servicios de alta disponibilidad:

- ✓ WAN Load balancing: permite encriptar el tráfico saliente a través de diferentes proveedores.
- ✓ VRRP (Virtual Router Redundancy Protocol): permite que un cluster de Vyattas actúen como un único dispositivo.
- ✓ Clustering: permite disponer de alta disponibilidad entre dos servidores Vyatta.
- ✓ Statefull NAT and Firewall Failover: permite replicar el estado de las conexiones entre los servidores de un cluster, de manera que en caso de error de un nodo, las conexiones de mantienen activas.
- ✓ RAID 1: permite implementar RAID 1 de discos, tanto para hardware como para software.
- ✓ Configuration Synchronization: sincroniza la configuración entre dos nodos en alta disponibilidad.

1. 2 XORP

XORP (eXtensible Open Router Platform) es una plataforma de código abierto que proporciona todas las funciones que implementan los protocolos de enrutamiento IPv4 e IPv6. Es la única plataforma que ofrece una capacidad integrada de multidifusión.

El desarrollo del proyecto fue fundado por Mark Handley en el año 2000. Recibió fondos de Intel y Microsoft y el software se lanzó al mercado por primera vez en julio del 2004. La última actualización se hizo en enero del 2012, por lo que podemos comprobar que es un software que está en constante desarrollo. Esto es importante, ya que los errores que puedan surgir al usuario, se puedan solucionar en nuevas actualizaciones.

Entre sus principales objetivos de XORP, está la tarea de soportar todos los requisitos de los Routers reales incluyendo los protocolos de enrutamiento, facilitar las APIs para crear extensibilidad en los protocolos de enrutamiento, mejorar el rendimiento del router y hacerlo robusto para que los paquetes no se colapsen o pierdan.

XORP puede convertirse en una alternativa de bajo coste en comparación a las costosas máquinas de Cisco Systems y otras empresas que dominan el mercado de enrutamiento, ya que se puede montar sobre PCs convencionales.
La base de código XORP consta de alrededor 670.000 líneas de código escritas en lenguaje C++ y desarrolladas principalmente en FreeBSD, pero totalmente compatibles con Linux y OpenBSD.

Según sus creadores soporta protocolos como RIP, BGP, OSPF en procesos independientes lo que significa que si un componente de software cae, no afectará a los otros que están utilizando el mismo hardware.
El proyecto soporta los siguientes protocolos de enrutamiento:

- ✓ Enrutamiento estático.
- ✓ RIP-2 y RIPng.
- ✓ BGP-4.
- ✓ OSPFv2 y v3.

- ✓ PIM-SM.
- ✓ IGMP v1, v2 y v3
- ✓ MLD v1 y v2.
- ✓ VRRP v2.

El lenguaje de líneas de comandos es el modelo de plataforma de Junipers Networks el cual funciona bajo el sistema operativo UNIX.

1. 3 Quagga

Es un sistema más fácilmente extensible, gestionado y modular. Quagga copia el estilo del modo de trabajo del CISCO IOS y tiene un proceso independiente para cada protocolo. Un sistema con Quagga instalado actúa como un router dedicado.

Quagga intercambia información con otros routers mediante protocolos de routing, utilizándola para actualizar el núcleo de las tablas de routing de forma que la información correcta esté en el lugar correcto. Quagga permite la configuración dinámica y es posible ver la información de la tabla de routing desde el interfaz de terminal de Quagga.

El proyecto soporta los siguientes protocolos de enrutamiento:

- ✓ RIPv1, v2, RIPng.
- ✓ BGP-4 y BGP-4 +.
- ✓ OSPFv2 y v3.

Además de soportar ipv4, también soporta ipv6. Posee una arquitectura avanzada que le proporciona una gran calidad y potencia, con un motor multiservidor de encaminamiento.

Quagga tiene un interfaz de usuario interactivo para cada protocolo de routing.

Está diseñado especialmente para NetBSD, FreeBSD, Solaris y Linux y solamente se tendrá que instalar el paquete quagga.

1. 4 BIRD

El nombre de “BIRD” es un acrónimo de “Internet Routing Daemon" que significa demonio de encaminamiento de internet.

BIRD se desarrolló en la facultad de matemáticas y física de la Universidad Charles en Praga, como un proyecto estudiantil en 1999. El proyecto estuvo detenido un tiempo hasta el 2003.

Puede ser libremente distribuido bajo los términos de la Licencia Publica General GNU y es una alternativa a Quagga/Zebra.

BIRD ha sido diseñado para trabajar en todos los sistemas tipo UNIX, FreeBSD, NetBSD y OpenBSD.

BIRD es un proyecto cuyo cometido es el desarrollo de un completo demonio de ruteo de IP, cuyo modo de acceso es a través de línea de comando y diseñado especialmente para ajustarse a cualquier sistema operativo de tipo UNIX. BIRD es compatible con el protocolo IPv4 o IPv6, aunque tendrán que ser compilados de forma separada ya que no son compatibles al mismo tiempo.

Protocolos de enrutamiento soportados:

- ✓ Rutas estáticas.
- ✓ RIPv2.
- ✓ BGPv4.
- ✓ OSPFv2 y v3.

BIRD brinda un mecanismo de configuración realmente flexible y fácil de maniobrar, basado en un fichero de configuración, que se activará mediante un SIGHUP.
En definitiva, BIRD ofrece un potente lenguaje para la aplicación de filtros de “ruteo” de IP dinámica, pero también de rutas estáticas, y con múltiples tablas de routing, ideal para dominar el conjunto de protocolos de “ruteo” vigentes en la red de redes hoy en día.

1. 5 Zebra

GNU Zebra es un software libre que gestiona protocolos de enrutamiento basados en TCP/IP. Es un proyecto libre, ya que forma parte del Proyecto GNU, y se distribuye bajo la Licencia Pública General GNU.

El proyecto Zebra se inició en 1996. La idea de Zebra vino originalmente de Kunihiro Ishiguro, quien había estado trabajando en NIS y en una empresa conjunta entre el ISP British Telecom y Marubeni. Cuando estaba trabajando para un ISP, se dio cuenta de una gran necesidad de un nuevo tipo de software de enrutamiento. Junto con Yoshinari Yoshikawa decidieron crear un nuevo motor de enrutamiento basado en licencia pública GNU.

Zebra es un software que permite montar routers sobre sistemas operativos tipo Unix. Este software dispone de una interfaz de configuración basada en el CISCO IOS, por lo que será útil a los administradores familiarizados con routers CISCO.

Tradicionalmente, la configuración de un router basado en UNIX se realizaba mediante los comandos ifconfig y route. El estado de las tablas se podía mostrar mediante la utilidad netstat. Estos comandos solamente se podían utilizar trabajando como root. Zebra, sin embargo tiene otro método de administración. En Zebra existen dos modos de usuario. Uno es modo normal y otro es el modo enable (habilitado). El usuario normal solo puede ver el estado del sistema y el enable puede cambiar la configuración del sistema. Como vemos esto se asemeja al modo de trabajo de los routers CISCO, en el que tendremos que entrar en el modo configuración para poder realizar los cambios.

El software tradicional de routing está compuesto por un programa o un proceso único que proporciona todas las funcionalidades de los protocolos de routing. Zebra sin embargo tiene una visión distinta. Está compuesto por una colección de varios demonios que trabajan juntos para construir una tabla. El demonio ripd maneja el protocolo RIP, mientras que el demonio ospfd controla el protocolo OSPFv2 y bgpd el protocolo BGP-4. Es sencillo añadir nuevos demonios de routing sin afectar a otros.

Los protocolos de enrutamiento que Zebra soporta son:

- ✓ RIPv1, v2, RIPng.
- ✓ BGP-4 y BGP-4.
- ✓ OSPFv2 y v3.

Esta opción no ofrece todos los servicios ni protocolos de enrutamiento necesarios para realizar las prácticas de LPR, y por lo tanto no se podría afirmar que este software podría sustituir completamente a los router CISCO. Desde su última actualización en Septiembre del 2005 el desarrollo de Zebra se ha detenido para dar lugar a su nuevo sucesor, Quagga.

Capítulo III: Diseño Metodológico.

Como primer paso se investigan diferentes programas de enrutamiento de código abierto que existen – estas fueron Vyatta, XORP, Quagga, Zebra y BIRD - y que tuvieran capacidad para trabajar con algunos protocolos que se utilizan para resolver prácticas frecuentes en los laboratorios de Redes de Computadoras.

Luego de la instalación y de la revisión de las especificaciones de estas herramientas (Verificación de protocolos de red y servicios IP soportados), se llegó a la conclusión de trabajar con Vyatta porque implementa completamente una gran variedad de protocolos y servicios que se utilizan en los laboratorios de Redes de Computadoras.

Finalmente se realizan y se explican algunas prácticas de laboratorios relacionadas con protocolos de encaminamiento (BGP, RIP y OSPF), servicios de red (DHCP, NAT, SQUID) y VLAN.

Capítulo IV: Casos de Estudio con Vyatta.

A continuación para mostrar parte de las tecnologías implementadas se muestra una tabla en la que se ven reflejadas las distintas características de los **Open Source Routing** y una versión del dispositivo comercial más popular del mercado, CISCO.

	XORP 1.8.5	**Quagga 0.99.18**	**BIRD 1.3.2**	**VYATTA VC 6.4**	**CISCO 2651**
Sistema Operativo	Linux	Unix-like	Unix-like	Linux	CISCO IOS
Open Source	SI	SI	SI	SI	NO
Instalación	SI/NO	SI	SI	SI/NO	NO
Rutas estáticas	SI	SI	SI	SI	SI
RIPv2	SI	SI	SI	SI	SI
OSPFv3	SI	SI	SI	SI	SI
BGPv4	SI	SI	SI	SI	SI
NAT	SI	SI	NO	SI	SI
VRRP	SI	NO	NO	SI	SI
Mapas de rutas	SI	SI	SI	SI	SI
VPN IPsec	NO	NO/Linux	NO	SI	SI
VPN SSL	NO	NO/Linux	NO	SI	NO
Cliente FTP	NO/Linux	NO/Linux	NO/Linux	SI	SI
Cliente TFTP	NO/Linux	NO/Linux	NO/Linux	SI	SI
Telnet	SI	SI	SI	SI	SI
Server SSH	NO/Linux	NO/Linux	NO/Linux	SI	SI/NO
Server HTTP	NO	NO	NO	SI	SI
Server DHCP	NO/Linux	NO/Linux	NO/Linux	SI	SI
Server NTP	NO/Linux	NO/Linux	NO/Linux	NO	SI/NO

	XORP 1.8.5	Quagga 0.99.18	BIRD 1.3.2	VYATTA VC 6.4	CISCO 2651
Cliente NTP	NO/Linux	NO/Linux	NO/Linux	SI	SI
SNMP	NO	NO	NO	SI	SI
ICMP	SI	SI	SI	SI	SI
Traceroute	SI	SI	SI	SI	SI

Las características mínimas para instalar el software Vyatta en un PC son:

- Procesador: Pentium III a 500 MHz
- Memoria RAM: 384 MB
- Disco Duro: 10 Gb

Haciendo una comparación de los servicios ofrecido por el sistema Vyatta contra XORP, mostramos que esta segunda herramienta solo tiene soporte a protocolos de enrutamiento, no cuenta con ningún servicio, también mencionar que XORP está disponible para su descarga como un Live CD o como código fuente a través de la página oficial del proyecto.

Por otra parte la herramienta Quagga es similar a XORP en la parte de soporte de protocolos, pero esta no cuenta con su propio sistema, debe descargar los paquetes, que posteriormente se tendrán que instalar en una distribución de Linux, los cuales actuarán como demonios independientes.

Finalmente se comparó contra BIRD, cabe mencionar que este al igual que Quagga cuenta con un archivo de configuración de paquetes pero debe de descargarse del sitio oficial un archivo tar.gz, no omitir mencionar que se tuvieron problemas a la hora de configurar esta herramienta.

Como conclusión hay decir que VYATTA posee las mismas funcionalidades que XORP, QUAGGA Y BIRD, e incorpora en el software, servicios IP como ssh, telnet, dhcp, dns, nat, ftp, tftp.

Además, Vyatta proporciona servicios de red privada virtual (VPN), QoS, VRRP, seguridad firewall y detección de intrusiones. Todo esto hace que Vyatta sea un duro competidor para diferentes tecnologías.

VYATTA es una, o quizás la mejor opción para disponer de un router de manera gratuita (solo nos hace falta un ordenador con unas características mínimas).

NOTAS:

Unix-like: es un sistema operativo que se comporta de manera similar a un sistema Unix, aunque no necesariamente conforme a su certificación.

No/Linux: significa que el software en cuestión no trae esa opción, pero al funcionar sobre el sistema operativo Linux, éste sí que dispone de esa característica.

Capítulo V: Prácticas Propuestas.

1. Práctica # 01

Implementación DHCP, NAT, PROXY

Objetivos:

- Describir los pasos necesarios para la configuración del esquema de red que se desarrollará.
- Usar los comandos de configuración de Vyatta para configurar un servidor DHCP – NAT – PROXY.

Introducción.

En esta práctica se pretende que el estudiante aplique todos los conocimientos teóricos que haya obtenido en el desarrollo del tema, su configuración y funcionamiento.

El protocolo DHCP es muy utilizado en la administración de redes, ya que permiten que los equipos configuren automáticamente los parámetros de red (dirección IP, mascara de red, dirección de difusión, encaminador por defectos, servidor de nombres).

La configuración NAT es necesaria por ser una alternativa ante la escasez de direcciones IP. NAT permite acceder a internet traduciendo las direcciones privadas en direcciones públicas.

La configuración del servidor Proxy es necesario porque este actúa como intermediario entre el ordenador de un usuario e internet, esto ayudaría a tener un mejor control de administración de una red y brindaría mayor seguridad a la misma.

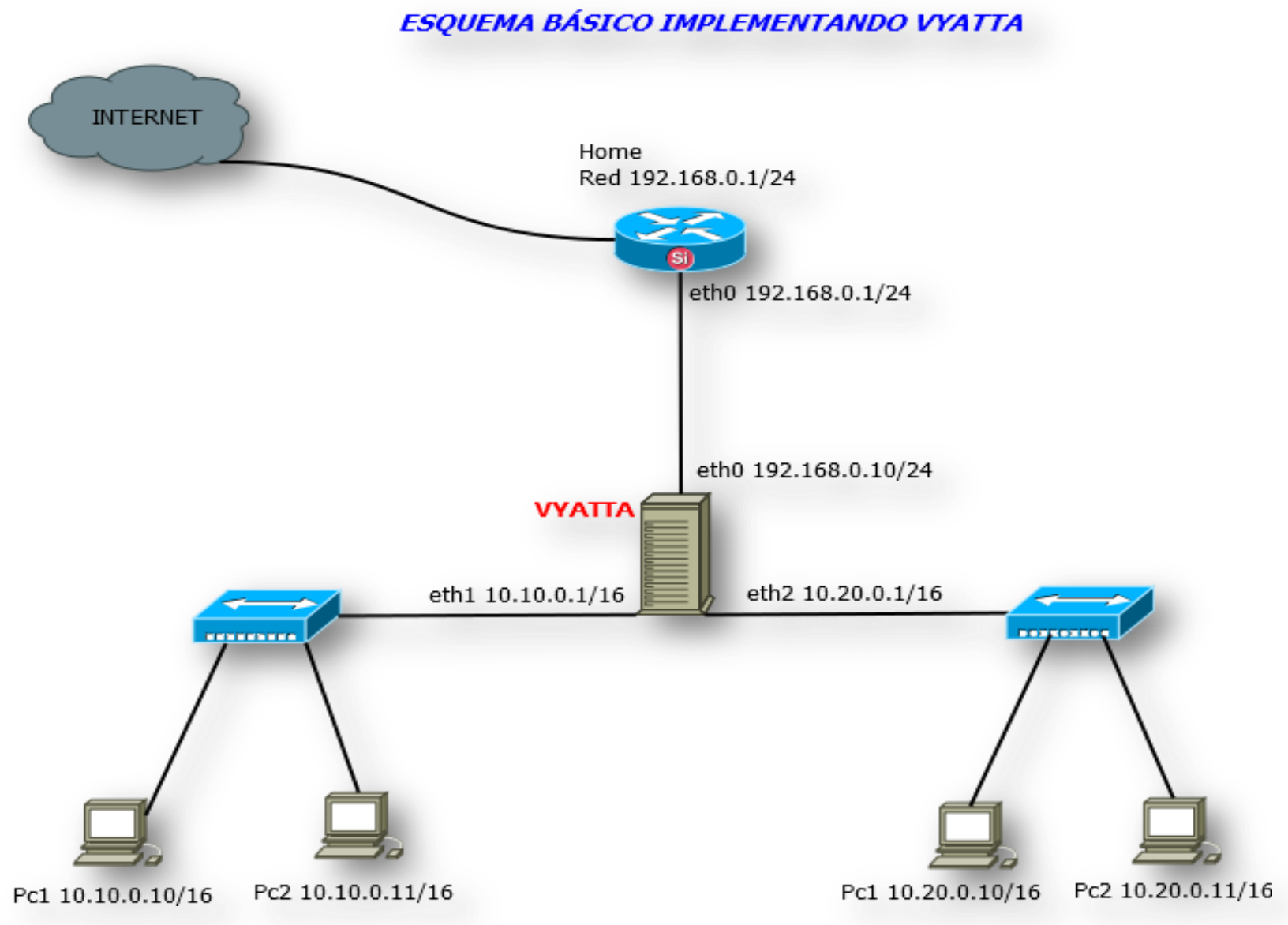


Figura1. Implementación DHCP - NAT en Vyatta 6.4

En la Figura1. Mostramos un esquema muy sencillo el cual fue de mucha utilidad para comprender configuración básica de Vyatta.

El primer paso consiste en instalar en una máquina virtual Vyatta 6.4, cabe mencionar en este caso se hicieron pruebas tanto en VirtualBox, como en VMWARE el cual funciono como se esperaba. Después de instalar el sistema Vyatta se debe configurar 3 interfaces de Red, cabe recalcar que VirtualBox permite hacer esto utilizando o creando adaptadores anfitriones.

Al arrancar VYATTA e introducir las credenciales, se muestra lo siguiente:

```
vyatta@vyatta:~$
```

Es el modo operacional, con el que se pueden realizar tareas como mostrar información, reiniciar y apagar el sistema, etc.

El siguiente paso que habrá que mirar será ver si el equipo actúa como router. Aunque esta opción viene por defecto activada, se puede comprobar que realmente lo está con la siguiente instrucción:

```
vyatta@vyatta:~$ show ip forwarding
```

A lo que el sistema responde "***IP forwarding is on".*** *Otra forma seria mostrando el ip_forward que trae integrado vyatta.*

```
root@vyatta:/home/vyatta# cat /proc/sys/net/ipv4/ip_forward
1
root@vyatta:/home/vyatta#
```

Figura2. Verificar Vyatta 6.4 para uso Ip forwarding

Como se comentó en la parte del manual, para realizar cualquier cambio en la configuración del router Vyatta, hay que entrar en modo configuración mediante el comando:

```
vyatta@vyatta:~$ configure
```

Al realizar esa acción, ahora se verá el prompt así

```
vyatta@vyatta#
```

A continuación, se va a configurar la dirección IP de uno de los interfaces Ethernet. Al contrario que en CISCO en el que se tenía que pasar al modo de configuración del interfaz que se quería modificar y después activarlo, en Vyatta, solamente mediante un comando se realizará la configuración.

```
vyatta@vyatta# set interfaces ethernet ethx address IP/netmask
```

En donde ethx corresponde al interfaz que se quiera modificar, IP a la dirección de red y netmask a la máscara de red en formato CIDR.

Para realizar una prueba, se va a realizar el escenario de la Figura3.

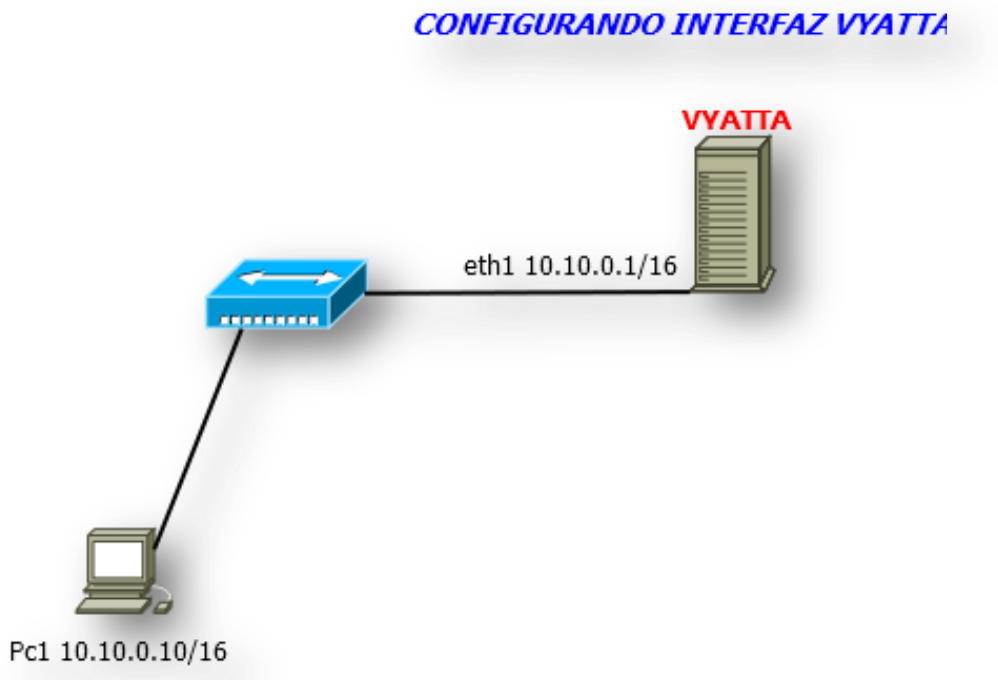


Figura3. Configuración de Interfaz Vyatta

Como está basado en GNU/Linux, la manera de nombrar las interfaces de red es mediante ethX, con X igual a 0 para la primera interfaz que reconoce el sistema, 1 la segunda, etc. Para saber las interfaces que tiene el router, se ejecutará el comando:

Este comando es muy útil, ya que gracias a él se verá el estado de los interfaces, si tienen asignada una dirección IP y alguna descripción.

El siguiente paso es configurar las interfaces para asignarles una IP. Por ejemplo, para asignar la dirección 192.168.0.10/24 a la interfaz eth0,10.10.0.1/16 a la interfaz eth1,10.20.0.1/16 a la interfaz eth2.

```
vyatta@vyatta:# set interfaces Ethernet eth0 address 192.168.0.10/24
vyatta@vyatta:# set interfaces Ethernet eth1 address 10.10.0.1/16
vyatta@vyatta:# set interfaces Ethernet eth2 address 10.20.0.1/16
```

Se puede configurar distintos parámetros como el nombre del host, el nombre de dominio, etc.

En este punto cabe destacar, que en el modo configuración, la instrucción ***set*** sirve para marcar cambios, pero éstos no se hacen efectivos hasta que no se confirmen con ***commit***. Esto se hace como medida de seguridad. Con la instrucción ***show*** se podrán ver los cambios que se hayan realizado, y aquellos que no estén confirmados aparecerán con un símbolo „+" por delante. Para comprobar todos los parámetros que se hayan configurado, se puede utilizar la instrucción ***show***, esto en modo no privilegiado.

```
root@vyatta:/home/vyatta# show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.10/24                   u/u
eth1             10.10.0.1/16                      u/u
eth1.50          10.50.0.1/16                      u/u
eth2             10.20.0.1/16                      u/u
lo               127.0.0.1/8                       u/u
                 ::1/128
root@vyatta:/home/vyatta#
```

Figura4. Mostrar Interfaces de Red Asignadas en Vyatta 6.4

En concreto si se quiere ver la configuración de los interfaces de red en modo privilegiado, se verá algo como esto:

```
root@vyatta# show interfaces
 ethernet eth0 {
     address dhcp
     duplex auto
     hw-id 00:0c:29:6a:b2:2f
     smp_affinity auto
     speed auto
 }
 ethernet eth1 {
     address 10.10.0.1/16
     duplex auto
     hw-id 00:0c:29:6a:b2:39
     smp_affinity auto
     speed auto
     vif 50 {
         address 10.50.0.1/16
     }
 }
 ethernet eth2 {
     address 10.20.0.1/16
     duplex auto
     hw-id 00:0c:29:6a:b2:43
     smp_affinity auto
     speed auto
```

Figura5. Visualizar Interfaz de Red desde la consola Vyatta 6.4

Como se ve, esta instrucción mostraría todas las características de las interfaces que se han configurado. En este caso se muestra las3 interfaz Ethernet eth0, eth1 y eth2.

Luego asignamos una IP a una PC prueba para verificar que funciona la comunicación.

```
root@debian:/home/lester# ping 10.10.0.1
PING 10.10.0.1 (10.10.0.1) 56(84) bytes of data.
64 bytes from 10.10.0.1: icmp_req=1 ttl=64 time=1.14 ms
64 bytes from 10.10.0.1: icmp_req=2 ttl=64 time=1.93 ms
64 bytes from 10.10.0.1: icmp_req=3 ttl=64 time=2.25 ms
64 bytes from 10.10.0.1: icmp_req=4 ttl=64 time=1.51 ms
^C
--- 10.10.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.146/1.712/2.257/0.421 ms
root@debian:/home/lester# _
```

Figura6. Realizar ping al Gateway para Verificar funcionamiento.

En la figura anterior vemos que todo funciona correctamente. Ahora lo que vamos hacer será configurar un servicio por DHCP para que los equipos que se conecten a nuestro VYATTA obtengan dirección de forma dinámica. Para eso haremos lo siguiente en nuestro equipo VYATTA:

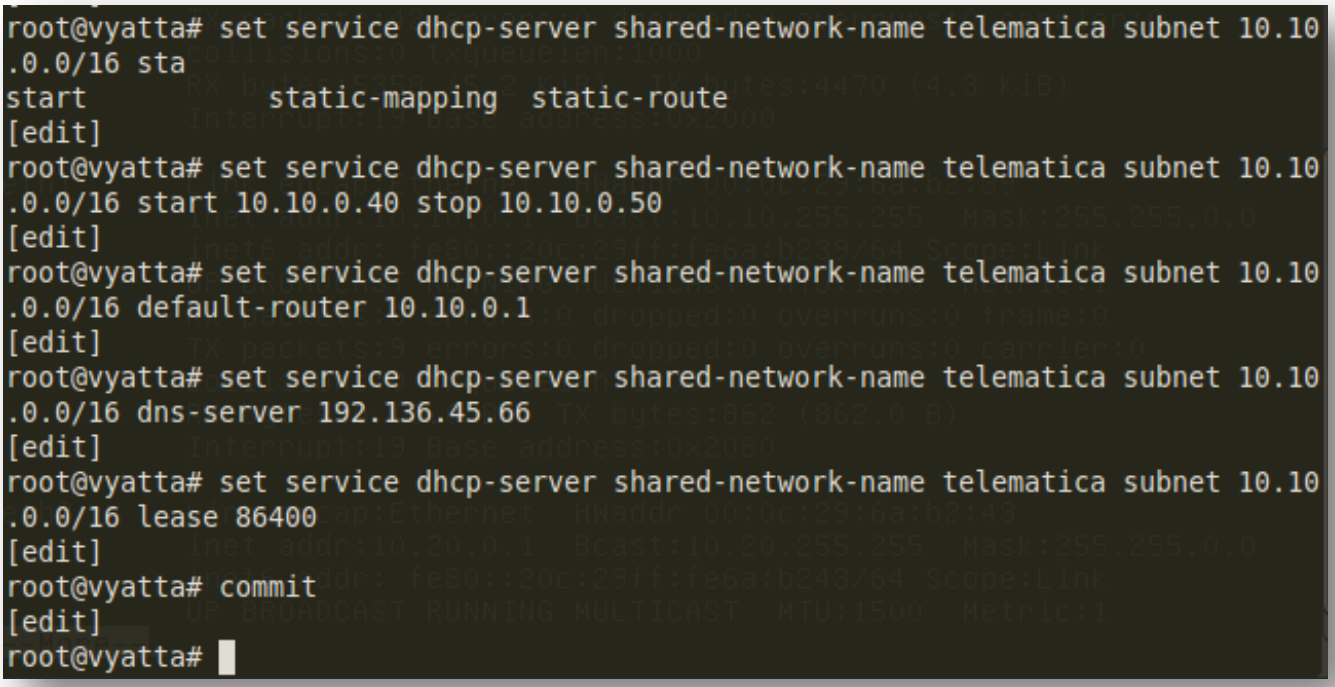


Figura7. Activar y Configurar Servicio DHCP en Vyatta 6.4

Lo que hicimos con los comandos mostrados anteriormente fue crear un dominio o pool que en este caso le llamamos ***telemática*** y le agregamos una subred 10.10.0.0/16, además decimos el rango de direcciones que este asignara en este caso 10.10.0.40 - 10.10.0.50 por defecto decimos que su Gateway será 10.10.0.1, también tendrá un dns 192.136.45.66 y por ultimo agregamos un tiempo de vida o lease a la dirección que asigne. Acá solamente se muestra que se un solo pool para DHCP pero tendremos que crear otro para luego verificar que hay comunicación entre segmentos diferentes de red. A este otro le llamaremos ***telemática*** es lo mismo pero cambiamos los segmentos de red y por último es importante enviar las modificaciones que hallamos realizado con el comando ***commit.***

Ahora vamos verificar como quedo nuestro servicio DHCP configurado, para eso usamos el siguiente comando:

```
[edit]
root@vyatta# show service dhcp-server
 disabled false
 shared-network-name sistema {
     subnet 10.20.0.0/16 {
         default-router 10.20.0.1
         dns-server 192.136.45.66
         lease 86400
         start 10.20.0.40 {
             stop 10.20.0.50
         }
     }
 }
 shared-network-name telematica {
     subnet 10.10.0.0/16 {
         default-router 10.10.0.1
         dns-server 192.136.45.66
         lease 86400
         start 10.10.0.40 {
             stop 10.10.0.50
         }
     }
 }
[edit]
root@vyatta#
```

Figura8. Mostrar Configuración de Servicio DHCP en Vyatta 6.4

Ahora verificamos que funcione bien en este caso PC1 lo agregamos al adaptador vmnet1 y PC2 al adaptador vmnet2 todo de tipo anfitrión esto en VMWARE, cabe mencionar que el adaptador anfitrión ayuda a simular como si fuera un Switch.

```
root@debian:/home/lester# ifconfig  eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:1a:7c:b9
          inet addr:10.20.0.40  Bcast:10.20.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:fe1a:7cb9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1550 (1.5 KiB)  TX bytes:1562 (1.5 KiB)
          Interrupt:19 Base address:0x2000

root@debian:/home/lester# _
```

```
root@debian:/home/lester# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:b0:cc:a3
          inet addr:10.10.0.40  Bcast:10.10.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:feb0:cca3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1550 (1.5 KiB)  TX bytes:1562 (1.5 KiB)
          Interrupt:19 Base address:0x2000

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:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:560 (560.0 B)  TX bytes:560 (560.0 B)

root@debian:/home/lester# _
```

Figura9. Verificar IP asignadas a PCs

Podemos notar que ambos equipos tienen asignado dirección IP correctamente, también podemos mostrar el Gateway de cada equipo con el siguiente comando:

```
root@debian:/home/lester# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.10.0.0       0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         10.10.0.1       0.0.0.0         UG    0      0        0 eth0
root@debian:/home/lester# _
```

```
root@debian:/home/lester# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.20.0.0       0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         10.20.0.1       0.0.0.0         UG    0      0        0 eth0
root@debian:/home/lester# _
```

Figura10. Verificar Gateway asignado a PCS

Hasta este momento tenemos configurado un servidor DHCP, pero falta algo muy importante, debemos configurar NAT para que las PC que están conectadas a nuestro VYATTA puedan comunicarse entre diferentes segmentos y también puedan navegar

hacia internet sin problemas. Antes de continuar configurando NAT vamos a configurar el servicio SSH en nuestro VYATTA esto para que podamos acceder de forma remota.

```
vyatta@vyatta:# set services ssh port 2022
vyatta@vyatta:# set services allow-root
vyatta@vyatta:# commit
```

Verificamos lo que hicimos:

```
root@vyatta# show service ssh
 allow-root
 port 2022
 protocol-version v2
[edit]
root@vyatta#
```

Figura11. Mostrar Configuración de Servicio SSH en Vyatta 6.4

Por ultimo accedemos de forma remota, cabe mencionar que para tener mayor seguridad cambiamos el puerto de acceso que por defecto es el 22.

```
root@Info:/home/lester# ssh -p 2022 vyatta@192.168.0.10
The
tablished.
RSA key fingerprint is 73:75:cf:29:fd:d0:da:08:c8:c9:27:20:35:d8:cb:db.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.0.10]:2022' (RSA) to the list of known host
s.
Welcome to Vyatta
vyatta@192.168.0.10's password:
Linux vyatta 3.0.23-1-586-vyatta #1 SMP Fri Mar 23 19:00:31 PDT 2012 i686
Welcome to Vyatta.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
Last login: Thu Nov 15 20:27:26 2012
vyatta@vyatta:~$ sudo su
root@vyatta:/home/vyatta#
root@vyatta:/home/vyatta#
```

Figura12. Acceder a través de SSH mediante configuración de puertos

Todo funcionó perfectamente ahora seguimos con la configuración de NAT, para esto lo haremos usando excluded o exclusiones ya que a veces es deseable excluir los paquetes de traducción NAT que responden a ciertos criterios.

Lo primero creamos una regla de control de acceso.

```
vyatta@vyatta:# set nat source rule 10
```

Ahora aplicamos la regla a paquetes que vienen desde la dirección 10.0.0.0/8 con destino 192.168.0.0/24 a través de la interfaz eth0.

```
vyatta@vyatta:# set nat source rule 10 source address 10.0.0.0/8
vyatta@vyatta:# set nat source rule 10 destination address 192.168.0.0/24
vyatta@vyatta:# set nat source rule 10 outbound-interface eth0
```

Ahora excluimos paquetes de NAT que vienen incluido en la regla 10.

```
vyatta@vyatta:# set nat source rule 10 excluded
```

Ahora creamos otra regla de acceso (20) y aplicamos la regla a paquetes que vienen desde la dirección 10.0.0.0/8 que saldrán a través de la interfaz eth0.

```
vyatta@vyatta:# set nat source rule 20
vyatta@vyatta:# set nat source rule 20 source address 10.0.0.0/8
vyatta@vyatta:# set nat source rule 20 outbound-interface eth0
```

Ahora se hace un enmascaramiento para que permita comunicación entre segmentos de red y haga una traducción de paquetes a través de la interfaz de salida y guardamos cambios.

```
vyatta@vyatta:# set nat source rule 20 translation address masquerade
vyatta@vyatta:# commit
```

Ahora podemos hacer algunas pruebas haciendo ping de las PC pruebas hacia google y gmail para verificar que NAT está funcionando de forma correcta.

```
root@debian:/home/lester# ping www.google.com
PING www.google.com (74.125.130.104) 56(84) bytes of data.
64 bytes from gh-in-f104.1e100.net (74.125.130.104): icmp_req=1 ttl=127 time=188
 ms
64 bytes from gh-in-f104.1e100.net (74.125.130.104): icmp_req=2 ttl=127 time=187
 ms
64 bytes from gh-in-f104.1e100.net (74.125.130.104): icmp_req=3 ttl=127 time=172
 ms
^C
--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 172.534/182.806/188.409/7.273 ms
root@debian:/home/lester# _
```

```
root@debian:/home/lester# ping www.gmail.com
PING googlemail.l.google.com (74.125.229.181) 56(84) bytes of data.
64 bytes from mia04s04-in-f21.1e100.net (74.125.229.181): icmp_req=1 ttl=127 tim
e=183 ms
64 bytes from mia04s04-in-f21.1e100.net (74.125.229.181): icmp_req=2 ttl=127 tim
e=181 ms
^C64 bytes from mia04s04-in-f21.1e100.net (74.125.229.181): icmp_req=3 ttl=127 t
ime=162 ms

--- googlemail.l.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 162.483/175.670/183.343/9.379 ms
root@debian:/home/lester# _
```

Figura13. Verificar NAT en Vyatta 6.4 a través de un ping

También podemos verificar el servicio NAT en VYATTA escribiendo el comando **show nat:**

```
[edit]
root@vyatta# show nat
 destination {
 }
 source {
     rule 10 {
         destination {
             address 192.168.0.0/24
         }
         exclude
         outbound-interface eth0
         source {
             address 10.0.0.0/8
```

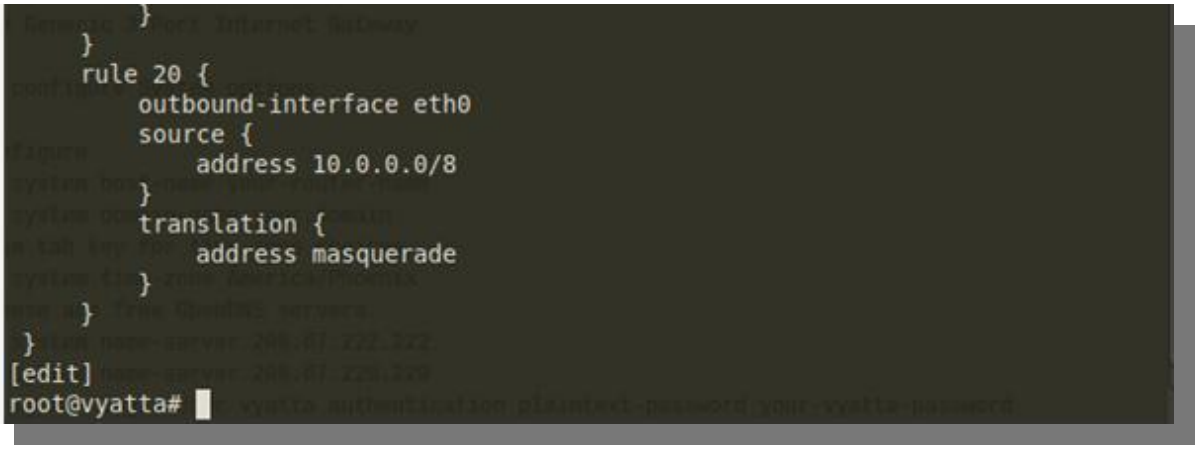


Figura14. Mostrar Configuración NAT en Vyatta 6.4

Ahora vamos a implementar **SQUID** en Vyatta esto con el fin de bloquear algunas url y palabras no deseadas en nuestra red.

Lo primero es habilitar el servicio WEBPROXY en nuestro servidor.

```
vyatta@Central:# set service webproxy
```

A continuación, configuramos por cual dirección IP el servicio se mantendrá a la escucha de las peticiones clientes. En nuestro caso, la dirección IP que se encuentra escuchando en la red interna configurada en nuestro Vyatta es*10.10.0.1*. Para ello ejecutamos siguiente comando:

```
vyatta@Central:# set service webproxy listen-address 10.10.0.1
vyatta@Central:#commit
vyatta@Central:#save
```

Si no deseásemos dicho filtrado y sólo un servidor proxy para mejorar el rendimiento de la conexión a Internet usando la cache almacenada, sería suficiente con los pasos realizados.

El siguiente paso es descargar las listas negras que usará nuestro filtrado Web si las activásemos. Para poder efectuar dicha descarga tenemos que estar fuera del modo configuración y ejecutar el comando **update webproxy blacklists** y confirmamos el mensaje.

```
vyatta@Central:~$update webproxy blacklists
vyatta@Central:#
```

Ahora vamos activar el filtrado Web esto con el fin de bloquear algunas páginas web.

```
vyatta@Central:# set service webproxy url-filtering squidguard
vyatta@Central:#commit
vyatta@Central:#save
```

El siguiente paso es configurar nuestro filtrado Web, nosotros activaremos todas las categorías disponibles en las listas negras, para ello ejecutamos el comando **set service webproxy url-filtering squidguard block-category all**. Aplicamos los cambios con **commit** y grabamos la configuración con **save**.

```
vyatta@Central:# set service webproxy url-filtering squidguard
block-category all

vyatta@Central:#commit
vyatta@Central:#save
```

Con la configuración realizada ya se dispone de Vyatta con filtrado Web. Si queremos realizar otros tipos de configuraciones podemos utilizar los siguientes comandos dentro de la sentencia **set service webproxy url-filtering squidguard**:
Ahora para filtrar www.youtube.com ejecutamos el siguiente comando:

```
vyatta@Central:# set service webproxy url-filtering squidguard
local-block www.youtube.com

vyatta@Central:#commit
vyatta@Central:#save
```

Con esto es suficiente para bloquear YouTube, cuando un usuario quiera accede hacia esa dirección el proxy lo re direccionará hacia google.

```
root@Central# show service webproxy
 cache-size 100
 default-port 3128
 listen-address 10.10.0.1 {
 }
 url-filtering {
     squidguard {
         default-action allow
         local-block www.youtube.com
         local-block www.facebook.com
         local-block 66.220.158.27
         local-block www.claro.com.ni
         local-block-keyword porno
         redirect-url http://www.google.com
     }
 }
[edit]
root@Central#
```

Figura15. Mostrar Configuración SQUID en Vyatta 6.4

En la figura anterior podemos observar que hay 3 páginas web bloqueadas y que estas cuando quieran estas Re direccionadas hacia google. Cabe mencionar que no solamente se pueden bloquear URL acá tenemos un ejemplo donde tenemos bloqueadas 1 palabra en este caso simplemente anexamos keyword al final del comando anterior.

```
vyatta@Central:# set service webproxy url-filtering squidguard
local-block keyword porno

vyatta@Central:#commit
vyatta@Central:#save
```

Listo con esto ya tenemos funcional nuestro servidor proxy.

2. Práctica # 02

Configuración VLAN implementados con Switch Cisco Catalyst 3560

Objetivos

- Crear LANs virtuales diferentes en un router Vyatta.
- Administrar un router mediante consola con asignaciones IP y máscara de subred.
- Usar los comandos de configuración de Vyatta para la creación de VLANS.
- Asignar puertos a las VLANs en Switch Cisco Catalyst 3560.

Introducción

En esta práctica se pretende que el estudiante aplique todos los conocimientos teóricos que haiga obtenido en el desarrollo del tema de configuración de VLANs su creación y configuración. El estudiante tendrá la posibilidad de crear y observar el funcionamiento de una red física en la cual se podrán crear VLANs en un router Vyatta y posteriormente serán distribuidas en el Switch. La creación de VLANs es necesaria para dar seguridad a la red ya que brinda la facilidad para agrupar en VLANs los hosts de una red con el fin de que se creen restricciones de acceso entre hosts pertenecientes a VLANs diferentes.

Las VLANs son un medio muy poderoso a la hora de gestionar redes de área local de mediano y gran tamaño. Ampliamente utilizadas hoy en día, el conocimiento y comprensión de las mismas es fundamental para cualquier administrador de redes.

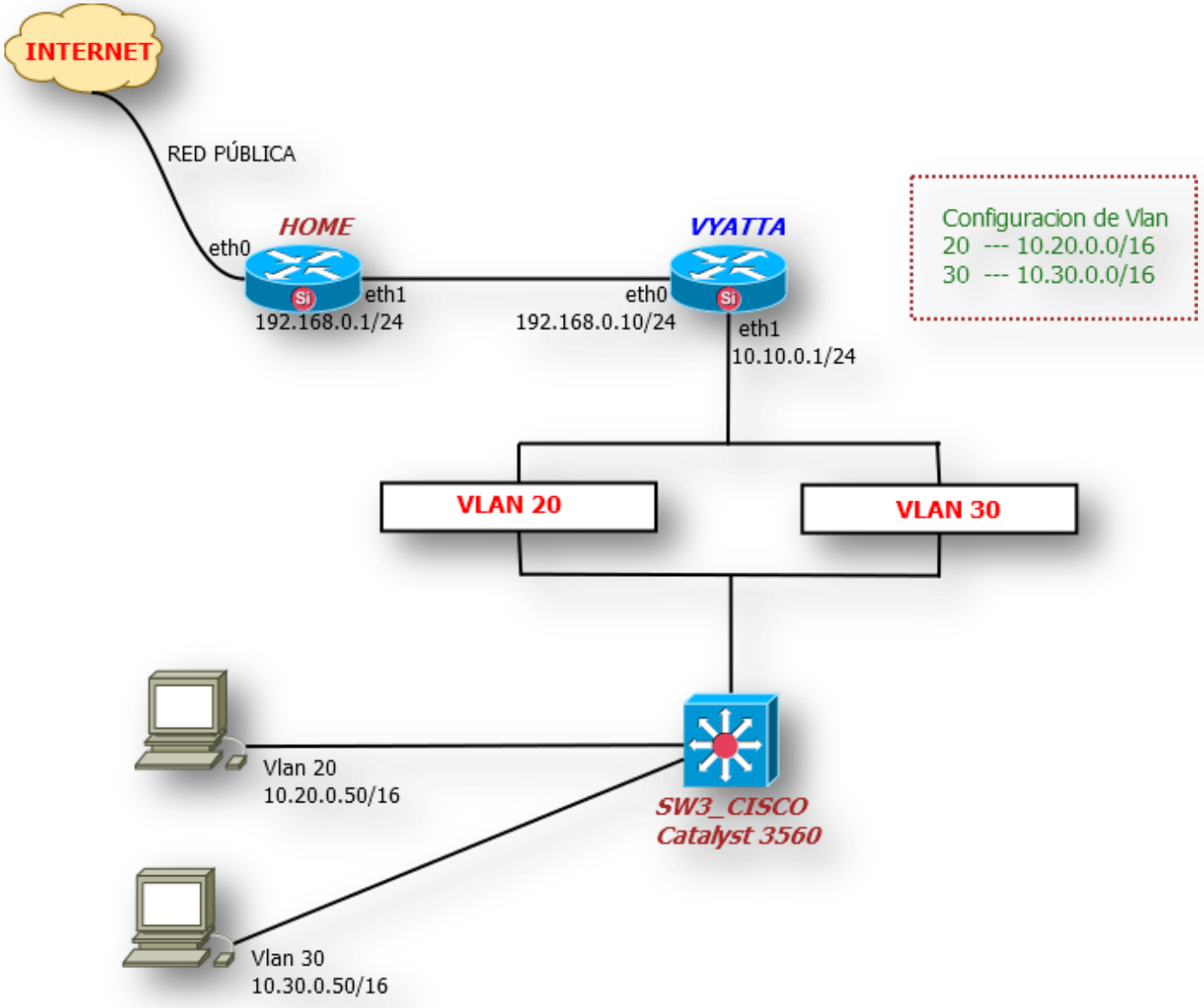


Figura16. Configuración VLAN con Switch Catalyst 3560

Cabe mencionar que este escenario es el mismo que se utilizó en la práctica anterior por tal razón no vamos a volver a configurar las interfaz de nuestro servidor Vyatta. Simplemente nos enfocaremos en la parte de creación de VLAN y realizaremos algunas pruebas para verificar que todo es correcto.

a) Primero nos creamos la interfaz de VLAN y asignamos una dirección de red.

```
vyatta@Central:#set interfaces Ethernet eth1 vif 20 address 10.20.0.254/16
vyatta@Central:#set interfaces Ethernet eth1 vif 30 address 10.30.0.254/16
vyatta@Central:# commit
vyatta@Central:#save
```

Podemos verificar que las VLAN se crearon correctamente con el comando **show interfaces**

```
root@Central:/home/vyatta#
root@Central:/home/vyatta# show int
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
eth0             192.168.0.11/24                u/u
eth1             10.10.0.1/16                   u/u
eth1.20          10.20.0.254/16                 u/u
eth1.30          10.30.0.254/16                 u/u
lo               127.0.0.1/8                    u/u
                 ::1/128
root@Central:/home/vyatta#
```

Figura17. Mostrar Configuración de Interfaz de Vlan en Vyatta 6.4

b) Es necesario crear unos pool con dhcp para que cuándo se asignen las Vlan a un puerto predeterminado no se esté escribiendo direcciones estáticas en las PC.

```
vyatta@Central:#set service dhcp-server shared-network-name Vlan20 network 10.20.0.0/16 start 10.20.0.50 stop 10.20.0.60
vyatta@Central:#set service dhcp-server shared-network-name Vlan20 network 10.20.0.0/16 default-router 10.20.0.254
vyatta@Central:#set service dhcp-server shared-network-name Vlan20 network 10.20.0.0/16 dns-server 192.136.45.66
vyatta@Central:#set service dhcp-server shared-network-name Vlan20 network 10.20.0.0/16 dns-server 200.85.160.4
```

```
vyatta@Central:#set service dhcp-server shared-network-name Vlan20 network 10.20.0.0/16 lease 86400
vyatta@Central:#set service dhcp-server shared-network-name Vlan30 network 10.30.0.0/16 start 10.30.0.50 stop 10.30.0.60
vyatta@Central:#set service dhcp-server shared-network-name Vlan30 network 10.30.0.0/16 default-router 10.30.0.254
vyatta@Central:#set service dhcp-server shared-network-name Vlan30 network 10.30.0.0/16 dns-server 192.136.45.66
vyatta@Central:#set service dhcp-server shared-network-name Vlan30 network 10.30.0.0/16 dns-server 200.85.160.4
vyatta@Central:#set service dhcp-server shared-network-name Vlan30 network 10.30.0.0/16 lease 86400
vyatta@Central:# commit
vyatta@Central:#save
```

Ahora ya tenemos asignación dinámica para las VLAN que creamos anteriormente.

NOTA:

Ya que tenemos creadas nuestras VLAN no necesitamos hacer en el puerto eth1 ningún mode trunk en la parte de Vyatta como se hace con Cisco.

c) Configurar el Switch Catalyst 3560 para distribuir las vlan que creamos en Vyatta.

Lo primero que haremos será configurar un puerto para recibir las vlan que se crearon en Vyatta. En nuestro caso configuramos FasEthernet0/2 pero puedes configurar el que tú gustes.

```
cisco(config)# interface FastEthernet0/2
cisco(config-if)# switchport trunk encapsulation dot1q
cisco(config-if)# switchport mode trunk
cisco(config-if)# do wr
```

Ahora configuramos el resto de puertos para distribuir las Vlan en este caso los puertos del 3-4 pertenecerán a la Vlan20 y los puertos 5-7 a la Vlan30.

```
cisco(config)# interface range FastEthernet0/3-4
cisco(config-if)# switchport mode trunk
cisco(config-if)# switchport access vlan20
```

```
cisco(config)# interface range FastEthernet0/5-7
cisco(config-if)# switchport mode trunk
cisco(config-if)# switchport access vlan30
cisco(config-if)# do wr
```

Podemos verificar que la configuración del Switcher es correcta con el comando **show running**

```
!
ip dhcp pool Info
   network 10.0.15.0 255.255.255.0
   default-router 10.0.15.1
!
!
!
!
no file verify auto
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
interface FastEthernet0/1
 no switchport
 ip address dhcp
!
interface FastEthernet0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
!
interface FastEthernet0/3
 switchport access vlan 20
!
interface FastEthernet0/4
 switchport access vlan 20
!
interface FastEthernet0/5
 switchport access vlan 30
!
interface FastEthernet0/6
 switchport access vlan 30
!
interface FastEthernet0/7
 switchport access vlan 30
!
interface FastEthernet0/8
 switchport access vlan 10
!
interface GigabitEthernet0/1
!
```

```
interface Vlan1
 no ip address
 shutdown
!
interface Vlan10
 ip address 10.0.15.1 255.255.255.0
!
ip classless
ip http server
!
!
control-plane
!
!
line con 0
 password 7 104F0D140C1943595F
 login
line vty 0 4
 password 7 104F0D140C1943595F
 login
line vty 5 15
 login
!
```

Figura18. Visualizar Configuración de Switch Catalyst 3560

Podemos observar que en la configuración del Router no hay nada más configurado que las contraseñas para acceder de forma remota y las interfaces con la que tiene el Router.

e) Conecta una PC al puerto 3 o 4 y verifica que dirección IP asigna.

Primeramente lo conectamos al puerto 3 y al verificar la interfaz de red nos retorna lo siguiente.

```
root@Info:/home/lester# ifconfig eth0
eth0      Link encap:Ethernet  direcciónHW bc:ae:c5:53:ee:69
          Direc. inet:10.20.0.50  Difus.:10.20.255.255  Másc:255.255.0.0
          Dirección inet6: fe80::beae:c5ff:fe53:ee69/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:397 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:581 errores:0 perdidos:0 overruns:0 carrier:1
          colisiones:0 long.colaTX:1000
          Bytes RX:93639 (93.6 KB)  TX bytes:49330 (49.3 KB)
```

Podemos observar que nos asignó 10.20.0.50/16 y ahora verificamos nuestro Gateway y veremos que nos retorna 10.20.0.254.

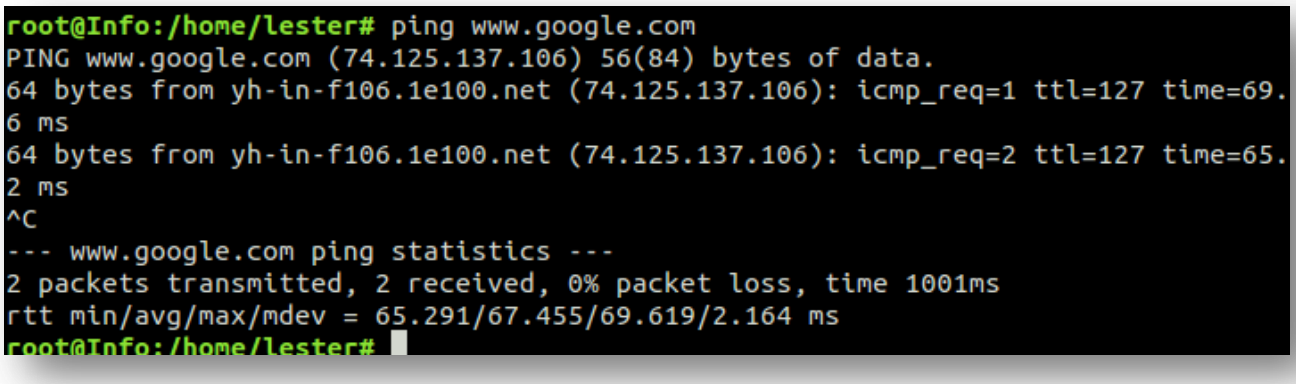
```
root@Info:/home/lester# route -n
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic Métric Ref    Uso Interfaz
0.0.0.0         10.20.0.254     0.0.0.0         UG    0      0        0 eth0
10.0.20.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet1
10.0.30.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet2
10.0.40.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet3
10.0.50.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet4
10.0.60.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet5
10.0.70.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet6
10.0.100.0      0.0.0.0         255.255.255.0   U     0      0        0 vmnet7
10.20.0.0       0.0.0.0         255.255.0.0     U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
root@Info:/home/lester#
```

f) Conecta una PC al puerto 5, 6 o 7 y verifica que dirección IP asigna.

Conectamos al puerto 5 y al verificar la interfaz de red nos retorna lo siguiente.

```
root@Info:/home/lester# ifconfig eth0
eth0      Link encap:Ethernet  direcciónHW bc:ae:c5:53:ee:69
          Direc. inet:10.30.0.50  Difus.:10.30.255.255  Másc:255.255.0.0
          Dirección inet6: fe80::beae:c5ff:fe53:ee69/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:420 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:638 errores:0 perdidos:0 overruns:0 carrier:5
          colisiones:0 long.colaTX:1000
          Bytes RX:97619 (97.6 KB)  TX bytes:60520 (60.5 KB)

root@Info:/home/lester# route -n
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic Métric Ref    Uso Interfaz
0.0.0.0         10.30.0.254     0.0.0.0         UG    0      0        0 eth0
10.0.20.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet1
10.0.30.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet2
10.0.40.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet3
10.0.50.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet4
10.0.60.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet5
10.0.70.0       0.0.0.0         255.255.255.0   U     0      0        0 vmnet6
10.0.100.0      0.0.0.0         255.255.255.0   U     0      0        0 vmnet7
10.30.0.0       0.0.0.0         255.255.0.0     U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
root@Info:/home/lester#
```

g) Realiza un ping desde cualquiera de las Vlan conectadas y verifica que navegas a internet sin ningún problema.



Excelente todo funciona muy bien.

3. Práctica # 03

Configuración de Protocolos de Encaminamiento BGP-RIP-OSPF

Objetivos:

- Hacer una simulación de múltiples ISPs, usando distintos protocolos de encaminamientos para lograr comunicación entre cada uno de estos, implementando exportación e importación de sus tablas de enrutamiento.
- Usar los comandos de configuración de Vyatta para el funcionamiento de los distintos protocolos de encaminamientos (BGP – RIP - OSPF).

Introducción:

Con la realización de esta práctica se pretende que el alumno adquiera los conocimientos prácticos sobre los diferentes protocolos de enrutamiento soportados por Vyatta como son: (OSPF, BGP y RIP). Con la implementación de los distintos protocolos se obtendrá información tanto de la Base de Datos topológica de la red como de las tablas de enrutamiento de rutas y puertos hacia cada red.

RIP es una implementación directa del encaminamiento vector-distancia para LANs. Utiliza UDP como protocolo de transporte, con el número de puerto 520 como puerto de destino. RIP opera en uno de dos modos: activo (normalmente usado por "routers") y pasivo (normalmente usado por hosts).

El Algoritmo OSPF, es un protocolo de enrutamiento de estado de enlace de Gateway interior. Este algoritmo de estado de enlace lo que hace es mantener una base de datos que refleja la topológica de la red en los routers; es decir, el estado de los enlaces de la red.

BGP es un protocolo de transporte fiable. Esto elimina la necesidad de llevar a cabo la fragmentación de actualización explícita, la retransmisión, el reconocimiento, y secuenciación.

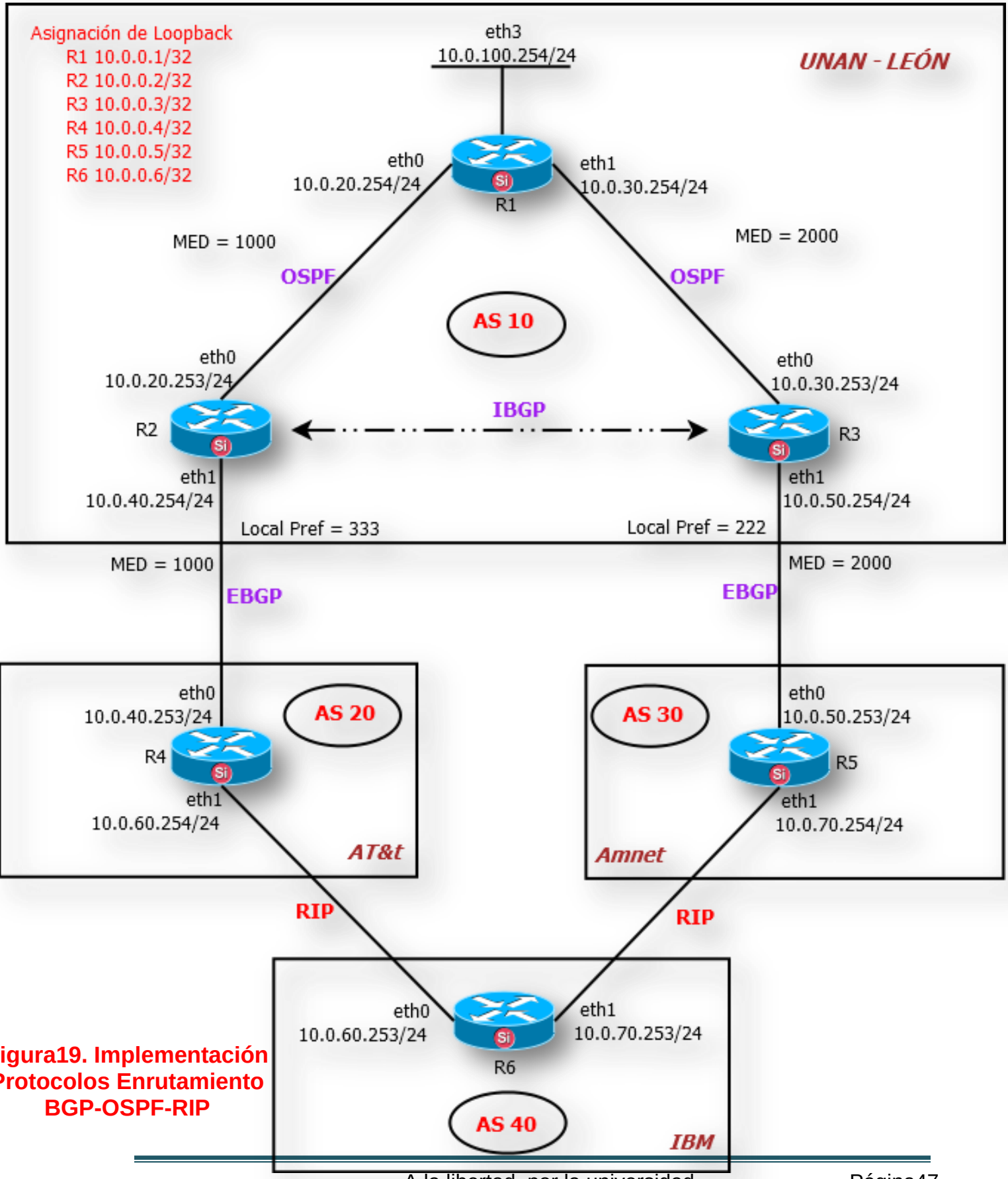


Figura19. Implementación Protocolos Enrutamiento BGP-OSPF-RIP

1. Escenario de red utilizando RIP, OSPF y BGP

Se desea crear un escenario de red como el que se muestra en la Figura19. En esta figura hay cuatro AS (Sistemas Autónomos): AS10, AS20, AS30 y AS40. Se ha configurado OSPF y BGP dentro de AS10, BGP y RIP dentro de AS20 y AS30 y por ultimo RIP dentro de AS40 y se desea configurar estos protocolos para conectar los 4 sistemas autónomos.

1.1 Configuración de los Router de AS10.

a) Como primer paso debemos asignarle una dirección IP a cada uno de los Router junto con su máscara y el Loopback al que pertenece.

```
vyatta@R1:# set interfaces Ethernet eth0 address 10.0.30.254/24
vyatta@R1:# set interfaces Ethernet eth1 address 10.0.20.254/24
vyatta@R1:# set interfaces Ethernet eth2 address 10.0.100.254/24
vyatta@R1:# set interfaces Loopback lo address 10.0.0.1/32
```

```
vyatta@R2:# set interfaces Ethernet eth0 address 10.0.20.253/24
vyatta@R2:# set interfaces Ethernet eth1 address 10.0.40.254/24
vyatta@R2:# set interfaces Loopback lo address 10.0.0.2/32
```

```
vyatta@R3:# set interfaces Ethernet eth0 address 10.0.30.253/24
vyatta@R3:# set interfaces Ethernet eth1 address 10.0.50.254/24
vyatta@R3:# set interfaces Loopback lo address 10.0.0.3/32
```

b) Configurar R1 para usar el protocolo OSPF pero además que este pueda redistribuir todo paquete que llegue hacia él.

```
vyatta@R1:# set protocols ospf area 0.0.0.11 network 10.0.20.0/24
vyatta@R1:# set protocols ospf area 0.0.0.11 network 10.0.30.0/24
vyatta@R1:# set protocols ospf area 0.0.0.11 network 10.0.100.0/24
vyatta@R1:# set protocols ospf area 0.0.0.11 network 10.0.0.1/32
vyatta@R1:# set protocols ospf parameters abr-type cisco
vyatta@R1:# set protocols ospf parameters router-id 10.0.0.1
vyatta@R1:# set protocols ospf redistribute connected metric-type 2
vyatta@R1:# commit
vyatta@R1:#save
```

c) Configurar R2 para usar el protocolo OSPF pero además que este pueda redistribuir todo paquete que llegue hacia él a través del protocolo BGP.

```
vyatta@R2:# set protocols ospf area 0.0.0.11 network 10.0.20.0/24
vyatta@R2:# set protocols ospf area 0.0.0.11 network 10.0.0.2/32
vyatta@R2:# set protocols ospf parameters abr-type cisco
vyatta@R2:# set protocols ospf parameters router-id 10.0.0.2
vyatta@R2:# set protocols ospf redistribute connected
vyatta@R2:# set protocols ospf redistribute bgp
vyatta@R2:# commit
vyatta@R2:#save
```

d) Configuramos R3 para encaminar por OSPF y además hacer que este redistribuya su tabla de enrutamiento a través de BGP.

```
vyatta@R3:# set protocols ospf area 0.0.0.11 network 10.0.30.0/24
vyatta@R3:# set protocols ospf area 0.0.0.11 network 10.0.0.3/32
vyatta@R3:# set protocols ospf parameters abr-type cisco
vyatta@R3:# set protocols ospf parameters router-id 10.0.0.3
vyatta@R3:# set protocols ospf redistribute connected
vyatta@R3:# set protocols ospf redistribute bgp
vyatta@R3:# commit
vyatta@R3:# save
```

e) Configurar R2 para encaminar paquetes a través del protocolo BGP y también hacer que este redistribuya su tabla de enrutamiento a través de OSPF y BGP.

Para resolver este el inciso anterior primero debemos crear algunas listas de acceso que nos permitirán comunicar un protocolo con otro para finalmente lograr que puedan llegar paquetes de una red hacia otra sin importar el protocolo que tenga en su extremo configurado cada enrutador.

```
vyatta@R2:# set policy access-list 100 rule 10 action permit
vyatta@R2:# set policy access-list 100 rule 10 destination any
vyatta@R2:# set policy access-list 100 rule 10 source any
vyatta@R2:#set policy access-list 150 rule 10 action permit
vyatta@R2:# set policy access-list 150 rule 10 destination any
vyatta@R2:# set policy access-list 150 rule 10 source inverse-mask 0.0.0.255
vyatta@R2:# set policy access-list 150 rule 10 source network 10.0.20.0
vyatta@R2:# set policy route-map SET-MED rule 10 action permit
vyatta@R2:# set policy route-map SET-MED rule 10 match ip address access-list 150
vyatta@R2:# set policy route-map SET-MED rule 10 set metric 1000
vyatta@R2:# set policy route-map SET-MED rule 20 action permit
vyatta@R2:# set policy route-map SET-MED rule 20 match ip address access-list 100
vyatta@R2:# set policy route-map SET-LOCAL-PREF rule 10 action permit
```

```
vyatta@R2:# set policy route-map SET-LOCAL-PREF rule 10 match ip address access-list
  100
vyatta@R2:# set policy route-map SET-LOCAL-PREF rule 10 set local-preference 333
```

Ahora que ya tenemos nuestras políticas de acceso podemos configurar BGP para importar y exportar las tablas de enrutamiento de OSPF configurado en R2.

```
vyatta@R2:# set protocols bgp 10 neighbor 10.0.30.253 remote-as 10
vyatta@R2:# set protocols bgp 10 neighbor 10.0.30.253 route-map import SET-LOCAL-PREF
vyatta@R2:# set protocols bgp 10 neighbor 10.0.40.253 nexthop-self
vyatta@R2:# set protocols bgp 10 neighbor 10.0.40.253 remote-as 20
vyatta@R2:# set protocols bgp 10 neighbor 10.0.40.253 route-map export SET-MED
vyatta@R2:# set protocols bgp 10 parameters router-id 10.0.0.2
vyatta@R2:# set protocols bgp 10 redistribute connected
vyatta@R2:# set protocols bgp 10redistribute ospf
```

f) Configuramos R3 para encaminar paquetes a través del protocolo BGP y también hacer que este redistribuya su tabla de enrutamiento a través de OSPF y BGP.

Igual que el apartado anterior debemos crear algunas listas de acceso que nos permitirán comunicar un protocolo con otro para finalmente lograr que puedan llegar paquetes de una red hacia otra sin importar el protocolo que tenga en su extremo configurado cada enrutador.

```
vyatta@R3:# set policy access-list 100 rule 10 action permit
vyatta@R3:# set policy access-list 100 rule 10 destination any
vyatta@R3:# set policy access-list 100 rule 10 source any
vyatta@R3:#set policy access-list 150 rule 10 action permit
vyatta@R3:# set policy access-list 150 rule 10 destination any
vyatta@R3:# set policy access-list 150 rule 10 source inverse-mask 0.0.0.255
vyatta@R3:# set policy access-list 150 rule 10 source network 10.0.30.0
vyatta@R3:# set policy route-map SET-MED rule 10 action permit
vyatta@R3:# set policy route-map SET-MED rule 10 match ip address access-list 150
vyatta@R3:# set policy route-map SET-MED rule 10 set metric 2000
vyatta@R3:# set policy route-map SET-MED rule 20 action permit
vyatta@R3:# set policy route-map SET-MED rule 20 match ip address access-list 100
vyatta@R3:# set policy route-map SET-LOCAL-PREF rule 10 action permit
vyatta@R3:# set policy route-map SET-LOCAL-PREF rule 10 match ip address access-list100
vyatta@R3:# set policy route-map SET-LOCAL-PREF rule 10 set local-preference 222
```

Ahora que ya tenemos nuestras políticas de acceso podemos configurar BGP para importar y exportar las tablas de enrutamiento de OSPF configurado en R3.

```
vyatta@R3:# set protocols bgp 10 neighbor 10.0.20.253 remote-as 10
vyatta@R3:# set protocols bgp 10 neighbor 10.0.20.253 route-map import SET-LOCAL-PREF
vyatta@R3:# set protocols bgp 10 neighbor 10.0.50.253 nexthop-self
vyatta@R3:# set protocols bgp 10 neighbor 10.0.50.253 remote-as 30
vyatta@R3:# set protocols bgp 10 neighbor 10.0.50.253 route-map export SET-MED
vyatta@R3:# set protocols bgp 10 parameters router-id 10.0.0.3
vyatta@R3:# set protocols bgp 10 redistribute connected
vyatta@R3:# set protocols bgp 10redistribute ospf
```

g) Verificar las interfaz de red deR1 - R2 - R3 con el comando **show interfaces**.

R1

```
root@R1:/home/vyatta#
root@R1:/home/vyatta# show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
eth0             10.0.30.254/24                 u/u
eth1             10.0.20.254/24                 u/u
eth2             10.0.100.254/24                u/u
lo               127.0.0.1/8                    u/u
                 10.0.0.1/32
                 ::1/128
root@R1:/home/vyatta#
```

R2

```
root@R2:/home/vyatta#
root@R2:/home/vyatta# show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
eth0             10.0.20.253/24                 u/u
eth1             10.0.40.254/24                 u/u
lo               127.0.0.1/8                    u/u
                 10.0.0.2/32
                 ::1/128
root@R2:/home/vyatta#
```

R3

```
root@R3:/home/vyatta#
root@R3:/home/vyatta# show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
eth0             10.0.30.253/24                 u/u
eth1             10.0.50.254/24                 u/u
lo               127.0.0.1/8                    u/u
                 10.0.0.3/32
                 ::1/128
root@R3:/home/vyatta#
```

h) Usa el comando **show protocols** en R1 y explica su contenido.

```
root@R1# show protocols
 ospf {
     area 0.0.0.11 {
         network 10.0.100.0/24
         network 10.0.20.0/24
         network 10.0.30.0/24
         network 10.0.0.1/32
     }
     parameters {
         abr-type cisco
         router-id 10.0.0.1
     }
     redistribute {
         connected {
             metric-type 2
         }
     }
 }
[edit]
root@R1#
```

Figura20. Mostrar Configuración Protocolos de Enrutamiento Vyatta 6.4

Con el comando show protocols podemos observar el protocolo que tiene configurado R1, vemos que es OSPF y este nos muestra las redes que este conoce además nos muestra el identificado asignado y por ultimo pues la redistribución de rutas que este tiene que aplicar.

i) Usa el comando **show protocols bgp** en R2 y explica su contenido.

```
root@R2# show protocols bgp
 bgp 10 {
     neighbor 10.0.30.253 {
         remote-as 10
         route-map {
             import SET-LOCAL-PREF
         }
     }
     neighbor 10.0.40.253 {
         nexthop-self
         remote-as 20
         route-map {
             export SET-MED
         }
     }
     parameters {
         router-id 10.0.0.2
     }
     redistribute {
         connected {
         }
         ospf {
         }
     }
 }
[edit]
root@R2#
```

Podemos observar que este tiene configurado el protocolo BGP y nos muestra algunas rutas que este importa o exporta de otra red. Además vemos cuales son las redes vecinas que tiene configurado y también el sistema autónomo al que pertenece y por ultimo vemos que se hace una redistribución de BGP a OSPF.

j) Usa el comando **show protocols ospf** en R2 y explica su contenido.

Cabe mencionar que R2 también tiene configurado el protocolo OSPF este para poder comunicarse con R1, acá se muestra las redes que este conoce, el identificador que tiene y una redistribución de protocolos entre OSPF y BGP esto para poder exportar e importar las tablas de enrutamiento de cada Router.

```
root@R2# show protocols ospf
 area 0.0.0.11 {
     network 10.0.20.0/24
     network 10.0.0.2/32
 }
 parameters {
     abr-type cisco
     router-id 10.0.0.2
 }
 redistribute {
     bgp {
         metric-type 2
     }
     connected {
         metric-type 2
     }
 }
[edit]
root@R2#
```

k) Usa el comando **show protocols bgp** en R3 y explica su contenido.

Podemos observar que este tiene configurado el protocolo BGP y nos muestra algunas rutas que este importa o exporta de otra red. Además vemos cuales son las redes vecinas que tiene configurado y también el sistema autónomo al que pertenece y por ultimo vemos que se hace una redistribución de BGP a OSPF.

```
root@R3# show protocols bgp
 bgp 10 {
     neighbor 10.0.20.253 {
         remote-as 10
         route-map {
             import SET-LOCAL-PREF
         }
     }
     neighbor 10.0.50.253 {
         nexthop-self
         remote-as 30
         route-map {
             export SET-MED
         }
     }
     parameters {
         router-id 10.0.0.3
     }
     redistribute {
         connected {
         }
         ospf {
         }
     }
 }
[edit]
root@R3#
```

l) Usa el comando **show protocols ospf** en R3 y explica su contenido

```
root@R3# show protocols ospf
 area 0.0.0.11 {
     network 10.0.30.0/24
     network 10.0.0.3/32
 }
 parameters {
     abr-type cisco
     router-id 10.0.0.3
 }
 redistribute {
     bgp {
         metric-type 2
     }
     connected {
         metric-type 2
     }
 }
[edit]
root@R3#
```

Observamos que está configurado el protocolo OSPF este para poder comunicarse con R1, acá se muestra las redes que este conoce, el identificador que tiene y una redistribución de protocolos entre OSPF y BGP esto para poder exportar e importar las tablas de enrutamiento de cada Router.

m) Verifica la tabla de enrutamiento de R1 y explica su contenido.

Usando el comando **show ip route** podemos ver las rutas que se guardan en la tabla de enrutamiento de R1, acá nos muestra que rutas aprendió mediante el protocolo OSPF y nos dice hasta la interfaz por la cual aprendió esa ruta.

```
root@R1:/home/vyatta#
root@R1:/home/vyatta#
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

O   10.0.0.1/32 [110/10] is directly connected, lo, 02:11:20
C>* 10.0.0.1/32 is directly connected, lo
O>* 10.0.0.2/32 [110/20] via 10.0.20.253, eth1, 02:11:05
O>* 10.0.0.3/32 [110/20] via 10.0.30.253, eth0, 02:11:04
O>* 10.0.0.4/32 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                      via 10.0.30.253, eth0, 02:11:03
O>* 10.0.0.5/32 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                      via 10.0.30.253, eth0, 02:11:03
O>* 10.0.0.6/32 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                      via 10.0.30.253, eth0, 02:11:03
O   10.0.20.0/24 [110/10] is directly connected, eth1, 02:11:21
C>* 10.0.20.0/24 is directly connected, eth1
O   10.0.30.0/24 [110/10] is directly connected, eth0, 02:11:20
C>* 10.0.30.0/24 is directly connected, eth0
O>* 10.0.40.0/24 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                       via 10.0.30.253, eth0, 02:11:03
O>* 10.0.50.0/24 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                       via 10.0.30.253, eth0, 02:11:03
O>* 10.0.60.0/24 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                       via 10.0.30.253, eth0, 02:11:03
O>* 10.0.70.0/24 [110/20] via 10.0.20.253, eth1, 02:11:03
  *                       via 10.0.30.253, eth0, 02:11:03
```

Figura21. Visualizar Tabla de Enrutamiento Vyatta 6.4

n) Verifica la tabla de enrutamiento de R2 y explica su contenido.

Usando el comando **show ip route** podemos saber cuál es la tabla de enrutamiento que tiene R3, esta nos mostrara cuales son las rutas que están directamente conectadas hacía el, además muestra que aprendió rutas mediante BGP y OSPF a través de alguna de las interfaces que tiene R3.

```
root@R2:/home/vyatta#
root@R2:/home/vyatta# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

O>* 10.0.0.1/32 [110/20] via 10.0.20.254, eth0, 02:14:42
O   10.0.0.2/32 [110/10] is directly connected, lo, 02:18:04
C>* 10.0.0.2/32 is directly connected, lo
O   10.0.0.3/32 [110/30] via 10.0.20.254, 02:14:32
B>* 10.0.0.3/32 [20/3] via 10.0.40.253, eth1, 02:15:08
O   10.0.0.4/32 [110/20] via 10.0.20.254, 02:14:31
B>* 10.0.0.4/32 [20/1] via 10.0.40.253, eth1, 02:16:08
O   10.0.0.5/32 [110/20] via 10.0.20.254, 02:14:31
B>* 10.0.0.5/32 [20/3] via 10.0.40.253, eth1, 02:15:08
O   10.0.0.6/32 [110/20] via 10.0.20.254, 02:14:31
B>* 10.0.0.6/32 [20/2] via 10.0.40.253, eth1, 02:14:38
O   10.0.20.0/24 [110/10] is directly connected, eth0, 02:18:05
C>* 10.0.20.0/24 is directly connected, eth0
O   10.0.30.0/24 [110/20] via 10.0.20.254, 02:14:42
B>* 10.0.30.0/24 [20/3] via 10.0.40.253, eth1, 02:15:08
O   10.0.40.0/24 [110/20] via 10.0.20.254, 02:14:31
C>* 10.0.40.0/24 is directly connected, eth1
O   10.0.50.0/24 [110/20] via 10.0.20.254, 02:14:31
B>* 10.0.50.0/24 [20/3] via 10.0.40.253, eth1, 02:15:08
```

o) Verifica la tabla de enrutamiento de R3 y explica su contenido.

R3 aprende rutas mediante OSPF y muestra por cual interfaz esta aprendió esta ruta. También este tiene guardada rutas a través del protocolo BGP, es acá donde tiene mucha importancia la parte de exportar rutas y por eso mismo fue que se crearon algunas listas de acceso para la configuración de BGP en R2 y R3.

```
root@R3:/home/vyatta#
root@R3:/home/vyatta# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

O>* 10.0.0.1/32 [110/20] via 10.0.30.254, eth0, 02:15:19
O   10.0.0.2/32 [110/30] via 10.0.30.254, 02:15:19
B>* 10.0.0.2/32 [20/3] via 10.0.50.253, eth1, 02:15:46
O   10.0.0.3/32 [110/10] is directly connected, lo, 02:17:39
C>* 10.0.0.3/32 is directly connected, lo
O   10.0.0.4/32 [110/20] via 10.0.30.254, 02:15:18
B>* 10.0.0.4/32 [20/3] via 10.0.50.253, eth1, 02:15:46
O   10.0.0.5/32 [110/20] via 10.0.30.254, 02:15:18
B>* 10.0.0.5/32 [20/1] via 10.0.50.253, eth1, 02:16:16
O   10.0.0.6/32 [110/20] via 10.0.30.254, 02:15:18
B>* 10.0.0.6/32 [20/2] via 10.0.50.253, eth1, 02:15:46
O   10.0.20.0/24 [110/20] via 10.0.30.254, 02:15:19
B>* 10.0.20.0/24 [20/3] via 10.0.50.253, eth1, 02:15:46
O   10.0.30.0/24 [110/10] is directly connected, eth0, 02:17:40
C>* 10.0.30.0/24 is directly connected, eth0
O   10.0.40.0/24 [110/20] via 10.0.30.254, 02:15:18
B>* 10.0.40.0/24 [20/3] via 10.0.50.253, eth1, 02:15:46
O   10.0.50.0/24 [110/20] via 10.0.30.254, 02:15:18
C>* 10.0.50.0/24 is directly connected, eth1
O   10.0.60.0/24 [110/20] via 10.0.30.254, 02:15:18
B>* 10.0.60.0/24 [20/2] via 10.0.50.253, eth1, 02:15:46
```

1.2 Configuración de los Router de AS20.

a) Primero debemos asignarle una dirección IP a cada uno de los Router junto con su máscara y el Loopback al que pertenece.

```
vyatta@R4:# set interfaces Ethernet eth0 address 10.0.40.253/24
vyatta@R4:# set interfaces Ethernet eth1 address 10.0.60.254/24
vyatta@R4:# set interfaces Loopback lo address 10.0.0.4/32
```

b) Ahora configuramos R4 para encaminar paquetes a través del protocolo BGP para redistribuir todo paquete que llegue hacia él y que venga tanto as AS10 como de AS30 o AS40.

Primero comenzamos configurando BGP y hacemos redistribución de paquetes de RIP ya que este es el otro protocolo que tendremos configurado en este Router.

```
vyatta@R4:# set protocols bgp 20 neighbor 10.0.40.254 remote-as 10
vyatta@R4:# set protocols bgp 20 parameters router-id 10.0.0.4
vyatta@R4:# set protocols bgp 20 redistribute connected
vyatta@R4:# set protocols bgp 20 redistribute rip
```

Ahora configuramos RIP en R4 y hacemos redistribución de paquetes que vengan a través de BGP

```
vyatta@R4:# set protocols rip network 10.0.60.0/24
vyatta@R4:# set protocols rip network 10.0.0.4/32
vyatta@R4:# set protocols rip redistribute connected
vyatta@R4:# set protocols rip redistribute bgp
```

c) Usa el comando **show protocols** en R4 y explica su contenido

```
root@R4# show protocols
 bgp 20 {
     neighbor 10.0.40.254 {
         remote-as 10
     }
     parameters {
         router-id 10.0.0.4
     }
     redistribute {
         connected {
         }
         rip {
         }
     }
 }
 rip {
     network 10.0.60.0/24
     redistribute {
         bgp {
         }
         connected {
         }
     }
 }
[edit]
root@R4#
```

Acá vemos que simplemente están configurado RIP y BGP pero no hay ninguna lista de acceso para poder comunicar RIP y BGP como se hizo en el apartado anterior, esto se debe a

que RIP es un protocolo más vulnerable. También observamos las redes vecina que tiene R4 que en este caso será R2 y R6.

d) Verifica la tabla de enrutamiento de R4 y explica su contenido.

Usando el comando show ip route en R4 podemos observar cuales son las rutas que están en la tabla de enrutamiento, vemos que este guarda rutas aprendida mediante el protocolo RIP y BGP además nos dice hasta la interfaz por la cual aprendió esa ruta el enrutador.

```
root@R4:/home/vyatta#
root@R4:/home/vyatta# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

B>* 10.0.0.1/32 [20/20] via 10.0.40.254, eth0, 02:18:54
B>* 10.0.0.2/32 [20/1] via 10.0.40.254, eth0, 02:20:24
R>* 10.0.0.3/32 [120/3] via 10.0.60.253, eth1, 02:19:27
C>* 10.0.0.4/32 is directly connected, lo
R>* 10.0.0.5/32 [120/3] via 10.0.60.253, eth1, 02:19:27
R>* 10.0.0.6/32 [120/2] via 10.0.60.253, eth1, 02:19:23
B>* 10.0.20.0/24 [20/1000] via 10.0.40.254, eth0, 02:20:24
R>* 10.0.30.0/24 [120/3] via 10.0.60.253, eth1, 02:19:27
C>* 10.0.40.0/24 is directly connected, eth0
R>* 10.0.50.0/24 [120/3] via 10.0.60.253, eth1, 02:19:27
C>* 10.0.60.0/24 is directly connected, eth1
R>* 10.0.70.0/24 [120/2] via 10.0.60.253, eth1, 02:19:27
B>* 10.0.100.0/24 [20/20] via 10.0.40.254, eth0, 02:18:54
C>* 127.0.0.0/8 is directly connected, lo
root@R4:/home/vyatta#
```

e) Verifica haciendo ping entre R1 y R4 para saber si todo funciona correctamente en el enrutamiento de protocolos.

```
root@R1:/home/vyatta#
root@R1:/home/vyatta# ping 10.0.40.253
PING 10.0.40.253 (10.0.40.253) 56(84) bytes of data.
64 bytes from 10.0.40.253: icmp_req=1 ttl=61 time=7.53 ms
64 bytes from 10.0.40.253: icmp_req=2 ttl=61 time=6.60 ms
^C
--- 10.0.40.253 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 6.608/7.069/7.531/0.469 ms
root@R1:/home/vyatta#
```

Cabe mencionar que todo funciona correctamente vemos que los tiempos de llegada son muy buenos y no hay ninguna pérdida de paquetes de R1 a R4.

1.3 Configuración de los Router de AS30.

a) Primero debemos asignarle una dirección IP a cada uno de los Router junto con su máscara y el Loopback al que pertenece.

```
vyatta@R5:# set interfaces Ethernet eth0 address 10.0.50.253/24
vyatta@R5:# set interfaces Ethernet eth1 address 10.0.70.254/24
vyatta@R5:# set interfaces Loopback lo address 10.0.0.5/32
```

b) Configurar R5 para encaminar paquetes a través del protocolo BGP para redistribuir todo paquete que llegue hacia él y que venga tanto as AS10 como de AS30 o AS40.

Primero comenzamos configurando BGP y hacemos redistribución de paquetes de RIP ya que este es el otro protocolo que tendremos configurado en este Router.

```
vyatta@R5:# set protocols bgp 30 neighbor 10.0.50.254 remote-as 10
vyatta@R5:# set protocols bgp 30 parameters router-id 10.0.0.5
vyatta@R5:# set protocols bgp 30 redistribute connected
vyatta@R5:# set protocols bgp 30 redistribute rip
```

Ahora configuramos RIP en R5 y hacemos redistribución de paquetes que vengan a través de BGP

```
vyatta@R5:# set protocols rip network 10.0.70.0/24
vyatta@R5:# set protocols rip network 10.0.0.5/32
vyatta@R5:# set protocols rip redistribute connected
vyatta@R5:# set protocols rip redistribute bgp
```

c) Usa el comando **show protocols** en R5 y explica su contenido.

Pues podemos observar que R5 tiene configurado BGP y RIP pero acá la configuración se hizo sin hacer uso de ninguna lista de acceso debido a que al hacer exportación de red de BGP a RIP y viceversa debido a que RIP es un protocolo de encaminamiento más vulnerable no es necesario hacer ninguna lista, es suficiente con que se conozcan cuáles serán los vecinos de R5 y hacer la redistribución de protocolos.

```
root@R5# show protocols
 bgp 30 {
     neighbor 10.0.50.254 {
         remote-as 10
     }
     parameters {
         router-id 10.0.0.5
     }
     redistribute {
         connected {
         }
         rip {
         }
     }
 }
 rip {
     network 10.0.70.0/24
     redistribute {
         bgp {
         }
         connected {
         }
     }
 }
[edit]
root@R5#
```

d) Verifica la tabla de enrutamiento de R5 y explica su contenido.

Usando el comando **show ip route** en R5 podemos ver la tabla de enrutamiento de este Router, acá nos muestra que hay rutas aprendidas mediante RIP y BGP, también podemos observar cuales son las redes conectadas directamente hacia él y por cual interfaz aprendió este estas rutas.

```
root@R5:/home/vyatta#
root@R5:/home/vyatta# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

R>* 10.0.0.1/32 [120/3] via 10.0.70.253, eth1, 02:20:41
R>* 10.0.0.2/32 [120/3] via 10.0.70.253, eth1, 02:21:10
B>* 10.0.0.3/32 [20/1] via 10.0.50.254, eth0, 02:21:32
R>* 10.0.0.4/32 [120/3] via 10.0.70.253, eth1, 02:21:10
C>* 10.0.0.5/32 is directly connected, lo
R>* 10.0.0.6/32 [120/2] via 10.0.70.253, eth1, 02:21:10
R>* 10.0.20.0/24 [120/3] via 10.0.70.253, eth1, 02:21:10
B>* 10.0.30.0/24 [20/2000] via 10.0.50.254, eth0, 02:21:32
R>* 10.0.40.0/24 [120/3] via 10.0.70.253, eth1, 02:21:10
C>* 10.0.50.0/24 is directly connected, eth0
R>* 10.0.60.0/24 [120/2] via 10.0.70.253, eth1, 02:21:14
C>* 10.0.70.0/24 is directly connected, eth1
R>* 10.0.100.0/24 [120/3] via 10.0.70.253, eth1, 02:20:36
C>* 127.0.0.0/8 is directly connected, lo
root@R5:/home/vyatta#
```

e) Verifica haciendo ping R1 a R5 para saber si la exportación de tablas de enrutamiento es correcto.

```
root@R1:/home/vyatta#
root@R1:/home/vyatta# ping 10.0.50.253
PING 10.0.50.253 (10.0.50.253) 56(84) bytes of data.
64 bytes from 10.0.50.253: icmp_req=1 ttl=63 time=3.68 ms
64 bytes from 10.0.50.253: icmp_req=2 ttl=63 time=2.29 ms
^C
--- 10.0.50.253 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.295/2.992/3.689/0.697 ms
root@R1:/home/vyatta#
```

Cabe mencionar que todo funciona correctamente vemos que los tiempos de llegada son muy buenos y no hay ninguna pérdida de paquetes de R1 a R5.

1.4 Configuración de los Router de AS40.

a) Primero debemos asignarle una dirección IP a cada uno de los Router junto con su máscara y el Loopback al que pertenece.

```
vyatta@R6:# set interfaces Ethernet eth0 address 10.0.60.253/24
vyatta@R6:# set interfaces Ethernet eth1 address 10.0.70.253/24
vyatta@R6:# set interfaces Loopback lo address 10.0.0.6/32
```

b) Ahora configuramos R6 para encaminar paquetes a través del protocolo RIP y hacemos que redistribuya paquetes del mismo.

```
vyatta@R6:# set protocols rip network 10.0.60.0/24
vyatta@R6:# set protocols rip network 10.0.70.0/24
vyatta@R6:# set protocols rip network 10.0.0.6/32
vyatta@R6:# set protocols rip redistribute connected
vyatta@R6:# set protocols rip redistribute bgp
```

c) Usa el comando **show protocols** en R6 y explica su contenido.

```
root@R6# show protocols rip
 network 10.0.60.0/24
 network 10.0.70.0/24
 redistribute {
     bgp {
     }
     connected {
     }
 }
[edit]
root@R6#
```

Vemos que RIP es el protocolo más vulnerable y sencillo de configurar, pues solamente pasamos las redes vecinas y le decimos que cualquier paquete que pase por acá lo redistribuya con BGP esto para poder llegar hasta R1 de ser posible.

d) Verifica la tabla de enrutamiento de R6 y explica su contenido.

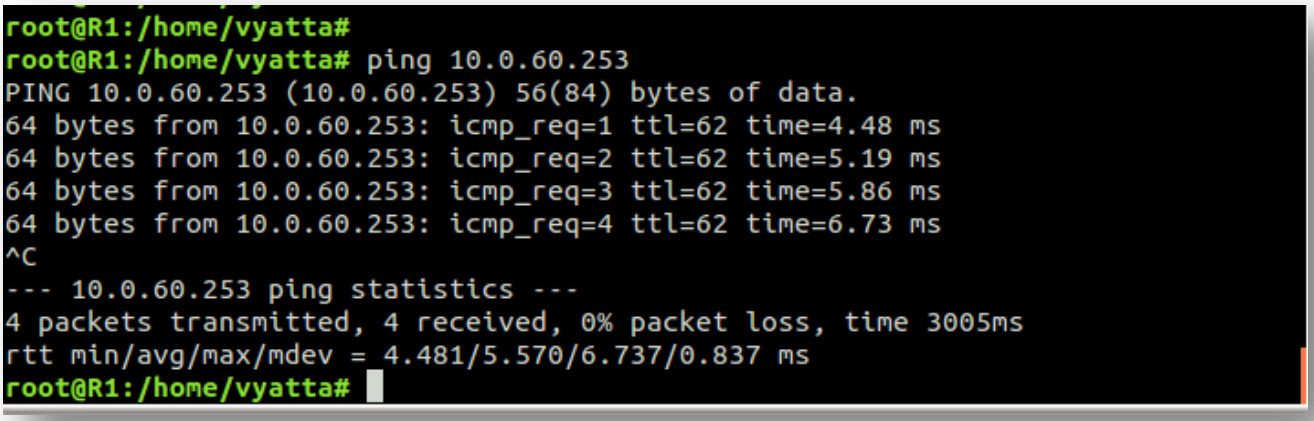
```
root@R6:/home/vyatta#
root@R6:/home/vyatta# sho ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

R>* 10.0.0.1/32 [120/2] via 10.0.60.254, eth0, 02:22:30
R>* 10.0.0.2/32 [120/2] via 10.0.60.254, eth0, 02:23:00
R>* 10.0.0.3/32 [120/2] via 10.0.70.254, eth1, 02:23:03
R>* 10.0.0.4/32 [120/2] via 10.0.60.254, eth0, 02:23:00
R>* 10.0.0.5/32 [120/2] via 10.0.70.254, eth1, 02:23:03
C>* 10.0.0.6/32 is directly connected, lo
R>* 10.0.20.0/24 [120/2] via 10.0.60.254, eth0, 02:23:00
R>* 10.0.30.0/24 [120/2] via 10.0.70.254, eth1, 02:23:03
R>* 10.0.40.0/24 [120/2] via 10.0.60.254, eth0, 02:23:00
R>* 10.0.50.0/24 [120/2] via 10.0.70.254, eth1, 02:23:03
C>* 10.0.60.0/24 is directly connected, eth0
C>* 10.0.70.0/24 is directly connected, eth1
R>* 10.0.100.0/24 [120/2] via 10.0.60.254, eth0, 02:22:25
C>* 127.0.0.0/8 is directly connected, lo
root@R6:/home/vyatta#
```

Como podemos observar R6 conoce todas las rutas que están distribuidas, también vemos que estas rutas son aprendidas mediante el protocolo RIP y a través de que interfaz de aprendieron,

e) Verifica haciendo ping R6 a R1 para saber si los paquetes llegan correctamente.

```
root@R6:/home/vyatta#
root@R6:/home/vyatta# ping 10.0.20.254
PING 10.0.20.254 (10.0.20.254) 56(84) bytes of data.
64 bytes from 10.0.20.254: icmp_req=1 ttl=62 time=21.2 ms
64 bytes from 10.0.20.254: icmp_req=2 ttl=62 time=4.73 ms
64 bytes from 10.0.20.254: icmp_req=3 ttl=62 time=4.73 ms
^C
--- 10.0.20.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 4.731/10.248/21.283/7.803 ms
root@R6:/home/vyatta#
```

Podemos observar que todo funciona correctamente vemos que los tiempos de llegada son muy buenos y no hay ninguna pérdida de paquetes de R6 a R1 y de igual forma funciona de R1 a R6 como se muestra en la siguiente figura.



Capítulo VI: Conclusiones y Futuros Trabajos

Una vez finalizada y probado el sistema se ha llegado a la siguiente conclusión:

1) Actualmente, existen una variedad de excelentes herramientas de **Open Source Routing** con diferentes características, y que podrían ser utilizadas en diversos entornos según las necesidades particulares de una red.

2) El sistema Vyatta es la herramienta más completa de las estudiadas, ya que implementa funcionalidades, tanto en la capa 2 (Switching), capa 3(Enrutamiento), capa 5 (Aplicaciones), así como seguridad de la red.

3) Después de haber realizado las prácticas de laboratorio, se puede comprobar que a través del sistema Vyatta es posible realizar configuraciones, tanto básicas como avanzadas, que se realizan en otras tecnologías como CISCO, JUNIPER, etc.

LÍNEAS DE TRABAJO FUTUROS.

Al finalizar este trabajo se proponen las siguientes líneas de trabajo:

1) A partir de las implementaciones básicas que se realizaron, se pueden plantear prácticas de laboratorios más complejas.

2) Actualizar las prácticas de laboratorios ya realizadas con las nuevas versiones de Vyatta.

3) Montar escenarios complejos combinando Vyatta con otras tecnologías tales como: CISCO, DELL, Pfsense, Zeroshell, etc.

Capítulo VII: Bibliografía.

Wikipedia. (Septiembre 2012). Obtenido de http://en.wikipedia.org/wiki/List_of_open_source_routing_platforms

VYATTA, INC (10 Septiembre del 2012). Creating and Using a LiveCD. INSTALLING AND UPGRADING. Obtenido de http://www.vyatta.com/downloads/documentation/ Vyatta-UsingLiveCD_R6.4_v01

VYATTA, INC (10 Septiembre del 2012). Basic Routing. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/Vyatta-BasicRouting_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). RIP. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/ Vyatta-RIP_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). BGP. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/ Vyatta-BGP_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). OSPF. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/Vyatta-OSPF_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). NAT. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/ Vyatta-NAT_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). LAN Interfaces. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/ Vyatta-LANInterfaces_R6.4_v01.pdf

VYATTA, INC (10 Septiembre del 2012). Firewall. Reference Guide. Obtenido de http://www.vyatta.com/downloads/documentation/VC6.4/ Vyatta-Firewall_R6.4_v01.pdf

Capítulo VII: Anexos.

1. Práctica # 01

Instalación y Configuración de Vyatta 6.4 en VMware Workstation 9

Objetivos:

- Usar VMware Worstation9 para crear máquinas virtuales.
- Instalar Vyatta en Máquina Virtual.

Introducción:

Con la implementación de esta práctica se pretende que el estudiante se familiarice con los comandos básicos de configuración, así como aprender a utilizar como instalar o crear máquinas virtuales en VMware Worstation9.

VMware es un sistema de virtualización por software. Un virtualizador por software permite ejecutar (simular) varios computadores (sistemas operativos) dentro de un mismo hardware de manera simultánea, permitiendo así el mayor aprovechamiento de recursos. No obstante, y al ser una capa intermedia entre el sistema físico y el sistema operativo que funciona en el hardware emulado, la velocidad de ejecución de este último es menor, pero en la mayoría de los casos suficiente para usarse en entornos de producción.

a) Primero descargaremos el livecd de Vyatta del sitio web oficial ---- así mismo VMware.

http://www.vyatta.org/downloads

https://my.vmware.com/web/vmware/login

b) Instalar VMware Workstation 9 en nuestro sistema (Ubuntu 12.10).

c) Una vez instalado VMware procederemos a crear nuestra primera máquina virtual de Vyatta. Ya iniciado VMware seleccionaremos la opción **Create a New Virtual Machine**.

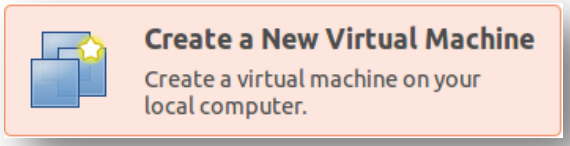


Figura22. Crear Nueva Máquina Virtual

d) Nos abrirá la siguiente ventana en la cual seleccionaremos el tipo de configuración para nuestra VM (máquina virtual) y dejaremos la opción por defecto que es la recomendada:

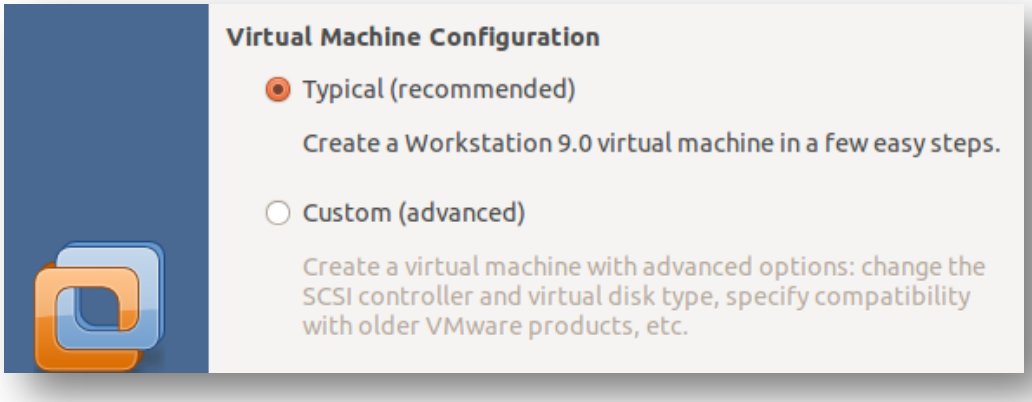


Figura23. Configurar Máquina Virtual

e) El siguiente paso es seleccionar la imagen de Vyatta con la cual instalaremos el sistema en nuestra VM. Seleccionar **Use ISO Image** y buscar el Livecd en nuestro ordenador.

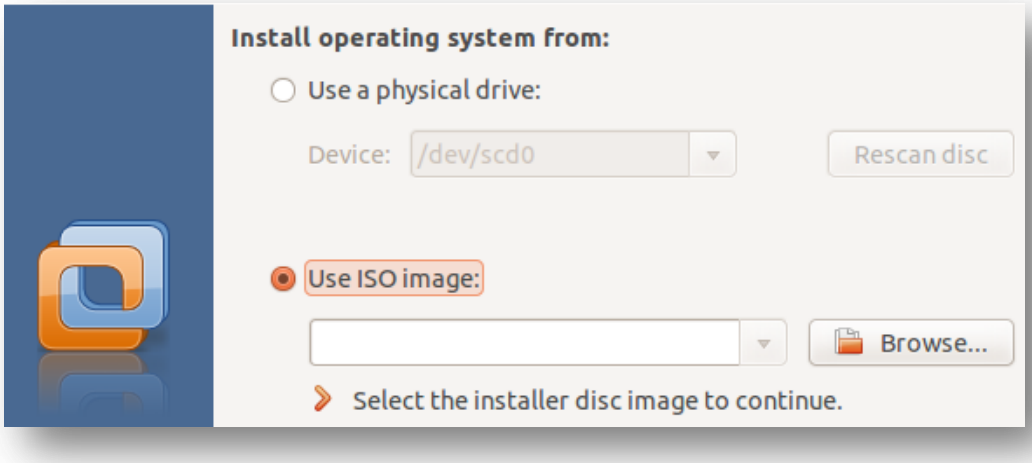


Figura24. Seleccionar Origen de Instalación del Sistema Operativo.

f) Seleccionar el Sistema Operativo de nuestra VM el cual será Linux y la versión Debían 6.

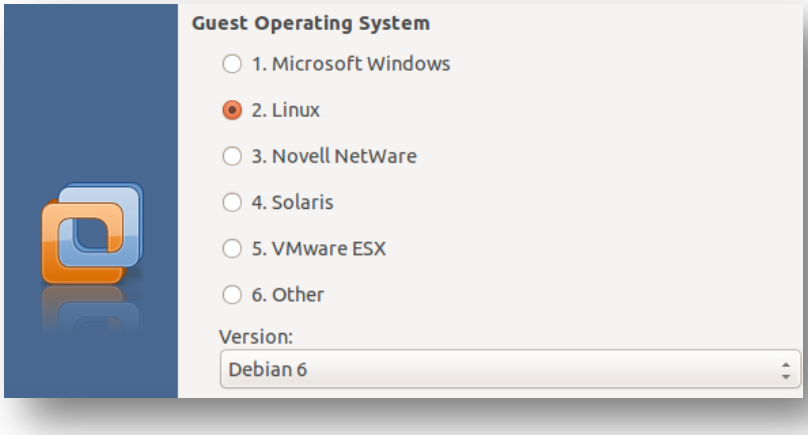


Figura25. Seleccionar Sistema Operativo Compatible

g) Luego nos pedirá el nombre de nuestra VM, la cual nombraremos **R1**. También en este paso podremos cambiar la localidad de nuestra VM, es recomendable dejar la ruta por defecto o bien cambiarla si así lo gustan.

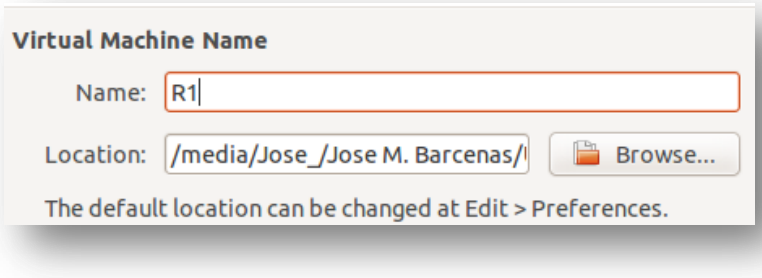


Figura26. Establecer Nombre a Máquina Virtual

h) Establecer el tamaño máximo del disco. Le asignaremos un tamaño de 8 GB (8.000).

Figura27. Asignar Tamaño Máximo a Disco Virtual

i) La siguiente ventana nos mostrara las características o configuraciones de nuestra VM.

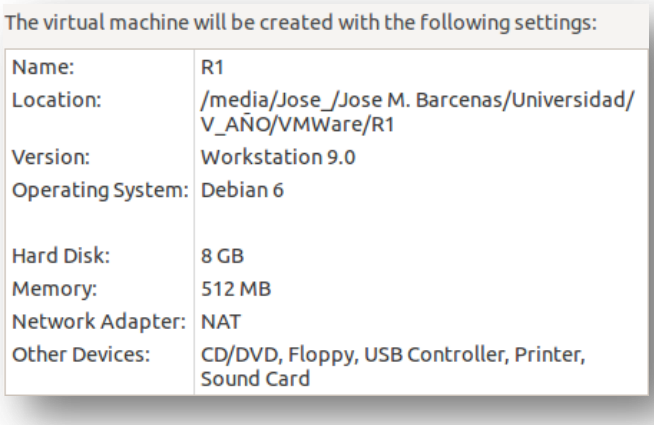


Figura28. Características y Configuraciones de Máquina Virtual

j) Luego, ya que el tamaño de memoria asignado por defecto es de 512 MB, lo cambiamos a 128 MB ya que con esta capacidad de memoria es suficiente para poner en marcha vyatta. Seleccionaremos el botón **Customize Hardware,** el cual nos mostrara la siguiente ventana, en donde asignaremos la cantidad de memoria antes mencionada:

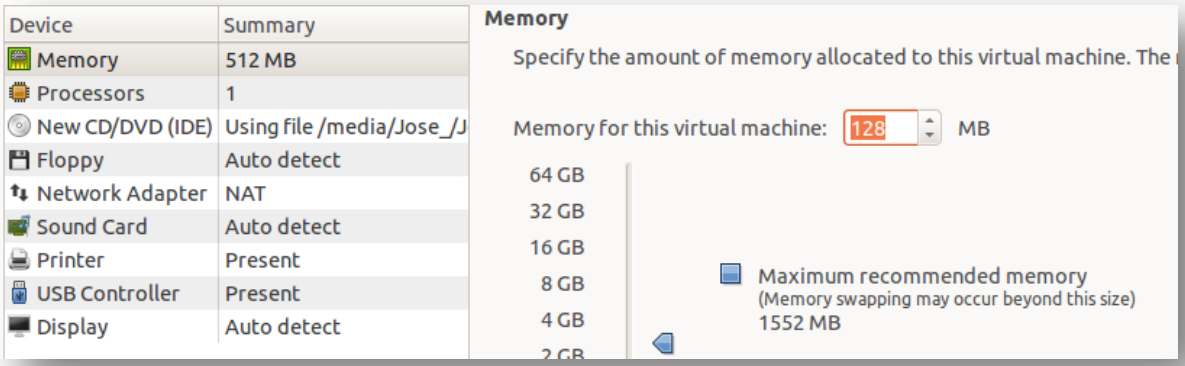


Figura29. Configuración General de Máquina Virtual

k) Una vez finalizado este paso ya estamos listo para iniciar nuestra VM, la cual iniciara automáticamente al finalizar con la anterior configuración. Por defecto al iniciar el nombre de usuario y la contraseña es **vyatta.** Luego se procederá con la instalación del sistema en el disco Duro de nuestra VM.

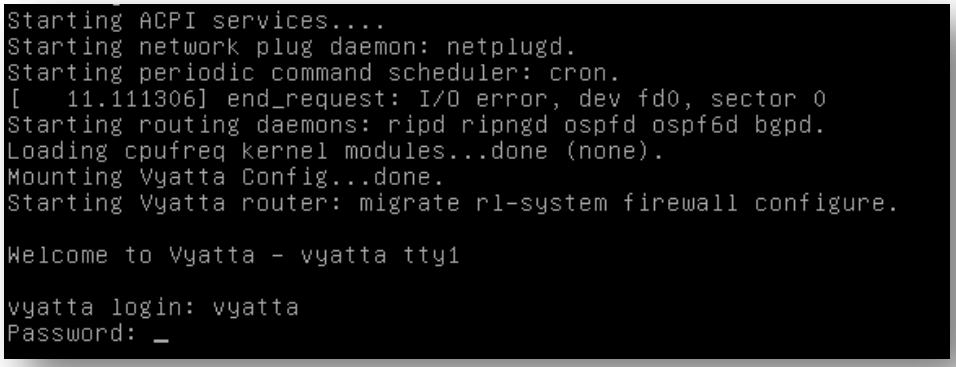


Figura30. Configuración Inicial de Vyatta en Máquina Virtual

l) Ya dentro del sistema Vyatta necesitamos ser el usuario root para proceder con la instalación del SO.

1) Iniciar como el usuario root: sudo su.

2) Instalar el Sistema: install system.

3) Responder Yes a la pregunta que a continuación se presenta.

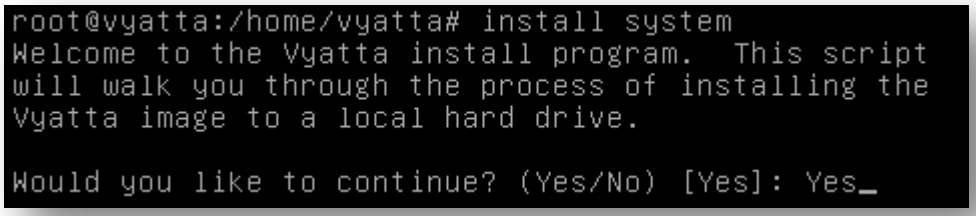


Figura31. Instalar Sistema Operativo Vyatta 6.4

m) Luego nos preguntara si queremos que las particiones del disco se realicen automáticamente o si queremos realizarlo manual, solo daremos enter ya q no queremos particionar la unidad. Encontrará una única partición y daremos enter nuevamente para que en esta se instale el sistema.

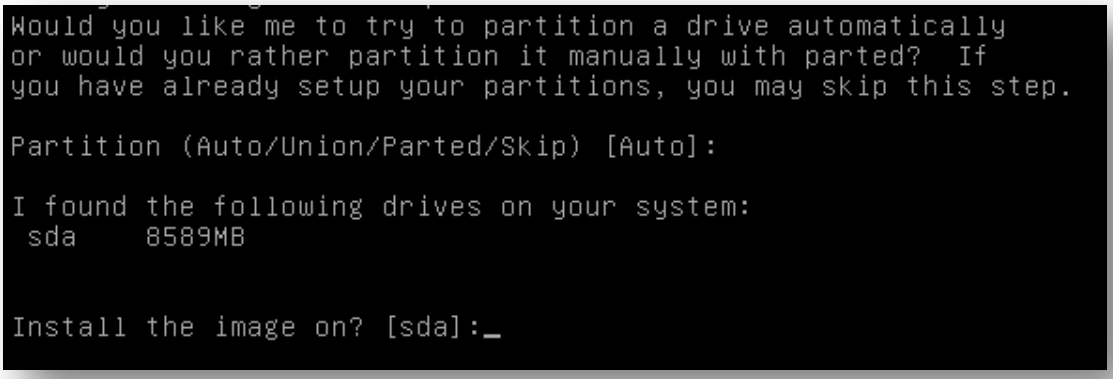


Figura32. Crear Particiones de Disco de Forma Automática

n) Nos informara que el siguiente paso destruirá todos los datos y si deseamos continuar, respondemos **yes,** y presionamos enter una vez más para indicar que deseamos ocupar todo el tamaño del disco para la instalación.

Figura33. Borrar Datos Existentes de Disco y Crear de Nuevo

o) Cuando la instalación del sistema de archivos finalice nos mostrara el archivo de configuración de Vyatta q se encuentra en el sistema y nos preguntara si de seleccionar otro, ya que no es el caso solo daremos enter.

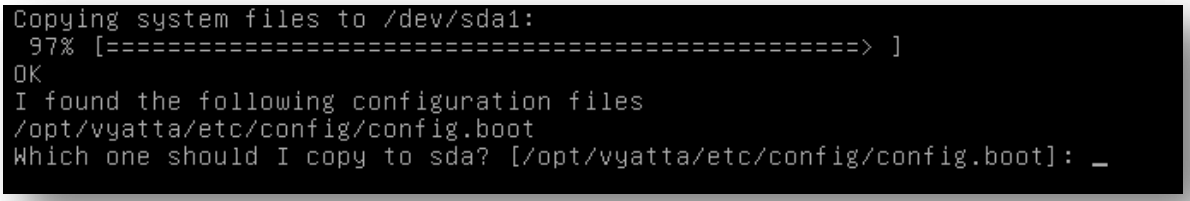


Figura34. Finalizar Instalación Vyatta 6.4

p) Luego nos pedirá la nueva contraseña para la cuenta del administrador del sistema, e introducimos la que mejor creamos conveniente.

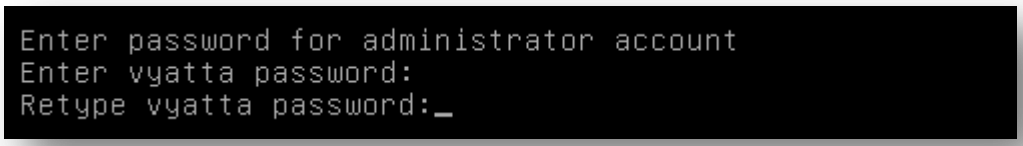


Figura35. Agregar Contraseña de Acceso a Vyatta 6.4

q) Luego daremos enter para que se instale el **GRUB Boot Loader** y con esto ya está nuestro sistema Vyatta instalado en la VM (Máquina Virtual)**.**

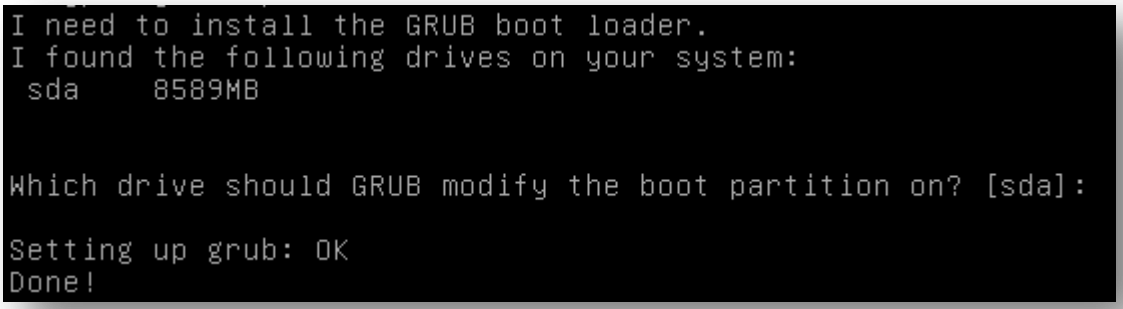


Figura36. Instalación de GRUB Boot Loader en Vyatta 6.4

2. Práctica # 02

Configuración de las interfaces de red de nuestra máquina virtual en VMware Workstation 9

Objetivos:

- Crear escenarios virtuales de segmentos de red con VMware Workstation 9.
- Aprender a configurar interfaces de red (virtuales) en entornos virtualizados con VMware Workstation 9.
- Asignar direcciones IP estáticas a las interfaces de red de nuestras máquinas virtuales, para lograr comunicación entre ellas.
- Iniciar a adquirir conocimientos de los diferentes comandos de configuración en Vyatta.

Introducción:

La compresión en la configuración de las interfaces de red es muy necesaria e importante, ya que nos permitirá crear las diferentes topologías de red, para la realización de las prácticas implementadas en el desarrollo del presente trabajo. Así mismo se está iniciando en el entorno de configuración de Vyatta, para de esta manera conocer algunos de los comando de configuración y comprobar la conectividad entre las diferentes redes creadas.

a) Para iniciar debemos volver a realizar todos los pasos mostrados en la práctica anterior, para crear otra VM de Vyatta la cual nombraremos R2, con el propósito de comunicar R1 y R2 a través de la interfaz red a configurar.

b) Una vez creada configuraremos una interfaz de red **Host only(Adaptador Anfitrión),** para esto en el menú ***Edit*** seleccionares la opción **Virtual Network Editor** y se nos abrirá la siguiente ventana:

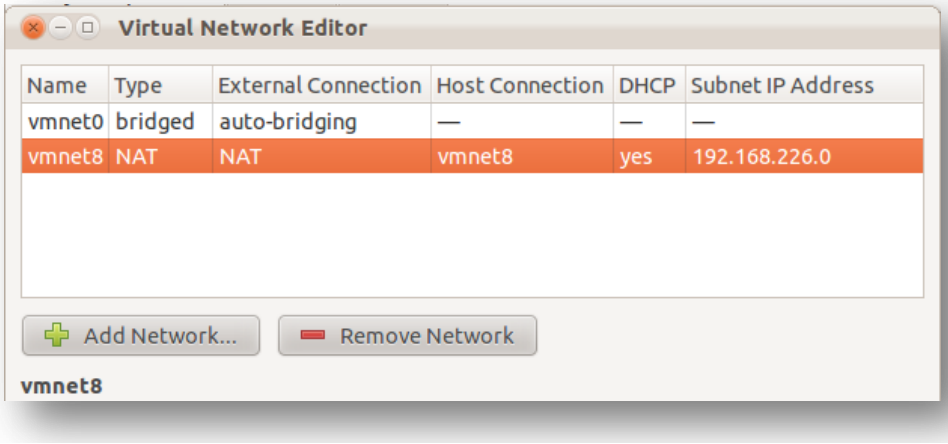


Figura37. Agregar Interfaz de Red en VMWARE WORSTATION 9

c) Clic en el botón **Add Network,** y luego en la siguiente ventana seleccionamos **Host-Only** y en el nombre de la red dejamos el que aparece por defecto que en este caso será **vmnet1** y le damos en **Add**.

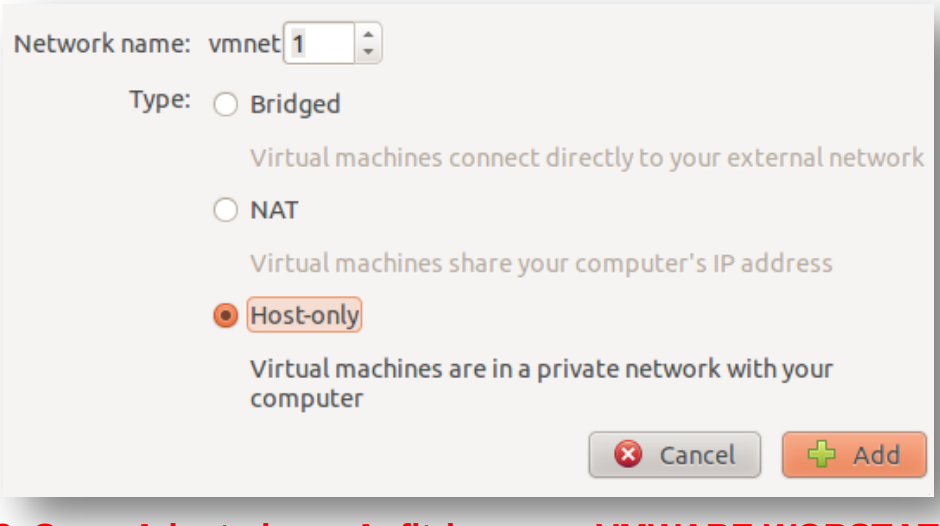


Figura38. Crear Adaptadores Anfitriones en VMWARE WORSTATION 9

d) Configuraremos la dirección de red a la que pertenece el adaptador la cual será 192.168.0.0 y deshabilitaremos la asignación de direcciones por DHCP.

Name	Type	External Connection	Host Connection	DHCP	Subnet IP Address
vmnet0	bridged	auto-bridging	—	—	—
vmnet1	host-only	none	vmnet1	no	192.168.0.0
vmnet8	NAT	NAT	vmnet8	yes	192.168.226.0

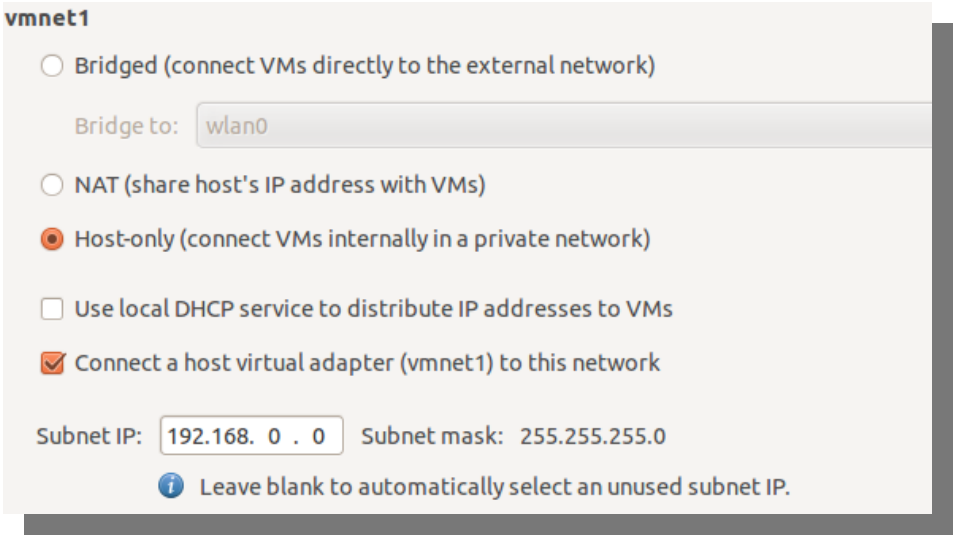


Figura39. Asignar Segmento de Red al Nuevo Adaptador en VMWARE

e) Encenderemos nuestras VMs tanto R1 como R2 y como usuario root ejecutaremos el comando **ifconfig** para ver nuestra interfaz de red de nuestras maquinas la cual no tendrá dirección Ip.

```
vyatta@vyatta:~$ sudo su
root@vyatta:/home/vyatta# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:77:0b:93
          inet6 addr: fe80::20c:29ff:fe77:b93/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:478 (478.0 B)
          Interrupt:19 Base address:0x2000
```

Figura40. Verificar Interfaz de Red asignadas en Vyatta 6.4

f) Para configurar la dirección IP de nuestra interfaz necesitamos entrar en el modo configuración de Vyatta lo cual conseguimos ejecutando como root el comando **configure.**

```
root@R1:/home/vyatta# configure
[edit]
root@R1# _
```

Figura41. Acceder al Modo Configuración de Vyatta 6.4

g) Ya en el modo configuración asignaremos la dirección de nuestra interfaz de red **eth0** para la cual ejecutaremos los comandos que siguen a continuación, en el Router R1:

1) Para establecer la dirección de la interfaz eth0:

```
vyatta@R1:# set interfaces Ethernet eth0 address 192.168.0.10/24
```

2) Para confirmar la configuración realizada:

```
vyatta@R1:# commit
vyatta@R1:# save
```

3) Verificamos nuevamente la dirección de la interfaz, y veremos los cambios.

```
root@R1# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:77:0b:93
          inet addr:192.168.0.10  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe77:b93/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:478 (478.0 B)
          Interrupt:19 Base address:0x2000
```

Figura42. Verificar Dirección IP Asignada a las Interfaces en R1

4) Realizar los mismos pasos para asignar la dirección **ip 192.168.0.20** para el router **R2** y mostrar los resultados:

```
vyatta@R2:# set interfaces Ethernet eth0 address 192.168.0.10/24
vyatta@R2:# commit
vyatta@R2:# save
```

a. Verificamos la dirección de la interfaz eth0 para **R2**

```
root@R2# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:98:ac:72
          inet addr:192.168.0.20  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe98:ac72/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:548 (548.0 B)
          Interrupt:19 Base address:0x2000
```

Figura43. Verificar Dirección IP Asignada a las Interfaces en R2

5) Ahora realizaremos pruebas de conectividad entre nuestras máquinas virtuales (ping de R1 a R2 y viceversa).

a. Ping de R1 a R2 (**ping 19.168.0.20**).

```
root@R1# ping 192.168.0.20
PING 192.168.0.20 (192.168.0.20) 56(84) bytes of data.
64 bytes from 192.168.0.20: icmp_req=1 ttl=64 time=0.734 ms
64 bytes from 192.168.0.20: icmp_req=2 ttl=64 time=1.10 ms
64 bytes from 192.168.0.20: icmp_req=3 ttl=64 time=0.635 ms
64 bytes from 192.168.0.20: icmp_req=4 ttl=64 time=0.678 ms
```

Figura44. Verificar Conectividad de R1 Hacia R2

b. Ping de R2 a R1 (**ping 19.168.0.10**).

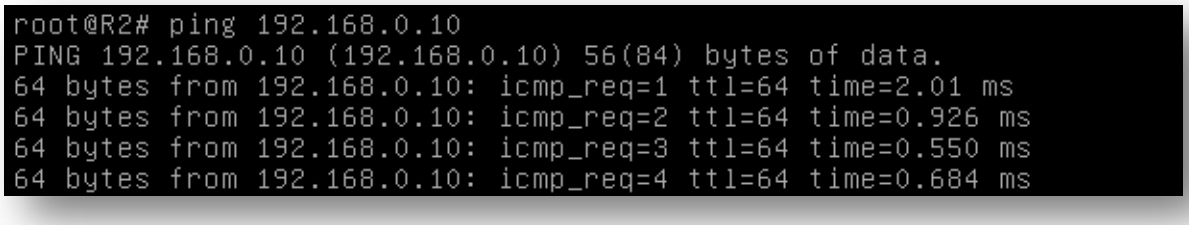


Figura45. Verificar Conectividad de R2 Hacia R1

Con estos pasos ya sabemos cómo comunicar nuestras máquinas virtuales Vyatta 6.4 en VMware Workstation 9.