Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

"Instalación y Configuración de Software Libre Zebra sobre sistema operativo Linux"

Trabajo de graduación previo a la obtención del título de Ingeniero de Sistemas

Autores: **Priscila Fernanda Andrade Vélez**

Cesar William Minga Espinosa

Director: Ing. Pablo Esquivel

Cuenca, Ecuador
2006

DEDICATORIA

El decir gracias últimamente se ha vuelto tan fácil que su significado ya no lleva la importancia que representa esa palabra. Aunque me doy cuenta de eso se también que cuando esa palabra viene desde el corazón y sale por los labios engloba tantas cosas que se o mejor podría decir estoy segura que a los que se los diga se darán cuenta de el gran sentimiento que va englobado en esas 7 letras.

Tenia que comenzar diciendo gracias para luego dedicarles este trabajo aunque se que talvez todo lo escrito y descrito en este no sea de su total conocimiento, pero si se lo que ustedes han hecho por mi para que yo pueda desarrollar este documento y eso es mas valioso que todo lo que pude haber aprendido en 5 años y un ciclo de la U. Eso fue lo que me daba fuerzas y más ganas para seguir adelante aunque no faltaban las rabietas, mal genios, discusiones, salidas continuas, excusas tontas, etc., pero al final de este proceso se impregno todo su sacrificio y esfuerzo en su pequeña.

Ya voy a ser ingeniera y todo es por y para ustedes, saben ustedes estuvieron en mi siempre en mis amanecidas aunque dormían igual tenían al día siguiente ojeras como si hubiesen estado junto a mi pero ya todo termino y ya soy ingeniera aunque no lo crean pero ya llego la hora de reflejar todo este tiempo de estudios y preparación para el mundo.

Ahora remplacemos ese famoso ustedes que he estado nombrado a cada momento en estos párrafos, quienes mas pueden ser que MA y PA, para ustedes va este trabajo y todo lo que SOY.

Una última línea SUCO nunca me cansare de decir, “Mi hermano si que es una bextia, bien pilas, realmente mis respetos” aunque no te lo he dicho pero ya es hora que te enteres, nunca es tarde.

PICHI

DEDICATORIA

Dios eres parte de cada esfuerzo realizado durante todos estos años y en esta ocasión te hago presente mi agradecimiento por todo el bien que has hecho conmigo, eres incomparable.

A mis padres José Minga y Alicia Espinosa, quienes que con gran desvelo y cobijo se preocuparon para que sea una persona útil a la sociedad.

Ahora que termino los estudios Superiores me doy cuenta que no sería un Profesional sino fuera por la ayuda hermosa y sacrificada que siempre me pudieron brindar en buenos y malos momentos. Por esta razón les dedico este trabajo que es él símbolo de agradecimiento por lo que han hecho por mi, los amo.

CES@R – CUCHUP 100% ORENSE

AGRADECIMIENTO

Queremos dar GRACIAS a nuestros profesores desde primer a décimo ciclo, agradecemos también a los profesores del curso que tomamos en Argentina ya que nos ayudaron a ampliar y a reforzar más nuestros conocimientos que de hoy en adelante reflejaremos al mundo. Y un agradecimiento muy especial a nuestro director de tesis Ing. Pablo Esquivel que nos tuvo que soportar con continuas preguntas e inquietudes durante el proceso de desarrollo de este documento.

PICHI Y CESAR

Las ideas, hechos y contenidos de esta tesis son de exclusiva responsabilidad de los autores

.....	.....
Priscila Andrade V.	Cesar Minga E.

INDICE

1. INTRODUCCION

Los avances tecnológicos a diario tienen un avance sin medida y este obviamente representa una inversión significativa que para una empresa mediana o pequeña son gastos que no están listas para cubrir. Pero la necesitad de interconectarse y tener acceso a la información de diferentes departamentos en tiempo real es decir la necesidad de diseñar, mantener y configurar una red es indispensable lo cual ha hecho que busquen otras formas de suplantar lo costoso que puedes representar la adquisición de equipos necesarios para este fin ya sean *hubs, switchs, routers*, etc.

Al analizar las necesidad inminente de compartir información y recursos hacen que se busquen otras formas de realizar el mismo objetivo pero obviando la adquisición de hardware con costos representativos así se llega a una muy buena opción que es la simulación de hardware sobre PCs los cuales no exijan un hardware y software base exagerado, ni costoso es decir cuyas características estén presentes en un CPU Terminal cualquiera. Al decir simulación de hardware es obvio tener presente el uso de software, en el mercado la presencia de este es muy amplia para resolver las necesidades de los usuarios pero lamentablemente no es totalmente explotada por muchas razones entre ellas la falta de conocimiento, o el temor al cambio lo cual puede ser el mayor error de los usuarios ya que simplemente no quieren experimentar con nuevas herramientas las cuales puedas representar ahorro de tiempo y dinero.

El Internet se ha convertido en una herramienta básica tanto a nivel empresarial, como de usuarios en general pero hay que destacar que se desconoce una valiosa ayuda que nos brinda es decir el acceso a software pero algo mas importante aun el hecho de que son de libre distribución, lo que al usuarios representa solo una descarga del sitio autorizado y ponerlo en funcionamiento.

Teniendo presente estos y otros beneficios que se pueden tener mediante el uso de software libre vemos la necesidad de dar a conocer uno de ellos ya que de esta manera se puede alentar ala investigación, desarrollo y uso de software libre. De esta manera tener la capacidad de aceptar, rechazar y opinar sobre ellos. Al darnos cuenta de lo expuesto hemos ahondado en un software libre denominado Zebra.

Con Zebra se pretende que las PCs sean usadas como un servidor de rutas (router server) y como un reflector de rutas (router reflector) también como lo haría un router normal. Zebra es un paquete de software de encaminamiento que proporciona encaminamiento basado en servicios de TCP/IP con protocolos de encaminamiento Este software es muy flexible con el hecho que se puede escoger con que protocolo poder configurar el router (PC), dependiendo de las necesidades y complejidad que se necesite.

Una PC con este software simula completamente a un router CISCO por lo que la configuración de este software es sencilla y útil. Es una muy buena opción para implementar y mantener una red siempre en funcionamiento ya que es un software dinámico que actualiza rutas y problemas que se presentan en la red de manera automática. Sintetizando podemos decir que es una herramienta productiva, gratuita y sobre todo al alcance de todos.

RESUMEN

El uso de software libre hoy en día no esta siendo muy explotado en el mercado a pesar de que algunos proporcionan productividad y garantía por lo que lo que pretendemos es fomentar el uso de software que se encuentra a nuestro alcance, a la vez que pueden ayudarnos a solucionar problemas de una forma rápida sin representar altas inversiones de dinero ni mano de obra.

Zebra es un paquete de software de routing que provee TCP/IP basado en los servicios de routing con varios protocolos soportados como: RIP, OSPF y BGP, además la posibilidad de implementar estos protocolos de routing con IPV4 o IPV6. Lo que pretende es que un PC se convierta o simule las funciones de un router dinámico permitiendo así tener actualizadas las rutas más próximas de una red.

Al realizar la practica nos dimos cuenta que no siempre la adquisición de un nuevo y moderno hardware solucionan problemas ni cubren necesidades inmediatas, creemos que el hecho de investigar, analizar, aplicar conocimientos adquiridos y utilizar herramientas de libre distribución ayudarían mucho en el desempeño de nuestras labores tanto académicas como laborales.

ABSTRACT

Nowadays using free software isn't exploded enough in the market, although some of them provide productivity and guarantee. For that reason we want to foment the use of software which is to our reach. Those can help us to solve problems of a fast way without huge investments of neither money nor labour.

Zebra is a software package of routing that provides TCP/IP based on the services of routing, it support many protocols like: RIP, OSPF and BGP, in addition it can implement these routing protocols with IPV4 or IPV6. This software pretends that a PC becomes or simulates the functions of dynamic router it means to have updated the next path of a network.

When we are making the practice job we concluded that the acquisition of a new and modern hardware can't solve problems nor cover immediate necessities. We think that the fact to investigate, to analyze, to apply acquired knowledge and to use tools of free distribution would help in a better way in the performance of our academic workings as much as labour.

2. ZEBRA

2.1.1 Introducción

El proyecto de *Zebra* empieza en el año 1996, la idea original de zebra fue del Sr. *Kunihiro Ishiguro*, trabajando para la *ISP*, *Ishiguro* comprendió la gran necesidad de crear un nuevo tipo de software para *routing*. Se unió al proyecto *Yoshinari Yoshikawa*, estos unirían sus conocimientos y recursos para crear el primer software de ingeniería en *routing* basado en la *GNU* (*General Public License*). Esta entidad llamada Proyecto *Zebra* consiste en la especialización comercial de *IP Infusión* combinada con las técnicas de ingeniería de encaminamiento.

El Sr. *Kunihiro Ishiguro* fundador del proyecto *Zebra* a tenido una especialización en Dominios y *Routing*, miembro activo de la *JPNIC* (Cuerpo de Administración de Dominios y Direcciones IP de Internet para el Japón).

2.1.2 Definición

Zebra es un *software* de *routing* (Encaminamiento de paquetes de datos) de disponibilidad pública, la cual esta distribuida bajo licencia *GNU* (Licencia General Pública) es decir de libre distribución, único en su diseño ya que tiene un proceso para cada protocolo que corre en un *Kernel Linux* ya sean estos de encaminamiento basados en *IPV4* o *IPV6*. Entre los protocolos de encaminamiento mencionamos los siguientes:

- Rip
- Ripng
- Ospf
- Bgp
- Ospf6

Zebra posee una interfaz de usuario interactivo para cada protocolo de *routing* y soporta comandos de cliente en sus interfaces. Debido a su diseño es posible añadir nuevos protocolos fácilmente, con este software, una máquina intercambia información de routing con otros routers, lo cual permite la configuración dinámica entre estos, además es posible ver la información de la tabla de *routing* a través de su interfaz de terminal. Se puede configurar las direcciones de las interfaces, rutas estáticas y muchas más cosas.

En Los Sistemas Operativos *Linux* la configuración de un router se realizaba mediante los comandos *ifconfig* y los comandos del tipo *route*, el estado de las tabla mediante el comando *netstat*, estos comandos solo se los podían utilizar operando como *root*, sin embargo *Zebra* nos da una gran facilidad para su administración, el usuario podrá realizar sus operaciones mediante dos modos.

- Modo normal
- Modo de enable (habilitado)

El usuario de modo normal únicamente puede ver el estado del sistema, sin embargo el usuario de modo enable puede cambiar la configuración del sistema mostrando así una interfaz de configuración basada en el *Cisco IOS*. Actualmente, *Zebra* soporta los protocolos de *unicast* más comunes, se ha diseñado para utilizarse como *Route Server y Route Reflector*.

"El objetivo de Zebra es conseguir un software de routing productivo de calidad y gratuito".

2.1.3 Arquitectura

La arquitectura es de gran calidad, con un motor *multi servidor*, formado por un conjunto de demonios (programas que escuchan en puertos consecutivos) para efectuar el encaminamiento de paquetes y así construir la tabla de *routing*.

El demonio *ripd* maneja el protocolo *RIP*, mientras que el demonio *ospfd* controla el protocolo *OSPFv2*, *bgpd* soporta el protocolo *BGP-4,* para cambiar la tabla de routing del kernel y la redistribución de rutas entre distintos protocolos de routing, tenemos la tabla de routing del kernel controlada por el demonio zebra.

El administrador del router será capaz de ejecutar un determinado demonio y enviar reportes a la consola central de routing, no es necesario ejecutar esos demonios en la misma máquina. Es posible ejecutar varias instancias del mismo demonio de routing en la misma máquina.

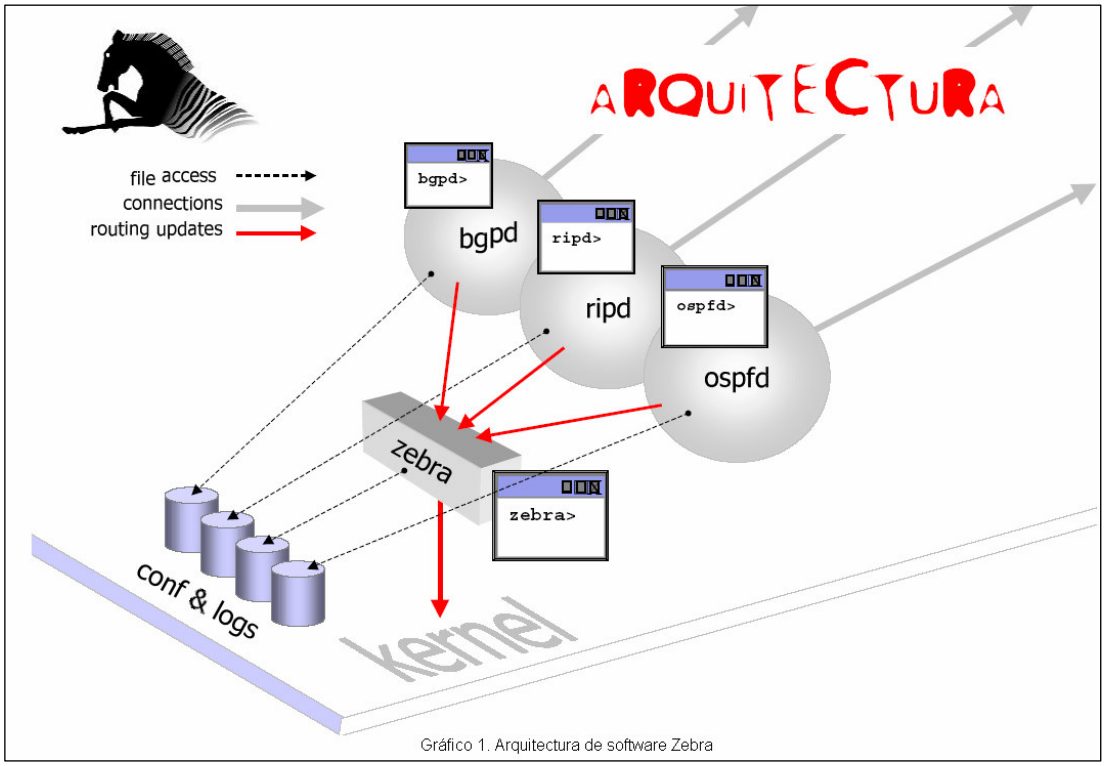

Gráfico 1. Arquitectura de software Zebra

La arquitectura multiproceso permite tener un sistema parcialmente modular es decir nos permite tener varios ficheros de configuración e interfaz de terminal, en los dispositivos routers básicamente existen dos maneras de enrutar a otros *hosts* fuera del nodo local y son utilizando enrutamiento estático y enrutamiento dinámico, cada método tiene ventajas e inconvenientes, pero cuando una red crece finalmente el enrutamiento dinámico es la única manera factible de gestionar la red.

Por este motivo se plantea la necesidad de utilizar protocolos de enrutamiento dinámico en vez de usar rutas estáticas en todos los nodos. Existen programas para llevar el enrutamiento dinámico en la mayoría de los sistemas operativos como *Zebra*.

El uso del enrutamiento dinámico evitará las modificaciones manuales y asegurará que la conexión a nuevos nodos sea inmediata en toda la red. Por este motivo se recomienda su uso cuando sea posible.

Debido a que Zebra posee demonios para cada protocolo de enrutamiento, estos tienen su propio archivo de configuración e interfaz de terminal, cuando queremos configurar una ruta estática, se la realiza en los archivos de configuración (.conf) de cada demonio, por ejemplo si queremos configurar de forma estática una red Rip

hay que hacerlo en el fichero de configuración Rip, esto es bastante fastidioso, para solucionar este inconveniente existe un interfaz *shell* integrado llamado vsh. vsh conecta cada demonio funcionando como un *proxy* para la entrada del usuario.

Al ejecutarse *Zebra* bajo el kernel *GNU/Hurd* actuara como tabla de routing del *kernel*, aquí se proporcionan todos los servicios de *TCP/IP* por los procesos de usuarios llamados *pfinet*.

GNU/Zebra ha sido probado en los sistemas operativos como:

GNU/Linux 2.0.37
GNU/Linux 2.2.x
GNU/Linux 2.3.x
FreeBSD 2.2.8
FreeBSD 3.x
FreeBSD 4.x
NetBSD 1.4
OpenBSD 2.5
Solaris 2.6
Solaris 7
Centos 4.0

2.1.4 RFC soportados en Zebra

Los *RFCs* soportados por Zebra son los siguientes:

- *RFC1058 - Routing Information Protocol. C.L. Hedrick. 1 de Julio de 1998*
- *RFC1771 - A Border Gateway Protocol (BGP-4). Y. Rekher & T. Li. Marzo 1995*
- *RFC1997 - BGP Communities Attribute. R. Chandra, P. Traina & T.Li. Agosto 1996*
- *RFC2080 - RIPng for IPv6. G.Malkin, R.Minnear. Enero 1997*
- *RFC2283 - Multiprotocol Extension for BGP-4. T.Bates, R.Chandra, D.Katz, Y. Rekhter. Febrero 1998*
- *RFC2328 - OSPF Version 2. J. Moy. Abril 1998*
- *RFC2453 - RIP Versión 2. G.Malkin. Noviembre 1998*
- *RFC2545 - Use of BGP-4 Multiprotocol Extension for IPv6 Inter-Domain Routing. P. Marques, F. Dupont. Marzo 1999*

- *RFC2740 - OSPF for IPv6. R. Coltun, D. Perguson, J.Moy. Diciembre 1999.*
- *RFC2796 - BGP Route Reflection An alternative to full mesh IBGP T. Bates R. Chandrasekeran. Junio 1996.*
- *Cuando está habilitado el SNMP soportado, la siguiente RFC está soportada*
- *RFC1227 - SNMP MUX protocol and MIB. M.T. Rose. Mayo 1991.*
- *RFC1657 - Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2. S. Willis, J.Burruss, J. Chu, Editor. Julio 1994.*
- *RFC1850 - OSPF Version 2 Management Information Base. F. Baker, R. Coltun. Noviembre 1995.*

2.1.5 Conclusiones

Zebra es un paquete software de routing que provee TCP/IP basado en los servicios de routing con varios protocolos soportados como pueden ser RIP, OSPF y BGP. Soporta además la posibilidad de implementar estos protocolos de routing con IPV4 o IPV6.

Con este software se pretende que las PCS sean usadas como un servidor de rutas (router server) y como un reflector de rutas (router reflector) también como lo haría un router normal.

3. DEMONIOS Y SUS PROTOCOLOS

3.1.1 Introducción

Demonio es como un programa que permanece en segundo plano ejecutándose continuamente para dar algún tipo de servicio. Zebra utiliza 7 de ellos los cuales están relacionados a un protocolo en específico a la vez que asigna un puerto sobre el cual escuchara. Cada demonio tiene su propio archivo de configuración (.conf) sobre los cuales se graban las actualizaciones que se realizan en modo privilegio con los comandos que se describirán mas adelante.

3.1.2 Tipos de demonios en Zebra

Zebra instala 7 demonios los cuales podemos ver en el archivo de servicios del sistema, el directorio en el cual están instalados estos servicios es: /etc/vi services.

Estos demonios son:

- zebrasrv
- zebra
- ripd
- ripng
- ospfd
- ospf6d
- bgpd

Hay cinco demonios en uso. Cada uno de estos demonios escuchará en un puerto particular para las conexiones de VTY entrantes. Los demonios de routing son:

Zebra	2601 tcp
Ripd	2602 tcp
Ripngd	2603 tcp
Ospfd	2604 tcp
Bgpd	2605 tcp

3.1.2.1 Zebra

Es el demonio gerente que se utiliza con el resto de demonios. Cada demonio de protocolo envía las rutas al demonio *Zebra*, es decir gestiona qué rutas están instaladas en la tabla de *forwarding* (reenvió).

Este demonio es el primero ha ser invocado, ya que una vez comenzado los otros archivos lo leer y revisan la configuración inicial. Tiene su propio archivo de configuración que es el zebra.conf el cual se encuentra ubicado en /usr/local/etc. Zebra –d esta forma de invocar puede ser aplicado al resto de demonios indicando el demonio a invocar y la opción –d.

3.1.2.1.1 Arranque

Zebra posee una interfaz denominada *VTY (INTERFACE VIRTUAL DE TERMINAL)*, esta presta facilidades para la administración del router el funcionamiento es el mismo al utilizar el protocolo *telnet*, es decir que se puede conectar al demonio vía protocolo *telnet* dentro de la terminal *VTY*.

Accedemos a zebra mediante el protocolo *Telnet* de la siguiente manera:

[root@price cesar]# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.92a).
Copyright 1996-2001 Kunihiro Ishiguro.

User Access Verification

Password:
Router>

3.1.2.1.2 Comandos Modo Terminal

Estos son comandos en los que muestra solo información de la configuración del demonio, no se pueden cambiar configuraciones. El signo de interrogación? nos permite desplegar en pantalla una lista de comandos que se pueden utilizar el modo previsto.

Router> ?

enable	cambia al modo privilegio
exit	sale del modo actual
help	*Ayuda*
list	*presenta lista de comandos*
quit	sale del modo que esta corriendo y queda en el modo anterior
show	muestra información del sistema que esta corriendo con siguientes opciones:
	version información del paquete
	history crea bitácora de comandos utilizados
	interface muestra la configuración de los interfaces del router
terminal	fija parámetros para línea terminal
who	presenta quienes están en vty

Router>

3.1.2.1.3 Privilegios de usuario

Dentro del router existen dos modos para que el usuario trabaje.

Los modos pueden ser:

- Modo deshabilitado.
- Modo Habilitado.

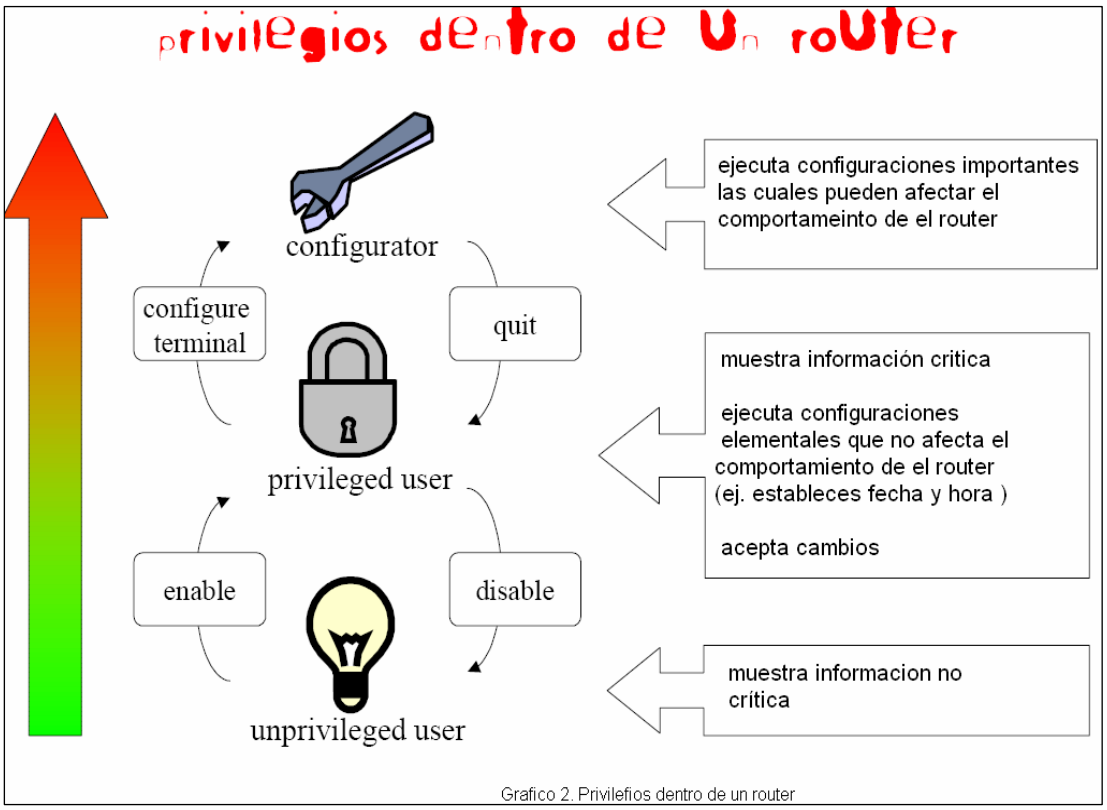

Grafico 2. Privilefios dentro de un router

Al iniciar el router por *default* se inicia en modo deshabilitado, el cual nos permitirá mostrar la configuración establecida hasta el momento.

Para cambiarnos de modo ejecutamos el comando *enable* seguido de su contraseña, esto nos permitirá cambiarnos al modo de habilitado, permitiendo realizar cualquier tipo de configuración en el router.

3.1.2.1.4 Comandos de interfaz

El signo numeral # nos indicará que nos encontramos en modo habilitado.

Router> enable

Password:

Router# ?

configure	configura desde interfaz vty
	terminal estado para el modo de configuración
	de vty
copy	copia la configuración
debug	depurar funciones (ver 'undebug')
	zebra
	events

kernel

packet

disable salir de modo privilegio

end *salir del modo que esta corriendo*

exit *sale del modo que esta corriendo y queda en el modo anterior*

help *ayuda*

list *muestra lista de comandos*

no niega comando

zebra

events

kernel

packet

quit sale del modo que esta corriendo y queda en el modo anterior

show muestra information del sistema

ip route muestra la tabla de rutas ip, rutas que actualmente tiene zebra en su base de datos

terminal se establece en la línea terminal

who muestra quienes están en vty

write guarda los cambios en la terminar, en archive o e en la interfaz

file

memory

terminal

Router#

Hay que tener en cuenta que los comando expuestos anteriormente no son todos los que existen ya que cada vez que se acceden a uno de ellos mas opciones van apareciendo las cuales podemos ver con (?), teniendo en cuenta que cuando termine todas las opciones de un comando se encontrara <cr>.

Para configurar una interfaz primero entrar al modo configure terminal (router1(config)#), dentro de este tenemos las siguientes opciones:

Interface nombre_de_interfaz: Nombre de la interfaz a configurar

Una vez dentro de la interfaz tenemos otras opciones:

Description descripción:	permite establecer una descripción para la interfaz
ip address direccion/mascara	Especifica la dirección y su mascara para la interfaz mencionada anteriormente
Show run	Nos enlista la configuración actual del proceso, nos permite ver las interfaces y las rutas definidas.
write file	Nos enlista la configuración actual del proceso, nos permite ver las interfaces y las rutas definidas. Para escribir en archivo la configuración actual.
Shutdown	Levanta la interfaz
no shutdown	Da debaja la interfaz
multicast:	Habilita la bandera de multicast para el interfaz.
no multicast	Deshabilita la bandera de multicast para el interfaz

3.1.2.1.5 Seguridad y tiempo de conexión

service password-encryption	Encripta el password
exec-timeout minute exec-timeout minute second	Establece el tiempo de conexión en los VTY. Cuando sólo se especifica un argumento, este se utiliza como el tiempo máximo en minutos. Opcionalmente se puede añadir un segundo argumento que es utilizado como tiempo en segundos. El valor por defecto es de 10 minutos. Si se configura un tiempo 0, este es entendido como que no hay límite de tiempo. no exec-timeout: Este comando es igual a exec-timeout 0 0

3.1.2.2 RIP (Routing Information protocolo, protocolo de información de encaminamiento).

Es un protocolo de encaminamiento interno, es decir para la parte interna de la red, la que no está conectada al *backbone* de *Internet*. Es muy usado en sistemas de

conexión a *Internet* como infovia, en el que muchos usuarios se conectan a una red y pueden acceder por lugares distintos.

Es un protocolo de enrutamiento por vector de distancia que calcula las distancias hacia la máquina destino en función de cuántos routers debe atravesar un paquete para llegar a su destino (saltos), enviando cada paquete de datos por el camino que en cada momento muestre una menor distancia. *RIP* actualiza las tablas de enrutamiento a intervalos programables, generalmente cada 30 segundos. Es un buen protocolo de enrutamiento, pero necesita que constantemente se conecten los routers vecinos, generándose con ello una gran cantidad de tráfico de red.

3.1.2.2.1 Arranque

Ripd soporta *RIP* versión 2 tal y como viene descrito en el *RFC2453* y *RIP* versión 1 tal y como viene descrito en el *RFC1058*.

De la misma forma que accedimos a zebra es para rip, usaremos el protocolo telnet para ingresar al demonio ripd. RIP requiere información del interfaz mantenida por el demonio zebra es decir la ejecución previa es obligatoria, luego invocamos rip con ripd –d.

[root@price cesar]# telnet localhost ripd
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.92a).
Copyright 1996-2001 Kunihiro Ishiguro.

User Access Verification

Password:
ripd>

En rip existen dos modos de configuración:

- Habilitado

- Deshabilitado

Permitiendo de esta forma realizar las configuraciones pertinentes

3.1.2.2.2 Comandos Básicos de RIPD

ripd>enable

ripd #

clear	resetea una función
configure	configura desde interfaz vty
	terminal
	access list
copy	copia la configuración
quit	sale del modo que esta corriendo y queda en el modo anterior
list	*muestra lista de comandos*
exit	*sale del modo que esta corriendo y queda en el modo anterior*
show	muestra la tabla de rutas basadas en protocolo RIP
	ip rip
……	
……	
……	

Los demás comandos son los mismos que dentro del demonio zebra.

3.1.2.2.2 Filtros

Es posible negar y dar acceso a paquetes de diferentes redes mediante el comando access-list.

access_list ifname deny permit remark	Añade listas de acceso se define el nombre de la lista y una de las opciones ya sea *deny, permit o remark*

access-list private permit 10 10.0.0.0/8

access-list private deny any

Una vez creada la lista se debe asignar a una interfaz sobre la cual se aplicara,

además especificando para que paquetes se quiere aplicar ya sea de entrada o salida (in, out).

distribute-list private in eth0

3.1.2.2.3 Comandos de interfaz

router rip	Es necesario para habilitar RIP. Para deshabilitar RIP hay que utilizar el comando no router rip. RIP debe habilitarse antes de llevar a cabo cualquiera de los comandos.
no router rip	Deshabilita RIP

Al utilizar estos comandos entramos a la interfaz ripd (config-router)# dentro del cual podemos utilizar otros comandos como:

Network no network	Selecciona el interfaz habilitado por red. Los interfaces que posean direcciones que dicha red son habilitados. Para especificar una lista de redes directamente conectadas al equipo, por las cuales queremos que nuestro router encamine paquetes con el protocolo RIP.
network ifname no network ifname	Configura a habilitado el interfaz descrito como ifname. Se habilita tanto el envío como la recepción de paquetes RIP en el puerto especificado en el comando network ifname. El comando no network ifname deshabilita RIP en el interfaz especificado.
version version	Configura la versión del proceso RIP, esta puede ser "1" o "2".

3.1.2.2.5 Depurando paquetes

En los routers se pueden activar funciones de depuración para que nos digan lo que está haciendo ante ciertos paquetes o eventos. No es recomendable tenerlas activas más que cuando se busquen problemas de configuración o de red porque ralentizan el funcionamiento del router.

ROUTER (PC)

ripd(config)# debug rip zebra

```
ripd(config)# debug rip pack
ripd(config)# debug rip event
```

La primera de ellas hará que el router nos indique cuando envía o recibe paquetes de zebra. Con la segunda nos notificará cuando se produzcan cambios en los paquetes. Se desactivan sin más que poner no delante de cada una de ellas.

3.1.2.2.6 Anunciando rutas

Es importante redistribuir información desde las entradas de encaminamiento del kernel a las tablas de rip.

```
ROUTER (PC)
ripd(config-router)# redistribute kernel
ripd(config-router)# redistribute static
ripd(config-router)# redistribute connected
```

Redistribuye kernel.- redistribuye la información de encaminamiento desde las entradas de encaminamiento del kernel a las tablas de RIP, no redistribute kernel deshabilita esas rutas.

Redistribute static redistribuye la información de encaminamiento desde las entradas de rutas estáticas a las tablas de RIP, el comando no redistribute static deshabilita dichas rutas.

Redistribute connected.-Redistribuye las rutas de conexiones directas en las tablas de RIP. no redistribute connected deshabilita dichas rutas. Este comando redistribuye las rutas conectadas directamente al interfaz en el cual no está habilitado RIP. Esta funcionalidad está habilitada por defecto.

3.1.2.3 RIPNG

Soportado en zebra por el demonio Ripngd, es la adaptación del protocolo RIP a IPv6, es decir un tipo de direccionamiento de longitud mas grande.

La invocación del demonio ripngd es la siguiente:

Ripngd –d

Ripngd soporta los comandos que Ripd pero con la variación que al momento de especificar las direcciones de red estas deben estar con direccionamiento IPv6. Para configurar una red con este tipo de protocolo referirse a configuración con Ripd tomando en cuenta la variación descrita. (ver capitulo práctica)

3.1.2.3 *BGP (Border gateway protocol*, protocolo de la pasarela externa)

Es un protocolo muy complejo que se usa en la interconexión de redes conectadas por un *backbone* de internet. Este protocolo usa parámetros como ancho de banda, precio de la conexión, saturación de la red, denegación de paso de paquetes, etc. para enviar un paquete por una ruta o por otra.

Un router *BGP* da a conocer sus direcciones IP a los routers *BGP* y esta información se difunde por los routers *BGP* cercanos y no tan cercanos. BGP tiene sus propios mensajes entre routers, no utiliza *RIP*.
Este protocolo se maneja en base a números de sistema autónomo (*ASN*). Básicamente, lo que hace *BGP* es establecer una relación de pares entre dos routers que pertenecen a distintos sistemas autónomos y esos routers se informan entre si acerca de que redes conocen y como hacer llegar paquetes.
BGP es usado por grandes proveedores de conectividad a *internet*.

3.1.2.4.1 Comandos básicos

bgpd> ?

enable	cambia al modo privilegio
ex*it*	sale del modo actual
help	*Ayuda*
list	*presenta lista de comandos*
quit	sale del modo que esta corriendo y queda en el modo anterior
show	muestra información del sistema que esta corriendo con siguientes opciones:
	version información del paquete
	history crea bitácora de comandos utilizados

interface muestra la configuración de los interfaces del router

terminal fija parámetros para línea terminal

who presenta quienes estan en vty

bgpd>

Ingresando en el modo de configuración (bgpd(congif)#) en el cual tiene comandos básicos de configuración tenemos opciones tales como:

access_list	añade listas de acceso
banner	
bgp	información de bgp
debug	depura funciones
dump	paquete dump
enable	modifica el password
end	termina el modo actual para cargar un nuevo
exit	salir del modo actual
help	ayuda
hostname	establece un nombre a la red del sistema
ip	imformacion ip
ipv6	informacion ipv6
line	configura una linea terminal
list	muestra lista de comandos
no	niega comandos
password	asigna una clave a la conexión de la terminal
quit	sale del modo actual
route-map	crea route-map o entra al modo de comando de route-map
router	habilita el proceso de routing
service	establece varios servicios
show	muestra información actual que esta corriendo
write	presenta la información actual del sistema que esta corriendo

Debemos tener en cuenta que se pueden ir desplegando mas comandos según se vayan digitando. Aquí se han presentado los principales de los cueles parten mas para mayor productividad de cada demonio.

3.1.2.4 *OSPF (Open shortest path first*, El camino más corto primero)

Se usa, como *RIP*, en la parte interna de las redes, su forma de funcionar es

bastante sencilla. Cada router conoce los routers cercanos y las direcciones que posee, además de esto cada router sabe a que distancia (medida en routers) están sus vecinos, así cuando tiene que enviar un paquete lo envía por la ruta en la que tenga que dar menos saltos.

Es un protocolo de *routing link-state* no propietario, es de libre uso y suele estar soportados por la mayoría de los equipos destinados a ofrecer servicios a la red y Segundo el ser un *link-state* quiere decir que a diferencia de *RIP* o *IGRP* que son *Distance-vector*, no mandan continuamente la tabla de rutas a sus vecinos sino que solo lo hacen cuando hay cambios en la topología de red, de esta forma se evita el consumo de ancho de banda innecesario. En un cambio de topología *OSPF* envía el cambio inmediatamente de forma que la convergencia de la red es mas rápida que en los distance-vector donde depende de *timers* asignados, de forma que en un *link-state* el tiempo de convergencia puede ser de 4 o 5 segundos comparado con *rip* que es de 180 segundos.

3.1.2.5 Comandos básicos

ospfd> ?

enable	cambia al modo privilegio
ex*it*	sale del modo actual
help	*Ayuda*
list	*presenta lista de comandos*
quit	sale del modo que esta corriendo y queda en el modo anterior
show	muestra información del sistema que esta corriendo con siguientes opciones:
	version información del paquete
	history crea bitácora de comandos utilizados
	interface muestra la configuración de los interfaces del router
terminal	fija parámetros para línea terminal
who	presenta quienes estan en vty

ospfd>

Al entrar al modo terminal de ospfd tenemos los mismos comandos que en bgpd por lo que resultaría redundante repetirlos.

NOTA: Cabe recalcar que nuestra practica esta orientada al uso exclusivo de

protocolo RIP de ahí el hecho que no hemos profundizado en los demonios ospf, ospfd.

3.1.3 Conclusión

Este software al tener varios demonios los mismos que están asociados a un protocolo en particular nos brinda movilidad y facilidad para configurar redes ya que no es necesario el conocimiento de todos ellos para su funcionamiento. Al ser la marca CISCO muy conocida en la actualidad permite a los administradores de red acoplarse rápidamente con esta herramienta, ya que los comandos utilizados en el mismo son iguales.

4. APLICACIÓN PRÁCTICA

4.1 Introducción

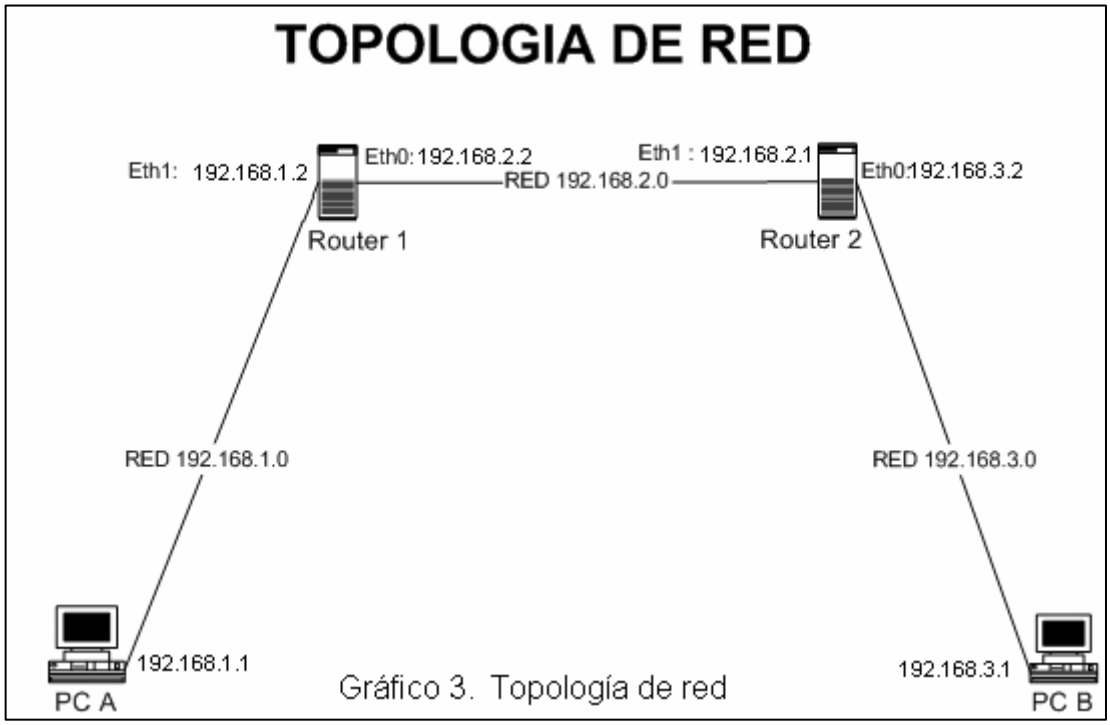

Gráfico 3. Topología de red

Diseñar e interconectar 2 LANS a través del uso de 2 PCS que simulan el funcionamiento de routers CISCO. Para lo cual utilizaremos software libre denominado “Zebra “.

4.2 Objetivo práctica

Demostrar la conectividad entre 2 redes, mediante la configuración de 2PCS que actúen como routers de pasarela interior bajo protocolo RIP.

Para la demostración de nuestro objetivo utilizaremos el comando PING.

4.3 Materiales

Hardware

2PCS S.O Windows XP

2PCS S.O Centos

tarjetas de red cada CPU

cables cruzados categoría UTP con conectores RJ45

Software

Zebra-0.94.tar.gz

Ethereal Version 0.10.14

Microsoft Visio Professional 2002 (10.0.525)
Centos 4.0

4.4 Instalación

4.4.1 ¿Qué paquetes instalar?

El Sistema operativo sobre el cual se va instalar Zebra es en Linux Centos que posee un núcleo bash y una arquitectura i386.

Para esta versión de sistema operativo hemos decidido instalar el siguiente paquete de zebra:

zebra-0.94.tar.gz

La dirección de donde se obtuvo este paquete es: www.zebra.org

4.4.2 **Pasos**:

1. Accedemos al sistema operativo como usuario root.
2. Colocamos el archivo ejecutable en el siguiente directorio:
   /usr/local/sbin/zebra-0.94.tar.gz
3. Con esta instrucción se descomprime el paquete

   tar xvfz zebra-0.94.tar.gz

   El comando **tar** agrupa uno o más archivos en un llamado paquete, que luego se puede comprimir.

   -f La información generada por el proceso se escribe en un archivo y no se muestra en la pantalla

   -x Desempaqueta archivos de un paquete

   -v Da los nombres de los archivos procesados.

   -z Comprime con **gzip** el paquete generado.

   [root@localhost sbin]# tar xvfz zebra-0.94.tar.gz
   zebra-0.94/
   zebra-0.94/vtysh/

```
zebra-0.94/vtysh/Makefile.in
zebra-0.94/vtysh/Makefile.am
……
……
……
zebra-0.94/init/redhat/zebra.init
zebra-0.94/init/redhat/zebra.logrotate
zebra-0.94/init/redhat/zebra.spec.in
[root@localhost sbin]#
```

4. Nos colocamos en el directorio

   ```
   cd zebra-0.94
   ```

5. Digitamos ./configure

   prepara el *Makefile*, configura las opciones de compilación y verifica si el sistema posee las bibliotecas de desarrollo necesarias para la compilación.

   ```
   [root@localhost sbin]# cd zebra-0.94
   [root@localhost zebra-0.94]# ./configure
   checking for a BSD-compatible install... /usr/bin/install -c
   checking whether build environment is sane... yes
   checking for gawk... gawk
   checking whether make sets $(MAKE)... yes
   ……
   ……
   ……
   config.status: config.h is unchanged
   config.status: executing depfiles commands

   zebra configuration
   -------------------
   zebra version          : 0.94
   host operationg system  : linux-gnu
   source code location    : .
   compiler               : gcc
   compiler flags         : -g -O2 -Wall
   ```

```
directory for pid files : /var/run

[root@localhost zebra-0.94]#
```

6. Digitamos

```
make      realiza la compilación del código fuente
[root@localhost zebra-0.94]# make
make  all-recursive
make[1]: Entering directory `/usr/local/sbin/zebra-0.94'
Making all in lib
……
……
……
make[2]: Leaving directory `/usr/local/sbin/zebra-0.94'
make[1]: Leaving directory `/usr/local/sbin/zebra-0.94'
[root@localhost zebra-0.94]#
```

7. Digitamos

```
su
```

8. Digitamos

make install instalación del los binarios y módulos compilados en los lugares correctos

```
[root@localhost zebra-0.94]# su
[root@localhost zebra-0.94]# make install
Making install in lib
make[1]: Entering directory `/usr/local/sbin/zebra-0.94/lib'
make[2]: Entering directory `/usr/local/sbin/zebra-0.94/lib'
……
……
make[2]: Leaving directory `/usr/local/sbin/zebra-0.94'
make[1]: Leaving directory `/usr/local/sbin/zebra-0.94'
[root@localhost zebra-0.94]#
```

9. Encontramos los directorios generados por zebra:

```
[root@localhost zebra-0.94]# find / -name zebra
/usr/local/sbin/zebra-0.94/zebra
/usr/local/sbin/zebra-0.94/zebra/zebra
```

```
/usr/local/sbin/zebra
[root@localhost zebra-0.94]#
```

10. Verificamos la instalación de archivos de configuración de los demonios en el directorio: /usr/local/etc. Realizamos un listado de los archivos

```
[root@localhost zebra-0.94]# cd /usr/local/etc/
[root@localhost etc]# ls -l
total 44
-rw-------  1 root root  572 Feb 23 10:32 bgpd.conf.sample
-rw-------  1 root root 2801 Feb 23 10:32 bgpd.conf.sample2
-rw-------  1 root root 1170 Feb 23 10:32 ospf6d.conf.sample
-rw-------  1 root root  180 Feb 23 10:32 ospfd.conf.sample
-rw-------  1 root root  412 Feb 23 10:32 ripd.conf.sample
-rw-------  1 root root  393 Feb 22 17:36 ripd.conf.sav
-rw-------  1 root root  396 Feb 23 10:32 ripngd.conf.sample
-rw-------  1 root root  375 Feb 23 10:32 zebra.conf.sample
-rw-------  1 root root  369 Feb 22 15:52 zebra.conf.sav
[root@localhost etc]#
```

11. Renombramos los archivos los archivos con extensión “.sample” a extensión “.conf ”

```
[root@ localhost etc]# mv zebra.conf.sample  zebra.conf
[root@ localhost etc]# mv ospf6d.conf.sample  ospf6d.conf
[root@ localhost etc]# mv ripngd.conf.sample  ripngd.conf
[root@ localhost etc]# mv ripd.conf.sample  ripd.conf
[root@ localhost etc]# mv ospf.conf.sample ospf.conf
```

12. Se debe de crear una carpeta con el nombre “zebra” dentro del directorio << var/log >>. En cuya carpeta se crearan los archivos .log(errores) para cada demonio a utilizar.

```
[root@ localhost ]#        cd  /var/log
[root@ localhost log]#   mkdir zebra
[root@ localhost log]#   cd zebra
```

13. Creamos los archivos .log para el demonio ripd.

```
[root@ localhost  zebra]# vi ripd.log
```

14. Se asigna los permisos correspondientes al archivo.

[root@ localhost zebra]# chmod 777 ripd.log

15. Abrimos el archivo ripd.conf (/usr/local/etc) en modo edición.

[root@ localhost /]# cd /usr/local/etc

[root@ localhost zebra]# vi ripd.conf

```
!
! Zebra configuration saved from vty
!   2006/02/22 17:38:07
!
hostname ripd
password zebra
log file /var/log/zebra/ripd.log
log stdout
!
```

Se descomenta la línea “!log file” y se coloca la dirección en la cual este el .log

16. Guardamos los cambios realizados para este archivo de configuración.

17. Podemos ver los servicios que tiene el S.O con la instalación del nuevo software Zebra. En el la ubicación /etc, en el cual se encuentran ya insertados los servicios, el numero del puerto que utiliza y su tipo ya sea UDP o TCP. En caso de no encontrarse los mismo se los debe colocar.

[root@price etc]# vi services

Demonios instalados por zebra

```
#Local Services
zebrasrv 2600/tcp #zebra service
zebra      2601/tcp #zebra vty
ripd       2602/tcp #ripd vty
ripngd   2603/tcp #ripngd vty
ospfd      2604/tcp #ospfd vty
ospf6d   2606/tcp #ospf6d vty
```

Configurando ip en routers(pcs) ypcs

Al tener la topología de nuestra red empezamos por asignar las direcciones IP a cada una de las interfases ethernet.

PCA (S.O Windows XP)

Asignamos a este equipo la dirección IP, su respectiva mascara y puerta de enlace. Una vez asignada la dirección se debe deshabilitar el firewall del sistema para que puedan comunicarse.

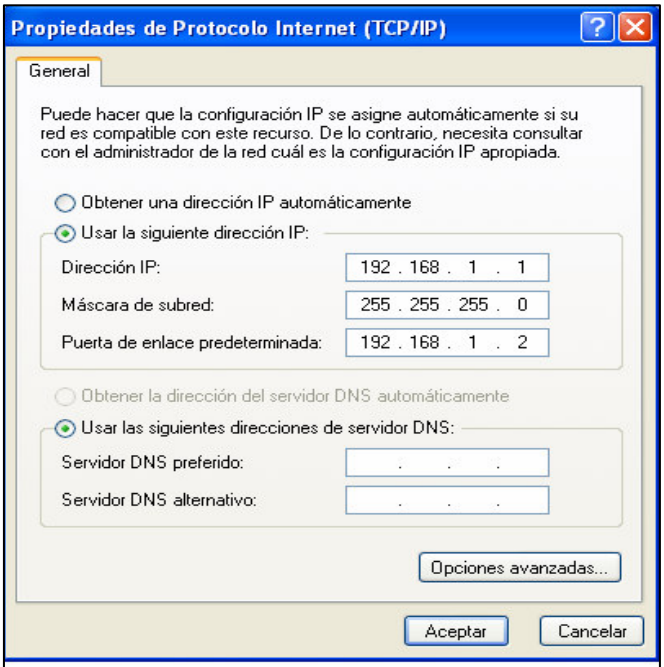

Gráfico 4. Configuracion Ip address PCA

Router 1 (pc) (s.o. centos)

En este caso se debe definir 2 direcciones IP con sus respectivas mascaras ya que tiene 2 tarjetas ethernet. Luego se deberá activar cada ethernet para que actualice los cambios

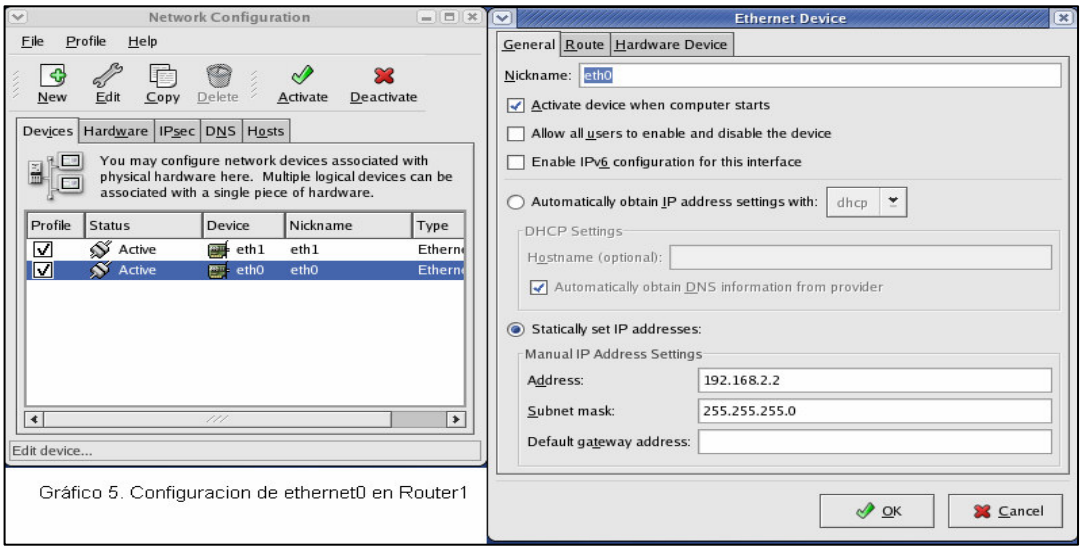

Gráfico 5. Configuracion de ethernet0 en Router1

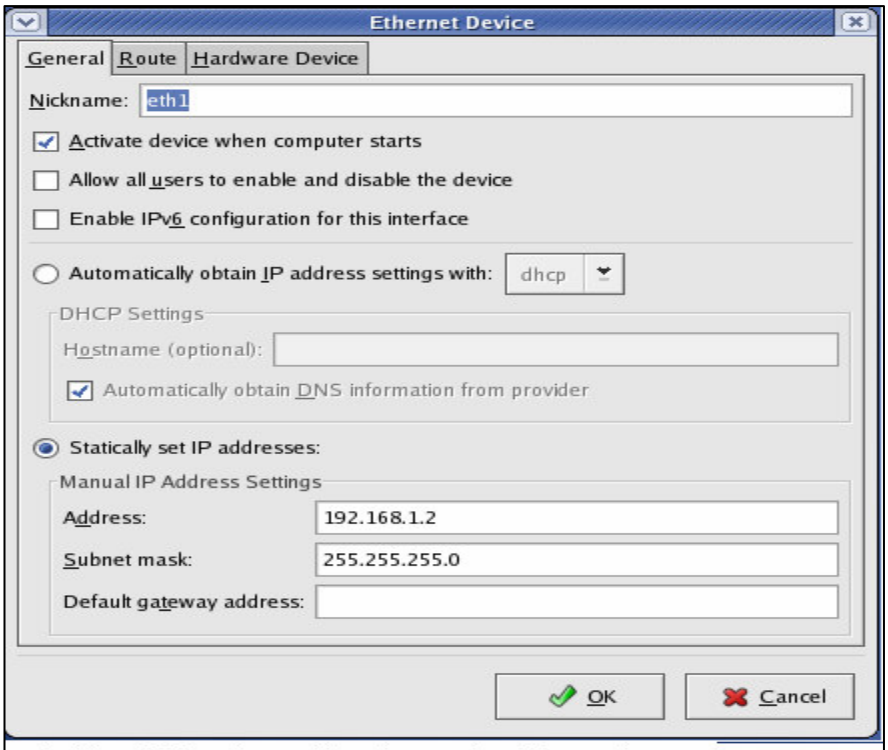

Gráfico 6. Configuración ethernet1 en Router1

PCB (S.O Windows XP)

Asignamos a este equipo la dirección IP, la mascara y puerta de enlace. Una vez asignada la dirección se debe deshabilitar el firewall del sistema para que puedan comunicarse.

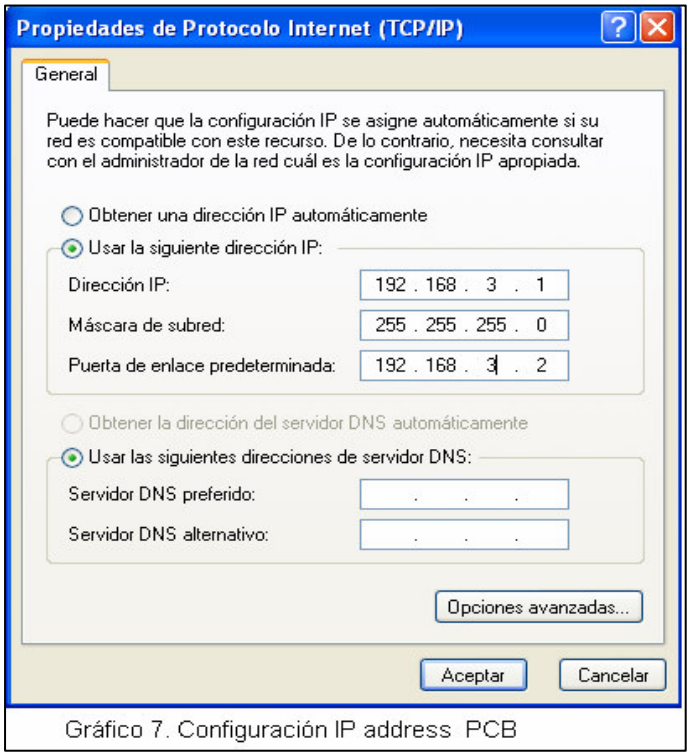

Gráfico 7. Configuración IP address PCB

ROUTER 2 (PC) (S.O. CENTOS)

En este caso se debe definir 2 direcciones IP con sus respectivas mascaras ya que tiene 2 tarjetas ethernet.

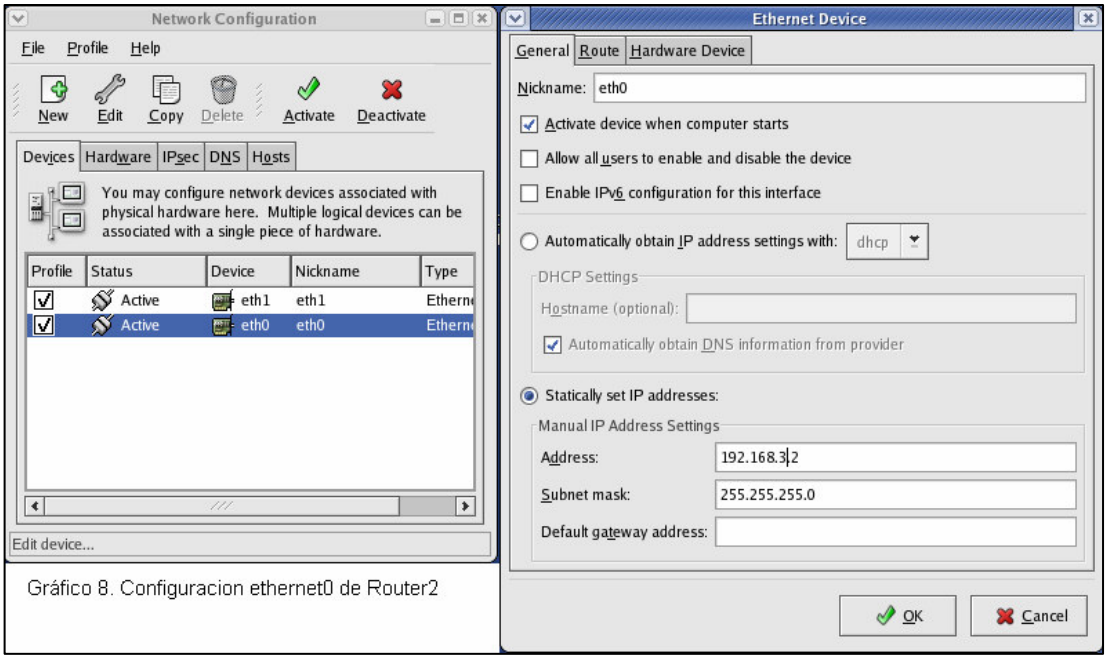

Gráfico 8. Configuracion ethernet0 de Router2

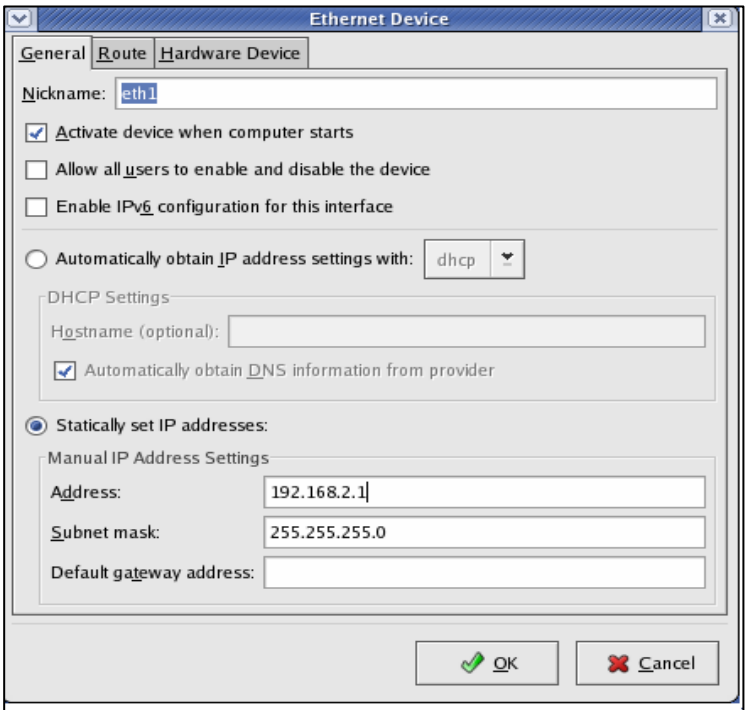

Gráfico 9. Configuración ehternet1 de Router2

TESTING: Antes de configurar zebra debemos comprobar la conectividad de cada PC a sus respectivas ethernet asociadas mediante el comando PING. Ejemplo:

De PC A hacia Eth1 del Router 1

Ping 192.168.1.2

De PC B hacia Eth0 del Router 2

Ping 192.168.3.2

Router1 (PC): trabajando en zebra

Levantando servicios

Una vez configurado todos los archivos y definidas las IP en cada una de las interfases se debe levantar los servicios.

Levantamos servicios zebra, a través de una Terminal (antes de cualquier otro demonio debe ser invocado zebra) de la siguiente manera:

[root@ localhost /]# zebra –d

-d invoca el demonio

Al levantar el servicio podemos ingresar mediante telnet a este demonio para comenzar la configuración de la red es decir indicar a zebra las IPS de sus interfaces.

```
[root@localhost]# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.94).
Copyright 1996-2002 Kunihiro Ishiguro.

User Access Verification

Password:
Router>
```

Introducimos la contraseña que por defecto es “zebra” o la puede buscar en los archivos de configuración, (/usr/local/etc/zebra.conf) esto si es que no esta configurada para ser encriptada.

Nos pasamos al modo enable:

```
Router> enable
Password:
Router#
```

Ingresamos a la configuracion de la Terminal

```
Router#configure terminal
Router(config)#
```

Para evitar confusiones procedemos a renombrar al router como Router1

```
Router(config)#hostname Router1
Router1(config)#
```

Entramos a cada una de las interfaces para asignar su dirección

```
Router1(config)#interface eth0
Router1(config-if)#ip address 192.168.2.2/24
Router1(config-if)#quit
Router1(config-if)#interface eth1
Router1(config-if)#ip address 192.168.1.2/24
Router1(config)#write f
Configuration saved to /usr/local/etc/zebra.conf
```

Después de las configuraciones realizadas los cambios en es archivo de configuración son los siguientes:

ZEBRA.CONF

```
!
! Zebra configuration saved from vty
!   2006/02/22 22:50:57
!
hostname Router
password zebra
enable password zebra
!
```

```
interface lo
!
interface eth0
 ip address 192.168.2.2/24
 ipv6 nd suppress-ra
!
interface eth1
 ip address 192.168.1.2/24
 ipv6 nd suppress-ra
!
interface sit0
 ipv6 nd suppress-ra
!
!
line vty
```

Router2(PC): trabajando en zebra

Levantando servicios

Levantamos servicios zebra, a través de una terminal.

[root@ localhost /]# zebra –d

-d invoca el demonio

Ingresar mediante telnet a este demonio para comenzar la configuración de la red.

```
[root@localhost]# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.94).
Copyright 1996-2002 Kunihiro Ishiguro.


User Access Verification
```

Password:
Router>

Nos pasamos al modo enable:

Router> enable
Password:
Router#

Ingresamos a la configuracion de la Terminal

Router#configure terminal
Router(config)#

Para evitar confusiones procedemos a renombrar al router como Router1

Router(config)#hostname Router2
Router2(config)#

Entramos a cada una de las interfaces para asignar su dirección

Router2(config)#interface eth0
Router2(config-if)#ip address 192.168.3.2/24
Router2(config-if)#quit
Router2(config-if)#interface eth1
Router2(config-if)#ip address 192.168.2.1/24
Router2(config)#write f
Configuration saved to /usr/local/etc/zebra.conf

Después de las configuraciones realizadas los cambios en es archivo de configuración son los siguientes:

ZEBRA.CONF

```
!
! Zebra configuration saved from vty
!   2006/02/22 22:50:57
!
hostname Router
password zebra
enable password zebra
!
interface lo
!
```

```
interface eth0
 ip address 192.168.3.2/24
 ipv6 nd suppress-ra
!
interface eth1
 ip address 192.168.2.1/24
 ipv6 nd suppress-ra
!
interface sit0
 ipv6 nd suppress-ra
!
!
line vty
```

ROUTER1(PC): TRABAJANDO EN ripv2

Levantamos servicios ripd, a través de una Terminal (antes de cualquier otro demonio debe ser invocado zebra) de la siguiente manera:

```
[root@ localhost /]# ripd –d
        -d invoca el demonio
```

Al levantar el servicio podemos ingresar mediante telnet al demonio para configurar las redes bajo protocolo RIPv2

```
[root@localhost]# telnet localhost ripd
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.94).
Copyright 1996-2002 Kunihiro Ishiguro.


User Access Verification
```

Password:
ripd>

Introducimos la contraseña que por defecto es “zebra” o la puede buscar en los archivos de configuración, (/usr/local/etc/ripd.conf) esto si es que no esta configurada para ser encriptada.

Nos pasamos al modo enable:

ripd> enable
ripd#

Ingresamos a la configuración de la Terminal

ripd #configure terminal
ripd (config)#

Habilitamos el proceso de encaminamiento

ripd (config)#router rip
ripd (config-router)#

Habilitamos el encaminamiento sobre cada red IP en la que queremos que actúe RIP

ripd (config-router)#network 192.168.1.0/24
ripd (config-router)#network 192.168.2.0/24

Definimos un router vecino con el que se intercambiara información de enrutamiento.

ripd (config-router)#neighbor 192.168.2.1

Guardamos los cambios en el archivo de configuración

ripd (config-router)#write f
Configuration saved to /usr/local/etc/ripd.conf

Después de las configuraciones que hemos realizado el archivo ripd.conf quedara de la siguiente manera:

RIPD.CONF

```
!
! Zebra configuration saved from vty
!   2006/02/22 23:00:52
!
hostname ripd
```

```
password zebra
log file /var/log/zebra/ripd.log
log stdout
!
interface lo
!
interface eth0
!
interface eth1
!
interface sit0
!
router rip
 network 192.168.1.0/24
 network 192.168.2.0/24
 neighbor 192.168.2.1
!
line vty
```

ROUTER2(PC): TRABAJANDO EN RIPv2

Levantamos servicios ripd, a través de una Terminal (antes de cualquier otro demonio debe ser invocado zebra) de la siguiente manera:

```
[root@ localhost /]# ripd –d
        -d invoca el demonio
```

Al levantar el servicio podemos ingresar mediante telnet al demonio para configurar las redes bajo protocolo RIPv2

```
[root@localhost]# telnet localhost ripd
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

Hello, this is zebra (version 0.94).
Copyright 1996-2002 Kunihiro Ishiguro.
```

User Access Verification

Password:
ripd>

Nos pasamos al modo enable:

ripd> enable
ripd#

Ingresamos a la configuración de la Terminal

ripd #configure terminal
ripd (config)#

Habilitamos el proceso de encaminamiento

ripd (config)#router rip
ripd (config-router)#

Habilitamos el encaminamiento sobre cada red IP en la que queremos que actúe RIP

ripd (config-router)#network 192.168.3.0/24
ripd (config-router)#network 192.168.2.0/24

Definimos un router vecino con el que se intercambiara información de enrutamiento

ripd (config-router)#neighbor 192.168.3.2

Guardamos los cambios en el archivo de configuración

ripd (config-router)#write f
Configuration saved to /usr/local/etc/ripd.conf

Después de las configuraciones que hemos realizado el archivo ripd.conf quedara de la siguiente manera:

RIPD.CONF

```
!
! Zebra configuration saved from vty
!   2006/02/22 23:00:52
!
hostname ripd
```

```
password zebra
log file /var/log/zebra/ripd.log
log stdout
!
interface lo
!
interface eth0
!
interface eth1
!
interface sit0
!
router rip
 network 192.168.2.0/24
 network 192.168.3.0/24
 neighbor 192.168.2.2
!
line vty
```

Una vez terminadas las respectivas configuraciones se procede a levantar los servicios tanto de zebra como rip ya que ellos son los que utilizaremos en esta practica.

Configurando la descripción para cada interface

```
Router1# configure terminal
Router1(config)# interface eth1
Router1(config-if)# description LAN Ecuador
Router1(config)# interface eth0
Router1(config-if)# description Backbone
Router1(config-if)# exit
Router1(config)# exit
```

4.5 Pruebas con Ethereal

Una vez realizadas las configuraciones correspondientes en cada PC de nuestra Topología, procedemos a ejecutar el Software Ethereal, este nos permitirá realizar la captura de tráfico sobre nuestras redes configuradas para trabajar con protocolos RIP v2 y entender mejor su funcionamiento.

Ejecutamos el Software Ethereal en cada PC es decir para PCA, PCB, ROUTER1 (PC), ROUTER2 (PC).

Aproximadamente esperamos 30 segundos para la captura de paquetes, luego paramos la captura en cada P.D., y así proceder a analizar los paquetes generados por el protocolo RIP.

Resultado de las capturas

PCA

Una de las mejoras de Ripv2 es el envió de actualizaciones de sus tablas mediante la dirección de multicast 224.0.0.9

En esta captura podemos ver la estructura de un paquete generado por el demonio Ripd, a la PcA le ha llegado un paquete con dirección multicast 224.0.0.9 enviado por el Router1 con dirección 192.168.1.2, este paquete es de tipo respuesta el cual contiene la información de tablas de enrutamiento del Router1, haciéndole conocer las redes de tipo Rip que contiene y su distancia para acceder a la misma, el campo (IP Address) contiene la dirección de red almacenada por el Router1 y el campo (Metric) Contiene el numero de saltos que hay que realizar para acceder a la misma.

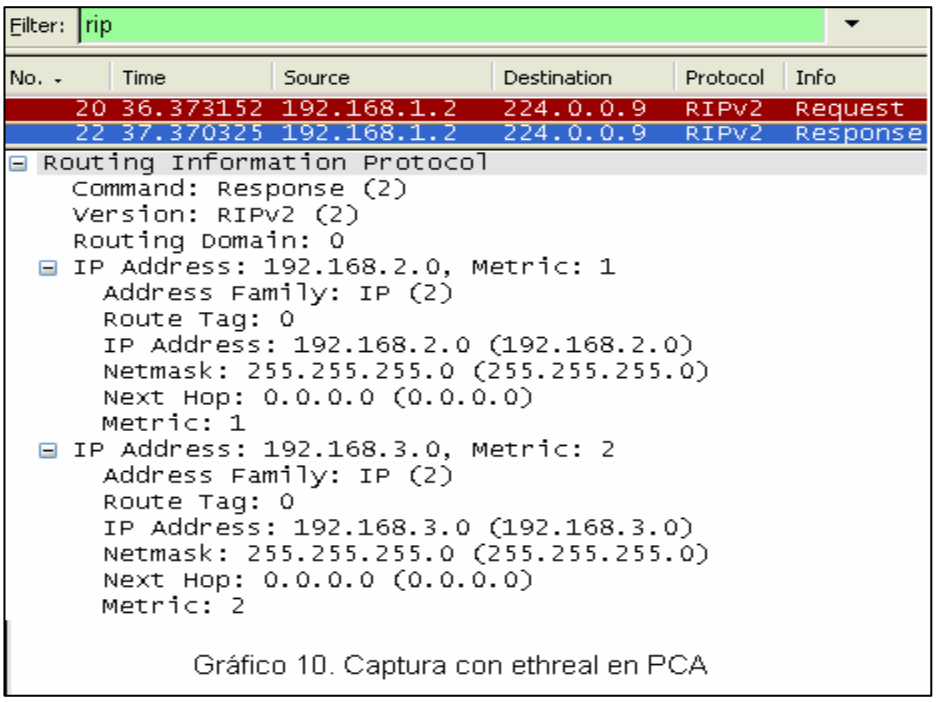

Gráfico 10. Captura con ethreal en PCA

ROUTER1 (PC)

Observando el paquete de respuesta numero 9 enviado por el router2 (PC) a el router1, contiene información de las redes rip que conoce este y su respectiva distancia. En el capo IP Address se da a conocer la siguiente red:

192.168.3.0 Metric: 1

Es decir le esta informando al router1 que el conoce esta red y sabe que distancia se necesita para acceder a la misma.

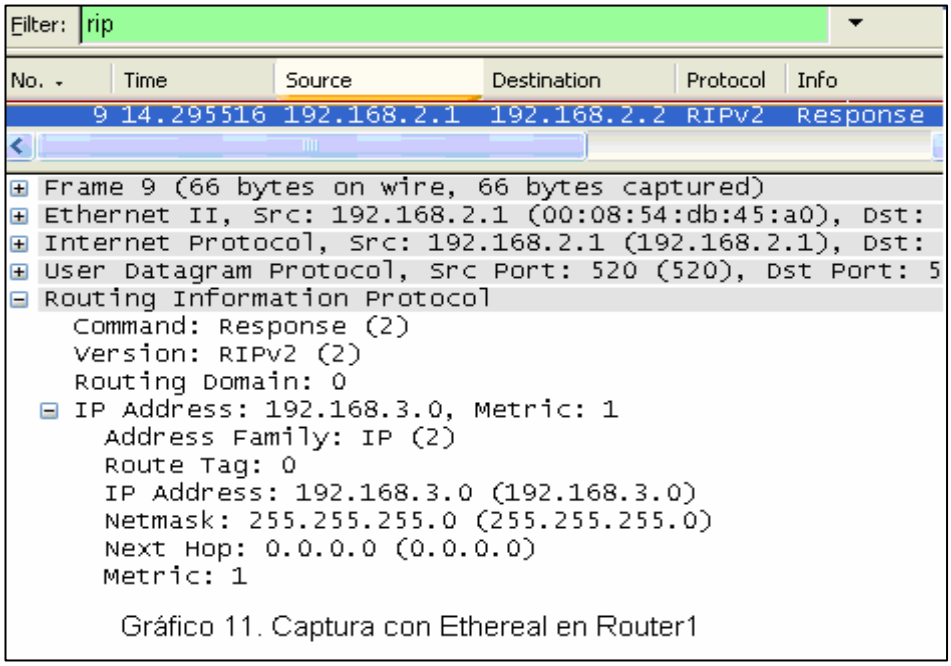

Gráfico 11. Captura con Ethereal en Router1

Ahora entramos en el demonio Ripd y digitamos el siguiente comando.

```
ripd# show ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP
       (n) - normal, (s) - static, (d) - default, (r) - redistribute,
       (i) - interface

     Network           Next Hop         Metric From            Time
C(i) 192.168.1.0/24    0.0.0.0               1 self
C(i) 192.168.2.0/24    0.0.0.0               1 self
R(n) 192.168.3.0/24    192.168.2.1           2 192.168.2.1     00:04
```

Gráfico 12. Tabla de routing Router1

Este comando nos visualizará la tabla de rutas contenidas para este router1, se presentan 2 redes conectadas directamente (C) y una red rip aprendida por este la cual es 192.168.3.0 y la puerta por la cual se accederá a la misma, la cual es la 192.168.2.1

Rip asigna una métrica de 1 para aquellas redes que están directamente conectadas.

La cabecera Time de la tabla de rutas rip especifica el tiempo transcurrido desde que se ha recibido la última actualización de esa ruta, cabe recalcar que RIP utiliza tres tiempos importantes, el tiempo de actualización que se estable en 30 segundos y el tiempo de desactivación que se establece en 180 segundos (3 minutos) y el tiempo de borrado se establece en 300 segundos (5 minutos).

ROUTER2 (PC)

Este paquete fue enviado por el router1 mediante *broadcast* y recibido en esta maquina, el tipo de paquete es *response* es decir información de rutas.

El rouer1 le informa a router2 que conoce la red 192.168.1.0 y su distancia es de 1.

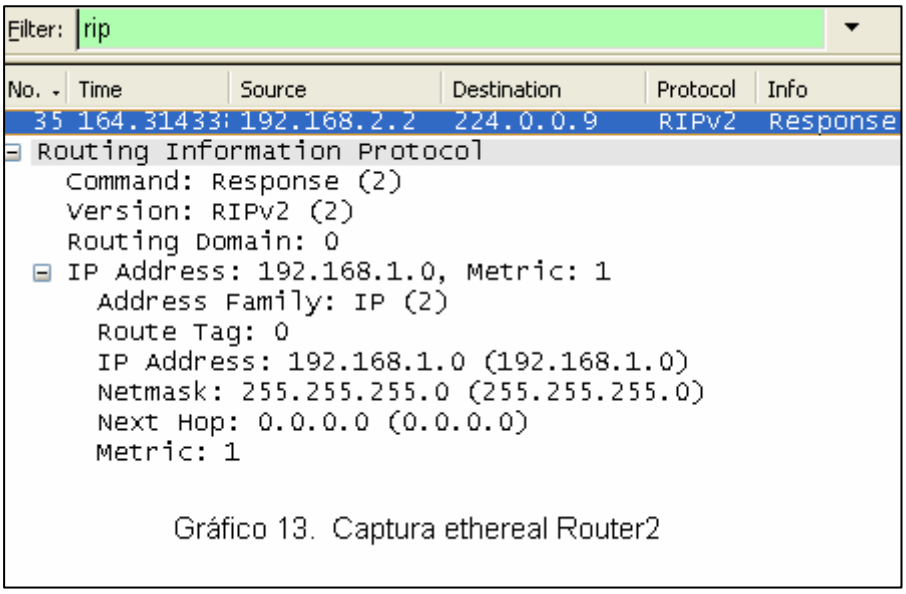

Gráfico 13. Captura ethereal Router2

Observando el campo next hop vemos que posee una dirección 0.0.0.0 con mascara 0.0.0.0 esto nos quiere decir que la puerta de acceso a esa red es por la dirección de la interfaz la cual esta especificada la cabecera origen (*source)* de este paquete.

PCB

Uno de los tantos paquetes multicast es escuchado en PCA el cual es originado por el router2 dando a conocer sus rutas y el estado de las mismas que para este caso se encuentran en óptimas condiciones.

Las redes que se dan a conocer son las siguientes:

192.168.1.0 con métrica 2

192.168.2.0 con métrica 1

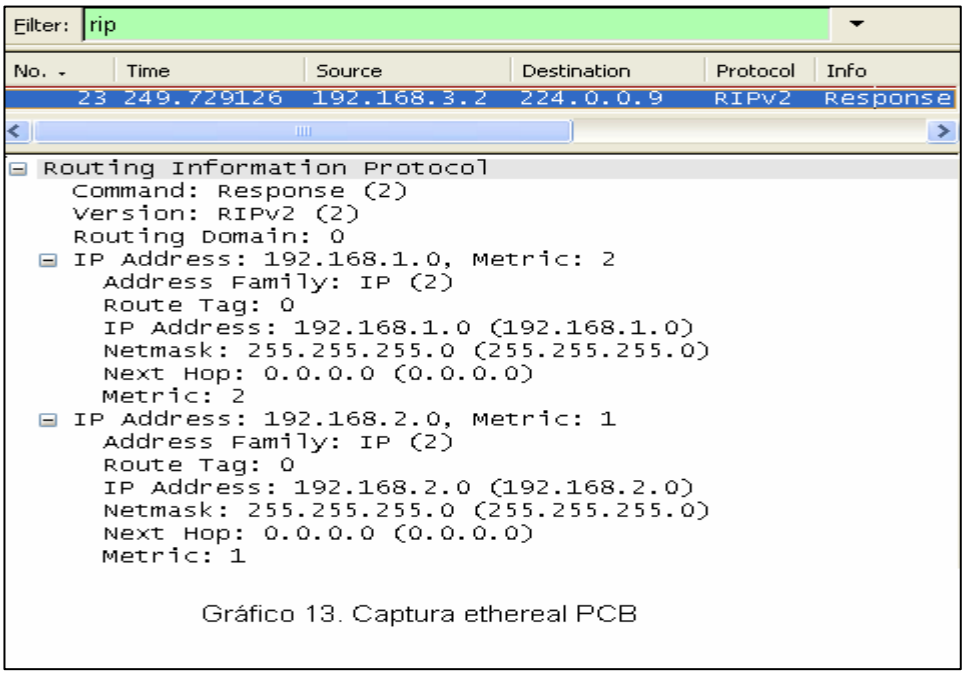

Gráfico 13. Captura ethereal PCB

4.6 Reacciones ante desconexión de red

Analizaremos el potencial de rip ante un suceso como es la desconexión de unos de los cables en la red 192.168.2.0.

ROUTER1

Recordemos que tiempo de actualización se considera al tiempo máximo a transcurrir entre el envío de los mensajes de actualización de los vecinos, y el tiempo de desactivación se considera al tiempo máximo que puede esperar un router sin recibir actualizaciones de vecino, una vez pasado este tiempo, el vecino que no ha enviado la actualización se considera que ha caído y con lo cual el router no está activo en la red, se establece la métrica a valor 16, es decir destino inalcanzable y posteriormente el tiempo de borrado implica que una vez transcurrido ese tiempo todas las rutas de ese router supuestamente caído son eliminadas de la tabla de routing.

Este router hace conocer a todos sus vecinos por medio de paquetes multicast que la red 192.168.3.0 es inalcanzable por que no a obtenido mensajes de actualización de la misma y por ello se asignado un métrica 16.

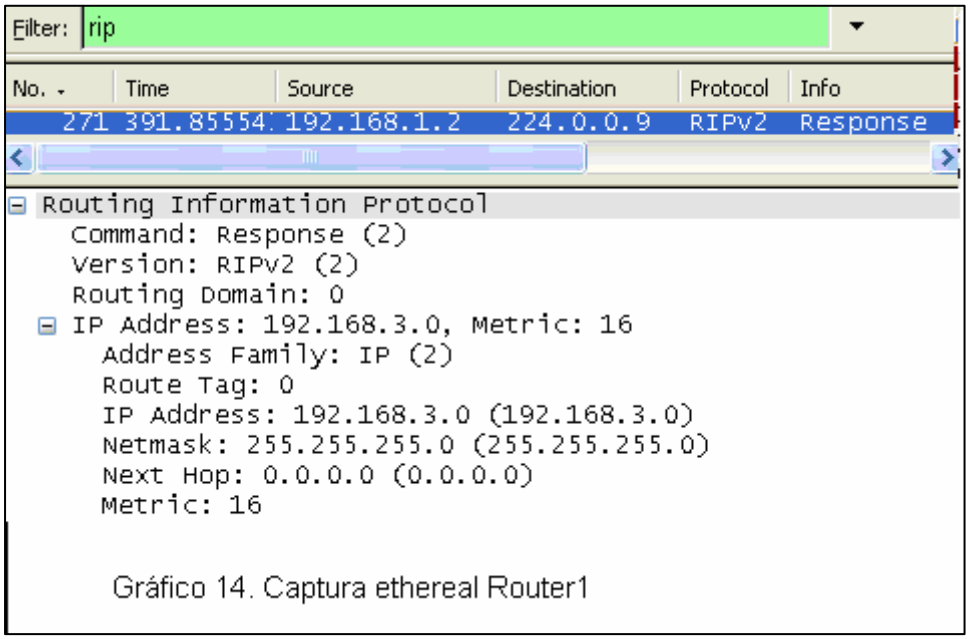

Gráfico 14. Captura ethereal Router1

Ejecutamos el siguiente comando. En el cual observamos que a la red 192.168.3.0 se le a asignado un métrica de 16 dando a conocer que esta caída.

```
ripd# show ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP
       (n) - normal, (s) - static, (d) - default, (r) - redistribute,
       (i) - interface

     Network           Next Hop         Metric From            Time
C(i) 192.168.1.0/24    0.0.0.0               1 self
C(i) 192.168.2.0/24    0.0.0.0               1 self
R(n) 192.168.3.0/24    192.168.2.1          16 192.168.2.1     01:41
ripd#
```

Grafico 15. Tabla de routing router1

ROUTER2

En este caso el router2 da a conocer como inaccesible la red 192.168.2.2 lo que es correcto.

```
ripd# show ip rip
Codes: R - RIP, C - connected, O - OSPF, B - BGP
       (n) - normal, (s) - static, (d) - default, (r) - redistribute,
       (i) - interface

     Network           Next Hop         Metric From            Time
R(n) 192.168.1.0/24    192.168.2.2          16 192.168.2.2     01:44
C(i) 192.168.2.0/24    0.0.0.0               1 self
C(i) 192.168.3.0/24    0.0.0.0               1 self
```

Gráfico 15. Tabla de routing Router2

4.7 Seguridad

Los aspectos de seguridad que posee zebra son:

- Cambio de passwords para la interfaz VTY.
- Cambio de passwords para la interfaz VTY para modo enable.
- Encripcion de password

Cambio de passwords para la interfaz VTY.

```
Router1# configure terminal
Router1(config)# password cisco
Router1# write terminal
```

Cambio de passwords para la interfaz VTY para modo enable

```
Router1# configure terminal
Router1(config)# enable password cisco
Router1# write terminal
```

Encripcion de password

```
Router1(config)# service password-encryption
Router1# write terminal
```

Tiempo de la conexión de la interfaz VTY

Establecemos el tiempo de conexión para la interfaz.

Al especificar un argumento, es utilizado como el tiempo máximo en minutos.

El valor por defecto es de 10 minutos.

```
Router1(config)# line vty
Router1(config-line)# exec-timeout 120
Router1(config)# write terminal
```

4.8 Filtros

Una forma de bloquear paquetes que vengan de alguna dirección no deseada es a través del comando access-list.

Aplicaremos el siguiente filtro para el router1.

ROUTER1(PC)

```
ripd(config)# access-list private permit 10.0.0.0/8
ripd(config)# access-list private deny any
```

Hemos configurado al router1 para que solo acepte paquetes de la red 10.0.0.0/8, en caso de pertenecer a otra red serán rechazados.

Asociamos la lista de acceso a la interfaz eth0.

```
ripd(config)# router rip
ripd(config-router)# distribute-list private in eth0
ripd(config-router)# write terminal
```

Al archivo de configuración de rip se ha añadido las siguientes líneas.

```
Ripd.conf
!
router rip
 distribute-list private in eth0

!
access-list private permit 10 10.0.0.0/8
access-list private deny any
!
```

El filtro ha sido puesto en marcha bloqueando cualquier paquete que no pertenezca a la red 10.0.0.0/8

4.9 Conclusión

Zebra tiene bastantes más características que cualquier software de encaminamiento y ostenta una interesante e interactiva línea de interfaz de comandos al estilo de Cisco. Cada protocolo (RIP, OSPF, etc.) tiene su propia configuración, y se pueden ejecutar múltiples protocolos a la vez (aunque podría originar confusiones/problemas).

Entre las configuraciones más importante de zebra resaltan:

- Cambiar passwords por omisión
- Encriptar Passwords
- Cambiar nombre del host
- Identificar las interfaces y añadir la descripción
- Configurar rutas estáticas.

A través de la realización de esta práctica se pudo probar la validez del paquete Zebra al lograr interconexión entre diferentes redes con mascaras de longitud variable, comprobando su comunicación mediante el comando ping.

Zebra esencia del fenómeno de la transmisión de información a través de redes de datos logrando un routing con un equilibrio entre la simplicidad y la exactitud con un entorno Amigable e Intuitivo para el usuario.

5. Conclusiones Generales

Software libre zebra es una nueva y buena opción para simular el funcionamiento de routers CISCO, ya que la plataforma que utiliza es fácil de entender y no requiere de mucho conocimiento para su configuración, la consola provee ayuda en todo momento una vez ingresado al demonio que se desee.

Este software es flexible al momento del uso de protocolos así que tenemos 4 de ellos inmersos en 7 demonios respectivamente, de ahí el hecho que se tiene la libertad de configurar según el conocimiento del administrador.

Zebra posee un nivel de seguridad aceptable tanto Terminal y de configuración las que consisten en cambios de claves de acceso para evitar que personas no autorizadas manipulen rutas de la red.

Al ser un software que se basa sobre un sistema operativo en este caso Linux nos debemos dar cuenta que el administrador no toma atención solo a las funciones de ruteo sino también a las del sistema operativo ya que es vital el buen funcionamiento de este para garantizar la configuración de la red.

Simulando las PCS como routers presenta un limitante de interfaces, por lo que el hecho de ser un software libre no quiere decir que no represente gastos algunos. Aunque no es comparable al monto de adquisición de un router.

6. Recomendaciones

Este software puede ser muy bien utilizado para interconectar redes pequeñas sobre las cuales no resulte crítica la caída de la red, y también es muy valido a nivel académico para hacer prueba.

Zebra al correr sobre sistema operativo Linux se requiere conocimiento del mismo para de esta manera poder estar seguros de que las configuraciones que realizamos sobre los demonios corran exitosamente. Esto es de gran ayuda ya que si se detecta problemas saber cual es el origen ya sea zebra o el sistema operativo.

Al existir algunas versiones del paquete en el Word Wide Web es importante conocer si la arquitectura del sistema operativo es compatible a la del paquete. De no ser así los directorios de configuración no se graban ni se crean en los lugares correctos, lo cual al momento de configuración de los demonios pueden presentarse confusiones, y pensar que los comandos de configuración están errados.

7. Referencias

7.1 Glosario

TCP/IP: (*Transmission Control Protocol/Internet Protocol*) Protocolo de control de transmisiones/protocolo Internet. Protocolos de comunicaciones desarrollados bajo contrato del Departamento de Defensa de los Estados Unidos para intercomunicar sistemas diferentes. Es un estándar UNIX de hecho, pero está respaldado por sistemas operativos de micro a mainframe. Es utilizado por muchas corporaciones y casi todas las universidades y entidades federales.

Telnet: Protocolo de emulación de terminales usado con frecuencia en Internet. Permite al usuario tener acceso a un programa y ejecutarlo desde un terminal o computador remote. Fue desarrollado originalmente por ARPAnet y es parte del protocolo de comunicación TCP/IP.

Proxy: Proxy Server, Servidor Proxy. También se le conoce como Proxy; es una aplicación que opera como barrera de fuego al interrumpir la conexión entre el emisor y el receptor. Toda la información de entrada es enviada a un puerto diferente, cerrando el camino directo entre dos redes, previniendo así que un intruso (hacker) pueda obtener direcciones internas y detalles acerca de una red privada.

RFC: (Acrónimo de *Request for Comments*, 'Solicitud de comentarios'). Los RFC discuten todos los aspectos de las comunicaciones informáticas, pero atendiendo especialmente a estándares y protocolos. Esta larga serie se inicia en 1969, con el RFC 1, "*Host software*".

Protocol: *Communications Protocol*, Protocolo de comunicaciones. Estándares de software o de hardware que controlan las transmisiones entre dos estaciones

Demonio: Es como un programa que permanece en segundo plano ejecutándose continuamente para dar algún tipo de servicio.

Router Reflector: Refleja todas las rutas aprendidas por un IBGP hacia sus clientes

y de las rutas aprendidas hacia sus IGP (no clientes), esto reduce el número de vecinos en toda la red reflejándose en menos procesamiento para los equipos enrutadores y brindando mejores tiempos de convergencia además de facilitar la administración de la red

Router Server: Ruteador que activa la funcionalidad BGP de Router Reflector Intercambia la Información de Ruteo con ruteadores de proveedores de servicios. No envía paquetes únicamente maneja la lógica de paquetes. Se usan generalmente en puntos de intercambio

Routing Protocol: Protocolo de enrutamiento. Fórmula usada por los enrutadores para determinar la ruta apropiada para el envío de datos. También especifica la manera de reportar los cambios y cómo compartir información con otros enrutadores en la red. Permite a la red ajustarse dinámicamente a las condiciones cambiantes, de otro modo todas las decisiones de enrutamiento deberán predeterminarse y permanecer estáticas.

IP unicast: La dirección corresponde a un solo receptor y será éste el único que procese los datagramas IP con ese destino (conexión uno-a-uno).

IP broadcast: La dirección corresponde a todos los equipos conectados en un mismo tramo de red local y el datagrama IP es procesado por todos ellos (conexión uno-a-todos dentro de la misma subred).

IP multicast: La dirección corresponde a un grupo de equipos, y sólo estos procesarán los datagramas IP con ese destino (conexión uno-a-muchos, o uno-a-varios).

Access list: tienen acciones que permiten o deniegan la conexión.

7.2 Bibliografía

www.128.ibm.com/developerworks/library/l-emu/
Cimafranca, Dominique
Young, Rex

www.tecs.ecu.edu/tsys/labs/reports/RIPv2%20with%20Quagga%20for%20Adios%204%20with%20Appendix.pdf
DeTiberus, Jason

http://manticore.2y.net/doc/zebra/ripd.html
Ishiguro, Kunihiro

www.tlm.unavarra.es/~daniel/docencia/lpr/lpr03_04/practicas/practica7.html
Morató, Daniel

www.wl0.org/~sjmudd/wireless/network-structure/html/x284.html
Mudd, Simon
Moncada, Ángel
Béjar, Joaquín

www.redes-linux.com/manuales.php?catId=Routing
Sevilla, Víctor

http://iie.fing.edu.uy/ense/asign/redatos/laboratorio/064-Quagga.pdf
Valdés, Álvaro

www.pointless.net/~jasper/zebra-html/zebra_36.html
Wallace, Jasper

www.quagga.net

www.zebra.org