UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE SERVICIO DHCP DE ALTA DISPONIBILIDAD CON SOFTWARE LIBRE

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN

PRESENTAN:

GALLARDO HERNÁNDEZ LUIS ENRIQUE

MENDOZA VARELA JEANETTE ROCÍO

DIRECTORA DE TESIS:

M.C. MA. JAQUELINA LÓPEZ BARRIENTOS

CIUDAD UNIVERSITARIA **2010**

A los que Siempre me apoyaron:

Gracias a **DIOS**, por un logro más en mi vida.

Gracias **Mamá**: ¡Eres la mejor mamá del mundo¡, gracias por tu amor, tu ejemplo y tus buenos consejos, por apoyarme todo el tiempo, día y noche para aconsejarme y escucharme, por confiar en mí, por tu carácter siempre alegre y por ayudarme a lograr esta meta, que sin ti hubiera sido imposible, te quiero mucho.

Lo logré **Papá**: Gracias por todo, por tu ejemplo y tu apoyo incondicional para lograr mis sueños, por permitirme realizar mi sueño desde niño de ser Ingeniero, por tus desvelos y tu trabajo para darme lo mejor, por ser el héroe y el líder de nuestra familia, te quiero mucho. Gracias.

Hermanos, gracias por su cariño apoyo para lograr mis sueños, Lic. **Beto** por ser el mayor y poner el ejemplo de que querer es poder, **Vero** eres la consentida y sin duda se que lograras tus sueños y no seré el último en graduarme. Los quiero mucho.

Gracias **UNAM** por enseñarme que la universidad, somos todos, por abrirme la mente y el corazón al conocimiento universal, por hacerme un ciudadano con ética y valores, y por regalarme una de las mejores experiencias de mi vida.

A nuestra asesora oficial **M.C. Jaqui López B.**, por su tiempo y apoyo para realizar esta tesis, y a nuestro asesor y amigo **Ing. Álvaro Rojas Martínez**, por su tiempo, paciencia y conocimientos para realizar este proyecto.

Al área de red interna y especialmente al **Ing. Humberto Brito.**

A la **familia Gallardo** y **familia Hernández**, por su apoyo y cariño, a los que están y los que se adelantaron en el camino.

A **Rocío**, por apoyarme en las buenas y en las malas, aguantar mi carácter y por ser amiga y compañera y lograr esta meta con un verdadero trabajo en equipo.

A mis hermanos y amigos: **Gabo, Luis y Rafa**, el escuadrón de la muerte. A mis amigos: **Joss, Emmanuel, Piotr, Claudia, Odetette, Karla, Adriana, Israel, Magaliii, Omar rata, Daniel Remi, Payola, More, Mony**, por estar ahí, siempre.

Luis Enrique Gallardo Hernández

AGRADECIMIENTOS

*Gracias a **Dios** por permitirme llegar a este momento tan importante en mi vida.*

*A la **UNAM**, por ser una escuela de excelencia y darme la oportunidad de formarme en ella.*

*A mis **padres**, por siempre estar conmigo, por su cariño, comprensión y apoyo incondicional para cumplir una meta más en mi vida. Por darme la vida y ser lo que ahora soy, gracias a ellos.*

*A mis hermanos **Jessica y Raúl**, por su compañía y por estar siempre conmigo.*

*A la profesora **Jaquelina**, por ser una excelente maestra y por su dedicación brindada para la realización de esta tesis.*

A red interna, Alvaro, gracias por tu tiempo, dedicación y paciencia. Al Ing. Humberto Brito por creer en mí y por todo el apoyo que me brindo.

*A **Kike**, por su ayuda que me dio durante la carrera, por ser un gran amigo y compañero.*

*A **David**, por tu apoyo, comprensión y amor que me brindaste para lograr este proyecto, a pesar de las circunstancias.*

A toda mi familia que siempre ha estado conmigo, en especial a Cocoliso.

*A mis amigos, **Magalis, Payolas, More, Kika, Virus, Omar, Remi, Eva, Viri, Israelo, Rico** que me acompañaron durante toda la carrera y han estado conmigo siempre que los he necesitado.*

Grcias a todos por su cariño y apoyo.

Jeanette Rocío Mendoza Varela

Índice General

Relación de figuras y tablas

Capítulo 2

- FIGURAS

- TABLAS

Capítulo 3

- FIGURAS

- TABLAS

Capítulo 4

- FIGURAS

- TABLAS

Introducción

Actualmente el número de usuarios de las redes de datos se ha incrementado considerablemente debido a la reciente integración de servicios a las redes de datos, tal es el caso de la telefonía IP, teléfonos celulares, entre otros. Como consecuencia, los servicios de administración y asignación de direcciones IP deben ser más eficientes y fáciles de administrar. Para ayudar a las organizaciones en dicha tarea existen en el mercado soluciones *appliance* y de *software* que permiten gestionar de forma centralizada los servicios de administración de direcciones, en la mayoría de los casos las implementaciones comerciales son una buena opción, pero tienen una gran limitante para las organizaciones tanto gubernamentales como privadas, que es el costo del producto.

Debido al alto costo de los productos comerciales, así como de las licencias y mantenimientos, las organizaciones están buscando alternativas de bajo costo, que cubran los requerimientos establecidos. Una buena alternativa a las soluciones propietarias es el Software libre, ya que ofrece soluciones específicas para cada requerimiento a un bajo costo y de muy buena calidad.

Para realizar la administración de direcciones se utiliza el protocolo llamado DHCP por sus siglas en inglés *Dynamic Host Configuration Protocol* o Protocolo de configuración dinámica de host el cual permite gestionar las direcciones IP en las redes de datos, así como asignar automáticamente paramentaros de red a los clientes. El servicio DHCP es de gran utilidad en lugares donde el número de equipos es muy grande, ya que permite al administrador gestionar de forma centralizada el servicio.

Como se trata de un servicio muy importante para los usuarios de la red, debido a que si el servicio no es capaz de brindar los parámetros de configuración de red el cliente no será capaz de conectarse a la red, es necesario utilizar un esquema de servicio redundante o de alta disponibilidad que permita disponer del servicio en el momento que sea requerido por el usuario. La alta disponibilidad es poder tener la continuidad operacional del servicio, con un diseño que permita al sistema recuperarse en caso de alguna falla.

Dicho lo anterior el objetivo de este trabajo es:

Implementar un servicio de DHCP de alta disponibilidad para una institución financiera, empleando tecnológicas de software libre, para minimizar costos de licenciamiento originado por soluciones propietarias, además de brindar el mismo o un mejor servicio que el ofrecido por las soluciones propietarias. Se requiere de un servicio DHCP que sea totalmente compatible con los equipos actuales de la institución, por lo que se propone la implementación del servicio bajo una plataforma Linux, que ofrece un alto nivel de funcionalidad, estabilidad y seguridad.

Adicionalmente para cubrir totalmente los requerimientos que establece el proyecto se proponen diferentes esquemas de alta disponibilidad del servicio, para así compararlos y escoger la mejor opción para dar alta disponibilidad al servicio DHCP.

Este trabajo se divide en 4 capítulos, en el primero se habla de los requerimientos que establece el proyecto para la implementación de un servicio DHCP, así como los objetivos y alcances del proyecto.

En el capítulo 2 se habla de software libre, su definición, cuáles son sus ventajas, porque resulta útil para una implementación del servicio DHCP, entre otros. Además se muestra un análisis de los diferentes productos en el mercado y se hace una comparación entre dichos productos y los productos de software libre.

En el capítulo 3 se habla a detalle del protocolo DHCP, sus características, funcionamiento e implementación en un sistema basado en Software Libre, puntualmente del ISC DHCP, así como de los detalles configuración y la implementación de su interfaz gráfica basada en Webmin.

Y como aportación a este proyecto se creó un módulo de búsqueda de direcciones IP llamado QuiCho que fue desarrollado en lenguaje de programación Perl, el cual ayuda a simplificar la administración, su uso se muestra en el capítulo 3 y los detalles del código se encuentra en el apéndice III.

En el capítulo 4 se presentan propuesta para dotar de alta disponibilidad al servicio DHCP para diferentes casos de implementación, en la primera parte se propone una arquitectura basada en heartbeat y el proyecto Linux HA, la segunda es utilizando el

Failover protocol y finalmente utilizando Webmin Cluster, de acuerdo con los requerimientos definidos por la organización.

Capítulo 1. Requerimientos DHCP de alta disponibilidad con software libre

En este capítulo se definirán los requerimientos tanto funcionales como no funcionales que deben ser considerados por cualquier usuario interesado en implementar un sistema basado en software libre, en este caso en particular la implementación del servicio DHCP, de igual manera se darán a conocer los alcances del proyecto y los objetivos a seguir para lograr una correcta implementación.

1.1 Problemática actual

Actualmente el software y los appliance comerciales han invadido el mercado, sin embargo, los administradores de TI (tecnologías de información) deben tener muy en cuenta el costo de dichas soluciones. Los costos monetarios de dichas soluciones "propietarias" suelen ser muy altos y dado que son soluciones generales no siempre satisfacen las necesidades del usuario correctamente, es decir, hay soluciones que integran una serie de servicios que quizá no son del todo necesario para el usuario, sin embargo las adquiere como un "TODO" sin tener opción de seleccionar sólo los servicios deseados y las funcionalidades exactas a su medida. El servicio de DHCP del cual se hablará a lo largo de esta tesis ha sido comparado con soluciones de ciertas marcas comerciales (software propietario) que mencionaremos posteriormente en el Capítulo 2.

Si bien es cierto que dichas soluciones comerciales operan correctamente conforme a las políticas y procedimientos establecidos por los usuarios y brindan una administración simplificada a través de una interfaz gráfica amigable y sencilla, en la cual se pueden configurar parámetros básicos de operación del servicio así como la realización de consultas de estado del servicio, estado de asignación de una dirección, etc. Esto resulta útil en condiciones ideales de operación, pero en caso de requerir configuraciones más avanzadas del servicio, el software propietario no permite hacer modificaciones sobre el producto, lo que lo convierte en una "caja negra" para el administrador del servicio. Efectivamente se podría contar con contratos de mantenimiento y soporte de estas soluciones, sin embargo, esto incrementará aún más los costos contemplados, en el capítulo 1 se habla más a detalle de los costos reales.

Otro de los factores importantes a considerar es el tiempo de respuesta para dar solución a una falla en las soluciones comerciales, debido a que generalmente existe una ventana de tiempo soportada para la ausencia de los servicios, definida en los Análisis de Riesgos propios de las empresas, misma que deberá ajustarse al tiempo que el fabricante o vendedor de respuesta y resuelva la falla.

Cada vez está más en desuso la idea del cobro por actualizaciones de un sistema ya adquirido, y las empresas han colocado como parte de sus políticas el entregar las actualizaciones necesarias durante la vida útil de la solución, sin embargo, algunas de ellas aun continúan haciendo cobros en el caso de actualizaciones considerables (cuando una versión se mueve por ejemplo de la 4.1 a 5.1) o mejoras (gráficas por ejemplo).

1.1 Propósito del proyecto

Con la problemática anteriormente planteada es importante comenzar a plantear los requerimientos, por lo cual, el objetivo general de este proyecto es implementar un nuevo esquema de servicio DHCP de alta disponibilidad usando una plataforma que nos permita reducir los costos de licenciamiento y mantenimiento presentes que el software propietario en una institución financiera del cual se hablará en el capítulo 2, proporcionar administración simplificada, de libre modificación y código abierto, para que el administrador de dicho sistema tenga la libertad de hacer modificaciones en caso de contingencia, cuando se detecte alguna falla o simplemente para realizar mejoras o desarrollo de herramientas que nos ayuden aun más con la administración de dicho servicio, además, la alta disponibilidad hará de esta implementación una propuesta confiable para un servicio esencial como lo es la conexión a la red de voz y datos.

1.2 Alcance del proyecto

Este proyecto abarcará los siguientes puntos:

- Implementación del servicio DHCP para IPv4 con software libre.
- Presentación de tres propuestas para dar alta disponibilidad al servicio DHCP, así como sus características, ventajas y desventajas de operación, administración e implementación de cada uno de ellos.
- Establecimiento de políticas de operación del servicio DHCP para PC y telefonía IP.
- Mostrar la base teórica del funcionamiento del servicio DHCP sobre el protocolo IP versión 6.
- Proporcionar un panorama general y actual del software libre.

- Proponer un plan de contingencia y recuperación del servicio en caso de falla.

1.3 Requerimientos funcionales

Funcionalidades ofrecidas por software propietario

El esquema de asignación de direcciones vía DHCP generalmente está compuesto de una serie de políticas de operación en las que se establece la manera en la cual se configurarán, equipos de cómputo o estaciones de trabajo, impresoras, teléfonos IP, máquinas virtuales, etc. Por lo cual uno de los requerimientos funcionales es crear un sistema basado en software libre que permita cumplir las políticas establecidas por la organización donde se implementará el servicio.

Fácil Administración

Una funcionalidad básica para poder hacer una comparación lineal entre el software propietario y el que proponemos, es la parte gráfica que es la que permitirá al administrador reducir la carga de trabajo y simplificarla.

Existen diferentes fabricantes de software (appliance en algunos casos) que ofrecen este servicio (servidor DHCP), según un estudio realizado por Gartner algunas de las marcas más reconocidas para el servidor DHCP son [20]:

- Alcatel - Lucent
- BlueCat Networks
- BT Diamond
- Cisco CNR
- Infoblox

Dichos fabricantes ofrecen en sus productos interfaces gráficas muy similares en las cuales el administrador puede configurar de manera sencilla parámetros tales como, tiempo de arrendamiento de direcciones, número de clientes por subred, estado del servicio, etc. En esta implementación se requiere tener una interfaz gráfica que permita al igual que los productos comerciales, administrar y configurar de manera sencilla el servicio, además de ser confiable y brindar administración centralizada en esquemas como el Failover que es un arreglo de servidores que permite tener

redundancia en el servicio en caso de falla, o en distribución del servicio en sucursales, donde no es posible administrar físicamente el servicio.

1.4 Requerimientos no funcionales

Los requerimientos no funcionales son los que definen los puntos generales del proyecto que deben ser cumplidos, pero que no son necesarios para la operación básica del servicio, en este proyecto se implementarán los siguientes diez:

A) Disponibilidad

Una funcionalidad importante que debe cumplir este servicio, es la alta disponibilidad con la finalidad de que el servicio se proporcione en el momento que un usuario desee conectarse a la red, es decir, que el servicio esté disponible un 99.9%, dicho lo cual, sólo puede fallar 8 horas y 45 minutos al año, por lo que es necesario proponer una infraestructura que brinde redundancia del servicio, ya que como se mencionaba anteriormente, el servicio es proporcionado a estaciones de trabajo, teléfonos IP, impresoras, etc., y de él depende que éstos tengan conexión todos el tiempo y de forma continua a la red, en el capítulo 4 se hablará más a detalle de los diferentes niveles de disponibilidad.

B) Escalabilidad

Se implementará un servicio basado en software libre el cual permitirá en un futuro, soportar nuevas tecnologías, como es el caso de IP versión 6 y además proporcionará actualizaciones en caso de detectarse alguna vulnerabilidad o cuando se hagan mejoras o agreguen nuevas funcionalidades en el servicio o sistema operativo incluso cuando cambie la versión del mismo.

C) Mantenimiento

El mantenimiento es un punto clave en el desempeño óptimo de un sistema, ya que este previene en gran manera fallas en el servicio, permite detectar si este opera de forma distinta a la establecida. Por lo cual es necesario establecer un esquema de mantenimiento, donde se hagan revisiones periódicas del estado del servicio, aplicación de parches o actualizaciones.

D) Plataforma

En el caso de los appliance comerciales, el sistema base o sistema operativo que soportará el servicio ya está incluido por lo cual no hay que preocuparse por éste, pero en el caso de algunas soluciones de software propietario o en el software libre hay que tomar en cuenta la plataforma donde se montará el servicio, ya que de ésta dependerán factores muy importantes como la capacidad de procesamiento, la compatibilidad de hardware, la compatibilidad del servicio, entre otros. Por lo cual el servicio DHCP se implementará sobre un Sistema Operativo basado en Linux ya que brinda un servicio muy confiable y es libre, posteriormente en el capítulo 2 se hablará más a detalle de los diferentes sistemas operativos basados en Linux.

E) Costo

Uno de los principales requerimientos de este proyecto es el costo total que tendrá el producto final, ya que uno de los objetivos es brindar una solución que permita reducir los costos de licenciamiento y mantenimiento del servicio, con esto no queremos decir que el costo de licenciamiento se reducirá a cero, sino que se buscarán diferentes implementaciones de software que nos permitan cumplir con los requerimientos funcionales al menor costo posible.

F) Seguridad

La seguridad del sistema es un punto muy importante para evitar intrusiones y fallas no contempladas en el sistema, por lo que en esta tesis se propone un esquema de seguridad básico que debe tener todo servidor de red.

G) Compatibilidad

Ya que en la mayoría de las organizaciones se cuenta inicialmente con equipos que han sido adquiridos y funcionaban correctamente con soluciones propietarias, se requiere que estos sean compatibles con la nueva implementación basada en software libre, lo cual se logrará evaluando las características y estándares bajo los que opera el servicio de DHCP con software libre y que estos sean a su vez compatibles con los estándares

de operación de todos los equipos configurables vía DHCP con los que cuente la organización.

H) Soporte

En las soluciones comerciales el soporte en la mayoría de los casos significa un costo adicional al costo del producto lo cual no sucede con el software libre, ya que en este caso el soporte lo da el mismo administrador del servicio apoyado por comunidades de desarrolladores los cuales colaboran conjuntamente para dar solución a los problemas que llegase a presentar el sistema. Adicionalmente existen empresas dedicadas a ofrecer soluciones de software libre, donde el negocio es brindar soporte a los usuarios, más que cobrar por las licencias de uso del software, por ello en esta tesis se analizan diversas soluciones y se determina la más conveniente para los diferentes casos de implementación del servicio, tomando en cuenta los factores económicos, de tiempo y recursos humanos.

I) Documentación

Debido a que en la mayoría de los casos, el servicio no es administrado por una sola persona, o esta persona no administrará por siempre el servicio, es necesario diseñar manuales que permitan a los futuros usuarios tener una referencia de la operación del servicio para así, conocerlo, modificarlo y configurarlo de acuerdo con las especificaciones definidas y así evitar confusiones, además de tener una base de conocimientos que sirvan como referencia del funcionamiento actual o para futuras modificaciones.

J) Gestión de errores y plan de contingencia

En caso de que el sistema deje de funcionar o se detecte alguna falla que provoque que éste opere de una forma diferente a la de condiciones ideales, es necesario contar con un plan de contingencia que indique los pasos a seguir para corregir la falla, además de indicar qué hacer en caso de ausencia del servicio. El plan de contingencia debe estar planeado para poder recuperar el servicio lo antes posible y que el tiempo de ausencia del servicio sea mínimo.

K) Hardware

Antes de poner en producción el servicio DHCP que se propone y en general cualquier servicio, es importante hacer pruebas de funcionamiento y simular el comportamiento de éste en distintas condiciones de operación, con el fin de conocer la operación del sistema en diferentes escenarios, por lo cual se utilizará para la realización de las pruebas del servicio el siguiente equipo:

- Router
- Switch
- Equipos terminales (teléfono, PC y máquinas virtuales)
- PC con software para virtualización de servidores (VMware Server)
- Cable UTP

En el laboratorio de pruebas se utiliza equipo con características similares a las del equipo que se utilizará para implementar el servicio que se pondrá en producción, ya que por las especificaciones establecidas inicialmente, se dispondrá de máquinas virtuales para la implementación de los servidores, y estos se ajustaran a las características de rendimiento que requiera el servicio de DHCP. Dicho equipo es proporcionado por un área encargada de administrar la capacidad de procesamiento y memoria de los servidores para evitar desperdicio de recursos de hardware.

Capítulo 2. Software libre como alternativa al software propietario

El término software libre, para la mayoría de las personas, suele asociarse erróneamente con el adverbio gratis, esta asociación dista mucho de la filosofía del software libre, por lo cual uno de los objetivos de este capítulo es brindar conceptos que ayuden a distinguir las principales diferencias entre los conceptos de software libre, software gratuito y software propietario; además de mostrar las ventajas que tiene el software libre y la filosofía con la que se desenvuelve.

2.1 Software Libre

El concepto software libre o free software en inglés, se basa principalmente en la filosofía de que un software debe tener la libertad de ser ejecutado, copiado, distribuido, estudiado, cambiado y mejorado libremente, es decir, deberá estar disponible a cualquier usuario que requiera su implementación o mejora para aplicarlo a sus necesidades particulares tomando en cuenta que al final, él también deberá compartir su desarrollo [10].

La tendencia de los últimos años ha sido la venta de soluciones completas de software y hardware integrados en un sólo equipo conocido como appliances. Como mencionamos con anterioridad, estos equipos se convierten en una caja negra que hacen simplemente lo que se indica con la limitante de tener que adaptarse incluso a las carencias del sistema si es que las tiene.

2.1.1 Un poco de historia

Desde el nacimiento de las computadoras de gran capacidad en los años 60's el software empezó a formar parte importante de los sistemas de cómputo, aunque el mayor peso se le daba a la parte del hardware, poco a poco el software fue considerado como un agregado para hacer funcionar dichos equipos o simplemente que se consideraran los equipos de cómputo como un todo.

Debido a esto se considera que el software nació libre, ya que simplemente no existía la limitante entre software libre y propietario, simplemente se cumplían las condiciones que actualmente existen para decir que un software es libre. Debido a esto los grupos de usuarios como SHARE (Usuarios de sistemas IBM) o DECUS (usuarios de DEC, actualmente y desde mayo de 2008 parte de Connect) intercambiaban libremente el código fuente de sus programas, ya sea para modificarlo y adaptarlo a sus necesidades, hacerle mejoras o simplemente darlo a conocer a la comunidad.

Posteriormente a finales de los 60's y principios de los 70's IBM dio a conocer que sus nuevos equipos de cómputo incluirían sistemas operativos cerrados que traerían como consecuencia prohibiciones en su modificación y distribución, conforme pasaron los años fue tomando más terreno el software privativo, ya que los usuarios se

familiarizaron con la idea de que el software tuviera un costo intrínseco al hardware y además los fabricantes de equipo de cómputo comenzaron a ver futuro en este nuevo negocio. Esta nueva tendencia trajo consigo diferentes perspectivas, una parte de la comunidad desarrolladora comenzó a interesarse en el software propietario, mientras otros cuantos seguían manteniendo la creencia de que el software debía mantenerse como hasta algunos años atrás "libre", debido principalmente a las limitaciones que tenía el software propietario, ya que éste no permitía las modificaciones y eso provocaba que en caso de alguna falla, había que notificar al fabricante de dicho problema y esperar a que éste fuera corregido, lo que ocasionaría pérdidas innecesarias de recursos [24].

2.1.2 UNIX

A principios de la década de los 70's y 80's comenzaron a desarrollarse notables proyectos como son SPICE y Tex, ambos enfocados a mantener la filosofía que poco a poco iba perdiendo fuerza, el primero desarrollado en 1973 en la Universidad de California por Donald Pederson estaba enfocado a la simulación de circuitos electrónicos y el segundo desarrollado por Donald Knuth en 1978 el cual es un sistema de escritura cuyo objetivo es producir documentos con un formato de calidad, además de uno de los más importantes en materia de software libre UNIX.

UNIX creado por Thompson y Ritchie desde 1972 e impulsado por los laboratorios Bell de AT&T, fue un pilar fundamental para el software libre, ya que éste fue desarrollado inicialmente bajo los términos de licencia que permitían su libre distribución, modificación y estudio, éste tuvo su mayor impulso en la Universidad de California en Berkeley quien posteriormente por problemas de licenciamiento y falta de acceso al mismo fue encareciendo el proyecto hasta llegar al grado en que dicha institución seguía un proceso legal con la división *Unix System Laboratories* de AT&T por publicar el código de este sistema operativo, lo que ocasionó que se perdieran los términos de distribución de versiones que actualmente están establecidos en la filosofía del software libre [24].

2.1.3 Nacimiento del proyecto GNU/Linux

Hasta este momento los intentos por consolidar el software libre eran muy vagos y poco fundamentados o simplemente no cumplían con el objetivo de hacer libre el software. Fue hasta los años 80's cuando Richard Matthew Stallman a consecuencia de malas experiencias que tuvo con el software propietario en el laboratorio de Inteligencia Artificial del MIT donde trabajaba, decidió abandonar el software propietario para comenzar a desarrollar lo que actualmente conocemos como el proyecto GNU proveniente del acrónimo *GNU is Not Unix* (véase figura 2.1).

Figura 2.1. Icono de proyecto GNU.

Una de las principales preocupaciones de Stallman eran los términos de licencia bajo los cuales se desarrollaría el proyecto, es por ello que creó lo que actualmente se conoce como GPL (General Public License) con el fin de garantizar que un software fuera libre. De ahí el surgimiento del termino *copyleft.*

Fue hasta 1985 cuando se creó la Fundación para el Software Libre. Ésta fue la que proporcionó un sólido soporte financiero, logístico y legal al proyecto GNU. En 1990 con el nacimiento de nuevas herramientas como emacs, gcc y algunas librerías fundamentales para tener un sistema Unix fue creciendo el proyecto, aunado a esto todavía no se contaba con la parte más importante del sistema: el kernel. Debido a la amplia difusión que ya tenía el proyecto, comenzó a buscarse un kernel para GNU, y en esta búsqueda se encontró con la comunidad BSD, quien se dirigía en la misma dirección. BSD logró hacer una implementación de un sistema operativo y en 1992 Bill Jolitz concreta los desarrollos y crea un sistema operativo llamado 386BSD, el cual se

convertiría en la base de los sistemas NetBSD, FreeBSD y OpenBSD. Posteriormente en julio de 1991, Linus Torvalds hace público su interés por crear un sistema operativo similar a Minix. Fue hasta marzo de 1994 cuando se liberó la versión 1.0 del kernel que se consideraba la versión más estable, que fue la base para que los desarrolladores empezaran a trabajar en Linux integrando a su alrededor todo el software GNU y muchos programas libres que se distribuirían bajo la los términos de licencia GPL.

Con el surgimiento de nuevas y variadas versiones de Linux nace el término *distribución* que se refiere a la utilización del mismo núcleo (Kernel) de Linux pero con modificaciones específicas en el entorno de herramientas que lo compone. De las primeras distribuciones (MCC Interim Linux de la Universidad de Manchester, TAMU de Texas A&M, y la más conocida SLS que más tarde dio lugar a Slackware, que aún se distribuye hoy día), suelen ofrecer sus propias herramientas para gestionar los paquetes, la instalación inicial en un equipo, y la administración del sistema operativo. A partir de ese momento el proyecto GNU/Linux comenzó a popularizarse aún más naciendo distribuciones como las que actualmente conocemos. Algunas de las más reconocidas son Debian, Suse, Redhat, Mandrake, etc.

A mediados de los 90's había mucho por hacer principalmente en materia de interfaces gráficas, aun así, esto se expandió al grado de empezar a tomar gran importancia en los sectores empresariales y empezar a ser considerado por dicho sector como algo digno de estudio. Esto ocasionó que el sector financiero pusiera sus ojos en el software libre, tal es el caso de Red Hat Linux enfocado en el manejo y mantenimiento del sistema por parte de usuarios sin conocimientos en informática.

Linux es un sistema operativo basado en Unix pero bajo la licencia de GNU, lo que quiere decir que es un sistema operativo que no tiene costo, puede ser distribuido y modificado según se requiera ya que viene con su código fuente que puede ser modificado según las necesidades del usuario.

Tiene sus inicios en la década de los noventa por Linux Torvalds, aunque sigue en continuo desarrollo con el apoyo de muchos colaboradores a través de internet. Linux

se inicio inspirado en el proyecto Minix, que es un pequeño sistema Unix desarrollado por Andy Tannenbaum. Torvalds quería crear un "mejor Minix que el Minix". La primera versión oficial de Linux salió el 5 de octubre de 1991, que fue la versión 0.002.

Características principales:

- *Sistema multitarea*

Pueden ejecutar varios procesos a la vez sobre un mismo procesador [11].

- *Sistema multiusuario*

Proveen de recursos y aplicaciones a varios usuarios de manera simultánea, y como también es multitarea pueden estar ejecutando cada uno de los usuarios varias tareas a la vez.

- *Shells Programables*

Un shell conecta las ordenes del usuario con el núcleo del sistema, y al ser programables se puede modificar para adaptarlo a las necesidades del usuario.

- *Independencia de dispositivos*

Linux admite la mayoría de dispositivo de E/S, ya que una vez instalado se añade el controlador al kernel por lo que Linux es muy adaptable.

- *Eficiencia*

Linux es un sistema operativo muy eficiente ya que aprovecha al máximo los recursos del equipo. No necesita de mucha capacidad para poder funcionar.

- *Multiplataforma*

Puede ejecutarse en diferentes plataformas de hardware.

- *Conectividad*

Linux ofrece una variada gama de posibilidades al interconectarse con otros servidores. Soporta TCP/IP. Tiene la capacidad de actuar como cliente o servidor según se requiera.

- *Convivencia*

Puede convivir con otros sistemas operativos en el mismo disco duro.

- *Herramientas de desarrollo*

Existen una gran variedad de lenguajes de programación y herramientas de desarrollo disponibles para Linux (GNU C/C++, Java, Simula, Perl, etc).

- *Costos*

Es un sistema operativo sin costo ya que no se tiene que pagar por ningún tipo de licencia, además la forma en la que está diseñado lo hace resistente a virus, además que en cuanto a requerimientos de Hardware los sistemas Linux ofrecen más flexibilidad ya que requieren menos recursos de memoria y procesador, y por lo tanto hardware menos complejo, Como se detalla en la tabla 2.1.

Tabla 2.1 Tabla comparativa entre sistemas operativos.

	Precio	Virus	Complejidad de Hardware	Tipo de licencia
Linux	$0	No	No	Libre
Windows	$US	Sí	Sí	Propietario
Novell	$US	Sí	Sí	Propietario

Distribuciones

Una distribución es un software particular basado en el núcleo de Linux pero que incluye determinados paquetes de software para satisfacer las necesidades de un grupo determinado de usuarios, creándose así los tipos de ediciones: domésticas, empresariales y para servidores. Por lo general utilizan software libre y en algunas ocasiones llegan a utilizar software propietario dependiendo del tipo de distribución.

Actualmente, existen una gran variedad de distribuciones Linux, a continuación se listan algunas de las más conocidas:

- Gentoo

  Esta distribución puede ser modificada desde cero por lo que es orientada a usuarios con más experiencia en estos sistemas operativos libres.

- Knoppix

  Está basada en Debian su principal característica es ser un LiveCD por lo que es útil cuando se desea utilizar un sistema Linux sin realizar modificaciones en los equipos.

- CentOS

  Es un sistema operativo desarrollado a partir del código fuente de Red Hat por lo que es muy similar a él y ofrece características similares a diferencia que es gratuito.

- Fedora

  Distribución financiada por Red Hat y mantenida por desarrolladores y la comunidad, informando y reparando las fallas encontradas.

- Ubuntu

  Basado en Debian, con gran facilidad de uso e instalación para el usuario, utiliza la interfaz gráfica GNOME.

- Kubuntu

  Distribución de fácil uso, similar a Ubuntu, la principal diferencia es que utiliza escritorio KDE por defecto.

- Red Hat Enterprise Linux

  Sistema operativo donde el soporte y actualización sólo se proporcionan por parte de la empresa que lo distribuye bajo la adquisición de licencia.

2.1.4 Software libre del siglo XXI

A principios del siglo XXI el software libre comenzó a tomar fuerza en materia de servidores y empezó a subsanar una de sus debilidades que impedía su crecimiento, la interfaz gráfica. Con sistemas de gestión gráfica de gran facilidad de uso e instalación como Gnome 2.x y KDE 3.x, tomó gran fuerza y fue tomado en serio por grandes fabricantes como IBM, HP, Oracle, Corel, Apple, Sun, etc. Así mismo estos gestores gráficos se adaptaron para ser compatibles con las diferentes distribuciones de software libre.

La apertura del mercado del software libre trajo consigo el interés por éste en la comunidad universitaria, por ejemplo, en México se crearon proyectos para impulsar estas tecnologías a pesar de no tratarse de un país desarrollador de software. Casos particulares fueron el Laboratorio de Investigación y Desarrollo de Software Libre (LIDSOL) en la Facultad de Ingeniería de la UNAM y uno de los desarrollos más importantes, el proyecto GNOME en 1999 de Miguel de Icaza.

2.1.5 Acuerdos y tipos de licencias

Formalmente, lo que marca la diferencia entre software libre y software propietario es la licencia, con esto nos referimos a un contrato establecido entre el autor o propietario de los derechos y los usuarios; en él se establecen los usos de lo que se puede hacer con su obra, así como las modificaciones permitidas y la redistribución de la misma, por citar tan sólo algunos de los lineamientos.

- *La licencia BSD (Berkeley Software Distribution)*

Se originó de las versiones del sistema UNIX realizadas en la universidad de California en Berkley, en EE.UU. Ésta establece que es obligatorio dar crédito a los autores, además, permite la redistribución del código fuente y los archivos binarios aunque no es necesario que esto se realice. Por otra parte, permite la modificación e integración con otros programas casi sin restricciones.

- *La Licencia Pública General de GNU (GPL de GNU)*

En este tipo de licencias el creador conserva los derechos de autor y protege las libertades fundamentales del software que son:

1. La libertad de ejecutar el programa en cualquier equipo, cualquiera que sea nuestro propósito (libertad 0).
2. La libertad de estudiarlo y adaptarlo según nuestras necesidades, y para esto es necesario conocer el código fuente (libertad 1).
3. La libertad de redistribuirlo de modo que podamos brindar copias para ayudar al prójimo (libertad 2).
4. Libertad para hacer mejoras al programa y hacerlas públicas en todas sus versiones, nuevamente y para esto es necesario disponer del código fuente (libertad 3).

Estas libertades permiten hacer modificaciones, distribuir ya sea de manera gratuita o con algún costo a cualquier persona en cualquier parte del mundo. En general se dice que se es libre de hacer lo que se quiera sin tener que pagar costo alguno, por lo cual puede usarse en el ambiente de trabajo y hacerle modificaciones y guardar esas versiones sin tener que notificar a nadie, y en caso de hacerlo, no tendría que notificarse a alguien en específico.

2.1.6 Mercado actual del software libre en México

Actualmente el país atraviesa una difícil situación financiera debida a la crisis mundial, lo cual trae como consecuencia que ante la ausencia de recursos se busquen alternativas para reducir costos; dado esto, la oportunidad del software libre de incursionar en el mundo real es mayor, ésta es una de las razones que motiva la elaboración del presente trabajo, ya que como se dijo anteriormente uno de los objetivo es comenzar a utilizar el software libre como una alternativa al software propietario, consiguiendo con ello grandes beneficios como lo son el ahorro de recursos y una mayor eficiencia.

2.2 Ventajas del software libre

Mucho se ha hablado del software libre, del cómo nació y bajo qué términos se rige, pero aún no se ha dicho como es que todas estas cualidades repercuten en su calidad, así como las cualidades de las que éste goza, para ello listamos algunas características sobresalientes:

1. *Economía*

El bajo costo que representa el software libre es una de las principales ventajas, ya que en el mercado actual el costo del software ha superado el costo del hardware mismo y esto refleja que más del 50% del costo de un equipo con software propietario recae sobre el software. El ahorro mencionado podría permitir a las organizaciones utilizar esos recursos para proporcionar más servicios o simplemente para disminuir los costos invertidos en TI.

2. *Libertad de uso*

En comparación con productos de software propietario, la implementación de software libre permite instalar en el número de máquinas que sea necesario, ya que no existen limitaciones en cuanto al número de equipos que pueden correrlo ni número de nodos a los que se asignan direcciones.

3. *Independencia Tecnológica*

Una de las principales ventajas es que debido a la disposición del código fuente para su libre modificación el usuario puede hacer las modificaciones necesarias al código sin necesidad de esperar a que el fabricante corrija dicha falla como ocurre en el caso del software propietario.

4. *Soporte y compatibilidad a largo plazo*

Debido al interés económico que existe en los productos de software propietario una vez que este producto ha vendido la mayoría de sus licencias se vuelve poco rentable para los vendedores del mismo por lo que se les deja de dar soporte y por lo tanto a la corrección de errores que puedan contener, para dar paso a nuevas versiones de dichos productos, dejando al usuario sin más opción que comprar nuevos programas de software aunque el software adquirido previamente siga cubriendo sus necesidades. En el software libre es lo contrario, debido a que el principal interés es el desarrollo de un producto de calidad además que se cuenta con colaboradores para lograr este fin, el software tiene un mayor tiempo de vida y en caso de que dicha implementación satisfaga las necesidades del cliente no hay necesidad de cambiarla al ocurrir errores, ya que él mismo puede corregirlos.

5. *Sistemas más seguros, robustos y sin backdoors*

Nuevamente, debido a la disposición del código se vuelve muy difícil que existan puertas traseras que además contraponen la filosofía de software libre, además, que si se conoce el código fuente por la comunidad es más fácil detectar agujeros de seguridad que podrían provocar intrusiones no deseadas como sucede en algunos sistemas que utilizan la seguridad por oscuridad.

6. *Métodos simples y unificados de gestión de software*

La comunidad actual proporciona servicios de software unificados, algunas veces basta con teclear un simple comando o seleccionar alguna casilla para obtener librerías y complementos de fuentes seguras, con la certeza de que no se trata de programas de dudosa procedencia que pueden contener spyware, malware o algún tipo de software malintencionado. Ver Figura 2.2. Diagrama que muestra las ventajas del software libre.

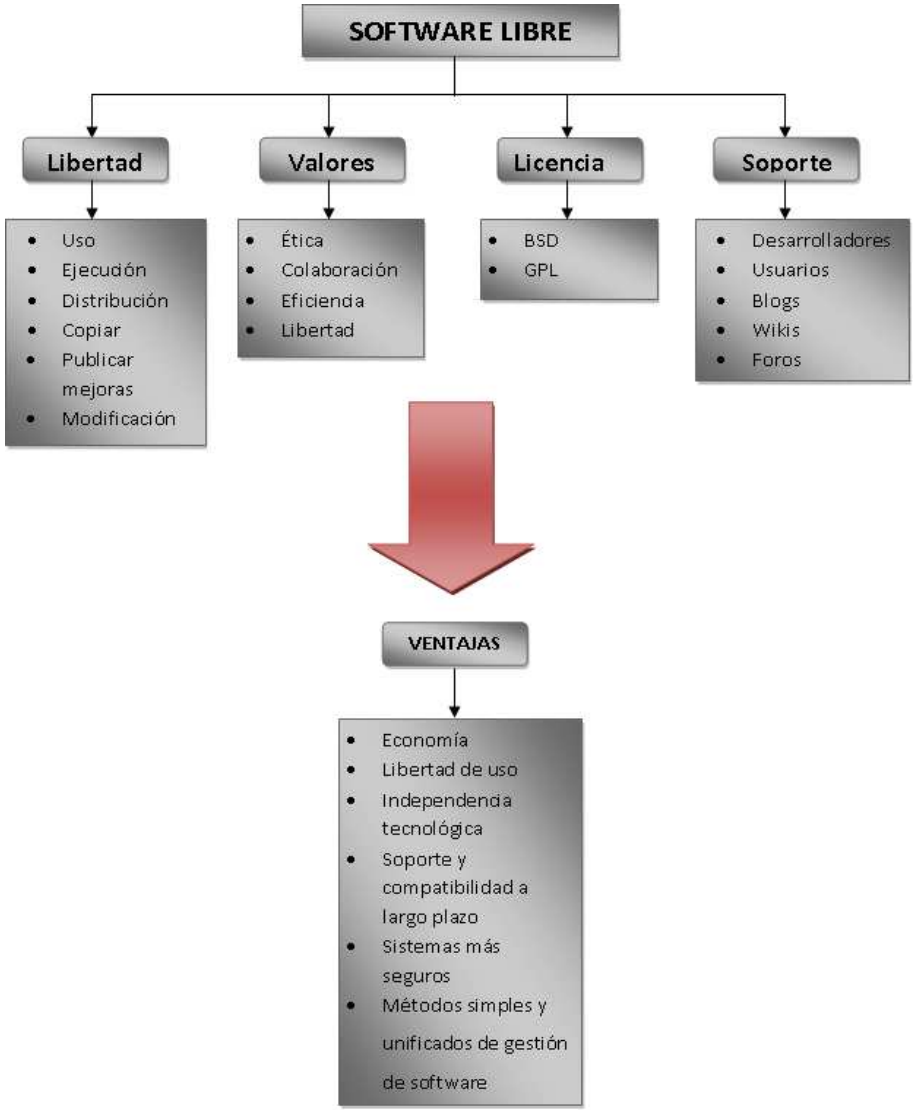


Figura 2.2. Diagrama Software Libre

2.3 Costos actuales de licenciamiento

Actualmente el mercado de appliance y software dedicado a brindar servicio DHCP ofrece a las organizaciones sistemas de administración donde se integran varias soluciones en una llamada *DDI,* estas integran en uno sólo el servicio de DNS, DHCP e IPAM, con el fin de permitir al administrador centralizar la administración de dichos servicios.

Algunas de estas implementaciones ofrecen servicios de muy buena calidad, confiables, de fácil implementación y administración, lamentablemente el costo económico es muy alto. En el caso de las soluciones de software libre tenemos las mismas características, aunque no necesariamente todas se brindan en un sólo

producto, sino que es necesario integrar varios proyectos para ofrecer las mismas características que las soluciones con costo. En la Tabla 2.2 se muestran algunas de las soluciones que se encuentran en el mercado y los costos de cada una.

Tabla 2.2 Tabla comparativa de costos de licenciamiento.

Concepto	ISC DHCP	Cisco CNR	BlueCat	Infoblox
Licenciamiento de servidor de DHCP	ISC DHCP versión 3 Libre	Kit de licencia Base y 10K nodos IP $20,000 usd	Adonis 500 DHCP appliance $ 7,444.33 usd	Infoblox - 1050 $10,995.00 usd
Soporte y mantenimiento (1 año)	Mantenimiento a cargo del administrador del sistema	N.D	$ 5,004.00 usd	En sitio $8,581.62 usd
appliance	No	No	Sí	Sí
Instalación Incluida	Sí	No	Sí	Sí
Seleccionar Módulos	Sí	No	No	No

2.4 Software libre y appliance comerciales

Como se mencionó anteriormente el mercado de soluciones que ofrecen el servicio de DHCP está enfocado a soluciones *DDI* en las que se integran en una misma base de datos los servicios de DNS y DHCP para evitar la complejidad del sistema.

Actualmente, el mercado de los DDI está creciendo de forma importante, debido a que cada vez son más los dispositivos que requieren de este servicio, tal es el caso de la telefonía IP, dispositivos móviles, servicios virtualizados, entre otros.

Para proporcionar dichos servicios los fabricantes de appliance comerciales en algunos casos hacen uso de soluciones de software libre de DHCP y DNS de las que se sabe, brindan servicio de manera satisfactoria con ayuda de las implementaciones de software libre y la implementación interna. Estas soluciones se ofrecen en 3 diferentes presentaciones:

1. *Software:* Corren en sistemas Windows y Unix/Linux
2. *Appliance dedicados*: appliance de un sólo propósito DNS/DHCP y funciones IPAM. Varias de estas soluciones están basadas en software libre y tienen capacidad de alta disponibilidad y mejoras incluidas.
3. *Appliance Hibridas:* aplicaciones para las WAN, optimización de controladores (WOC) y sistemas virtuales.

En la Tabla 2.3 se detallan las características de algunas implementaciones del mercado.

Tabla 2.3 Características de soluciones comerciales

	Software	Appliance dedicados	Appliance Hibrido	Servicio DNS/DHCP
Alcatel-Lucent	Sí	Sí	No	Desarrollo interno
BlueCat Networks	No	Sí	No	ISC BIND y DHCP
BT Diamond IP	Sí	Sí	No	ISC BIND y DHCP
Cisco (CNS Network Registrar)	Si	Si	No	Desarrollo interno
EfficientIP	Si	Si	No	ISC BIND y DHCP
Infoblox	No	Si	Si	ISC BIND y DHCP

Fuente: www.gartner.com (2008)

Como se puede observar muchos de los appliace comerciales utilizan el Servicio DHCP de ISC, lo cual nos asegura que dicha implementación es fiable y cumple con los requerimientos de implementación en una LAN con un gran número de usuarios. Adicionalmente un estudio de mercado realizado por Gartner en 2009 muestra que las marcas líderes (tabla 2.4), utilizan internamente en sus productos el ISC DHCP.

Tabla 2.4 Marcas líderes en 2009

Clasificación					
	Negativo	***Precaución***	***Prometedor***	***Positivo***	***Excelente***
Alcatel-Lucent			**x**		
BlueCat Networks				**x**	
BT Diamond				**x**	
EfficientIP		**x**			
Inflobox					**x**
Men & Mice		**x**			
A partir del 11 de Noviembre del 2009 **Fuente: Gartner (Noviembre 2009)**					

Características sobresalientes de Appliance comerciales

Como se observa en la tabla 2.4 Infoblox, es la solución comercial más recomendable, ya que ofrece características que sus competidores no ofrecen, o las ofrecen pero en forma limitada, por lo que servirá como punto de referencia para esta tesis. Algunos de los puntos importantes que hacen que Infoblox sea el líder para servicios LAN son:

- Administración de una única base de datos, que es replicada automáticamente a todos los appliance
- Ofrece redundancia, es decir si el appliance falla automáticamente otro appliance en la red se hace cargo de brindar el servicio de DHCP/DNS
- Tiene una interfaz gráfica basada en Web

2.5 Sistemas operativos para servidores

Para el desarrollo del proyecto se ocuparon los sistemas operativos Red Hat y CentOS a continuación se dará una breve descripción, así como las ventajas de estas distribuciones

2.5.1 Red Hat Enterprise Linux (RHEL)

Se conforma por software libre y código abierto, se publica en formato binario y a diferencia de otras distribuciones Linux, RHEL es un sistema Linux creado principalmente para el uso empresarial y ofrece una licencia con costo que proporciona actualizaciones y soporte técnico a través de Redhat Networks, por lo es una muy buena alterativa a un costo competitivo a diferencia de otros sistemas comerciales.

Red Hat Enterprise posee la capacidad de ofrecer soporte para aplicaciones de alta disponibilidad, escalabilidad, óptima utilización de recursos, ofrece flexibilidad y máximo control operativo. Es utilizado en entornos donde se requiere operación continua, muchos recursos y altos niveles de flexibilidad.

Características:

- *Interfaz gráfica*

  Utiliza GNOME como interfaz gráfica, es fácil de utilizar por el usuario estándar ya que es similar a la interfaz utilizada en los sistemas Windows.

- *Requerimientos de hardware mínimos y máximos.*

  Para un funcionamiento ideal el sistema requiere preferentemente 25 Gb de espacio en disco duro y memoria RAM de 2 GB como mínimo, así como un procesador preferentemente doble núcleo a 2.4 Ghz. Debido a su capacidad de procesamiento y rendimiento es capaz de soportar la más alta tecnología como los procesadores Intel® Xeon® 7500, AMD's Opteron™ 6000 y IBM POWER7, con la capacidad de soportar hasta 32 procesadores y 512 GB de memoria.

- *Código abierto*

  Al ser de código abierto puede ser modificado para ajustarlo a las necesidades del usuario o simplemente se puede saber cómo opera internamente una aplicación.

- *Seguridad*

  SELinux es una arquitectura de seguridad que viene integrada en el kernel de Linux, consta de un conjunto de políticas para brindar seguridad para dotar de mayor seguridad al sistema. En algunos sistemas operativos SELinux no viene instalado por defecto, pero puede ser integrado si así se requiere.

- *Fiabilidad*

  Posee mecanismos que permiten conocer de forma precisa las fallas que ocurren en el sistema, posee mecanismos para atrapar las fallas, asi como configuraciones de alta disponibilidad.

- *Soporte para procesadores multi-core*

  Soporta procesadores con más de un núcleo en un mismo circuito integrado, esta es una gran ventaja ya que algunos sistemas operativos comerciales no aprovechan la capacidad real de los procesadores multi-nucleo.

- *Arquitecturas*

  Soporta gran cantidad de arquitecturas como son x86 (32-bit Pentium, AMD), x86-64 (AMD64, EM64T), entre otros.

Costos de licenciamiento Red Hat Enterprise Linux

En la Tabla 2.5 se muestran los costos de licencia en cada una de las versiones del RHEL

Tabla 2.5 costos RHEL

Precios	RHEL Básica	RHEL Estándar	RHEL Premium	RHEL AP Estándar	RHEL AP Premium
Precios MXP con instalación	$575	$919	$1,499	$1,799	$2,949
Precios USD sin instalación	$349	$799	$1,299	$1,499	$2,499
Precios Mantenimiento anual MXP	$44,160.00	ND	ND	ND	ND

2.5.2 CentOS (Community Enterprise Operating System)

Sistema operativo creado por desarrolladores a partir del código fuente que libera RedHat, es muy similar a RH a diferencia que este es gratuito y ofrece capacidades "recortadas" además que el soporte es proporcionado por la comunidad, es una distribución gratuita [12].

Características:

- *Interfaz gráfica*

  Utiliza interfaz gráfica GNOME, se caracteriza por su facilidad de uso para usuarios principiantes. Si se requiere se puede cambiar de interfaz gráfica, como KDE.

- *Requerimientos de hardware mínimos*

  Requiere una memoria RAM mínima de 64 MB y un disco duro mínimo de 1 GB (recomendado 2 GB) por lo que el hardware no es un impedimento para su implementación

- *Soporte para sistemas*

  Intel x86, Power PC/32, AMD 64, Intel ME64T, entre otras.

- *Código abierto*

  Es un sistema operativo libre por lo que no tiene costo

- *Seguridad*

  SELinux puede ser integrado si se requiere. Además están disponibles actualizaciones de seguridad.

- *Soporte técnico*

  El soporte técnico se da por parte de desarrolladores mediante foros, manuales, chat IRC, entre otros.

- *Aplicaciones de desarrollo*

  Soporto muchas aplicaciones de desarrollo, como PHP, MySQL, Apache entre muchos otros, además de que es casi compatible con todos los productos de RHEL.

2.6 Webmin

En el punto 2.4 se hablaba de las características principales de los sistemas propietarios y una de las más sobresalientes es la interfaz gráfica que permite a los usuarios administrar de manera sencilla el servicio DHCP.

El servicio de ISC DHCP por defecto no trae ningún gestor gráfico para configurar el sistema, todo se hace en un archivo de texto plano, por esto es necesario utilizar un módulo adicional, en este caso se utilizará Webmin, que cumple con las características que requiere el proyecto, en el capítulo 3 y 4 se hablará más a detalle de cada uno de sus módulos.

Webmin es una herramienta de administración de sistemas UNIX desarrollada en lenguaje Perl por el australiano Jamie Cameron y está liberado bajo Licencia BSD. Esta herramienta permite configurar y administrar equipos de manera sencilla ya que cuenta con una interfaz gráfica accesible vía web como se muestra en la figura 2.3. El puerto por defecto por el cual se comunica es el 10000 TCP, adicionalmente puede ser configurado para usar SSL, siempre y cuando se tenga el módulo OpenSSL instalado.

El objetivo de esta herramienta es administrar de en una misma interfaz web la mayoría de los programas/servicios comúnmente utilizados en servidores UNIX, para una fácil administración.

Webmin está dividido en módulos lo que hace que sea más fácil agregar o eliminar servicios de forma independiente, por lo que se pueden instalar sólo los necesarios y pueden ser descargados desde la página oficial de webmin http://www.webmin.com.

Figura 2.3.Interfaz gráfica de Webmin usando el explorador web

Webmin permite realizar tareas como: configuración de equipos de manera local o remota, archivos de configuración del sistema, así como controlar o modificar aplicaciones Apache, Samba, DHCP, etc. Otra ventaja que es que permite controlar varias máquinas a través de una interfaz simple, o iniciar sesión en otros servidores Webmin de la misma subred o red de área local.

En está implementación es utilizado para administrar el servidor DHCP y poder acceder a los demás servidores vía web. Webmin permite administrar el servidor DHCP de manera remota de manera fácil y cuenta con una interfaz gráfica para administrar el servicio DHCP ISC, donde se pueden asignar parámetros de configuración de direcciones IP, editar archivo de configuración, entre otros.

Lamentablemente este módulo no cuenta con funcionalidades para la búsqueda rápida del estado de asignación de direcciones IP, por esto para cumplir con los requerimientos del proyecto se desarrollará un módulo Webmin en Perl para dicha tarea, del cual se hablará en el Capítulo 3 de esta tesis.

Capítulo 3. DHCP

(Dynamic Host Configuration Protocol)

Debido a la creciente incorporación de dispositivos a las redes de datos como teléfonos IP, máquinas virtuales, teléfonos celulares, PDA, cámaras de seguridad, entre otros, la demanda de direcciones IP es cada vez mayor y por tanto su administración se hace más complicada; es por esto que el servicio DHCP toma gran importancia en la administración de direcciones IP, ya que permite administrar direcciones de forma automatizada y centralizada (dinámicamente). En este capítulo se hablará a detalle del protocolo DHCP, sus características, funcionamiento e implementación en un sistema basado en Software Libre.

3.1 DHCP

El protocolo DHCP es un conjunto de reglas que proporcionan de manera automática parámetros de configuración de red a los equipos que pueden ser configurados vía DHCP como direcciones IP, máscara de red, puerta de enlace predeterminada, servidores DNS, entre otros.

DHCP es un protocolo cliente/servidor, donde el servidor contiene una lista (pool) de direcciones IP que puede asignar a los clientes que realizan una petición (DHCP Discover) para obtener parámetros de configuración de red.

Dentro de sus funcionalidades, cuenta con un archivo llamado *dhcpd.leases* que contiene toda la información de las direcciones IP dinámicas que han sido asignadas, el tiempo de préstamo y el historial de asignación de esa dirección.

El protocolo DHCP se publicó en octubre de 1993 y está documentado en el RFC 2131, por la IETF (Internet Engineering Task Force) y en el RFC 2132 para las opciones DHCP. Mediante él, los clientes obtienen parámetros de configuración del servidor por un periodo definido, y cuando el arrendamiento expira el cliente debe renovar su dirección, generalmente se le reasigna la misma dirección.

DHCP es un protocolo que se encarga de asignar direcciones IP de forma automática a los equipos a través de la red, lo que significa que no hay que ingresar manualmente la dirección en el equipo que se desea configurar. El servicio DHCP permite al administrador supervisar y distribuir de forma centralizada las direcciones IP automáticamente, es decir, asignar y enviar una nueva IP si el equipo lo solicita, independientemente de si está conectado en una red diferente de la del servidor.

3.1.1 Breve historia

El protocolo DHCP desciende del protocolo Bootstrap (*BootP*). BootP se desarrolló para permitir la configuración de estaciones de trabajo sin disco, fue uno de los primeros métodos para asignar direcciones IP de forma estática (asocia un dirección física con una dirección IP) a equipos de cómputo, impresoras, entre otros. Al ser las redes cada vez más grandes, BootP ya no era adecuado, ya que tenía muchas limitaciones por lo

que DHCP fue lanzado como complemento para él. DHCP es compatible con BOOTP y elimina sus limitaciones.

A continuación se muestra la Tabla 3.1 que permite observar algunas comparaciones de estas diferencias mencionadas [2]:

Tabla 3.1 Principales diferencias entre BOOTP y DHCP

BOOTP	DHCP
Número limitado de parámetros de configuración (4 parámetros principales).	Mayor número de parámetros de configuración (30 parámetros).
Asignación estática de direcciones.	Asignación dinámica y estática de direcciones.
Asignación permanente.	Asignación arrendada.

DHCP y BOOTP utilizan los puertos asignados por el IANA (Autoridad de Números Asignados en Internet según siglas en inglés):

- 67 (UDP): Para el tráfico del servidor
- 68 (UDP): Para el tráfico de los clientes

Estos puertos son utilizados tanto para enviar como para recibir mensajes y son conocidos como puertos BOOTP.

3.1.2 Parámetros básicos configurables en un equipo.

Un servidor DHCP brinda parámetros de configuración a equipos que les permitan acceder a la red. A continuación se muestran los parámetros configurables mínimos que se requieren para que un equipo pueda a la red:

- *Dirección IPv4*

Es una dirección de 32 bits, que identifica a un equipo de manera única, por lo que no deben existir equipos con la misma dirección IP en un mismo segmento de red.

- *Servidor DNS*

Provee a los equipos clientes una dirección IP a quien podrán consultarle los nombres o alias de equipos que se conocen a través de la red. Este proceso de asociación de nombre de dominio a una dirección IP es un proceso transparente al usuario, dejando el trabajo a los servidores DNS.

- *Puerta de enlace o Gateway*

Es una dirección IP que se usa para reenviar paquetes de una subred a otra, siempre que no esté disponible otra información de enrutamiento, por lo general se trata de un equipo que hace posible que un cliente conectado a una red (LAN) tenga acceso a una red exterior u otra LAN de la misma institución, para el primer caso generalmente realizando operaciones de traducción de direcciones IP (NAT).

- *Máscara de subred*

Identifica que bits pertenecen al identificador de red, el número de subred y el host indica si una dirección IP pertenece a la subred.

3.1.3 Formato del mensaje DHCP

En la Figura 3.1 se muestra el formato del mensaje DHCP, con sus campos y su tamaño en bytes.

Código OP 1 byte	Tipo de hardware 1 byte	Longitud de Hardware 1 byte	Saltos (HOPS) 1byte
ID de transacción (XID): 4 byte			
Segundos: 2 byte		Indicadores (flags) : 2 byte	
Dirección IP del cliente (CIADDR): 4 bytes			
Dirección IP asignada (YIADDR): 4 bytes			
IP del servidor DHCP (SIADDR): 4bytes			
IP de Gateway (GIADDR): 4 bytes			
Dirección hardware del cliente (CHADDR): 16 bytes			
Nombre del servidor (SNAME): 64 bytes			
Nombre del archivo: 128 bytes			
Opciones CDP			

Figura 3.1 Formato del mensaje DHCP

- *Código OP*

Mensaje de código de operación. Hay dos tipos:

- Se utiliza “1” cuando el mensaje es enviado por un cliente.
- Se utiliza “2” cuando el mensaje es enviado por el servidor.

- *Tipo de hardware*

Existen diversos tipos, a continuación se muestran algunos:

- Ethernet (1).
- Ethernet experimental 3Mb (2)
- Amateur Radio AX.25 (3)
- Proteon ProNET Token Ring (4)
- IEEE-802 (6)
- ARCNET (7)
- Hyperchannel (FDDI) (8)
- Frame Relay (15)

- *Longitud de hardware*

Número de bytes de la dirección física.

- *Saltos (Hops)*

Puesto por el cliente a 0, se incrementa por cada máquina que redirige el mensaje.

- *ID de transacción (XID)*

Identificador de transacción. Es un número aleatorio elegido por el cliente y usado por él y el servidor para asociar mensajes y respuestas entre ellos. Asocia la petición de inicio con la respuesta generada.

- Tiempo

Segundos transcurridos desde que el cliente inicia el proceso de DHCP, tomando a partir del arranque de la máquina.

- *Indicadores (Flags)*

El bit que se encuentra a la izquierda de este campo se usa como indicador de broadcast, cuando está fijado en 1 y todos los demás están en cero, esto porque son reservados para futuros usos. Por lo general los servidores DHCP tratan de entregar los mensajes DHCPREPLY directamente al cliente usando el método de transmisión unicast. La dirección de destino en la cabecera IP se pone al valor de la *dirección IP* fijada por el servidor DHCP, y la dirección MAC a la *dirección hardware* del cliente DHCP. Si un equipo no puede recibir un datagrama IP en transmisión unicast hasta saber su propia dirección IP, el bit de broadcast se debe poner a 1 para indicar al servidor que el mensaje DHCPREPLY se debe enviar como un broadcast en IP y MAC. En caso contrario debe ponerse en cero.

- *Dirección IP cliente (CIADDR)*

Fijada por el cliente o 0.0.0.0 si no la conoce. Sólo se rellena si el cliente conoce su dirección IP y puede responder peticiones ARP.

- *Dirección IP asignada (YIADDR)*

Dirección IP del cliente. Es la dirección IP que el servidor le ofrece al cliente en caso de que el campo anterior (CIADDR) esté puesto en 0.0.0.0, si el cliente ya cuenta con alguna se la deja.

- *Dirección IP del servidor DHCP (SIADDR)*

Dirección IP del servidor que ofrece la concesión.

- *IP de Gateway (GIADDR)*

  Dirección IP del Gateway. Este campo sólo se utiliza para arrancar a través de un agente transmisor.

- *Dirección hardware del cliente (CHADDR)*

Dirección física del cliente usada por el servidor para identificar cual de los clientes registrados está arrancando.

- *Nombre servidor DHCP (SNAME)*

Nombre del servidor DHCP. Es optativo.

- *Nombre del archivo de arranque*

Optativo. Dado por el servidor para indicar al cliente un archivo de arranque (BOOT) en el cliente o en un servidor de configuración.

- *Opciones CDP*

Este campo lleva una secuencia de opciones, estas transportan configuraciones adicionales de información entre el cliente y el servidor. Cada opción incluye un código de opción, tamaño y datos de la opción. El código de opción identifica la opción específica y la información llevada. La parte de datos es la información actual y el tamaño de la opción es el tamaño de los datos y es en bytes. Este campo es optativo.

3.1.4 Funcionamiento

En la figura 3.2 se muestra el proceso de negociación de direcciones IP mediante DHCP.

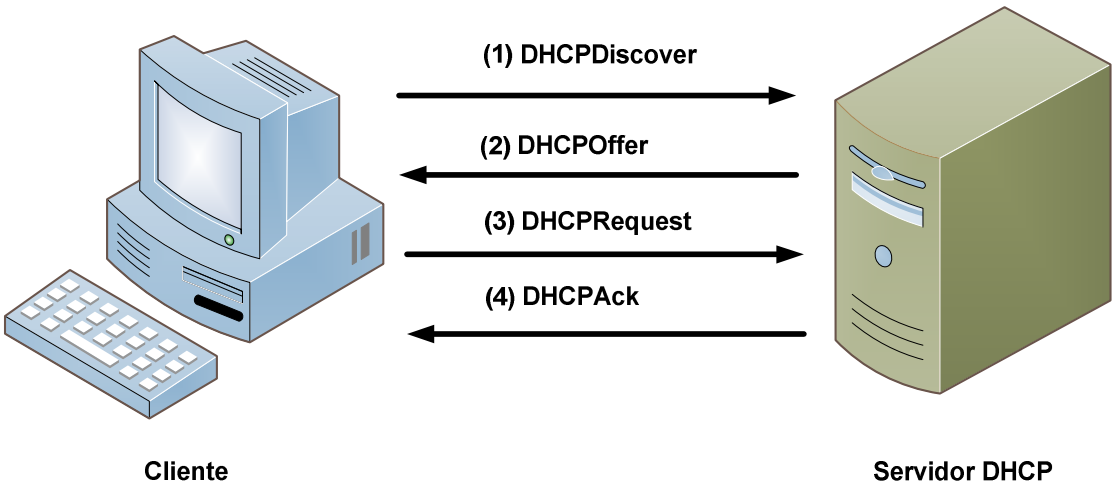


Figura 3.2 Proceso de negociación de DHCP

(1) Discover

El cliente envía un paquete DHCPDISCOVER de difusión broadcast en su segmento local de red para descubrir un servidor DHCP. Un paquete DHCPDISCOVER es un mensaje que los clientes envían la primera vez que intentan conectarse la red y solicitar información de direcciones IP de un servidor DHCP. En este paquete el cliente puede solicitar la dirección IP que desearía usar en el caso de ya haber contado con alguna que utilizó anteriormente; si el servidor DHCP la tiene disponible se la asigna, en caso contrario el servidor le ofrece alguna otra que esté tiene libre.

Si el cliente no recibe una oferta después de cuatro mensajes de solicitud (DHCPDISCOVER), se auto configura con una de sus direcciones IP previamente definidas en su sistema Operativo (APIPA).

APIPA (Automatic Private Internet Protocol Addresing), es un protocolo utilizado en sistemas operativos Windows en versiones 98 en adelante, este protocolo proporciona una configuración básica (Dirección IP y máscara de subred) a los equipos que no logran comunicarse con un servidor DHCP. Asigna una dirección IP privada de clase B (169.254.0.1-169.254.255.254) y la máscara de red típica de esta clase (255.255.0.0), con estos dos parámetros garantiza la comunicación entre los clientes en una subred sin un servidor DHCP disponible. La configuración que provee APIPA sólo permite un esquema de red local por lo que no proporciona salida a internet y los clientes configurados con este protocolo sólo podrán comunicarse con clientes que también tengan la configuración del protocolo (APIPA). El cliente DHCP continúa intentando localizar un servidor DHCP disponible cada cinco minutos. En cuanto un servidor DHCP pasa a estar disponible, los clientes reciben direcciones IP válidas.

Existe un protocolo similar para equipos Linux llamado “Avahi”, su funcionalidad principal es la exploración de la red para descubrir servicios, una de sus funciones es asignar automáticamente una dirección IP sin presencia de un servidor DHCP.

(2) Offer

Cada servidor que recibió el paquete DHCPDISCOVER determina si puede despachar la petición desde su propia base de datos, si puede atenderla envía un mensaje unicast (únicamente al cliente) DHCPOFFER para ofrecer una posible dirección IP al cliente y otras opciones de configuración como son puerta de enlace predeterminada, máscara de red, servidor de DNS entre otros. Si el servidor DHCP no puede atender la solicitud recibida entonces intenta reenviar la petición a otro servidor para que pueda ser despachada, esto depende de la configuración en los servidores. Los servidores tienen en sus tablas las direcciones ofertadas y sus tiempos de préstamo, evitando así ofrecerlas en posteriores mensajes de DHCPOFFER, de hecho cuentan con un mecanismo de identificación de red (ping), para que validen por completo si la dirección efectivamente esta libre antes de otorgarla.

(3) Request

Si el cliente considera que la oferta es adecuada entonces envía un mensaje broadcast DHCPREQUEST solicitando los parámetros que le ofrecieron para su configuración. El paquete DHCPREQUEST es el mensaje que el cliente envía de manera de difusión para que todos los servidores DHCP se enteren que ya aceptó una oferta, este paquete incluye la identificación del servidor que oferta y el cliente que aceptó, después, el resto de los servidores DHCP retiran sus ofertas y conservan sus direcciones IP para otras solicitudes de concesión IP. Este paquete se envía tanto para solicitar o renovar una dirección IP.

(4) Ack Reconocimiento

El paquete DHCPACK es un mensaje que el servidor DHCP envía al cliente como acuse de recibo y finalización del proceso asignación de dirección IP, por lo tanto el cliente inmediatamente puede hacer uso de la dirección concedida. Este mensaje contiene una concesión válida para la dirección IP y otros datos de configuración IP.

Una vez que el cliente DHCP recibe el reconocimiento, TCP/IP se inicializa con los datos de configuración IP suministrados por el servidor DHCP, así como dos tiempos T1 y T2 relacionados con el tiempo máximo en que la dirección IP es válida.
El cliente también enlaza el protocolo TCP/IP a los servicios de red y al adaptador de red, lo que permite la comunicación del cliente en la red.

Comúnmente a todo el proceso de asignación de direcciones IP se le conoce como "DORA" (**D**iscover, **O**ffert, **R**equest y **A**ck).

En la Figura 3.3 se muestra un fragmento de una captura del proceso de descubrimiento DHCP, donde se observa el proceso de negociación de una dirección IP, , donde un cliente marca Dell hace una petición para obtener una dirección IP y el servidor (170.70.1.7) le asigna al cliente la dirección IP 170.70.1.22.

Figura 3.3 Captura del proceso de descubrimiento DHCP

Si el cliente descubre que la dirección que le fue asignada ya está en uso en el segmento local, envía un mensaje **DHCPDECLINE** y el proceso se inicia de nuevo. Si el cliente recibe una **DHCPNACK** por parte del servidor DHCP después de recibir la DHCPREQUEST, entonces reinicia el proceso.
DHCPRELEASE es el proceso por el cual un cliente libera la dirección IP, ya sea porque ya no la necesita, para renovarla o para actualizar sus parámetros de configuración IP. El cliente renueva la configuración IP antes de la expiración del tiempo (cuando va a la mitad del tiempo total que se le asignó). Si el periodo de asignación expira y el cliente aun no ha renovado su configuración IP, pierde los parámetros de la configuración IP y comienza nuevamente el proceso de concesión. **DHCPRENEW** es el proceso por el cual el cliente renueva o actualiza sus datos de configuración IP en el servidor.

DHCPIFORM es un mensaje enviado por el cliente para el servidor solicitando otros parámetros de configuración. Aquí el cliente ya cuenta con una dirección IP pero solicita otros parámetros.

3.1.5 Agente de transmisión DHCP (dhcp-relay)

El agente de transmisión DHCP permite transmitir las peticiones DHCP desde una subred sin un servidor DHCP a uno o más servidores en otras subredes.

Como ya se ha mencionado, cuando un cliente quiere hacer una petición envía un paquete DHCPDISCOVER, que es un mensaje de difusión para descubrir un servidor DHCP que atienda su petición, este proceso se realiza en el mismo segmento de red local. Pero cuando el servidor DHCP y el cliente se encuentran en diferentes segmentos de red y además separados por un router, la solicitud del cliente no le llega al servidor, ya que los routers no retransmiten los mensajes broadcast [18].
Para retransmitir mensajes broadcast entre servidores DHCP que se encuentran en diferentes segmentos de red se utiliza el agente de transmisión mediante el comando **IP Helper-Address** en el router. La sintaxis es la siguiente:

"ip helper-address xxx.xxx.xxx.xxx"

Donde las “x” indican la dirección IP del servidor DHCP que se localiza en otra subred.

Un router puede ser configurado para aceptar mensajes broadcast para un servicio UDP y reenviarlo como unicast a una dirección IP específica (dirección IP de algún otro servidor que se localiza en distinto segmento de red). El comando ip helper-address reenvía los siguientes ocho servicios UDP:

- Servidor BOOTP/DHCP
- Cliente BOOTP/DHCP
- DNS
- TACACS (Sistema de control de acceso al controlador de acceso al terminal)
- Tiempo
- TFTP

- Nombre del servicio NetBIOS
- Servicio de datagrama NetBIOS

Cuando un cliente DHCP hace una petición, el agente de transmisión DHCP reenvía la petición a la lista de servidores DHCP específica, el agente de transmisión escuchará las peticiones de todas las interfaces a menos que el argumento *–i* se use para especificar sólo una o varias interfaces donde escuchar cómo se muestra en la Figura 3.4.

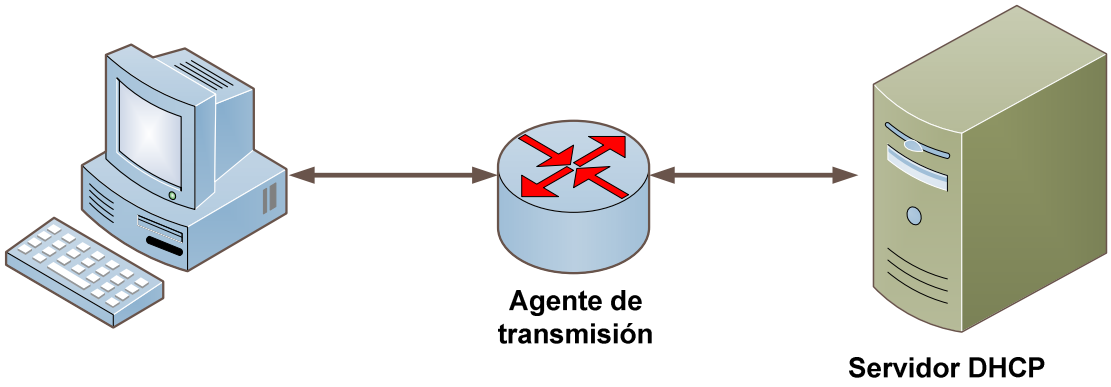


Figura 3.4 Retransmisión de mensajes DHCP a servidores en otros segmentos de red

Cuando el comando ip helper-address esta activado en el router y un cliente envía un paquete DHCPDISCOVER en su segmento de red local, el router toma el paquete y llena el campo GIADDR con la dirección IP del gateway de ese segmento de red, después el router envía el paquete y agrega la dirección IP del gateway por el cual será enviado el paquete para lograr llegar al servidor DHCP (esto es en la dirección origen), en la dirección destino coloca la dirección del servidor DHCP que fue dada en el comando ip helper-address. (Ver Figura 3.5)

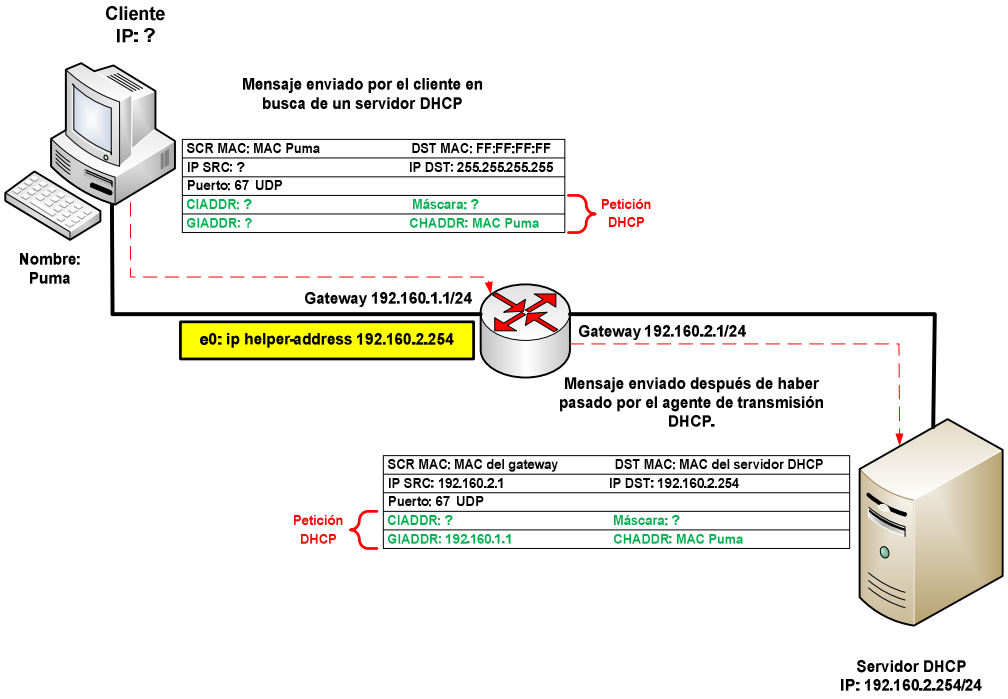


Figura 3.5 Proceso de petición de cliente DHCP utilizando un dhcp-relay

El servidor DHCP recibe el paquete de descubrimiento. El servidor utiliza el campo GIADDR para cotejar la lista del conjunto de direcciones y encontrar una que contenga la dirección del gateway asignada al valor en GIADDR. Este conjunto entonces se utiliza para brindar al cliente su dirección IP. (Ver Figura 3.6)

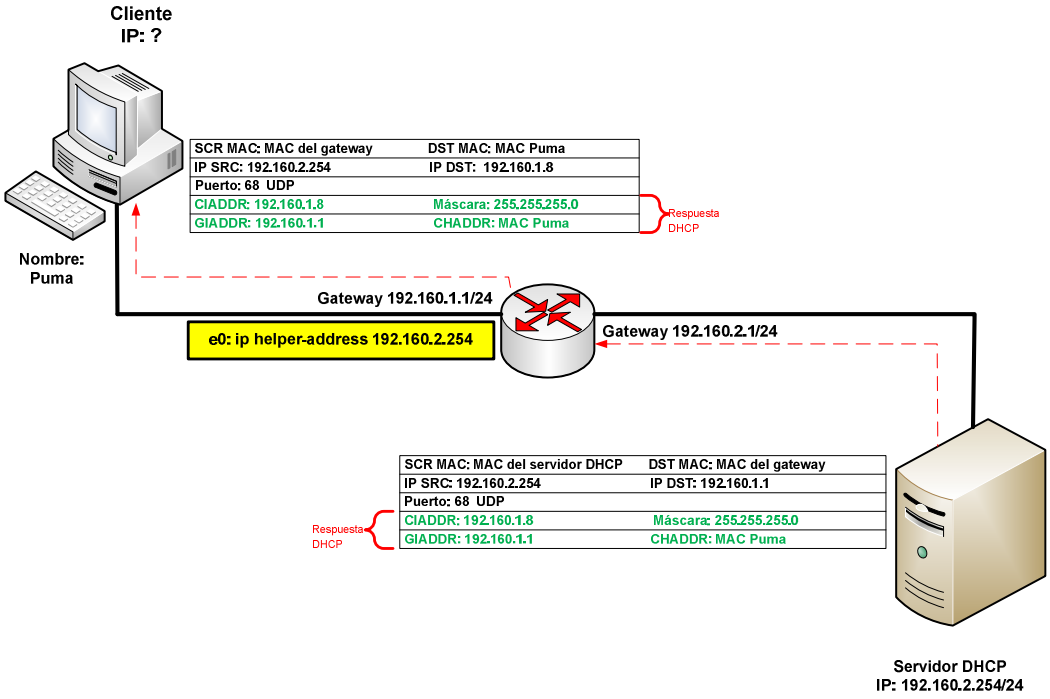


Figura 3.6 Proceso de respuesta del servidor DHCP utilizando un dhcp-relay.

3.1.6 Tipos de asignaciones de direcciones IP

- *Automática*

DHCP asigna una dirección IP de manera permanente a un cliente la primera vez que hace la solicitud al servidor DHCP y la retira hasta que el cliente la libera. Se suele utilizar cuando el número de clientes no varía demasiado.

- *Dinámica*

El servidor DHCP asigna la dirección IP al cliente durante un periodo limitado de tiempo, también llamado arrendamiento. Es el único método que permite la reutilización dinámica de las direcciones IP. El administrador de la red determina un rango de direcciones IP y cada equipo conectado a la red está configurado para solicitar su dirección IP al servidor DHCP cuando la tarjeta de red se inicializa. El procedimiento usa un concepto muy simple en un intervalo de tiempo controlable. Esto facilita la instalación de nuevas máquinas clientes a la red. Este esquema se da regularmente por periodos cortos de préstamo y es muy común entre los prestadores de servicio inalámbrico como cafés, aeropuertos, redes de universidades, entre otros.

- *Manual o estática*

El administrador asigna la dirección IP del cliente de manera manual en el archivo de configuración del servidor DHCP. Se suele utilizar cuando se quiere controlar la asignación de dirección IP a cada cliente, y evitar, también, que se conecten clientes no identificados. En resumen este esquema asocia una dirección MAC con una dirección IP específica.

3.2 Archivos necesarios en el servidor DHCP

Existen dos archivos básicos para que un servidorDHCP pueda funcionar de manera correcta, el primero es *dhcpd.conf* y el segundo es *dhcpd.leases*.

3.2.1 dhcpd.leases

El archivo *dhcpd.leases* es una archivo de texto que almacena la información sobre el alquiler de los clientes DHCP. Guarda de manera automática toda la información referente a los préstamos realizados, como son la fecha y hora del préstamo y el servidor que la proporcionó. Por lo general se encuentra en *"/var/lib/dhcp/dhcpd.leases*" pero puede llegar a cambiar dependiendo la distribución que se use.

Contiene todos los datos relacionados con el alquiler de la direcciones IP. Este archivo no debe modificarse manualmente, incluye información sobre los préstamos de las direcciones IP, a quién se ha asignado la dirección IP, la fecha inicial y final del arrendamiento, la dirección MAC de la tarjeta de red.

Este archivo se debe crear de vez en cuando para que su tamaño no sea excesivo. Antes de crearlo se debe crear un respaldo de este archivo, se recomienda guardarlo en la misma ubicación del archivo inicial con el nombre de *dhcpd.leases~.* Después de haber creado el respaldo, ahora sí se crea el archivo nuevo con el nombre original (dhcpd.leases). Si hubiese algún problema con el archivo dhcpd.leases, sólo hay que borrar el archivo y cambiar el nombre del archivo de respaldo (*dhcpd.leases*~) por dhcpd.leases y listo.

3.2.2 dhcpd.conf

El archivo *dhcpd.conf* es el que almacena toda la información de alquiler, los scopes, los "pooles" dinámicos, las políticas y parámetros adicionales que se le proporcionan a los clientes para que puedan conectarse a la red. Se pueden declarar opciones globales para todas las subredes o hacer declaraciones particulares para subredes o clientes específicos ver la tabla 3.1. Es en este archivo también donde se definen los servidores secundarios o redundantes del sistema en caso de utilizar el Failover protocol del cual se hablará en el capítulo 4.

El primer paso para configurar el servidor de DHCP será editar el fichero *dhcpd.conf* al cual le añadiremos la información de nuestra LAN. El archivo de configuración puede contener tabulaciones o líneas en blanco adicionales para facilitar el formato así como el símbolo numeral (#) para agregar comentarios y facilitar su comprensión. Las palabras clave no distinguen entre mayúsculas y minúsculas.
Se debe tener mucho cuidado en la manipulación de este archivo dado que algún error de sintaxis ocasiona fallas del servicio DHCP y sólo puede ser arrancado hasta que la falla sea corregida.
Por lo general se encuentra ubicado en */etc/dhcpd.conf* pero puede llegar a cambiar según la distribución.

Parámetros de configuración en el archivo dhcpd.conf

El archivo dhcpd.conf tiene una gran cantidad de parámetros de configuración. En la tabla 3.2 se muestran los parámetros y una breve descripción.

Tabla 3.2 Parámetros de configuración del archivo dhcpd.conf

Parámetro	Descripción
ignore/allow client-updates	Permite la actualización de las asignaciones de un cliente a requerimiento de éste, o bien las asignaciones se actualizan cuando el servidor así lo requiera (***ignore***).
shared-network redLocal	Parámetro que describe las subredes que compartirán la misma red física las cuales se especifican dentro de esta declaración.
subnet	Segmento de subred sobre el cual actuará el DHCP.
netmask	Máscara de red de la subred.
option routers	Parámetro que especifica mediante IP la ubicación del router.
option subnet-mask	Máscara de red de la subred.
option broadcast-address	Parámetro que especifica la IP de broadcast.
option domain-name	Parámetro que describe el nombre de dominio.
option domain-name-servers	Parámetro que especifica mediante IP la ubicación del DNS.
range	Rango sobre el cual el DHCP asignara direcciones IP.
default-lease-time	Parámetro que indica el tiempo entre cada nueva asignación de IP a los equipos.
max-lease-time	Parámetro que indica el tiempo de vigencia de la dirección IP para cada equipo.
host nombreDeLaMaquina	Parámetro que describe el nombre del equipo.
option host-name "nombreDeLaMaquina.tuDominio.com"	Parámetro que describe el nombre de la computadora y el nombre de dominio asociado a la misma.
hardware ethernet	Parámetro que describe la dirección MAC asociada a la tarjeta ethernet del equipo.
fixed-address	Parámetro que describe la dirección IP destinada a un equipo.
authoritative	Indica que este servidor DHCP es el servidor principal para este segmento de red. El cree que su configuración que tiene para proveerles a los clientes es la correcta, y si localiza algún cliente con mala configuración tratara de reasignar sus parámetros al cliente. Este parámetro puede ser global o sólo para ciertas subredes especificas.
not authoritative	La función de este parámetro es todo lo contraria del anterior, es decir, si el servidor detecta algún cliente mal configurado, lo dejara igual. No realizara ningún cambio en sus parámetros.
ddns-domainname <*nombre*>	Mediante el uso de este parámetro, se añadirá ***<nombre>*** al final del nombre de la máquina cliente, para formar un nombre de dominio totalmente calificado (*FQDN*).

ddns-hostname <nombre>	Por defecto, el servidor DHCP utiliza como nombre para la solicitud el nombre que el cliente tiene asignado a su máquina. Mediante este parámetro se asigna un nombre concreto a una máquina o a todas en general.
ddns-updates (on/off)	Activa la actualización DNS mediante los valores asignados por DHCP.
group	Inicia la declaración del grupo.
min-lease-time (duración)	Especifica la cantidad mínima de tiempo, en segundos que será mantenida la asignación de direcciones.
one-lease-per-client (on/off)	Cuando la opción se iguala a "on" y un cliente solicita una asignación de dirección (DHCPREQUEST), el servidor libera de forma automática cualquier otra asignación asociada a dicho cliente.
range <ip-menor ip-mayor>	En una declaración de subred, este parámetro define el rango de direcciones que serán asignadas. Pueden darse dos instrucciones range seguidas.
Server-name (nombre)	Nombre del servidor que será suministrado al cliente que solicita la asignación.
Server-identifier (IP)	Identifica la máquina donde se aloja el servidor de DHCP. Su uso se aplica cuando la máquina en cuestión tiene varias direcciones asignadas en una misma interfaz de red.

Asignación manual

Este tipo de asignación es cuando a un equipo con cierta dirección MAC requiere una dirección IP, entonces se le asigna una dirección IP específica.

La sintaxis que debe llevar dentro del archivo de configuración dhcpd.conf es:

```
host equipo1{
      option hostname "equipo1.dhcp.com";
      hardware ethernet 00:2a:32:f2:ba:c1;
      fixed-address 192.145.4.5;
}

host equipo2{
      option hostname equipo2.dhcp.com";
      hardware ethernet 00:ac:2f:e3:4c:14;
      fixed-address 192.145.4.8;
}
```

En el ejemplo anterior, al host llamado "equipo1" con dirección física 00:2a:32:f2:ba:c1 se le asocia la dirección 192.145.4.5, y siempre que solicite una dirección IP se le asignará la que tiene asociada.

Asignación automática

En la asignación automática sólo se declara un rango de direcciones IP sin declarar un tiempo de préstamo específico. El servidor las va asignando conforme sus clientes las requieran. A continuación se muestra la sintaxis de declaración como debe ir en el archivo de configuración dhcpd.conf.

```
subnet 192.145.5.0 netmask 255.255.255.0{
      option routers 172.135.5.5;
      option subnetmask 255.255.255.0;
      option broadcastaddress 172.135.5.255;
      option domainname "dhcp.com.mx";
      option domainnameservers 172.135.5.7;
      range 192.145.4.5  192.145.4.50;
}
```

En el ejemplo de asignación automática anterior, cualquier equipo que llegue a la red y se conecte obtendrá una dirección IP dentro del rango 192.145.4.5 a 192.145.4.50 por tiempo ilimitado y sólo será liberada cuando el cliente ya no la requiere y de un DHCPRELEASE.

Asignación dinámica

En este tipo de asignación se declara de igual manera un rango de direcciones IP, pero en este caso sí se declara un tiempo de préstamo específico. La siguiente sintaxis muestra el ejemplo de un servidor DHCP que asigna direcciones IP aleatorias dentro del rango de 192.145.5.15 al 192.145.5.100, las cuales serán renovadas cada cierto tiempo, asignando de nuevo direcciones IP aleatorias dentro del rango dado.

```
subnet 192.145.5.0
      netmask 255.255.255.0{
      option routers 192.145.5.105;
      option subnetmask 255.255.255.0;
      option broadcastaddress 192.145.5 .255;
      option domainname "dhcp.com.mx";
      range 192.145.5.15   192.145.5.100;
      defaultleasetime 21600;
      maxleasetime 43200;
}
```

El parámetro default-lease-time específica la cantidad de tiempo (en segundos) que la dirección será asignada. Al término de este tiempo trata de realizar una renovación, si

el servidor puede la renueva en caso contario el cliente sólo se la queda por el tiempo indicado en max-lease-time, que es el tiempo de vigencia de la dirección IP.

Declaración de subred

Para este tipo de configuración, se debe incluir la declaración *subnet* para cada subred perteneciente a la red, si no, el servidor DHCP no arrancará. A continuación se muestra la sintaxis.

```
subnet 192.145.3.0 netmask 255.255.255.0 {
      option routers 192.145.3.1;
      option subnetmask 255.255.255.0;
      option domainnameservers 192.145.3.1;
      option timeoffset 3600;
      range 192.145.3.1  192.145.3.100;
}
```

Basándose en el ejemplo anterior, esta declaración informa al servidor que la subred 192.145.3.0 existe y por lo tanto tiene el conocimiento necesario para responder cuando una dirección IP es inválida. Por ejemplo, si un cliente envía un mensaje DHCPREQUEST sugiriendo una dirección 192.3.23.1 el servidor DHCP responderá un DHCPNACK informando que la dirección IP solicitada es invalida por que no pertenece a ninguna de las subredes que tiene declaradas. También observamos opciones globales para cada cliente del servidor DHCP en la subred, así como el parámetro "range", este parámetro da un rango de direcciones que pueden ser asignadas por el servidor DHCP a los clientes, siempre y cuando no estén ocupadas.

Declaración de red compartida

Todas las subredes que comparten la misma red física deben especificarse dentro de una declaración *shared-network*. Se utiliza cuando un segmento de red es compartido por dos o más subredes.

Los parámetros dentro de *shared-network* pero fuera de las declaraciones *subnet* se consideran parámetros globales. El nombre de *shared-network* debe ser el título descriptivo de la red, como, por ejemplo redPruebas. También el nombre puede ser una dirección IP.

```
Sharednetwork redPruebas {

      Subnet 192.145.1.2 netmask 255.255.255.0 {
      #parameters for subnet
      range 192.145.1.3  192.145.1.650;
      }
      subnet 192.145.1.51 netmask 255.255.255.0 {
      #parameters for subnet
      range 192.145.1.52  192.145.1.100;
      }
           }
```

En esta declaración se definen dos subredes 192.145.1.2 y 192.145.1.51, las cuales comparten un mismo segmento de red.

Declaración de grupo

El parámetro *group* puede utilizarse para aplicar parámetros globales a un grupo de declaraciones. Puede agrupar redes compartidas, subredes, hosts u otros grupos.

```
group {
   option routers                  192.15.1.254;
   option subnet_mask              255.255.255.0;

   option domain_name              "pruebas.com";
   option domain_name_servers       192.15.1.1;

   option time_offset              _1600;

   host equipo1 {
      option host_name "equipo1.pruebas.com";
      hardware ethernet 00:5A:F8:1E:9B:1C;
      fixed_address 192.15.1.4;
   }

   host equipo2 {
      option host_name "equipo2.pruebas.com";
      hardware ethernet 00:F1:A3:C4:B3:0A;
      fixed_address 192.15.1.6;
   }
}
```

Para esta declaración, los parámetros globales aplicados al grupo conformado por equipo1 y equipo2 son:

- option routers
- option subnet_mask
- option domain_name

- option domain_name_servers
- option time_offset

3.3 Desempeño DHCP

Para comprobar el buen desempeño del servicio DHCP, se realizaron pruebas de laboratorio en un equipo con capacidad de 2Gb en memoria RAM a una velocidad de 1.6 GHz. Se realizaron pruebas con distintas cargas, así como para distintos tiempos y poder conocer el número de clientes que puede atender en determinado tiempo. Para llevar a cabo este experimento se utilizó la herramienta *dhcperf*.

Dhcperf permite ejecutar pruebas de rendimiento del servicio, proporciona un medio para simular el comportamiento del servicio bajo diferentes cargas y así comprobar el rendimiento del mismo.

Para ello, a continuación se definen los siguientes términos:

a) Carga: número de transacciones por segundo.
b) Tiempo de prueba: intervalo de tiempo en el que se simula el funcionamiento del servicio.

En las tablas 3.3 a 3.7 se muestran los resultados y en la figura 3.7 la gráfica de dichos resultados, para poder observar el rendimiento del servicio a diferentes cargas

Tabla 3.3 Rendimiento DHCP con carga al 100% *Tabla 3.4 Rendimiento DHCP con carga al 80%*

carga= 1000 lease/segundo (100%)	
usuarios atendidos	tiempo
91	30
129	60
99	90
-	120
-	180

carga= 800 lease/segundo (80%)	
usuarios atendidos	tiempo
45	30
89	60
153	90
152	120
209	180

Tabla 3.5 Rendimiento DHCP con carga al 50%

carga= 500 lease/segundo (50%)	
usuarios atendidos	tiempo
582	30
1162	60
1643	90
2225	120
3209	180

Tabla 3.6 Rendimiento DHCP con carga al 30%

carga= 300 lease/segundo (30%)	
usuarios atendidos	tiempo
1609	30
3202	60
4868	90
6470	120
9821	180

Tabla 3.7 Rendimiento DHCP con carga al 20%

carga= 200 lease/segundo (20%)	
usuarios atendidos	tiempo
2083	30
4225	60
6109	90
8590	120
12916	180

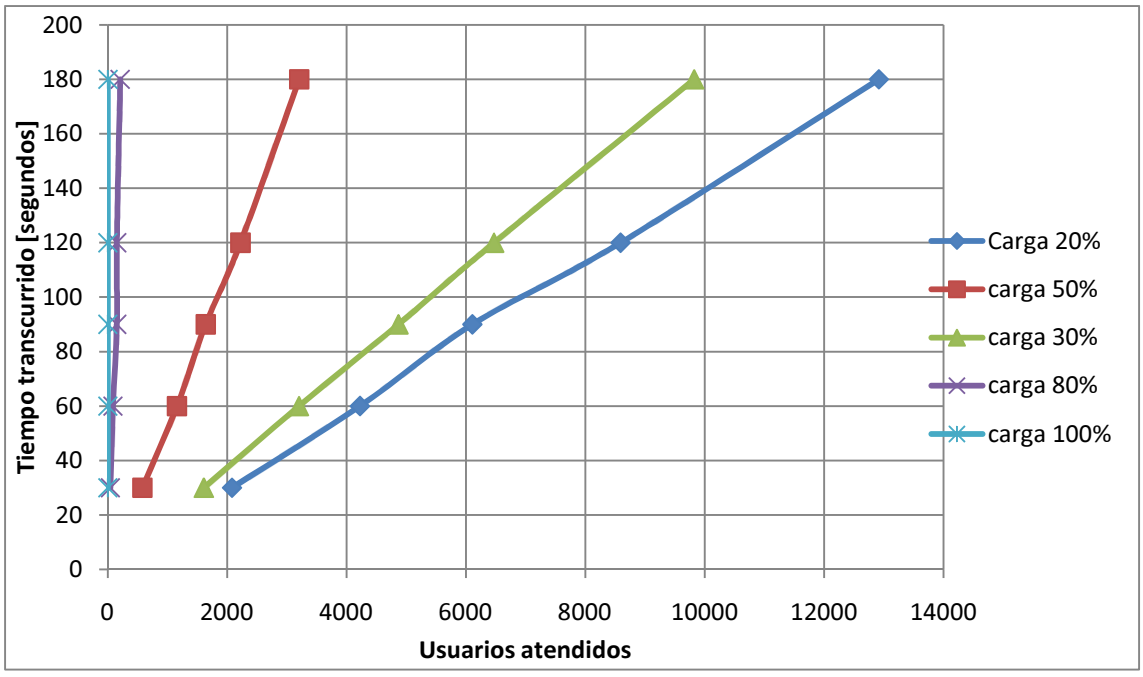


Figura 3.7 Gráfica de desempeño del servicio DHCP.

3.4 DHCP en Webmin

El servicio ISC DHCP se debe instalar en un sistema operativo Linux, en este caso de estudio, las implementaciones se efectuaron en sistemas operativos CentOS y Red Hat, de hecho las primeras pruebas se llevaron a cabo en un Linux SUSE, por lo que seguramente otras distribuciones podrán soportar el ISC DHCP. Hasta este punto la instalación del servicio DHCP es funcional pero no está completa (para ver manual de instalación de Webmin dirigirse al apéndice A1), debido a que parte de las necesidades que se especificaron en el capítulo 1 no están presentes como es el caso de una consola gráfica que permita configurar, manipular y visualizar el estado del servidor, para cumplir con los requerimientos se utiliza Webmin.

Para la instalación de Webmin se requieren los siguientes paquetes:

- Webmin-1.470.noarch.rpm

Para el funcionamiento óptimo del módulo DHCP en Webmin, se deben tener instalados los siguientes paquetes:

- Netools -1.060.1.wbm
- Dhcp-server
- Dhcp-tools
- Openssl-devel
- perl-OPENSSL
- perl-Net_SSLeay
- gcc-java criptix
- kdeblindings java
- puretls

Y se debe desinstalar Dhcp-client del sistema operativo, ya que por tratarse de un servidor DHCP su dirección IP tiene que ser fija.

Una vez instalado Webmin y el servicio DHCP, se configura el módulo de DHCP, para esto dar clic en la pestaña configuración de módulo como en la figura 3.8.

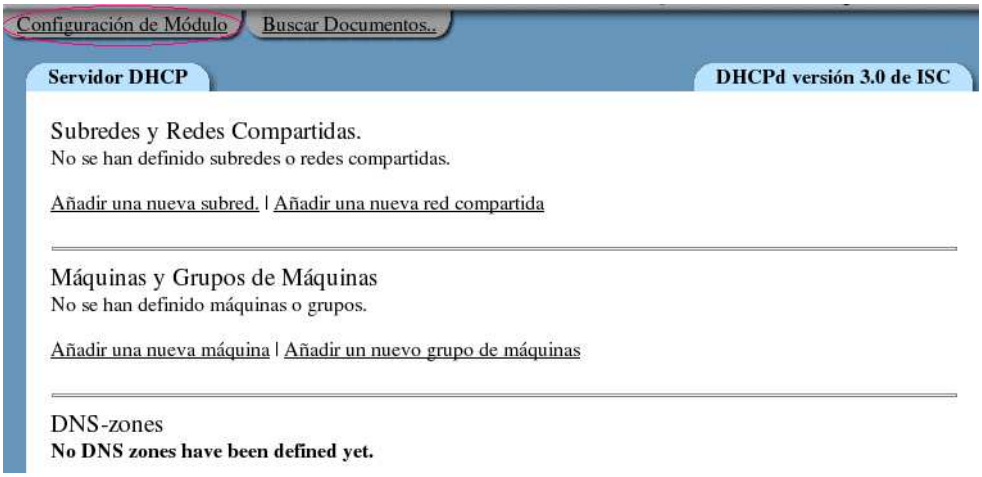


Figura 3.8 Configuración de archivo dhcpd.conf de manera gráfica en Webmin.

Anteriormente se dio la sintaxis para configurar los parámetros del archivo *dhcpd.conf* de manera manual (línea de comandos o CLI), pero Webmin cuenta con una interfaz gráfica que permite añadir:

- una nueva subred
- una nueva red compartida
- un nueva máquina
- un nuevo grupo de máquinas.

De manera gráfica y más rápida, ya que se encarga internamente de crear la sintaxis de lo que añade y lo crea dentro el archivo dhcpd.conf.

Comandos para el servicio DHCP

- *Mediante Webmin*

  Se puede iniciar, detener o reiniciar el servicio dhcpd desde Webmin.

- *Mediante línea de comandos*

  Se puede realizar de manera manual utilizando una terminal del sistema. Los comandos son los siguientes:

  Iniciar el servicio

  ```
  service dhcpd start
  ```

  Detener el servicio

  ```
  service dhcpd stop
  ```

  Reiniciar el servicio

  ```
  service dhcpd restart
  ```

3.5 Módulo de búsqueda de direcciones QuiCho v1.0

Como se mencionó anteriormente, la implementación del servicio DHCP de ISC es el motor que brindará servicio de DHCP, pero como tal no proporciona una interfaz gráfica para administrarlo, por lo que sugerimos utilizar Webmin que nos permite configurar de forma gráfica todas las opciones del ISC DHCP y guardarlas en el archivo de configuración dhcpd.conf. Webmin proporciona un entorno gráfico donde podemos administrar todas las opciones del servicio, así como conocer el estado actual de asignación de direcciones dinámicas, no así con las direcciones estáticas, por lo que se consideró necesario crear un módulo Webmin que permitiera conocer el estado de asignación de direcciones basado en los logs del sistema y el archivo dhcpd.leases.

Esta necesidad llevó a desarrollar un módulo de Webmin programado en Perl que se llamó QuiCho que es un módulo especialmente creado para cubrir la necesidad de conocer el estado de las direcciones IP tanto estáticas como dinámicas ya que el ISC DHCP ni el módulo de Webmin tienen soporte para conocer el estado de asignación de las direcciones estáticas por lo que fue creado desde cero ya que no hay otro que se pueda encontrar en la red. Usa funciones del API proporcionado por Webmin para el desarrollo de módulos adicionales. QuiCho permite realizar búsqueda de direcciones tanto en logs (Messages) como en archivos de control de arrendamiento (dhcpd.leases) del servicio de DHCP. Ofrece dos modos principales de búsqueda que son el de búsqueda por dirección IP o MAC (completa) y otra por subredes.

Este módulo de Webmin permite realizar búsqueda de direcciones tanto en logs como en archivos de control de arrendamiento (leases) del servicio de DHCP. Ofrece dos modos principales de búsqueda que son el de búsqueda por dirección IP o MAC (completa) y otra por subredes.

3.5.1 Búsqueda por dirección IP o MAC

a) Para realizar una búsqueda especifica por dirección IP o MAC, la tecleamos en el campo **Dirección IP o MAC.** Ver Figura 3.9.

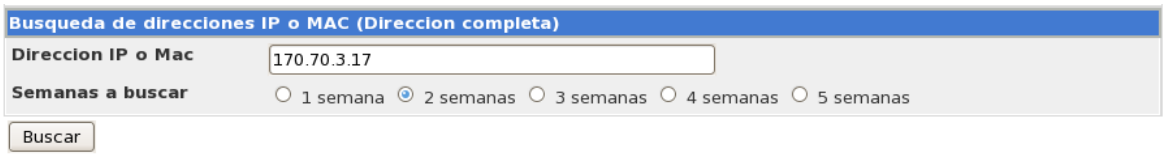


Figura 3.9 Búsqueda de direcciones IP o MAC

b) Seleccionamos la profundidad de la búsqueda, ósea con cuantas semanas anteriores a la actual se va a realizar la búsqueda, presionar el botón buscar.

La función del Radio Button correspondiente a "Semanas a buscar" lo que hace es indicar cuales archivos de logs va a incluir en la búsqueda, por ejemplo cuando se seleccionan 3 semanas el sistema buscará en los logs:

/var/log/messages

/var/log/messages.1

/var/log/messages.2

Que son los correspondientes a las últimas dos semanas, más la semana actual, que corresponde al archivo *messages.* Cada uno de estos *logs* guarda los eventos correspondientes a una semana anterior, se van numerando automáticamente, de menor a mayor respecto a su antigüedad.

Hasta éste momento el sistema no sabe si se ésta IP pertenece a una subred estática (ETB) o dinámica (Teléfonos) por lo que va a realizar la búsqueda de ésta en dos archivos, Los Logs y el archivo /var/lib/dhcp.leases,

c) Resultados de la búsqueda

En la ventana de resultados de la búsqueda se muestran dos secciones, la primera:

Direcciones Estáticas muestra las coincidencias de la dirección buscada y los logs, algunos datos adicionales como el inventario los obtiene del archivo /etc/dhcp.conf.

Está compuesta de 8 columnas, una es el número consecutivo la dirección IP, MAC, día de inicio, que es el día en que el servidor dio un ACK al cliente, hora de inicio, y Fecha y Hora Vence, que simplemente son la fecha en que se dio la dirección más un mes, estas no se pueden obtener de algún otro lado ya que el servidor en el caso de las

estáticas no tiene control sobre la fecha de expiración, por lo que ésta se calcula haciendo una suma entre el tiempo definido en el archivo de configuración dhcp.conf y el día en que fue asignada.

Direcciones Dinámicas muestra las coincidencias de la dirección buscada en el archivo dhcpd.leases

En ésta sección se mostraran las direcciones encontradas en el archivo de control de asignaciones dinámicas dhcpd.leases pero en este caso si podemos saber la hora y fecha de vencimiento de hora exacta por que en el caso de las direcciones dinámicas si se proporciona. Ver Figura 3.10.


Indice de Módulo

Resultado de busqueda

Consulta 2 semanas

Subredes Banxico						
IP	Direccion Mac	Inventario	Fecha de inicio	Hora de inicio	Fecha vence	Hora vence
Direcciones Estaticas:						
1 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 8	09:55:30	Apr 8	09:55:30
2 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 9	10:19:26	Apr 9	10:19:26
3 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 10	09:28:44	Apr 10	09:28:44
4 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 11	09:45:53	Apr 11	09:45:53
5 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 12	09:40:13	Apr 12	09:40:13
6 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 12	10:01:59	Apr 12	10:01:59
7 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 12	11:40:19	Apr 12	11:40:19
8 170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 17	09:34:04	Apr 17	09:34:04
Direcciones Dinamicas:						

Regresar a website list

Figura 3.10 Resultado de búsqueda de direcciones IP.

d) Mostrar de forma detallada la actividad de esa dirección IP o MAC (DORA)

Para saber con mayor exactitud que ha hecho la dirección a lo largo del tiempo se puede hacer una consulta más detallada dando clic en las ligas mostradas en los resultados de la consulta. Ver Figura 3.11.

Figura 3.11 Imagen de información de una dirección IP específica.

Al dar clic en una de las ligas, mostrará información del DORA correspondiente a la dirección MAC o IP según el caso. Véase Figura 3.12.

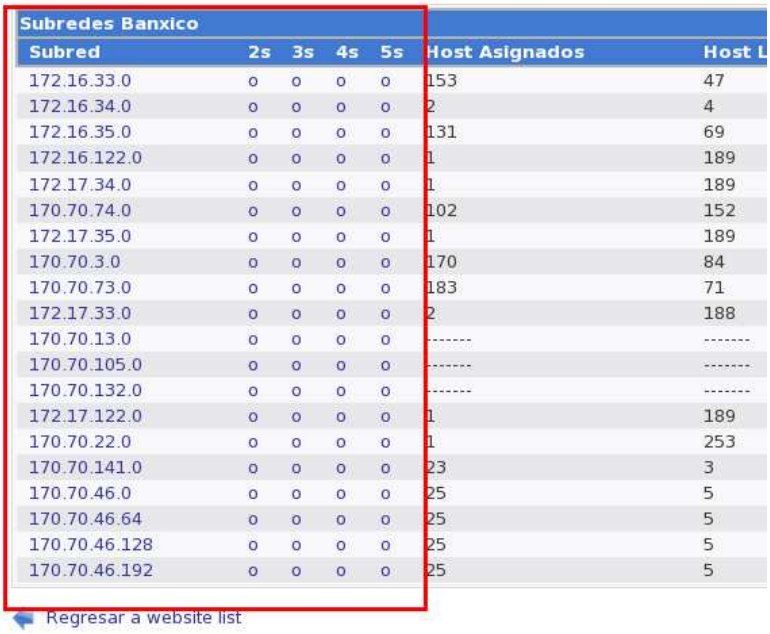
Figura 3.12 Resultados de la búsqueda detallada de una dirección IP específica.

NOTA: El resultado de la consulta del DORA está en función de las semanas seleccionadas inicialmente en la búsqueda, si se seleccionaron 2 semanas para la búsqueda el histórico sólo mostrará las dos últimas semanas en el resultado.

3.5.2 Búsqueda por Subredes

La búsqueda por subredes nos permite saber el estado de cada una de las direcciones, así como la última vez que dicho equipo renovó su dirección.

a) Al igual que la búsqueda individual, aquí es necesario seleccionar la profundidad de la búsqueda para indicarle con cuantas semanas de anterioridad se realizara la consulta, para esto damos clic en la liga correspondiente a la semanas a consultar. Ver Figura 3.13.



Subredes Banxico

Subred	2s	3s	4s	5s	Host Asignados	Host L
172.16.33.0	o	o	o	o	153	47
172.16.34.0	o	o	o	o	2	4
172.16.35.0	o	o	o	o	131	69
172.16.122.0	o	o	o	o	1	189
172.17.34.0	o	o	o	o	1	189
170.70.74.0	o	o	o	o	102	152
172.17.35.0	o	o	o	o	1	189
170.70.3.0	o	o	o	o	170	84
170.70.73.0	o	o	o	o	183	71
172.17.33.0	o	o	o	o	2	188
170.70.13.0	o	o	o	o	-------	-------
170.70.105.0	o	o	o	o	-------	-------
170.70.132.0	o	o	o	o	-------	-------
172.17.122.0	o	o	o	o	1	189
170.70.22.0	o	o	o	o	1	253
170.70.141.0	o	o	o	o	23	3
170.70.46.0	o	o	o	o	25	5
170.70.46.64	o	o	o	o	25	5
170.70.46.128	o	o	o	o	25	5
170.70.46.192	o	o	o	o	25	5

Regresar a website list

Figura 3.13 Resultado de búsqueda por subredes

Para realizar búsquedas en la semana actual sólo basta con dar clic en la dirección de la subred, que son las que aparecen en la columna Subred de la Figura 3.12. Para realizar una búsqueda en 2 o más semanas, selecciona el círculo de la columna y subred que se quiera consultar de acuerdo con la columna deseada:

2s = 2 semanas de antigüedad

3s = 3 semanas de antigüedad

…

5s = 5 semanas de antigüedad

b) El resultado de la consulta será mostrado en una página como la siguiente (Véase Figura 3.14.)

Indice de Módulo

Resultado de busqueda

Consulta 1 semanas

Subredes Banxico							
	IP	Direccion Mac	Inventario	Fecha de inicio	Hora de inicio	Fecha vence	Hora vence
Estaticas:							
	170.70.3.11	00:18:8b:67:62:8a	ADI07002715	---- leo: 4520 ACK	----	----	----
	170.70.3.14	00:1a:a0:5c:5f:f5	ADI07001760	Mar 17 367	09:27:50	Apr 17	09:27:50
	170.70.3.15	00:07:4d:2c:db:e0	ZBR2939872	---- leo: 4520 ACK	----	----	----
	170.70.3.16	00:18:8b:66:ee:35	ADI07002190	Mar 17 105	10:26:46	Apr 17	10:26:46
	170.70.3.17	00:19:b9:25:77:d6	ADI07005977	Mar 17 342	09:34:04	Apr 17	09:34:04
	170.70.3.18	00:12:3f:ed:b6:42	ADI05000603	---- leo: 4520 ACK	----	----	----
	170.70.3.19	00:18:8b:67:5a:ce	ADI07002056	Mar 17 353	09:32:25	Apr 17	09:32:25
	170.70.3.20	00:18:8b:67:58:e8	ADI07002085	Mar 17 331	09:35:23	Apr 17	09:35:23
	170.70.3.21	00:18:8b:65:b4:69	ADI07000570	Mar 17 388	09:23:50	Apr 17	09:23:50
	170.70.3.22	00:18:8b:67:79:dc	ADI07002676	Mar 17 336	09:34:47	Apr 17	09:34:47
	170.70.3.23	00:18:8b:67:74:88	ADI07002507	---- leo: 4520 ACK	----	----	----
	170.70.3.24	00:18:8b:67:6d:c7	ADI07001219	Mar 16 1469	13:14:00	Apr 16	13:14:00
	170.70.3.25	00:18:8b:64:8f:67	ADI07002139	Mar 17 513	09:02:58	Apr 17	09:02:58
	170.70.3.26	00:18:8b:67:79:ab	ADI07002718	Mar 17 267	09:43:46	Apr 17	09:43:46
	170.70.3.27	00:18:8b:67:64:09	ADI07001630	Mar 16 1340	14:43:57	Apr 16	14:43:57
	170.70.3.28	00:18:8b:67:70:be	ADI07002313	Mar 17 567	08:44:10	Apr 17	08:44:10
	170.70.3.30	00:18:8b:66:df:0f	ADI07002622	---- leo: 4520 ACK	----	----	----
	170.70.3.32	00:18:8b:66:ff:0b	ADI07001144	Mar 17 591	08:37:35	Apr 17	08:37:35
	170.70.3.33	00:18:8b:67:72:10	ADI07002117	Mar 17 86	10:49:47	Apr 17	10:49:47
	170.70.3.34	00:18:8b:66:de:10	ADI07002583	---- leo: 4520 ACK	----	----	----
	170.70.3.35	00:18:8b:67:75:11	ADI07002763	Mar 17 383	09:24:22	Apr 17	09:24:22
	170.70.3.36	00:18:8b:66:e6:87	ADI07002822	---- leo: 4520 ACK	----	----	----
	170.70.3.37	00:18:8b:67:73:5a	ADI07002026	Mar 17 189	10:00:03	Apr 17	10:00:03

Figura 3.14 Resultado de búsquedas por semanas

Al igual que en las búsquedas individuales genera una tabla con el estado actual de las direcciones tanto dinámicas como estáticas en el cual se permite profundizar en las búsquedas haciendo clic en la liga de dirección IP solamente, éste no genera una liga para la dirección MAC ya que se sugiere en caso de requerirse una búsqueda más profunda usar la sección de búsqueda individual. Véase Figura 3.13.

NOTA: en la columna Fecha de Inicio aparece un numero después del día de inicio, éste corresponde al número de línea en que se encuentra el último ACK de la dirección, en caso de que no exista ninguna coincidencia escribe: "leo: [Ultima línea leída] ACK".

3.6 DHCPv6

Con el crecimiento exponencial y sostenido de usuarios de internet ha sido cada vez más difícil seguir usando el esquema inicial con el que se contaba en principios del

internet, este esquema es el que utilizamos actualmente y que lleva por nombre IPv4, actualmente resulta limitado por el número de direcciones que proporciona por lo cual se ha propuesto un nuevo protocolo llamada IPv6 (Internet Protocolo versión 6) o IPng.

3.6.1 El protocolo IP versión 4

La versión del protocolo IP se comenzó a implementar en 1983 y forma una parte muy importante del proyecto de internet IP versión 4 que usa un esquema de direccionamiento de 4 bytes de 8 bits cada uno (32 en total) con lo cual se tienen 2^{32} = 4,394,967,296 direcciones IP, lo que a primera vista parece un número muy grande, a menos para la época cuando se diseñó, IPv4 se consideraba un número suficiente para proporcionarles a los clientes una dirección IP; además, este número no es del todo real considerando que este obedece a una estructura jerárquica, lo que provoca que se desperdicie un número importante de direcciones.

No se estimó que internet creciera tan rápido y a tal grado, varias de las características no estaban planeadas, tales como la seguridad, la capacidad de enrutamiento y el tiempo que éste utiliza. Debido a que es muy variable por que no existe la reservación de recursos, cada datagrama tiene que competir con los demás y esto provoca que el tráfico sea variable y pueda congestionarse. Para dar solución a las deficiencias que mostraba este protocolo se buscaron mecanismos para mejorar lo logrado anteriormente, tras varios años de trabajo, en 1994 se lograron crear nuevas fórmulas para poder soportar la creciente demanda de direcciones, por lo cual nace SIPP (Simple IP Plus), quien cambiaría el tamaño de direcciones de 32 a 128 bits y se denominó como IPng (IP next generation). Siguió evolucionando hasta 1995, cuando fue rebautizado con el nombre IPv6.

Hoy en día el número de equipos de red ha aumentado significativamente, no sólo ocupado por los equipos de cómputo como en un principio, sino que ahora también entran los usuarios de redes móviles que requieren una dirección pública propia, para los cuales se siguen utilizando mecanismos como NAT (Traducción de direcciones de red).

El cambio entre ambas tecnologías no es necesariamente sustituir el protocolo ipv4 con la versión ipv6, sino que ambas tecnologías pueden coexistir de forma que puedan trabajar a la par si es que los dispositivos así lo necesitan.

3.6.2 El protocolo IPv6

Como se mencionó anteriormente una de las características principales de IPv6 es la capacidad de direccionamiento que es de 2^{128}, lo que es un número casi inagotable de direcciones IP. A diferencia del protocolo ipv4 las direcciones en ipv6 son asignadas a las interfaces en lugar de los nodos, a sabiendas que un nodo puede contener más de una interfaz, y a su vez se pude asignar más de una dirección IP a una interfaz.

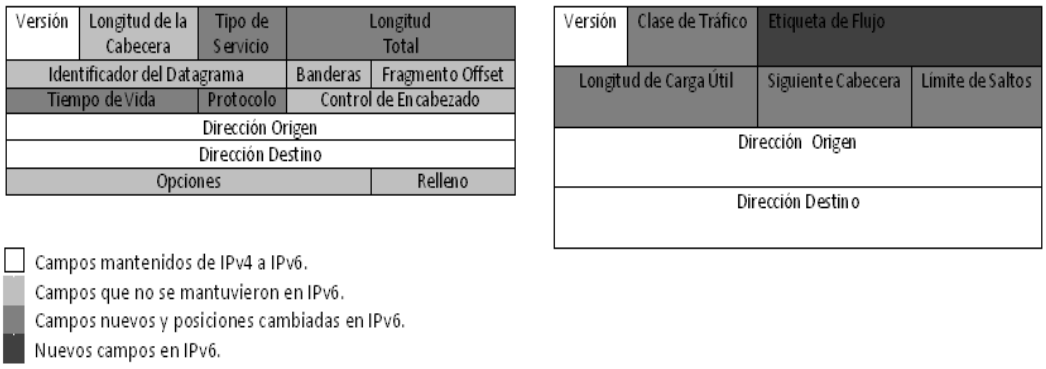


Figura 3.15. Comparación del encabezado ipv4 con ipv6

El encabezado en IPv4 consta de 20 octetos y en la versión 6 de 40 octetos. Otro de los cambios importantes en ipv6 es el formato de las cabeceras de los paquetes, ya que se eliminan algunos y se añaden otros (ver figura 3.15). El formato del encabezado de IPv6 convierte en opcionales algunos campos de la cabecera ipv4, a pesar de estos cambios la nueva cabecera consume el mínimo de ancho de banda posible, aunque las direcciones IPv6 son cuatro veces mayores a las de IPv4, la cabecera ocupa tan sólo el doble a comparación de su predecesora [19].

Configuraciones automáticas

Una de las características que no existía en IPv4 era la autoconfiguración de direcciones, en IPv6 existe un mecanismo de configuración automática a través de un

protocolo llamado ND (Neighbor Discovery o descubrimiento de vecinos) creado por cisco. Básicamente lo que hace es darle la capacidad a un host para generar automáticamente su dirección IP, esto lo lleva acabo de la siguiente forma:

- El host crea una dirección del tipo Linklocal (del cual se hablará posteriormente) para cada interfaz la cual no involucra un Router en el vínculo.
- Verifica que exista esa dirección se exclusiva en el vinculo.
- Determina si las direcciones deben obtenerse a través de un mecanismo, con estado (*stateful*) o sin estado (*staless)*, para lo cual es necesario la intervención del router.

Una ventaja de esta configuración es que no necesita configuración manual del host, esto permite que un host genere sus propias direcciones. Para generar las direcciones, el mecanismo sin estado usa la información local y la no local mostrada por los Router.

Descripción general de tipos de direcciones IPv6

Las direcciones IPv6 se clasifican en tres tipos. Ver Figura 3.16.

- Unidifusión (unicast)

Identifican a una sola interfaz. Cuando un paquete es enviado a una dirección *unicast*, este es únicamente entregado a la interfaz que tenga dicha dirección.

- Multidifusión

Ésta identifica un grupo de interfaces, generalmente en nodos distintos. Los paquetes que se envían a una dirección multidifusión se dirigen a todos los miembros de un grupo de multidifusión. El paquete es enviado a todos y cada uno de los miembros individualmente.

- Anycast

Identifican a un conjunto de interfaces de red. El paquete se enviará a cualquier interfaz que forme parte del conjunto. El paquete enviado a una dirección *anycast* será entregado en una de las interfaces (cualquiera) identificadas con dicha dirección, por lo general es la más cercana dependiendo de los protocolos de ruteo.

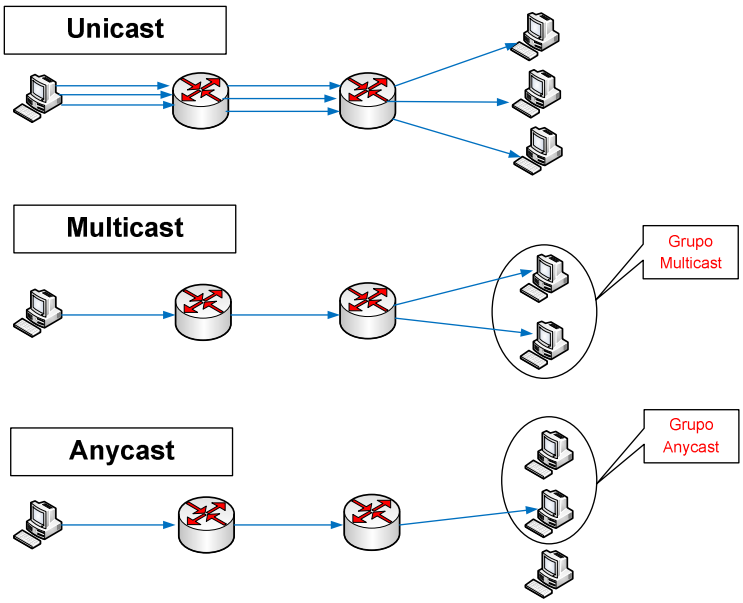


Figura 3.16. Tipos de direcciones IPv6

Partes que componen una dirección IPv6

Una dirección IPv6 tiene un tamaño de 128 bits y se compone de ocho campos de 16 bits, cada uno de ellos unido por dos puntos. Cada campo debe contener un número hexadecimal, a diferencia de la notación decimal con puntos de las direcciones IPv4. En la figura 3.17, las equis representan números hexadecimales.

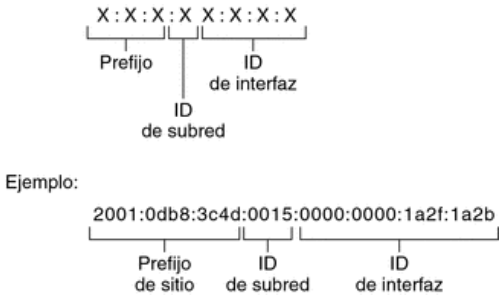


Figura 3.17. Campos de una dirección IPv6

Los tres campos que están más a la izquierda (48 bits) contienen el prefijo de sitio. El prefijo describe la topología pública que el ISP o el RIR (Regional Internet Registry, Registro Regional de Internet) suelen asignar al sitio.

El campo siguiente lo ocupa el ID de subred de 16 bits que el administrador asigna al sitio. El ID de subred describe la topología privada, denominada también topología del sitio, porque es interna del sitio.

Los cuatro campos situados más a la derecha (64 bits) contienen el ID de interfaz, también denominado token. El ID de interfaz se configura automáticamente desde la dirección MAC de interfaz o manualmente en formato EUI-64.

3.6.3 DHCPv6

DHCPv6 es la siguiente generación de DHCPv4, como ya se ha mencionado IPv6 surgió para cubrir las necesidades de IPv4. El objetivo de este nuevo protocolo es la misma que la del protocolo en su versión cuatro, proveer una forma de automatizar y manejar los parámetros de red de equipos de cómputo, impresoras, teléfonos celulares entre otros. Tiene algunas diferencias en cuanto a su versión anterior en las cuales no entraremos a detalle, ya que el propósito de este proyecto es trabajar con DHCPv4, sólo se dejaran las bases del protocolo DHCPv6 para futuros proyectos o implementaciones.

El protocolo DHCPv6 fue desarrollado por la IETF y se puede consultar en el RFC 3315. Los clientes escuchan los mensajes por el puerto 546 y los servidores por el puerto 547.

Formato del mensaje DHCPv6

El formato DHCP del protocolo versión seis es muy simple. Ver Figura 3.18.

Tipo del mensaje	ID de transacción
Opciones (variables)	

Figura 3.18 Formato del mensaje DHCPv6

Equivalencia de mensajes entre DHCPv4 y DHCPv6

En DHCPv6 cambia el nombre de los mensajes enviados para el proceso de asignación de una dirección IP.

Los mensajes enviados durante el proceso DHCP en IPv4 e IPv6 son muy similares, hay algunos que tienen su equivalente en DHCPv4 y algunos otros son exclusivos de

DHCPv6. A continuación se muestra la Tabla 3.8, donde podemos observar el equivalente de los mensajes DHCPv6 respecto a DHCPv4.

Tabla 3.8 Equivalencia de mensajes DHCPv4 y DHCPv6

Mensajes DHCPv4	Mensajes DHCPv6
DHCP Discover	Solicit (1)
DHCP Offer	Advertise (2)
DHCP Request	Request (3), Renew (5), Rebind (6)
DHCP Ack / DHCP Nack	Reply (7)
DHCP Release	Release (8)
DHCP Inform	Information-Request (11)
DHCP Decline	Decline (9)
Ninguno	Confirm (4)
DHCP Force renew	Reconfigure (10)
Ninguno	Relay-form (12), Relay-reply (13)

A continuación se describe brevemente los mensajes utilizados en DHCPv6.

(1) SOLICIT : el cliente envía el mensaje en busca de un servidor DHCP

(2) ADVERTISE: mensaje enviado por el servidor DHCP para notificarle al cliente que está disponible su servicio. Este mensaje es enviado para responderle al cliente por su mensaje SOLICIT.

(3) REQUEST: Mensaje enviado por el cliente para solicitar parámetros de configuración incluyendo la dirección IP. Este mensaje sólo es enviado a un servidor determinado.

(4) CONFIRM: Mensaje de confirmación por parte del cliente hacia el servidor para comprobar si la dirección que le fue asignada es correcta para el enlace en el que el cliente está conectado.

(5) RENEW: Mensaje de renovación de el cliente para el servidor que le asigno la dirección IP y los parámetros de configuración para prolongar su tiempo de préstamo o actualizar sus parámetros de configuración.

(6) REBIND: El cliente envía este mensaje cuando no ha recibido respuesta de un mensaje RENEW que envió con anterioridad, para conseguir una renovación u otros parámetros.

(7) REPLY: El servidor envía un mensaje de respuesta que contiene las direcciones asignadas y los parámetros de configuración en respuesta a un mensaje que fue enviado por cliente SOLICIT, REQUEST, RENEW o REBIND.

(8) RELEASE: Mensaje de liberación enviado por el cliente para el servidor que le asignó su dirección IP, para notificarle que ya no requiere más esa dirección.

(9) DECLINE: El cliente envía este mensaje de inconformidad al servidor que le asignó una dirección, para informarle que la dirección que le dio ya está en uso en la red que se conectó.

(10) RECONFIGURE: Un servidor envía un mensaje a un cliente para informar que el servidor tiene nuevos parámetros de configuración y que el cliente puede solicitarlos.

(11) INFORMATION-REQUEST: El cliente envía el mensaje al servidor para solicitarle parámetros de configuración sin la asignación de dirección IP.

(12) RELAY-FORW: Un agente de transmisión envía un mensaje al servidor cuando este recibe un mensaje por parte del cliente o de otro agente de transmisión que desea comunicarse con otro servidor en otro segmento de red.

(13) RELAY-REPL: El servidor envía un mensaje al agente de transmisión, que está haciendo de puente entre él y un cliente.

Funcionamiento

El funcionamiento de DHCPv6 es muy similar al de DHCPv4. Su funcionamiento es el siguiente:

(1) El cliente envía un mensaje SOLICIT en busca de un servidor DHCP que le pueda atender su solicitud.

(2)El servidor que recibe este mensaje le responde con un ADVERTISE, para notificarle que está en servicio y puede atender peticiones.

(3)El cliente envía un REQUEST para solicitarle una dirección IP y otros parámetros de configuración.

(4)Por último el servidor le envía un mensaje REPLY que contiene la dirección IP que le asignó y sus parámetros de configuración.

En la Figura 3.19 se aprecia el proceso de asignación de dirección IP mediante DHCPv6.

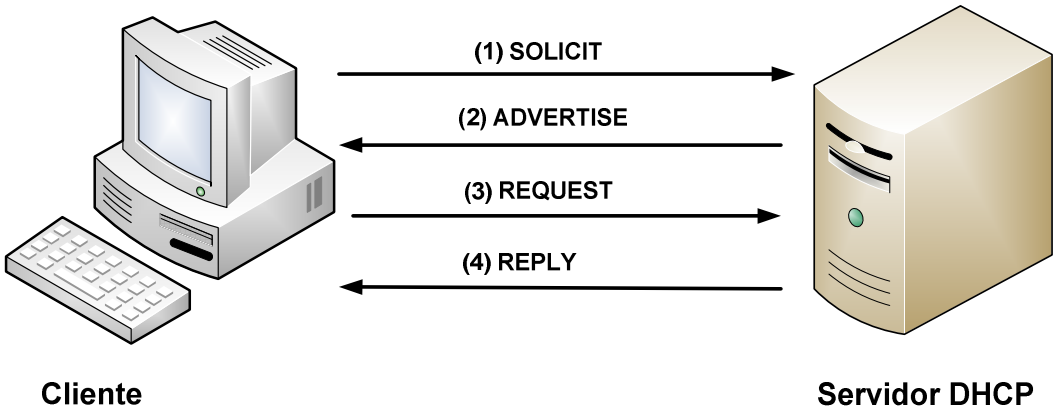


Figura 3.19 Proceso de negociación DHCPv6.

Capítulo 4. Alta disponibilidad en Linux

En la mayoría de servicios de red es importante tener un buen diseño que contemple las posibles fallas que puedan afectar la disponibilidad del servicio, es decir, crear un sistema tolerante a fallos, poniendo especial atención en los servicios críticos de una organización, ya que en caso de una falla generaría altos costos. En este capítulo se presentan soluciones que dotan de alta disponibilidad al servicio DHCP para diferentes casos de implementación de acuerdo a las necesidades de la organización.

4.1 Fundamentos de alta disponibilidad

Al hablar de disponibilidad la primera relación que se establece es el poder acceder a un recurso cuando se necesite; de hecho formalmente disponibilidad es el poder acceder a la información deseada cuando se requiera y tantas veces como sea necesario [3]. En el siguiente punto, se definirán los conceptos más importantes para entender el concepto de alta disponibilidad [7].

- *Outage:* Es cuando un servicio o aplicación no es proporcionado durante periodo específico de tiempo. El *outage* puede clasificarse en dos tipos, el planificado y el no planificado.
    - *Planificado:* Es cuando el servicio o las aplicaciones son detenidas para realizar cambios o mantenimientos planeados**.**
    - *No planificado:* Se da cuando el servicio o las aplicaciones se detienen por causas no planeadas, como desastres naturales, errores humanos, de hardware, fallas de software, entre otras.
- *Uptime (tiempo en servicio):* Es el tiempo durante el cual el servicio o las aplicaciones han estado en funcionamiento.
- *Downtime (tiempo de inactividad):* Es el tiempo durante el cual el servicio o la aplicación no están disponibles, se mide a partir de la pérdida del servicio (*outage en inglés)* y termina hasta que el servicio o la aplicación vuelve a funcionar correctamente.
- *Acuerdo de nivel de servicio (SLA): Service Level Agreements* en inglés es aquel que determina el grado de responsabilidades para mantener el servicio disponible, el costo, los recursos, la complejidad del servicio y las penalizaciones en caso de incumplimiento. Si el servicio se cae, los usuarios son directamente afectados y como consecuencia generará un costo para la organización el cual se penalizará de acuerdo a lo establecido. El grado de responsabilidad varía dependiendo de las necesidades de los usuarios y éste aplica para los departamentos de la organización con fines de calidad.
- *Operación continua:* Es la característica de los sistemas donde los cambios de hardware y software son transparentes para el usuario. En estos sistemas

generalmente los *outages* son planificados, por el hecho de ser sistemas diseñados para brindar operación continua.

- *Disponibilidad continua:* Es un nivel de servicio el cual indica que el servicio debe brindarse al usuario en forma constante y sin interrupciones. Este es el nivel más alto de disponibilidad por lo que los *outages* planificados como no planificados no pueden existir en este tipo de sistemas**.**
- *Punto único de falla SPOF: Single Point Of Failure* en inglés es cuando un componente de hardware o software del sistema puede potencialmente "tirar el sistema" en caso de falla y sin la posibilidad recuperarlo fácilmente. En sistemas de alta disponibilidad se tienden a prevenir los SPOF usando redundancia en cada operación.

4.1.1 Alta Disponibilidad

La alta disponibilidad (*High Availability* en inglés también conocida como HA) es la característica de un sistema que le permite protegerse o recuperarse de alguna falla en un periodo corto a través de mecanismos automatizados, no importando si las fallas son parte misma del sistema, provocadas por su entorno, o por fallas humanas, el sistema debe ser capaz de recuperarse.

Para la definición anterior es importante tomar en cuenta los siguientes 3 factores:

1) *Clasificación de las interrupciones:* Se deben conocer los posibles escenarios de fallas del sistema para poder definir mecanismos que permitan corregirlas en el menor tiempo posible, al saber el comportamiento del las fallas es posible calcular el tiempo de inactividad del servicio (o *downtime* en inglés) que dichas fallas pudieran provocar.
2) *Clasificación del sistema:* Es necesario conocer el tiempo máximo que el sistema puede estar fuera de operación (o*utage*) para poder determinar si es factible implementar la alta disponibilidad, en caso de que el *downtime* del servicio sea pequeño, será conveniente implementar la alta disponibilidad, en caso contrario en aquellos servicios que puedan estar fuera por una semana y

no afecte el funcionamiento de los equipos terminales, no sería conveniente un esquema de HA. En la tabla 4.1 se muestran las categorías de los sistemas basado en el *downtime.*

3) *Protección o Recuperación automatizada:* Hay diferentes enfoques tecnológicos que permiten dar solución a los problemas de disponibilidad. Probablemente una misma solución de HA pueda implementarse en dos servicios diferentes, pero esto no quiere decir que en las dos sea necesaria, posteriormente se hablará más a detalle de los diferentes esquemas de recuperación automatizada.

Hay que diferenciar la alta disponibilidad de la disponibilidad continua. Ésta última implica que el servicio nunca sea detenido o interrumpido, la disponibilidad continua es un subconjunto de la alta disponibilidad pero en la disponibilidad continua cada componente está protegido o respaldado para que nunca se detenga el servicio, contrario a la alta disponibilidad donde el proceso de recuperación comienza después de la falla.

En esta tesis se abordarán los sistemas de alta disponibilidad de misión crítica, los que son considerados importantes y fundamentales para la organización (sombreados en la tabla 4.1), así como los elementos necesarios que debe tener un sistema para considerarse de alta disponibilidad.

4.1.2 Clasificación de fallas

Como tal la disponibilidad de los servicios de TI y en este caso los servicios de red están relacionados directamente con la ejecución de las funciones de la organización, estas funciones son diferentes y muy variadas, algunas dependen más que otras de ciertos servicios, por lo que el impacto de una falla sería diferente en cada servicio, para asignar a cada sistema una categoría basado en el tiempo de falla, en la tabla 4.1 se muestran las categorías de cada sistema.

Tabla 4.1 sistemas y categorización basado en tiempo de falla

Categoría	Inactividad Mínima	Inactividad Máxima
Disponibilidad continua	**1min.**	**2 hrs.**
Misión crÍtica	**10 min.**	**8 hrs.**
Fundamentales para la organización	**1 hora laboral**	**3 días**
Actividades internas de la organización	**1 día laboral**	**1 semana**

Para poder clasificar los sistemas basta con responder 5 preguntas:

1. ¿Qué fallas tienen que ser manejadas de forma transparente para el usuario, es decir, donde *no puede* ocurrir una falla o debe existir *protección contra fallas*?
2. ¿Cuánto tiempo puede durar una pequeña interrupción, 1 día, 1 semana o un mes? Estas pequeñas interrupciones son llamados *outages menores*.
3. ¿Cuánto tiempo puede durar una interrupción medianamente larga, que es poco probable que ocurra pero en caso de ocurrir significa efectos indeseables graves para el sistema de TI? También llamadas *outages mayores o desastres.*
4. ¿Cuánta información puede perderse durante los *outages mayores*?
5. ¿Qué tipo de fallas son consideradas pequeñas o que fallas están fuera del alcance del proyecto?

Protección contra fallas

Es la respuesta a la primera de las preguntas anteriores. La protección contra fallos permite proteger el sistema contra las causas que podrían dejar el sistema o servicio indisponible. Esta proporciona operación continua del servicio en caso de falla. Por cuestiones económicas y para evitar complejidad del sistema, sólo para algunos componentes es importante proteger la disponibilidad, comúnmente son los componentes de hardware, unidades de almacenamiento, fuente de poder y tarjetas de E/S. Para otros componentes del sistema, especialmente el software, la protección contra fallas es muy cara y sólo se considera necesaria en diseños donde se requiere disponibilidad continua, posteriormente se hablará más de este tipo de protección.

Outages menores

Esto responde a la segunda pregunta, hay que evitar ir por el camino fácil y decidir que si un usuario no cuenta con un servicio específico ya no pueda trabajar, en su lugar hay que preguntarse si el sistema donde ocurrió la falla es representativo para la organización. Ciertamente hay situación donde algunos usuarios no pueden trabajar en caso de que alguno servicio falle, en estos casos se debe evaluar en qué casos estas pérdidas de servicio son significativas para la organización para así optar por una solución de alta disponibilidad. De hecho en la categoría actividades internas de la organización bastaría con establecer un contrato que asegure reparar un sistema en el intervalo de tiempo dado pero en algunos sistemas donde la recuperación no es tan rápida o no es posible recuperarlos en el periodo de tiempo dado y el servicio es importante para la organización, entonces necesitamos considerar la alta disponibilidad.

Outages mayores

Se considera un *outage mayor* o desastre, al outage que resulta muy dañino para la organización incluyendo desastres físicos como un huracán, un incendio o una bomba. Después de todo si algún administrador borra información importante es un error humano pero, dependiendo de la importancia de los datos puede considerarse como

un desastre para la organización, es por ello que necesitamos estar preparados con un plan para manejar un desastre como el que ocurre después de un huracán. La cuarta pregunta sólo considera con la pérdida de datos en caso de un desastre, pero en caso particular de esta tesis no se cuenta con grandes volúmenes de información por lo que no se profundizará en este punto.

Out of Scope

La quinta pregunta radica en que todos los servicios de TI están diseñados e implementados en un contexto de negocios. Un análisis de riesgo para algunos escenarios puede decidir cuales fallas pueden ser consideradas y cuáles no, generalmente se excluyen los *outages* mayores y sólo se consideran los *outages* menores, el problema es que los *outages* mayores son comúnmente desconocidos o ignorados. Existen escenarios de falla radicales como un incendio simultáneo en el datacenter primario y su backup, comúnmente se ignoran por que el riesgo de mitigación puede ser muy alto. Anteriormente se mencionó que los *outages* mayores son poco probables pero aun así no deben ser menospreciados, tal es el caso de un sabotaje, esta consideración generalmente queda fuera del alcance del proyecto y es tan rara y poco probable que no es considerada pero aun así podría provocar un *outage* mayor, o en el mejor de los casos un *outage* menor. No es realista asumir que se tiene el 100% de soluciones para proteger o para recuperar las posibles fallas de un sistema, económicamente hablando el hecho de que una solución así no tenga sentido e incluso no sea posible de implementar, ocurre en caso de que todas las locaciones remotas, tengan un outage al mismo tiempo, si agregamos otro sistema de backup, ¿qué pasa si también tiene un outage al mismo tiempo?, por lo que se puede saber que no todos los riesgos son controlables, entonces el alcance del proyecto se reduce a analizar cuidadosamente el riesgo para los sistemas de IT y crear una buena estrategia de mitigación, aceptando los riesgos residuales existentes [4].

Causas de downtime

En la figura 4.1 y 4.2 se presentan las causas más comunes de *downtime*, en particular en la figura 4.1 se muestran las causas más comunes según Gartner, y en la 4.2 según

CNT ambos corporaciones analista de sistemas. De acuerdo a Gartner la mayor causa del *downtime* del sistema es el software, y sólo el 23 al 27% a hardware. En la figura 4.2 las fallas de hardware causan un 44% del *downtime*, más del triple del estimado para software. Se concluye que al encuestar a diferentes organizaciones los resultados pueden variar considerablemente.

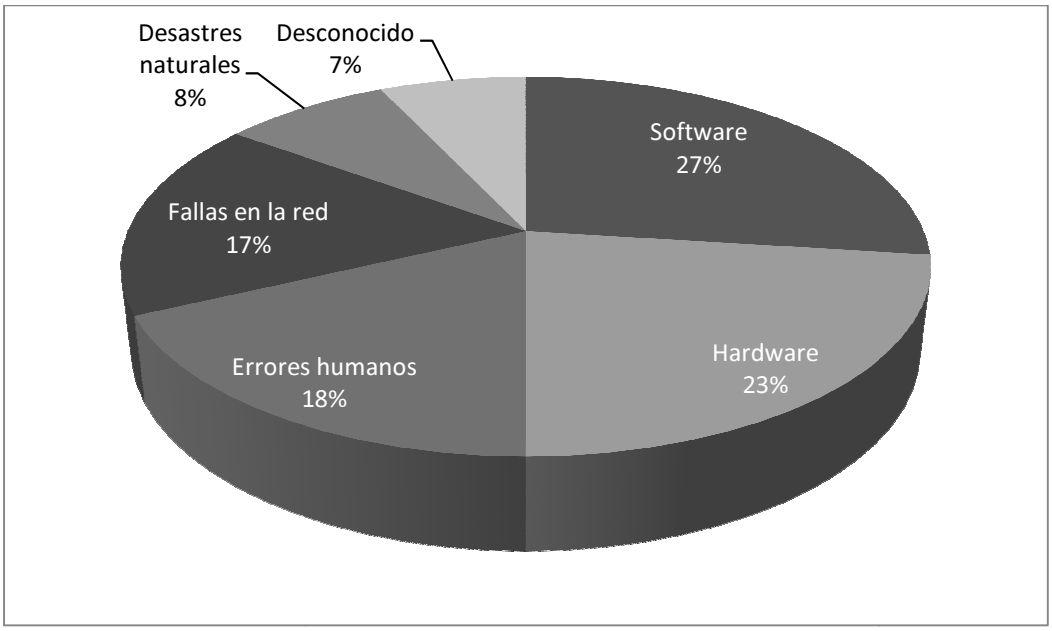


Figura 4.1 Las causas más comunes de downtime no planeado según Gartner.

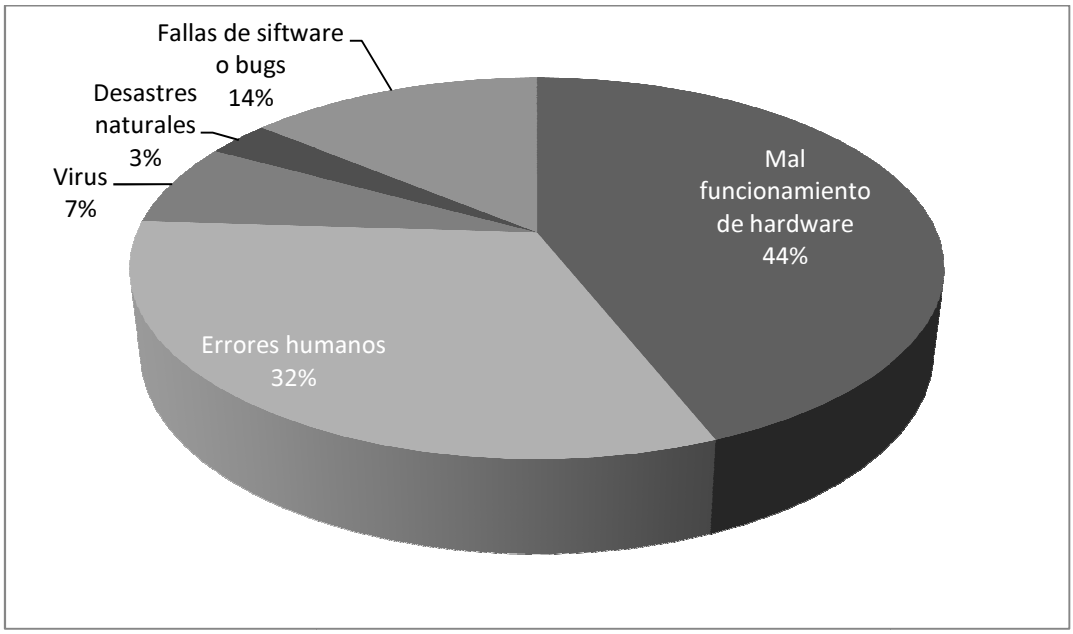


Figura 4.2 Las causas más comunes de downtime no planeado según CNT

En ambos gráficos un punto en que ambos están de acuerdo es que una de las causas más importantes de *downtime* son los errores humanos. La gente una de las causas principales generalmente por dos razones principales [7]:

- ✓ Errores "tontos" o descuidos
- ✓ No siempre entienden la forma en que opera el sistema

La mejor forma de combatir las fallas causadas por errores humanos es la capacitación combinada con un diseño simple. Para esto es necesario capacitar constantemente al personal a cargo de los sistemas, y elaborar documentación sólida a la mano y actualizada, con el fin de reducir los *outages* no planificados.

4.1.3 Disponibilidad en números

Al diseñar un sistema y establecer los requerimientos de disponibilidad generalmente los usuario y los administradores desearían que el sistema invariablemente estuviera disponible el 100% del tiempo, "El proyecto es tan importante que no puede tener *downtimes*", pero esta perspectiva se ve mermada cuando se establecen los costos de una disponibilidad del 100%, entonces la interrogante se traduce a términos de dinero. Como se puede ver en la tabla 4.1 para varias aplicaciones el 99% de *uptime* es adecuado [4]. Si el sistema falla una hora y media por semana, podría no ser dañino. Claro está que el tiempo de falla depende de la hora en que esta ocurra, si ocurre entre las 3:00 am y las 4:30 am del sábado será más "tolerable" que una falla a las 11:30 am en miércoles o todos los fines de semana a las 2 pm por 15 o 20 minutos.

Tabla 4.2 Midiendo la disponibilidad

Porcentaje de *uptime*	Porcentaje de *downtime*	*Downtime* por año	*Downtime* por semana
98%	**2%**	7.3 días	3 horas, 22 minutos
99%	**1%**	3.65 días	1 hora, 41 minutos
99.8%	**0.2%**	17 horas, 30 minutos	20 minutos, 10 segundos

99.9%	**0.1%**	8 horas, 45 minutos	10 minutos, 5 segundos
99.99%	**0.01%**	52.5 minutos	1 minuto
99.999%	**0.001%**	5.25 minutos	6 segundos
99.9999% (seis nueves)	**0.0001%**	31.5 segundos	0.6 segundos

Un punto de negociación del proyecto son las horas durante las que el sistema debe tener un *uptime* del 100%, si se requiere que esté disponible por unas cuantas horas al día, esta meta es razonablemente alcanzable. Por ejemplo en una casa de bolsa entre las 9:30 am y las 4 pm, más unas 3 o 4 horas de cada lado el 100% de *uptime* es requerido. En un periódico puede requerirse un 100% durante las horas de producción, pero no el resto del día. Sin embargo si el 100% de *uptime* es requerido por tiempo 7 x 24 x 365, el costo es muy elevado, casi prohibitivo que sólo las aplicaciones más críticas y en empresas muy grandes puede ser considerado, incluso de esta forma, si se trata de hacer a largo plazo resulta casi imposible de mantener.

Cuando se avanza progresivamente entre los diferentes niveles de disponibilidad, el costo aumenta muy rápido. En este caso, considerando un servidor de DHCP (Red Hat) con medidas de protección especiales, como los respaldos y copias de archivos entre discos, y considerando que tiene una disponibilidad del 99%, si se acopla este servidor con uno idéntico a él (servidor DHCP CentOS) y se configura para entrar en operación cuando el primer servidor falle, y este también ofrezca un 99% de disponibilidad, entonces teóricamente se puede asegurar una disponibilidad combinada del 99.99%. Matemáticamente se multiplica el *downtime* del primero (Red Hat) que es 1% por el *uptime* del segundo, es decir 99%, el servidor CentOS sólo estará en operación durante el 1% de *downtime* del Red Hat, entonces el resultado será 0.99%, sumando esto a la disponibilidad original de 99%, se obtendrá el 99.9%, que es el *uptime* teórico para la pareja o también llamado Failover, del cual se hablará más a detalle posteriormente. Claro está que en realidad el 99.99% no ocurrirá simplemente combinado dos servidores. El incremento en la disponibilidad no es puramente lineal,

en realidad hay un tiempo de espera entre el "switcheo" usualmente llamado Failover, y el periodo en que el conjunto de servidores esta fuera. Adicionalmente hay fallas externas que pueden afectar el acceso a ambos servidores, como la conectividad en la red o los cortes de corriente. Estas fallas sin duda disminuirán la disponibilidad por debajo del 99.99%. Aunque, sólo usemos los 9 con propósitos de modelado, en realidad creemos que los nueves se han convertido en un estándar de los fabricantes de sistemas y sistemas operativos como un punto de referencia para establecer el tiempo "realista" de *uptime* de sistemas ya que consideramos por lo anteriormente dicho en el punto 4.1.2 pueden haber tantas posibles fallas que es imposible tener control sobre todas éstas.

El mito de los 9's

Hasta el momento se ha hablado de los "5 nueves" o más de disponibilidad, el término es utilizado frecuentemente en cuestiones de marketing, porque con éste podemos medir el tiempo medio entre fallas (MTBF) del hardware y así proyectar su tiempo de inactividad a lo largo de un año. Los factores que determinan la disponibilidad de un sistema están basados en las configuraciones de software, la carga, las expectativas del usuario y el tiempo que tarda en ser reparada. Antes de tomar una decisión de cual solución de alta disponibilidad será implementada es bueno hacer un análisis para saber cuáles son los requerimientos reales de nuestro sistema y así cuadrarlos con el número de 9's de cada solución. Para ello hay que tomar en cuenta los siguientes puntos [7].

- *Los nueves son sólo un valor promedio:* Debe considerarse que los *outages* máximos, o tiempos máximos para reparar, son más importantes que el promedio del tiempo en funcionamiento o *uptime*.
- *Los nueves sólo miden lo que puede ser modelado:* La carga del servicio y el software implementado usualmente son difíciles de modelar, por esto la disponibilidad se medirá en términos de la disponibilidad actual, es decir, se tienen que medir en intervalos de reparación real en sistemas corriendo cargas reales.

- *Los nueves reflejan sólo un sistema de un conjunto:* La fiabilidad de un servicio de red tiene que estar basada en la fiabilidad de la red de computadoras es decir, depende de la pila de componentes que componen la red. El sistema más confiable y tolerantes a fallas del mundo es inútil si se pone detrás de un router mal configurador. En general los vendedores de soluciones utilizan los nueves como una buena forma de expresar la disponibilidad, pero carecen de algunos puntos.

Todos los *downtime* no son iguales. Si un *outage* produce que los clientes o usuarios se queden fuera, entonces es más costoso que un *outage* que produce meramente algunos inconvenientes para el usuario. Considerando el costo del *downtime* en un sitio de comercio electrónicos como amazon o ebay, tenemos que durante el curso de un año existe un *outage* de 30 minutos, aparentemente el sistema tiene un *uptime* respetable de 99.9904%, Aunque el *outage* ocurra el primer viernes de diciembre en la noche, y cueste mucho más para la organización que si el mismo *outage* ocurriera en un domingo a las 4:00 am en julio. Por lo que las estadísticas de disponibilidad no hacen distinción entre el verdadero costo que un *outage* pudiera tener para una organización.

Cálculo de la disponibilidad individual

La disponibilidad se puede definir como el tiempo que un servicio o un componente del sistema está disponible para su uso. El downtime de un componente es relevante para el servicio de disponibilidad si éste es indispensable para la operación del servicio, en el caso del DHCP uno de estos componentes es la interfaz de red ya que en caso de falla podría irrumpir contra la disponibilidad del servicio.

La medida base de la disponibilidad es la relación entre el tiempo de actividad respecto al total de tiempo transcurrido.

$$Disponibilidad = \frac{uptime}{uptime + downtime}$$

El tiempo transcurrido incluye el tiempo programado así como el tiempo de inactividad no programado. Un punto sensible en el cálculo de la disponibilidad es conocer si el tiempo transcurrido se interpreta como tiempo de reloj o tiempo de servicio, ambas definiciones son válidas para el cálculo de la alta disponibilidad. En esta tesis se tomará como referencia la hora de reloj, que es la mejor para sistemas de alta disponibilidad por su continuidad, la única desventaja es el efecto de los mantenimientos preventivos regulares con la disminución de la disponibilidad.

La disponibilidad puede ser expresada en valores absolutos (239 o 240 hrs. el último mes) o como porcentaje (99.6% el último mes). La disponibilidad también pude expresarse en términos del usuario, es decir que un periodo de tiempo se multiplica por el número de usuarios, por ejemplo tomando un tiempo de actividad de 239 hrs. y 1000 usuarios del sistema, serían 239 000 de 240 000 horas hombre de trabajo lo que da un mejor indicador sobre lo que significaría 1 hora de *downtime.*

En una forma más simple, la disponibilidad, sea alta, baja o intermedia, vista de otra forma se puede calcular con:

$$A = \frac{MTBF}{MTBF + MTTR}$$

Donde *A* es el grado de disponibilidad expresado como porcentaje, MTBF es el tiempo medio entre fallas, y MTTR es el tiempo máximo para reparar o resolver un problema en particular.

Observaciones:

Cuando el MTTR se aproxima a cero, A se acerca al 100%.

Mientras el MTBF se hace mayor, MTTR tiene un menor impacto en A.

Por ejemplo si un sistema en particular tiene un MTBF de 10,000 horas y un MTTR de 1 hora tiene un valor bastante alto de disponibilidad ósea 100,000/100,001 o 99.999%. Si se reduce el MTTR a 6 minutos o 1/10 de hora la disponibilidad aumenta un 9 a 99.9999%. Pero para lograr este nivel de disponibilidad es necesario un sistema con

una duración entre falla de 100,000 horas, lo que significa que éste tendrá que estar en buenas condiciones durante 11 años.

Disponibilidad de un sistema

En la mayoría de los casos consideramos un sistema como una "caja" independiente con el fin de evitar considerar variables que aunque pueden afectar la disponibilidad del sistema, son despreciables para simplificar el modelado. Tal es el caso de la disponibilidad de un sistema, que se calcula modelando el sistema como una interconexión de partes en serie y en paralelo. Las siguientes reglas son usadas para calcular la disponibilidad del sistema tanto en serie como en paralelo.

- Si una parte del sistema falla y hace que el sistema colapse en su totalidad, de manera que se considera que todas las partes están conectadas en serie.
- Si la falla se da en una parte y permite a las otras seguir en operación se considera que los sistemas están operando en paralelo.

Disponibilidad en serie

Como se muestra en la figura 4.3, dos sistemas X e Y se considera que operan en serie, si alguna de las partes falla, lo que provoca que el conjunto falle. El sistema combinado XY puede operar sólo si ambas partes X e Y están disponibles. De esto podemos decir que la disponibilidad combinada es el producto de la disponibilidad de las dos partes.

$$A = A_x * A_y$$

Donde A representa la disponibilidad total del sistema, Ax y Ay la disponibilidad respectiva de cada sistema.

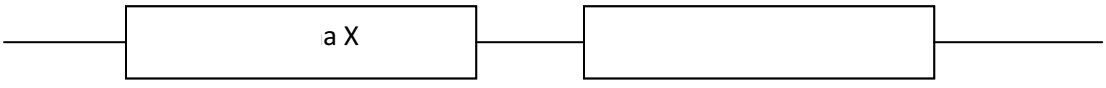


Figura 4.3 Sistemas en serie

La implicación de las ecuaciones mostradas dicen que la disponibilidad de dos sistemas en serie siempre es menor que la disponibilidad de los sistemas individuales, lo cual se

muestra en la tabla 4.3 donde se presenta la disponibilidad y el *downtime* para cada uno de los sistemas y la combinación en serie.

Tabla 4.3 Disponibilidad individual y en serie

Componente	Disponibilidad	Downtime
X	99% (2-nueves)	3.65 dias/año
Y	99.99% (4-nueves)	52 minutos/año
X e Y Combinados	98.99%	3.69 dias/año

De la tabla 4.3 es claro que aunque en el sistema Y se tiene una disponibilidad muy alta, la disponibilidad total disminuye por la baja disponibilidad del sistema X. Esto demuestra que la cadena es tan fuerte como el eslabón más débil.

Disponibilidad en paralelo

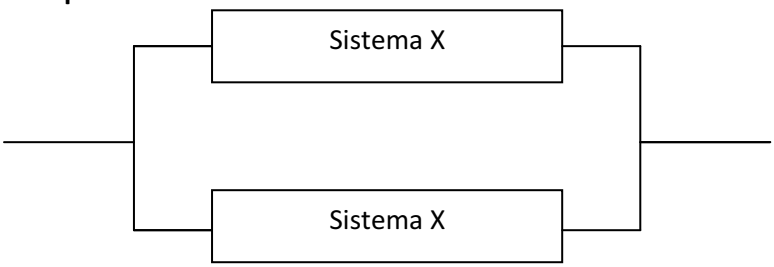


Figura 4.4 Sistemas en paralelo

Como se muestra en la figura 4.4, dos sistemas X e Y se considera que operan en paralelo, de tal manera que cuando los dos sistemas fallan provoca que el conjunto falle; así, la combinación de los sistemas es operativa si ambos están disponibles. De esto la disponibilidad combinada es 1 – (ambas partes indisponibles). La disponibilidad combinada está dada por la ecuación.

$$A = 1 - (1 - Ax)^2 \quad [13]$$

Esta ecuación implica que la disponibilidad combinada de dos componentes en paralelo siempre es más alta que la disponibilidad de componentes individuales, lo cual se aprecia en la figura 4.4 y cuya disponibilidad de presenta en la tabla 4.4 donde

se muestra la disponibilidad y el *downtime* para componentes individuales y la combinación en paralelo.

Tabla 4.4 individual y en paralelo

Componente	Disponibilidad	Downtime
X	99% (2-nueves)	3.65 días/año
Dos componentes X operando en paralelo	99.99% (4-nueves)	52 minutos/año
Tres componentes X operando en paralelo	99.9999% (6-nueves)	31 segundos/año

En la tabla 4.4 es claro que aunque en el sistema X tiene una disponibilidad baja, la disponibilidad total es mucho más alta. Así la operación en paralelo provee un poderoso mecanismo para hacer un sistema altamente confiable y disponible. Por esta razón todos los sistemas de misión crítica están diseñados con componentes redundantes en paralelo.

4.2 Implementación basada en Linux HA

Como se dijo en el capítulo 1, se propondrán varias implementaciones para dotar de alta disponibilidad un servicio DHCP para los diferentes niveles de disponibilidad del servicio, así como sus ventajas y desventajas en función de algunos factores.

En esta implementación se hablará del proyecto Linux-HA. Linux HA dota de alta disponibilidad para Linux a través de una comunidad abierta, la mayoría de Software de Linux-HA está bajo licencia de *GNU General public License* (GPL) y la *Lesser General Public Licence* (LGPL).

La versión 2 de Linux-HA proporciona [5]:

- Configuración Activo/Activo y Activo/Pasivo.
- Failover y Failback en nodos, Dirección IP, o falla en recursos.
- Soporte para el estándar Open Cluster Framework (OCF) y Linux Standard Base (LSB).
- Interfaz por línea de comandos (CLI) e interfaz gráfica de usuario (GUI) para configuración y monitoreo.

- Soporte para cluster de hasta 16 nodos.
- Soporte multi-estado (maestro/esclavo).
- Configuración basada en XML
- No depende del kernel o del hardware.
- Capacidad de balanceo de cargas con Linux Virtual Server (LVS).

Para entender más a detalle las características anteriores definiremos a continuación algunos conceptos.

Cluster: Un cluster es un grupo de servidores y recursos que se comportan como uno sólo, es decir como una entidad para proporcionar la capacidad de alta disponibilidad y en su caso el balanceo de cargas.

Failover: Failover es el proceso en el cual uno o más servicios de un servidor se transfieren a otro servidor o servicio en el mismo cluster por fallas o mantenimiento. También se utiliza para nombrar al otro miembro del cluster en un cluster de 2 nodos.

Failback: Es el proceso mediante el que uno o más recursos de un servidor que han fallado regresan a su operación original.

Servidor Primario (Activo): Un servidor primario o activo es el miembro de un cluster que posee los recursos del cluster y corre procesos usando esos recursos. Cuando el servidor está comprometido se detiene y asigna los recursos a otro servidor que se encuentre en *standby* o al secundario.

Servidor secundario (Standby, pasivo o Failover): Un servidor secundario es un miembro de un cluster capaz de acceder a los recursos y procesos activos del cluster, generalmente está en estado de espera hasta que el servidor primario sea comprometido o tenga que ser detenido. Cuando esto ocurre, todos los recursos del Failover se pasan al secundario que se convertirá en el servidor primario.

Split-brain (Cerebro dividido): En un escenario de Cerebro-dividido más de un servidor o aplicación que pertenece al mismo cluster trabajan con un mismo recurso, por lo que puede que es más probable que ocurran fallas debido a que cuando un

servidor del cluster piensa que su otro se encuentra abajo por pérdida de comunicación entre ellos toma sus recursos aunque no necesariamente el otro servidor se encuentre abajo.

Fencing: Fencing es el mecanismo utilizado en soluciones de alta disponibilidad para evitar que un miembro inestable del cluster acceda a los recursos compartidos y comunicaciones de otros miembros del sistema. Cuando se aplica *fencing*, el servidor inestable no puede correr ningún proceso hasta que su comunicación con otro miembro del cluster se restablezca. Disparar al otro nodo en la cabeza en inglés *Shoot The Other Node In The Head (STONITH)* es una de las técnicas más utilizadas para realizar *fencing*.

Quorum

Es un mecanismo que trata de resolver el problema de cerebro-dividido seleccionando un subconjunto de servidores del cluster para representar el cluster completo, cuando es forzado a dividir en varios subconjuntos de cluster por problemas de comunicación.

STONITH (*Disparar a otro nodo en la cabeza*): Algunas veces también llamado STOMITH ("Shoot The Other Member/Machine In The Head") es una técnica para hacer fencing, la idea principal del STONITH es que si un nodo está operando en condiciones diferentes a las establecidas en las políticas de operación, se le enviará una señal de STONITH para detenerlo. Dos nodos en un mismo cluster tratando de realizar una misma tarea podrían ocasionar graves problemas, como en el caso que ambos recursos accedan a un disco compartido.

4.2.1 Heartbeat

Es el núcleo de Linux HA, Heartbeat es llamado así por su analogía con el latido del corazón, asociándolo con estar vivo, Heartbeat proporciona la capacidad de clustering, es decir, conocer la presencia o ausencia de algún otro servidor y para intercambiar mensajes entre sí, lo que asegura la alta disponibilidad de recursos críticos como datos, aplicaciones y servicios. Éste permite monitorear los recursos del cluster, el Failover y Failback.

La comunidad de desarrollo brinda a Heartbeat gestores de recursos para varios servicios como son DB2, Apache, DNS y en este caso DHCP, dependiendo de los requerimientos de disponibilidad Heartbeat puede administrar varios recursos a la vez.

Arquitectura Heartbeat versión 2

Internamente Heartbeat está diseñado con una arquitectura basada en capas, las cuales interactúan internamente para brindar capacidades de alta disponibilidad. La figura 4.5 muestra un ambiente Heartbeat con tres nodos del cluster.

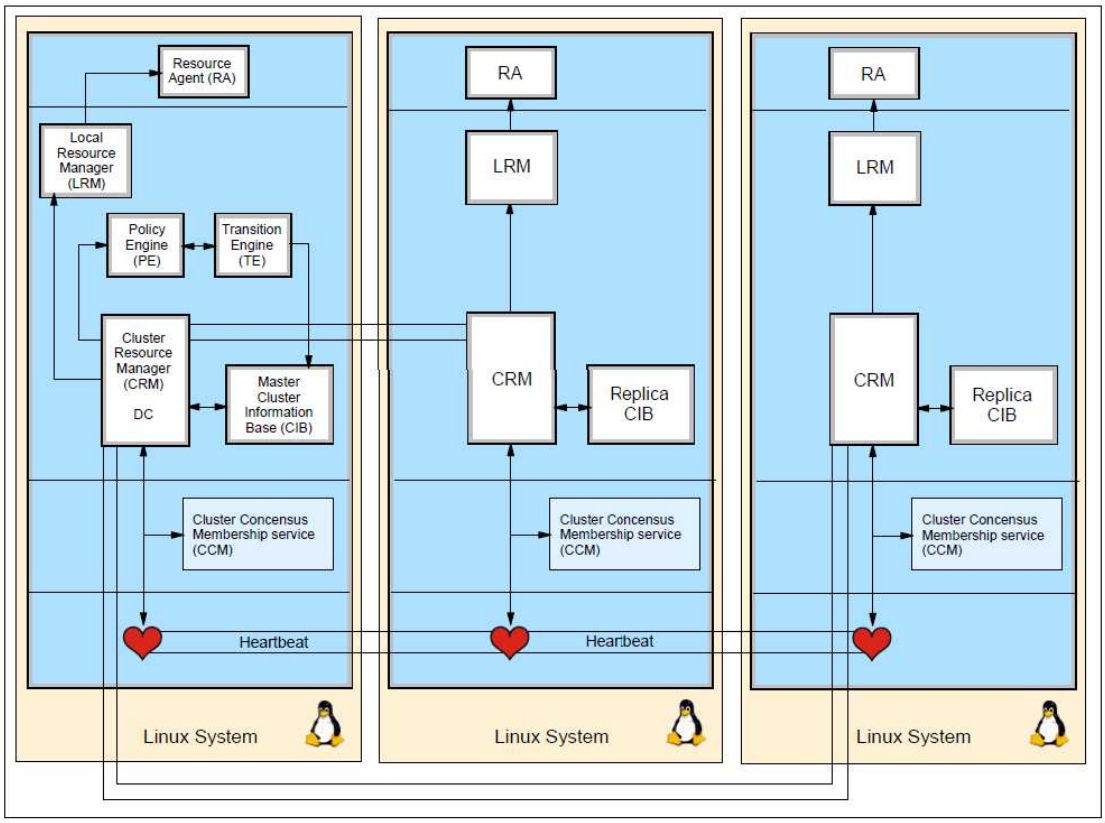


Figura 4.5 Arquitectura Heartbeat

La arquitectura de Heartbeat está compuesta de los siguientes elementos y capas [5]:

a) *Messaging and infrastructure layer (mensajes e infraestructura)*

Está compuesta de los componentes que integran el Heartbeat. El Heartbeat está representado por el corazón de la figura 4.5, éste envía una señal de "Estoy vivo" y

mensajes relativos al cluster a los demás miembros del cluster para que conozcan su estado.

Heartbeat es crucial para el funcionamiento correcto del cluster, el canal de comunicaciones sobre el cual Heartbeat envía sus señales debe ser altamente disponible, ya que si alguno de los canales de comunicación se daña, el cluster completo se cae, porque los miembros no se pueden comunicar entre sí. Es posible dar alta disponibilidad a los canales de comunicación de Heartbeat teniendo al menos dos medios de comunicación que permitan a los nodos del cluster comunicarse entre sí.

b) Membership layer (Capa de afiliación)

De acuerdo con la guía de Novell *SUSE Linux Enterprise Server Heartbeat*, la segunda capa es la de afiliación o *membership* en inglés. La capa de afiliación es la responsable de calcular el número de nodos conectados al cluster y actualizarlo entre los miembros del cluster, basado en la información que recibe de la capa de hertbeat. La lógica que se encarga de esta tarea está contenida en el servicio *cluster consensus Membership (CCM)*, que brinda una organización simplificada de la topología del cluster (node-wise) a los componentes del cluster que están en las capas superiores.

c) Resource allocation layer (Capa de asignación de recursos)

Es donde todas las reglas del cluster e información de estado son establecidas y almacenadas. Los componentes en esta capa toman decisiones basadas en las reglas del cluster, está compuesta de varias partes de las cuales se hablará posteriormente.

d) Cluster information base o CIB (Base de información del cluster)

Es una representación completa del estado y la configuración del cluster en formato XML, incluyendo afiliación de nodos, recursos, restricciones, entre otros. Hay sólo una CIB maestra en el cluster, los demás elementos del cluster poseen una réplica, esta puede ser modificada vía cibadmin o con una interfaz gráfica o GUI.

Cualquier cambio en la CIB es rápidamente replicado a los otros miembros, una falla en un nodo puede ser ocasionada por la CIB maestra y propagada a los otros nodos.

e) Cluster Resource Manager o CRM (Administrador de recursos del cluster)

Cada nodo del cluster tiene un CRM local que mantiene la CIB y se comunica con el LRM (citado en el siguiente inciso) para llamar los RA o Resource Agentents (agentes de recursos) locales de los cuales se hablará posteriormente. El CRM procesa todas las transacciones que pasan a través de la capa de asignación de recursos. Un nodo del cluster es elegido como el Coordinador designado (DC), el DC debe mantener la CIB maestra y comunicar los cambios a los otros CRM's en los demás nodos. El CR; en turno, actualización de la CIB local y su réplica. El DC es la entidad en el cluster que decide cuándo debe ejecutarse un cambio general en todo el cluster, como fencing en un nodo o la migración de recursos a otro nodo.

f) Local Resource Manager o LRM (Administrador de recursos locales)

El *local resource manager* se encarga de hacer llamadas a los agentes de recursos locales en nombre del CRM. Así, él puede arrancar, detener o monitorear operaciones y reportar los resultados al CRM. El LRM es la fuente autoritaria de recursos para todos los recursos de información relacionados al nodo local.

g) Policiy engine and transition engine o PE y TE (motor de políticas y transacciones)

Cada vez que el Coordinador designado necesita hacer un cambio general en el cluster (Reaccionar a una nueva CIB) el Motor de políticas *o policy engine* en inglés (PE) es usado para calcular el siguiente estado del cluster y la lista de recursos o acciones requeridas para lógralo. Los comandos computados por el Motor de políticas después son ejecutados por el motor de transición o *transition engine (TE)* en inglés. El DC envía mensajes al *Cluster Resoruce Magnager*, que luego usa su LRM para la manipulación de recursos necesaria.

h) Resource Layer o RL (Capa de recursos)

La capa de recursos es la capa más alta en la arquitectura Heartbeat y contiene un RA o agente de recursos que controla las operaciones de los recursos del cluster. Un RA es el programa que ha sido escrito para arrancar, detener y monitorear los recursos. Para

cada recurso del Heartbeat hay un RA asociado que maneja todas las operaciones, el RA es llamado por el LRM para arrancar, detener o monitorear recursos.

Mientras que la mayoría de los otros componentes de Hearbeart trabajan de forma automática e independiente, en el RA un administrador está en contacto directo con él. Hay 4 clases de RA.

i) Cluster Consensus Membership Services o CCM (Servicio de consenso de Miembros del cluster)

CCM proporciona un consenso seguro de los miembros conectados al cluster, este asegura que cada nodo pueda hablar con los otros nodos del cluster

Open cluster framework RA

- Linux Estándar Base RA
- Classic Heartbeat RA
- STONITH RA

Flujo del proceso de actuación del cluster

Hay varios eventos, que pueden disparar un cambio en el cluster como:

- La caída de un nodo o un recurso o servicio
- Agregar o quitar un recurso del cluster
- Iniciar o detener un recurso
- Cambiar una restricción para un recurso
- Agregar o quitar un nodo
- Migrar un recurso de un nodo a otro

El flujo del proceso cambia de la siguiente manera cuando hay un cambio en la afiliación de nodos:

1. Regularmente los miembros del cluster se comunican con una señal de “Estoy vivo” entre ellos usando el componente de heartbeat en la capa de *messaging and infraestructure* constantemente para informar su estado. Esto pasa aunque se haga algún cambio en el cluster.

2. Si hay un cambio en el cluster este cambio es enviado a la CCM en la capa de afiliación o *membership layer*.
3. El CCM envía paquetes a sus compañeros del cluster para conocer su estado y determina exactamente cuales nodos están afiliados y cuáles no.
4. Tras confirmar cambios en la afiliación el CCM notifica al CRM del coordinador designado o DC (por sus siglas en inglés) en la capa de asignación de recursos o *resource allocation layer*.
5. El CRM del DC actualiza la CIB maestra y se la envía al CRM de los miembros del cluster. La CRM en turno actualiza y réplica la CIB. Si el nodo que falla es el DC, los miembros del cluster que aun están en contacto votan para seleccionar otro nodo que se convertirá en el DC, entones ese nodo procesa y propaga todos los cambios.
6. Después que la CIB maestra cambia, el DC CRM notifica al *policy engine*. (PE), entonces el PE busca en el CIB (incluyendo la sección de estado) y ve que tareas necesitan ser realizadas para actualizar el estado del cluster con las políticas definidas.
7. El PE genera una lista de acciones para poner el cluster en línea con las políticas y dárselas al CRM. El CRM envía la lista al *transition engine* (TE) para ser ejecutadas. Si el fencing está habilitado entonces el demonio de STONITH es invocado para "resetar" el nodo que falló.
8. El DC envía mensajes al CRM del cluster, el cual luego usa su LRM para realizar las manipulaciones necesarias locamente.
9. El DC es notificado cuando las acciones se completan en los miembros del cluster.
10. Después que se completan todas las acciones el cluster regresa a un estado de inactivo y espera para eventos futuros.

De forma similar cuando un recurso cambia, los cambios son enviados al CRM del DC, que actualiza la CIB maestra con las nuevas políticas, o estado de información, después el CRM del DC propaga los cambios a la CRM de los miembros del cluster, que actualizan sus CIB locales. Luego el CRM del DC ejecuta los procesos necesarios para poner el estado actual en línea con las políticas. Con la ayuda del PE y el TE, las

acciones requeridas son ejecutadas. Luego el cluster regresar a un estado de inactividad y espera para futuros eventos.

Consideraciones de seguridad en Hearbeat versión 2

Cuando los nodos del cluster se comunican entre sí, deben poder determinar que están recibiendo los mensajes relacionados con el cluster de una fuente legítima y no peligrosa, es decir, que estos deben de poder autenticarse adicionalmente los nodos deben saber que los mensajes del cluster no han sido interceptados y manipulados. Para proteger el cluster de estos ataques, con Hearbeat se pueden configurar métodos de autenticación para las comunicaciones entre los nodos del cluster. Estos 3 métodos son:

- CRC
- Algoritmos de Secure hash 1 (SHA1)
- Mensaje de algoritmos de resumen 5 (MD5)

El método de CRC por sí mismo no tiene autenticación de mensajes, sólo protege de la modificación de los mensajes, por lo que usando solamente el método CRC el cluster se vuelve vulnerable a ataques.

SHA1 y MD5 son métodos de resumen que requieren un secreto compartido. El secreto compartido es una contraseña que es establecida por el administrador y es usada para cifrar y autenticar mensajes. El método SHA1 es recomendado por que proporciona una autenticación más segura. La clave de autenticación (el secreto compartido *shared key*) es conocida por todos los miembros del cluster, cuando un miembro del cluster envía un mensaje a otro miembro del cluster, esta clave de autenticaciones es usada para cifrar el mensaje, este mensaje es enviado al nodo destino del cluster y éste usa la clave compartida o *shared key* para descifrar el mensaje. En este sentido los miembros del grupo están protegidos de ataques de red. El método de autenticación y la contraseña son establecidos en la primera instalación de Heartbeat.

4.2.2 Resource Agents

Un agente de recursos es un programa, comúnmente un script Shell, que ha sido escrito para arrancar, detener y monitorear un recurso. Para cada recurso que se desee manejar con Heartbeat, debe existir un agente de recursos que administre todas las operaciones. Heartbeat versión 2 es compatible con 4 clases de agentes de recursos [5]:

- Agentes basados en OCF
- Scripts de inicio LSB en /etc/init.d
- Scripts de recursos clásicos de Heartbeat
- Agentes STONITH

Agente de recursos OCF (Open Cluster Framework)

El proyecto Open Cluster Framework define APIs para las capacidades básicas del cluster como el servicio de nodos, servicio de recursos y servicios de clustering. Los recursos basados en agentes OCF sólo son soportados por Heartbeat versión 2. Hearbeat contiene alrededor de 40 OCF basados en agentes de recursos y están en el directorio /usr/lib/ocf/resource.d/heartbeat, como apache, my-sql, tomcat entre otros. En el caso del servicio DHCP no se cuenta con un agente de recursos OCF.

Agentes de recursos LSB

Heartbeat también trabaja con scripts LSB (*Linux Standard Base en inglés*) contenidos en el directorio /etc/init.d. Los sistemas basados en SUSE Linux Enterprise Server (SLES) y Red Hat Enterprise Linux brindan estos scripts de forma estándar en sus distribuciones. En el caso de DHCP el agente de recursos LSB se instala automáticamente en el sistema, bajo en nombre de dhcpd.

Cuando se usa un agente de recursos basado en init LSB, se le otorga a Heartbeat el control total de este recurso por lo que no se debe operar directamente el servicio a través del script init mientras Heartbeat esté corriendo. Por ejemplo, en el servicio de DHCP, mientras el servicio de Heartbeat esté activo, sólo Heartbeat es el responsable de iniciar y detener el servicio, y no debe ser ingresado manualmente con

/etc/init.d/dhcpd start y /etc/init.d/dhcpd stop. Algunas veces Heartbeat bloquea los scripts en init.d para prevenir estos casos.

Agentes de recursos Heartbeat

Son scripts similares a LSB. Varios RA Heartbeat están relacionados con los RA OCF. Estos scripts manejan argumentos para analizar y controlar, así como de verificación y automatización. Cuando se observa un recurso con el "ocf/Heartbeat" quiere decir que hay una relación entre los RA OCF y los RA Heartbeat. Los RA Heartbeat están en el directorio /etc/ha.d/resources.d/ Estos vienen empaquetados por defecto con Hearbeat.

Agentes de recursos STONITH

Disparo al otro nodo en la cabeza o Shoot The Other Node In The Head (STONITH) en inglés es una técnica de Linux-HA's para realizar fencing. Actualmente el proyecto cambio de nombre a *cluster glue*, pero se utilizará el término STONITH en general para referirse a *cluster glue*. Fencing es el término usado para las acciones tomadas para asegurar que un nodo "mal portado" de un cluster no acceda a los recursos compartidos y no responda a las respuestas entrantes a través del apagado del nodo o *"halting".* El recurso STONITH proporciona una interfaz de alto nivel para hacer fencing a uno o más nodos errantes. EL demonio STONITH que viene con la versión 2 de Heartbeat usa el API de la versión 2 de STONISH, hay varios métodos de restablecer un nodo, como el lic_vps o ssh. Estos son proporcionados a través de *plugins* y son proporcionados por el paquete *cluster-glue*.

Cluster Information Base

La CIB es un archivo en formato XML que contiene todo lo necesario para entender el cluster, específicamente los recursos y el estado actual del cluster. LA CIB maestra es mantenida por el Coordinador designado del cluster, el DC réplica la CIB maestra a los otros nodos del cluster. La CIB representa dos partes:

- La configuración del cluster que incluye los nodos y la información de los recursos y las constantes que existen en el cluster.
- Una instantánea del estado actual del cluster.

La información en la instantánea nos dice que nodos y que recursos están vivos, ésta sirve para que el PE en el DC compare el estado de la información y determine la lista de acciones a tomar conforme a las políticas de configuración.

Hay varios métodos para crear y manipular el cib.xml. El primer métodos es usando las herramientas CLI. El segundo método es usando el administrador gráfico, el tercer método es crear o editar un XML y luego usar el **cibadmin** para importarlo al cluster, de preferencia hay que usar el CLI o el método gráfico aunque cualquiera de los tres es indistinto.

Mientras el cluster se encuentra activo se puede obtener la CIB usando el siguiente comando

#Cibadmin –Q > cib.xml

4.2.3 Fencing

Fencing es el término usado en soluciones de alta disponibilidad para describir el acto de reiniciar un nodo o nodos errantes que están accediendo a los recursos de un cluster y responden a las aplicaciones o peticiones de red. Un nodo aislado o fenced en inglés no puede hacer nada útil hasta que sea reparado y vuelva a ser un nodo activo del cluster, en Linux-HA hay dos métodos para hacer fencing.

- Auto-fencing
- STONITH

Con el auto-fencing el recurso posee la capacidad de aislarse automáticamente, esto garantiza que sólo accederá a sus recursos, usualmente con un mecanismo de bloqueo.

Cuando no se tiene este tipo de recursos se puede usar el STONITH para brindar fencing al cluster. De acuerdo al sitio web de Linux-HA www.linux-ha.org los sistemas con discos compartidos deben ser configurados con STONITH para evitar corrupción de datos por los nodos errantes.

Hay varios mecanismos para el STONITH, desde desconectar de la red el nodo errante, hasta métodos de intervención humana. Heartbeat es capaz de controlar estos mecanismos y prevenir fallas potenciales en el nodo, en el caso de que un nodo falle

Heartbeat ejecutará las siguientes tareas:

1. Se percata que un nodo no está enviando la señal de “Estoy vivo” al cluster
2. Envía una notificación de pre-parada a los demás nodos del cluster. Esta notificación incluye los mensajes de que el nodo que falló será apagado.
3. Le indica al servicio STONITH apagar el nodo errante
4. Envía una señal de post-parada a los otros miembros del cluster siempre y cuando la parada del cluster haya sido satisfactoria. Estas notificaciones incluyen mensajes de que el nodo que fallo fue detenido.

Para conocer las especificaciones del stonith basta con teclear el comando:

```
#man stonith
```

Para determinar cuándo usar STONITH, hay que considerar:

Cuando un cluster está usando un disco compartido. Si los miembros del cluster están usando un disco compartido entonces se debe utilizar STONTIH para prevenir comportamientos no esperados con los datos. Las configuraciones de alta disponibilidad que no requieren compartir un disco, como firewall y servidores Web estáticos donde el contenido proviene de una sola fuente no requieren fencing por que los nodos afectados sólo pueden corromper sus propios datos.

4.2.4 Herramientas de administración del cluster Heartbeat

Heartbeat versión 2 brinda varias formas de configurar y administrar un cluster. Un método es usando herramientas vía CLI, otro es usando una interfaz gráfica la cual está incluida en la versión 2 de Heartbeat. Adicionalmente existe una forma de administración basada en Web utilizando Webmin.

Administración por línea de comandos (CLI)

Algunas de las instrucciones de línea de comandos son:

Cibadmin CLI. Administra la CIB, puede actualizar, modificar o borrar parte o toda la CIB.

Crm_attribute Está equipado con herramientas para consular y manipular atributos de nodo y opciones de configuración del cluster. Estos atributos también pueden modificarse usando la GUI.

Crm_verify CLI para verificar la CIB, verifica que la configuración de la CIB sea correcta, no es configurable vía GUI.

Crm_mon Monitorea el estado del cluster, incluyendo los nodos, recursos y el estado de las acciones. Está disponible en la GUI.

Crm_diff Se utiliza directamente en dos XML para identificar fácilmente las actualizaciones y parches aplicados. No está disponible en la versión GUI.

Crm_resource Administra las configuraciones de los recursos. Esta capacidad está disponible en la GUI tool.

Crm_failcount Este recurso puede monitorear y "reinciar" el número de veces que un recurso ha fallado en un nodo dado. No está en la GUI.

Crm_uuid Genera y recupera el UUID de un nodo. No está disponible en la GUI.

Crm_standby Administra el estado de *standby* de un nodo, es decir no puede correr ningún recurso, es útil cuando se realizan actualizaciones, esta capacidad si está disponible en la GUI.

Restricciones

En Linux-HA las restricciones son especificaciones para definir dónde y cómo se desea que corran los recursos. Es complicado dominarlas pero ayudan en cierta forma a tener un cluster funcional.

Heartbeat versión 2 tiene tres tipos de restricciones:

- Restricciones de ubicaciones: especifican cuáles nodos del cluster van a correr un recurso específico
- Restricciones de orden: especifican el orden en que los recursos van a correr

- Restricciones de colocación: Dice al cluster que recursos pueden o no juntos en un nodo.

4.2.5 Configuración Activo/pasivo con Heartbeat

Una configuración activo/pasivo consiste en un servidor que tiene asignados todos los recursos a un miembro del cluster, y otro servidor que es capaz de acceder a los recursos y se encuentra en estado de espera hasta que el cluster que posee los recursos falle [14].

La ventaja de esta configuración es que no hay degradación del servicio y sólo se reinicia cuando el servidor principal no está disponible, aunque una desventaja es que el servidor pasivo no brinda ningún tipo de servicio mientras se encuentra en *standby*, haciéndolo menos eficiente que una configuración activo/activo. Otra desventaja es que toma tiempo realizar el cambio o Failover entre los recursos. En la figura 4.6 el servidor de la izquierda que es el que brinda el servicio (servidor activo) y el de la izquierda permanece en estado de *standby*, es decir no da servicio a los clientes mientras que el servidor activo esté corriendo.

Con una configuración activo/pasivo la versión 2 de Hearbeat tiene un nodo activo y uno o más nodos pasivos. Los recursos son configurados para correr en el nodo activo cuando el cluster es iniciado sólo el nodo activo brinda el servicio.

En la figura 4.7 si el sistema Redhat falla, Heartbeat restablecerá la CIB, por lo que el Failover cambiará el servicio DHCP al servidor CentOS. Si STONITH está habilitado, Heartbeat primero llama al demonio STONITH para hacer fencing a Redhat para que no corrompa los datos compartidos o responda a una petición de DHCP entrante. Luego la LRM del CentOS pide el servicio DHCP para comenzar a dar servicio, después de que arranca satisfactoriamente el LRM informa al manejador de recursos del DC. El CRM en turno actualiza la CIB maestra y réplica la información a los otros miembros del cluster. En este caso sólo el servidor CentOS actualizará la información y ahora él es el DC.

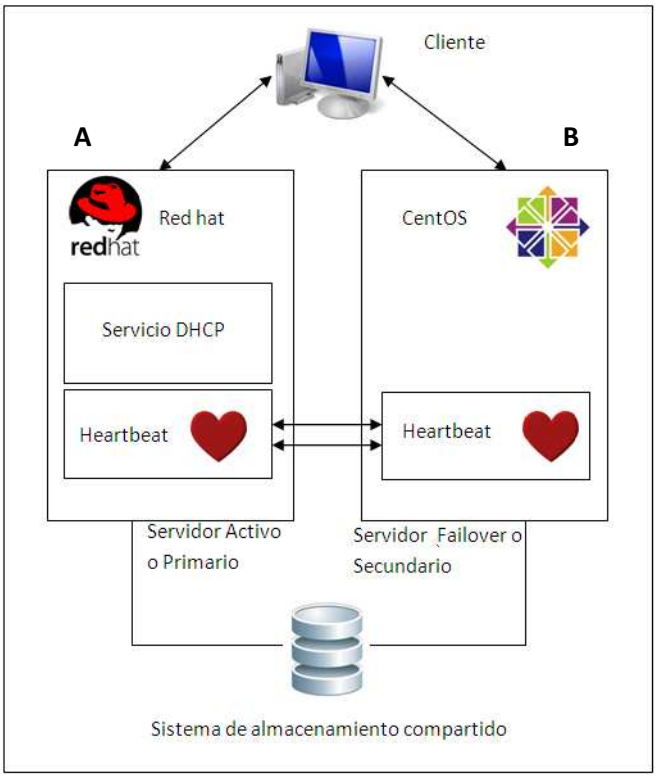


Figura 4.6 esquema Failover activo/pasivo

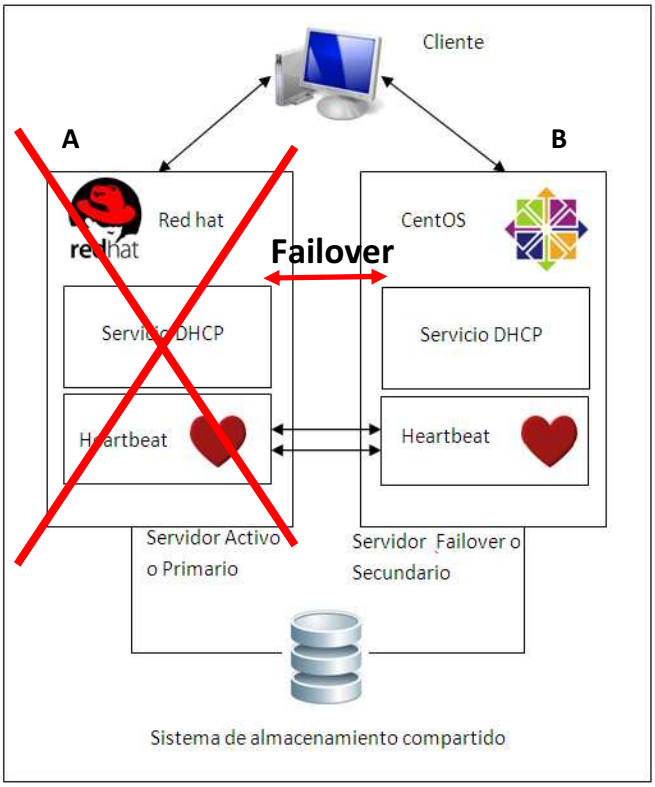


Figura 4.7 esquema Failover activo/pasivo después de una falla

El administrador del sistema es el encargado de diagnosticar y reparar el sistema A y así recuperarlo tras una falla, entonces las siguientes opciones pueden ocurrir dependiendo de la configuración del Failover:

1. Si el administrador habilita la configuración *Failback*, el Heartbeat toma las acciones necesarias para regresar el servicio DHCP al sistema A.
2. Si el administrador tiene el Failback deshabilitado entonces el servicio permanece en B hasta que sea forzado a regresar a A o el servidor B falle.

Este tipo de configuraciones son fáciles de realizar por que no tienen muchas limitaciones, es decir, sólo es necesario configurar algunos recursos o en nuestro caso un servicio (DHCP) para poder correr en un nodo a la vez. Dependiendo de los acuerdos de nivel de servicio (SLA), la configuración activo/pasivo puede o no brindar los requerimientos de alta disponibilidad necesarios. Si es necesario un cluster de muy alta disponibilidad o disponibilidad continua probablemente sería necesario usar una configuración activo/activo.

Quorum en Heartbeat

Quorum es el nombre del mecanismo que utiliza Heartbeat para logar el escenario de "cerebro dividido" o Split-brain en inglés. En un cluster con dos nodos se experimenta un escenario de cerebro dividido cuando un nodo pierde la comunicación del Heartbeat con el otro nodo, en este caso, cada nodo considera inseguro el estado del otro nodo y asume que el otro nodo esta caído. Si el fencing está habilitado, entonces cada nodo trata de apagar al otro, si el fencing no está habilitado, entonces cada nodo se solicita a si mismo convertirse en el DC, entonces las configuraciones de la CIB puede perder sincronización.

Con Quorum, si un nodo pierde comunicación con el cluster, el mismo se aísla lo que significa que este nodo no habilitará su Heartbeat ni sus recursos. Quorum establece que al menos un nodo debe poder estar en comunicación con al menos un nodo, para establecer un Quorum e iniciar sus recursos, es decir, que se necesita al menos tres nodos para que el Quorum sea factible, por lo que si se tiene solamente dos y se pierde uno, se perderá totalmente el servicio.

Pruebas básicas de operación

Para realizar las pruebas de operación se asume que se han realizado los pasos para instalar y configurar heartbeat listados en el Apéndice AII “Instalación y configuración de Heartbeat y Failover”, es altamente recomendable revisar el Apéndice AII ya que en éste se describen paso a paso las configuraciones, lo que facilitará entender el funcionamiento del Heartbeat.

Los siguientes dos métodos simularan las fallas:

- Simular una falla de red desconectando de la red el servicio de la interfaz eth0 con el comando: `#ifconfig eth0 down`
- Simular una falla que obliga al sistema forzando el sistema a apagarse con el comando `#halt`

Para verificar que los cambios se realizaron, conectar un equipo a la red para solicitar dirección IP fuera del cluster, esto quiere decir que le dará dirección por el servidor inicial 192.168.1.XX

Para simular una falla en el sistema en un modo activo/pasivo, basta con apagar el sistema activo y disparar el servicio Heartbeat para mover los recursos. Cuando el sistema que falla deja el recurso, en este caso es por la falla en el equipo el recurso se mueve a la mejor locación en los nodos restantes del cluster ver figura 4.8

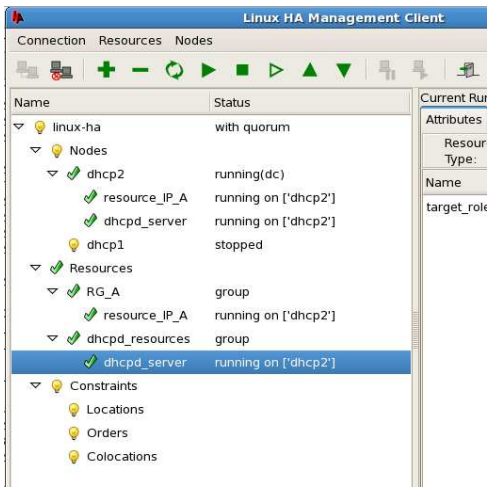


Figura 4.8 estado de asignación de recursos de grupo

Una verificación rápida usando el comando ping muestra que el objetivo aún es alcanzable.

Pérdida de conectividad en el modo activo/pasivo

Después de verificar que el recurso se movió en caso de una falla, se verifica que la configuración de *pingd* está trabajando y regresamos el sistema nuevamente a un estado donde ambos servidores estén trabajando en orden.

Después ingresar en la consola de administración del cliente que tiene el recurso y desconectarlo de la interfaz principal en este caso la eth0, la pérdida de conectividad de red es detectada y provoca que Heartbeat ajuste el score de los nodos basado en la constante *pingd.* Para conocer el score de cada nodo es posible descargar scripts para conocer el valor analizando la salida del comando ptest que viene con el paquete Heartbeat. Desafortunadamente la salida de *ptest* tiende a cambiar del software de hearbeat. El objetivo principal del cálculo del score es encontrar la mejor localización para correr un recurso de acuerdo a los tamaños introducidos por varios factores involucrados (valor de stickness, constantes y otros). El nodo con el score más alto es escogido para correr el recurso.

En este caso de implementación en particular donde se consideró un cluster de dos nodos como ejemplo, el valor de *pingd* no es muy relevante ya que no hay más servidores donde se pueda migrar el recurso, pero si es importante definirlo para un mejor funcionamiento de Heartbeat en caso de falla.

Adicionalmente por tratarse de una configuración Activo/Pasivo no es necesario establecer parámetros de configuración para realizar Quorum en los nodos.

4.3 Implementación basada en DHCP ISC Failover Protocol

En este punto se hablará de la segunda solución de alta disponibilidad propuesta para el servicio DHCP; el Failover protocol. Éste tiene la capacidad para que dos o más servidores DHCP trabajen en conjunto para brindar un servicio de DHCP confiable y con soporte de recuperación ante desastres a un conjunto definido de direcciones o pool. En este punto también se dará un panorama general del protocolo Failover para entender su funcionamiento y así poder implementarlo.

Para conocer más a detalle algunos puntos de la implementación del Failover protocol, se puede obtener la última versión del ietf-dhc-Failover.txt a través del grupo de trabajo DHC en la página web de IETF: http://linux.die.net/man/3/omapi. A la fecha de publicación de esta tesis el protocolo se encuentra en "draft state" o como en estado de proyecto o borrador, por lo que aun no se le ha asignado un RFC, pero a su vez hay una parte de éste documentado en los RFC 2131 y RFC 2132.

4.3.1 Funcionamiento del DHCP Failover Protocol

El *DHCP synchronization protocol* comúnmente llamado *Failover protocol* fue diseñado para brindar al servicio DHCP la funcionalidad para que un servidor de DHCP actuara como servidor primario y otro como respaldo. Existen dos configuraciones básicas para el Failover protocol. En la primera el *servidor secundario no proporciona* servicio de DHCP cuando está en contacto con el servidor primario, simplemente acepta actualizaciones del primario, por lo que el secundario solamente da servicio si pierde comunicación con el primario, es decir, funcionan en una configuración activo/pasivo. La segunda configuración es más avanzada, en ésta ambos servidores proporcionan servicio al mismo tiempo, usando un algoritmo de balanceo de cargas para determinar que servidor responderá a cada petición.

Hay varias formas en que el servidor primario y secundario podría perder contacto entre sí, algunas son:

- Uno de los dos falla por uno o más problemas de hardware o software.
- La red local a la que uno o ambos servidores está conectado falle.

- La red o backbone entre los dos servidores falle.

En el caso de que alguno de los servicios o servidores fallen, el servidor no tiene la capacidad de diferenciar entre un fallo en la red y un fallo en el servicio por lo que no puede saber si los demás servidores continúan operando, entonces, cuando los dos servidores no están en contacto, cada uno funciona asumiendo que el o los otros siguen funcionando, ajustando su operación para que el servicio de DHCP permanezca disponible, en este estado el archivo de arrendamiento de direcciones no se actualiza con información que pueda entrar en conflicto, y los dos servidores no asignan la misma dirección IP a diferentes clientes, el detalle se describe poco más adelante.

Sincronización de archivo de préstamo de direcciones

El Failover protocol utiliza una técnica llamada *lazy updates,* en la que cada servidor trata de mantener a su par actualizado pero no necesariamente se tienen que completar las actualizaciones para que el protocolo funcione correctamente, es decir, el servicio de DHCP puede seguir operando mientras se realizan las actualizaciones correspondientes o las actualizaciones no han sido completadas debido a que el protocolo actualiza a su pareja o *peer* después de habar entregado una dirección. Dichas actualizaciones se dan en el momento en que el servidor de DHCP considera que ha pasado un tiempo en que la comunicación entre ambos servidores ha permanecido estable, o tras aplicar los cambios en ambos manualmente. Los servidores siguen un conjunto de reglas que previenen que servidor tenga comportamientos incorrectos en caso donde las actualizaciones no han podido ser completadas, esto permite al servidor secundario asignar una dirección IP al cliente DHCP antes de que se actualice el archivo de arrendamiento al otro. Lo que significa que no hay penalizaciones por desempeño al protocolo DHCP como resultado del uso del Failover protocol.

Restricciones en la asignación de direcciones

Las actualizaciones perezosas o *lazy updates* trabajan estableciendo una serie de reglas de cómo el servidor de DHCP va a asignar direcciones IP. Si ambos servidores de DHCP siguen las mismas reglas, no hay posibilidad de que ambos servidores asignen la misma

dirección IP a diferentes clientes, aunque ambos servidores no estén en comunicación. Estas reglas involucran tres principios:

1. Los servidores primario y secundario dividen el pool de direcciones a asignables en un segmento de red dado en direcciones libres o *free* y direcciones de respaldo o *backup addresses*. Las direcciones libres están disponibles para ser asignadas por el servidor primario, las direcciones de respaldo están disponibles para ser asignadas por el servidor secundario.

2. Los servidores de DHCP pueden asignar o extender un préstamo solamente por un límite de tiempo definido adicional al tiempo dado este tiempo limitado es llamado *máximum client lead time (MCLT)*, y es el tiempo máximo que un servidor puede extender el tiempo de arrendamiento. El MCLT comúnmente es pequeño, por lo general menor a una hora, el servidor o servidores pueden seguir extendiendo el MCLT indefinidamente, pero cuando esto pasa, el cliente tiene que renovar su préstamo con mayor frecuencia. A fin de asignar un tiempo mayor de préstamo al cliente, el servidor que asigna la dirección puede cooperar con el otro servidor estableciendo un *acknowledged potential lease expiry time*, cuando éste tiempo es establecido, el servidor puede extender el préstamo al cliente por ese tiempo más el MCLT, ya que el valor del *acknowledged potential lease expiry* es fijo en el tiempo y no una duración como lo es el MCLT, siempre que el servidor extienda un préstamo tiene que restablece el *acknowledged potential lease expiry time*.

3. El tercer principio es que en condiciones de operación normal, una dirección que ha sido asignada a un cliente no puede ser asignada a otro cliente a menos que ambos servidores de DHCP estén de acuerdo que el primer cliente ya no la está utilizando.

Comunicación entre nodos

Los nodos del Failover se comunican entre sí usando una conexión TCP persistente asíncrona, es decir que cualquiera de los nodos puede enviar un mensaje a otro en cualquier momento y no hay restricción en cuanto al orden de las respuestas, por lo

que cada nodo del Failover puede conectarse al otro nodo en cualquier momento. Esto permite establecer conexión inmediatamente después del inicio del sistema.

En el proceso de establecimiento de conexión, el nodo primario envía un mensaje CONNECT, este mensaje contiene información de identificación y autenticación, así como información de configuración del nodo primario y particularmente el MCLT. Si durante la conexión el nodo secundario reconoce al nodo primario y es capaz de autenticarse, este envía un mensaje CONNECTACK, el cual contiene información de autenticación similar al mensaje CONNECT. Además de información del nodo secundario. Después de que estos mensajes han sido intercambiados satisfactoriamente, los nodos pueden comunicarse normalmente entre sí.

Una vez que los nodos han establecido una conexión, intercambian entre sí el estado en que se encuentran y en caso de ser necesario sincronizan el archivo de direcciones IP (dhcpd.leases). Durante una comunicación normal cuando el server DHCP recibe una solicitud DHCPREQUEST de un cliente, este responde con un DHCPACK y después envía un *binding update (BNDUPD)* al nodo Failover, cuando el Failover recibe la actualización, éste pone la actualización en cola para ser procesada y a su vez, el Failover manda un mensaje *binding acknowledgement (BNDACK)* en respuesta.

Durante los periodos de inactividad, cada nodo envía periódicamente mensajes CONTACT a los demás nodos para decir que está vivo, si no se reciben mensajes de los otros nodos en un cierto periodo, el nodo asume que la conexión se rompió y comienza a operar de forma independiente. La conexión entre nodos puede romperse incluso si uno de los nodos ha sido apagado, en este caso el servidor desconectado manda un mensaje DISCONNECT al otro nodo y entonces se cierra la conexión entre esos nodos.

Tiempos de arrendamiento

El protocolo DCHP en operación sin modo Failover permite que las direcciones IP estén solamente en uno de dos estados. En el primero es cuando el tiempo de expiración de una dirección IP esta en el pasado, es decir la fecha de vencimiento de su préstamo ha caducado, lo cual significa que la dirección está disponible para ser asignada a un

cliente. El otro estado es cuando el tiempo de expiración del préstamo está en el futuro, lo que significa que la dirección no está disponible para ser asignada. En el caso del protocolo Failover éste introduce una mayor complejidad en términos del número de estados que una dirección IP puede tener además el cálculo del tiempo de préstamo para la dirección suele ser igualmente complejo, los estados definidos en el protocolo Failover son los mostrado en la tabla 4.5.

Tabla 4.5 Estados de asignación de direcciones en modo Failover

Estado	Descripción
ABANDONED	La dirección ha sido abandonada como resultado de un conflicto entre ambos servidores
ACTIVE	La dirección está siendo utilizada por un cliente
BACKUP	La dirección está disponible para asignarse por el servidor secundario
EXPIRED	Se asume que la dirección ya no está en uso por el cliente, pero aun así sigue asignada al cliente.
FREE	La dirección está disponible para ser asignada por el primario
RELEASED	La dirección ha sido liberada por el cliente, pero aun no está disponible para ser asignada
RESET	La dirección ha sido liberada por el administrador, pero aún no está disponible para su asignación.

Cuando el nodo Failover hace un cambio de asignación a un cliente, envía un mensaje BNDUPD al nodo primario, que incluye la actualización del nuevo estado de asignación de la dirección, la fecha y hora en la que se realizó el cambio, el tiempo actual de vencimiento del préstamo y tiempo potencial de expiración; cuando el nodo principal procesa el BNDUPD, envía en respuesta al primario un mensaje BNDACK. Después que el nodo secundario recibe el BNDACK, ambos nodos tienen la misma información del estado de préstamo.

En cada nodo existe solamente un tipo de dirección que indica que la dirección está disponible para asignarse, en el servidor primario las direcciones que están disponibles para asignarse están en estado FREE, en las direcciones que están disponibles en el

servidor secundario para asignarse están en estado BACKUP, las direcciones nunca están disponibles en ambos servidores para ser asignadas al mismo tiempo, cuando una dirección ha sido asignada a un cliente, por el servidor primario o el secundario, entra en estado ACTIVE; una direcciones en estado activo no puede ser asignada a otro cliente hasta que alcance el estado de FREE o de BACKUP en caso del secundario. Cuando vence el préstamo de una dirección asignada por el servidor primario o secundario puede ser extendida por ambos servidores y cuando esta dirección expira la dirección cambia de estado ACTIVE a EXPIRED, el servidor primario actualiza el estado de la dirección y lo manda al secundario el cual a su vez también lo actualiza Cuando los cambios son realizados en ambos servidores el estado de la dirección cambia de EXPIRED a FREE, en este momento la dirección está lista para ser asignado por el primario o el secundario, según sea el caso, este saludo de dos vías o *two-way handshake* es necesario debido a la posibilidad de extender el préstamo, ya que si el servidor actualiza una dirección que ha vencido y mueve el estado de la dirección a FREE inmediatamente, podría asignar posteriormente esa dirección a un nuevo cliente mientras que el otro servido la extiende al cliente original.

Esto trae consigo una complicación adicional, después que el servidor ha cambiado una dirección a EXPIRED la notifica al servidor secundario con un mensaje BNDUPD, por lo que este no puede extender el préstamo, esto por que cuando el nodo secundario recibe la actualización, el inmediatamente cambia el estado del préstamo a FREE, por lo que si recibe una petición de asignación la puede atender con la dirección que fue notificada como expirada. Adicionalmente hay dos estados similares al EXPIRED, el RELEASED y RESET. Cuando un nodo del Failover recibe un mensaje DHCPRELEASE, pone el cliente en estado RELEASED, el cual es manejado como el estado EXPIRED en términos de cómo son realizadas las actualizaciones.

Tiempo de préstamo en modo Failover

En ausencia del Failover protocol, el servidor DHCP sigue un proceso de asignación de tiempo de préstamo o arrendamiento muy simple, donde el cliente solicita un tiempo de préstamo en el mensaje DHCPDISCOVER y en el DHCPREQUEST, si éste no se puede

asignar, el servidor le asigna un tiempo por defecto o *default lease time,* que el administrador del sistema establece en el archivo de configuración *dhcpd.conf*.

Con el Failover protocol el *default lease time* es conocido como el *desired lease time*, o tiempo deseado de préstamo, la razón de ser de este nombre es que es el tiempo que el servidor "le gustaría" asignar al cliente, este tiempo depende principalmente del estado del préstamo.

Hay 3 entidades que deben conocer el estado de préstamo de una dirección IP: el cliente DHCP, el servidor primario y el servidor secundario; para los primeros dos no hay problema en la sincronización del tiempo de expiración ya que el servidor primario recibe el mensaje DHCPDISCOVER de un nuevo cliente que no tiene un préstamo activo, entonces encuentra una dirección que esté en estado FREE y se la asigna al cliente, hasta este momento el nodo secundario aún no conoce la dirección asignada por el primario al cliente, por lo que se tiene que hacer un proceso de sincronización de estado de asignación de direcciones que fue definido en el punto anterior, *tiempo de arrendamiento*. Hasta este punto el servidor primario no puede extender el préstamo por más tiempo que el especificado en el MCLT ya que así lo indica el Failover protocol, es decir el servidor primario compara el MCLT con el *desired lease time,* si el MCLT ocurre antes que el *desired lease time,* (que por lo general es así ya que se desea que el MCLT sea menor), el servidor primario asigna un tiempo de vencimiento de préstamo al cliente igual al tiempo actual más el MCLT; si el cliente confirma este tiempo de préstamo a través del protocolo DHCP normal de 4 paquetes o DORA, este terminará en un tiempo igual al MCLT.

Hasta este momento el cliente tiene un tiempo menor al tiempo deseado y, no hay forma de evitar en el inicio que se asigne este tiempo que es menor al tiempo deseado, aunque para lograrlo después de la asignación inicial de tiempo de préstamo, el Failover protocol trata de dar al cliente un tiempo de préstamo cada vez más cercano al tiempo deseado para esto el servidor asigna al cliente un *desired lease time* estimando, y un *potential lease expiry time*, que es el tiempo que espera el cliente para renovar su estado. Entonces el servidor asume que el cliente va a renovar su dirección a la mitad del tiempo actual. Cuando el servidor primario actualiza el servidor

secundario, le manda al secundario en un mensaje BNDUPD con el tiempo actual de expiración y también el *pontential lease expiry time*, cuando el secundario recibe el mensaje BNDUPD, graba el *potential lease expiry time* en su archivo de estado de asignación, entonces envía un mensaje BNDACK al servidor primario cuando el servidor primario recibe el mensaje BNDACK, sabe que el servidor secundario se ha actualizado y tiene el mismo *potential lease expiry time* que él tiene.

Una vez que se realizan dichas actualizaciones, el cliente tiene un préstamo con duración del valor del MCLT, entonces a la mitad del MCLT va a tratar de renovar su préstamo, hasta este momento, el servidor de DHCP va nuevamente a calcular el *desire lease time*, que probablemente será el mismo que en la transacción previa, pero esta vez cuando el cliente renueve, si se le asignará está vez una duración igual al *desired lease time* en lugar del MCLT, ya que fue asignado el *desired lease time* cuando este vence el servidor hace el mismo proceso de cálculo que antes, para determinar cuando el cliente debe renovar, que es ahora la mitad del *desired lease time*, en consecuencia pone el *potential lease expiry time* que es tiempo potencial de expiración en 1 ½ veces el *desired lease time*, con la finalidad de que cuando el cliente renueve su dirección, obtenga el *desired lease time,* es decir deja transcurrir el tiempo del *desired lease time* y le suma la mitad del tiempo para dar oportunidad de actualizar el *desired lease time*.

Haciendo el cálculo del *lease time* en esta forma y manteniendo un *potential expiry time,* un servidor de DHCP que está operando de forma norma en modo Failover, hace parecer al cliente que está operando de una forma similar a la que operaria en modo normal. La única diferencia real es que cuando un cliente obtiene una dirección por primera vez o trata de adquirirla nuevamente una dirección expirada, el primer préstamo que obtendrá tendrá la longitud del MCLT en lugar del tiempo deseado.

Casos de operación del protocolo Failover

Existen diversos casos de operación en que el conjunto de servidores en Failover puede operar. En condiciones de operación normal hay dos tipos de estados uno Activo/Respaldo, y una configuración en balanceo de cargas. Cuando un nodo no es

capaz de comunicarse con otro, entra en un estado de COMMUNICATIONS-INTERRUMPTED. Si un nodo no está operando el otro nodo puede ser puesto en estado de PARTNER-DOWN, tanto por el administrador como por un proceso automático. Adicionalmente existen otros estados durante la transición de un estado de operación a otro o para resolver conflictos entre los nodos.

Funcionamiento normal y configuración Activo/Respaldo

En estado NORMAL, sólo un nodo del Failover responderá a los mensajes DHCP del cliente. Para saber con certeza cual servidor responderá a las peticiones dependerá de de la forma en que esté operando el conjunto, si opera en Activo/Respaldo el primario siempre responderá a los mensajes DHCP y si es en balanceo de cargas ambos responderán según el algoritmo de balanceo de carga el cual es algoritmo hash determinístico). Este algoritmo está definido en el RFC 3074 donde se explica detalladamente, adicionalmente cabe resaltar que en el ISC DHCP no es posible configurar el sistema en este modo, solamente en balanceo de cargas, pero es importante conocerlo. Durante la operación normal cada nodo del Failover actualiza al otro acerca de las asignaciones de dirección IP a un cliente o la renovación de arrendamiento de un cliente, aunque las direcciones son asignadas desde un pool definido, el Failover que asigna direcciones puede ver cuántas direcciones IP quedan para asignar y compara ese número con el número de direcciones que el otro nodo tiene para asignar, si el número no está balanceado, se puede ejecutar una tarea de rebalanceo de pool.

Operación en estado COMMUNICATIONS-INTERRUMPTED

En estado COMUNICATIONS-INTERRUMPTED ninguno de los nodos del Failover pueden conocer si el otro está dando servicio DHCP, por lo que ambos nodos brindan servicio a todos los clientes de los que reciben peticiones debido a que no pueden conocer el estado en que se encuentran, por lo que no pueden enviar actualizaciones, ni existe forma de balancear los pools en este estado cada servidor puede extender el préstamo de cualquier cliente, aunque no puede asignar nuevas direcciones. Este estado tiene una desventaja adicional debido a que el servidor de DHCP sólo puede

asignar un préstamo no mayor al MCLT sin antes actualizar a su compañero, y como el MCLT generalmente es corto los clientes tienen que renovar frecuentemente su préstamo, esto significa que un servidor en estado de COMMUNICATION-INTERRUPTED experimenta una carga mayor a la normal, debido a que el préstamo es menor, por lo que un *outage* en el servicio podría ocasionar la pérdida de direcciones en los equipos.

Operación en estado PARTNER-DOWN

Por varias razones, es posible que un miembro del Failover DHCP pueda dejar de operar, esto puede ser por un *outage* planeado o no planeado, mantenimiento del sistema, pruebas, entre otros. Para que estos *outage* no afecten, o afecten de la menor manera se sugiere poner el servicio en estado PARTNER-DOWN. Cuando un servidor se encuentra en estado PARTNER-DOWN el servidor que queda activo en el Failover toma el control de todo el servicio de DHCP, por lo que asume que el otro nodo no está funcionando, esto significa que él puede reclamar todas las direcciones IP que le pertenecen al Failover, y asignarlas a nuevos clientes que la soliciten. Para poner el servicio en estado PARTNER-DOWN se utiliza el protocolo OMAPI [11].

Cuando se dan mantenimientos planificados el servidor puede ponerse en estado de SHUTDOWN, por lo que el servidor que está siendo apagado o desconectado, envía al otro nodo un mensaje donde dice que se irá a estado SHUTDOWN. Cuando el otro miembro del Failover recibe el mensaje de SHUTDOWN, automáticamente cambia su estado a PARTNER-DOWN.

Un miembro del Failover también puede fallar de forma inesperada, en este caso el otro nodo cambia a estado COMMUNICATIONS-INTERRUPTED. Como se mencionó en el punto anterior, no es muy conveniente, aunque el servicio puede continuar en este estado de forma permanente. Para evitar esto es posible configurar el Failover con un periodo seguro o *safe perio.* El periodo seguro es el periodo entre el tiempo que el servidor entra en COMMUNICATIONS-INTERRUPTED y el tiempo en que se determina que el otro servidor no está operando.

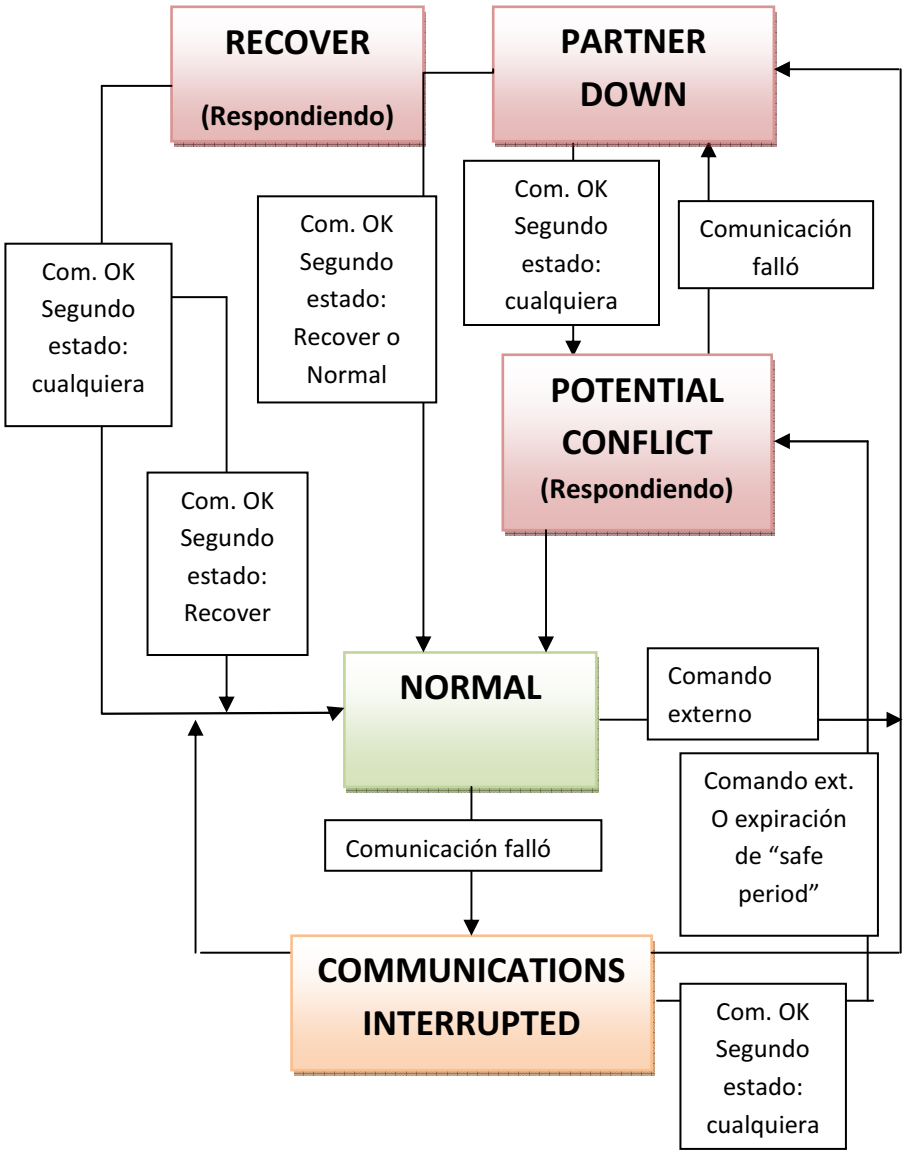


Figura 4.9 Diagrama de estados del servidor principal

En la figura 4.9 se muestran un diagrama de flujo del servidor principal y los diferentes estados por los que puede atravesar en modo Failover.

Operación en estado STARTUP

Es un estado temporal mientras el servicio inicia, en este estado trata de conectarse con los otros nodos del Failover.

Operación en estado RECOVER

El estado RECOVER permite al Failover obtener una actualización completa del otro servidor mientras inicia.

Operación en estado POTENTIAL-CONFLICT

En el proceso de sincronización es posible que los servidores pierdan sincronía debido a un error administrativo, o una transición de estado incorrecta. Cuando un conflicto es detectado, ambos servidores inmediatamente cambian al estado POTENTIAL-CONFLICT y dejan de atender a los clientes DHCP, debido a que está fuera de control, ya que es posible que ambos servidores asignen diferentes direcciones IP a un mismo cliente. Cuando el servidor primario entra al estado POTENTIAL-CONFLICT, inmediatamente envía un UPDATE-REQUEST al servidor secundario. Cuando el secundario entra en POTENTIAL-CONFLICT espera el UPDATE-REQUEST del primario, cuando éste lo recibe envía un BNDUPD con las direcciones que no han sido actualizadas o no se conoce su estado. Después que han sido enviadas todas las actualizaciones en secundario envía un mensaje UPDATE-DONE al primario, y cuando éste lo recibe cambia al estado CONFLICT-DONE Una vez realizado se actualiza el estado de ambos servidores a NORMAL.

Configuración del ISC DHCP

Para realizar este punto es necesario haber revisado el Apendice AII Instalación y configuración de Heartbeat y Failover, ya que en él se detallan los parámetros más importantes de configuración del servicio así como la descripción de cada uno de ellos, de lo contrario, difícilmente se entenderá cómo funciona el Failover protocol.

Ya que se conoce como opera cada uno de los parámetros de configuración básicos de operación de Failover sin balanceo de cargas, en la tabla 4.5 se mostrará un ejemplo de cómo debería ir escrito el archivo de configuración tanto en el servidor primario como en el secundario. Primeramente en el archivo de configuración del primario se tiene que poner la configuración maestra, en el ejemplo esta configuración es agregada al archivo de configuración /etc/dhcpd.conf, y la configuración del maestro en un archivo llamado /etc/dhcpd.master, con el fin de separar la configuración de cada servidor y tener un archivo dhcpd.conf “genérico”.

Tabla 4.6 Extracto del archivo /etc/dhcpd.conf correspondiente al Failover protocol

Configuración del servidor primario	Configuración del servidor primario

Failover peer "casoFailover" {
primary;
address 192.168.70.20;
port 847;
peer address 192.168.70.18;
peer port 647;
max-response-delay 180;
mclt 1800;
split 128;
load balance max seconds 3;
}
include "/etc/dhcpd.master";
	Failover peer "casoFailover" {
	secondary;
	address 192.168.70.18;
	port 647;
	peer address 192.168.70.20;
	peer port 847;
	}
	include "/etc/dhcpd.master";

Detalles de configuración

En los ejemplos de configuración mostrados en la tabla 4.5 se muestra solamente la parte de configuración del archivo dhcpd.conf correspondiente al Failover, pero cabe destacar que este configuración sólo aplica para las direcciones que serán manejadas en un sólo pool, es decir no hay pools separados de cada uno de los servidores y el Failover. Para hacer esto en la tabla 4.6 se muestra un ejemplo de cómo quedaría un pool compartido que sería administrado por el Failover.

Tabla 4.7 Pool en modo Failover

Configuración del pool en modo Failover
pool { **Failover peer "casoFailover";** **option domain-name-servers 192.168.70.9;** **deny dynamic bootp clients;** **range 192.169.70.40 192.168.70.200;** **}**

Otra cosa importante es que en los pool en modo Failover el protocolo no soporta asignación de direcciones utilizado el protocolo BOOTP, por lo que se debe indicar con

la línea *deny dynamic bootp clients* que el pool no soporta BOOTP para evitar problemas, ya que de no hacerlo el servidor de DHCP marcará error y no iniciará.

Puesta en marcha del servicio DHCP en modo Failover

Cuando se inicia el Failover por primera vez, los dos servidores generalmente no dan servicio hasta sincronizarse entre sí, entonces lo primero que se debe hacer es sincronizarlos, si se configuraron de forma correcta este procedimiento debe hacerse de forma automática al iniciar el servicio. Debido a los problemas de sincronización en con respecto a la hora y la fecha se sugiere que se conecten los servidores a un servidor NTP para actualizar hora y fecha.

El proceso de arranque empieza con un estado RECOVER. Después que han establecido comunicación, cada servidor envía al otro una lista completa de los préstamos, una vez sincronizados los servidores pueden esperar un tiempo igual al MCLT antes de comenzar a dar servicio.

Problemas en el Failover protocol

El Failover posee un mecanismo para restaurar un archivo de préstamos o *dhcpd.leases* perdido, para restaurarlo sólo basta con generar un nuevo archivo vacío y reiniciar el servicio, automáticamente el servidor se conecta al otro nodo el cual descarga y actualiza la el *dhcpd.leases*.

Pérdida de arrendamiento

Con el servicio de ISC DHCP, es posible entrar en un estado donde, por problemas de errores internos en el sistema la base de datos no esté actualizada. Este tipo de problemas pueden persistir mostrando mensajes como: "*No free leases" o "Peer hold all free leases*", en el caso de que no todas las direcciones están ocupadas, o simplemente los clientes siempre tienen un tiempo igual al MCLT aunque el servicio este en estado NORMAL. Hasta el momento no hay solución para este problema, lo que se puede probar es si se sabe que la versión actual funciona, forzar a los dos servidores a re-sincronizarse, o en caso de que esto no funcione, detener uno de los dos servidores y borrar el archivo *dhcpd.leases* para una nueva sincronización.

Problemas conocidos en el Failover protocol

El ISC DHCP tiene algunos problemas de funcionamiento debido a que algunas funcionalidades del servicio de DHCP no han sido implementadas, algunas son:

- El ISC DHCP no tiene algún mecanismo para deshabilitar el balanceo de cargas
- El protocolo ISC DHCP no implementa el *safe period* en el Failover protocol
- Tiene problemas de sincronización usando el protocolo OMAPI
- El servidor ISCDHCP no tiene un control sobre las direcciones estáticas, como resultado la reserva de direcciones en modo Failover no está soportada

4.4 Implementación basada en cluster Webmin

Hasta el momento se han definido diferentes propuestas para brindar alta disponibilidad al servicio DHCP. La primera basada en Heartbeat proporciona redundancia en el servicio con una plataforma para dar soporte en caso de fallas del servicio DHCP, donde la arquitectura de alta disponibilidad trabaja de forma independiente del servicio DHCP, es decir, no se tiene que alterar la configuración del servicio DHCP para soportar la alta disponibilidad, lo cual implica que el servicio como tal podrá configurarse como si se tratara de un servidor independiente. La segunda propuesta el Failover protocol proporciona una plataforma de alta disponibilidad dentro del servicio de DHCP, por lo que brinda administración centralizada del sistema, ya que todos los parámetros de configuración se encuentran dentro del servicio mismo, pero con la gran desventaja que al aumentar las configuraciones aumenta la complejidad del sistema por lo que se vuelve más difícil tener control total de su operación.

Webmin como tal es una plataforma de administración de servicios de sistemas Linux vía Web que proporciona administración simplificada para servidores a través de módulos diseñados exclusivamente para operar diversos servicios, como es el caso del DHCP, Sendmail, apache, DNS, administración de clusters, entre otros. Para esta propuesta de implementación se abordará especialmente dos módulos Webmin, uno para el monitoreo del servicio y el estado del servidor y otro para las tareas

planificadas, que al igual que Heartbeat proporciona una plataforma de alta disponibilidad independiente del servicio DHCP, con la ventaja de que su administración y configuración resulta más sencilla y puede resultar igualmente útil para algunos casos donde se requiera alta disponibilidad.

4.4.1 Descripción del Webmin Cluster

Clustering con Webmin

Como se dijo anteriormente Webmin se encarga de la administración de equipos terminales o servidores a través de una interfaz web de fácil acceso donde sólo es necesario tener un cliente con un explorador Web para poder acceder al servicio. Para poder usar el módulo Webmin cluster y monitoreo de servicio en cualquier servidor, sólo es necesario instalar el binario de Webmin, ya que la instalación inicial contiene los módulos necesarios para configurar un cluster (para conocer más a detalle la configuración de Webmin puede revisar el Apéndice 1 de esta tesis).

Módulo de Sistema y estado del servicio

Para poder contar con un servicio de alta disponibilidad es necesario implementar mecanismos que permitan recuperar el servicio de forma automática en caso de falla, ya que de no ser así, el servicio dejaría de estar disponible por un tiempo considerable debido a que el tiempo en que se detecta la falla y la corrección de la misma es muy grande, por esta razón es recomendable contar con un servidor de respaldo o Failover que proporcione el servicio de DHCP mientras se repara la falla en el servidor principal. Para realizar un monitoreo del servicio de DHCP y brindar un mecanismo de Failover automático se utiliza el módulo de sistema y estado de servicio.

El módulo de Sistema y Estado de servicio permite monitorear el estado de varios servidores y demonios activos de un sistema, en el cual se puede ver de forma simplificada que servicios están activos en forma gráfica simplificada, al igual que los que no están funcionando o presentan fallas. Este módulo tiene la capacidad de verificar el estado del servicio de forma periódica, es decir, es posible definir horarios fijos para realizar el monitoro del servicio, y en caso de falla notificar al administrador

vía correo electrónico o ejecutando un comando en caso de pérdida de un servicio. Esto resulta útil en sistemas de alta disponibilidad, como es el caso del servicio DHCP, ya que el servicio de Webmin puede monitorear demonios que corren en otros servidores, no solamente en donde se tenga instalado el servicio.

El monitoreo del servicio ofrece dos formas para realizar el monitoreo de los servicios:

La primera es estableciendo una conexión TCP o HTTP al puerto donde el servidor está corriendo el recurso o comunicándose con el servidor Webmin de forma remota y preguntarle el estado del servicio DHCP, y la segunda es utilizando el método de conexión con Webmin éste es más poderoso debido a que puede ser usado para monitorear demonios que no aceptan conexiones de red.

Cada servidor o servicio que se quiera monitorear usando el módulo de monitoreo de Webmin debe tener un monitor definido, cada monitor tiene configuraciones diferentes que indican el tipo de servicio o estado que se va a monitorear. Un monitor puede ser ejecutado fuera del sistema en que se tiene instalado el módulo, es decir, en otro servidor que este corriendo Webmin, en este caso es necesario que el servidor se encuentre definido en el módulo *Webmin Servers Index* del cual se hablará a detalle posteriormente.

El monitor Webmin para DHCP toma parámetros de configuración propios del servicio DHCP para poder ejecutarse, por lo que es necesario ajustarlos para evitar conflictos con los parámetros de configuración del sistema, es decir si el servicio DHCP se instaló en un directorio diferente a los que se utilizan de forma estándar en LINUX, las configuraciones del módulo para el DHCP deben ser ajustadas para usar las rutas correctas, porque de lo contrario el monitor no sabrá dónde buscar el PID del servicio de DHCP.

Cuando se accede al módulo de Sistema y Estado de servidores, la tabla inicial muestra los monitores configurados algunos monitores para algunos servicios estarán configurados por defecto, pero es posible editarlos o borrarlos de acuerdo a los requerimientos definidos. En la figura 4.10 se muestra un ejemplo de la pantalla inicial de estado de los monitores.

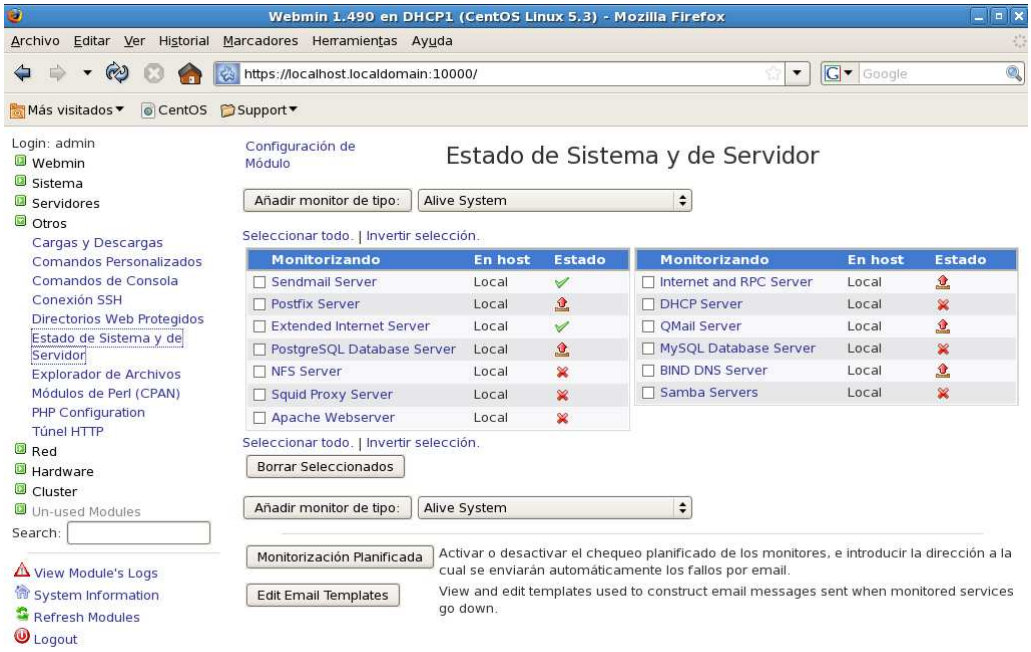


Figura 4.10 Ejemplo de estado de monitores Webmin

Cada monitor está compuesto de una descripción del servicio Webmin que se está monitoreando y el estado en que se encuentra el servicio. Un monitor puede estar en los siguientes estados:

- *Up:* significa que el servicio o servidor está operando de forma correcta, este estado es indicado por una paloma verde en el servicio.
- *Down:* Significa que el monitor está caído, este estado es indicado por una X de color rojo.
- *Webmin down:* Significa que Webmin en los servidores remotos esta caído, y por lo tanto el monitor no puede activarse, este estado es indicado por una W de color rojo.
- *Timed out:* Indica que el monitor no regresó una respuesta en 60 segundos, quizá porque ejecuto un comando que nunca pudo ser completado, es indicado por un reloj de color rojo.
- *Not installed:* Significa que el servidor que está siendo monitoreado no está instalado en el sistema, este estado es indicado por un círculo negro atravesado por una línea.

Por defecto, el estado de cada monitor es requerido cada vez que se accede a la página inicial del módulo, por lo que algunas veces puede tomar un poco de tiempo si se tienen varios monitores o si se está verificando el estado de varios servidores remotos, para evitar esto hay una opción de configuración en el módulo que puede ser utilizada para mostrar el estado de el último estado.

Módulo de índice de servidores Webmin

Para realizar el monitoreo de otros servidores del cluster y hacer copias programadas de archivos entre nodos, Webmin verifica que los servidores se encuentren dentro de una lista de servidores conocidos, ya que de no ser así, no podrán realizarse acciones sobre los externos. El módulo índice de servidores Webmin es el encargado de crear el índice de servidores maestros de Webmin, es decir, crea un índice de los servidores Webmin que están corriendo en la misma red, los cuales son mostrados como un icono que al dar clic sobre él permiten acceder de forma remota al servidor, cada nodo puede ser configurado como una liga normal o un túnel que permite autenticarse en otro servidor automáticamente enviando el tráfico desde el servidor inicial. El módulo también puede ser usado para definir sistemas que pueden ser controlados por un servidor Webmin maestro usando el módulo de monitoreo de sistemas mencionado en el punto anterior uno de los requerimientos para agregar servidores en esta tabla es que cada uno de los sistemas debe tener Webmin instalado y un protocolo RPC definido usado por el primario para comunicarse con los secundarios.

El motivo principal para utilizar este módulo es crear un índice de servidores para administrar el servicio de DHCP de forma centralizada en caso de tener más de dos servidores es muy útil debido a que desde un servidor Principal se pueden administrar servidores dentro de la misma red que se encuentren en una ubicación geográfica a la que no es posible acceder físicamente.

Copia de archivos en módulo cluster

El módulo de cluster Webmin tiene muchas funcionalidades que permiten desde ejecutar comandos de consola simultáneamente en los miembros de un cluster, instalar paquetes de forma simultánea, configurar usuarios y grupos del cluster, hasta

realizar copias de archivos de forma remota, este último en especial es muy útil para la implementación de alta disponibilidad del servicio DHCP ya que hay un sub-módulo llamado copia cluster el cual permite copiar archivos de forma automática a los miembros del cluster o de forma manual solamente dando un clic para hacer la copia.

La copia de archivos es una poderosa herramienta para hacer un servicio de DHCP redundante sin necesidad de usar mecanismos tan complejos como en el caso del Failover. En el servicio de DHCP como se mencionó en el capítulo 3 hay dos archivos principales para la operación del servicio, el archivo *dhcpd.leases y el dhcpd.conf* estos contienen toda la información relacionada con el control de préstamo de direcciones y los parámetros de configuración del servicio, por lo que si se instalaran dos servidores de DHCP, con la ayuda de este módulo es posible hacer modificaciones en uno de estos, que llamaremos nodo primario y tener otro servidor secundario al cual se realizará la copia de estos archivos, esto con el fin de construir un Failover de forma manual, es decir realizar la copia del archivo *dhcpd.leases* y el *dhcpd.conf* de forma automática al secundario para así tener un servidor de respaldo que pueda brindar servicio de DHCP en caso de que el servidor primario falle.

En los siguientes puntos se hablará de las configuraciones necesarias para realizar un Failover usando los módulos nombrados anteriormente.

Webmin RPC

Una parte muy importante de la arquitectura de cluster Webmin son los RPC (*Remote procedure Call en inglés*), ya que estos permiten acceder a cualquier archivo o ejecutar de forma remota cualquier comando como usuario root o admin que son los únicos que pueden hacer llamadas a los RPC.

El protocolo RPC que utiliza Webmin para controlar los elementos del cluster es exclusivo de Webmin y no está basado en protocolos similares, como los Sun RPC, SOAP o RMI. Tiene dos diferentes modos, el modo clásico donde sólo se utilizan HTTP request para enviar comandos y el más nuevo que utiliza conexiones TCP permanentes. El método HTTP es más rápido y eficiente, pero puede fallar si existe algún firewall que bloque el tráfico y algunas veces produce errores entre el maestro y

el secundario. El TCP utiliza el puerto 10001 hacia arriba de forma estándar, a diferencia del clásico que utiliza solamente el puerto 10000 que es el puerto estándar de Webmin.

Las versiones de Webmin superiores a la 0.82 tienen funciones similares para ejecutar código en los servidores Webmin remotos. Son utilizados por algunos módulos estándar, como el de copia de archivos y monitoreo para controlar varios servicios desde una interfaz única, estas funciones las cuales en el código empiezan con `remote_`, permiten llamar funciones para ser evaluadas en código Perl y transferir los datos desde y para un servidor Webmin. Antes de que un servidor maestro pueda hacer una llamada RPC a un host o servidor remoto, tiene que ser registrado en el *Webmin Server Index* en el nodo principal, el campo *Link type* debe ser establecido en *Login via Webmin* y el usuario y password deben ser establecidos. El usuario especificado debe ser root o admin, ya que si no son aceptados en las llamadas RPC.

4.4.2 Instalación y Configuración de Webmin Cluster

Para configurar el servicio de clustering en Webmin se asumirá que ya se tiene instalado Webmin en dos servidores, uno para el primario y otro para el secundario, así como las configuraciones del servicio DHCP y los ajustes iníciales de Webmin, de no ser así revisar la guía de instalación del Anexo 1.

Configuración de monitores

Para controlar el estado del servicio de DHCP es necesario agregar un monitor del servicio, en este caso es recomendable utilizar dos monitores, uno para el servidor, que será un monitor de red con ICMP utilizando el monitor remote ping, y otro para monitorear el servicio propiamente o en el caso de Linux el demonio de DHCP (dhcpd).

El *remote ping* envía y recibe paquetes ICMP para determinar si un host esta activo o inactivo, este es muy útil para probar la conectividad de la red y la disponibilidad de un servidor tiene dos parámetro principales el *Host ping* que es la dirección IP o el hostname del sistema que se quiere verificar, si el host falla al responder el ICPM echo-

request el monitor fallará, y el *time to wait for response* que es el número de segundos que el monitor debe esperar por un ICMP echo reply.

Una vez que se han determinado los monitores a utilizar, se realizan los siguientes pasos:

1. Seleccionar el monitor ping y dhcp que son los que se desea configurar.
2. Llenar el campo de *Description* con la descripción del monitor, dhcp o ping según sea el caso
3. Para tener el monitor ejecutándose en otro servidor Webmin seleccionar Run on host.
4. Para ejecutar un comando cuando el monitor se cae ingresar en el campo *si el monitor se cae ejecutar el comando* y poner service dhcpd start que arrancará el servicio de DHCP en el otro nodo en caso que el servicio se caiga, en caso del monitor por ping se colocará el mismo comando ya que el fin de ambos es arrancar el servicio en el nodo Failover de forma automática tras una falla en el primario.
5. En el campo si *el monitor se levanta ejecutar el comando*, se pondrá el service dhcpd stop ya que quiere decir que el servicio en el primario se ha restablecido.

Los monitores para ambos servicios deben quedar como en la figura 4.11 y 4.12 respectivamente.

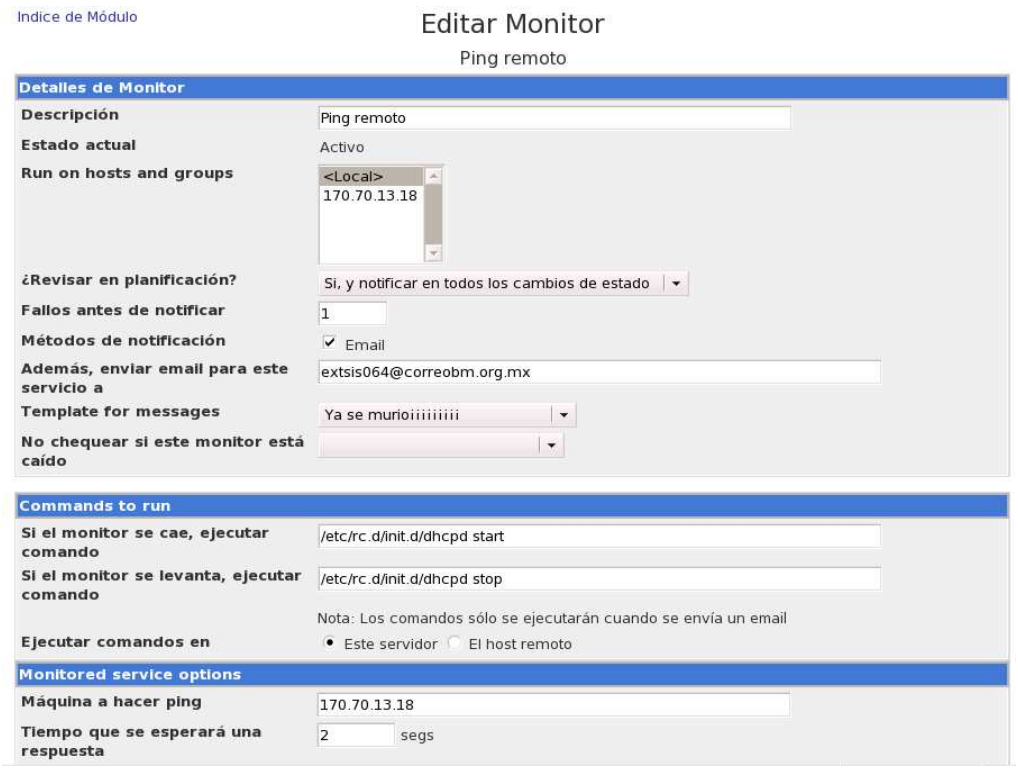


Figura 4.11 Configuración del monitor ping remoto

Figura 4.12 Configuración del monitor DHCP

Hasta este punto la configuración de los monitores debe funcionar correctamente aunque la configuración del servicio de alta disponibilidad Webmin no está completa debido a que no se han configurado las copias programadas de cada uno de los archivos importantes del DHCP.

Configuración de copias programadas

Debido a que para el correcto funcionamiento del servicio DHCP se necesita de dos archivos básicos como mencionamos anteriormente, es necesario replicarlos a los miembros del cluster de forma periódica con la finalidad de que se mantengan actualizados para que en caso de falla puedan funcionar sin problemas y de la forma más actualizada posible, estos tiempos de actualización pueden variar dependiendo de las políticas establecidas por cada organización, ya que a un menor tiempo de replicación generará mayor carga en la red, por lo que se debe analizar principalmente dos factores:

1. La frecuencia con la que se realizan modificaciones al archivo de configuración *dhcpd.conf* debido a que si se ingresan equipos de forma periódica, es decir el crecimiento de la red es muy rápido. Se sugiere establecer un tiempo de replicación más pequeño, este podría ser diariamente a la media noche o a lo largo de ésta de forma automática.
2. Que tan necesario es realizar las actualizaciones. En el caso del archivo *dhcpd.conf* si no se realizan cambios con mucha frecuencia, se puede dejar la actualización hasta por una semana y no tener ningún problema, pero en el caso *dhcpd.leases* es diferente debido a que éste archivo es el encargado de controlar el tiempo de arrendamiento de las direcciones dinámicas y la dirección asignada a cada equipo, por lo que si se deja un largo sin actualizar perdería información del cliente y en caso que el servidor secundario requiera renovar las direcciones probablemente tenga un archivo no actualizado que contenga información errónea, es decir que contenga arrendamientos ya expirados y por lo tanto no renovará al equipo la dirección que tenia originalmente por lo que generara conflictos de asignación de direcciones en la red.

Para realizar las copias programadas con Webmin basta con ir al módulo Cluster > copia de archivos para realizar la copia de los archivos para el servidor DHCP, y replicarlos en los miembros del cluster, como lo muestra la figura 4.13.

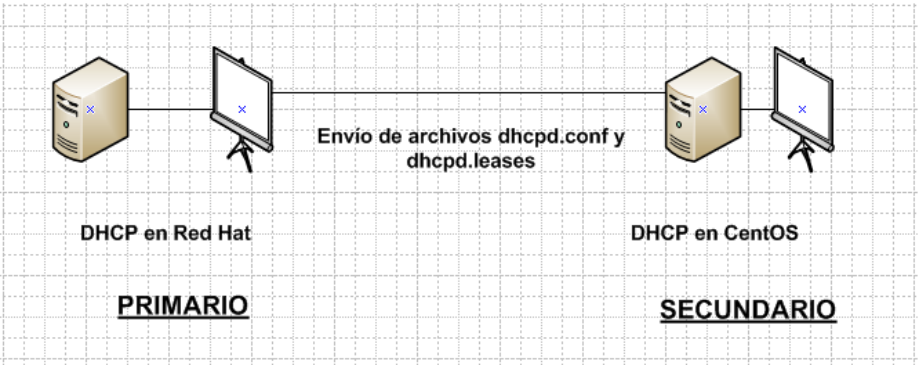


Figura 4.13 Copias programadas

Se requiere realizar una copia de archivos de forma manual del archivo *dhcpd.conf* del servidor primario hacia el servidor secundario, se considera en este caso que será de forma manual ya que sólo en caso de realizar alguna modificación en el archivo *dhcpd.conf* del primario se ejecutará esta tarea, en caso que ninguna modificación se realizada en este tiempo no tiene caso estar enviando este archivo de forma automática.

Para el caso del archivo *dhcpd.leases* se requiere realizar una tarea de manera automática del servidor primario al secundario, esta tarea se realizará como se dijo anteriormente todos los días, esto es porque se tiene que estar actualizando constantemente. Si el servidor primario falla automáticamente arrancara el servidor secundario y este archivo debe estar igual o por lo menos lo más parecido al del servidor primario para que cuando el secundario entre en funcionamiento no exista conflictos con las direcciones IP.

Configuración de las tareas para los archivos dhcpd.conf y dhcpd.leases

Antes de realizar estas tareas es necesario agregar el servidor Webmin del secundario en *índice de servidores Webmin,* (esto es para funciones de gestión para todos los equipos al alcance de Webmin, permitiendo crear tareas, cambiar usuarios, ejecutar comandos y copiar ficheros). Estos equipos se definen dentro de Índice de servidores

Webmin, sólo necesitamos introducir la dirección IP del servidor a registrar, usuario y clave.

Ir a Índice de servidores Webmin y dar clic en “Registrar un nuevo servidor” como muestra en la figura 4.14

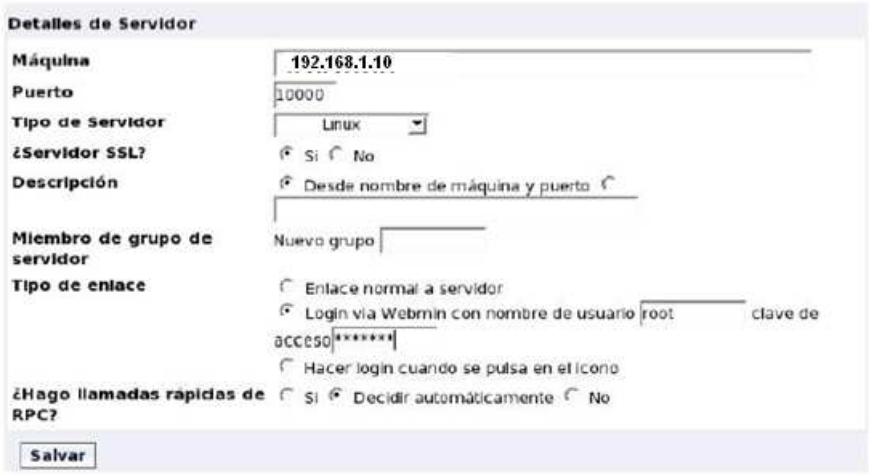


Figura 4.14 Registrar nuevo servidor Webmin

Aparecerá un icono y la dirección IP del servidor, al darle clic podremos acceder al equipo secundario.

Ahora es necesario crear las tareas programadas antes mencionadas, para esto ir a Cluster>Copia de archivos, una vez dentro de la pantalla de configuración se deben ingresar los campos siguientes:

- *Fichero a copiar:* es el nombre y la ruta del archivo que se desea copiar ej. /etc/dhcpd.conf
- *Directorio de destino:* El directorio en que será copiado el archivo ej: / que es la raíz.
- *Copiar a servidores:* Este campo permite seleccionar a los servidores destinos, es decir, los servidores donde se realizará la copia. En esta lista sólo aparecerán los servidores que se agregaron previamente en el índice de servidores.
- *Planificación horaria:* aquí se definirán las horas y días en que se realizará la copia, en caso de querer hacer la copia manualmente seleccionar en las opciones “No”.

La configuración para el archivo de configuración *dhcpd.conf* debe ser similar a la mostrada en la figura 4.15.

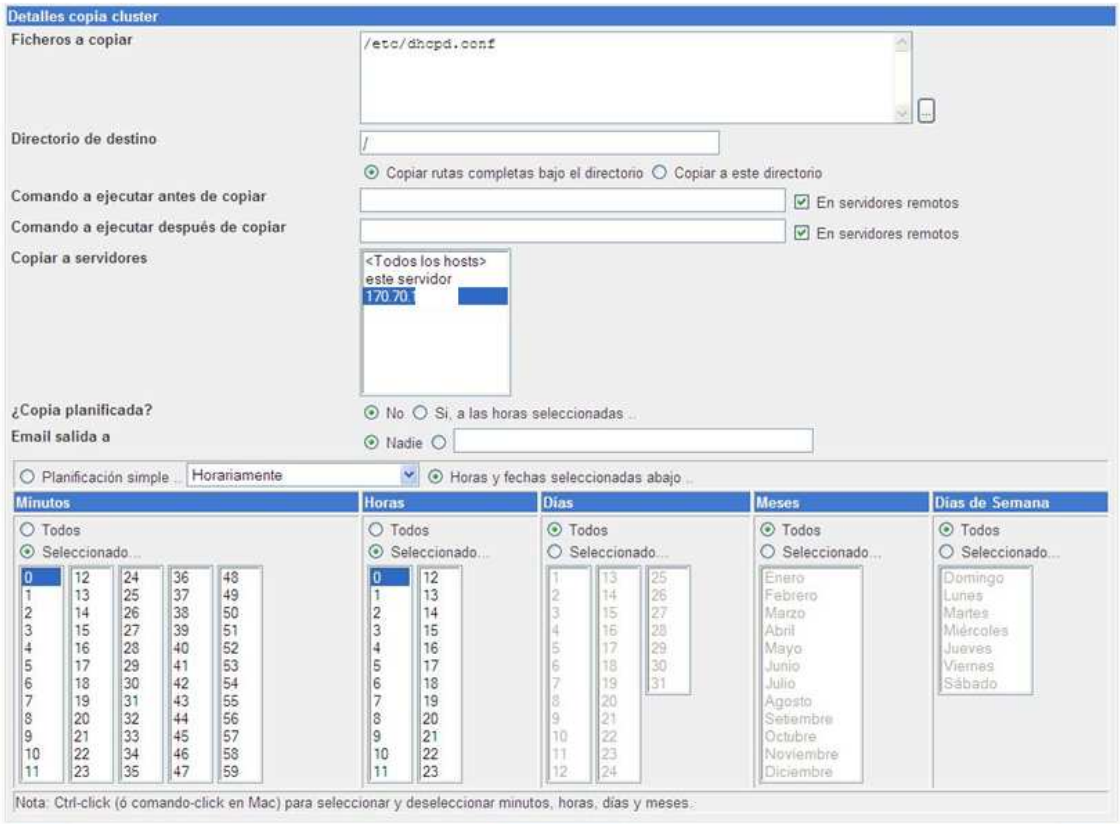


Figura 4.15 Copia programada de archivo dhcpd.conf

De la misma forma se realiza la configuración de las copias programadas en el archivo de préstamo de direcciones *dhcpd.leases* (sólo que en el caso de este archivo se programará la tarea de forma automática).

Una vez creadas las tareas deben verse como en la figura 4.16.

Figura 4.16 Copias planificadas

Para ejecutar la tarea manual para la copia del archivo *dhcpd.conf* se realiza dando clic en *ejecutar*.

Pruebas de funcionamiento

Para saber si las configuraciones de monitores y copias programadas se realizaron correctamente se pueden ejecutar dos tareas principales:

1. Detener el servicio dhcpd en el servidor primario con el comando *service dhcpd stop* para probar el monitor del servicio, y verificar que el servicio se inicia en el servidor secundario
2. Interrumpir la comunicaciones entre los servidores desconectando el primario de la red de forma física o utilizando el comando *ifconfig eth"X" down, D*e esta forma se comprueba el buen funcionamiento del monitor ping.

En caso de que no funcione o marque error en alguno de los monitores revisar a detalle la guía de configuración de Webmin en el anexo 1 y verificar que se realizaron correctamente los pasos para configurar los monitores.

En el caso de las copias programadas pulsar en ambas tareas el vinculo "Ejecutar" para probar si la copia se está realizando de forma correcta en caso de error el mismo Webmin indicará el motivo de la falla.

En las copias programadas algunas veces hay fallas en la copia de archivos que no notifica Webmin o copias incompletas que son visibles a la hora de iniciar el servicio en el servidor secundario, esto debido a la configuración de la velocidad de copia de los RPC por lo que se sugiere revisar que en ambos la configuración de la velocidad de los RPC sea *fast,* ya que es el modo en el que no se han detectado fallas en las pruebas de operación.

Conclusiones

Conclusiones

El caso de estudio de la presente tesis fue garantizar la operatividad del servicio DHCP y la reducción de costo del mismo (licencias y mantenimiento).

Para ello:

- Se realizó una comparación entre software libre y software propietario para el servicio DHCP para poder comparar características y determinar si el software libre puede brindar las mismas características que el software propietario y ofrecer el servicio de igual o mejor manera, al igual que la utilización de sistemas operativos Linux que son seguros, estables y de bajo costo. Todo esto para cumplir con el requerimiento de reducción de costos del servicio.
- Se implementó un software libre llamado Webmin que proporciona administración gráfica del servicio DHCP, es de código abierto y es eficiente, además de que permite administrar el servicio de forma más sencilla.
- Para garantizar la continuidad del servicio se probaron tres esquemas de alta disponibilidad del servicio (Webmin, Failover protocol y Linux HA), para compararlos y decidir cuál es la mejor solución cumpliendo los requerimientos planteados al inicio de este trabajo. Para este caso la mejor solución fue la utilización de Webmin y la creación de monitores, pero esto no quiere decir que es mejor que los otros esquemas, si no que para este caso en específico si lo fue. Los otros esquemas brindan una gran cantidad de servicios, pero siempre la mejor solución será seleccionada apegándose a cumplir los requerimientos proporcionados y de la forma más simple posible con el fin de evitar puntos de falla en el sistema.

Por esto se cumplieron los objetivos, ya que se implementó el servicio satisfactoriamente y se encontró la solución que cumple con los objetivos requeridos para la institución financiera.

Además se creó un programa de búsqueda de direcciones IP que no estaba contemplado en los objetivos de este trabajo. El esquema final seleccionado fue

Webmin pero éste, no contaba con una interfaz que permitiera realizar búsquedas de direcciones IP estáticas asignadas o libres por lo que se desarrolló un programa en lenguaje Perl que se incorporó a él, y que realiza consultas de las subredes, direcciones asignadas y libres, fecha y hora de préstamo, así como un historial de asignación de cada dirección, lo cual ayuda al administrador a conocer el estado de las direcciones de forma más simple.

Para seleccionar el esquema final, se realizaron distintas pruebas con cada unos de los esquemas propuestos y así seleccionar el que mejor cumpliera con los requerimientos.

Para verificar el funcionamiento de los tres esquemas propuestos se realizaron se realizaron las siguientes pruebas:

- Detener el servidor primario para comprobar que el secundario detectara su ausencia y empezara a funcionar, de igual manera quitando el cable de red. Se hacían capturas para verificar el comportamiento y funcionamiento. Así como ingresar nuevos equipos que solicitaran una dirección IP cuando el servidor primario estuviera fuera de servicio y el secundario tuviera el control.
- Regresar al servidor primario y que el secundario detectara su presencia y le devolviera le control al primario, de igual manera ingresar nuevos equipos para verificar que el primario tuviera de nuevo el control.
- Interrumpir la comunicación entre los servidores primario y secundario, para observar el comportamiento de ambos en capturas y comprobar su funcionamiento.

El esquema final elegido es un esquema redundante (Webmin), que permite tener el servicio operando, ya que se cuentan con dos servidores y en caso de que falle uno está el secundario que es una réplica del primario.

Como propuesta a futuro, se dejaron establecidas las bases de funcionamiento del servicio DHCP para el protocolo IPv6 para su implementación a futuro, ya que el protocolo IPv4, como se mencionó a la largo de este trabajo tiene algunas carencias, entre ellas el número total de direcciones, por lo que surgió el protocolo IP en su versión 6, el cual cuenta con un número mucho mayor de direcciones totales asignables, sumado a las ventajas de IP versión 6. Para mejora de este proyecto a futuro, se recomienda ampliar la parte de implementación del servicio DHCP en IP

versión 6, ya que aunque la forma de operar de ambas versiones es similar, el protocolo DHCP versión 6 ofrece nuevas funcionalidades.

Glosario

Glosario

Anycast. Identifica a un grupo o conjunto de interfaces de red. El paquete se enviara a cualquier interfaz que forme parte del conjunto o grupo y que sea la interfaz más cercana de acuerdo al protocolo de enrutamiento.

Appliance. Es un dispositivo de hardware que trae preinstalado un software listo para ser usado y cumplir sus tareas pero no puede ser modificado.

API. Una interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Backdoors o puerta trasera. Es un tipo de troyano que se instala en el equipo de cómputo y hace modificaciones para siempre estar ejecutándose y deja un puerto de comunicación abierto para que el equipo quede expuesto a riesgos.

Bootstrap Protocol (BOOTP). Siglas del protocolo Bootstrap. Antecesor de DHCP que asigna direcciones IP de manera estática a equipos clientes.

Broadcast o difusión. Envío de información a todos los nodos de la red.

Berkeley Software Distribution (BSD). Sistema operativo derivado de Unix y con aportes por la Universidad de California de Berkeley

Bugs. Falla o error que de un elemento de software que provoca el mal funcionamiento de un programa y hace que funciona incorrectamente.

Cluster Consensus Membership (CCM). Brinda una organización simplificada de la topología del cluster (nodo-wise) a los componentes del cluster que están en capas superiores de la arquitectura de hearbeat.

Cluster Information Base (CIB). Representación completa del estado y la configuración de cluster en formato XML. Sólo debe haber una CIB maestra en el cluster, las demás CIB que están en otros elementos son réplicas.

Cluster: Un cluster es un conjunto de servidores y recursos que se comportan como uno sólo, es decir como una entidad para proporcionar la capacidad de alta disponibilidad y en su caso el balanceo de cargas.

Cluster Resource Manager (CRM). Procesa todas las transacciones que pasan a través de la capa de asignación de recursos. Realiza tareas de iniciar o parar los recursos.

Coordinador Designado (DC). Entidad que decide cuándo debe ejecutarse un cambio general en todo el cluster.

DDI. Se refiere a la solución comercial que integra en un sólo producto, **D**NS, **D**HCP y administración **I**PAM en un appliance.

Disponibilidad. Capacidad de poder acceder a un recurso cuando se requiera y las veces que sea necesario.

Dispositivos de E/S. también abreviado **E/S** o **I/O** (del original en inglés *input/output*), es la colección de interfaces que usan las distintas unidades funcionales (subsistemas) de un sistema de procesamiento de información para comunicarse unas con otras.

Domain Name Systems (DNS). Sistema que se encarga de traducir nombres de dominio que son más fácil de recordar a direcciones IP y viceversa.

Downtime o tiempo de inactividad. Es el tiempo durante el cual el servicio o la aplicación no están disponibles, se mide a partir de la pérdida del servicio (*outage en inglés)* y termina hasta que el servicio o la aplicación vuelven a funcionar correctamente.

Dynamic Host Configuration Protocol (DHCP). Protocolo de red que ayuda a los administradores para proporcionar configuraciones de red de manera automática a los equipos para que puedan acceder a la red.

Failback. Proceso mediante el cual uno o más recursos de un servidor que fallaron regresan a su operación original.

Failover. Configuración de equipos en modo backup, con lo que garantiza que si el servidor primario falla o se interrumpe su servicio, entra un servidor secundario para seguir proveyendo el servicio, lo cual garantiza la continuidad del servicio.

General Public License (GPL). Licencia de software que permite su libre modificación, distribución y uso. Lo protege contra posibles apropiaciones.

Hearbeat. Paquete de software creado por Linux HA. Envía ping (latido del corazón por su nombre) para verificar que los nodos que lo conforman están "vivos". Estas pulsaciones se realizan por medio de ping y por pulsaciones en el cable.

High Availability (HA). Característica de un sistema que le permite responder y recuperarse ante alguna falla en un periodo corto.

Internet Engineering Task Force (IETF). Organización internacional que regula las propuestas y los estándares de internet. Se encarga del buen funcionamiento de internet. Es una entidad libre por lo que cualquier persona puede participar.

IPAM. Protocolo de Internet de gestión de dirección (IPAM) es el término usado para integrar los conceptos de planeación, seguimiento y gestión del espacio de direcciones de Protocolo de Internet en una red.

Lesser General Public License (LGPL). Licencia de software libre que permite la libertad de ejecutar, modificar, redistribuir y copiar el software, siempre y cuando cumpla con las condiciones indicadas en esta licencia. La licencia LGPL está destinada regularmente a bibliotecas de software a diferencia de la GPL que es más para ejecutables.

Master Cluster Information (CIM). Brinda una organización simplificada de la topología del cluster (node-wise) a los componentes del cluster que están en las capas superiores.

MAC. Identificador de 48 bits, único en el mundo, que identifica la dirección física de una tarjeta de red.

Max Client Lead Time (MCLT). Máximo tiempo que un servidor puede extender un préstamo sin contactar al otro servidor.

Multidifusión. Envió de información en conjunto a múltiples nodos de la red, pero no a todos.

Network Address Translation (NAT). El concepto de NAT consiste en utilizar una dirección IP enrutable (o un número limitado de direcciones IP) para conectar todas las máquinas a través de la traducción, entre la dirección interna (no enrutable) de la máquina que se desea conectar y la dirección IP de la pasarela.

Perl. Lenguaje de programación de libre uso, que toma características similares al lenguaje C, lenguaje interpretado Shell y es usado para extraer información de archivos de texto y generar informes a partir del contendido de los ficheros.

Pool. Es un intervalo de direcciones asignables definidas por el administrador del servicio DHCP.

Score. Es la medida de referencia con la que se evalua un servicio, en este caso DHCP.

Single Ponit of Failure (SPOF). Cualquier elemento crítico en un sistema, que en caso que falle provoque problema en la operación normal o interrupción del servicio.

Red Hat Linux. Sistema operativo con núcleo Linux con licencia GPL. Comúnmente utilizado para servidores.

Software libre. Tipo de software en el cual el usuario tiene la libertad de copiar, distribuir, modificar y ejecutarlo.

Software propietario. Es lo contrario a software libre, es decir no cumple con las libertades básicas como libertad para usar el software, libertad para distribuirlo, libertad para copiarlo y libertad para modificarlo

Shoot The Other Node In The Head (STONITH). Pégale un Tiro en la Cabeza al otro Nodo). Técnica usada por Heartbeat que se asegura de que un servidor supuestamente muerto no interfiera con el funcionamiento del cluster.

Service Level Agreements (SLA). *Acuerdo de nivel de servicio* es aquel que determina el grado de responsabilidades para mantener el servicio disponible, el costo, los recursos, la complejidad del servicio y las penalizaciones en caso de incumplimiento. Si el servicio se cae los usuarios son afectados.

TI. Acrónimo de Tecnologias de Información en inglés **T**ecnology **I**nformation.

Tiempo de reloj. Es el tiempo contado a partir de las horas de reloj, es decir 24 horas

Tiempo de servicio. Es el tiempo contado a partir de las horas laborales, ej. 8 horas + 2 horas de tolerancia

Webmin. Herramienta de administración de sistemas Linux y Unix, con interfaz gráfica y accesible vía web. Escrita en lenguaje de programación perl.

Outage. Cuando un servicio o aplicación no es proporcionado durante periodo específico de tiempo.

Unidifusión. Envío de información punto a punto, sólo un emisor y un receptor.

Uptime o tiempo en servicio: Es el tiempo durante el cual el servicio o las aplicaciones han estado en funcionamiento.

Apéndice AI

Instalación y configuración de Webmin

1- PROGRAMAS NECESARIOS

- Sistema operativo Linux (Para el caso de estudio de este proyecto se utilizaron RedHat9, CentOS y RedHat Enterpraise por lo que se cree que cualquier distribución Linux puede funcionar correctamente.)
- Webmin-1.470.noarch.rpm
- Webmin-tiger-dafault-1.79.tar.tar
- Netools -1.060.1.wbm

2- CONFIGURACIÓN DE SISTEMA OPERATIVO LINUX

Este manual fue elaborado utilizando un sistema operativo RedHat 9.

Ya teniendo instalado el sistema operativo se necesitan hacer algunas configuraciones previas a la instalación de Webmin. A continuación se muestran dichas configuraciones:

a) Configuración de la tarjeta de red

Entrar a *>>Menú principal>Configuración del sistema>Red* y seleccionar la tarjeta de red. Aparecerá una venta y dar clic en "Modificar" (Ver Figura AI.1). Abrirá una nueva ventana en la que debe estar seleccionado "Activar dispositivo cuando se inicie el ordenador y Configurar las direcciones IP de manera estática. " Llenar con los datos correspondientes al servidor (dirección IP manual, máscara de red, puerta de enlace predeterminada). Ver Figura AI.2.

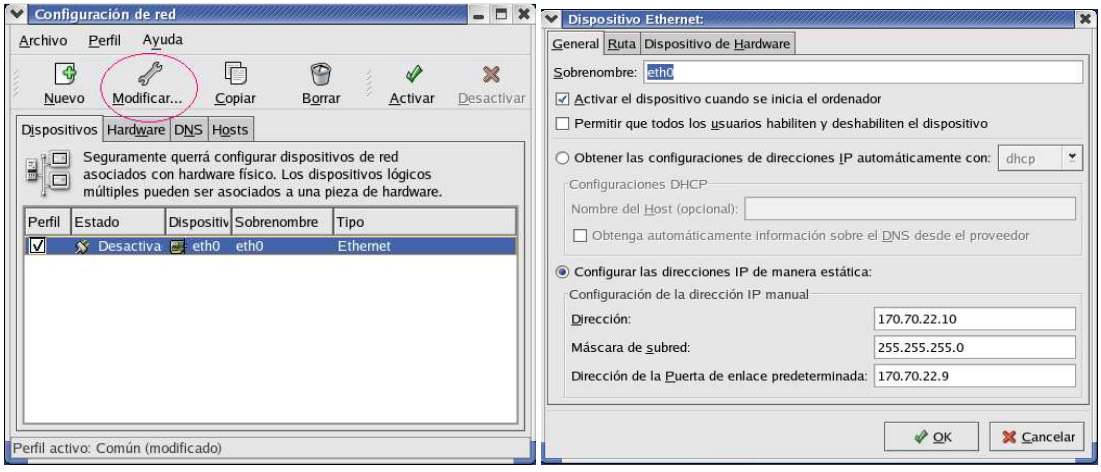


Figura AI.1Configuración de red y AI.2 Dispositivo Ethernet.

Presionar en modificar y configurar la tarjeta de red del servidor.

b) Configuración del firewall

Entrar en *>>Menú principal>Configuración del sistema>Nivel de seguridad*

Presionar "Personalizar" y en Dispositivos Fiables seleccionar eth0 y en Permitir la entrada seleccionar DHCP, SSH y HTTP. (Ver Figura AI.3).

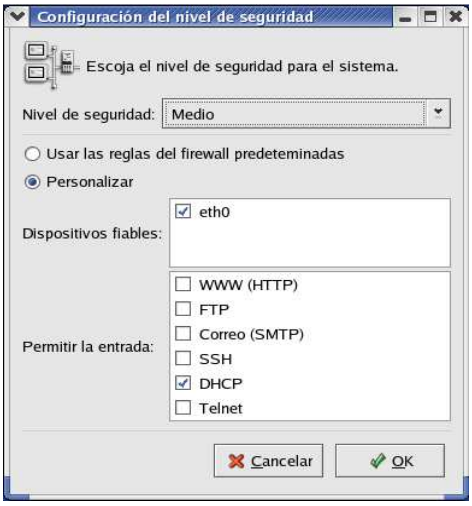


Figura AI.3 Configuración de firewall

Presionar OK para que se guarde la configuración.

3- WEBMIN

Previo a la instalación de Webmin se deben instalar los siguientes paquetes:

- Dhcp-server
- Dhcp-tools
- Openssl-devel
- perl-OPENSSL
- perl-Net_SSLeay
- gcc-java criptix
- kdeblindings java
- puretls

Y desinstalar

- Dhcp-client

a) INSTALACIÓN DE WEBMIN

Para instalar Webmin primero se debe descargar desde la página oficial http://www.webmin.com.

Ya que se descargó, se descomprimirá desde línea de comandos, para ello se introduce el siguiente comando en la terminal, ya que viene en formato .tar.gz.

Comando: tar xvzf Webmin-1.470.tar gz

Después de haberse descomprimido y colocarlo en /opt, se debe cambiar a la carpeta creada de openssl >>cd /opt/Webmin y ejecutar el siguiente comando*:*

./setup.sh

Con este script arranca la instalación e irán haciendo preguntas acerca de su configuración, si se quiere aceptar la propuesta que hace, pulsar intro, de lo contrario se introduce el valor que se desee.

Ejemplo

Config file directory [/etc/Webmin]:

Directorio de configuración del programa.

Log file directory [/var/Webmin]:

Directorio donde se almacenarán los logs de **Webmin**.

Full path to perl (default /usr/bin/perl):

Webmin está escrito en *Perl*, con lo que preguntará la ruta hasta el intérprete. Por lo general es la que traé por default.

For Webmin to work properly, it needs to know which operating system type and version you are running. Please select your system type by entering the number next to it from the list below

1) Sun Solaris 2) Caldera OpenLinux eS 3) Caldera OpenLinux
4) Redhat Linux 5) Slackware Linux 6) Debian Linux
7) SuSE Linux 8) United Linux 9) Corel Linux
10) TurboLinux 11) Cobalt Linux 12) Mandrake Linux
13) Mandrake Linux Corpo 14) Delix DLD Linux 15) Conectiva Linux
16) MSC Linux 17) MkLinux 18) LinuxPPC
19) XLinux 20) LinuxPL 21) Trustix
22) Cendio LBS Linux 23) Ute Linux 24) Lanthan Linux
25) Yellow Dog Linux 26) Corvus Latinux 27) Immunix Linux
28) Gentoo Linux 29) Lycoris Desktop/LX 30) Generic Linux
31) FreeBSD 32) OpenBSD 33) NetBSD

Operating system: 4

Web server port (default 10000):

Este es el puerto al que se debe dirigir el navegador para configurar la máquina con Webmin, se acepta o se introducen el puerto deseado.

Login name (default admin):root

Escribir el login y nombre de usuario para acceder a Webmin.

La instalación va a intentar detectar la librería SSLeay, si el equipo ya tiene instalado OpenSSL, esta librería va a ofrecer la posibilidad de que todos los datos que se envíen a través de la red mediante Webmin vayan de encriptados, de forma que si son interceptados por un atacante no puedan ser leídos de forma directa. Por lo que es recomendable utilizar esta opción.

Use SSL (y/n): y
Start Webmin at boot time (y/n): y

Y se le indica que arranque Webmin al inicio del sistema.

Completados todos los pasos anteriores ya está terminada la instalación, así que ahora se arranca Webmin con el siguiente comando:

linux:/usr/src/Webmin-1.070 # /etc/init.d/Webmin start

Se introduce la siguiente dirección en la barra de direcciones del navegador:

https://localhost:10000

Observe que es https y no http, ya que al ser encriptada la comunicación habrá que utilizar este protocolo. La primera vez que se accede a esta dirección, saldrá un mensaje advirtiendo de que no se puede verificar el certificado de seguridad y preguntando si se desea aceptar. Dicho certificado se generó durante la instalación de OpenSSL, por lo que se necesita aceptar dicho certificado.

Saldrá la imagen de inicio de Webmin ver Figura AI.4. Para ingresar se debe introducir nombre de usuario y contraseña. Ver Figura AI.5.

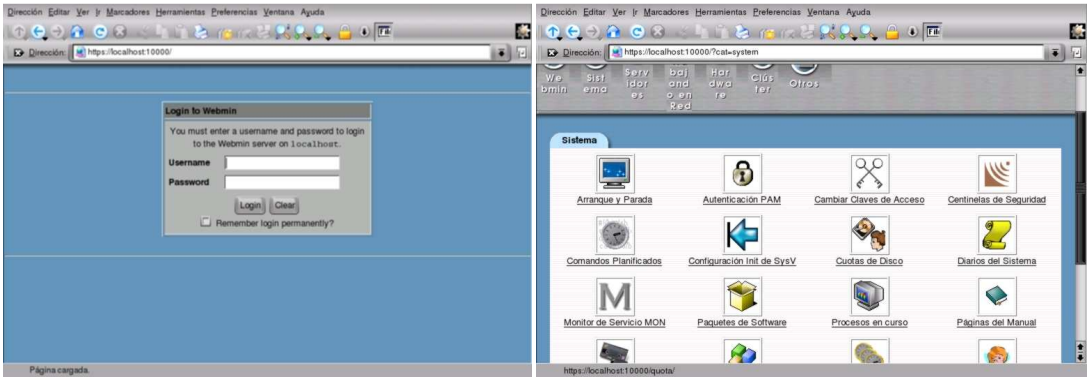


Figura AI.4 Pantalla de inicio de Webmin y Figura AI.5 Configuración de inicio de Webmin.

3.1- CONFIGURACIÓN DE WEBMIN

El siguiente paso es realizar ciertas configuraciones a Webmin, para ello una vez estando dentro dirigirse a *>>Webmin>Configuración de Webmin>Módulos de Webmin* y configurar lo siguiente:

a) Cambio de idioma y tema

Entrar a *>>Webmin>Cambio de idioma y tema.*

En Idioma de UI de Webmin en "Selección personal" se selecciona Spanish (ES) y en "Tema de UI de Webmin" se selecciona tema personal MSC.Linux Theme (Ver Figura AI.6).

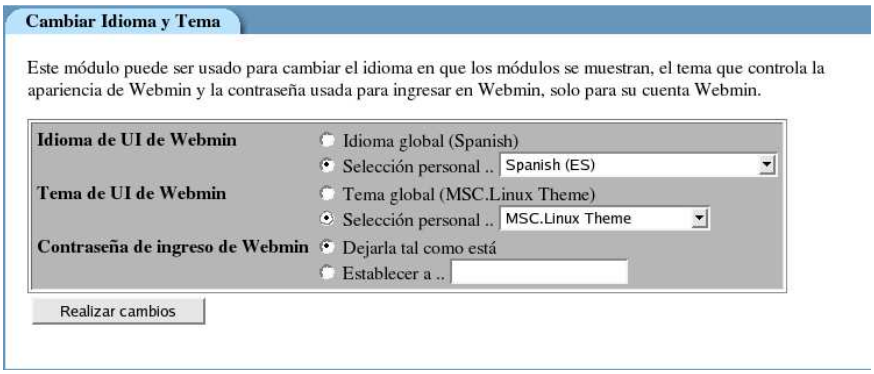


Figura AI.6 Cambio de idioma y tema

b) Opciones de página

>>Webmin>Configuración de Webmin>Opciones de página de índice

Seleccionar tras el "login ", ir siempre al módulo -> Servidor de DHCP (Ver Figura AI.7).

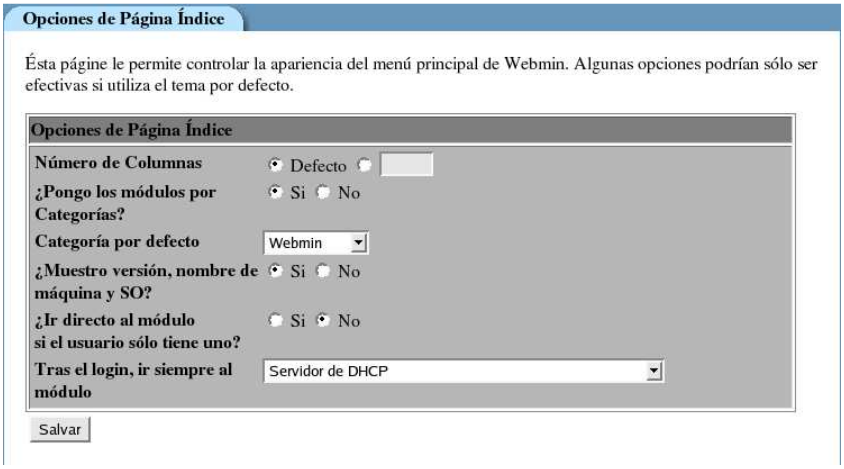


Figura AI.7 Configuración de opciones de página de índice

c) Autenticación

>>Webmin>Configuración de Webmin>Autenticación

Se deselecciona la opción:

¿Ofrezco recordar "login" permanentemente? (Ver Figura AI.8).

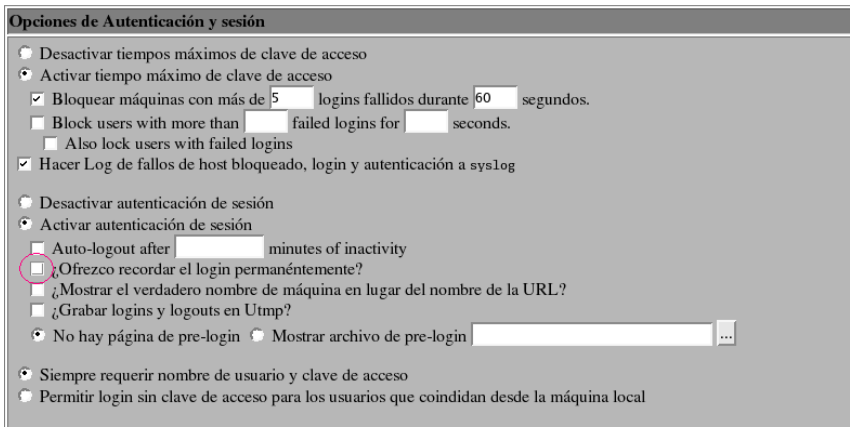


Figura AI.8 Configuración opciones de autenticación y sesión.

d) Editar categorías

Se necesita añadir algunas categorías, para ello dirigirse a:

>>Webmin>Configuración de Webmin>Editar categorías (Ver Figura AI.9)

Y agregar las siguientes categorías una por una. (Ver Tabla AI.1).

Tabla AI.1 Categorías nuevas

ID	DESCRIPCIÓN MOSTRADA
DHCP	DHCP
Failover	Failover
Status	Status
Logs	Logs
Respaldo de DHCP	Respaldo de DHCP

Figura AI.9 Editar categorías.

e) Títulos de módulos

Webmin>Configuración de Webmin>Títulos de módulos

Se renombran los siguientes módulos. Ver Figura AI.10

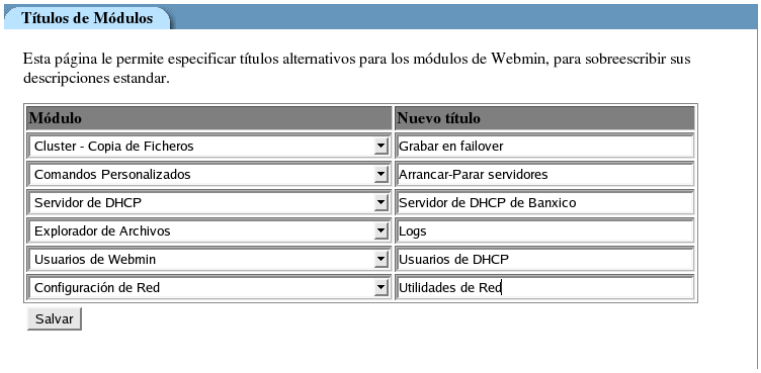


Figura AI.10 Títulos alternativos para los módulos.

f) Reubicar módulos

>>Webmin>Configuración de Webmin>Reubicar módulos

Reubicar los módulos, ver Tabla AI.2

Tabla AI.2 Reubicación de módulos

Módulos reubicados	Categoría
Arrancar parar servidores	Status
Copia de seguridad archivos de configuración	Respaldo de DHCP
Estado de sistema y servidor	Status
Grabar en Failover	Failover
Histórico de acciones Webmin	Logs
Logs	Logs
Tareas planificadas	Logs
Servidores de SSH	Sistema
Usuarios de DHCP	Sistema
Servidor de DHCP	DHCP
Configuración de red	DHCP

g) Encriptación SSL

>>Webmin>Configuración de Webmin>Encriptación SSL

Seleccionar “sí” donde dice:

¿Redireccionar peticiones no SSL al modo SSL?. Ver Figura AI.11.

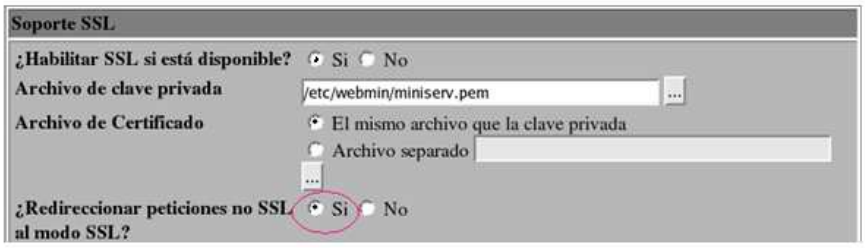


Figura A1.11 Configuración encriptación SSL

Muy probablemente el explorador envíe una petición de conexión mediante SSL, sólo se debe permitir dicho acceso.

h) Usuarios de Webmin

>>Sistema>Usuarios de DHCP

Es recomendable crear grupos de acceso, los usuarios que estén en dicho grupos compartirán los privilegios y/o restricciones del grupo. Se crearán dos grupos: el de administradores y el de usuarios.

h.1) Creación de un grupo de administradores

>Sistema>Usuarios de DHCP

Click en “Crear un nuevo grupo de Webmin”

Nombre de Grupo:=”GrupoAdmin”

Seleccionar los módulos a los que se desea que tengan acceso.

Elegir los siguientes módulos y crear el grupo:

- ✓ Sistema
  - o Servidor SSH
  - o Usuarios de DHCP
- ✓ Status
  - o Arrancar-Parar servidores
  - o Estado de Sistema y de Servidor
- ✓ Respaldo de DHCP
  - o Copia Seguridad Archivos de Configuración
- ✓ Logs
  - o Histórico de Acciones Webmin
  - o Logs
  - o Tareas planificadas (Cron)
- ✓ Failover
  - o Grabar en Failover
- ✓ DHCP
  - o Servidor de DHCP
  - o Utilidades de Red

Ahora regresar al grupo dando click y dar de alta las funciones a las que puedan acceder los usuarios pertenecientes del "GrupoAdmin"; esto se logra dando click en cada módulo. Se explicará uno por uno.

Dar click a:

- *Usuarios DHCP* (Ver Figura AI.12)

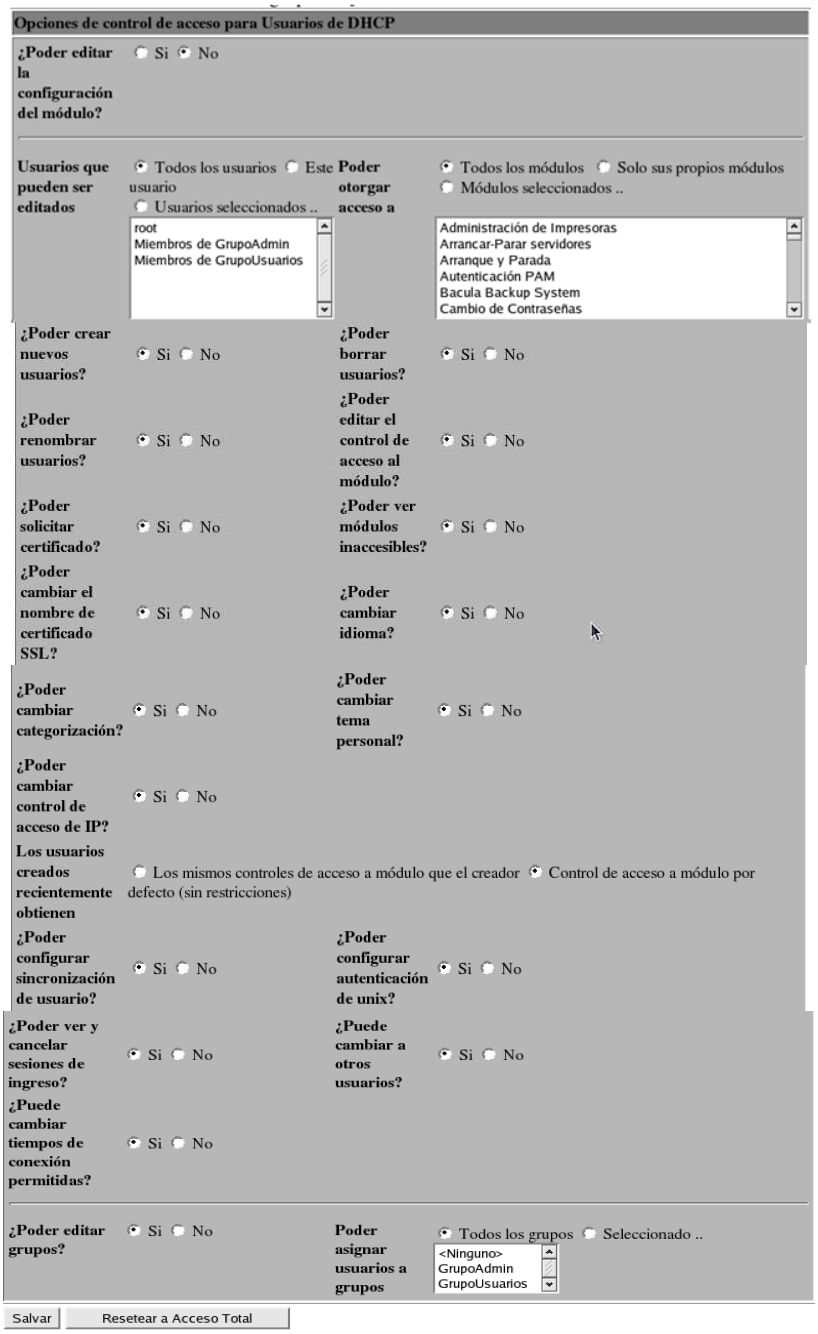


Figura AI.12 Privilegios grupo de usuarios Admin

- *Arrancar parar servidores* (Ver Figura AI.13)

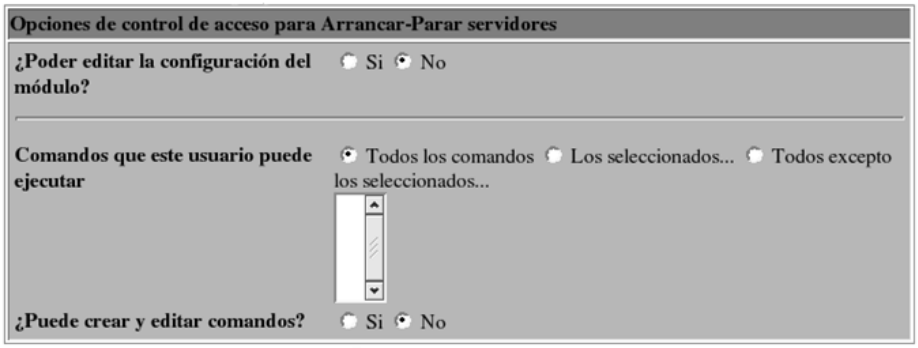


Figura AI.13 Privilegios del módulo Control de Acceso (arrancar-para servidor) para el grupo Admin.

- *Estado de sistema y de servidor* (Ver Figura AI.14)

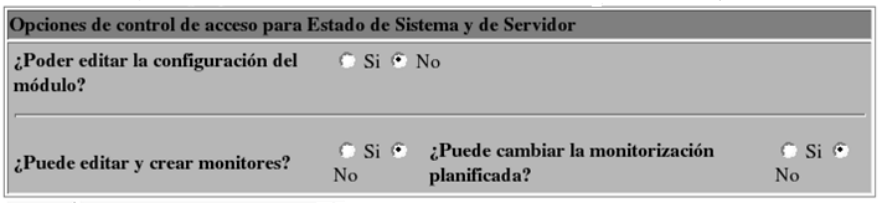


Figura AI.14 Privilegios del módulo Control de Acceso (Estado de sistema y servidor) para el grupo Admin.

- *Copia de seguridad de archivos de configuración* (Ver Figura AI.15)

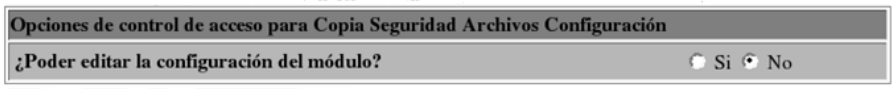


Figura AI.15 Privilegios del módulo Control de Acceso (Copia de seguridad de archivos de configuración) para el grupo Admin.

- *Histórico de acciones Webmin* (Ver Figura AI.16)

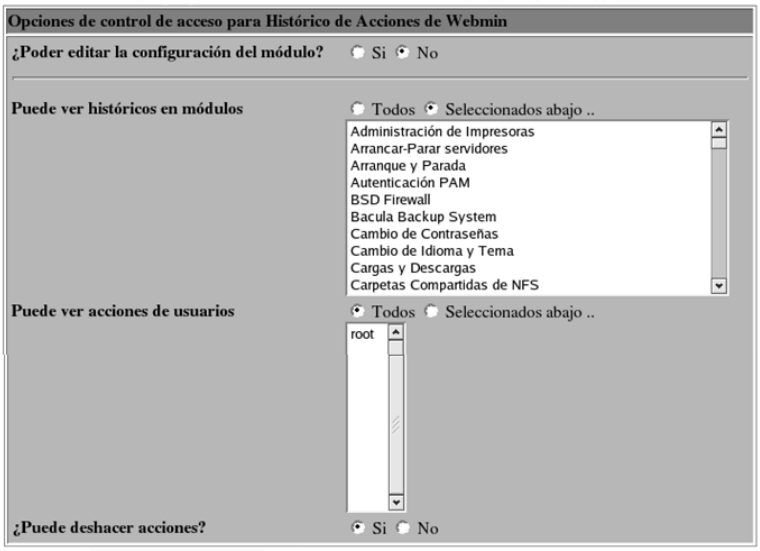


Figura AI.16 Privilegios del módulo Control de Acceso (Histórico de acciones Webmin) para el grupo Admin.

- *Logs* (Ver Figura AI.17)

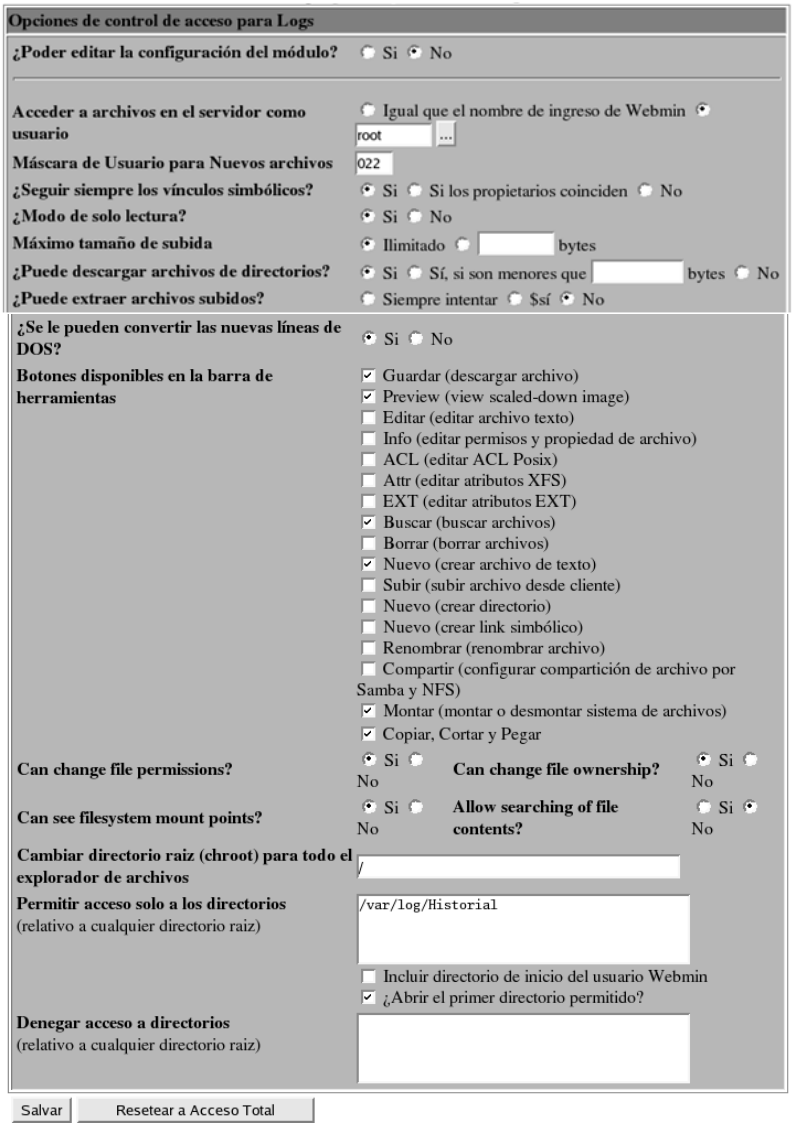


Figura AI.17 Privilegios del módulo Control de Acceso (Logs) para el grupo Admin.

- *Tareas Planificadas Cron* (Ver Figura AI.18)

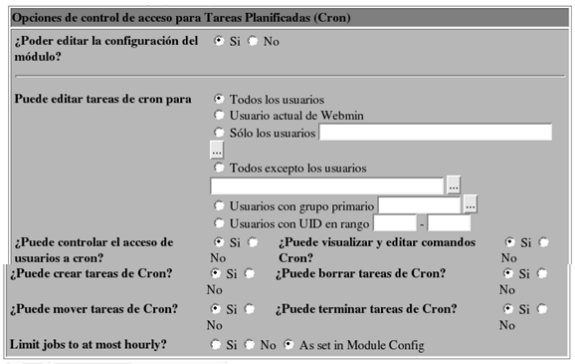


Figura AI.18 Privilegios del módulo Control de Acceso (Tareas planificadas) para el grupo Admin.

- *Grabar en failover* (Ver Figura AI.19)

Figura AI.19 Privilegios del módulo Control de Acceso (Grabar en Failover) para el grupo Admin.

- *Servidor de DHCP* (Figura AI.20)

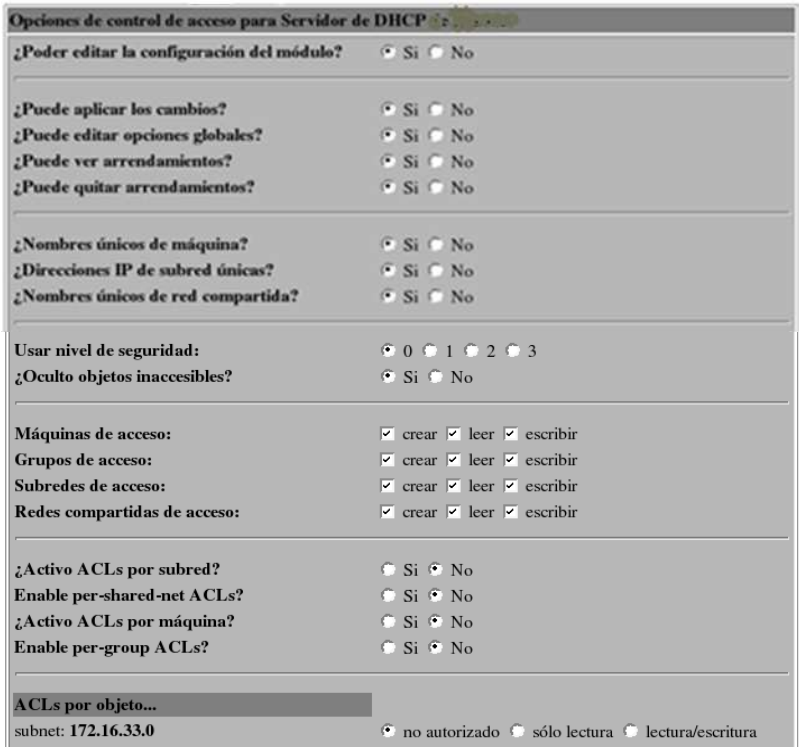


Figura AI.20 Privilegios del módulo Control de Acceso (Servidor DHCP) para el grupo Admin.

- *Utilidades de Red*

No hay cambios, no existen políticas de acceso.

H.2) Creación de un grupo de usuarios

>>Sistema>Usuarios de DHCP

Click en “Crear un nuevo grupo de Webmin”

Nombre de Grupo: “GrupoUsuarios”

Y seleccionar los módulos a los que se desea tenga acceso.

Seleccionar los siguientes módulos y crear el grupo:

- ✓ Status
  - o Estado de Sistema Servidor
- ✓ Logs
  - o Histórico de acciones Webmin
  - o Logs
- ✓ Failover
  - o Grabar en Failover
- ✓ DHCP
  - o Servidor de DHCP de Banxico
  - o Utilidades de Red

Ahora regresar al grupo dando click y dar de alta las funciones a las que tendrán acceso el “GrupoUsuarios”, para ello dar click en cada módulo. Se explicará uno por uno.

- *Estado de sistema y servidor* (Foto AI.21)

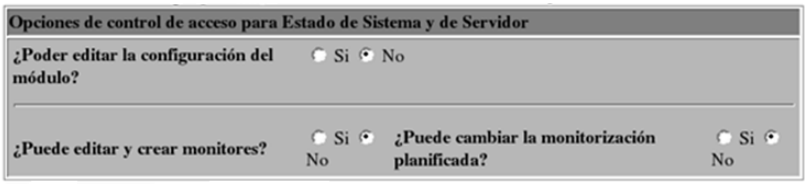


Figura AI.21 Privilegios del módulo control de acceso (Estado de sistema y de servidor) para el grupo Usuarios.

- *Histórico de acciones Webmin* (Ver Figura AI.22)

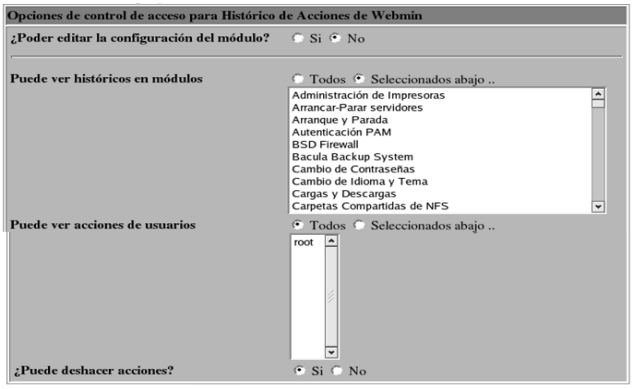


Figura AI.22 Privilegios del módulo control de acceso (Histórico de acciones Webmin) para el grupo Usuarios.

- *Logs* (Ver Figura AI.23)

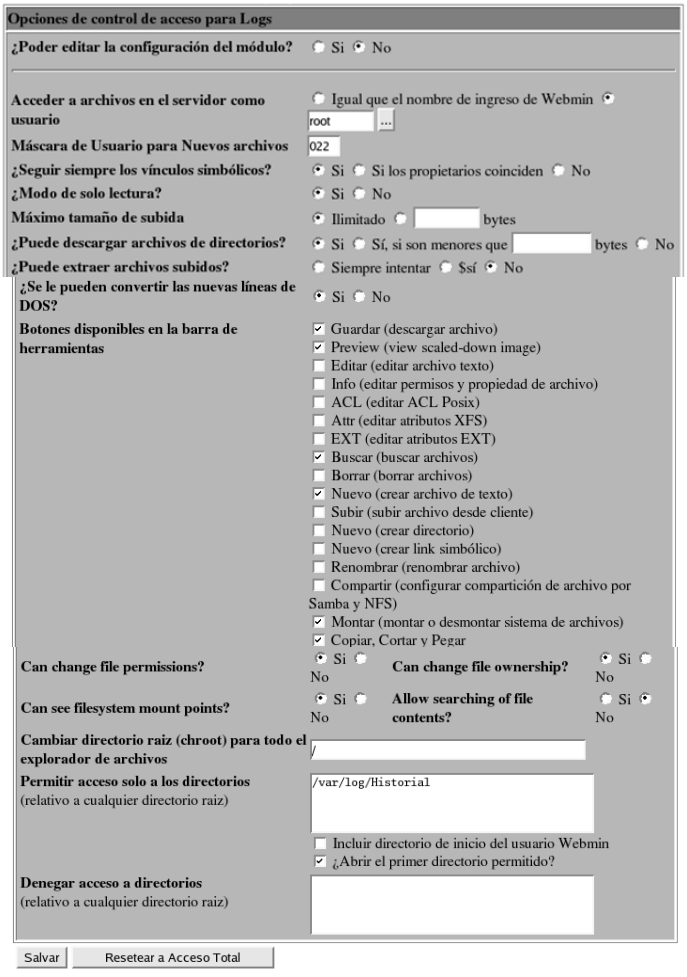


Figura AI.23 Privilegios del módulo control de acceso (Logs) para el grupo Usuarios.

- *Grabar en Failover* (Ver Foto AI.24)

Figura AI.24 Privilegios del módulo control de acceso (Grabar en Failover) para el grupo Usuarios.

- *Servidor de DHCP* (Ver Figura AI.25)

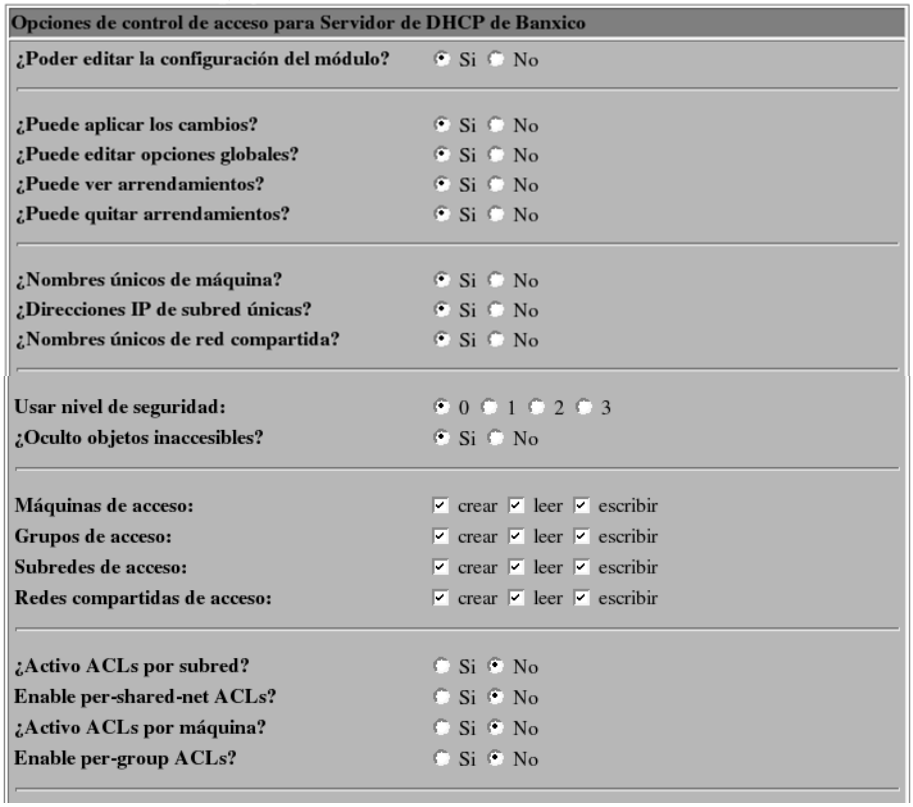


Figura AI.25 Privilegios del módulo control de acceso (Servidor de DHCP) para el grupo Usuarios.

- *Utilidades de Red*

No hay cambios, no existen políticas de acceso.

i) ¿CÓMO CREAR UN USUARIO?

Ahora ya es posible crear dos tipos de clases de usuarios con diferentes privilegios, uno de tipo administrador y otro de tipo usuario. Ver formato para crear un usuario. (Ver Figura AI.26).

Figura AI.26 Creación de un usuario en Webmin.

j) Configuración de módulos

j.1) *>>DHCP>Servidor de DHCP Banxico*

Por default el archivo de configuración de DHCP (dhcpd.conf) trae un ejemplo de cómo crear redes. Es necesario dejar en blanco dicho archivo, para lograr esto entrar en "Configfile" y dejarlo en blanco dicho archivo. (Ver Figura AI.27)

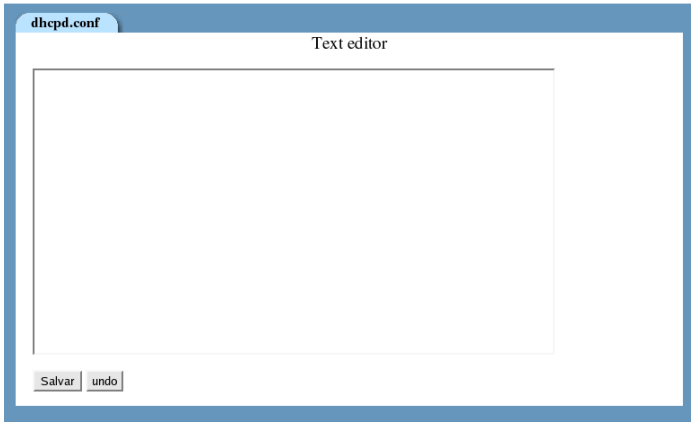


Figura AI.27 Archivo dhcpd.conf en blanco.

Ahora en "Edit Network Interface" seleccionar la interfaz por donde se escucharán las conexiones del DHCP, generalmente es el eth0 (Ver Figura AI.28).

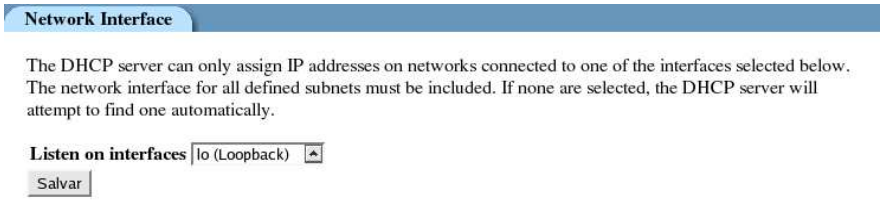


Figura AI.28 Configuración de interfaz de red.

Finalmente entrar en “Configuración del módulo” (Ver Figura AI.29 y AI. 30)

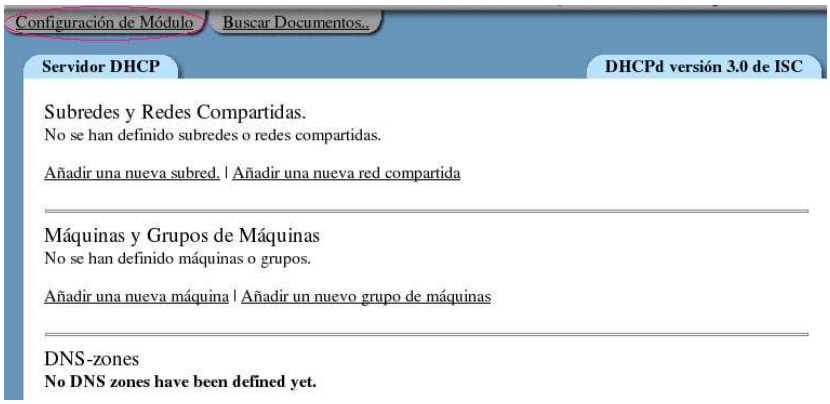


Figura A1.29 Configuración de módulo DHCP.

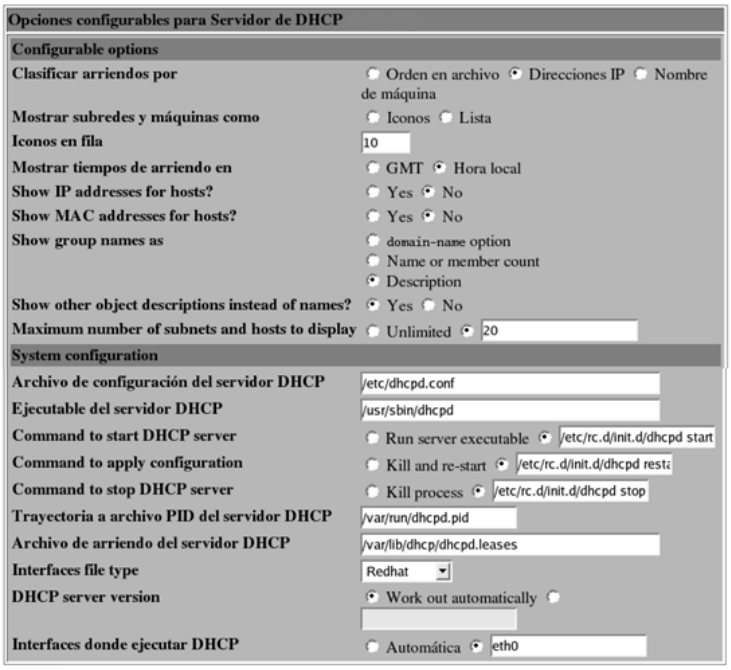


Figura A1.30 Opciones configurables para servidor DHCP

j.2) *>>SISTEMA >SERVIDOR DE SSH*

Entrar a “Autenticación y permitir la autenticación mediante contraseñas” (Ver Figura AI.31).

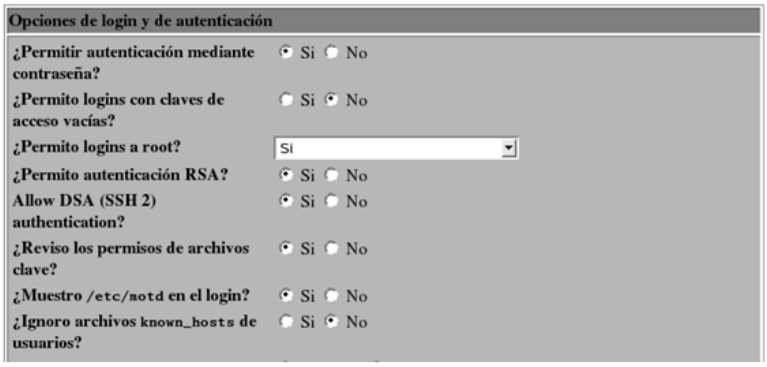


Figura AI.31 Opciones de Login y de autenticación.

h.3) *>>LOGS >HISTÓRICO DE ACCIONES WEBMIN*

Entrar a la configuración del módulo y marcar que muestre el servidor fuente de Webmin como "si" (Ver Figura AI.32 y Figura AI.33)

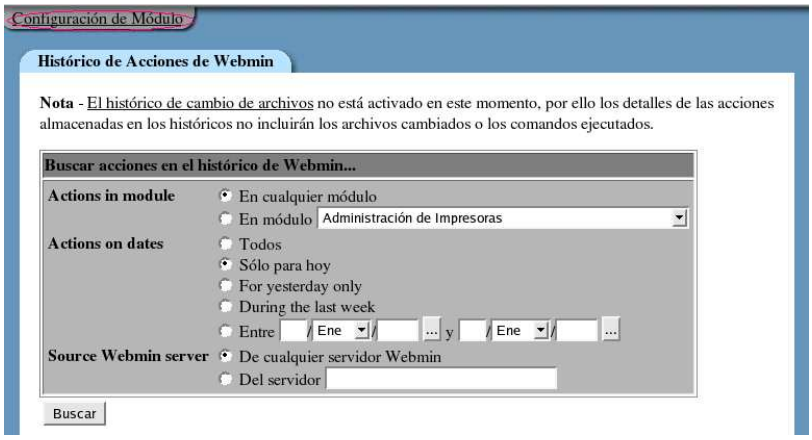


Figura AI.32 Histórico de acciones Webmin

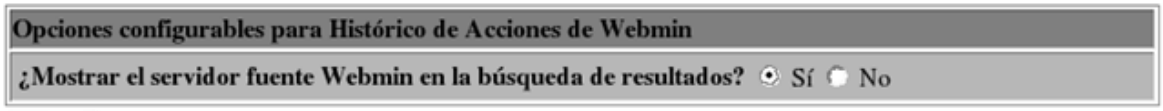


Figura AI.33 Opciones configurables para el módulo de histórico de acciones Webmin.

h.4) *>>LOGS> TAREAS PLANIFICADAS (CRON)*

Abrir una terminal y teclear lo siguiente:

```
dhcplinux@linux:~> sudo su
linux:/home/dhcplinux # mkdir /var/log/Historial
linux:/home/dhcplinux #
```

Ahora crear un archivo directo del archivo “messages” que está en la ruta /var/log. Para eso escribir en la terminal lo siguiente:

```
linux:/home/dhcplinux # ln /var/log/messages /var/log/Historial/Actualmente
linux:/home/dhcplinux #
```

Es necesario dar de alta las tareas en “crear una nueva tarea de cron de catálogo”, lo que dará los logs de paquetes de protocolos de red (AI.34).

Figura AI.34 Crear una nueva tarea de cron de catálogo

Escribir en el campo comando: cp/var/lo/messages/var/log/Historial/semanal1y2 & echo”>/var/log/messages.

¿Qué hace el comando anterior?

Limpia el archivo “messages” después de copiarlo con el nombre de semana1y2 (Ver Figura AI.35).

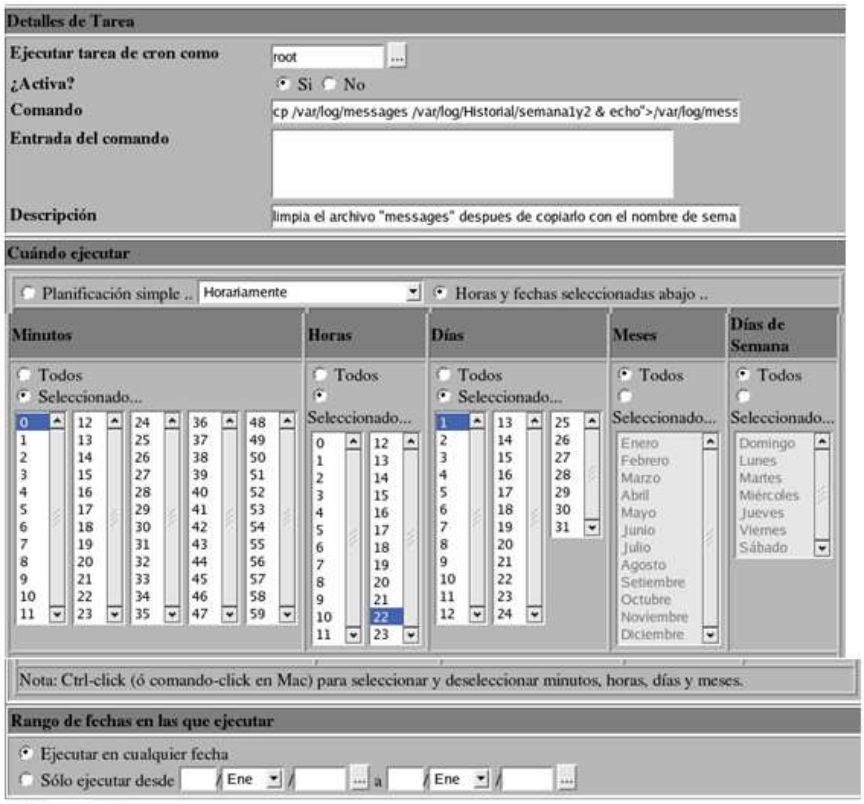


Figura AI.35 Creación de tareas cron

Ahora crear un ahora una nueva tarea y escribir en el campo comando: cp /var/log/messages/var/log/Historial/semana3y4 &echo">/var/log/messages.

¿Qué hace el comando anterior?

Limpia el archivo después de copiarlo con el nombre de semana3y4.

Apéndice AII

Instalación y configuración de Heartbeat y Failover

1 Instalación y configuración de servicio DHCP en Heartbeat versión 2

En este punto se establecerán los parámetros y elementos necesarios para instalar y configurar Heartbeat versión 2 en sistemas basados en Redhat, en esta implementación de Linux-Ha se utilizarán Redhat Linux Enterprise 5 y CentOS 5.

Para la instalación se necesitan los siguientes paquetes:

- Heartbeat
- Heartbeat-pils
- Heartbeat-Stonith
- Heartbeat-GUI

Se sugieren dos métodos para la instalación de paquetes, utilizando el administrador de paquetes del sistema o con el instalador de paquetes RPM, los archivos necesarios pueden descargarse de la página www.linux-ha.org.

Utilizando el administrador de paquetes seleccionamos los paquetes a utilizar como se muestra en la figura AII.1

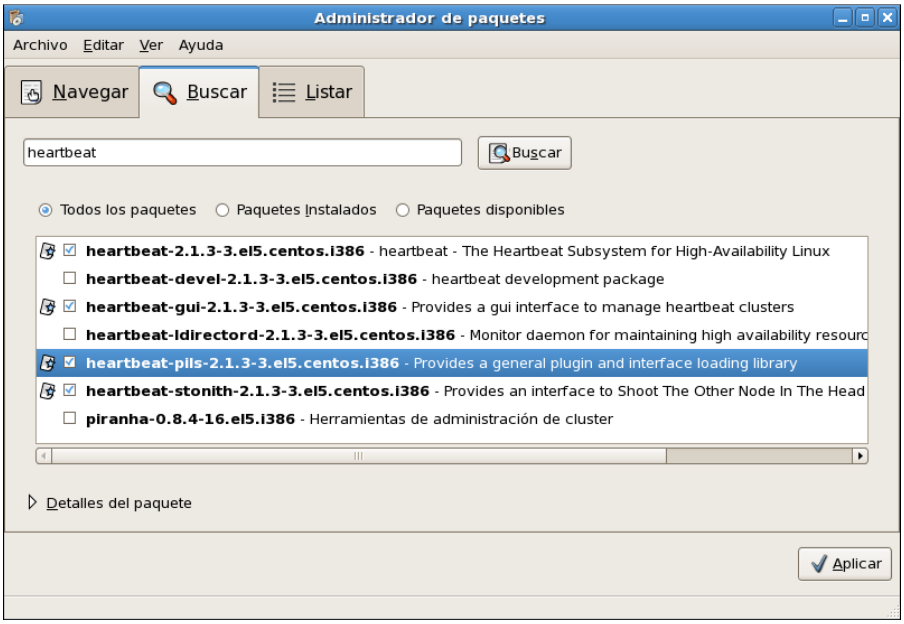


Figura AII.1 Administrador de paquetes CentOS

Para este caso se utilizará la versión 2.1.3 de Heartbeat. Como se mencionó anteriormente utilizaremos heartbeat-pils, PILS es el acrónimo de Plugin and Interface Loading System; PILS fue desarrollado como parte del proyecto Open Cluster Framework, principalmente se encarga de administrar plugins e interfaces para estos plugins.

El paquete Heartbeat-Stonith es un módulo que proporciona la interfaz para hacer fencing a un nodo, es decir de apagar un nodo en un cluster.

Configuraciones iníciales

Las configuraciones iníciales aplican para los diferentes sistemas operativos Linux, en este caso Redhat y centOS para lo cual se llevaran a cabo los siguientes 10 pasos:

1. **Copiar los archivos de configuración y ajustar sus permisos**

```
# cp -pi /usr/share/doc/heartbeat-2.1.3/*.cf /etc/ha.d
# cp -pi /usr/share/doc/heartbeat-2.1.3/authkeys /etc/ha.d
# cp -pi /usr/share/doc/heartbeat-2.1.3/haresource/etc/ha.d
# chmod 600 /etc/ha.d/authkeys
```

Heartbeat es muy preciso con los permisos por lo que hay que asignarlos cuidadosamente ya que de no ser así Heartbeat podría no arrancar.

2. ***Seleccionar las configuraciones básicas***

El primer archivo que debe ser ajustado es el `/etc/ha.d/ha.cf` *En éste pondremos los siguientes parámetros:*

```
autojoin other
node DHCP1 DHCP2
bcast eth1
crm on
```

Poniendo la variable *autojoin* en *other* facilita el proceso de agregar un nuevo nodo al cluster, esta configuración permite a nodos fuera del cluster (no están declarados en el ha.cf) ingresar automáticamente al cluster siempre y cuando se autentiquen.

La interfaz de broadcast en el nodo es eth1, ésta puede cambiar dependiendo del número de NICs o interfaces de red activas, es muy recomendable utilizar una interfaz dedicada solamente a la comunicación heartbeat, pude ser una interfaz Ethernet o serial según convenga, en este caso, como se utilizan plataformas de hardware virtualizadas es más sencillo agregar interfaces de red, teniendo cuidado de usar canales independientes de comunicación es decir usar de preferencia una red distinta para cada NIC.

En el archivo *authkeys* se configurarán la clave que se usará para la comunicación entre los nodos al cual le asignaremos lo siguiente:

```
auth 1
1 sha1 dhcpheartbeat
```

Esto indica que utilizaremos el método de autenticación sha1 con la clave dhcphearbeat.

3. Replicar a los demás elementos del cluster

Después de ajustar los parámetros de configuración del nodo, hay que propagar los archivos a los demás miembros del nodo.

NOTA: los pasos de instalación antes de las configuraciones iniciales se tienen que hacer de forma independiente en cada nodo.

Copiando los archivos al otro nodo:

```
#cd /etc/ha.d
#scp –p ha.cf DHCP2:/etc/ha.d/
# scp –p authkeys DHCP2:/etc/ha.d/
```

4. Iniciar el servicio de Heartbeat

Para iniciar el servicio Heartbeat tecleamos el siguiente comando:

```
#service heartbeat start
```

Hasta este punto si todas las configuraciones anteriores fueron ingresadas correctamente el servicio debe iniciar sin problemas.

5. Agregar Heartbeat al encender el servidor

Por defecto el servicio de Heartbeat no se inicia tras el encendido del equipo, para que el servicio inicie con el sistema operativo automáticamente usamos el comando:

`#chkconfig –-list heartbeat` para conocer el estado del servicio

`#chkconfig –-add heartbeat` si no está lo agregamos

`#chkconfig heartbeat on` finalmente habilitamos el inicio automático.

6. Habilitar password para la GUI

Para usar la interfaz gráfica se tiene que establecer un password, la asignación se puede hacer vía CLI utilizado el comando:

```
#passwd hacluster
```

el cual solicitará el nuevo password para acceder a la GUI de Heartbeat.

7. Iniciar la GUI

Teclear el comando `#hb_gui` el cual lanzara la interfaz gráfica como la figura AII.2

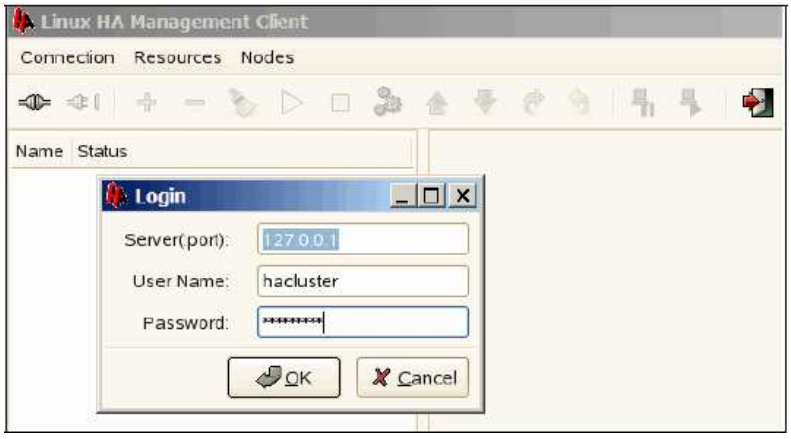


Figura AII.2 Interfaz Gráfica de Linux-HA

8. Configurar Heartbeat para evitar el auto-Failback

Una buena práctica en la mayoría de los sistemas de alta disponibilidad es prevenir el auto-Failback. Esto tiene sentido porque en la mayoría de los casos los administradores prefieren entender la raíz de la falla del Failover antes de permitir el Failback.

Para prevenir el auto-Failback en todos los recursos del cluster, en la GUI de Heartbeat se selecciona el cluster y se da clic en la pestaña *Configurations*, cambiando el *Default Resource Stickness* de 0 a un valor más alto o INFINITY, que significa el valor máximo para este campo como se muestra en la figura AII.3.

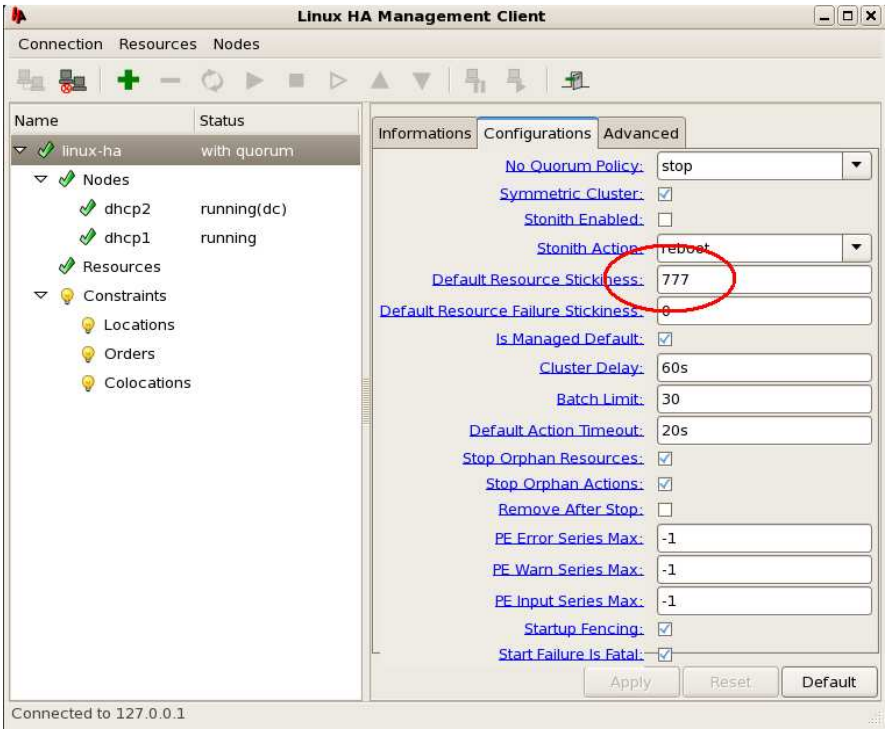


Figura AII.3 Configuración Failback Heartbeat

9. Verificar que hora y fecha se encuentren sincronizadas

Hay que tener cuidado con el ajuste de la hora y fecha del sistema ya que puede provocar que el cluster no funcione correctamente, para ello verificar que los ajustes de hora y fecha coincidan en casa elemento del cluster.

Escenario activo/pasivo con dos nodos para DHCP

En este escenario activo/pasivo se muestra las configuraciones para un cluster de dos nodos y un recurso de grupo, el objetivo es mostrar como el servicio IP se mueve entre los nodos. Las modificaciones se realizaran usando la GUI.

Antes de configurar las opciones del servicio DHCP en Heartbeat hay que verificar si el servicio de DHCP está instalado con el comando:

```
#rpm -q dhcpd
```

Se asume que el servicio se encuentra instalado, así como Heartbeat y sus dependencias, si no es así revisar la parte de configuración e instalación del servicio DHCP y Heartbeat en el capítulo 3 y 4 respectivamente.

Para el servicio de DHCP se considerará el escenario activo pasivo mencionado en el punto 4.2.5, asignándole a cada servidor el nombre de nodo DHCP1 para el primario y DHCP2 para el secundario. Una vez realizados todos los pasos del punto 4.2.6 se definirán los recursos para el cluster utilizando la interfaz gráfica. Para realizar la configuración del servicio hay que recordar que Heartbeat trabaja bajo un esquema de recursos, es decir, cada nodo tendrá asignado un grupo de recursos propios bajo el supuesto de que los recursos de cada grupo son independientes de los recursos de otros grupos de recursos. Algunos recursos del cluster dependen de otros componentes o recursos y requieren que cada componente o recurso sea iniciado al mismo tiempo y corran juntos en un mismo servidor. Un ejemplo de esto es el servidor de DHCP que necesita una dirección IP y el servicio de ISC DHCP como se muestra en la figura AII.4. En este caso cada componente es un recurso del cluster separado que es combinado en un grupo de recursos del cluster, por lo que el grupo de recursos puede correr en un servidor o servidores, en caso de que el hardware o el software fallen, como pasaría en un recurso individual del cluster.

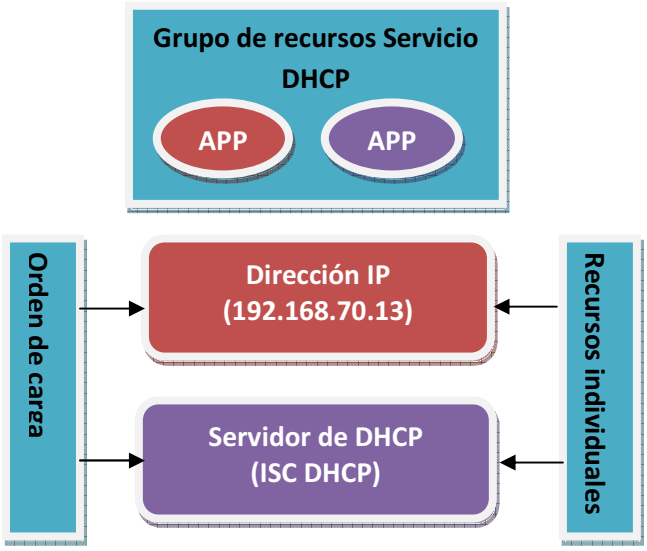


Figura AII.4 Grupo de recursos del cluster

Configuración de recursos

Para establecer los parámetros de configuración de los recursos se utilizará la interfaz gráfica con el comando:

```
#hb_gui
```

1. **Ajustar el Default Resource Stickiness**

Seleccionar la pestaña *Configuration*, y asignar el *Default Resource Stickiness* un número grande como 777 que es en cierta forma equivalente al *auto_Failback* de la versión 1 de Heartbeat, como se muestra en la figura AII.5 y aplicar cambios. Modificando el *Default Resource Stickiness*, se le da una relativa preferencia al nodo para que el recurso permanezca corriendo en éste, el tamaño de este parámetro es determinante para Heartbeat ya que con el determina cual nodo es el mejor para correr un recurso en particular. Como sólo tenemos dos servidores en este escenario y no se ha establecido ningún otro valor, el valor de *stickiness* tiene un efecto mínimo, en este caso se asigna de esta forma por que se desea que el recurso permanezca en el nodo, se dijo que se necesita establecer un valor grande pero a su vez debe ser menor al INFINITY por que la configuración *pingd* usa ajustes de score para determinar

cuando un recurso debe ser movido. Si se pone el valor a INFINITY, el recurso sólo se mueve en el caso que el Heartbeat falle.

2. Configurar el recurso ipaddr

Para agregar un recurso de grupo, damos clic en el botón + de la GUI figura AII.5

Figura AII.5 agregando un recurso

Se selecciona el tipo de recurso, en este caso se requiere un recurso que proporcione una dirección IP flotante, es decir una dirección IP para el recurso, por lo que se seleccionará un tipo native. Figura AII.6.

Figura AII.6 tipo de recurso

Se ingresa la información del recurso con los siguientes campos como se muestra en la figura AII.7:

- Resource ID: resource_IP_A
- Belong to Group: RG_A

- Type: ipaddr

- IP parameter: 192.168.100.10

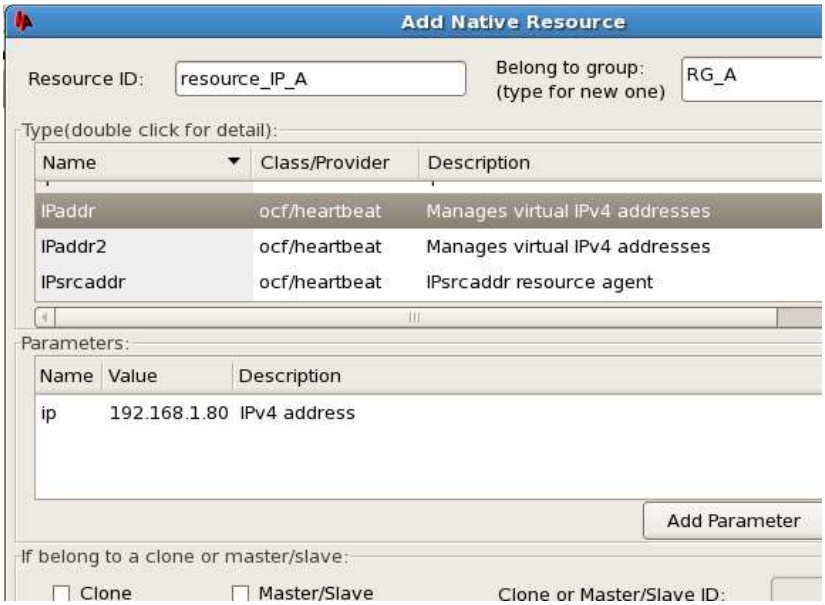


Figura AII.7 Parámetros de configuración del recurso

Para agregar parámetros de configuración adicionales como la NIC dar clic en el botón agregar parámetro y establecer el valor de eth0 que en este escenario es el dispositivo que será conectado a la red en producción la pestaña *Add Parameter* se muestra en la figura AII.8.

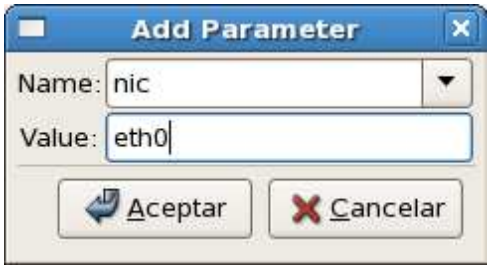


Figura AII.8 Agregar parámetros para la NIC

De la misma forma se agregaran los parámetros para la configuración de la máscara de red, sólo que ahora se seleccionará en el campo Name: el valor cidr_netmask como se observa en la figura AII.9

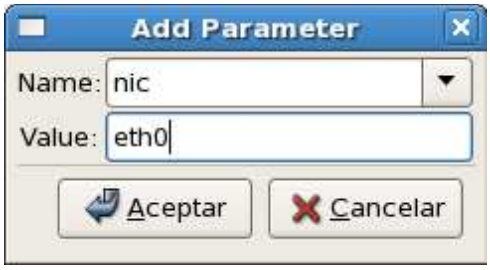


Figura AII.9 Configuración de la máscara de SR

Una vez verificado que toda la información está completa, agregar el recurso al cluster. Dar clic en el botón **Añadir** para ver el nuevo recurso en el panel principal de Heartbeat como se muestra en la figura AII.10.

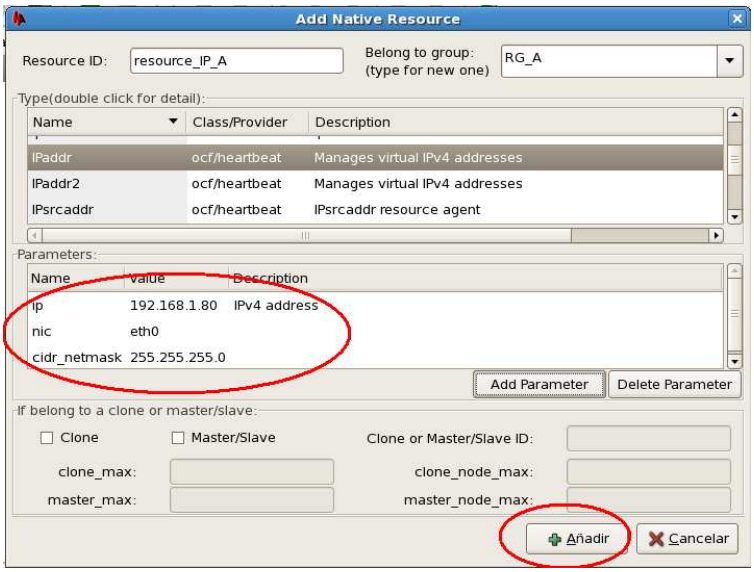


Figura AII.10 Aplicar cambios en recurso

Finalmente para arrancar el recurso de grupo en la GUI de Linux HA seleccionar RG_A y hacer clic en el icono *start resource* (icono del triángulo que está en la barra superior de la ventana), como muestra la figura AII.11.

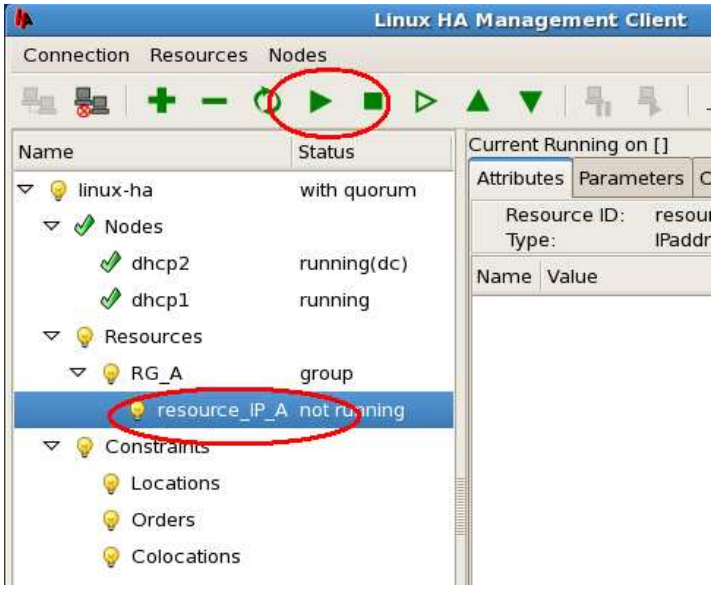


Figura AII.11 Activación del recurso de grupo

Después de iniciar el recurso de grupo la dirección IP 192.168.100.10 es ligada a la interfaz eth0 en el nodo DHCP1 (el nodo activo), para validar que se haya asignado el recurso verificar con el comando `ifconfig.`

Comprobación de migración de recursos del cluster

El cluster ahora está configurado y corriendo. El siguiente paso para una validación básica del funcionamiento es simular el problema en el nodo que el grupo de recursos está corriendo para simular el problema se detendrá el servicio de Heartbeat con el comando

```
# service heartbeat stop
```

Verificar que la dirección IP en el recurso de grupo haya sido movida del DHCP1 al DHCP2 con el comando #ifconfig y con la GUI, en ambas se debe observar el cambio el estado en la GUI se observa en la figura AII.12.

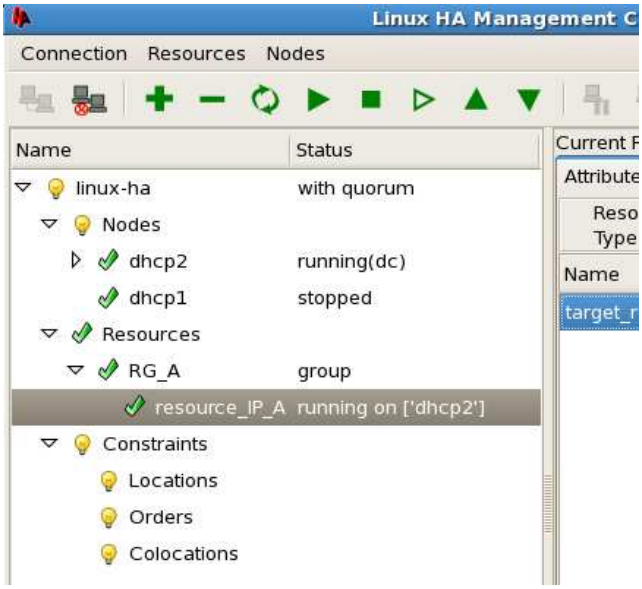


Figura AII.12 Estado de asignación del recurso de grupo

Configuración de DHCP como recurso de grupo

De la misma forma que se configuró el recurso de grupo para la dirección IP, se configurará el servicio de DHCP tomando en cuenta algunas consideraciones.

Para agregar el recurso DHCP se crea un recurso dhcpd tipo Linux Standard Base (LSB) para iniciar y detener el script por defecto en init, que fue instalado durante la instalación del servicio ISC DHCP, se agregara con el mismo nombre de grupo utilizado en el recurso IPaddr, esto se muestra la figura AII.13.

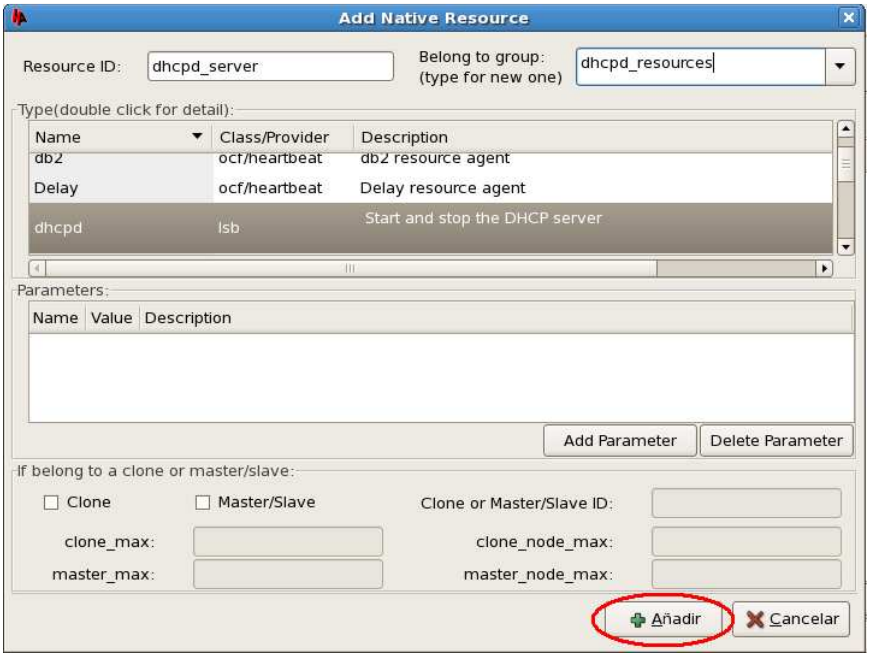


Figura AII.13 Configuración del recurso DHCP

Después de aplicar los cambios a las configuraciones del recurso de grupo para el servicio DHCP la pantalla de recursos de grupo debe ser como la mostrada en la figura AII.14.

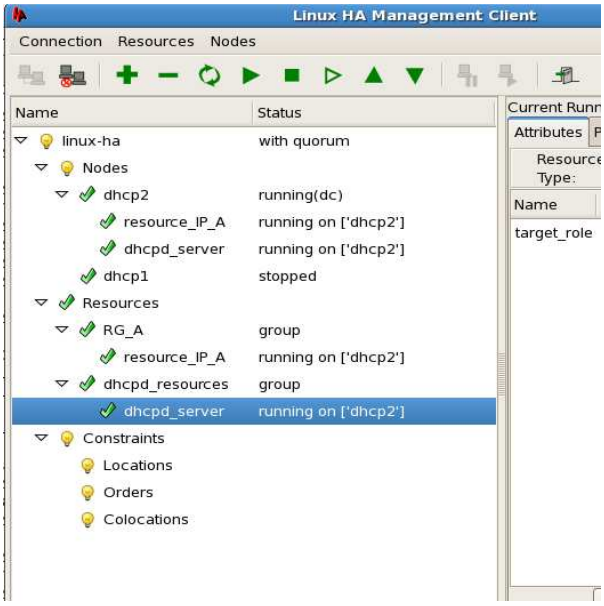


Figura AII.14 Consola de administración inicial con las configuraciones correctas de recursos de grupo.

2 Configuración e instalación del DHCP Failover Protocol

En este punto se establecerán las configuraciones básicas para poder hacer uso del Failover protocol, que servirán de base para crear configuración más avanzada, además se definirán los diferentes modos de operación en que puede configurarse el Failover.

Para configurar el Failover protocol en un conjunto de servidores hay que definir en las políticas de operación que tipo de configuración se va a utilizar, dependiendo de los requerimientos se pueden dividir las diferentes configuraciones en tres tipos principales:

- Servidores simultáneos
- Failover
- Servidor de respaldo

El funcionamiento en Failover siempre existe en modo Primario/secundario, es decir pueden existir varios nodos secundarios, pero sólo un primario, debido a la jerarquía que el protocolo Failover establece, generalmente suele utilizarse una configuración donde existe un nodo primario y uno secundario, pero puede darse el caso en que se tengan más de un servidor secundario. Dependiendo de la configuración elegida dependerá de la configuración que sea tendrá que aplicar al servicio, en la tabla AII.1 se muestran las diferentes configuraciones del servicio.

Tabla AII.1 Configuraciones disponibles en modo Failover

Rol	Escucha	Responde	Responsabilidad de asignación de direcciones
Servidores simultáneos			
Primario	Si	Si	50%
Secundario	Si	Si	50%
Activo/Backup			
Primario	Si	Si	50%
Secundario	Si	Algunas veces	50%
Servidor de respaldo			
Primario	Si	Si	100%

Secundario	No	No	0%

Configuración en nodos simultáneos

En esta configuración los dos servidores del Failover actúan como servidores activos en un mismo segmento de red, ambos servidores reciben los mensajes de los clientes, se realiza un balanceo de cargas. Este tipo de configuración se utilizan donde el tráfico entrante es muy grande y se desea redundancia en el servicio y balanceo de cargas, en caso de que un servidor falle el otro puede hacerse cargo de la carga.

Configuración en modo Failover

Una relación Activo/Respaldo proporciona redundancia pero no balanceo de cargas, en una relación Activo/Respaldo uno de los dos servidores es el servidor primario y otro actual como el secundario, ambos servidores DHCP escuchan las peticiones en un segmento de red pero en condiciones normales de operación sólo el primario responderá, el secundario responderá solamente en caso de que el cliente no sea capaz de contactar al servidor primario, además el secundario responde cuando no está en contacto con el primario.

Configuración servidor de respaldo

Cuando dos servidores están en configuración de servidor de respaldo, un servidor actúa como el respaldo de otro, solamente para respaldar el archivo de asignación de direcciones, en este tipo de configuración el servicio de DHCP se detiene si el servidor primario falla, ya que el secundario no tiene la capacidad de atender las solicitudes hasta que el primario vuelva a funcionar.

La ventaja de este tipo de configuraciones Failover es que permitirá tener el respaldo del archivo de asignación de direcciones no volátil, es decir, el servidor utiliza un servidor para realizar sus respaldos.

Instalación del Failover Protocol

Para realizar la configuración se asumirá que ya se tiene instalado y configurado el servicio de ISC DHCP, si no es así se proporciona una guía de configuración e instalación en el capítulo 3.

Para configurar el servicio DHCP en modo Failover se deben tener al menos dos servidores que conformarán el Failover, por lo que se sugiere que sean una misma distribución o en su caso tengan el mismo núcleo Linux para evitar problemas de compatibilidad entre versiones.

El Failover protocol no tiene un mecanismo para sincronizar configuraciones del servidor *(dhcpd.conf*), solamente posee un mecanismo para actualización del archivo de préstamo de direcciones *(dhcpd.leases),* por lo que primeramente se debe crear un archivo de configuración maestro para ambos miembros del Failover. Una vez creado el archivo de configuración del servicio, es necesario crear una configuración individual para cada uno de los dos servidores, basada en la configuración principal, la única diferencia entre ambos archivos es la parte correspondiente al Failover y el nombre de Failover que manejará cada pool, en caso de que no se desee compartir direcciones se declararán localmente en cada servidor para el caso del archivo de préstamo *dhcpd.lease* si se tenía un servidor que ya estuviera en funcionamiento o un par de servidores que funcionaran de forma independiente; no es necesario realizar modificaciones ya que el Failover protocol hace los cambios necesarios.

Nota: Antes de empezar, verificar que ambos servidores tengan la misma fecha y hora ya que de no ser así habrá muchos problemas de sincronización y el servicio no funcionará de forma correcta.

Configurando el servicio de ISC DHCP para hacer Failover

Antes de configurar los parámetros propios de cada servidor se debe analizar cada uno de los pool de direcciones definidos en cada servidor, con la finalidad de conocer cuáles serán balanceados por el Failover.

Parámetros de configuración

La configuración para cada miembro del Failover es muy similar, excepto por la parte de la configuración que describe la relación entre los nodos del Failover. La configuración del Failover es diferente en cada servidor por dos razones principales, la primera es que cada configuración describe como contactar al otro nodo, y la segunda es que cada servidor tiene configuraciones propias, es decir, una configuración específica para el primario y otra para el secundario, por lo que la configuración es diferente en cada uno de ellos.

Para configurar un Failover en el ISC DHCP, es necesario escribir una declaración del nodo vecino cada declaración del nodo vecino tiene un nombre y una secuencia de datos definidos para mostrar más claramente los parámetros a configurar se muestra un ejemplo de valores en la tabla AII.2

Tabla AII.2 Valores a considerar para configurar el Failover en ISC DHCP

Parámetro	Valor en Primario	Valor en Secundario
Dirección IP	192.168.70.20	192.168.70.18
Puerto	847	647
Dirección IP del vecino	192.168.70.18	192.168.70.20
Puerto del vecino	647	847
Contact time out	180	180
Maximum pending updates	100	100
MCLT	1800	No aplica
Free address balance	50%	50%
Load balance split	50%	No aplica
Load balance override	3	3

Dirección IP y puerto

Cada servidor tiene un puerto de contacto y una dirección IP, estos son la dirección IP y el puerto a través del cual el servidor escuchará las peticiones de su vecino, a su vez las

direcciones IP y puerto del vecino son por las cuales escuchará las peticiones del otro vecino, es decir el principal, esto es necesario incluso para servidores que tengan más de una dirección IP, ya que la dirección IP es utilizada como identificador para la conexión si el servidor envía una dirección de contacto diferente a la que tiene el otro servidor, es decir la que espera recibir, simplemente rechazará la conexión. Los puertos mostrados en la tabla AII.2 son los utilizados por el protocolo pero si se desea pueden ser cambiados. En el servidor ISC la dirección de contacto se especificará en el archivo de configuración con la palabra *address* y el puerto de contacto con la palabra *port*, la dirección del nodo vecino se escribirá como *peer address* y el puerto como *peer port.*

Contact Timeout

El contact timeout determina cuanto tiempo un servidor va a esperar sin recibir mensajes del nodo vecino antes de asumir que la conexión ha fallado. En la tabla AII.2 se estableció un timeout de 180 segundos o 3 minutos, esto permite a los servidores darse cuenta rápidamente de una falla en la conexión, previniendo un outage en la red de mucho tiempo;, en el ISC se especifica con la oración *max-response-deley* en el archivo de configuración.

Número máximo de actualizaciones pendientes

El número máximo de actualizaciones pendientes define el número de actualizaciones que el servidor puede aceptar sin bloquear la entrada, Éste parámetro puede ser complicado de configurar en el servidor, pero en el ISC DHCP el servidor procesa las actualizaciones conforme van llegando , así que la razón para escoger un valor particular para este parámetro es para evitar que el servidor mande muchas actualizaciones a la vez, En general no es necesario configurar este parámetro, en caso de que el servicio no lo exija, toma un valor de 100 por default, pero si se quiere configurar se puede hacer con la línea *max-pending-updates.*

MCLT

El MCLT es la máxima cantidad de tiempo que un servidor puede extender un préstamo sin contactar al otro servidor. Este valor tiene que ser establecido tomando en cuenta el tiempo de préstamo que se le dará al cliente y el tiempo de recuperación del Failover, es decir se debe escoger un valor que sea razonablemente largo para que el cliente que le permita al cliente trabajar durante un tiempo considerable para que no afecte su funcionamiento, aunque no se debe escoger un valor para el MCLT muy largo debido a que el tiempo de recuperación del servicio generalmente no es tan largo, es decir que tan largo como sea el MCLT es el tiempo que tomará regresar al estado normal del Failover tras una falla.

Otro punto a considerar es que si los clientes generalmente tienen un préstamo corto, estos necesitaran renovar con más frecuencia que si tuvieran un préstamo mayor si el intervalo normal del préstamo está definido en 5 horas y el MCLT en 30 minutos, cuando el servidor opere en COMMUNICATIONS-INTERRUMPTED, la carga en cada servidor es 10 veces más grande.

El MCLT es configurado sólo en el servidor primario para evitar conflictos entre el servidor primario y el secundario respecto a su valor, en el servidor ISC el MCLT es configurado usando la palabra *mclt.*

Balanceo de direcciones libres

El balanceo de las direcciones libres es el balance que el servidor primario trata de hacer entre las direcciones en estado libre o FREE y en BACKUP, las direcciones en estado libre están disponibles para su asignación en el servidor primario y las direcciones en el estado de BACKUP están libres para asignarse por el servidor secundario. El servidor de DHCP ISC no soporta configurar el balanceo de las direcciones libres, sólo soporta el balance de 50% para de libres y 50% de backup.

Balanceo de cargas Split

El balanceo de cargas Split, le dice al servidor primario que porción de todos los clientes debe atender, cada cliente es asignado a uno de 256 diferentes grupos, de acuerdo a la información de identificación que envía. Usando el balanceo de cargas Split, el servidor primario construye un arreglo de 256 valores y establece algunos elementos del arreglo a uno y otros a cero. Si la entrada del cliente en el arreglo es 1, el servidor primario asignará dirección a ese cliente, si es cero será asignada por el secundario. Esto permite a los dos nodos del Failover dividir la carga de trabajo. El servidor ISC DHP permite especificar el balance de cargas Split o en valores de forma individual para el arreglo de 256 elementos, en general no hay razón para especificar el arreglo directamente.

Es posible utilizar el valor de Split para poner dos servidores de DHCP en el estado Activo/Backup. En una relación Activo/primario, el servidor primario atiende a todos los clientes y el servidor backup no atiende a ninguno, esto ocurre cuando se pone el valor Split en 256, esto es que todos los 256 elementos en el arreglo son puestos a 1. En el servidor ISC DHCP se configura usando la palabra *Split*, y si se desea establecer el valor manualmente se usa la palabra *hba.*

Load Balance Override

El parámetro *load balance override*, determina cuando el servidor primario o secundario van atender la carga y la respuesta a los clientes de su compañero, aunque supuestamente el servicio funcione de forma correcta. Cada mensaje del DHCP client incluye un campo *secs*, que indica por cuantos segundos el cliente DHCP a tratado de contactar al servidor DHCP si el valor del campo *secs* en un servidor de DHCP es mayor al *load balance override*, el servidor de DHCP, siempre tratara de responder al cliente, no importando el balanceo de cargas, en el servidor de DHCP ISC, el load balance override es especificado por la oración load balance max seconds.

APÉNDICE III

CÓDIGO QUICHO

El código de la aplicación está estructurado en 3 scripts principales en Perl:

1. Edit.cgi: Es la página de inicio que muestra las opciones de consulta.
2. Save.cgi: Es el motor de búsqueda que mostrará el estado de asignación de cada dirección del servicio DHCP.
3. dora.cgi: Se encarga de buscar todos los *four way handshake* del DHCP correspondientes a la dirección buscada.

Edit.cgi

```
#!/usr/bin/perl

###  Formulario para busqueda de de direcciones y estado de subredes  edit.cgi ###

#librerias necesarias
require 'quicho-lib.pl';
ReadParse();
ui_print_header(undef,$text{'create_title'},"");


# Declaración de tabla inicial e  icono búsqueda
@pages = ("searchBanxico");
@titles = map { $text{$_i."_title"} } @pages;
@icons = map { "images/busca.gif" } @pages;
@links = map { "" } @pages;
&icons_table(\@links, \@titles, \@icons, scalar(@titles),undef,179,179);
print "<tr> <td><b>"$text{'create_title'}"</b></td>\n";
print "<td>",&ui_radio("allow", $_[0]->{'allow'},
       [ [ 1, $text{'yes'} ], [ 0, $text{'no'} ] ]),"</td>\n";

# Inicia formulario a enviar a save.cgi, que es el motor de busqueda
print ui_form_start('save.cgi',undef,undef);
print ui_table_start($text{'edit_header'}, 'width=100%', 2,undef);

# Busqueda por dirección ip o mac
print ui_table_row($text{'edit_domain'},ui_textbox('domain',$site->{'domain'},40));
print ui_table_end();
print ui_form_end([[undef,$text{'Buscar'}]]);

# Comieza creación tabla de estado de subredes
$esdin=0;
@tds = ("width=5");
@subred;

# Abre archivos para lectura, NOTA: como modificar lo que está en las primeras comillas,
se encuentra en modo lectura
# Pero si se modifica a modo lectura y escritura puede borrar los archivos.
open CONFIG, "/etc/dhcpd.conf" or print " Error 001: No se puede acceder al archivo de
configuración: /etc/dhcpd.conf \n";
open LEASES,  "/var/lib/dhcpd/dhcpd.leases" or print "Error no se puede acceder al
archivo .leases";
print ui_columns_start(["",$text{'search_subred'},
                        $text{'search_ocupados'},
                        $text{'search_libres'},
                        $text{'search_porcentaje'}],100, 0 ,
\@tds,$text{'search_title'})
@leases=<LEASES>; #paso dhcp.leases a memoria para lectura rapida
@config=<CONFIG>; #paso dhcpd.conf a memoria
close CONFIG;#revisar por que esta doble apertura del archivo
```

```
open CONFIG, "/etc/dhcpd.conf" or print " Error 001: No se puede acceder al archivo de
configuración: /etc/dhcpd.conf \n";
$flag=1;#inicialización de bandera, indica que busque en subred dinamica o estática
# recorre el archivo .conf y busca subredes validas
while (<CONFIG>) {      if(/subnet (\d+.\d+.\d+)(.\d+)/
                         $subnet=$1;
                        $subnetcomplete=("$1"."$2");
                         $esdin=1;  # bandera es dinámica
                         push(@subred,"");
                        push (@subred,
"<ahref=\"save.cgi?subred=$subnetcomplete\"method=post>".&html_escape($subnetcomplete)."
</a>");
                        $flag=0;  # apaga bandera de busqueda de subred y obtiene los
datos de la subred encontrada
                     }
# si lo primero que encuentra es host quiere decir que es estática
                    if (/host/ and $flag == 0){  # si la bandera $flag se apaga obtiene
los datos de los host para calcular uso de subred
#lee dentro de config y cuenta
                        foreach $clientes (@config){
                             if ($clientes =~ /fixed-address $subnet/){
                                  $conta1=$conta1+1;
                             }
                         }
                         $capacidad1=254;
                         $libres1=$capacidad1-$conta1;
                         $porcentaje1=sprintf("%.0f",($conta1/$capacidad1)*100);
#convierte a formato de porcentaje
                         push (@subred, " $conta1"); #imprime en tabla ocupados
                        push (@subred, "$libres1"); #imprime libres
                        push (@subred, "$porcentaje1 % Tipo Estatica");
                        print &ui_columns_row(\@subred, \@tds);
                        pop(@subred);
                        pop(@subred);
                       pop(@subred);
                       pop(@subred);
                       pop(@subred);
                       $flag=1;
#le dice que teminó de buscar en esa subred el número de host
                       $conta1=0;  #regresa contador de clientes a cero
                  }
# si lo primero es range quiere decir que la subred es dinámica
                  if(/range (\d+.\d+.\d+.)(\d+) (\d+.\d+.\d+.)(\d+)/ and $flag == 0){
                         ### lee dentro de leases
                         $capacidad=("$4"-"$2");
                         foreach $clientes (@leases){  #recorre leases en busca de
coincidencias de clientes en esa subred
                              if ($clientes =~ /lease $subnet/){
                                 $conta=$conta+1;
                              }
                         }

                         $disponibles=$capacidad-$conta;
                         $porcentaje=sprintf("%.0f",($conta/$capacidad)*100);
                         push (@subred, "$conta");
                         push (@subred,"$disponibles ");
                         push (@subred,"$porcentaje %  Tipo Dinamica");
                        print &ui_columns_row(\@subred, \@tds);
# imprime datos obtenidos en la fila
                         pop(@subred);
                         pop(@subred);
                         pop(@subred);
                         pop(@subred);
                        pop(@subred);
                        $flag=1;
                       $conta=0;
                }
```

```
                #si no encuentra lease ni host y en su lugar esta un "}" fin de subred
imprime que no hay clientes en la subred


                if(/}/ and $flag == 0){
                        ### lee dentro de config
                         push (@subred,"-------");
                         push (@subred,"-------");
                         push (@subred,"No hay clientes");
                        print &ui_columns_row(\@subred, \@tds);
                        pop(@subred);
                        pop(@subred);
                        pop(@subred);
                        pop(@subred);
                        pop(@subred);
                        $flag=1;
                }
}
close CONFIGNORM;
close CONFI;
print &ui_columns_end();
ui_print_footer('', $text{'index_return'});
```

Save.cgi

```
#!/usr/bin/perl
# Motor de busqueda de busqueda de direcciones de DHCP
# Search engine for IP and MAC state at ISC DHCP
require 'quicho-lib.pl';
ReadParse();
ui_print_header(undef,$text{'edit_header'},"");
#abre logs para su análisis
open CONFIG, "/etc/dhcpd.conf" or print 'Error al abrir el archivo dhcpd.conf';
#abre dhcpd.leases para ver la asignación dinámica
open LEASES, "/var/lib/dhcpd/dhcpd.leases" or print 'Error al abrir el archivo
dhcp.leases';
# abre archivo temporal de logs y lo vacia
open TEMPORAL, ">/var/log/Historial/templog" or print 'Error al abrir el archivo
templog';
#variables globales
$norm;
$veces;
$flag;
$bSemanas;
close TEMPORAL;
#lee entrada de variable subred
if($in{'subred'}){
        while(<$in{'subred'}>){
                if/(\d+.\d+.\d+)(.\d+)/){
                $norm = $1;
                }
        }
}
#lee entrada de variable semana
if($in{'semana'}){
        while(<$in{'semana'}>){
                if/(\d+)/){
                print "Consulta $1 semanas";
                $bSemanas=$1;
                }
        }
}
#concatena logs deacuerdo a la entrada
```

```
if ( "$bSemanas" eq 1 or "$bSemanas" eq undef) {
     open TEMPORAL, ">>/var/log/Historial/templog" or print 'Error al abrir el archivo
templog para concatenar';
     open MESS, "/var/log/messages" or print 'Error al abrir el archivo messages';
     foreach $temsemana2 (<MESS>){
          #concateno en el archivo
          print TEMPORAL $temsemana2;
     }
     close TEMPORAL;
     close MESS;
     #print "no concateno";
} elsif ( "$bSemanas" eq 2) {
     open TEMPORAL, ">>/var/log/Historial/templog" or print 'Error al abrir el archivo
templog para concatenar';
     open MESS, "/var/log/messages" or print 'Error al abrir el archivo messages';
     open MESS1, "/var/log/messages.1" or print 'Error al abrir el arvivo messages.1';
     foreach $temsemana2 (<MESS1>){
#concateno en el archivo
          print TEMPORAL $temsemana2;
     }
     foreach $temsemana (<MESS>){
#concateno en el archivo
     print TEMPORAL $temsemana;
    }
     close TEMPORAL;
     close MESS;
     close MESS1;
     #print "concateno messages con messages.1";
} elsif ( "$bSemanas" eq 3) {
     open TEMPORAL, ">>/var/log/Historial/templog" or print 'Error al abrir el archivo
templog para concatenar';
     open MESS, "/var/log/messages" or print 'Error al abrir el archivo messages';
     open MESS1, "/var/log/messages.1" or print 'Error al abrir el arvivo messages.1';
     open MESS2, "/var/log/messages.2" or print 'Error al abrir el arvivo messages.2';
foreach $temsemana3 (<MESS2>){
#concateno en el archivo
print TEMPORAL $temsemana3;
     }
     foreach $temsemana2 (<MESS1>){
#concateno en el archivo
  print TEMPORAL $temsemana2;
     }
 foreach $temsemana (<MESS>){
 #concateno en el archivo
 print TEMPORAL $temsemana;
     }
     close TEMPORAL;
     close MESS;
     close MESS1;
     close MESS2;
    #print "concateno m m.1 m.2";
} elsif ( "$bSemanas" eq 4) {
     open TEMPORAL, ">>/var/log/Historial/templog" or print 'Error al abrir el archivo
templog para concatenar';
     open MESS, "/var/log/messages" or print 'Error al abrir el archivo messages';
     open MESS1, "/var/log/messages.1" or print 'Error al abrir el arvivo messages.1';
     open MESS2, "/var/log/messages.2" or print 'Error al abrir el arvivo messages.2';
     open MESS3, "/var/log/messages.3" or print 'Error al abrir el arvivo messages.3';
     foreach $temsemana4 (<MESS3>){
     #concateno en el archivo
     print TEMPORAL $temsemana4;
     }
     foreach $temsemana3 (<MESS2>){
          #concateno en el archivo
          print TEMPORAL $temsemana4;
```

```
     }
     foreach $temsemana2 (<MESS1>){
          #concateno en el archivo
         print TEMPORAL $temsemana2;
     }
     foreach $temsemana (<MESS>){
          #concateno en el archivo
          print TEMPORAL $temsemana;
     }
     close TEMPORAL;
     close MESS;
     close MESS1;
     close MESS2;
     close MESS3;
     #print "concateno m m.1 m.2 m.3";
} elsif ( "$bSemanas" eq 5){
     open TEMPORAL, ">>/var/log/Historial/templog" or print 'Error al abrir el archivo
templog para concatenar';
open MESS, "/var/log/messages" or print 'Error al abrir el archivo messages';
open MESS1, "/var/log/messages.1" or print 'Error al abrir el arcvivo messages.1';
open MESS2, "/var/log/messages.2" or print 'Error al abrir el arcvivo messages.2';
open MESS3, "/var/log/messages.3" or print 'Error al abrir el arcvivo messages.3';
open MESS4, "/var/log/messages.4" or print 'Error al abrir el archivo messages.3';
     foreach $temsemana5 (<MESS4>){
    #concateno en el archivo
     print TEMPORAL $temsemana5;
     }
     foreach $temsemana4 (<MESS3>){
     #concateno en el archivo
    print TEMPORAL $temsemana4;
    }
    foreach $temsemana3 (<MESS2>){
          #concateno en el archivo
          print TEMPORAL $temsemana4;
     }
    foreach $temsemana2 (<MESS1>){
          #concateno en el archivo
          print TEMPORAL $temsemana2;
     }
     foreach $temsemana (<MESS>){
          #concateno en el archivo
          print TEMPORAL $temsemana;
     }
     close TEMPORAL;
     close MESS;
     close MESS1;
     close MESS2;
    close MESS3;
    close MESS4;
    #print "concateno m m.1 m.2 m.3 m.4";
}
#termina concatenacion de archivos
open NORMALIZED, "/var/log/Historial/templog" or print 'Error al abrir el archivo temlog
para lectura';
print ui_columns_start(["",$text{'save_ip'},
                        $text{'save_mac'},
                        $text{'save_inven'},
                        $text{'save_fechini'},
                          $text{'save_horaini'},
                          $text{'save_fechafin'},
                       $text{'save_horafin'}],100, 0 , \@tds,$text{'search_title'});
#busqueda de direcciones IP
if( $in{'domain'} ne undef ){
@con=<CONFIG>;
$textor;
```

```
$count=0;
@header=("","Direcciones Estaticas:");
print &ui_columns_row(\@header, \@tds);
foreach $textor(@con){
         if ( $textor =~ /host ([A-Z0-9]*)/){
              push (@temp, "$1");
         }
         if(  $textor =~ /hardware ethernet ([a-z0-9]{2}:[a-z0-9]{2}:[a-z0-9]{2}:[a-z0-
9]{2}:[a-z0-9]{2}:[a-z0-9]{2})*/){
           push(@temp, "$1") ;
         }
         if(  $textor =~ /fixed-address (\d+.\d+.\d+.\d+)/){
             push(@temp, "$1");
         }
}
@tempr = reverse(@temp);
$tempmac;
$labuena;
        #print "leyendo de busqueda ip especifica";
        while (<NORMALIZED>) {
           if(/([A-Za-z]+)(\s+)([0-9]+)(\s+)(\d\d:\d\d:\d\d)(\s+)([a-z0-9]+)(\s+)(dhcpd:
DHCPACK)(\s+)([a-z]*)(\s+)(\d+.\d+.\d+)(.\d+)(\s+)([A-Za-z(\s]+)([a-z0-9]{2}:[a-z0-
9]{2}:[a-z0-9]{2}:[a-z0-9]{2}:[a-z0-9]{2}:[a-z0-9]{2})/){
                                $s8= "$13"."$14";
#si coincide con la ip buscada abre log  y pasa los parametros
                              if( $s8 eq $in{'domain'} ){
                                  $veces=$veces+1;
                                  push(@subred, $veces);
                                  push (@subred,"<a
href=\"dora.cgi?ipbusca=$s8&semana=$bSemanas\"method=post>".
&html_escape($s8)."</a>");

                                 foreach $hi (@tempr){
                                 $count=$count+1;
                                       if($hi eq $in{'domain'}){
                                       $labuena=$count;
                                       }
                                  }

#validar campos ver si existe en el de configuracion el nombre del host
                                  if($s8 eq $in{'domain'}){
                                 #$tempmac= $tempr[$labuena];
                                $tempmac=$17;

push(@subred,
"<ahref=\"dora.cgi?ipbusca=$tempmac&semana=$bSemanas\"method=post>".&html_escape($tempma
c)."</a>");
                                          $temphost=$tempr[$labuena+1];
                                          push(@subred,$temphost);
                                          push (@subred,"$1 $3");
                                          push (@subred,"$5");
                                  }
                                   if ( "$1" eq "Jan") {
                                          push(@subred,"Feb $3");
                                    } elsif ( "$1" eq "Feb") {
                                          push(@subred,"Mar $3");
                                    } elsif ( "$1" eq "Mar") {
                                          push(@subred,"Apr $3");
                                    } elsif ( "$1" eq "Apr") {
                                          push(@subred,"May $3");
                                    } elsif ( "$1" eq "May") {
                                          push(@subred, "Jun $3");
                                    } elsif ( "$1" eq "Jun") {
                                          push(@subred,"Jul $3");
                                    } elsif ( "$1" eq "Jul") {
```

```
                                    push(@subred,"Aug $3");
                                  } elsif ( "$1" eq "Aug") {
                                    push(@subred,"Sep $3");
                                  } elsif ( "$1" eq "Sep") {
                                    push(@subred,"Oct $3");
                                  } elsif ( "$1" eq "Oct") {
                                    push(@subred,"Nov $3");
                                  } elsif ( "$1" eq "Nov") {
                                    push(@subred, "Dec $3");
                                  } elsif ( "$1" eq "Dec") {
                                    push(@subred, "Jan $3");
                                  }
                                push(@subred,"$5");
                                  $count=0;
                                  print &ui_columns_row(\@subred, \@tds);
                                  pop(@subred);
                                  pop(@subred);
                                  pop(@subred);
                                  pop(@subred);
                                  pop(@subred);
                                  pop(@subred);
                                pop(@subred);
                                pop(@subred);
                                  $labuena=0;
                           }
                           # busqueda por MAC

                           if( $17 eq $in{'domain'} ){
                                $veces=$veces+1;
                                push(@subred, $veces);
 #push (@subred,"<a href=\"dora.cgi?ipbusca=$17&semana=$bSemanas\"method=post>".
                           #&html_escape($17)."</a>");


                             foreach $hi (@tempr){
                                    $count=$count+1;
                                   if($hi eq $in{'domain'}){
                                        $labuena=$count;
                                        print "encontre $labuena";
                                   }
                                }
                                if($17 eq $in{'domain'}){
                                     $tempmac= $tempr[$labuena];
push(@subred, "<a href=\"dora.cgi?ipbusca=$s8&semana=$bSemanas\"method=post>".
                                &html_escape($s8)."</a>");
  push (@subred,"<a href=\"dora.cgi?ipbusca=$17&semana=$bSemanas\"method=post>".
                                &html_escape($17)."</a>");
                                       $temphost=$tempr[$labuena];
                                       push(@subred,"------");
                                       push (@subred,"$1 $3");
                                       push (@subred,"$5");
                                }
                                if ( "$1" eq "Jan") {
                                       push(@subred,"Feb $3");
                                  } elsif ( "$1" eq "Feb") {
                                       push(@subred,"Mar $3");
                                  } elsif ( "$1" eq "Mar") {
                                       push(@subred,"Apr $3");
                                  } elsif ( "$1" eq "Apr") {
                                       push(@subred,"May $3");
                                  } elsif ( "$1" eq "May") {
                                       push(@subred, "Jun $3");
                                  } elsif ( "$1" eq "Jun") {
                                       push(@subred,"Jul $3");
```

```
                                    } elsif ( "$1" eq "Jul") {
                                        push(@subred,"Aug $3");
                                    } elsif ( "$1" eq "Aug") {
                                           push(@subred,"Sep $3");
                                    } elsif ( "$1" eq "Sep") {
                                        push(@subred,"Oct $3");
                                    } elsif ( "$1" eq "Oct") {
                                        push(@subred,"Nov $3");
                                    } elsif ( "$1" eq "Nov") {
                                        push(@subred, "Dec $3");
                                    } elsif ( "$1" eq "Dec") {
                                        push(@subred, "Jan $3");
                                    }
                                  push(@subred,"$5");
                                    $count=0;
                                    print &ui_columns_row(\@subred, \@tds);
                                    pop(@subred);
                                    pop(@subred);
                                    pop(@subred);
                                    pop(@subred);
                                    pop(@subred);
                                    pop(@subred);
                                  pop(@subred);
                                  pop(@subred);
                                  $labuena=0;

                               }
           }
        }
```

dora.cgi

```
#!/usr/bin/perl
# Consulta en logs todo lo relacionado a la direccion MAC o IP solicitada
require 'quicho-lib.pl';
ReadParse();
ui_print_header(undef,$text{'dora_header'},"");
#abre Logs para su analisis
open NORMALIZED, "/var/log/Historial/templog" or print 'Error al abrir templog';
@norm=<NORMALIZED>;
#lee en templog generado por save.cgi
print ui_columns_start($text{'todo_dora'},100, 0 ,undef,$text{'dora_title'});
if( $in{'ipbusca'} ne undef ){
     #### busca relacionado con ip
     $gallo=$in{'ipbusca'};
     for $textor1 (@norm){
  if($textor1 =~ /(\d+\.\d+\.\d+\.\d+)/){if($1 eq $gallo){
                    push(@dora,"$textor1");
                   print &ui_columns_row(\@dora, undef);
                   pop(@dora);
              }
         }if($textor1 =~ /([a-z0-9]{2}:[a-z0-9]{2}:[a-z0-9]{2}:[a-z0-9]{2}:[a-z0-
9]{2}:[a-z0-9]{2})/ ){
              if($1 eq $gallo){
                   push(@dora,"$textor1");
                   print &ui_columns_row(\@dora, undef);
                   pop(@dora);
              }
         }
     }
     #push(@dora,"estoy leyendo ip");
     #print &ui_columns_row(\@dora, \@tds);
}
print &ui_columns_row(\@dora, undef);
```

```
print &ui_columns_end();
close NORMALIZED;
ui_print_footer('',$text{'index_return'});
```

Fuentes de información

Fuentes de información

Libros

- [1] Ralph Droms y Ted Lemon, *the DHCP handbook, second edition*, USA, Sams Publishing, 2003.
- [2] Academia de Networking de Cisco Systems: "*Guía del segundo año CCNA 3 y 4"*, Cisco Systems, Inc., tercera edición, 2004.
- [3] López B. María Jaquelina y Quezada, R. Cintia; *Fundamentos de seguridad informática*, México, UNAM, Facultad de Ingeniería, 2006.
- [4] Whey-Jen Chen, Masafumi Otsuki, et. Al., *High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows,* IBM Redbooks, 2009.
- [5] Lydia Parziale, Antonio Días, et. Al., *Achieving High Availability on Linux for System z with Linux-HA Release 2*, IBM Redbooks, abril 2009.
- [6] Klaus Schmidt, *High Availability and Disaster Recovery Concepts, Design, Implementation,* Frankfurt, Alemania, Springer-Verlag, 2006.
- [7] Evan Marcus y Hal Stern, *Blueprints for High Availability 2ed., USA,* Wiley Publishing, Inc., 2003.
- [8] Chris Oggerino, *High Availability Network Fundamental*. Indianapolis, USA, Cisco Press. 2001.

Artículos de revistas

- [9] dig1t4l, alta *disponibilidad y rendimiento en redes de comunicaciones*. CTICRM Digital - Número 3 - julio 2009,

Referencias Electrónicas

- [10] La definición de software libre *http://www.gnu.org/philosophy/free-sw.es.html*, consulta enero 2010.
- [11] Diez ventajas del software libre y propietario *http://www.abadiadigital.com/articulo/diez-ventajas-del-software-libre-y-propietario/*, consulta enero 2010.
- [12]Comparación entre RH y centOS http://www.singlehop.com/blog/2007/01/31/decision-time-centos-vs-rhel/, consulta febrero de 2010.

- [13]System Reliability and Availability http://www.eventhelix.com/realtimemantra/faulthandling/system_reliability_availability.htm, consulta abril 2010.
- [14]Heartbeat Guide - SUSE Linux Enterprise Server http://www.novell.com/documentation/sles10/pdfdoc/heartbeat/heartbeat.pdf, consulta diciembre 2009.
- [15]Configuración Heartbeat http://bizkaia.no-ip.biz/?p=6 consulta abril 2010.
- [16]Servidores de alta disponibilidad: Heartbeat http://bytecoders.homelinux.com/content/servidores-de-alta-disponibilidad-heartbeat.html, consulta junio 2010.
- [17]Clustering *Definición de Clúster* http://www.codigolibre.org/index.php?option=com_content&view=article&id=5389:clustering&catid=36:gnu-cat&Itemid=53, consulta junio 2010.
- [18]Descripción de los agentes de retransmisión *DHCP* http://technet.microsoft.com/es-es/library/cc779610(WS.10).aspx http://www.cicei.com/ocon/gsi/tut_tcpip/3376c418n.html, consultas junio 2010.
- [19]IPv6 *DHCPv6 Based IPv6 Access Service* http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/whitepaper_C11-472610.html, consulta junio 2010.
- [20] Marke for DNS,DHCP and IP Addres Management, www.gartner.com, consulta febrero 2010.

Whitepapers

- [21]Citrix, The *three levels of high availability – Balancing priorities and cost,* agosto 2009.
- [22]RedHat; Red Hat Enterprise Linux 5 Continuous Availability, marzo 2007.
- [23]Cisco Systems Inc.; *How-to Manage CNS Cisco Network Registrar*, enero 2006
- [24]Jesús González Barahona et. Al., Introducción al software libre, UOC 2003.