ARQUITECTURAS DE CORREO ELECTRONICO DE ALTA DISPONIBILIDAD

Maestría en Computación con énfasis en Gestión de la Información

UNAN-Leon

2006-2008

PRESENTADO POR:

Ing. Edison D. Cuevas V.

TUTOR: **Dr. David Fernández Barrero**

Managua 12 de Diciembre 2008.

Resumen

En este documento se hace un análisis e implementación de arquitecturas de alta disponibilidad para servicios de correo electrónico. Empezaremos abordando una solución desarrollada en un ámbito de producción (en la red de datos de la Universidad Nacional Autónoma de Nicaragua-Managua) para posteriormente tratar casos de estudios desarrollados en un ambiente de laboratorio, no obstante las soluciones propuestas bien podrían ser desarrolladas en ambientes reales. Los casos de estudio están enfocados en situaciones muy particulares en los que la tolerancia a fallos no es una alternativa por ser una solución muy costosa, manteniendo siempre ligada la alta disponibilidad a los servicios de correo electrónico.

El correo electrónico es una tecnología utilizada en universidades, gobierno y empresas, cuyos servicios son generalmente brindados por ellos mismos, sin embargo no siempre se cuenta con el presupuesto suficiente como para adquirir hardware tolerante a fallos. Tomando en cuenta que este representa un servicio altamente crítico, ya que la caída por un intervalo de tiempo significativo (horas) puede representar la pérdida de información muy importante, la solución es una arquitectura de correo electrónico de alta disponibilidad.

No podemos abordar una solución de correo electrónico de alta disponibilidad sin antes tratar los protocolos e implementaciones de protocolos en los cuales se basa, por lo que se hace un amplio estudio de los estándares SMTP, POP3, DNS y encaminamiento de correo electrónico. Posteriormente se preparará la infraestructura (hardware y software) sobre la que correrán los servicios, utilizando SuSE Linux 10.3 como sistema operativo y haciendo uso por comodidad de las aplicaciones facilitadas dentro de la distribución como el servidor SMTP Postfix, Cyrus-IMAP, Open SSL y BIND. El resultado final será una solución corporativa de correo electrónico con software casi en su totalidad de libre distribución.

Con este documento se hace un estudio e implementación de los protocolos y servicios de correo electrónico más utilizados en Linux, así como el diseño de algunas arquitecturas de alta disponibilidad adaptadas a las necesidades de este tipo de servicios.

INDICE DE CONTENIDO

1. INTRODUCCIÓN

Es mucha la documentación que explica la forma de administrar un sistema Linux, esto incluye: administración de usuarios, archivos y recursos de red, servicios DNS, Web, Proxy y DHCP. Todos ellos tienen ciertas cosas en común, son servicios proveídos por un solo fabricante de software o su administración está ligada a un protocolo. No se puede decir lo mismo con el servicio de correo electrónico, la complejidad de su funcionamiento lo hace imposible, por lo que históricamente se ha intentado seguir más o menos de manera similar la filosofía empleada en la mayoría del software base del sistema operativo Linux, programas que realizan tareas sencillas, que integrados pueden resolver tareas complejas.

Implementando algunos de los protocolos mas populares en sistemas Linux, el administrador podrá dar soluciones a sus usuarios fiables, rápidos y seguras de:

- Envío y recepción de correos con SMTP,
- Acceso a buzones a través de los protocolos IMAP/POP3 y HTTP,
- Métodos de autenticación segura con PAM y bases de datos.
- Filtrado de correos.
- Gestión de listas de correo.
- Y sobre todo brindar todos los servicios con características cercanas a la tolerancia a fallos, a través de una arquitectura de alta disponibilidad.

Para la realización de este trabajo se ha hecho un estudio Analítico y de Aplicación, ya que se realizó un estudio preliminar de las tecnologías de corre electrónico, mismo que permitió luego realizar el diseño y la implementación de algunas arquitecturas de alta disponibilidad.

El presente documento hace un recorrido por los estándares de protocolos necesarios para desarrollar una solución completa de correo electrónico y muestra como

implementar una a partir de software en su mayoría de libre distribución. Luego se enfoca en el diseño e implementación tanto en ámbitos reales como de laboratorio de algunas arquitecturas de alta disponibilidad para sistemas de correo electrónico, proponiendo y resolviendo casos de estudio. Se ha tomado como caso principal, resolver los problemas de almacenamiento, distribución de carga y disponibilidad del servicio de correo brindado dentro de la Universidad Nacional Autónoma de Nicaragua (Managua).

Se ha organizado el documento en tres partes bien definidas, manteniendo siempre una relación estrecha con la administración de servicios de correo electrónico en sistemas Linux:

- Un estudio detallado de los protocolos de correo electrónico más utilizados POP3 y SMTP.
- Preparación e implementación de una solución completa de correo electrónico bajo Linux.
- Estudio e implementación de arquitecturas de alta disponibilidad para servicios de correo.

Con este estudio se esperan dar algunas propuestas de arquitecturas de correo electrónico de alta disponibilidad para ser implementadas con software libre y poner bajo producción al menos una de ellas.

Objetivos

Objetivo General

- Diseñar e implementar arquitecturas de correo electrónico de alta disponibilidad.

Objetivos Específicos

- Estudiar algunas de las tecnologías y protocolos de correo electrónico más utilizadas en Linux.
- Analizar distintos diseños convencionales de sistemas de correo electrónico.
- Diseñar arquitecturas de alta disponibilidad para sistemas de correo electrónico.
- Implementar al menos una de estas arquitecturas en la red de la UNAN-Managua.
- Evaluar la solución de alta disponibilidad implementada en la UNAN-Managua

2. GENERALIDADES DE CORREO ELECTRÓNICO

Hoy en día el Correo Electrónico se ha convertido en una tecnología tan difundida como la radio, la televisión o el teléfono móvil. Servicios gratuitos como hotmail, yahoo y gmail han hecho del correo electrónico una herramienta cuyo alcance está limitado por la disponibilidad que se tenga de una PC conectada a internet, sin embargo al usuario común le es indiferente toda la tecnología e infraestructura que hay detrás de estos sistemas.

Los que tienen cierta experiencia en administración de sistemas saben que la implementación de un servidor de correo electrónico no es una tarea sencilla, más aún con la proliferación de virus y gusanos en correos electrónicos, correo spam y el phishing han hecho más compleja la protección de los servidores, hay que recordar que los usuarios esperan un servicio siempre disponible, además la idea es que el correo le resuelva y no que le genere problemas como podría hacerlo un virus. A todo esto entonces ¿Qué pasa con software que se ofrece en internet gratis, ofreciéndole al usuario protección contra virus, spam, etc.? Dicho software de manera ilegal sin que el usuario se dé cuenta instala un programa espía (spyware), que enviará información a un sitio generador de *spam* sobre los hábitos de navegación del usuario y al final el susodicho programa resulta que le genera mas spam de lo que recibía en un principio.

Este capítulo tratará aspectos claves de este complejo sistema, necesarios para que el lector conozca y se dé una idea general de lo que implica esta tecnología, incluyendo la estructura de un mensaje de correo electrónico, la terminología empleada, aspectos sobre seguridad no sin antes echar un vistazo a su historia y evolución. Algunos de los temas tratados se profundizaran en el **Capitulo 3**.

2.1 Historia y evolución del correo electrónico

El correo electrónico no es una tecnología que se inventó por casualidad, más bien fue visto como un producto cuyo fin seria resolver parte de las necesidades de comunicación humana, a como lo fueron en su momento otras tecnologías como el telégrafo, el teléfono o la televisión. Las primeras formas de correo electrónico tienen su origen en los sistemas colosales del MIT y la Universidad de Berkeley y consistían en copiar archivos de texto de persona a persona. Variantes menos rusticas comenzaron a utilizarse a partir de 1965 con una nueva innovación, permitían acceder desde terminales remotas a sistemas de tiempo compartido, permitiendo guardar archivos en disco.

Desde la invención de ARPANET en 1966 y sus primeros intentos en crear tecnología para comunicar equipos de cómputo pasaron 6 años hasta que la infraestructura y tecnología ofreciera las capacidades para el envío y recepción de datos de considerable tamaño. En 1971 Ray Tomlinson del BBN (Bolt, Beranek and Newman), compañía contratada por la ARPANET para el diseño de sus equipos de comunicaciones, inventa el primer programa de correo electrónico para mandar mensajes en redes distribuidas. El programa original es producto de otros dos, un programa interno de correo electrónico (SENDMSG) y un programa experimental de transferencia de archivos (CPYNET). Un año más tarde Tomlinson modifica el programa y lo adapta a ARPANET donde este se convierte en un éxito. Elige la @ como carácter separador entre lo que define el nombre del buzón y la maquina donde se encuentra el usuario, a la vez Larry Roberts crea el primer cliente de correo llamado RD, capaz de listar, leer selectivamente, guardar, reenviar y responder mensajes. En un principio Tomlinson no imaginó que su invento fuera algo tan valioso e innovador más bien lo vió como un paso necesario en la tecnología de comunicaciones, y mucho menos imaginarse que tan solo un año más tarde (1973) según un estudio de ARPA (Agencia de Investigación de Proyectos Avanzados) el 75% del tráfico total de ARPANET era transferencia de correos electrónicos. Esto marca el comienzo de una tecnología cuya evolución forma parte esencial en la historia de Internet.

En 1977 se publica el primer documento (RFC 733) que describe de manera formal la especificación de formatos en el correo electrónico, en 1983 Dave Crocker (uno de los colaboradores del RFC 733) publica el RFC 822, el cual fue el primer estándar que describe la sintaxis de los nombres de dominio.

A principios del año 1980, la transmisión de correo electrónico fue mejorada usando tecnología UUCP (Mecanismo de transporte store-and-forward) en la Universidad de Berkeley, donde fue desarrollado el sistema BSD Unix. Más tarde Eric Allman creó un programa llamado ***delivermail*** junto a múltiples servicios de transporte de correo electrónico, creando, en efecto un sistema de correo más integrado con capacidades de almacenamiento y reenvío. Con esta experiencia previa Allman construye el programa Sendmail, que fue distribuido con los sistemas BSD Unix; Sendmail mas tarde se convertiría en el Servicio SMTP mas usado en Internet.

Los sistemas comerciales de correo electrónico aparentemente inician en 1989 con un acuerdo realizado entre el proveedor comercial de comunicaciones MCI y una red de investigación llamada NSFNET para interconectarse a través de la Corporación para Iniciativas de Investigación Nacional (CNRI). Poco después CompuServe conectado al NSFNET a través de la Universidad de Ohio, pone a la disposición en Internet su propio servicio de correo comercial.

Hoy en día existen numerosas empresas online que brindan servicios de correo electrónico gratis, con múltiples funcionalidades.

2.2 Estructura de un mensaje de correo electrónico

En un principio las formas de correo electrónico fueron abordadas de manera simple (ver sección anterior), construidas sobre algunas de las convenciones escritas por Ray Tomlinson; sin embargo se tuvo la necesidad, con el paso del tiempo de actualizarlas para adaptarse a las necesidades de los usuarios y al desarrollo de Internet. Esta evolución nos lleva a la especificación definida en el *RFC 2821*[1] o Protocolo Simple de Transferencia de Correo (SMTP), que es la tecnología actual, subyacente a la transferencia de correos entre servidores en Internet. En los primero días de Internet, se necesitaban aplicaciones (Gateway) especiales para transferir correos de un sistema a otro; SMTP vino a resolver este inconveniente. En el capítulo 3 se abordará ampliamente este protocolo.

Un mensaje de correo en Internet debe tener un formato específico, formato que también se define en otros borradores o estándares de internet, por el ejemplo el *RFC 822*, especifica el formato para el cuerpo de un mensaje de correo. En resumen un correo electrónico tiene dos partes bien definidas, la cabecera y el cuerpo del mensaje.

[1] SMTP estaba definido originalmente en el RFC 821, ahora actualizado en el RFC 2821 (www.ietf.org/rfc/rfc2821.txt)

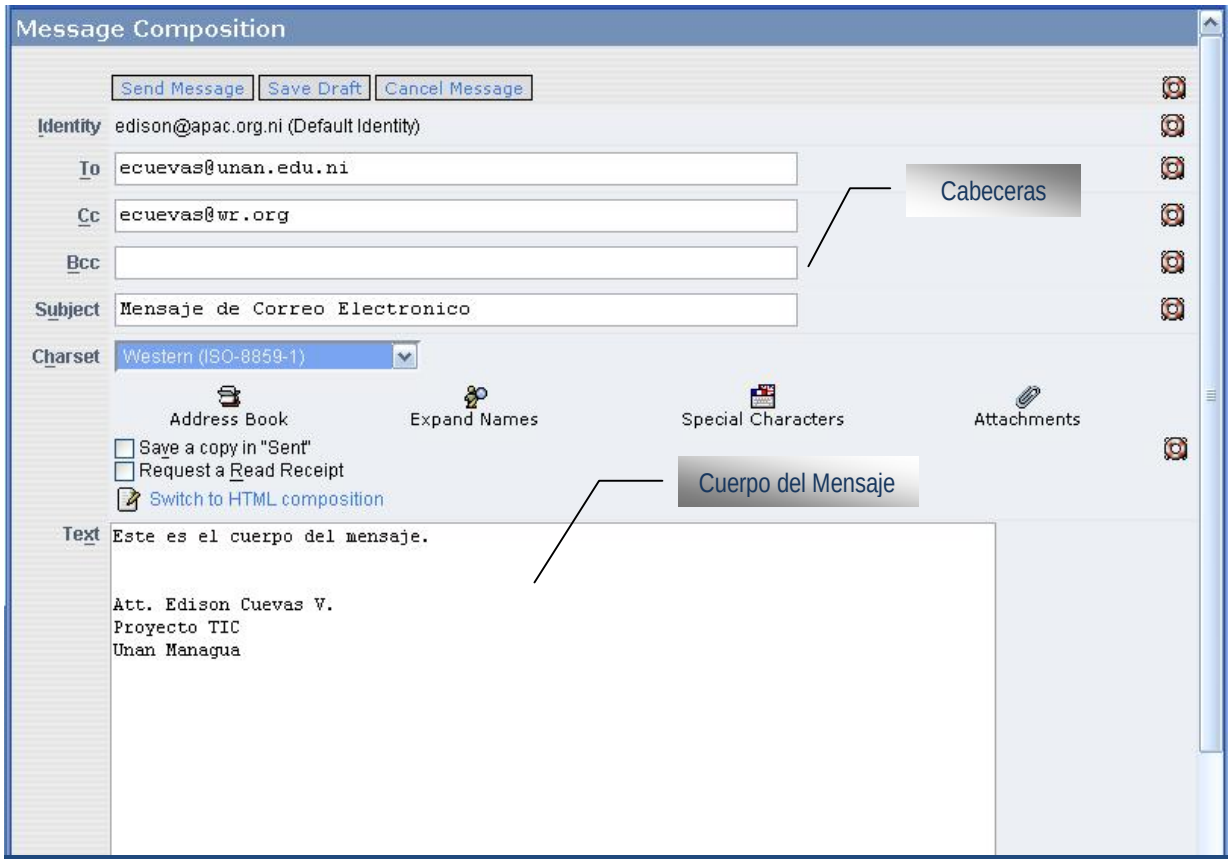


Figura 2.1 *Un típico mensaje de correo electrónico*

2.2.1 Cabecera del mensaje

Todo correo bien formado que cumpla con las especificaciones para formatos de mensajes, comienza con una cabecera[2]. Una cabecera de un mensaje de correo es una serie secuencial de líneas, cada línea está compuesta por una serie de campos de texto concatenados por un espacio. Algunos campos de la cabecera son agregados por la aplicación cliente de correo electrónico (MUA) y otros por cada uno de los servidores

[2] El RFC 822 es el estándar original que detalla el formato de cabecera de los mensajes de correo en Internet, y con frecuencia es llamado formato 822. En el RFC 1123 se muestran algunas correcciones y clarificaciones al RFC 822. En la actualidad existe el estándar, RFC 2822 (`www.ietf.org/rfc/rfc2822.txt`) que ha venido a reemplazar a los RFC 822 y 1123. Algunos concideran al RFC 2822 menos comprensivo que sus predecesores. El RFC 2822 con frecuencia es llamado 822bis. Es importante reconocer que no todas las aplicaciones clientes de Internet pueden adaptarse perfectamente al RFC 822, algunos clientes ignoran restricciones detalladas en el RFC 822.

SMTP que manipulan y procesan el mensaje de correo electrónico a lo largo del recorrido hacia su destino final. El siguiente es un ejemplo que muestra el contenido de la cabecera de un correo electrónico.

```
Received: from aut.uah.es ([127.0.0.1])
      by localhost (correo [127.0.0.1])(amavisd-new, port 10024) with
LMTP
      id 01101-01-26 for <ecuevas@unan.edu.ni>;
      Fri, 14 Mar 2008 12:53:36 +0100 (CET)
Received: from [172.29.16.100] (unknown [172.29.16.100])
      (using TLSv1 with cipher AES128-SHA (128/128 bits))
      (No client certificate requested)
      by aut.uah.es (Postfix) with ESMTP id 5F2FB127A08C
      for <ecuevas@unan.edu.ni>; Fri, 14 Mar 2008 12:31:20 +0100 (CET)
Message-Id: <38461F99-B36B-451F-BB6A-D22969F3922D@aut.uah.es>
From: "David F. Barrero" <dfbarrero@aut.uah.es>
To: ecuevas@unan.edu.ni
In-Reply-To: <20080228191432.4791B1C6E774@mail.unan.edu.ni>
Content-Type: multipart/mixed; boundary=Apple-Mail-6--446175565
Mime-Version: 1.0 (Apple Message framework v919.2)
Subject: Re: Monografia
Date: Fri, 14 Mar 2008 12:31:30 +0100
References: <20080228191432.4791B1C6E774@mail.unan.edu.ni>
X-Mailer: Apple Mail (2.919.2)
X-Virus-Scanned: by amavisd-new-20030616-p10 (Debian) at aut.uah.es
```

Los campos agregados por cada servidor de Internet (campo ***Received***) se utilizan para detallar cada uno de los servidores que han procesado el correo electrónico, así como la hora y la aplicación servidor encargada del transporte del correo. Los campos añadidos por aplicaciones clientes son útiles para indicar quien es el remitente del mensaje, cual es el destino o para informar donde y cuando se originó el mensaje. La cabecera del mensaje termina con una línea vacía o simplemente con el fin del mensaje mismo, si es que no se requiere del cuerpo. Una línea que contiene solamente espacios o tabulaciones, aunque parezca estar vacía no lo está, y es considerada aún como parte de la cabecera del mensaje, a pesar de ser una cabecera mal formada que no cumple con los estándares de Internet.

Algunos campos de la cabecera son utilizados para la entrega host a host, mientras que otros son leídos por aplicaciones clientes y mostrados con fines informativos al usuario,

por ejemplo los campos *From* y *To* (remitente y destinatario). Aunque en los estándares se sugiere un orden en la ubicación de cada uno de los campos, en la práctica los campos pueden estar en cualquier orden. En el **Capitulo 5** se mostraran ejemplos de cómo pueden ser manipulados los campos de la cabecera. Aprender como leer los campos de la cabecera de un mensaje, puede ser de mucha ayuda para resolver problemas de entrega de correo entre servidores, especialmente cuando están relacionados con las entradas mostradas en los logs del sistema de correo.

La mayoría de los campos presentes en la cabecera de un correo electrónico, una vez que este llega a su destino, son agregados por cada uno de los servidores por los cuales el correo fue encaminado; sin embargo, como se dijo antes alguna información es agregada por la aplicación cliente. Por ejemplo el campo *To*, es automáticamente agregado por el MUA del usuario remitente. El valor de este campo depende particularmente de la configuración de la aplicación cliente, en todo caso su fin es informar a quien va dirigido el correo. En cuanto al campo *From*, su valor depende de la configuración inicial de la aplicación cliente y típicamente indica al usuario final, quien es el remitente del correo electrónico recibido.

Ninguna aplicación de correo electrónico o servidor SMTP puede dar fe de la autenticidad del campo *From*, está en cada uno de ellos determinar si el valor del campo es confiable o no. Por tal razón es muy fácil para un atacante enviar un correo a un usuario conocido con algún tipo de gusano o virus adjunto, falsificando el campo *From*, el usuario final abrirá el correo con solo ver el remitente (alguno de sus contactos). Lo anterior es una amenaza real, basándose en la información manejada por los capos de la cabecera de un mensaje. Pero, ¿cómo podemos identificar tal amenaza? y ¿cómo combatirla para proteger a nuestros usuarios finales?, este es un tema que será abordado en el **Capítulo 4.**

2.2.2 Cuerpo del mensaje

En un mensaje de correo electrónico el cuerpo se encuentra a continuación de la cabecera. El cuerpo del mensaje por lo general es texto plano legible sin formato, de cualquier forma, esto no siempre es así. Los estándares de Internet no solamente tratan los mensajes de correo en formato de texto ASCII, de hecho es común adjuntar archivos a un mensaje de correo electrónico, siendo estos a menudo datos binarios ininteligibles para el usuario. Además de los archivos binarios, también es posible crear a través de aplicaciones clientes cuerpos de mensajes con formato distinto al ASCII, eventualmente gracias a las extensiones tipo MIME, abordadas en el **Capitulo 2.**

Cuando se adjunta un archivo binario a un mensaje de correo electrónico, o el cuerpo del mensaje está en otro formato distinto al ASCII, el texto y el archivo binario deben ser codificados e incluidos como líneas de texto ASCII al cuerpo del mensaje de correo, un método común para realizar esto es a través del algoritmo de codificación UUEncode[3], utilizado en sistemas Unix.

Además del UUEncode, se ha desarrollado el estándar MIME[4] (Multipurpose Internet Mail Extensions) para estructurar y definir una convención de codificación y nombrado para el envío de mensajes de correo electrónico cuyo cuerpo sea compuesto por caracteres no-ASCII.

MIME debe ser soportado tanto por aplicaciones clientes como por los servidores, el cuerpo de un mensaje de correo es codificado y formateado de acuerdo a las

---

[3] UUEncode proviene de Unix to Unix Encoding, este algoritmo transforma código binario en texto. La entrada es un bloque bytes (8 bits) correspondientes a un archivo binario, no necesariamente; y la salida es un bloque de texto de varias líneas con saltos de línea LF o CRLF y caracteres de texto estándar del alfabeto UUEncode.

[4] MIME está definido en una serie de 5 documentos IETF: RFC 2045 (www.ietf.org/rfc/rfc2045.txt), RFC 2046 (www.ietf.org/rfc/rfc2046.txt), RFC 2047 (www.ietf.org/rfc/rfc2047.txt), RFC 2048 (www.ietf.org/rfc/rfc2048.txt) y RFC 2049 (www.ietf.org/rfc/rfc2049.txt).

especificaciones MIME. El cuerpo del mensaje así como los mensajes adjuntos (attachements) aparecen como una serie de caracteres ASCII. Estas líneas son decodificadas a través de la misma especificación en el destino del mensaje. La mayoría de las aplicaciones clientes de correo modernas soportan MIME.

2.3 Componentes del Sistema de Correo Electrónico

Un correo electrónico durante su recorrido desde el origen hacia su destino final es procesado por tres agentes, estos son:

1. El agente de usuario de correo (MUA).
2. El agente de transferencia de correo (MTA).
3. El agente de entrega de correo (MDA).

2.3.1 El agente de usuario de correo (MUA)

El usuario final de un sistema de correo electrónico utiliza el agente de usuario de correo para construir y componer un mensaje de correo electrónico. MUA es un término técnico utilizado para describir cualquier aplicación cliente de correo electrónico, como Microsoft Outlook, Eudora, Mozilla Thunderbird o Mutt. El MUA específicamente maneja las cabeceras del correo electrónico y el usuario usa el editor de texto dentro del MUA para componer el cuerpo del mensaje. Cuando el usuario final hace click en el botón Enviar, el mensaje es entregado al servidor de correo saliente SMTP predeterminado en la configuración inicial del agente.

2.3.2 El agente de transporte de correo (MTA)

El agente de transporte de correo (MTA) no es más que el servidor de correo saliente. MTA es un término técnico utilizado para referirse a un servidor SMTP de Internet. Un servidor de correo puede residir en el host del usuario final, sin embargo en la práctica es más probable que el servidor de correo exista como un host dedicado y administrado por

alguien especializado en el tema. El servidor de correo se encarga de determinar si el destinatario de correo se encuentra en el servidor local o en uno remoto, en este último caso el servidor iniciará una conexión SMTP con el servidor destino. Esta comunicación servidor a servidor continuará hasta que el mensaje de correo electrónico alcanza el servidor que almacena el buzón del destinatario final. En el **Capitulo 5** verá la instalación y configuración de *Postfix*, uno de los servidores SMTP mas utilizados y mejor documentados en sistemas Unix.

2.3.3 El agente de entrega de correo (MTA)

Una vez que el servidor final de correo electrónico recibe el mensaje, este es manipulado por el *agente de entrega de correo (MDA)*. El MDA también llamado a veces *agente de entrega local (LDA)*, es el término técnico para el mecanismo que entrega los mensajes de correo a los buzones finales, en muchas implementaciones estos mecanismos son completamente transparentes**.** Sin embargo como se verá en el **Capitulo 5**, existen ciertas aplicaciones como amavisd-new que procesan el mensaje (filtrando y escaneando) antes de ser entregado al buzón. Estas aplicaciones son muy útiles para el escaneo, desinfección y marcado de correos electrónicos que podrían contener código malicioso.

Cuando el mensaje de correo es entregado al destino final, el destinatario puede acceder a los mensajes de distintas maneras. Una de las formas más utilizadas para descargar el correo es a través del protocolo de oficina de correos versión 3 (POP3), sin embargo un método mas robusto es brindado por el protocolo de acceso a mensajes de internet (IMAP). Ambos servicios de acceso a buzones pueden ser considerados como los estándares actuales para el acceso o descarga de correos desde una computadora personal. Estos protocolos son completamente independientes uno del otro, pero ambos servicios pueden ser ofrecidos desde un mismo Servidor, brindando la flexibilidad al usuario final de escoger el protocolo que más se adapte a sus necesidades. Se dará una explicación detallada de los protocolos POP3 e IMAP en el **Capítulo 3** e

implementaciones reales en el **Capitulo 5**. En la *figura 2.2* se muestra el ciclo completo de un mensaje de correo electrónico, desde que sale de la aplicación cliente del usuario, hasta que llega a su destinatario final.

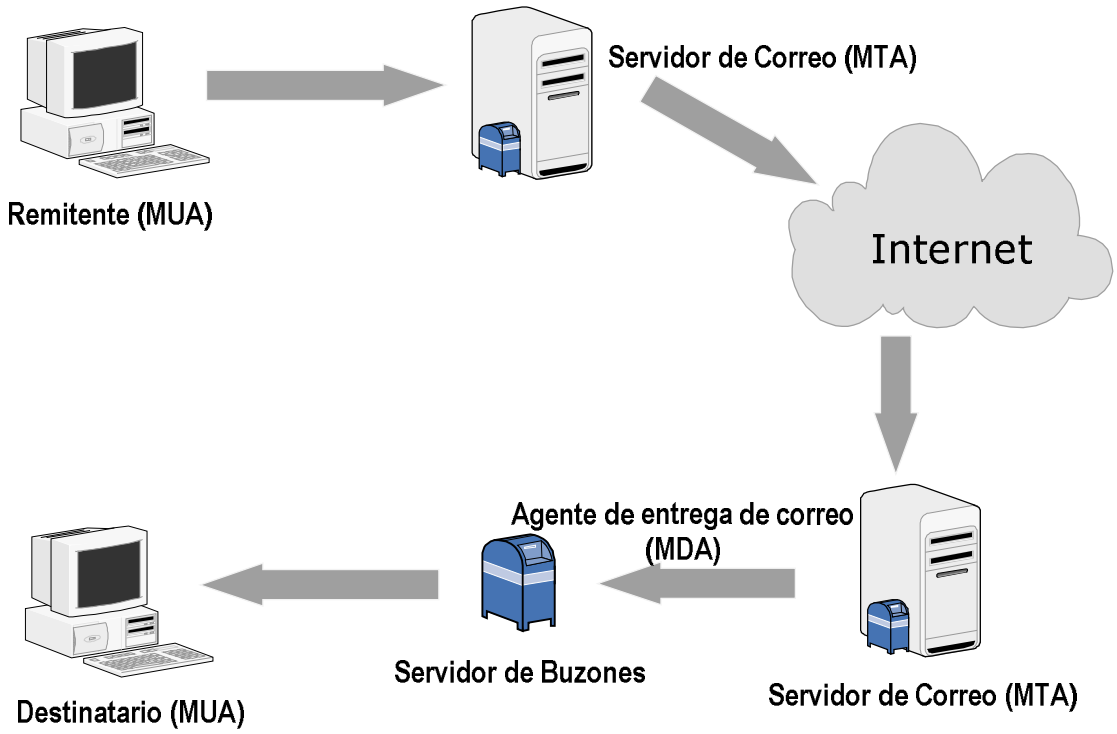


Figura 2.2 *Ciclo de un correo electrónico*

2.4 Implementaciones MTA

En el *sección 2.3.2* se trató de explicar a rasgos generales que es y cuál es la función del MTA, para aclarar un poco más a continuación se dará una breve introducción de algunos de los servidores SMTP más populares en la actualidad:

2.4.1 Sendmail

Originalmente escrito por Eric Allman mientras era estudiante de la Universidad de Berkeley, ha tenido un largo desarrollo desde los origines del Internet mismo, que lo ha llevado a ser considerado como el servidor SMTP más popular de Internet, corre bajo sistemas Unix y aunque se le critica por su alto número de fallos[5] de seguridad encontrados (la mayoría parchados) últimamente sus autores han trabajado mucho en cambiar esta situación. *Sendmail* es capaz de brindar soluciones que se adapten a las exigencias de Administradores con mucha experiencia, sin embargo esto se paga con la complejidad que conlleva su configuración.

2.4.2 Postfix

Postfix fue creado como una alternativa más rápida, fácil de administrar y segura al ampliamente utilizado *Sendmail*. Originalmente era conocido como *VMailer* e *IBM Secure Mailer*, fue escrito por Wietse Venema durante su estancia en el *Thomas J. Watson Research Center de IBM*.

No se necesita ser un experto para echar andar *Postfix*, es mas sus configuraciones por defecto son lo suficientemente seguras como para poner al servidor en producción. Su diseño es modular, de tal forma que cada modulo corre con los privilegios mínimos para realizar la tarea para la que está hecho. *Postfix* fue diseñando con la seguridad en mente, iniciando a un nivel más alto que el código mismo.

El rendimiento de *Postfix* es admirablemente alto porque está centrado fundamentalmente en las tareas de transporte de correo, lo que se hizo fue tratar de no reinventar la rueda, tomando funcionalidades implementadas en otras aplicaciones MTA

[5] Ejemplo de fallos graves encontradas en sendmail, se dió en el año 2003, mediante la manipulación de ciertos valores de las cabeceras de los mensajes, se podía producir un desbordamiento de búfer en el programa, cuando este realizaba la lectura de las mismas para procesar el correo. Este fallo fue considerado como grave por lo que implicaba, en ese momento sendmail era el servidor SMTP de casi el 75% de los servidores en Internet.

(sobre todo de *Sendmail*). *Postfix* brinda los medios para conectar aplicaciones externas cuando una tarea compartida está fuera del área de transporte del mensaje. *Postfix* utiliza todo el poder ofrecido por los sistemas Unix para realizar su trabajo, esta estrecha relación con el sistema operativo no solo hace más fácil el acceso a las aplicaciones externas, sino también mejora el rendimiento.

Un vistazo a las modernas tareas de manipulación y transporte de correo, revela que *Postfix* es el núcleo de lo que implica una suite de transporte de correo. En la *figura 2.3* puede ver que *Postfix* está rodeado de aplicaciones y herramientas especializadas para ayudar en el control de contenidos, conexiones y *relaying*[6] de correos.

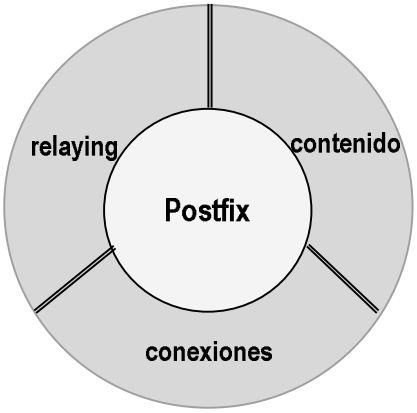


Figura 2.3 *Postfix como núcleo de una suite de transporte*

En este documento se describe como configurar *Postfix* para ser usado como relay de correo y filtro de virus y spam en una arquitectura de alta disponibilidad.

2.4.3 Qmail

Según sus autores qmail es un moderno Servidor de Correo hecho para sistemas Unix, y es el reemplazo ideal al obsoleto *sendmail.* Utiliza el formato *Maildir* para almacenar

[6] En este documento se utiliza indistintamente el término técnico en ingles *relay o relaying*, que se define como la funcionalidad brindada por un MTA que permite a un servidor de correos transmitir correos de sus usuarios a otros servidores de correo en Internet.

mensajes (un archivo por mensaje), eliminando los problemas asociados al formato mbox. Creado por Daniel J. Bernstein en búsqueda de una mejor alternativa a *Sendmail* y otros *MTA*, su arquitectura es simple, pues Bernstein se planteo que mientras más simple fuera el diseño, más seguro, estable y eficiente seria su operación. El programa ha demostrado ser absolutamente funcional en este sentido, permaneciendo virtualmente inalterado en su versión 1.03 desde entonces.

Entre sus características más importantes podemos encontrar:

- Aparte de SMTP, qmail puede utilizar también los protocolos QMTP y QMQP.
- Los mensajes recibidos en cada cacilla pueden ser filtrados a voluntad por medio de archivos ".qmail". Cada usuario puede tener su propia regla de filtros/reenvíos o incluso ejecutar algún programa o script que haga tareas arbitrarias con los mensajes entrantes.
- Además de maildir se puede utilizar el formato mbox si asi se desea.
- Las casillas pueden estar asociadas a usuarios reales (del sistema Unix) o virtuales, y también es posible atender simultáneamente varios dominios en un solo servidor.

En la actualidad todas estas características son implementadas también por otros MTA, por ejemplo *Postfix*.

2.4.4 Exim

Exim es un Agente de Transporte de Mensajes (MTA) desarrollado en la Universidad de Cambridge, corre bajo sistemas Unix y esta libremente disponible bajo términos de la licencia GPL (General Public Licence). Ofrece gran flexibilidad en la forma en cómo pueden ser enviados y leídos los correos, ofreciendo en esto último diferentes opciones. *Exim* es una buena alternativa a *Sendmail*, aunque su instalación y configuración es bastante diferente.

Entre las principales características de *Exim* podemos nombrar:

- Fácil configuración, sus archivos son fácil de entender y editar.
- Es altamente escalable, puede ser implementado para unos cuantos usuarios de oficina como también para un ISP con millones de usuarios.
- Es muy rápido comparado con otros MTA como *Sendmail*.
- Es la solución de correo por defecto en muchas distribuciones Linux, y esta también disponible para Unix.
- *Exim* es realmente fiable, está diseñado para no perder ningún mensaje aún ante caídas del hardware.
- Ofrece flexibilidad en autenticación de usuarios, puede almacenar usuarios en servidores LDAP, base de datos SQL, archivos en texto plano o incluso usuarios del sistema.

2.4.5 Microsoft Exchange Server

Exchange Server no es exactamente una implementación *MTA* como tal, más bien es un conjunto de herramientas conocidas como Software colaborativo para trabajo en grupo, incorpora un Servidor *SMTP* entre las muchas funcionalidades que brinda, a diferencia de los servicios vistos anteriormente Exchange server trabaja en sistemas Windows y su licencia es de pago, aunque existe una versión de prueba por 120 días ofrecida a desarrolladores y administradores de sistemas.

Entre sus principales características se encuentran:

- Posee un modelo administrativo basado en interfaces gráficas, aunque en su última versión (Exchange Server 2007) ofrece un complemento administrativo a través de Power Shell (similar al Shell de sistemas Unix) que permite a los administradores correr programas para automatizar muchas operaciones.
- Manejo no solo el protocolo *SMTP*, sino otros como *POP3, IMAP, LDAP y HTTP*.

- Aunque ofrece una excelente capacidad de filtrado de virus y spam con productos de terceros, proporciona filtrado interno.
- Mensajería unificada que permite a los usuarios recibir correo de voz, faxes y correos en su buzón, a la vez se puede acceder al buzón a través de teléfonos celulares y otros dispositivos móviles.
- El método de autenticación de usuarios utilizado esta bajo un directorio LDAP (Active Directory[7]).

2.5 Amenazas por Correo Electrónico

En los últimos años tanto administradores como usuarios han tenido que enfrentarse a la proliferación de numerosos tipos de amenazas a través del correo electrónico. Estas amenazas van desde los molestos correos *Spam*[8] hasta los devastadores *Virus*, que pueden llegar a colapsar completamente un sistema. Los daños no solo son ocasionados al usuario final; el rendimiento de un servidor mal protegido puede decaer dramáticamente a causa de un ataque masivo de *Spam.* En las secciones siguientes se verán tres tipos diferentes de amenazas reales a las cuales están expuestos los sistemas de correo. La forma de cómo tratar con cada una de ellas será abordada en el **Capítulo 5**.

2.5.1 Virus y Gusanos

Los virus y gusanos en correo electrónico figuran entre las amenazas más destructivas en Internet. Un virus, es un pedazo de código adjuntado a un mensaje de correo electrónico que intenta infectar o modificar otro archivo. Un gusano al igual que un virus se puede

[7] Active Directory es una tecnología propietaria de Microsoft utilizada para la administración de recursos; almacena usuarios, equipos y recursos de la red. Controla el acceso a dichos recursos por medio de usuarios y aplicaciones. Su función principal es la centralización de los recursos de la red, de esta forma almacena los objetos de forma segura en una estructura lógica basada en el Sistema de Nombres de Dominio optimizando el tráfico.

[8] El término Spam comúnmente se refiere al envío en masa de copias de un mismo mensaje de correo electrónico (con contenido comercial) por parte de un remitente generador de Spam que encubre o falsifica su identidad.

adjuntar a un correo electrónico y su fin es hacer copias de sí mismo una y otra vez, dentro de la propia maquina o sobre una red de computadoras. En otras palabras los gusanos utilizan el correo electrónico para reproducirse e infectar otras maquinas con una copia de sí mismo. Ambos (virus y gusanos) infectan una computadora explotando alguna vulnerabilidad del sistema operativo o de una aplicación especifica.

Todos los días se descubren nuevas variantes de gusanos y virus, por lo cual vendedores de antivirus e investigadores constantemente buscan como protegerse contra ellos. Los administradores experimentados deben mantener una lista de los últimos virus y gusanos propagados por correo electrónico. Un administrador debe buscar una solución antivirus que le brinde el confort y la plena seguridad de que estará bien protegido, además un buen soporte de actualizaciones. De esta forma se tendrá la confianza que los usuarios están protegidos contra la mayor cantidad posible de virus y gusanos propagados por correo electrónico.

En el **Capitulo 5** se verá como instalar y configurar el antivirus de libre distribución *Clamav*, que ayudará a mantener protegido los buzones de correo, hay que señalar que en la mayoría de los casos no basta con la protección de un solo antivirus, existen ámbitos en los cuales se pueden encontrar varios niveles de protección antivirus. Por ejemplo, es común contar con un proxy *SMTP* que escanea (contra ***virus o spam***) los correos entrantes provenientes de Internet, de la misma forma se pueden colocar otro antivirus en el servidor local que escaneará los mensajes de correo electrónico generados en la red interna.

Según estudios recientes el 80% de los virus y gusanos propagados por Internet lo hacen a través del correo electrónico.

2.5.2 Spam

Aunque seria de mucha utilidad dar una definición clara de lo que es el Spam, en la actualidad no parece existir un acuerdo común. Normalmente se le conoce como Spam al correo basura, con fines comerciales o cualquier correo recibido sin la previa autorización del usuario.

La OCDE (Organización para la Cooperación Económica y Desarrollo) clasifica las características del Spam como Primarias y Secundarias. Las características primarias incluyen los mensajes electrónicos comerciales, no solicitados y correo basura, el resto de las características son clasificadas como secundarias. La *tabla 2.1* muestra estas características:

Características primarias	Características Secundarias
Correo basura	Uso de direcciones recopiladas sin el previo consentimiento o conocimiento del usuario.
No solicitado	Uso de dirección desconocidas
	Envío a direcciones de manera repetitiva
Comercial	Envío a direcciones al azar e indiscriminadamente
	Incontrolable
	Envío de remitente anónimo y/u oculto
	Contenido ilegal u ofensivo
	Contenido falso o fraudulento

Tabla 2.1 *Características primarias y secundarias del Spam*

A pesar de la confusión y el desacuerdo en una definición precisa, existen ciertas características muy extendidas en la comunidad de Internet:

- Un Spam es un mensaje electrónico[9].

[9] Para la mayoría de los propósitos, esto se refiere a un correo electrónico, sin embargo existen otros métodos de envío de Spam, como el Servicio de Mensajería SMS utilizado en telefónica móvil.

- Un Spam no es solicitado, si el receptor ha acordado aceptar un mensaje, entonces este no es un Spam. Sin embargo, la forma en cómo y cuándo debe ser aceptado un correo no está del todo clara, especialmente cuando ya existe una relación entre el remitente y el destinatario.
- Spam es el envío de correo en grandes volúmenes. Esto implica que el remitente distribuye un gran número de mensajes idénticos y que los receptores son elegidos indiscriminadamente.

Los tres puntos anteriores definen lo que es el correo electrónico por lotes no solicitado (UBE). Puede existir otra definición de carácter comercial, la clase de mensajes resultantes se refiere a correo comercial no solicitado (UCE).

Numerosas estadísticas relacionadas al Spam han sido publicadas por muchas organizaciones, tales como proveedores de servicios de internet (ISP), compañías de investigación de mercado, universidades y empresas de seguridad informática. La mayoría de ellos coinciden que el Spam supera más del 50% del correo circulante en Internet, de igual forma el mayor porcentaje de Spam es generado por Estados Unidos y Asia. Para darnos una idea general en la figura 1.4 se muestran los datos obtenidos por un estudio hecho en Febrero del 2008 por la compañía *ieinternet.com.*

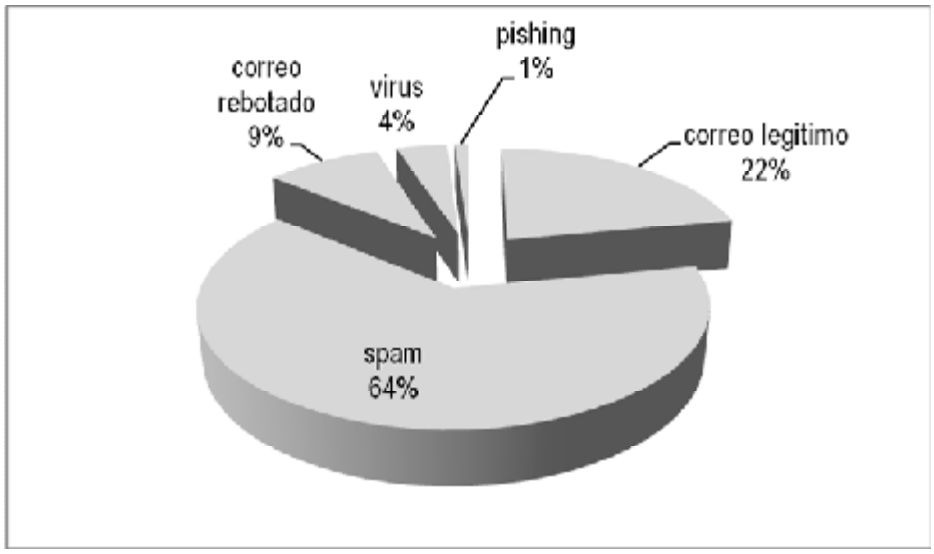


Figura 2.4 *Estadísticas globales de correo electrónico*

Viendo estos porcentajes de Spam es de esperar que nos enfrentaremos a una amenaza muy grande y sobre todo molesta, no se puede calificar como buena una implementación de sistema de correo sin un buen filtro Antispam, es por eso que en el **Capitulo 5** se tratará de configurar de manera optima Spamassassin, el servicio antispam mas utilizado en sistema Linux.

2.5.3 Phishing

Últimamente se ha hecho bastante común un nuevo tipo de ataque, perpetrado por hackers maliciosos que tratan de engañar a usuarios para que den información personal sensible con el pretexto de verificar y asegurar sus datos privados, obviamente dicha información será utilizada para beneficio propio. A la práctica de estas técnicas fraudulentas a través del correo electrónico se le conoce como Phishing[10]. El Phishing se diseña deliberadamente para que aparente ser proveniente de fuentes legítimas como bancos, tiendas en línea, etc. Por ejemplo el mensaje de correo electrónico podría sugerir

[10] Coloquialmente el término Phishing fue acuñado para describir la forma maliciosa en que los hackers utilizan el correo electrónico para atraer a la gente y persuadirlos a dar información privada de manera similar a como los pescadores utilizan la carnada para atraer a los peces.

la necesidad de actualizar los datos personales de un usuario e indicar seguir un enlace para completar el proceso, el enlace puede parecer legítimo, pero en realidad no tiene nada que ver con la entidad que pretende representar.

2.6 Extensiones a los servicios básicos de Correo Electrónico

Existen dos componentes adicionales que no forman parte de los servicios elementales de un sistema de correo, estos componentes nos son necesarios en una implementación completa de correo electrónico; sin embargo son servicios que serán instalados porque brindarán en primera instancia la flexibilidad al usuario de accesar a su buzón de manera remota no importando el lugar donde se encuentre en Internet (Correo Web) y por otra parte ofrecerá un servicio adicional que le permitirá la distribución masiva de mensajes entre múltiples usuarios (lista de correo).

2.6.1 Correo Web

Es una de las razones por las cuales el correo electrónico se ha vuelto tan popular, permite a un usuario revisar y administrar su buzón utilizando la Web. La utilización del correo electrónico se reduciría dramáticamente si no se contara con el acceso a través de HTTP[11] (Protocolo de Transferencia de Hipertexto). El correo web brinda acceso al buzón independientemente del sistema operativo utilizado y de la zona geográfica donde se encuentre el usuario, solo basta con tener una conexión a Internet. Ofrecer una solución de correo web para enviar y recibir correo electrónico es un servicio de valor agregado que aumentará la productividad de los usuarios.

El correo web se ha convertido en unos de los recursos mas utilizados en Internet gracias a compañías que ofrecen el servicio en forma gratuita como Hotmail, Yahoo o Gmail. Lo

[11] HTTP fue desarrollado por el W3C (World Wide Web Consortium) y la IETF (Internet Engineering Task Force) en una serie de RFCs, siendo el más importante de ellos el el RFC 2616, que especifica la versión 1.1 del protocolo.

mas seguro es que usted la primera ves que hizo uso del correo electrónico fue a través del correo web. En el **capitulo 5** se abordará como instalar y configurar el servicio de correo web, el software utilizados será horde webmail, un proyecto open source con licencia GPL.

2.6.2 Listas de Correo

Si estuviéramos en la necesidad de enviar continuamente correos electrónicos a cientos de usuarios, no sería lo mas práctico enviar una copia a cada uno ellos y a la vez irlos agregando a nuestra libreta de direcciones. Existe una manera más amigable de hacer lo mismo, a través de una solo dirección de correo que englobará todas las buzones subscritos a dicha cuenta, a este servicio se le conoce como lista de correo electrónico.

Las listas de correo electrónico están diseñadas para manejar y compartir información entre múltiples usuarios, cuando usted envía un mensaje a la lista de correo `lista@unan.edu.ni`, el correo llegará masivamente a cada uno de los usuarios subscritos, aunque dependiendo de cómo este configurada la lista el usuario podrá tener o no la posibilidad de enviar el correo; lo común es que cuando un usuario cualquiera envía un correo a una lista de correo, el administrador de la lista recibirá un correo cuyo contenido indica que un remitente está intentando enviar un correo, el administrador dependiendo del contenido y del remitente del correo tomará la decisión si dejar pasar el mensaje o no.

Por lo general las listas de correo son utilizadas por las distintas comunidades de Internet para tratar temas de discusión, el objetivo es establecer un debate público, cuyo contenido esta accesible a cada uno de los usuarios subscritos a la lista.

3. PROTOCOLOS DE CORREO ELECTRÓNICO

No existe protocolo alguno que por si solo cubra todas las características y necesidades de una implementación avanzada de correo electrónico. Desde sus inicios los documentos y estándares propuestos por el IETF han tratado de cubrir tareas sencillas, simplificando y modularizando las tareas de envío y recepción de mensajes. Por ejemplo el RFC 821 (SMTP) explica los métodos y respuestas intercambiados entre un cliente y un servidor para la transacción de mensajes de correo electrónico mientras que el RFC 1939 (POP3) trata un protocolo utilizado por los usuarios para accesar al buzón. Así mismo existen estándares (a veces propuestas para convertirse en estándar) que se refieren a los siguientes temas:

- Formato de cabeceras de los mensajes.
- Extensiones a los servicios del protocolo SMTP.
- Enrutamiento de mensajes de correo electrónico.
- Extensiones de Seguridad para correo electrónico.
- Tipos de formatos de cuerpo de mensajes.

3.1 DNS y Encaminamiento de Correo Electrónico

Existe una relación muy estrecha entre el Sistema de Nombre de Dominio y el Correo electrónico. Los sistemas de correo encaminan los mensajes basándose en la información brindada por el DNS, es por eso que su buen funcionamiento está en total dependencia de la correcta configuración del sistema DNS. En esta sección se explica que es el DNS, como funciona, los servicios que ofrece al correo electrónico y como utiliza este dicha información para encaminar los mensajes.

3.1.1 El Sistema de Nombres de Dominio

El sistema de nombres de dominio DNS (Domain Name System) es necesario para poder referenciar los recursos de una red mediante nombres que sean cómodos de manejar por los usuarios. El sistema de nombres traslada estas cadenas proporcionadas por los usuarios a direcciones de red manejables por los programas y dispositivos de la red.

Adicionalmente el sistema de nombres facilita la organización distribuida de los servidores de nombres, posibilitando la delegación del mantenimiento de ciertos conjuntos de recursos a una organización y evitando así la centralización del servicio.

3.1.2 Resolución directa e inversa

Se dice que se hace una *resolución directa* cuando el usuario proporciona al servidor de nombres el nombre de un recurso de la red y el servidor le devuelve la dirección de red de dicho recurso.

La resolución inversa es el proceso contrario, el usuario proporciona al servidor una dirección de red de un recurso y obtiene el nombre o nombres con los cuales es conocido dicho recurso en la red. El estándar no obliga a un servidor implementar la resolución inversa, lo define como un servicio opcional.

3.1.3 Elementos del servicio DNS

El RFC 1034 establece que el Sistema de Nombres de Dominio está compuesto por los siguientes tres elementos:

1. **El espacio de nombres de dominio y registro de recursos**: son especificaciones para un árbol estructurado de espacio de nombres y datos asociados con los nombres. Cada nodo del árbol tiene una etiqueta de un tamaño máximo de 63 caracteres y además tiene asociado un conjunto de recursos. El nombre de dominio de un nodo es la lista de etiquetas desde el nodo raíz.
2. **Servidores de nombres:** son programas que manejan parte de la estructura del árbol, la cual tienen almacenada localmente en sus bases de datos. Un servidor de

nombres podrá contestar a solicitudes que se hagan referentes a su parte del árbol y también podrá ofrecer referencias a otros servidores que conozcan las restantes partes del dominio de nombres. Se dice que un servidor de nombres está autorizado sobre cierta información cuando ésta se mantiene localmente en este servidor.

3. **Resolvers:** son programas que hacen de intermediario entre los procesos cliente y toda la estructura de DNS. Un resolver debe poder acceder al menos a un servidor directamente, para obtener de él la información solicitada o, en su defecto, obtener referencias a otros servidores a través de los cuales continuar la búsqueda. El resolver generalmente ofrece un conjunto de llamadas al sistema que permiten a cualquier proceso cliente realizar peticiones DNS.

3.1.4 Servidores primarios y secundarios

Una zona (sección del árbol que forma el dominio de nombres), será mantenida por un servidor de nombres, el cual será el servidor primario de esa zona. Las consultas de nodos en esta zona serán contestadas por este servidor primario, el cual dará respuestas autorizadas.

Es obligatorio que exista un servidor secundario que respalde al primario, para que las consultas a esta zona no se vean interrumpidas en caso de fallo del servidor primario.

3.1.5 Consultas recursivas e iterativas

Con una consulta recursiva, el cliente realiza una petición y se le devuelve directamente la respuesta a esa petición o un error. Si el servidor consultado conoce la respuesta la devuelve directamente al cliente, en caso de no conocerla contactará con otro servidor de nombres para preguntarle a él. Esta consulta en cadena se puede extender hasta que el servidor reciba una respuesta o un error que pueda pasar al cliente.

En una consulta iterativa, el cliente además de una respuesta directa o un error, puede recibir una referencia a otro servidor de nombres para que realice la consulta en su lugar. El servidor de nombres consultado puede devolver la respuesta en caso de conocerla, si

no la conoce devolverá al cliente una referencia a un servidor para que le realice la consulta.

Mientras que el método de consulta iterativa siempre está presente, las consultas recursivas son opcionales, y un servidor de nombres no tiene por qué aceptarlas.

3.1.6 Registros de Recursos (Resource Records)

Los registros de recursos (RR) son estructuras de datos que contienen la información intercambiada entre un cliente y el servidor de nombres. El formato de un RRs es el siguiente:

Figura 3.1 *Formato de un Registro de Recurso DNS*

Donde:

- Propietario: es el nombre de dominio donde se encuentra el registro, comúnmente implícito en vez de formar parte integral del RR. Por ejemplo, muchos servidores de nombres internamente forman una estructura de árbol o hash para el espacio de nombres y encadenan RRs fuera de los nodos. El resto de las partes del RR son la cabecera indexada (tipo, clase, TTL) que permanece para todos los RRs, y a parte variable (RDATA) que mantiene las necesidades del recurso descrito.

- Tipo: es un valor codificado de 16 bits que especifica el tipo de recurso en este registro de recursos, los tipos más comunes son:

-

Tipo	Descripcion
A	(Address) una direccion de host
CNAME	(Canoncial Name) identifica un nombre canonico de un alias
HINFO	(Host Information) identifica información del host como CPU y el Sistema Operativo
MX	(Mail Exchange) identifica un servidor de correo para el dominio
NS	(Name Server) el servidro de nombres autoritativo
PTR	(Pointer) puntero a un nombre de dominio
SOA	(Start Of Authority) identifica el comienzo de la zona autoritativa

Tabla 3.1 *Tipos de Registros de Recursos DNS*

- Clase: un valor codificado de 16 bits que identifica la clase de un recurso, las clases más comunes son:

Tipo	Descripcion
IN	(Internet) el sistema de Internet
CH	(Chaos) el sistema Caos

Tabla 3.2 *Clases de Registros de Recursos DNS*

- TTL: tiempo de vida del RR. Este campo es un entero de 32 bits, que identifica unidades de segundo y fundamentalmente lo utilizan los resolutores cuando cachean RRs. El TTL describe el tiempo de duración que puede estar cacheado un RR antes de que pueda ser eliminado, en otras palabras es el tiempo límite que un RR permanece en cache.

- Longitud: longitud del campo de datos (RDATA).
- RDATA: es el tipo y a veces los datos que dependen de la clase que describen el recurso, los datos de registro más utilizados se muestran en la Tabla 3.3.

Tipo	Descripcion
A	para la clase IN, almacena una direccion IP de 32 bits
CNAME	especifica un nombre de dominio

MX	Un valor de de preferencia de 16 bits (si tienes menos mejor) seguido de un nombre de de host que actúe como servidor de correo para el dominio propietario
NS	Especifica un host que será el que actúe como servidor de nombres para la clase y dominio especificados
PTR	Un nombre del espacio de nombres, resolución inversa
SOA	Almacena varios valores como servidor de nombres (NS), correo electrónico, numero de serie, intervalo de actualización, intervalo de reintento, finalización del servicio y TTL

Tabla 3.3 *Registros de Recurso RDATA*

3.1.7 Consultas y respuestas

La principal actividad de un servidor de nombres es responder consultas estándar. Tanto la consulta como su respuesta tienen un formato de mensaje estándar descrito en el RFC 1035. Una consulta está compuesta por un QTYPE, QCLASS y un QNAME, que son los tipos, clases y nombres de la información deseada.

La forma en que el servidor de nombres depende de si opera en recursivo o no:

- No recursivo: es el modo más simple para el servidor, puede responder consultas utilizando solo la información local; la respuesta puede contener un error, la respuesta o una referencia a otro servidor cercano a la respuesta. Toda implementación DNS debe implementar consultas no-recursivas.

- Recursivo: el modo más simple para el cliente. En este modo el servidor de nombre tiene el rol de un resolutor y devuelve un error o la respuesta, pero nunca una referencia a otro servidor. Este servicio es opcional en el servidor de nombres.

Un servicio recursivo es útil en diversas situaciones:

- Un cliente relativo que solo pueda soportar una respuesta directa a la consulta.
- Un cliente que necesite pasar de un protocolo a otro o que tenga otros límites y pueda enviar consultas a un servidor que puede actuar como intermediario.

- Una red donde queramos centralizar la cache en vez de tener una caché separada para cada cliente.

Un servicio no-recursivo es apropiado si el origen es capaz de seguir las referencias y puede guardar información para futuras consultas. Un servicio recursivo está limitado a casos donde tanto el cliente como el servidor de nombres están de acuerdo en utilizarlo.

3.1.8 Encaminamiento de Correo Electrónico[12]

Para el envío de un correo electrónico a un dominio, no basta con abrir una conexión SMTP al dominio dado, es cierto que para ciertos casos esto suele funcionar, sin embargo en la mayoría de las situaciones esto no funciona. Por ejemplo, en la práctica un servidor SMTP remoto no puede simplemente establecer una conexión SMTP al dominio *est.unan.edu.ni*, sino que en su lugar tiene que preguntar al sistema de dominios donde deben entregarse los mensajes dirigidos a *est.unan.edu.ni*, entonces el sistema de dominios puede redirigir al gestor de correo a un host completamente diferente, en este caso a ALT1.ASPMX.L.GOOGLE.COM. Sin embargo realmente la respuesta del servicio DNS es un poco más complicada, indicando que se dispone de una gama más amplia de servidores SMTP capaces de procesar el correo para este dominio:

```
est.unan.edu.ni mail is handled by 10 ASPMX3.GOOGLEMAIL.COM.
est.unan.edu.ni mail is handled by 10 ASPMX4.GOOGLEMAIL.COM.
est.unan.edu.ni mail is handled by 10 ASPMX5.GOOGLEMAIL.COM.
est.unan.edu.ni mail is handled by 5 ASPMX.L.GOOGLE.COM.
est.unan.edu.ni mail is handled by 5 ALT1.ASPMX.L.GOOGLE.COM.
est.unan.edu.ni mail is handled by 10 ALT2.ASPMX.L.GOOGLE.COM.
est.unan.edu.ni mail is handled by 10 ASPMX2.GOOGLEMAIL.COM.
```

Los programas gestores de correo electrónico (Servidores SMTP) requieren la información brindada por el servicio DNS para el encaminamiento de los mensajes, esta información solo indica como debe ser enrutado el mensaje y no la manera en como

[12] El encaminamiento de correo electrónico es tratado en el RFC 974 (www.ietf.org/rfc/rfc949.txt).

hacen los servidores para realizar la entrega final de los mensajes al buzón de usuario, es decir el DNS mantiene información en el registro MX de cuál o cuáles son los servidores encargados de gestionar el correo electrónico para el dominio sobre el cual se tiene la autoridad de la zona, sin embargo la entrega al buzón de correo destino es tarea del protocolo SMTP.

Una vez que el servidor SMTP origen conoce cual es el MX del dominio, nuevamente realiza una consulta al servidor DNS, preguntándole en esta ocasión cual es la dirección IP del MX devuelto. En este momento el servidor ya es capaz de abrir la conexión SMTP con el servidor destino y llevar a cabo las tareas de intercambio de mensajes. La *figura 3.2* muestra este proceso:

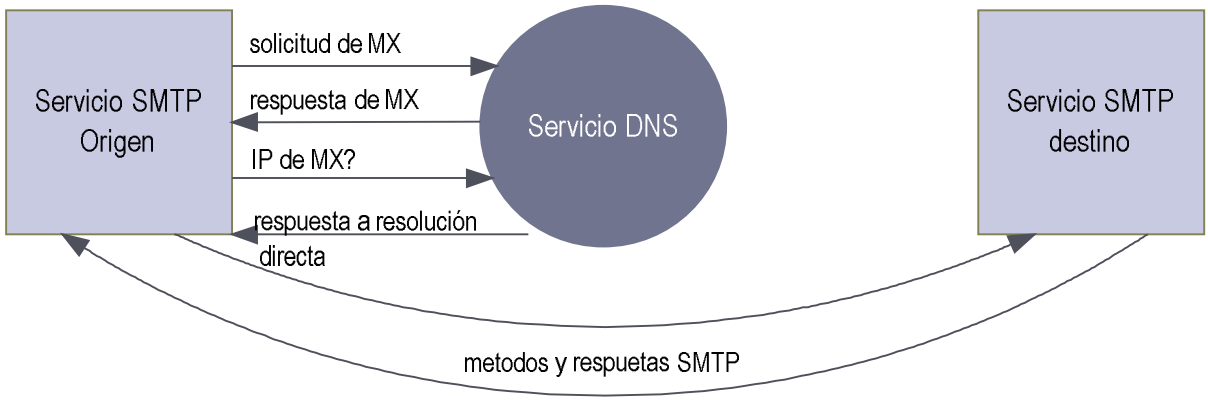


Figura 3.2 *Encaminamiento de correo electrónico*

Como se observó en la sección anterior los servidores DNS guardan la información en una serie de Registros de Recursos (RR), cada uno de los cuales contiene información particular acerca de un nombre de dominio dado (por lo general un host). La forma mas simple de entender lo que es un RR es viéndolo como un par estructurado de datos, un nombre de dominio asociado a sus datos. En el sentido estricto de utilizar la información del DNS para encaminar los mensajes de correo electrónico, el sistema dispone de un registro conocido como MX (Mail Exchanger). Cada MX asocia un nombre de dominio con dos datos, un número de preferencia (un entero de 16 bits sin signo) y el nombre de

un host. A continuación puede ver como luce en una implementación (sistema bind) comúnmente utilizada en sistemas Unix.

```
unan.edu.ni.            IN MX 05          mail.unan.edu.ni.
```

El número de preferencia se utiliza para indicar en que orden el gestor de correo debería intentar entregar el correo a los servidores MX, siendo el MX con número de preferencia menor al que se intentaría primero la entrega. Se permiten varios MXs con el mismo número de preferencia, teniendo estos la misma prioridad.

Además de la información de correo, los servidores guardan otros tipos de RRs que los gestores de correo pueden encontrar y elegir usar. Estos son: el RR nombre canónico (CNAME), que simplemente establece que el nombre de dominio solicitado es realmente un alias de otro nombre de dominio, el nombre auténtico, o canónico; y el RR Well Known Service (WKS, Servicios Públicos), que guarda la información acerca de qué servicios de red (como SMTP) soporta un nombre de dominio dado.

3.1.9 Interpretando la lista de RRs de tipo MX

Los gestores de correo deben elegir a que nombres deben intentar enviar un mensaje de los obtenidos de una lista de RRs te tipo MX.

Para explicar como hacen los servidores para elegir el MX adecuado supondremos que se ha hecho una consulta de tipo MX desde el host local con fqdn *ns1.midominio.com.ni* hacia el dominio remoto *est.unan.edu.ni*, para esto se presentan las siguientes dos situaciones*:*

1. Es posible que la lista de MXs en la contestación a la consulta esté vacía. Esto constituye un caso especial. Si la lista está vacía, los servidores SMTP tratan este caso como si contuviera un único RR, es decir un RR de tipo MX con un valor de preferencia 0 y como nombre de host remoto *est.unan.edu.ni*. El servidor de

correo no debería realizar más procesamiento de la lista, sino que debería intentar entregar el mensaje directamente a *est.unan.edu.ni*. La idea aquí es que si un dominio falla en comunicar cualquier información acerca de un nombre particular, se le concede el beneficio de la duda y se intenta la entrega.

2. Si la lista no está vacía, el gestor de correo debería eliminar los RRs irrelevantes de la lista de acuerdo a los siguientes pasos:
    - Para cada MX que compone la lista, se debería realizar una consulta WKS para averiguar si el nombre de dominio devuelto en la lista soporta el servicio de correo electrónico. Los RRs de tipo MXs que contengan nombres de dominio que no soportan el servicio deberían ser descartados. Este paso es opcional pero se recomienda realizarlo.
    - Si el nombre de dominio local *(ns1.midominio.com.ni)* desde donde se ha realizado la consulta aparece como un RR de tipo MX, todos los RRs de tipo MX con un número de preferencia mayor o igual que el del host local deben ser descartados.

Después de eliminar los RRs irrelevantes, la lista puede que, de nuevo, este vacía. Esto es considerando como una condición de error y puede ocurrir de varias formas. El caso más simple consiste en que las consultas WKS hayan descubierto que ninguno de los hosts de la lista soportan el servicio deseado. El mensaje entonces queda considerado como imposible de enviar, aunque el host local podría intentar enviar el correo directamente al host identificado por el dominio *est.unan.edu.ni*.

Si la lista de RRs de tipo MX no está vacía, el gestor de correo debería intentar enviar el mensaje a los MXs por orden (comenzando por los de número de preferencia menor). El servidor de correo local está obligado a intentar el envío al MX con valor más bajo. Comúnmente se intentará enviar a los MXs en orden hasta que uno de los MXs acepte el mensaje, o hasta que se haya probado con cada uno de los que conforman la lista final. Se puede dar el caso en que varios MXs tengan el mismo número de preferencia, para lo

cual el servidor SMTP puede probar de manera ordenada con cada uno de ellos, hasta que uno de ellos acepte el mensaje.

3.2 Protocolo Simple de Envío de Correo (SMTP)

El objetivo de SMTP (Simple Mail Transfer Protocol) es enviar correo de una manera fiable y eficiente. SMTP es independiente del sistema de transmisión y únicamente requiere un canal de envío de datos fiable y ordenado.

Una de las características de SMTP es la capacidad de funcionar a través de los entornos de servicios de transporte. Un servicio de transporte proporciona un entorno de comunicación entre procesos. Este entorno puede cubrir una o varias redes.

3.2.1 Funcionamiento de SMTP

Como todos los sistemas de comunicación, tendremos que distinguir a los dos implicados. El emisor de correo y el receptor de correo.

El protocolo SMTP está basado en el siguiente modelo de comunicación:
Como resultado de una petición de envío de correo por parte del usuario, el emisor SMTP, que actúa como cliente, establece una conexión TCP con el receptor SMTP, que hace la función de servidor. Este ultimo puede ser el equipo final al cual va dirigido el mensaje o bien un sistema intermedio. El puerto definido para llevar a cabo una conexión SMTP es el 25.

Los comandos SMTP son generados por el cliente y son enviados hacia el servidor. Las respuestas a dichos comandos, por su parte, funcionan de manera inversa, es decir, son enviadas por el servidor hacia el cliente.

Una vez que el canal ha sido establecido (comando HELO), el emisor envía un comando MAIL indicando quien envía el mensaje. Si el receptor SMTP puede aceptar correo, responderá con una respuesta OK. Posteriormente, el emisor SMTP envía un comando RCPT (recipient) identificando al destinatario del mensaje. Si el receptor SMTP puede aceptar mensajes para dicho destinatario, responderá mediante un OK, en caso contrario rechazará el mensaje. Tanto el cliente como el servidor pueden negociar varios destinatarios. Una vez que han llegado a un acuerdo, el cliente envía los datos del mensaje (comando DATA). Si el servidor recibe los datos de manera correcta, responderá con OK. El canal se cerrará con el comando QUIT.

SMTP proporciona diferentes mecanismos para la transmisión de correo:

- Directamente desde el host del usuario emisor hasta el host del usuario receptor, cuando ambos host están conectados al mismo servicio de transporte.
- A través de uno o más servidores SMTP (que transmiten el mensaje), cuando origen y destino no se encuentran en el mismo servicio de transporte.

Para poder proporcionar la capacidad de retransmisión, el servidor SMTP necesita conocer el nombre del host destino, así como el nombre del buzón al cual va dirigido el mensaje de correo.

El argumento que admite el comando MAIL especifica el emisor del mensaje. También conocido como camino inverso (reverse-path). Este camino se va modificando a medida que el mensaje atraviesa diferentes servidores de correo y se utiliza para poder obtener el camino de vuelta cuando exista la necesidad de notificar mensajes de error.
El argumento que admite el comando RCPT especifica a quien va dirigido el mensaje (forward-path).

Cuando el mensaje va dirigido a varios destinatarios dentro de un mismo host, el servidor SMTP únicamente envía una copia de dicho mensaje.

3.2.2 Comandos y respuestas SMTP

Los comandos y respuestas pueden ir escritos en letras mayúsculas, minúsculas o una mezcla de ambas. Sin embargo, los nombres del emisor y receptor deben escribirse de la forma en que fueron definidos cuando se dieron de alta, debido a que muchos servidores realizan una distinción entre letras mayúsculas y minúsculas.

Estos comandos y respuestas se componen de caracteres ASCII. Cuando el servicio de transporte proporciona un canal de transmisión de 8 bits, cada carácter de 7 bits se envía colocando un cero en el bit de mayor peso.

Tras establecer el canal se darán tres pasos para la realización de una transacción SMTP. El primero de ellos se produce al escribir el comando MAIL, el cual identifica al usuario emisor. A continuación sigue uno o varios comandos de tipo RCPT, los cuales identifican a los usuarios finales a quienes se les quiere hacer llegar el mensaje. El tercer paso es el envío del mensaje de correo en sí, precedido por la orden DATA. El final del mensaje se indica mediante la aparición de una línea cuyo primer carácter es un punto, el esquema es el siguiente:

1. Avisar al receptor SMTP de que una nueva transacción está comenzando y por lo tanto vaciar los buffers de memoria con el motivo de dejar espacio libre para el mensaje entrante:

   ```
   MAIL FROM: <origen> <CRLF>
   ```

   La sintaxis utilizada en el argumento `<origen>` generalmente indica que el texto que aparece entre los corchetes es simbólico y por lo tanto todo el argumento se debe sustituir por una cadena de texto. Sin embargo, dichos ángulos se necesitan para indicar el nombre del usuario origen cuando se trata de un camino inverso. La secuencia `<CRLF>` indica la pulsación de la tecla ENTER (que produce un

retorno de carro y un avance de línea). En caso de que el receptor SMTP acepte el comando, este envía una respuesta `250 OK`.

2. El segundo paso es el envío del comando RCPT:

```
RCPT TO: <destino> <CRLF>
```

Este comando identifica el destino al cual va dirigido el mensaje. En caso de ser aceptado, el receptor SMTP envía una respuesta `250 OK`. De tratarse de un destino desconocido, el receptor SMTP envía una respuesta con el identificador 550.

Tanto `<origen>` como `<destino>` pueden indicar más de un usuario. Sin embargo, se recomienda que tanto el host que envía el mensaje como aquel que lo recibe sean los primero de sus respectivas listas.

3. El tercer paso se lleva a cabo mediante el argumento DATA:

```
DATA <CRLF>
```

Este pasova a permitir el envío de la información desde el origen al destino. Si el argumento es aceptado, inmediatamente se envía una respuesta con el código 354. Desde ese momento, y hasta que se encuentre una línea del mensaje que comienza con un punto, el receptor considera que todas las líneas que le llegan forman el mensaje.

Los tres pasos anteriores se resumen en el siguiente ejemplo real:

```
MAIL FROM: ecuevas@unan.edu.ni
250 OK
RCPT TO: edison@est.unan.edu.ni
250 OK
DATA
354 Start mail input, end with <CRLF>.<CRLF>
```

```
Este es el contenido del mensaje
Para finalizar lo hacemos con una línea que comience
con el carácter “.”+la tecla enter.
.
```

Existen situaciones en las que el destinatario es incorrecto pero el receptor SMTP conoce la dirección correcta. En tales casos, el receptor puede enviar al emisor:

- Una respuesta indicando que el receptor SMTP sabe que el buzón del usuario destino se encuentra en otro servidor e indica el destino que se debe utilizar en futuras ocasiones. La responsabilidad del correcto envío al usuario destino recae en el receptor SMTP.

  ```
  251 User not local; will forward to <destino>
  ```

- O una segunda respuesta indicando que el receptor SMTP sabe que el buzón del usuario esta en otro host e indica la dirección correcta que se debe utilizar. El receptor SMTP no acepta el envío de correo para el usuario indicado. La responsabilidad del envío recae en manos del emisor del mensaje.

  ```
  551 User not local; please try <destino>
  ```

La *tabla 3.4* resume los comandos SMTP mas utilizados, más adelante se explicara en detalle algunos de ellos.

Comando	Descripción
HELO	Identifica el remitente al destinatario
MAIL FROM	Identifica una transacción de correo e identifica al emisor
RCPT TO	Se utiliza para identificar un destinatario individual. Si se necesita identificar múltiples destinatarios es necesario repetir el comando
DATA	Permite enviar una serie de líneas de texto. El tamaño máximo de una línea es de 1000caracteres. Cada línea va seguida de un retorno de carro y avance de línea <CRLF>. La ultima línea debe llevar únicamente el carácter punto “.” Seguido de <CRLF>.
RSET	Aborta la transacción de correo actual.

NOOP	No operación. Indica al extremo que envíe una respuesta positiva.
QUIT	Pide al otro extremo que envíe una respuesta positiva y cierre la conexión.
VRFY	Pide al receptor que confirme que un nombre identifica a un destinatario valido.
EXPN	Pide al receptor la confirmación de una lista de correo y que devuelva los nombres de los usuarios de dicha lista.
HELP	Pide al otro extremo información sobre los comandos disponibles.
TURN	El emisor pide que se inviertan los papeles, para poder actuar como receptor. El receptor puede negarse a dicha petición.
SOML	Si el destinatario está conectado, entrega el mensaje directamente al terminal, en caso contrario lo entrega como correo convencional.
SAML	Entrega del mensaje en el buzón del destinatario. En caso de estar conectado también lo hace al terminal.
SEND	Si el destinatario está conectado, entrega el mensaje directamente al terminal.

Tabla 3.4 *Comandos SMTP*

En la *tabla 3.5* se describen las respuestas a los comandos anteriores, cada tipo de respuesta está identificada por un número de manera ascendente:

Código	Descripción
211	System status. Estado del sistema
214	Help message. Mensaje de ayuda
220	<domain> service ready. Servicio preparado
221	<domain> service clsing transmission cannel. Servicio cerrando e canal de transmisión
250	Rquested mail action okay completed. Solicitud competada con éxito
251	User not local; will forward to <forward-path>. Usuario no local, se enviara a …
354	Start mail input; end with <CRLF>.<CRLF>. Introduzca el texto; finalice con <CRLF>.<CRLF>
421	<domain> serice not available. Servicio no disponible
450	Requested mail action not taken; mailbox unavailable. Solicitud de correo no ejecutada, buzon no disponible. (buzon ocupado)
451	Requested action aborted: local error in processing. Acción no ejecutada, error local de procesamiento
452	Requested action not taken: Insufficient system storage. Acción no ejecutada, insuficiente espacio de almacenamiento en el sistema
500	Syntax error. Command unrecongnized. Error de sintaxis, comando no reconocido

501	Syntax error in parameters or arguments. Error de sintaxis en parámetros o argumentos.
502	Command not implemented. Comando no implementado
503	Bad sequence of commands. Secuencia de comandos errónea
504	Command parameter not implemented. Parámetro no implementado
550	Requested action not taken; mailbox unavailable. Solicitud no ejecutada, buzón no disponible
551	User not local, please try <forward-path>. Usuario no local, pruebe …
552	Requested mail action aborted. Acción de correo solicitada abortada
553	Requested action not taken.. Solicitud no realizada. (error de sintaxis)
554	Transaction failed. Fallo en la transacción

Tabla 3.5 *Respuestas SMTP*

3.2.3 Comandos de verificación de usuarios

SMTP proporciona características adicionales, tales como verificar el nombre de un usuario o expandir una lista de correo. Para ello se utilizan los comandos VRFY y EXPN.

Para el comando VRFY, el argumento utilizado es el nombre de un usuario. La respuesta puede incluir el nombre completo del usuario así como su buzón. Para el comando EXPN, el argumento hace referencia a una lista de distribución y la respuesta puede incluir el nombre completo de todos los usuarios y sus respectivos buzones.

3.2.4 Comandos de envío de mensajes

El propósito principal de SMTP es enviar mensajes a los buzones de los usuarios, sin embargo, algunos host proporcionan un servicio de entrega al terminal del usuario. Los comandos de entrega de correo son los siguientes:

```
SEND FROM: <usuario@host> <CRLF>
SOML FROM: <usuario@host> <CRLF>
SAML FROM: <usuario@host> <CRLF>
```

El comando SEND requiere que el mensaje de correo se envíe al terminal del usuario.

El comando SOML se utiliza para enviar mensajes de correo al terminal, o al buzón, si el terminal no está activo.

El comando SAML se utiliza para que el mensaje de correo se envíe tanto al terminal del usuario como al buzón del mismo. Se considera que el envío se ha realizado con éxito si el mensaje se ha enviado correctamente al buzón del usuario.

3.2.5 Comandos de apertura y cierre del canal

En el proceso de apertura de un canal, se produce un intercambio de datos que tiene como fin la identificaion de los hosts.

Para realizar la apertura del canal se utiliza el comando HELO. Tiene la siguiente sintaxis:

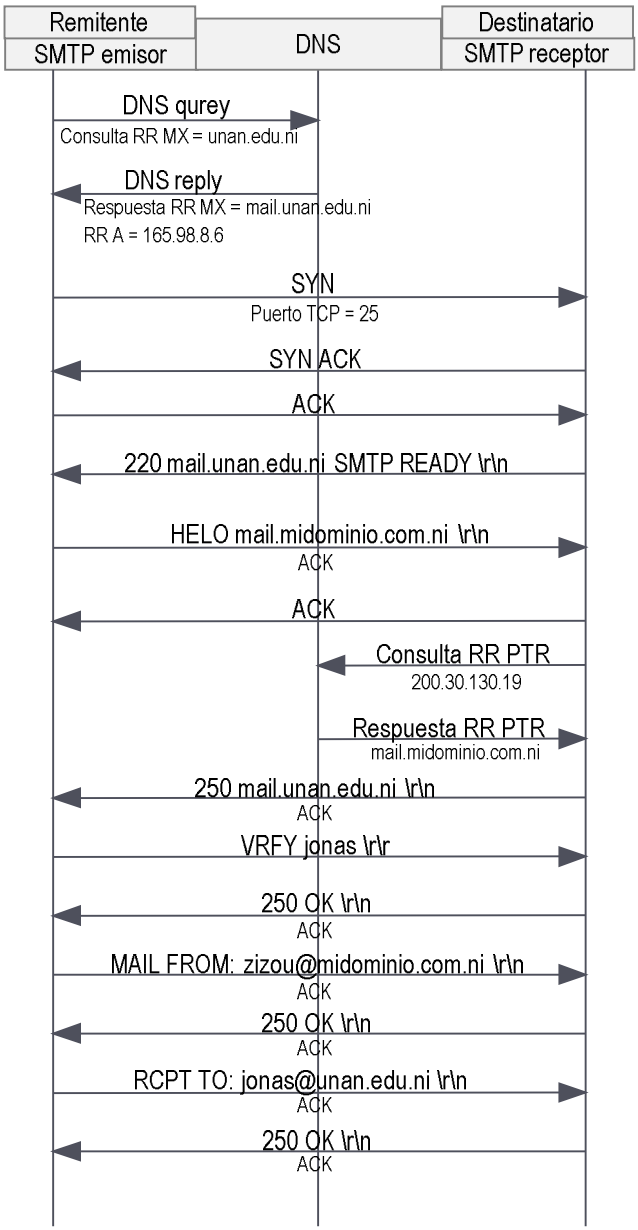
```
HELO <dominio> <CRLF>
```

Para cerrar la conexión se utiliza el comando QUIT. Este comando no tiene argumentos. Su formato es:

```
QUIT <CRLF>
```

3.2.6 Diagrama de secuencia SMTP

En el diagrama de secuencia para el protocolo SMTP interviene un tercer sistema, el DNS. Recuerde que previo a establecer la sesión SMTP, el servidor de correo origen tiene que consultar con el DNS cual es el servidor encargado de gestionar el correo para el dominio del destinatario de correo.



El servidor origen mail.midominio.com.ni lo primero que hace es consul registro MX para unan.edu.ni.
El servidor DNS encuentra que el servidor de correo con mayor preferencia el dominio unan.edu.ni es el host mail.unan.edu.ni. El registro A que indica de mail.unan.edu.ni se incluye en la respuesta.

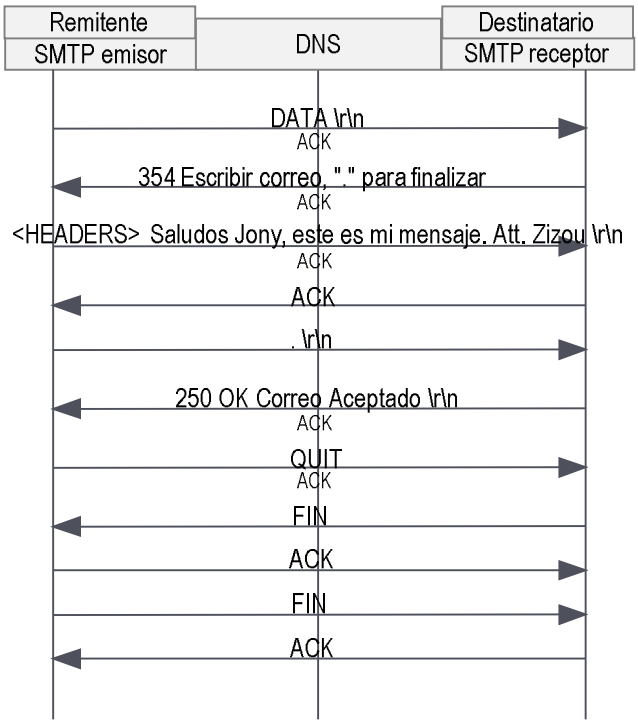
mail.midominio.com.ni inicia el saludo de tres vías para establecer la cor TCP enviando un SYN a la dirección IP de mail.unan.edu.ni. La solicit enviada al puerto 25 TCP.
El saludo de tres vías ha sido completado. La conexión TCP se ha estab con mail.unan.edu.ni.

El receptor SMTP responde con una cadena iniciada con el código 220. indica que el receptor está preparado para la transacción con el emis correo.
El servidor origen se identifica el mismo con el comando HELO. Este con contiene el nombre del servidor.

El receptor chequea el nombre de dominio del servidor origen con una co al servicio de nombres usando la dirección IP.
El servicio DNS responde facilitando el nombre del servidor de correo.

El comando HELO es respondido utilizando la información devuelta consulta DNS anterior. El servidor mail.unan.edu.ni acepta el comando respondiendo con el código 250.
El emisor solicita la verificación del usuario jonas.
El usuario es verificado como valido.

El SMTP emisor envía el comando MAIL FROM indicando el usuario rem del correo.
El receptor acepta el usuario remitente pasado como argumento al con MAIL FROM.
El servidor origen envía el comando RCPT TO con el usuario jonas argumento.
El receptor acepta, y está a la espera que se envíen los datos del mensaje el usuario jonas.



El emisor avisa que se iniciará el envío de los datos con el comando DATA.

El receptor está disponible para el inicio del correo.
Se envían las cabeceras y el cuerpo del mensaje escrito por el u
remitente.

Se indica el fin de los datos, una línea con el carácter ".".

El receptor indica que el correo ha sido aceptado.

Señal de fin de sesión SMTP.

El receptor SMTP libera la conexión TCP.

El emisor SMTP libera la conexión TCP.

3.3 POP3 (Post Office Protocol Version 3)

POP3[13] (Protocolo de Oficina de Correos) versión 3 es utilizado para transferir correo desde un servidor hasta una estación de trabajo. Forma parte junto a IMAP (Protocolo de Acceso a Mensajes de Internet) del conjunto de protocolos denominados protocolos de acceso al buzón.

El propósito original de protocolo es muy simple, no proporciona amplias operaciones de operación en el servidor; con POP3 el MUA transfiere el correo desde el servidor y después se borra.

[13] POP3 está especificado en el RFC 1939 (www.ietf.org/rfc/rfc949.txt).

El servicio POP3 se encuentra disponible de forma estándar en el puerto TCP 110. Cuando un cliente desea utilizar dicho servicio, establece una conexión TCP con el servidor. Una vez que la conexión ha sido realizada, tanto el servidor como el cliente comienzan un dialogo alternado que se explicará a continuación.

3.3.1 Funcionamiento de POP3

Inicialmente, el sistema configurado como servidor inicia el servicio POP3 escuchando en el puerto TCP 110. Cuando un cliente desea hacer uso del servicio, establece una conexión TCP con el servidor, una vez establecida la conexión, el servidor POP3 envía un saludo. Desde este momento el cliente y el servidor POP3 ya pueden intercambiar comandos y respuestas (respectivamente) hasta que se cierre o interrumpa la conexión.

Un servidor POP3 puede tener un periodo de inactividad de hasta 10 minutos, momento en el cual se cierra la conexión. Los comandos y respuestas se componen de caracteres ASCII imprimibles. Los comandos tienen tres o cuatro caracteres de longitud y consisten de una serie de palabras claves sin diferenciar mayúsculas de minúsculas, posiblemente seguidas de uno o más argumentos, donde cada argumento puede tener hasta 40 caracteres de longitud. Todos los comandos terminan con un par de <CRLF> (retorno de carro y nueva línea).

Las respuestas en el POP3 están formadas por un indicador de estado y una palabra clave, a veces seguida de información adicional, las respuestas están terminadas por un <CRLF>. Estas respuestas pueden ser afirmativas (+OK) o negativas (-ERR) y pueden ocupar una o más líneas.

Una sesión POP3 pasa por varios estados durante su periodo de vida. Una vez que el cliente inicia una conexión TCP al puerto 110, el servidor envía un saludo de bienvenida. En este momento, la sesión entra en un estado denominado AUTHORIZATION STATE (estado de autorización). Dentro de este estado, el cliente debe identificarse ante el

servidor POP3 enviando su nombre de usuario así como la contraseña. El servidor se encarga de validar dicha información y aceptará o rechazará la conexión del cliente dependiendo de si los datos son correctos o si por el contrario son erróneos. Si los datos han sido validados satisfactoriamente, la sesión entra en un estado denominado TRANSACTION STATE (estado de transacción), en el cual el servidor prepara toda la información relativa al cliente, para que éste pueda manipular su buzón de correo mediante una serie de comandos específicos. Tras la ejecución del comando QUIT, la sesión entra en un estado denominado UPDATE STATE (estado de actualización), en el cual se cierra la sesión de manera ordenada y por lo tanto se cierra también la conexión TCP.

3.3.2 Estados del protocolo POP3

Como se indicó en la sección anterior una sesión POP3 pasa por tres fases o estados:

3.3.2.1 Estado de Autorización

Una vez que el cliente ha iniciado una conexión TCP al puerto 110, el servidor POP3 le responde con un mensaje de bienvenida. Dicho mensaje depende de cada servidor, ya que puede ser cualquier texto finalizado con `<CRLF>`, el servidor POP3 siempre debe dar una respuesta afirmativa como saludo inicial:

```
+OK POP3 server ready
```

En este momento se puede decir que la sesión ha entrado en el estado de *autorización*. El cliente debe identificarse ante el servidor enviando en primer lugar su nombre de usuario y a continuación la contraseña.

El nombre de usuario va precedido del comando USER y la contraseña va precedida del comando PASS[14].

```
USER jonas <CRLF>
+OK
PASS jonas0000 <CRLF>
```

Cuando el cliente envía el comando USER seguido del nombre del usuario, el servidor responderá de una forma u otra dependiendo de la validez de dicho nombre. En el caso de que el nombre de usuario sea correcto, el servidor enviará la cadena +OK, y el cliente procederá al envío de la contraseña (comando PASS) o bien puede cerrar la conexión mediante el comando QUIT. Si el nombre del usuario es incorrecto, el servidor enviará la cadena –ERR y el cliente tendrá que volver a enviar el comando USER o bien finalizar la conexión.

En la *tabla 3.6* se muestran las posibles respuestas del servidor POP3.

Comando	Respuesta
USER	+OK nombre_usuario is a valid mailbox -ERR never heard of mailbox nombre_usuario -ERR no mailbox for nombre_usuario here
PASS	+OK -maildrop locked and ready -ERR invalid password -ERR unable to lock maildrop

Tabla 3.6 *Respuestas POP3 en la fase de autorización*

[14] Igual a cualquier otro protocolo TCP/IP, en POP3 la información enviada por la red viaja en texto plano (texto ASCII), quedando expuesta a cualquier intruso que desee hacerse con ella. Aunque el protocolo en si incorpora el comando opcional APOP para identificar a un usuario, en el capítulo 6 veremos cómo tratar estos problemas de seguridad a través de la extensión del protocolo POPS (POP3 seguro).

3.3.2.2 Estado de transacción

Una vez que el cliente se ha identificado a sí mismo con éxito ante el servidor POP3, y éste abre el buzón adecuado, la sesión se encuentra en la fase de *transacción*. El cliente ahora puede enviar repetidamente cualquiera de los comandos POP3 detallados en la *tabla 3.7*. Tras recibir cada comando, el servidor POP3 envía una respuesta. Eventualmente, el cliente puede enviar el comando QUIT y la sesión POP3 entrará en la fase de *actualización.*

Comando	Descripción
STAT	Solicita el estado del buzón. El servidor responde afirmativamente y devuelve el número de mensajes que hay en el buzón y el tamaño que ocupan (en bytes).
LIST	Se utiliza para solicitar la estadística del buzón. Se puede invocar pasando como argumento el número del mensaje para el cual se desea ver la estadística. El servidor devuelve una respuesta afirmativa con el número de cada mensaje y el tamaño de cada uno. En el caso de indicar un número de mensaje erróneo, el servidor devuelve una respuesta negativa.
RETR	Permite recuperar un mensaje del buzón siempre que éste no esté marcado para borrar. El comando se invoca pasando como argumento un número de mensaje. La respuesta del servidor será afirmativa si el número del mensaje es válido. El servidor enviará el mensaje solicitado. Si el número del mensaje es inválido, el servidor enviará una respuesta negativa.
DELE	Marca un mensaje para eliminar. La eliminación se produce cuando se entra en la fase de actualización.
NOOP	El servidor POP3 no hace nada ante este comando, simplemente responde afirmativamente. Permite mantener activa la conexión POP establecida mediante TCP cuando no hay actividad por parte del cliente.
RSET	Elimina las marcas de los mensajes marcados para borrar mediante el comando DELE.
TOP	Permite recuperar cierto número de líneas de un mensaje que se le indica como argumento.
UIDL	Solicita el identificativo único de los mensajes almacenados. Este identificativo es una cadena (de hasta 70 caracteres) que identifica cada mensaje del servidor.

Tabla 3.7 *Comandos POP3 en la fase de transacción*

3.3.2.3 Estado de actualización

La llegada a este estado se produce tras la ejecución del comando QUIT. En esta fase, el buzón de correo del usuario se desbloquea y se eliminan todos aquellos mensajes que fueron marcados para borrar. Las respuestas posibles que devuelve el servidor son:

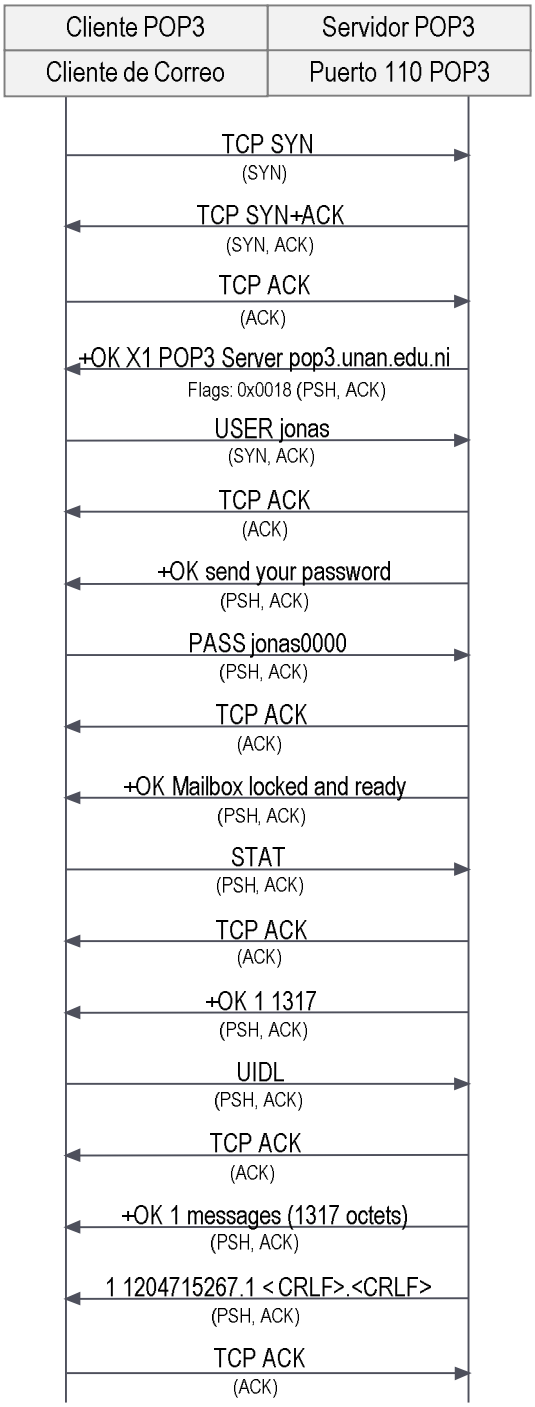
```
+OK nombre_servidor POP3 server signing off
-ERR some deleted messages not removed
```

3.3.3 Diagrama de secuencia POP3

Abajo se muestra el ejemplo de una sesión POP3, todos los comandos intercambiados entre el cliente de correo y el servidor POP3 son para recuperar un mensaje de correo del buzón de usuario, el siguiente flujo describe la comunicación entre un cliente y un servidor POP3.

```
Servidor: +OK X1 POP3 Server pop3.unan.edu.ni
Cliente: USER jonas@unan.edu.ni
Servidor: +OK Name is a valid mailbox
Cliente: PASS jonas0000
Servidor: +OK Mailbox locked and ready
Cliente: STAT
Servidor: +OK 1 1317
Cliente: UIDL
Servidor: +OK 1 messages (1317 octets)
1 1204715267.1
.
Cliente: LIST
Servidor: +OK 1 messages (1317 octets)
1 1317
.
Cliente: RETR 1
Servidor: +OK 1317 octets
Servidor: Mensaje de correo
 .
Cliente: DELE 1
Servidor: +OK message deleted
Cliente: QUIT
Servidor: +OK Connection closed by foreign host.
```

El correspondiente diagrama de secuencia a la sesión anterior es el siguiente:



El cliente POP3 ejecuta el saludo de tres vías para establecer la conexión TCP.

El servidor POP3 acepta el establecimiento de la conexión POP3 enviando la respu
+OK al cliente.

El cliente POP3 envía el nombre de usuario de correo como argumento del coma
USER.

El servidor POP3 acepta el usuario y solicita su clave.

El cliente POP3 envía la clave del usuario al servidor POP3. La clave es enviada en t
plano.

El servidor POP3 indica que el buzón de usuario está preparado.

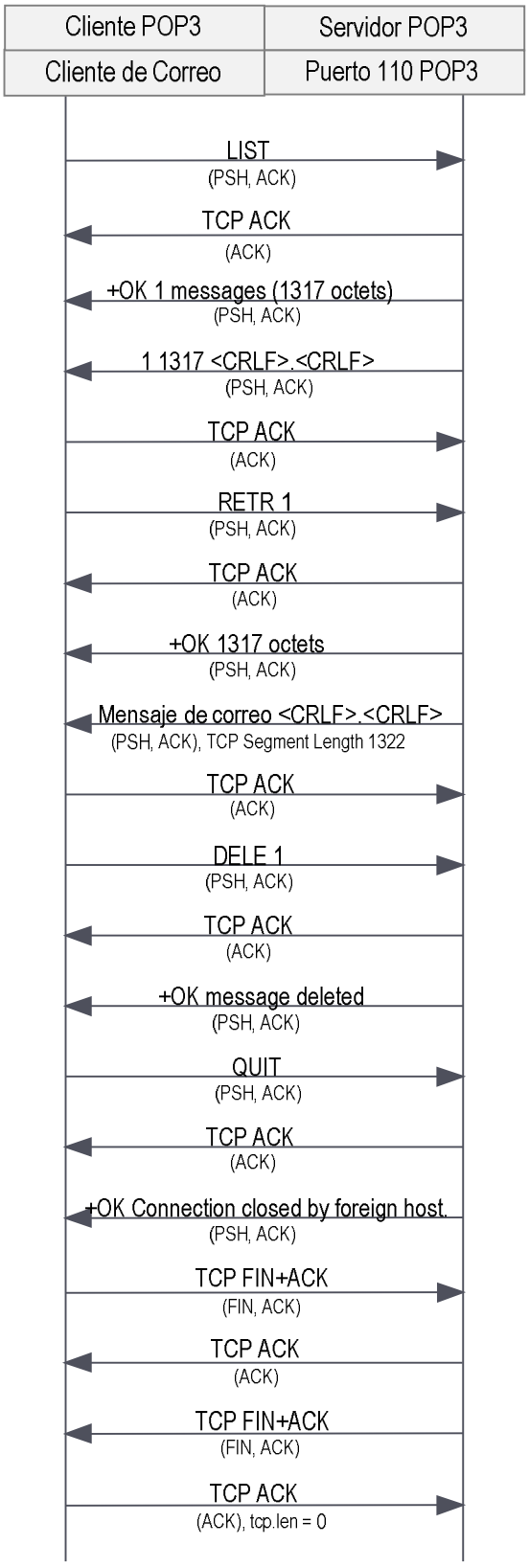
El cliente emite el comando STAT para solicitar información de los correos actuales e
buzón de usuario.

El servidor POP3 indica que existe un correo identificado por el numero 1 y qu
tamaño es de 1317 bytes.

El cliente POP3 solicita un Identificador de lista único (UIDL) para todos los cor
almacenados en el buzón de correo.

El servidor POP3 indica que tiene un correo electrónico con 1317 bytes de longitud.

El ID único asociado con el mensajes es enviado al cliente. Una línea vacía con un p
indica el fin de la lista.



El cliente POP3 emite el comando LIST para ver todos los mensajes.

El servidor POP3 responde al comando LIST con el número total de mensajes tamaño ocupado en el buzón de correo.

El servidor POP3 envía una lista que contiene el número y el tamaño del único mer existente en el buzón. El punto significa el fin del listado.

El cliente envía una solicitud al servidor para recuperar el mensaje 1. Esta es solicitud para obtener el mensaje 1 del buzón de correo.

El servidor POP3 acepta la solicitud e indica que a continuación se enviará un mer de 1317 bytes.

Los 1317 bytes del mensaje de correo son transportados en un segmento TCF longitud real del segmento es de 1322 bytes ya que incluye la secue <CRLF>.<CRLF> al final. Nuevamente el punto especifica el fin del mensaje.

En estos momentos el correo electrónico ha sido almacenado en el cliente de co por lo tanto el cliente solicita eliminar el mensaje 1 del buzón de usuario.

El servidor POP3 acepta la solicitud enviada por el cliente de eliminar el correo y no que el mensaje ha sido eliminado.

El cliente POP3 envía el comando QUIT para iniciar el proceso de liberar la sesión.

El servidor acepta el comando QUIT.

El cliente POP3 envía FIN para solicitar del lado del cliente que se libere la cone TCP.

El servidor acepta el FIN.

El servidor POP3 envía FIN para solicitar del lado del servidor que se libere la cone TCP.

El cliente acepta el FIN.

4. PREPARANDO LA INFRAESTRUCTURA

En este capítulo se construirá la plataforma que albergará la solución de correo electrónico, esto incluye el diseño del servicio así como el software y hardware utilizado. Se tratarán de determinar las necesidades del hardware de acuerdo a ciertos requerimientos, tales como sistema operativo base, cantidad de usuarios, capacidad de buzones, demanda de recursos por parte de servicios (demonios), transferencia de datos, etc. Con poca experiencia esto podría ser un proceso difícil, sin embargo empresas como Dell o IBM ofrecen un buen servicio gratuito de asesoramiento para seleccionar la mejor solución.

En cuanto al sistema operativo, este documento se centra en la distribución OpenSuSE Linux 10.3, recomendada para usuarios con poca experiencia en administración de sistemas sin embargo esto no le quita que sea una de las distribuciones Linux más potentes, utilizada en los entornos más exigentes. Recuerde que lo primero a considerar antes de instalar cualquier sistema operativo o software es el hardware del servidor.

Cabe mencionar que en el desarrollo de las instalaciones y configuraciones de los servicios descritos a partir de este capítulo, hacen referencia de manera singular a un solo servidor, sin embargo debe estar claro que el sistema de alta disponibilidad está formado por dos nodos, siendo el servidor MTA2 un replica del servidor MTA1.

4.1 Diseño del servicio

El servicio de correo electrónico debe adecuarse a la infraestructura de red con que se cuente, su diseño está estrechamente relacionado con la topología física de la organización y por lo tanto debe intentar acoplarse a ella. La estructuración idónea del servicio está basada en un encaminamiento de correo centralizado. Básicamente, consiste en dos servidores locales ubicados en una subred de alta disponibilidad y centralizados

en un proxy SMTP[15] ubicado entre la red local e Internet, el cual será el responsable de encaminar los correos desde el exterior. ¿Entonces como serán tratados los correos que vayan de la red interna hacia Internet? La manera más idónea de tratar el correo dirigido fuera de la organización es delegar directamente a cada servidor interno su tratamiento, es decir, cualquiera de los dos servidores SMTP locales se encargarán de la entrega directa del correo, sin embargo el correo proveniente de Internet será entregado al proxy SMTP, este lo almacenará temporalmente y dependiendo de la disponibilidad que tenga uno u otro servidor, será entregado. La *figura 4.1* aclara esta situación:

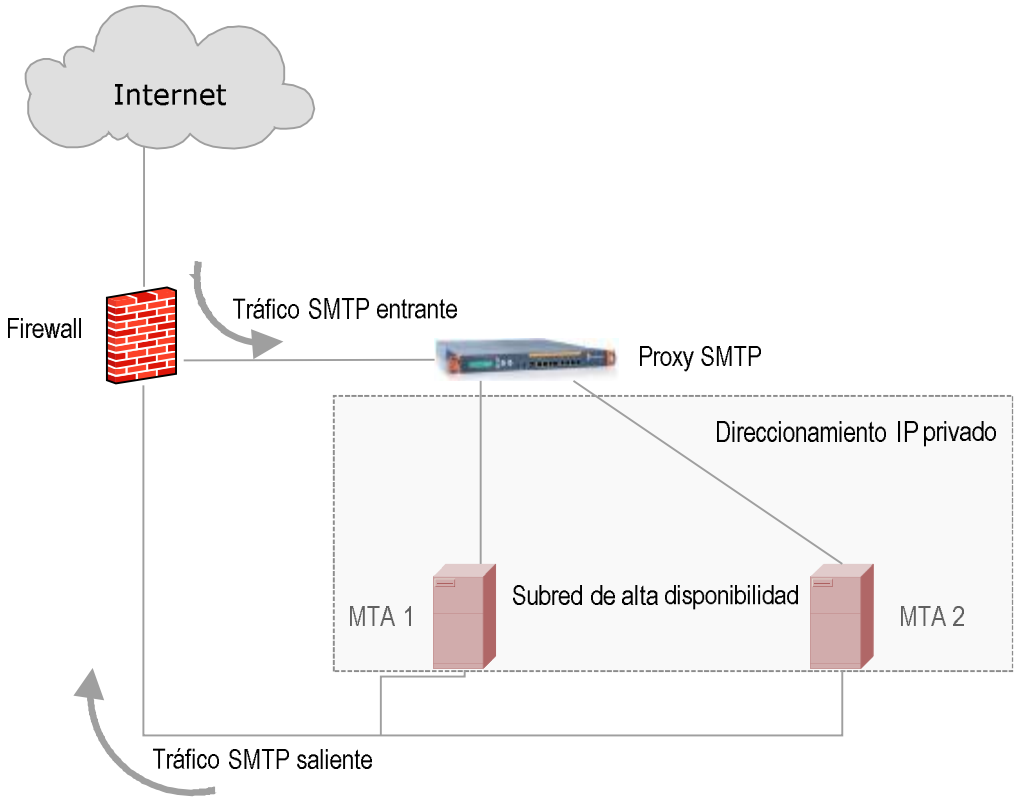


Figura 4.1 *Arquitectura de alta disponibilidad*

[15] La solución proxy SMTP utilizada en este proyecto es de licencia por pago, Astaro Security Gateway, básicamente este es un Linux empotrado en un appliance configurado eficientemente para filtrado Web, filtrado de paquetes y seguridad de correo; Sin embargo aquí será utilizado exclusivamente para servir de backup y reenvío SMTP.

La topología propuesta utiliza un encaminamiento SMTP indirecto para el correo entrante (desde Internet) y a la vez un encaminamiento directo para el correo saliente (hacia Internet), el porqué se ha definido esta estructura tiene sus razones:

- Utilizar el proxy SMTP nos dice que este es un punto crítico, sin embargo este es un equipo tolerante a fallos, con fuentes y puertos Ethernet redundantes, en el capítulo 6 vera lo que ello implica.
- Los servidores SMTP se pueden encargar directamente de la entrega del correo remoto, evitando así sobrecargar el proxy SMTP.

En cuanto al funcionamiento de este diseño podemos afirmar que:

- El proxy SMTP tendrá la función exclusiva de tomar y reenviar hacia cualquiera de los MTA todo el correo proveniente desde Internet.
- Los servidores MTA 1 y MTA 2, no solamente alojan el servicio SMTP, sino también los servicios POP3, IMAP, autenticación de usuarios, webmail, en si todos los servicios descritos en las configuraciones propuestas en el capítulo 5.
- El correo local es gestionado por los servidores MTA 1 o MTA 2, el proxy SMTP no interviene en su distribución interna.
- El sistema de nombres de dominio debe tener configurado el registro MX para entregar el correo al proxy SMTP.
- El correo local no debe hacer uso del registro MX.
- Para los usuarios (MUA) el proxy SMTP es totalmente transparente, para ellos solo existe un solo servidor de correo.
- El cortafuego filtra el puerto 25 para el tráfico SMTP, solo el equipo identificado como proxy SMTP en el diagrama, es accesible por este puerto desde Internet.

Es de mucha importancia tener bien definido un plan de direccionamiento para los servidores, se recomienda definir las direcciones IP a utilizar antes de dar inicio a las configuraciones de los sistemas. El direccionamiento propuesto para cada uno de los servidores implicados en el sistema de correo se detalla en la *tabla 4.1*.

Equipo	Hostname	IP publica	IP privada	IP HA (Red Privada)
MTA 1	mail-ha1	165.98.8.6	10.1.120.3	192.168.1.1
MTA 2	mail-ha2	165.98.8.7	10.1.120.3	192.168.1.2
Proxy SMTP	correo	165.98.8.8	10.1.120.6	
Firewall		165.98.8.1		

Tabla 4.1 *Direccionamiento de servidores de correo electrónico*

4.2 Requerimientos de Hardware

Elegir el hardware más adecuado de acuerdo a las necesidades que se vayan a tener puede ser una de las consideraciones más importantes, sin embargo esto puede ser también una de las tareas más difíciles durante todo el proyecto. Las características del hardware tendrán una implicación directa en el rendimiento del sistema, tomando en cuenta situaciones como, picos de carga, ataques (por ejemplo un DoS), recuperación ante fallos, etc. Lamentablemente no existe una fórmula que determine exactamente que procesador, cuanta memoria física y cuanto espacio en disco duro se necesita para levantar un servidor que se capaz de procesar eficientemente miles de correos por hora. Tal vez el punto más importante a tener en cuenta será siempre planificar y sobrepasar lo estimado; una vez que el sistema se pone en producción, lo más seguro es que la demanda solo tienda al incremento a través del tiempo. Afortunadamente el costo del hardware se ha reducido considerablemente en los últimos años, sistemas operativos como Linux pueden correr muy bien en sistemas de bajo costo, conocidos como clones.

Para propósitos de este documento, se asumirá que el sistema final estará en dos equipos con características similares. Servicios complejos redundantes como equilibrio de carga y alta disponibilidad serán tratados en los capítulos 5 y 6. El hardware del servidor que se elija debe reflejar coherentemente el tamaño de la organización y la cantidad de

usuarios del sistema. Como se dijo antes diversos fabricantes (DELL o IBM) brindan buen asesoramiento para ayudar a elegir el hardware adecuado.

4.2.1 Procesador

El poder de procesamiento necesario es bastante difícil de calcular. Por simple intuición se puede estar seguro que contar con servidores que tengan más de un procesador nos asegurará un aumento en la eficiencia de la solución de correo. Tener más de un procesador significa la ejecución de más ciclos de cómputo que puedan ser completados simultáneamente para mejorar el rendimiento. Se asume que la plataforma de los servidores está basada en arquitectura x86.

4.2.2 Memoria

Muchos servicios de la solución de correo construida en este documento pueden tomar ventaja de la RAM que aquí se recomienda. Por ejemplo, el servicio de antivirus del sistema será considerablemente más rápido si las definiciones de virus son cargadas en memoria a la velocidad y eficiencia de la RAM física disponible. Si el sistema no cuenta con la memoria RAM suficiente para hacer esto, entonces el sistema operativo tendrá que estar constantemente administrando la carga de las definiciones de virus en memoria swap, que es lo mismo que la lectura de disco duro; se recomienda siempre reducir al mínimo las escrituras y lecturas en disco duro (para efectos de manipulación de swap). Se propone un mínimo de memoria RAM de 2 GB.

4.2.3 Disco duro

Las capacidades de espacio en disco duro son muy dependientes de las necesidades que se tengan. Solo para el sistema operativo se recomienda reservar al menos 5 GB (sin tomar en cuenta la partición que almacenará los buzones). Más adelante en la instalación del SO se hará un desglose detallado del tamaño de cada una de las particiones del disco. El espacio en disco será extremadamente importante, ya que muchas veces los usuarios suelen valorar la calidad del servicio de correo basándose en la capacidad (en

megabytes) del buzón ofrecido. Para determinar la quota del buzón de correo deben responderse las siguientes interrogantes: ¿Cuanto espacio se dispone en disco duro?, ¿Cuál es la cantidad máxima de usuarios admitidos?, ¿Cuales son las necesidades de los usuarios?, ¿A qué usuarios se les permitirá mantener copia de sus correos en el servidor? y algunas más que aparecerán en el camino.

La experiencia nos dice que no es común que los usuarios borren sus correos una vez los hayan leído, más aún cuando la administración del buzón está en manos de ellos; con el tiempo es probable que aunque la cantidad de buzones permanezca inalterada, el espacio total en disco del servidor irá disminuyendo, esto por las razones ya comentadas.

En el caso particular del servicio de correo de la UNAN-Managua, se tienen aproximadamente 1200 cuentas de correo. La política oficial es brindar una quota de 200MB por usuario, esto sin tomar en cuenta usuarios con mas privilegios que pueden llegar a requerir hasta 1GB. En base a estos números, se puede asumir que se necesita aproximadamente 240GB de espacio total en disco duro, solo para quotas de usuario. Además se necesitará espacio extra para el manejo de las colas de correo, cuarentenas, almacenamiento temporal (archivos adjuntos). La cantidad exacta depende de la política y proyección de crecimiento.

Si el presupuesto lo permite, también debe ser considera la redundancia de discos duros. Un Arreglo Redundante de Discos Independientes (RAID) requiere más de un disco físico, sin embargo proporciona integridad y redundancia de datos. La elección del tipo de redundancia, dependerá de la aplicación especifica, en el caso particular de soluciones de correo electrónico se recomienda una combinación RAID0 (DATA stripping) y RAID1 (DATA mirroring), lo que equivale a la elección de un RAID0+1 (espejo sobre división de discos). También se debe tomar en cuenta la solución RAID a implementar (hardware o software), si la elección es un RAID por hardware, implica mayor inversión

económica, compensada en el mejor rendimiento ofrecido que una solución RAID por software.

El RAID1 es recomendado para sistemas de alta disponibilidad, por ejemplo, para hacer una copia de seguridad de un servicio siempre disponible, se marcaria uno de los discos como inactivo, se hace la copia de dicho disco, mientras tanto el primer disco del arreglo seguiría en funcionamiento. Al terminar de hacer el respaldo, el espejo nuevamente es reconstruido.

La *tabla 4.2* describe brevemente algunos de los niveles RAID más utilizados.

Nivel Raid	Características	Minimo No de HD	Ventajas	Desventajas
0	Conjunto dividido de discos, los datos se dividen en bloques y los bloques se escriben en diferentes discos.	2	Baja sobrecarga Incrementa el rendimiento	Realmente no equivale a un RAID; no brinda redundancia; la pérdida de un disco resulta en la perdida de todos los datos.
1	Espejo y dualidad	2	Redundancia completa de datos; la tasa de transferencia es igual que si se tuviera un solo disco.	Menos eficiente y alta sobrecarga de discos.
5	División de datos a nivel de bloques y distribución de la información de paridad entre todos los discos.	3	Altos porcentajes de lectura; bajo coste de redundancia	Un fallo de discos tiene un impacto considerable en el rendimiento y dificulta la reconstrucción del arreglo.
0+1	Replicación de datos entre varios discos; varios niveles	4	Cuando un disco duro falla los datos perdidos	No es tan robusto, no pudiendo tolerar dos

	raid, esto es un espejo (RAID1) sobre divisiones (RAID0).		pueden ser copiados del otro conjunto del nivel 0 para reconstruir el conjunto global.	fallos simultáneos de disco salvo que sean en la misma división.

Tabla 4.2 *Niveles RAID más utilizados*

Cuando se utiliza un sistema RAID, si un disco duro falla, el sistema seguirá funcionando sin pérdida de datos. La recuperación ante un fallo puede ser más fácil y así reducir el tiempo de inactividad del servicio, sin embargo aquí se propondrán dos sistemas idénticos que brindarán doble redundancia, la probabilidad de que un disco duro falle en el segundo sistema se reduce, de esta manera la disponibilidad del servicio será mucho más alta.

La mayoría de las distribuciones Linux soportan ya sea RAID por emulación con software o RAID con controladoras hardware. Una solución RAID en hardware suele ser más cara, sin embargo es transparente al sistema operativo, mientras que un RAID por software requiere que el sistema operativo emule características de un controlador hardware, lo que implica gastos adicionales en procesamiento.

En este documento se configurará RAID1 por software, haciendo uso de la solución facilitada en el menú de particionamiento de SuSE Linux 10.3, en la *sección 4.2* verá cómo hacerlo.

La *tabla* 4.3 resume las características del hardware utilizado para los servidores, notar que en ambos servidores se instalará OpenSuSE Linux 10.3 con servicios de correo electrónico totalmente replicados.

Equipo	Procesador	Memoria	1er Disco Duro	1do Disco Duro
MTA 1	Intel Xeon PIV 3GHz	2GB	160GB	160GB
MTA 2	Intel Xeon PIV 3GHz	2GB	160GB	160GB

Tabla 4.3 *Características hardware de servidores de correo*

4.3 Introducción a OpenSuSE Linux

El proyecto descrito en este documento usa OpenSuSE Linux 10.3[16], sistema operativo desarrollado por Novel como un proyecto abierto a la comunidad de desarrolladores en Linux (opensuse.org). SuSE Linux se utiliza como base exclusivamente para las instrucciones y ejemplos aquí descritos, destacando que cada uno de los puntos tratados pueden ser implementados en un sistema real. Sin embargo se recomienda al lector utilizar estrictamente SuSE Linux si desea probar cada una de las configuraciones, y ponerlas en práctica en un entorno de producción; en la mayoría de las distribuciones se encontraran ligeras variaciones.

La elección del sistema operativo es cuestión de gustos personales, todo depende de la experiencia que se tenga con una u otra distribución, aunque en este trabajo se ha utilizado SuSE por ser una de las distribuciones que cuenta con uno de los mejores soporte de software (repositorios con actualizaciones periódicas) en la actualidad, talvés por la política de mercado adoptada por Novel (fabricante del SO), por una parte ofrecen el sistema operativo "SuSE Enterprise Linux" por pago de licencia (mas que todo por soporte) y por otra nos encontramos con el sistema libre y abierto Open SuSE Linux. Algunas de las ventajas ofrecidas son:

- Respaldado por Novel, una de las empresas con más experiencia en Sistemas Operativos para redes.

[16] La versión actual liberada por Novell es OpenSuSE 11.0 beta, sin embargo aquí se utilizará OpenSuSE 10.3 por ser la última versión estable.

- Fácil interfaz de instalación.
- Alto rendimiento, ofrecido para los entornos más exigentes.
- Excelente selección de paquetes pre compilados.
- Excelente administración grafica para usuarios noveles, a través de YAST.

4.3.1 Obteniendo el sistema operativo

La última versión liberada de SuSE puede ser obtenida de muchas maneras. También se cuenta con la opción de obtener el sistema operativo completo en dos formatos, en una imagen para DVD o en un conjunto de cuatro imágenes para CD-ROM, tanto las versiones en DVD como las de CD-ROM son de arranque automático.

Las imágenes en formato ISO están disponibles en muchos sitios espejos a través de http y ftp (http://en.opensuse.org/Mirrors_Released_Version). Otra manera no menos popular de obtener el sistema operativo es a través de BitTorrent (protocolo de intercambio de archivos), el mismo sitio de SuSE nos brinda esta opción como alternativa a los protocolos http y ftp.
Elegir la versión para DVD o CD-ROM depende de las características del hardware con que se cuente, se recomienda utilizar un solo DVD.

4.3.2 Verificando la integridad de la descarga

Se dará por entendido que la imagen ISO descargada es en formato DVD; una vez obtenido dicho archivo se recomienda verificar su integridad. Es común que la distribución proporcione una suma de comprobación md5sum (archivo que contiene la suma md5 original), que puede ser utilizada para compararlo con la suma obtenida del archivo descargado. Si las dos sumas md5 coinciden, entonces es razonable suponer que el archivo fue descargado correctamente. La forma de obtener la suma es a través del comando md5sum:

```
edison@mail:~> md5sum openSUSE-10.3-GM-DVD-i386
```

4.3.3 Instalación del Sistema Operativo

La objetivo no es mostrar una instalación de Linux, sino mas bien dar un ejemplo de cómo realizar una instalación del sistema reduciendo al mínimo la instalación de software y servicios innecesarios, es decir se instalará un sistema para albergar solamente servicios relacionados con el correo electrónico. Adema de identificar los requerimientos mas optimos para un servior de esta clase.

Los requerimientos mínimos de hardware para una instalación de Open SuSE Linux 10.3 se resumen en la *tabla 4.4*.

Procesador	RAM (mínimo)	RAM (recom.)	Espacio en disco (mínimo)	Espacio en disco (recom.)
Intel Pentium 1-4, Celeron, AMD Duron, Athlon, Athlon 64, Semprom u Opteron, PowerPC de IBM.	256 MB	512MB	500MB	3GB

Tabla 4.4 *Requerimientos mínimos de hardware de OpenSuSE 10.3*

En la instalación se pueden identificar dos puntos claves, el particionamiento y la creación del RAID1; tenga en cuenta que una instalación Linux requiere por lo menos de tres particiones:

- Partición de arranque, donde se guarda la imagen del sistema operativo, también conocida como boot.
- Partición de intercambio de memoria o SWAP.
- Y la raíz del sistema de archivos o el root.

Recuerde que la instalación se realizará de forma idéntica en cada uno de los sistemas descritos en la *tabla 4.3*.

La *tabla 4.5* propone una estructura más o menos razonable, de acuerdo a las capacidades de los discos duros utilizados en la implementación real de los sistemas:

Disco Duro	Tipo de partición	Punto de montaje	Formato	Tamaño
/dev/sda	Primaria	/boot	ext3	100MB
/dev/sda	Primaria	/tmp	Ext3	900MB
/dev/sda	Primaria		No formatear (0xFD Linux RAID)	159GB
/dev/sdb	Primaria	swap	swap	1000MB
/dev/sdb	Primaria		No formatear (0xFD Linux RAID)	159GB

Tabla 4.5 *Estructura de discos de servidores de correo*

4.4 Ajustes Post Instalación

Aunque la instalación realizada es para albergar exclusivamente todos aquellos servicios relacionados con correo electrónico, es necesario en este momento hacer algunos ajustes al sistema:

- Configuraciones de red
- Instalación y reinstalación de software
- Preparación de servicios de los que depende directamente el correo electrónico (DNS y Servicio Web)
- Configuraciones de seguridad.

4.4.1 Login al sistema y configuraciones iniciales

Una vez que se complete el proceso de arranque del servidor Linux, se le presentará la pantalla de login, la apariencia de dicha pantalla dependerá del escritorio gráfico seleccionado en el momento de la instalación del sistema:

Figura 4.2 *Login de acceso al sistema*

En una instalación típica de Linux, lo más probable es que solo pueda loguearse con el usuario root y podrá ingresar al sistema con el usuario sin privilegios creado previamente. Se recomienda siempre ingresar al sistema con un usuario sin privilegios, posteriormente si la tarea administrativa lo amerita, proceda a cambiarse al usuario root.

4.4.1.1 Configuraciones de Red

Aunque se pudieron haber configurado las interfaces de red desde el programa de instalación, hemos escogido hacerlo manualmente, editando directamente los archivos de configuración asociados a las dos interfaces reconocidas por el sistema, dichos archivos puede ubicarlos en el directorio `/etc/sysconfig/network/`. Por convención para cada servidor hemos identificado la interfaz eth0 como privada y la eth1 como pública.

El archivo de configuración para la interfaz eth0 (dirección IP privada) nombrado como ifcfg-eth0 del servidor mail-ha1 será:

```
BOOTPROTO='static'
BROADCAST='10.1.120.255'
IPADDR='10.1.120.3'
```

```
NETMASK='255.255.255.0'
NETWORK='10.1.120.0'
ETHTOOL_OPTIONS=''
MTU=''
NAME='AMD PCnet - Fast 79C971'
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
```

A continuación se explican brevemente las opciones más importantes:

`BOOTPROTO`: Protocolo de inicio, define el tipo de direccionamiento utilizado, estático o dinámico cuyos valores posibles son static y dhcp respectivamente. Existen otros valores menos comunes como autoip, dhcp+aoutoip y 6to4.

`BROADCAST`: Establece la dirección de difusión de la interfaz. Si se deja en blanco se utilizará el establecido por defecto en el archivo `/etc/sysconfig/network/config`.

`IPADDR`: Dirección IP, para múltiples direcciones utilice el nombre de la variable + numero de dirección IP, por ejemplo, `IPADDR_1, IPADDR_2,` etc.

`NETMASK`: Establece la máscara de red para la dirección IP, para múltiples direcciones use el mismo sufijo, como en la variable `IPADDR`.

`NETWORK`: Establece la dirección de red.

El archivo correspondiente a la interfaz eth1 (dirección IP pública) es similar al anterior, veamos su contenido:

```
BOOTPROTO='static'
BROADCAST='165.98.8.255'
IPADDR='165.98.8.6'
NETMASK='255.255.255.0'
NETWORK='165.98.8.0'
ETHTOOL_OPTIONS=''
MTU=''
NAME='AMD PCnet - Fast 79C972'
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
```

Posterior a la edición de los archivos, solo queda reiniciar el servicio network, encargado de la administración de las interfaces de red:

```
mail-ha1:/etc/sysconfig/network # rcnetwork restart
```

4.4.1.2 Estableciendo rutas por defecto

La tabla de enrutamiento es establecida automáticamente de acuerdo a las configuraciones detectadas en los archivos de configuración de cada interfaz de red, dichas rutas son conocidas como rutas directas. Además de estas rutas se necesita una ruta por defecto, que se encargue de encaminar hacia el exterior todo el tráfico saliente, para ello debe crear un archivo nombrado routes en el directorio `/etc/syconfig/network/` con el siguiente contenido:

```
#Destination     Dummy/Gateway     Netmask            Device
#
 default           165.98.8.1      0.0.0.0            eth1
```

El archivo anterior establece como ruta por defecto la dirección 165.98.8.1 disponible a través de la interfaz eth1.

Nota: Nuevamente deberá reiniciar el servicio `network` para cargar los cambios.

La tabla de enrutamiento para el sistema quedará de la siguiente forma (ver tabla de rutas con comando route):

```
mail-ha:/etc/sysconfig/network # route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.120.0      *               255.255.255.0   U     0      0        0 eth0
loopback        *               255.255.255.0   U     0      0        0 lo
165.98.8.0      *               255.255.255.0   U     0      0        0 eth1
link-local      *               255.255.0.0     U     0      0        0 eth0
loopback        *               255.0.0.0       U     0      0        0 eth0
default         165.98.8.1      0.0.0.0         UG    0      0        0 eth1
```

4.4.2 Configuración del DNS

El sistema de nombres de dominio es considerado como la raíz de Internet, prácticamente todos los servicios dependen de él, mas aún el servicio SMTP, recordemos que el protocolo SMTP solo define la manera de establecer una sesión para transferir un correo desde un sistema local hacia un sistema remoto, sin embargo este no dice nada o poco de cómo hace para identificar o localizar el sistema remoto. Es aquí donde entra en juego el DNS, previo a establecer una sesión SMTP el servidor de correo envía una consulta al DNS, preguntándole por el registro MX del dominio identificado en la cabecera RCPT TO del mensaje de correo. Suponiendo una respuesta afirmativa por parte del DNS, esta indicaría el host remoto encargado de procesar el correo del dominio (lo más conveniente es que dicha respuesta sea un RR de tipo A y no un alias CNAME), con esta información el servidor SMTP ya sabría a que dirección remoto deberá encaminar el correo.

Aunque lo más práctico es que el servicio DNS se encuentre físicamente en un equipo distinto al que aloja la solución de correo electrónico, en este documento se configurará en el mismo sistema.

Esta sección revisa el servidor de nombres incluido en SuSE Linux 10.3, el servidor DNS Berkeley Internet Name Domain (BIND), enfatizando una configuración para el servicio de correo electrónico.

4.4.2.1 Configuración del servidor de nombres BIND

BIND realiza la resolución de nombres a través del demonio `/usr/bin/named`. Este también incluye una utilidad de administración llamada `rndc`, sin embargo no será abordada porque está fuera de los propósitos de este documento.

Se pueden identificar dos tipos de archivos de configuración, almacenados en los siguientes directorios:

- `/etc/named.conf`: este es el archivo de configuración principal, en el se definen cada una de las zonas de dominio. Está formado por una colección de declaraciones usando opciones anidadas encerradas entre llaves**.** Debe tener mucho cuidado cuando este editando este archivo para evitar errores sintácticos, puesto que hasta el error más pequeño impedirá que el servicio `named` arranque.
- Directorio `/var/lib/named`: el directorio de trabajo de `named` el cual almacena las zonas, estadísticas y archivos caché.

Aunque el archivo de configuración principal `/etc/named.conf` contiene muchos más parámetros configurables, solo editaremos los correspondientes a la definición de las zonas de dominios, tanto directa como inversa, la mayoría de parámetros funcionan bien con sus valores por defecto. Se agregará la zona de dominio `“unan.edu.ni”`, además de su respectiva zona inversa, recuerde que este tipo de zona es opcional.

En organizaciones con presencia en Internet (con dominio público), lo común es que se definan dos zonas, una privada y una pública, el dominio interno es de carácter administrativo por lo tanto es opcional. Para nuestros propósitos nos interesa solamente la zona pública.

```
zone "unan.edu.ni" in {
        type master;
        file "unan.edu.ni.zone";
};

zone "8.98.165.in-addr.arpa" in {
        type master;
        file "165.98.8 .zone";
};
```

La declaración `zone` define las características de la zona `“unan.edu.ni”` como su tipo (master) y la ubicación del archivo de zona que contiene los datos de configuración como registros A, PTR, MX, etc.

El siguiente paso es configurar los correspondientes archivos de zona. Los archivos de zona contienen información sobre un espacio de nombres particular y son almacenados en el directorio de trabajo de named `/var/lib/named`. Cada archivo de zona es nombrado de acuerdo a la opción `file` en la declaración `zone`, usualmente es un nombre que relaciona el dominio en cuestión, tal como en nuestro ejemplo (unan.edu.ni.zone).

Cada archivo de zona contiene directivas y registros de recursos. Las directivas le dicen al servidor de nombres que realice tareas o aplique configuraciones especiales a la zona. Los RR definen los parámetros de la zona y asignan identidades a hosts individuales. Las directivas son opcionales, pero los RR se requieren obligatoriamente para proporcionar servicios de nombres a la zona.

El archivo de zona directo `unan.edu.ni.zone` es el siguiente:

```
1 $TTL 1W
2 @               IN SOA  @   root (
3                                  42             ; serial (d. adams)
4                                  2D             ; refresh
5                                  4H             ; retry
6                                  6W             ; expiry
7                                  1W )           ; minimum
8
9                 IN NS           mail-ha1
10                IN A            165.98.8.6
11 mail-ha1       IN A            165.98.8.6
12 mail-ha2       IN A            165.98.8.6
13 correo         IN A            165.98.8.8 //Este es el proxy SMTP
13 unan.edu.ni.   IN MX 05        correo
```

Opcionalmente hemos de crear el archivo de resolución inversa (utilizado para traducir una dirección IP en un espacio de nombres particular en un FQDN). Note que este archivo es muy similar al anterior:

```
$TTL 1W
@               IN SOA          unan.edu.ni.   root.unan.edu.ni. (
                                42              ; serial (d. adams)
                                2D              ; refresh
                                4H              ; retry
                                6W              ; expiry
                                1W )            ; minimum

                IN NS           mail-ha1.unan.edu.ni.
6               IN PTR          mail-ha1.unan.edu.ni.
6               IN PTR          mail-ha2.unan.edu.ni.
8               IN PTR          correo.unan.edu.ni.
```

En el archivos anteriores puede identificar la directiva `$TTL`. Se encarga de ajustar el tiempo de vida predeterminado para la zona, este es el tiempo en segundos, que un registro de recurso de zona es válido. Cada recurso puede tener su propio valor TTL. Cuando se decide aumentar este valor, permite a los servidores de nombres remotos hacer caché a la información de zona para un periodo más largo de tiempo, reduciendo el número de consultas para la zona y alargando la cantidad de tiempo requerido para proliferar cambios de registros de recursos

Se utilizan las siguientes directivas numéricas:

- `serial`: Es un valor numérico que se debe incrementar cada vez que se haga un cambio en el archivo de zona. Este número es utilizado por los servidores esclavos para determinar si está usando datos de la zona desactualizados y si debería refrescarlos.
- `refresh`: indica el tiempo que los servidores esclavos utilizan para determinar cuánto tiempo debe esperar antes de preguntar al servidor de nombres maestro si se han realizado cambios en la zona.
- `retry`: es un valor numérico usado por los servidores esclavos para determinar el intervalo de tiempo que tiene que esperar antes de emitir una petición de actualización de datos en caso de que el servidor de nombres maestro no responda.

- `expiry`: si el servidor maestro no ha respondido a una petición de actualización de datos antes que se acabe el intervalo de tiempo indicado por `expiry`, los servidores esclavos paran de responder como una autoridad a peticiones al espacio de nombres.
- `mínimum`: es la cantidad de tiempo que otros servidores de nombres guardan en cache la información de zona.

Aunque los tiempos anteriores son referenciados en segundos, en el ejemplo se han utilizado abreviaturas como (M) minutos, (D) días, (H) horas y (W) semanas.

Los registros de recursos identificados son:

- `A`: Especifica una dirección IP que se debe asignar a un nombre, en este caso son cuatro, uno para el dominio (implícito en el campo vacio de la línea 10) `unan.edu.ni` y los otros tres para los hosts `mail-ha1`, `mail-ha2` y `correo.unan.edu.ni`.
- `MX`: Este es el registro a destacar, el `MX` o Mail eXchange, indica donde debería ir el correo enviado a un espacio de nombres en particular, en nuestra configuración seria al dominio `unan.edu.ni`, el registro contiene una clasificación numérica (16 bits) que da preferencia a los servidores, cuanto más bajo sea este número mayor será la prioridad.
- `NS`: Registro NameServer, el cual anuncia los nombres de servidores con autoridad para una zona particular, en el ejemplo es `NS` es el mismo servidor de correo mail-ha.
- `PTR`: Registro PoinTeR o puntero, diseñado para apuntar a otra parte del espacio de nombres, es utilizado principalmente para la resolución inversa de nombres, pues apunta a una dirección IP.
- `SOA`: Registro de recurso Start Of Authority, que declara información importante de autoridad relacionada con el espacio de nombres.

El símbolo @ de la línea 2 identifica el nombre de la zona `unan.edu.ni.`

En este punto el servidor de nombres está configurado para resolver el dominio `unan.edu.ni,` para cargar los cambios ejecute el script de arranque del servicio:

```
mail-ha1:/var/lib/named # rcnamed restart
Shutting down name server BIND - Warning: named not running!         done
Starting name server BIND                                            done
mail-ha1:/var/lib/named #
```

BIND brinda tres herramientas para la prueba del servicio, dig, nslookup y host. A continuación vea ejemplos de cómo utilizarlas:

```
mail-ha1:~ # nslookup unan.edu.ni
Server:         192.168.203.129
Address:        192.168.203.129#53

Name:   unan.edu.ni
Address: 165.98.8.6
mail-ha1:~ # dig unan.edu.ni

mail-ha1:~ # dig unan.edu.ni
;; QUESTION SECTION:
;unan.edu.ni.                   IN      A

;; ANSWER SECTION:
unan.edu.ni.            604800  IN      A       165.98.8.6

;; AUTHORITY SECTION:
unan.edu.ni.            604800  IN      NS      mail-ha.unan.edu.ni.

;; ADDITIONAL SECTION:
mail-ha1.unan.edu.ni.    604800  IN      A       165.98.8.6

mail-ha1:~ # host -t MX unan.edu.ni
unan.edu.ni mail is handled by 5 correo.unan.edu.ni.
```

4.4.3 Servicios de autenticación de usuarios

Existen múltiples formas de autenticar usuarios en un sistema de correo, algunas de las más utilizadas son:

- Usuarios locales: Es la manera más común, se aprovecha la propiedad inherente de la mayor parte de las aplicaciones Unix de utilizar los usuarios del sistema, almacenados en los archivos `/etc/passwd` y `/etc/shadow`.
- Bases de datos: Es el mecanismo más utilizado en la actualidad, en este documento se utilizará una base de datos en MySQL para almacenar los usuarios del servicio de correo.
- Servicio de directorios: Es el mecanismo más indicado, un servicio de directorios está diseñado exclusivamente para almacenar información de usuarios (Nombres, ID, Teléfono, Dirección, etc.) similar a una guía telefónica,

4.4.3.1 Pluggable Authentication Modules (PAM)

Los programas que permiten a los usuarios accesar un sistema verifican la identidad de cada usuario a través de un proceso llamado autenticación. En Linux cada programa tiene su forma particular de realizar la autenticación. Bajo SuSE Linux, muchos de tales programas son configurados para usar un proceso de autenticación centralizado llamado Pluggable Authentication Modules (PAM).

PAM utiliza una arquitectura conectable y modular, que otorga al administrador del sistema de una gran flexibilidad en establecer las políticas de autenticación para el sistema.

En la mayoría de los casos, el archivo de configuración por defecto PAM para una aplicación tipo PAM es suficiente. Sin embargo, algunas veces es necesario modificar el archivo de configuración. Debido a que un error en la configuración de PAM puede comprometer la seguridad del sistema.

PAM ofrece las siguientes ventajas:

- Proporciona un esquema de autenticación común que se puede usar con una gran variedad de aplicaciones.

- Permite gran flexibilidad y control de la autenticación tanto para los administradores del sistema como para los desarrolladores de aplicaciones.
- Permite a los desarrolladores de aplicaciones desarrollar programas sin tener que crear sus propios esquemas de autenticación.

El directorio `/etc/pam.d/` contiene los archivos de configuración PAM, para cada aplicación tipo PAM debe haber su correspondiente archivo de configuración, por ejemplo el archivo PAM del servicio SMTP es `/etc/pam.d/smtp` y su contenido por defecto es el siguiente:

```
#%PAM-1.0
#<module interface>   <control flag>    <module name>      <module arguments>
auth                  required          pam_env.so
auth                  required          pam_unix2.so
account               required          pam_unix2.so
password              requisite         pam_pwcheck.so     nullok cracklib
password              required          pam_unix2.so       use_authtok nullok
session               required          pam_limits.so
session               required          pam_unix2.so
session               optional          pam_umask.so
```

La interpretación del archivo PAM anterior es el siguiente, la primera columna identifica la interfaz del modulo (`<module interface>`), existen cuatro tipos de módulos usados para controlar el acceso a los servicios:

- `auth`: Esta interfaz de módulo autentica el uso. Por ejemplo, solicita y verifica la validez de una contraseña. Los módulos con esta interfaz también pueden establecer credenciales, tales como membrecías de grupo o tickets Kerberos.
- `account`: Esta interfaz de módulo verifica que sea permitido el acceso. Por ejemplo, puede verificar que la cuenta no haya caducado o que el usuario tenga permiso de iniciar sesiones a una hora del día particular.
- `password`: Este módulo se usa para establecer y verificar contraseñas.
- `session`: Esta interfaz de módulo configura y administra las sesiones de usuarios. Los módulos con esta interfaz también pueden realizar tareas

adicionales que son requeridas para permitir acceso, como el montaje de directorios principales de usuarios y hacer el buzón de correo del usuario disponible.

Se puede notar en el archivo que existen varias directivas de interfaces apiladas (por ejemplo, las 2 primeras interfaces de modulo `auth`), con ello el administrador logra exigir diversas condiciones antes de permitir la autenticación del usuario.

La segunda columna corresponde a los indicadores de control (`<control flag>`). Todos los módulos PAM generan un resultado de éxito o fracaso cuando son ejecutados. Los indicadores de control le dicen a PAM que hacer con el resultado, existen cuatro indicadores de control:

- `required`: El resultado del módulo debe ser exitoso para que la autenticación continúe. Si un módulo `required` falla, el usuario no es notificado hasta que los resultados en todos los módulos referenciando esa interfaz sean completados.
- `requisite`: El resultado del módulo debe ser exitóso para que la autenticación continue. Sin embargo, si el resultado de un módulo `requisite` falla, el usuario es notificado inmediatamente con un mensaje reflejando el primer módulo `required` o `requisite` fracasado.
- `sufficient`: El resultado del módulo es ignorado si falla. Pero si el resultado del módulo con el indicador `sufficient` es exitoso y ningún módulo con indicador `required` ha fallado, entonces no se requiere ningún otro resultado y el usuario es autenticado para el servicio.
- `optional`: Se ignora el resultado del módulo. Un módulo con una bandera `optional` sólo es necesario para la autenticación exitosa cuando no hay otros módulos referenciando la interfaz.

La tercer columna indica el nombre del archivo (<module name>) que contiene la interfaz del modulo especificado, estos archivos se encuentran en el directorio /lib/security/.

La cuarta y última columna especifica los argumentos (<module arguments>) utilizados por PAM para transmitir información a un modulo conectable durante la autenticación de algunos módulos. Por ejemplo, el modulo pam_mysql.so abordado en la siguiente sección, utiliza argumentos para indicar el nombre del usuario, clave, IP del servidor, nombre de la base de datos, tabla, etc., logrando así la autenticación a MySQL.

4.4.3.2 Instalación del modulo pam_mysql[17]

Como ya se dijo anteriormente, los usuarios del servicio de correo electrónico serán almacenados en una base de datos MySQL, por lo que es requisito instalar el modulo PAM necesario para llevar a cabo la autenticación de los servicios POP, IMAP y SMTP contra el servidor MySQL. El paquete pam_mysql contiene este módulo, puede obtenerlo desde algún repositorio de la distribución 10.3 de SuSE, aunque la versión que instalaremos a continuación será la disponible en formato tar.gz, para ello ejecute:

```
mail-ha1: ~ # cd /usr/local/
mail-ha1:/usr/local # wget http://prdownloads.sourceforge.net/pam-mysql/pam_mysql-0.7RC1.tar.gz
mail-ha1:/usr/local # tar xvzf pam_mysql-0.7RC1.tar.gz
mail-ha1:/usr/local # cd pam_mysql-0.7RC1/
mail-ha1:/usr/local/pam_mysql-0.7RC1 # ./configure
mail-ha1:/usr/local/pam_mysql-0.7RC1 # make install
```

La instalación crea el archivo (librería) /lib/security/pam_mysql.so que contiene la interfaz del modulo pam_mysql, en la siguiente sección PAM será configurado para autenticar las conexiones a los servicios antes mencionados.

[17] La librería pam_mysql requiere de las librerías mysqlclient, proporcionadas por el paquete libmysqlclient-devel, por lo que debería instalarlas antes de intentar complilar pam_mysql.

4.4.3.3 Configuración del modulo pam_mysql

Es necesario hacerle saber a PAM que se ocupe del proceso de autenticación de los usuarios que intentan acceder a los servicios brindados por los servidores cyrus-imapd y postfix (ver *seccióne*s *5.1 y 5.2 respectivamente*), es decir PAM se encargará de la autenticación a los servicios POP, IMAP y SMPT. Edite y configure los ficheros `/etc/pam.d/pop`, `/etc/pam.d/imap` y `/etc/pam.d/smtp`, como se muestra a continuación:

```
auth sufficient pam_mysql.so user=mail passwd=secret host=localhost          \
   db=mail   table=accountuser usercolumn=username passwdcolumn=password     \
   crypt=1 logtable=log logmsgcolumn=msg logusercolumn=user                  \
   loghostcolumn=host logpidcolumn=pid  logtimecolumn=time

auth sufficient pam_unix_auth.so

account required pam_mysql.so user=mail passwd=secret host=localhost
   db=mail  table=accountuser usercolumn=username passwdcolumn=password      \
   crypt=1 logtable=log logmsgcolumn=msg logusercolumn=user                  \
   loghostcolumn=host logpidcolumn=pid logtimecolumn=time                    \

account  sufficient       pam_unix_acct.so
```

La explicación de cada uno de los parámetros ya fue detallada.

5. IMPLEMENTACIÓN DE SERVICIOS DE CORREO ELECTRÓNICO

En el capitulo anterior se preparó la infraestructura necesaria para montar la solución de correo corporativa, esto incluye sistemas que brindan servicios de enrutamiento, autenticación y acceso remoto al correo electrónico (DNS, PAM y servicio WEB respectivamente) además del diseño de red sobre el cual se apoya la arquitectura de correo de alta disponibilidad. En este capítulo se instalaran y configuraran los siguientes servicios

- Agente de transporte de correo Postfix
- Servidor Cyrus-imap que incluye los protocolos IMAP Y POP3 para lectura de correos.
- Antivirus clamav
- Antispam con spamassassim
- Filtro de correos con amavisd-new
- Horde para acceso remoto al buzón de correo vía web
- Gestión de listas de correo con mailman

Si bien la tareas administrativas del servidor como agregar y borrar usuarios, definir cuotas de correo, administración de dominio virtuales, alias de correo y reenvíos pueden ser llevadas a cabo desde la consola de comando a través de las herramientas brindadas por `cyrus-imapd`, en un entorno donde la cantidad de usuarios supera los cientos, lo mejor sería contar con una aplicación grafica que nos facilite la gestión del servidor, para lo cual se ha escogido la aplicación Web **web-cyradm.**

5.1 Servicio SMTP con Postfix

5.1.1 Introducción al servidor SMTP Postfix

La raíz de cualquier sistema de correo electrónico es el servicio SMTP. En el **capítulo 2** se vieron algunas de las soluciones SMTP más populares, si bien ahí se comentó que Sendmail ha sido el estándar utilizado a través de los años en Unix, existe una nueva tendencia de cambio en muchas distribuciones Linux, por ejemplo, Centos, Fedora, Redhat y SuSE incorporan como nuevo servicio SMTP por defecto Postfix. Además se han tomado en cuenta algunas de sus mejores características:

- Diseño modular para mejorar su seguridad, procesos ligeros con privilegios limitados son lanzados por un demonio master.
- Fácil instalación y puesta en marcha para cumplir necesidades básicas de envío de mensajes.
- Es posible realizar gran cantidad de tareas complejas a través de sus parámetros de configuración personalizables.

5.1.2 Configuración de Postfix

Antes de proceder a configurar Postfix debemos reinstalarlo, la causa de esto es que la versión precompilada para OpenSuSE desabilita las bibliotecas de conexión mysql. Para resolver este inconveniente contamos con la versión de código fuente del paquete (descargada de alguno de los repositorios de la OpenSuSE 10.3), a partir de este y con las herramientas de creación de paquetes precompilados rpm (redhat package manager) se generaran el conjunto de programas con las opciones requeridas.

Los pasos a seguir son:

- Desinstalar el Postfix actual.
- Instalar la versión binaria (la fuentes de Postfix) descargada.

- Crear los paquetes rpm con la herramienta `rpmbuild`[18].
- Instalar los paquetes rpm generados en el directorio `/usr/src/packages/RPMS/i586/`

```
mail-ha1:~ # rpm -ev --nodeps postfix
mail-ha1:~ # cd /usr/local/src/
mail-ha1:/usr/local/src # wget
http://mirrors.uol.com.br/pub/opensuse/distribution/10.3/repo/src-
oss/suse/src/postfix-2.4.5-20.src.rpm
mail-ha1:/usr/local/src # rpm -ivh postfix-2.4.5-20.src.rpm
mail-ha1:/usr/local/src # cd /usr/src/packages/SPECS/
mail-ha1:/usr/src/packages/SPECS # rpmbuild -ba postfix.spec
mail-ha1:/usr/src/packages/SPECS # cd ../RPMS/i586/
mail-ha1:/usr/src/packages/RPMS/i586 # rpm -Uvh postfix-2.4.5-20.i586.rpm
postfix-debuginfo-2.4.5-20.i586.rpm postfix-devel-2.4.5-20.i586.rpm postfix-
mysql-2.4.5-20.i586.rpm
```

En cualquier caso la instalación de Postfix (la versión adjunta a la distribución) brinda el demonio master `/usr/sbin/postfix`. Este se encarga de lanzar todos los procesos necesarios para manejar la entrega de correos.

Postfix almacena sus archivos de configuración en el directorio `/etc/postfix`. A continuación se detallan los más utilizados:

- `access`: Utilizado para el control de acceso, si desea aceptar o denegar algún sistema remoto lo puede hacer aquí.
- `aliases`: Una lista configurable que el protocolo requiere.
- `main.cf`: el archivo de configuración global de Postfix. La mayoría de opciones de configuración se encuentran este archivo.
- `master.cf`: configuración del proceso master de Postfix, especifica la forma en que Postfix interactúa con los diferentes procesos para lograr la entrega de correo.

[18] Para crear los paquetes rpm de Postfix con la herramienta `rpmbuild`, antes debe instalar los paquetes `pcre-devel` y `postgresql-devel` requeridos por la versión precompilada.

- `transport:` hace las correspondencias entre direcciones de correo electrónico y la entrega de mensajes a hosts.

5.1.2.1 Ajustes post instalación de Postfix

Si bien estos ajustes pudieron haberse hecho antes de la reinstalación de Postfix, primero debería saber cuál es la tarea de los archivos generados en el directorio `/etc/postfix/`. Se modificaran algunos ficheros ubicados en `/etc/sysconfig/` para luego generar a través del script `/sbin/conf.d/SuSEconfig.postfix` una configuración con los parámetros adecuados para mejorar la seguridad en cuanto al filtrado de correos (medidas antispam y antiralay), además de adaptar el servicio Postfix al manejo de listas de correo.

El directorio `/etc/sysconfig/` contiene archivos de configuración que establece parámetros por defecto con los que serán generados los archivos de configuración de los servicios de red, en particular Postfix hace uso de los siguientes:

- `/etc/sysconfig/amavis:` para habilitar el uso del filtro amavis en Postfix.
- `/etc/sysconfig/mailman:` parámetros de configuración que indican a Postfix que el servicio de manejo de listas es Mailman y a Mailman que el servidor MTA es Postfix.
- `/etc/sysconfig/postfix:` configuraciones por defecto de Postfix.

El archivo `/etc/sysconfig/amavis` queda de la siguiente forma:

```
USE_AMAVIS="yes"
AMAVIS_SENDMAIL_MILTER="no"
```

Los parámetros más importantes en `/etc/sysconfig/mailman` son:

```
MAILMAN_CGI_GID="www"
MAILMAN_MTA="Postfix"
MAILMAN_SMTPHOST="localhost"
```

```
MAILMAN_DEFAULT_EMAIL_HOST="unan.edu.ni"
MAILMAN_DEFAULT_URL_HOST="unan.edu.ni"
```

En el archivo /etc/sysconfig/postfix se agrega al final de la línea POSTFIX_MAP_LIST el argumento mysql para que establezca los mapas de conexión con la base de datos. POSTFIX_RBL_HOSTS establece una lista separada por comas de los hosts que contienen las listas negras en tiempo real (RBL), para ello se hará uso de las brindadas por SPAMHOUSE.NET y SPAMCOP.NET. POSTFIX_BASIC_SPAM_PREVENTION tiene tres opciones posibles off, medium y hard, con el definimos la agresividad de la política anti UCE[19].

```
POSTFIX_LOCALDOMAINS="unan.edu.ni"
POSTFIX_MAP_LIST="virtual transport access canonical sender_canonical
relocated sasl_passwd:600 relay_ccerts mysql"
POSTFIX_RBL_HOSTS="sbl.spamhaus.org, xbl.spamhaus.org, pbl.spamhaus.org,
bl.spamcop.net"
POSTFIX_BASIC_SPAM_PREVENTION=hard
```

Con las opciones adecuadas habilitadas en los archivos antes mencionados, podemos dar paso a la generación (nuevamente) de los archivos de configuración de Postfix. A continuación ejecute:

```
mail-ha:~ # /sbin/conf.d/SuSEconfig.postfix
```

5.1.2.2 Configurando mapas mysq

Postfix usa los mapas mysql para consultar la base de datos MySQL. Se mantiene la información (usuarios, dominios, alias, etc.) del servicio de correo en una base de datos para manejar el control de accesos. Con esto es posible mantener múltiples servidores de correo usando la misma información, evitándose así la molestia y atraso que implica copiar la información.

[19] UCE (Unsolicited Commercial Email) es otro de los términos utilizados para llamar al correo electrónico no deseado, conocido más comúnmente como spam.

Los parámetros a configurar son:

- `mydestination`: define los nombres de dominios que son locales, en este caso almacenados en la base de datos, este sistema soporta dominios virtuales.
- `virtual_alias_maps`: tablas opcionales de búsqueda de alias y dominios específicos.
- `sender_canonical_maps`: tabas de búsquedas de direcciones opcionales de direcciones de remitentes. Por ejemplo si usted desea reescribir la dirección del remitente `usuario@unan1.edu.ni` a `ususario@unan2.edu.ni` todavía podría seguir enviando mensajes a la dirección `usuario@unan1.edu.ni`.

En el archivo de configuración principal de Postfix `/etc/postfix/main.cf` los parámetros anteriores quedan de la siguiente manera:

```
mydestination = mysql:/etc/postfix/mysql-mydestination.cf
virtual_alias_maps = mysql:/etc/postfix/mysql-virtual.cf
sender_canonical_maps = mysql:/etc/postfix/mysql-canonical.cf
```

Lógicamente se deben crear los archivos de conexión, la estructura de estos es la siguiente:

```
##  archivo /etc/postfix/mysql-mydestination.cf
hosts = localhost
user = mail
password = secret
dbname = mail
table = domain
select_field = domain_name
where_field = domain_name
```

```
##  archivo /etc/postfix/mysql-virtual.cf
hosts = localhost
user = mail
password = secret
dbname = mail
table = virtual
select_field = dest
where_field = alias
additional_conditions = and status = '1'
```

```
##  archivo /etc/postfix/mysql-canonical.cf
hosts = localhost
user = mail
password = secret
dbname = mail
table = virtual
select_field = alias
where_field = username
additional_conditions = and status = '1' limit 1
```

Asegúrese que estos parámetros coincidan con los definidos en la creación de la base de datos (ver *sección 5.2*)

5.1.2.3 Completando la configuración

Con lo hecho hasta ahora Postfix solamente nos serviría para enviar correos desde el propio servidor, ya que hemos obligado una configuración por defecto con una política de seguridad muy alta.

Los parametros a configurar son:

- `inet_interfaces`: la dirección IP de la interfaz de red sobre la cual el servidor recibe los correos. Con la opción `all` el servidor escuchara por todas las interfaces.
- `best_mx_transport`: donde debería el cliente SMTP entregar los correos, configúrelo a `local` para evitar la condición de error *"mail loops back to myself"*.
- `mailbox_transport`: El servicio de buzones local que se encargará de los correos dirigidos a nuestro dominio.

Para estos parámetros, las configuraciones en el archivo `/etc/postfix/main.cf` son puestas a los siguientes valores:

```
inet_interfaces = all
best_mx_transport = local
mailbox_transport = cyrus
```

Aunque el servicio SMTP ya este configurado para funcionar a pleno, las pruebas reales serán hechas una vez completada la implementación de todos los servicios, debido a que postfix requiere de otros servicios como MySQL, mailman, amavisd, spamassassim y cyrus.

5.1.3 Arranque y prueba del servicio SMTP

Una prueba del servicio básico de Postfix (entrega de correos) debe ser llevada a cabo en este momento, otras funcionalidades como antispam, antivirus, listas de correo pueden ser realizadas una vez completado este capítulo. Con las opciones de configuración realizadas enviaremos el primer correo sin necesitar de una aplicación gráfica, simplemente desde la consola utilizando el comando telnet, y depurando cualquier problema de entrega de correo con la información brindada por el servicio syslog del sistema. Antes será necesario ejecutar el script de arranque de Postfix y verificar algunos de los procesos que deberían estar corriendo, para ello desde la consola ejecutar:

```
mail-ha1:/ # rcpostfix start
Starting mail service (Postfix)                                    done
mail-ha1:~ # ps aux | grep postfix
root      3550  0.0  0.3   6404  1716 ? Ss 13:32 0:00 /usr/lib/postfix/master
postfix   3559  0.0  0.3   7728  1960 ? S  13:32 0:00 qmgr -l -t fifo -u
postfix   3658  0.0  0.3   7688  1932 ? S  15:42 0:00 pickup -l -t fifo -u
```

En principio debería estar corriendo el proceso `master` de postfix y los demonios `qmgr` y `pickup`, el resto de demonios que forman la arquitectura de postfix se crean sobre demanda.

El demonio `master` es el proceso residente que se encarga de crear los demás demonios sobre demanda, demonios para enviar o recibir mensajes a través de la red, para entregar el correo localmente, etc. Estos demonios son creados en tiempo de ejecución, de acuerdo a un número máximo configurable por servicio.

El demonio `qmgr` esta a la espera de la llegada del correo entrante y planifica la entrega de este a través de los proceso de entrega de postfix.

`Pickup` es el demonio que recupera el correo local, su función es esperar una señal que le indique que el nuevo correo ha sido retenido dentro del directorio `maildrop`, y entonces lo injecta al demonio `cleanup`. Se espera que este programa sea ejecutado por el proceso administrador `master`.

Para una ayuda completa sobre la estructura de Postfix consulte su documentación, ubicada en el directorio `/usr/share/doc/packages/postfix`.

5.1.3.1 Archivos de Log

Todos los sistemas basados en Unix, y en particular las distribuciones Linux como Fedora, CentOS o SuSE tienen un sistema de registro de logs. El demonio syslog es un mecanismo que centraliza todos los eventos del sistema, recopilando información de diversas aplicaciones que se ejecutan en el sistema. Syslog escribe toda la información generada por los servicios en archivos de registro (texto plano), por lo general ubicados en el directorio `/var/log`.

Esta información escrita en los archivos de registro puede ser utilizada para identificar problemas en los servicios y puede variar dependiendo de la aplicación. Para el servicio de correo electrónico los archivos de logs generados son:

- `/var/log/mail`: registro de correos enviados, recibidos, filtrado de spam y antivirus por parte de Amavisd-new, etc.
- `/var/log/mail.err`: errores graves de configuración que imposibilitan la ejecución del demonio `master` de postfix.
- `/var/log/mail.warn`: problemas de configuración que aunque no imposibilitan la ejecución del demonio master, deberían ser revisadas.
- `/var/log/mail.info`: igual a `/var/log/mail`.

Habrá varios archivos de registros aparte de los anteriores que tendrá que revisar constantemente para identificar problemas también relacionados con los servicios de correo electrónico, por ejemplo los registros del sistema `/var/log/messages` y `/var/log/warn`.

5.1.3.2 Probando Postfix con telnet

Si bien el demonio `master` de postfix pudo haber arrancado sin ningún problema, no es garantía de que el sistema sea capaz de enviar y recibir correos. La manera más simple de probar el servicio SMTP es estableciendo conexiones de red con la aplicación telnet, utilizando los comandos más comunes del protocolo.

Primero desde una terminal de comandos del servidor de correo, intentamos conectar el servidor SMTP en el puerto 25 por la interfaz de localhost. Para suspender la sesión escriba el comando SMTP `quit`, a continuación mostramos el ejemplo en cuestión:

```
mail-ha1:~ # telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mail-ha.unan.edu.ni ESMTP Postfix
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Con lo mostrado con seguridad postfix está escuchando y atendiendo conexiones en la interfaz de localhost 127.0.0.1. La prueba más importante será establecer una conexión a la interfaz con la dirección IP 10.1.120.3, y tratar de enviar un mensaje a cualquier dirección de correo electrónico valida. Lo ideal sería realizar la prueba desde otra máquina conectada al segmento de red capaz de alcanzar el servidor, sin embargo lo más práctico es realizar la conexión desde el mismo servidor. Abajo se muestra una sesión SMTP exitosa:

```
mail-ha1:~ # telnet 10.1.120.3 25
Trying 10.1.120.3...
Connected to 10.1.120.3.
Escape character is '^]'.
220 mail-ha.unan.edu.ni ESMTP Postfix
HELO test
250 mail-ha.unan.edu.ni
MAIL FROM: jonas@unan.edu.ni
250 2.1.0 Ok
RCPT TO: edison@unan.edu.ni
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
ESTE ES EL CUERPO DEL MENSAJE, PUNTO "." PARA FINALIZAR
.
250 2.0.0 Ok: queued as CCF9B3E54E
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Cada uno de los comandos utilizados son explicados en el capítulo 3 (protocolos de correo electrónico)

5.2 Servicio de acceso a buzones con Cyrus IMAP

5.2.1 Introducción al servidor POP/IMAP Cyrus IMAP

En el **capítulo 3** (Protocolos de correo electrónico) se trató de explicar de manera general el funcionamiento del protocolo de acceso a buzones POP3, en este capítulo se muestra como proporcionar acceso remoto a los usuarios de correo electrónico a través de este. Después de todo lo que realmente importa en un servicio, es su capacidad de brindar acceso desde donde el usuario lo desee, la mejor alternativa a esto son los protocolos de acceso remoto (encima del servicio de webmail), sobre todo cuando no se cuenta con una buena conexión a Internet, como podría serlo una conexión dialup. Además de POP3 se configurará IMAP, un protocolo más complejo que POP3, sin embargo ofrece muchas más opciones y flexibilidad a los usuarios; las características de IMAP se vuelven más interesantes cuando se cuenta con una buena conexión, por ejemplo en redes de área local.

El servidor Cyrus IMAP difiere de otras implementaciones IMAP (dovecot, Courier-imap, etc.) en que generalmente este está diseñado para correr en servidores seguros, donde usuarios normales no están autorizados a entrar. La base de datos de buzones es almacenada en partes del sistema de archivos totalmente privadas al sistema Cyrus IMAP. Todos los usuarios acceden a los mensajes a través de los protocolos IMAP y POP3.

La base de datos de buzones privados está diseñada para dar al servidor grandes ventajas de eficiencia, escalabilidad y gestión. Se permiten múltiples conexiones concurrentes de lectura/escritura al mismo buzón. El servidor soporta listas de control de acceso sobre los buzones y cuotas de almacenamiento sobre jerarquía de buzones.

La base de datos de los buzones ubicada en el directorio `/var/lib/imap/`, almacena información referente a:

- Cuotas de correo.
- Alias de correo
- Información estadística de buzones, login de accesos, mensajes entregados, errores, etc.

Entre las características más destacadas de Cyrus están:

- Soporta el estándar IMAP4rev1 descrito en el RFC 3501.
- El servidor soporta cualquier mecanismo de autenticación disponible de la librería SASL[20].
- Soporte de los protocolos seguros imaps y pop3s (usandos SSL)
- Soporte de SIEVE para filtro de correos del lado del servidor.

[20] SASL (Simple Authentication and Security Layer) es una estructura que provee autenticación y servicios de seguridad de datos en protocolos orientados a conexión. Este brinda la interfaz de una estructura entre los protocolos y los mecanismos, permite a los nuevos protocolos el rehúso de mecanismos ya existentes y a los viejos protocolos el uso de los nuevos mecanismos. SASL está especificado en el RFC 442.

5.2.2 Configuración del servidor Cyrus IMAP

La configuración del servidor Cyrus es llevada a cabo en dos archivos de configuración:

- `/etc/cyrus.conf`: configuración del proceso master Cyrus. Este define los procedimientos iniciales, los servicios y eventos a ser cargados por el proceso master. El archivo tiene tres secciones bien definidas START, SERVICES y EVENTS con la forma:

```
section {
                    name arguments
                         ...
                         ...
             }
```

  Donde `section` es una palabra que puede ser sustituida por el tipo de sección START, SERVICES o EVENTS.

- `/etc/imapd.conf`: este es el archivo de configuración del servidor Cyrus. Este define los parámetros locales para IMAP. Cada línea del archivo tiene la forma:

```
<option>: value
```

  Donde `option` es el nombre del parámetro de configuración que ha sido establecido y `value` es el valor de la opción de configuración.

5.2.3 Archivo /etc/cyrus.conf

Los parámetros importantes a configurar en el archivo `/etc/cyrus.conf` serán los servicios atendidos por el proceso master de Cyrus (sección SERVICES), tomando en cuenta que a los usuarios remotos se les ofrecerá el protocolo POP3 solamente, mientras que el protocolo IMAP escuchara conexiones locales. El protocolo IMAP será el utilizado por el cliente de correo web (ver *sección5.6*), y ya que este está instalado en el mismo servidor físico no hay razón para poner a escuchar el servicio en todas las interfaces de red. La configuración propuesta es la siguiente:

```
START {
  recover       cmd="ctl_cyrusdb -r"
  idled         cmd="idled"
}

SERVICES {
  imap          cmd="imapd" listen="127.0.0.1:imap" prefork=0
  pop3          cmd="pop3d" listen="pop3" prefork=0
  lmtpunix      cmd="lmtpd" listen="/var/lib/imap/socket/lmtp" prefork=0
proto="udp" prefork=1
}

EVENTS {
  checkpoint    cmd="ctl_cyrusdb -c" period=30
  delprune      cmd="cyr_expire -E 3" at=0400
  tlsprune      cmd="tls_prune" at=0400
}
```

Si desea añadir mas servicios puede agregarlo en la sección SERVICES, servicios adicionales como imaps y pop3s, requieren la generación de certificados SSL.

5.2.4 Archivo /etc/imapd.conf

La librería SASL, subyacente al servidor Cyrus, permite la elección de utilizar el mecanismo de autenticación que más nos interese, incluyendo CRAM-MD5, DIGEST-MD5, KERBEROS_V4, PAM. El mecanismo de autenticación utilizado (ver *sección 4.4.3.1*) para todos los servicios de correo electrónico es PAM. El archivo de configuración `/etc/sysconfig/saslauthd` del proceso master `/usr/sbin/saslauthd` implementa PAM como método de autenticación por defeco:

```
SASLAUTHD_AUTHMECH=pam
SASLAUTHD_THREADS=5
```

Archivo de configuración `/etc/sysconfig/saslauthd`

El archivo de configuración `/etc/imapd.conf` para el servicio de buzones propuesto tiene la siguiente forma:

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
admins: cyrus
allowanonymouslogin: no
autocreatequota: 204800
quotawarn: 90
timeout: 30
poptimeout: 10
drachost: localhost
sasl_pwcheck_method: saslauthd
lmtp_downcase_rcpt: yes
```

Algunos de los parámetros más interesantes a detallar en el archivo anterior son:

- `configdirectory`: indica la ruta del directorio de configuración de servidor IMAP, en este directorio se guarda la base de datos de los buzones de usuario.
- `partition-default`: la ruta de la partición en donde se almacenan los buzones de correo.
- `admins`: lista de usuarios con derechos administrativos sobre el servidor Cyrus IMAP, el único usuario indicado a tener estos privilegios es Cyrus.
- `aoutocreatequota`: cuota por defecto (en bytes) para usuarios normales que crean su propia cuenta IMAP.
- `reject8bit`: si está habilitado (`yes`), lmtpd rechazara mensajes con cabeceras de 8 bits.
- `quotawarn`: porcentaje de utilización de la cuota del buzón, sobre el cual el servidor genera advertencias de que el buzón está llegando a su límite de almacenamiento.
- `poptimeout`: tiempo en minutos en que el servidor abandonará la conexión por inactividad por parte del cliente.
- `sasl_pwcheck_method`: el mecanismo utilizado por el servidor para verificar las claves en texto plano, como se se dijo anteriormente el proceso master `/usr/sbin/saslauthd` se encargará de esto.
- `sasl_pwcheck_method`: si está habilitado, lmtpd convertirá las direcciones de buzones a minúsculas.

5.3 Administración de servicios de correo electrónico

Aunque es posible administrar cada uno de los servicios configurados hasta el momento (IMAP, POP3 Y SMTP) desde la consola de comandos, esta es una tarea no muy sencilla, que aunque con un poco de practica con el tiempo se convertirían en tareas administrativas de rutina. En entornos donde la cantidad de usuarios es considerablemente alta, lo más indicado es contar con una herramienta de administración grafica que nos permita:

- Agregar, borrar y administrar buzones de usuarios y dominios de correo electrónico.
- Administración de alias y reenvíos de correo electrónico.
- Establecer servicios individuales y por dominio (SMTP, POP3 o IMAP).

5.3.1 Administración gráfica de servicios con Web-Cyradm

Web-cyradm es una aplicación Web usada por administradores de sistemas para facilitar las tareas de gestión de servicios de correo, a través de una interfaz web amigable. Web-cyradm es una alternativa gráfica a la administración de buzones desde la línea de comandos `cyradm`.

Antes de proceder con la descarga e instalación de web-cyradm, necesita instalar los módulos PHP5 `php5-pear`, `php5-gettext` y `php5-pear-db`, en el caso de los primero dos paquetes puede hacerlo desde la herramienta de administración gráfica YAST. `php5-pear-db` no se encuentra en el grupo de paquetes binarios de la versión instalada de SuSE, por lo cual deberá bajarlo desde algunos de los repositorios de Internet, en cualquier caso si tiene instalado `smart` y configurado con los repositorios adecuados ejecute:

```
mail-ha1:~ # smart install php5-pear php5-pear-db php5-gettext
```

También es requisito de Web-cyradm tener instalado y corriendo un servidor LAMP[21], por lo que debe de arrancar los servicios correspondientes:

```
mail-ha1:~ # rcapache2 start
Starting httpd2 (prefork)                                          done
mail-ha1:~ # rcmysql start
```

La instalación de web-cyradm puede ser llevada a cabo con unos sencillos pasos, vistos a continuación:

```
mail-ha1:~ # cd /srv/www/htdocs/
mail-ha1:/srv/www/htdocs # wget www.web-cyradm.org/web-cyradm-svn-
0.5.5.tar.gz
mail-ha1:/srv/www/htdocs # tar xvzf web-cyradm-svn-0.5.5.tar.gz
mail-ha1:/srv/www/htdocs # mv web-cyradm-svn-0.5.5 web-cyradm
```

Web-cyradm brinda los siguientes scripts SQL:

- `/srv/www/htdocs/web-cyradm/scripts/insertuser_mysql.sql:` primero inserta los usuarios que tendrán los permisos administrativos sobre la base de datos del servicio y después crea la base de datos cuyo nombre es *"mail"*.
- `/srv/www/htdocs/web-cyradm/scripts/create_mysql.sql:` crea la estructura de las tablas requeridas por el servicio, además inserta un admin-user (usurio administrador de la aplicación web-cyradm) y el usuario *"cyrus"* entre otras cosas.

Antes de ejecutar estos scripts debe cambiar las claves por defecto, *"secret"* para el usuario administrador de la base de datos y *"test"* para el usuario administrador de la aplicación Web.

[21] El servidor LAMP (Linux-Apache-MySQL-PHP) debería estar instalado si la instalación ha sido como servidor.

```
mail-ha1:/srv/www/htdocs # mysql -u root < web-
cyradm/scripts/insertuser_mysql.sql
mail-ha1:/srv/www/htdocs # mysql mail -u mail -p < web-
cyradm/scripts/create_mysql.sql
Enter password:
```

Web-cyradm posee un solo archivo de configuración `/srv/www/htdocs/web-cyradm/config/conf.php`, en el se establecen parámetros como el idioma utilizado, valores de conexión con el servidor Cyrus, valores de conexión con la base de datos MySQL, generación de archivos de log, cuotas por defecto y otros no menos importantes.

Utilizar la aplicación es una tarea sencilla, unos pocos minutos de prueba bastaran para saber qué es lo que se puede hacer con ella, la *figura 5.1* muestra la pantalla de bienvenida del sistema, para ingresar utilice el usuario *admin* y la clave *test*, si es no fue modificada en los scripts SQL.

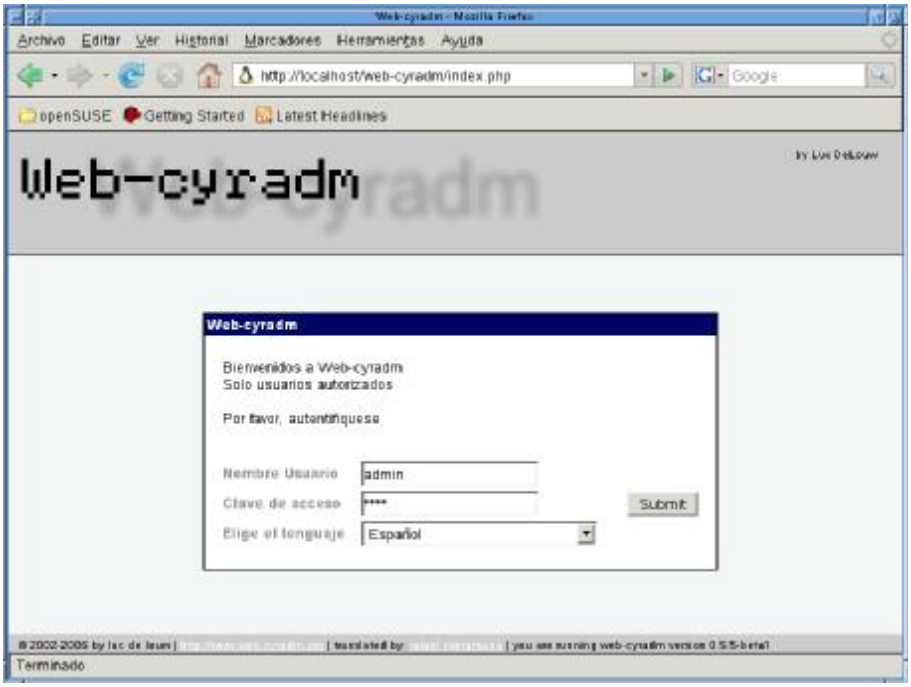


Figura 5.1 *Login de acceso a sistema web-cyradm*

5.3.2 Administración desde la línea de comandos con cyradm

Para aquellos no muy aficionados a las aplicaciones de ventanas, el servidor Cyrus IMAP brinda la herramienta `cyradm`. El comando `cyradm` (ver documentación completa en su página man) administra la creación y eliminación de ACLs (Access Control List) y cuotas de correo electrónico.

La forma de utilizar `cyradm` se asemeja al manejo de un intérprete de comandos, con sus propias órdenes internas, relacionadas a la administración exclusiva del servidor Cyrus IMAP. Para loguearse al servidor Cyrus ejecute `cyradm` con los parámetros adecuados:

```
mail-ha1:~ # cyradm --user cyrus --server localhost --auth plain
```

La clave de acceso será la establecida para el usuario Cyrus en el script SQL `/srv/www/htdocs/web-cyradm/scripts/create_mysql.sql` (ver la sección anterior).

Una vez que ha ingresado a la consola de trabajo, `cyradm` le mostrará un prompt con el carácter “>”, tecleando el comando ***help*** obtendrá una ayuda rápida de los comandos soportados (ver *tabla 5.1*).

Comando	Abreviatura	Descripción
`Createmailbox`	`cm`	Crea un buzón de correo
`Deleteaclmailbox`	`dam`	Elimina una ACL establecida sobre un buzón de correo
`Deletemailbox`	`dm`	Elimina un buzón de correo
`Help`		Obtiene una ayuda de los comandos
`Listaclmailbox`	`lam`	Lista las ACLs establecidas sobre un buzón de correo
`Listmailbox`	`lm`	Lista los buzones de correo existentes
`Lisquota`	`Lq`	Lista la cuota total desde la raíz
`Listquotaroot`	`lqr, lqm`	Lista las cuotas de los buzones desde la raíz
`Quit`		Salir y cerrar la conexión establecida con el servidor Cyrus IMAP
`Renamemailbox`	`renm`	Renombra un buzón de correo
`Setaclmailbox`	`sam`	Establece una ACL sobre un buzón de correo
`Setquota`	`Sq`	Establece la cuota limite de buzón de correo

Tabla 5.1 *Comandos cyradm para administración de servidor Cyrus IMAP*

5.4 Filtrado de correos, detección de Virus y Spam

Una de las características más poderosas que puede ser ofrecida a los usuarios es el filtrado de correos. Implementar un buen filtro de correos contra *Virus* y *Spam*, ayudará en gran medida a mejorar la eficiencia de los usuarios, solo habría que imaginar el tiempo perdido por un usuario, al tener que borrar diariamente decenas de correos basura, sumándole a esto los daños causados por correos infectados.

Existen varias maneras de filtrar correo, por ejemplo puede configurar Postfix para:

- Entregar el correo a un programa de filtrado (Procmail), de acuerdo a cierta información encontrada en las cabeceras del correo (dirección origen, dirección destino, asunto del mensaje, etc.) se le aplicará una regla u otra.
- Entregar el correo a un programa de filtrado (Amavisd-new), este realizará un análisis antivirus y antispam, haciendo uso de servicios como Clamav y Spamassassim. De acuerdo al resultado retendrá el correo o lo entregará al buzón de usuario.

Por otra parte la administración de los filtros puede ser delegada al usuario final, esta es una tarea que no muchos estarían dispuestos a asumir, sobre todo por el grado de complejidad medio que implicaría para un usuario promedio tener que editar sus propios filtros (ejemplo los filtros procmail). Sin embargo la mayoría de los MUA (*Mozilla Thunderbird, Eudora, Incredimail, etc.*) permiten filtrar el correo en el lado del cliente, note que en este caso el correo ya ha sido aceptado en el buzón, el filtro se hace una vez está en el equipo del usuario.

5.4.1 Filtrando Spam con Postfix

El servidor SMTP Postfix es capaz de aplicar filtros, basándose de la información encontrada en la cabecera y en el cuerpo de los mensajes de correo electrónico. En el archivo de configuración principal `/etc/postfix/main.cf` se encuentran los parámetros de configuración que posibilitan el filtrado del servidor SMTP Postfix.

Postfix se puede basar en los métodos *HELO, SENDER, RECIPIENT* y *DATA* recibidos por un cliente cualquiera (SMTP origen) para restringir la recepción del correo electrónico. Estas restricciones son conocidas como **smtpd_*_restrictions**[22], donde el * puede ser reemplazado por: ***client, helo, sender, recipient y data.***

- `smtpd_client_restriction`: se utilizan RBLs (***R**ealtime **B**lackhole **L**ist*) para rechazar correos a través de esta directiva, en la *sección 5.1.2.1* puede ver la lista de RBLs utilizadas. Con `smtpd_client_restriction` definimos a que clientes les serán permitidas o denegadas las conexiones al servidor SMTP. La línea completa luce de la siguiente forma:

```
smtpd_client_restrictions = permit_mynetworks, reject_rbl_client
sbl.spamhaus.org, reject_rbl_client xbl.spamhaus.org,
reject_rbl_client pbl.spamhaus.org, reject_rbl_client
bl.spamcop.net, reject_unknown_client
```

  Donde el argumento `reject_rbl_client` definen los sitios que mantienen las RBLs.

- `smtpd_helo_restrictions`: define las restricciones que se le aplican a un cliente cuando este se conecta (método HELO), se deniegan los nombres de hosts inválidos y por ultimo todos aquellos que no envían su FQDN en el mensaje HELO también son denegados.

```
smtpd_helo_restrictions = permit_mynetworks,
reject_invalid_hostname, reject_non_fqdn_hostname
```

  Observe que rechazar un correo electrónico por no recibir el FQDN del remitente puede causar muchos problemas. Existen muchos servidores de correo que están configurados para solo enviar como nombre *hostname* en vez de

[22] La mayoría de los parámetros ya deberían estar configurados como en los ejemplos mostrados, sin embargo se recomienda sobrescribir la configuración con del comando `postconf` (ver su página man).

hostname.dominio.com, por lo que es posible que se generen muchos falsos positivos. Se recomienda estrictamente no hacer uso de este argumento.

- `smptd_sender_restrictions`: las restricciones en el remitente deben ser bastante directas, abajo puede ver que se utiliza un archivo para saber si hay remitentes permitidos (`/etc/postfix/access`), hasta entonces se deniegan con `reject_unknown_sender_domain`.

```
smtpd_sender_restrictions = hash:/etc/postfix/access,
reject_unknown_sender_domain
```

`smtpd_recipient_restrictions`: nuevamente se comprueba un archivo para verificar cuales son los recipientes aceptados o rechazados, este archivo de acceso es uno de los más importantes.

```
smtpd_recipient_restrictions = permit_mynetworks,
reject_unauth_destination, check_recipient_access
hash:/etc/postfix/recipient_access
```

- `smtpd_data_restrictions`: previene la conexión de clientes con comandos *pipelining* SMTP. *Pipelining* es una manera de enviar correo rápida y sin espera de respuesta, es una táctica muy utilizada por los Spammers.

```
smtpd_data_restrictions =  reject_unauth_pipelining
```

Existen otras formas de filtrar correo con Postfix, por ejemplo chequeando el cuerpo y las cabeceras de los mensajes. Postfix se puede configurar para denegar los mensajes que tienen una cadena específica en su cuerpo o cabecera a través de los parámetros `header_checks y body_checks.`

5.4.2 Filtrando Virus y Spam con Amavisd-new

Es indispensable contar con un servicio *antivirus* y *antispam* que se encargue del escaneo y filtrado de los correos procesados por el servidor SMTP Postfix. No hace falta explicar la relevancia de un servicio antivirus, más aún cuando hablamos de servicios de correo electrónico; estadísticamente en su gran mayoría los virus se propagan en la red a través de los correos electrónicos.

Debe quedar claro que un sistema de correo con filtro antivirus no debe ser nunca un sustituto completo a un software antivirus instalado localmente.

Otro tipo de correo inofensivo pero bastante molesto es el spam. Luchar contra el spam es mucho más difícil que luchar contra los virus, esto se debe a que cada correo lleva una firma única y a que en el diseño del protocolo SMTP no está implícita la autenticación. El spam se caracteriza por su contenido arbitrario, algunos vienen en inglés otros en español, coreano, chino, etc. lo que complica aún más su detección. Estudios recientes revelan que más del 60% del tráfico de correo electrónico en Internet es spam.

Amavisd-new es el filtro utilizado para realizar esta tarea. Postfix envía el correo recibido al proceso master `/usr/sbin/amavisd` por el puerto 10024 escuchando en localhost (ver archivo de configuración `/etc/postfix/master.cf`) luego el proceso master `/usr/sbin/amavisd` escaneará el correo utilizando los servicios antispam spamassassim y antivirus clamav. Si amavis encuentra algún virus o determina que el correo es un spam, lo tratará de acuerdo a las políticas de configuración establecidas. En todo caso si amavis puede decidir enviar el correo al servidor SMTP a través del puerto 10025 en localhost, seguramente con algún cambio en la cabeceras.

Cada uno de las variables de configuración de amavis explicadas en las secciones siguientes pueden ser modificadas en el archivo de configuración `/etc/amavisd.conf`.

5.4.2.1 Estableciendo políticas de filtrado

Cuando amavis recibe un correo con virus, lo califica como spam o lo detecta como un algún tipo de archivo prohibido (banned files con extensiones .exe, .bmp, .cap, etc.), puede hacer lo siguiente:

1. Pasar El mensaje (*Pass*), el buzón obtendrá el mensaje.
2. Descartar el mensaje (*Discard*), el buzón no obtendrá el mensaje. el remitente no es notificado acerca del fallo del envío, y el mensaje es puesto en cuarentena si es que se ha activado la cuarentena.
3. Rebotar el mensaje (*Bounce*), el recipiente no obtendrá el mensaje, el remitente obtiene una notificación de que el mensaje ha sido descartado (*Bounce*).
4. Rechazar el mensaje (*Reject*), el recipiente no obtendrá el mensaje, el remitente debería conseguir un rechazo o una notificación de que no se realizó la entrega, pero amavis debería confiar en el MTA que está usando para publicarlo realmente. Esta configuración no debería ser usada con Postfix.

La diferencia principal entre rebotar (*Bounce*) y rechazar (*Reject*) un mensaje, está en quien prepara el DSN (Notificación del Estado de Entrega, por sus siglas en ingles *Deliverey Status Notification*):

- *Reject* significa que el MTA prepara y envía el DSN.
- *Bounce* significa que es amavis quien preparara y enviara el DSN (que además es más informativo).

Puesto que estamos utilizando Postfix, nuestras opciones serán solamente, Pasar (*Pass*), Descartar (*Discard*) y Rebotar (*Bounce*). No es lo más indicado Rechazar (Reject) los correos, sería como decirle a un spammer: *"estoy recibiendo tus correos, envíame más que estoy preparado para procesarlos"*. Lo mejor es siempre descartar el correo y no notificárselo al servidor SMTP origen.

En amavis estas opciones son nombradas como `D_PASS`, `D_DISCARD` y `D_BOUNCE`, y pueden usarse con las variables `$final_spam_destiny`, `$final_virus_destiny`, `$final_banned_destiny` y la variable `$final_bad_header_destiny`. Si lees la documentación de amavis, veras que estos parámetro son referidos como `$final_*_destiny`. En nuestra configuración aplicaremos la política *Discard* para los spam, virus y archivos prohibidos (banned files), también se activará la cuarentena de correos. De inhabilitar por completo la cuarentena, el correo descartado realmente se perderá.

5.4.2.2 Cuarentena de Correos

Amavis puede poner en cuarentena (por lo general en el directorio `/var/spool/amavis/`) un mensaje que es clasificado como spam, virus o archivo prohibido o si prefiere puede mandar el correo al buzón de otro usuario distinto al destinatario real.

Se recomienda enviar todo el correo spam detectado a una cuenta que gestione el spam (*spam@unan.edu.ni*), los correos con virus a la cuenta virus@midominio.com.ni, y así sucesivamente. Aunque con los virus lo mejor sería almacenarlos en un directorio aislado dentro del servidor que no esté asociado a ninguna buzón de correo. Esto se logra definiendo las variables de la siguiente forma:

```
$spam_quarantine_to spam@unan.edu.ni;
$virus_quarantine_to virus@unan.edu.ni;
$banned_quarantine_to banned@unan.edu.ni;
```

Para hacer que amavis almacene los archivos en cuarentena en un directorio distinto al buzón de correos, debe configurar el parámetro `$QUARENTINEDIR`, sin embargo es posible elegir poner los virus, spam y archivos prohibidos en directorios independientes, es decir cada uno colgado en un subdirectorio del definido en el parámetro `$QUARENTINEDIR`.

5.4.2.3 Envío de Notificaciones[23]

Amavis permite enviar avisos cada vez que se captura un spam, virus, archivo prohibido o una cabecera mal formateada, esto no es algo que en realidad se necesite implementar, pero explicaremos como hacerlo usando los virus como ejemplo:

```
$virus_admin                = "edison\@$mydomain";
```

Aquí la dirección de correo puede ser cualquiera, claro es necesario que sea una cuenta real del sistema, o si preferimos podemos utilizarlo como un alias.

Lo siguiente es establecer el valor de la variable `$mailfrom_notify_admin`, para entender este parámetro respondámonos lo siguiente: ¿Cuál será la dirección del remitente cuando se envíe una notificación al `virus_admin`?

```
$mailfrom_notify_admin      = "notificaciones\@$mydomain";
```

Las variables `$mailfrom_notify_recip` y `$mailfrom_notify_spamadmin` permiten fijar diferentes remitentes dependiendo de quien recibirá el mensaje, en realidad no necesitamos definir estos 2 parámetros, puesto que solo estamos notificando sobre los virus encontrados. Estas variables pertenecen a las notificaciones del *spamadmin*.

La última configuración relacionada con las notificaciones puede ser hecha con la variable `$hdrfrom_notify_admin`, este parámetro modifica la cabecera `Header-From`:

```
$hdrfrom_notify_admin = "Content Filter <edison\@$mydomain>";
```

Cuando fijamos esta variable, se eliminará el `$mailfrom_notify_admin` y al recibir el correo el cliente (cuenta del usuario notificado) mostrará su contenido en lugar de `$mailfrom_notify_admin`. Si no utilizáramos este parámetro lo que está en la variable

[23] Para deshabilitar las notificaciones de virus, se debe comentar el parámetro `$virus_admin`. La ausencia del parámetro `$virus_admin` significa ninguna notificación.

`$mailfrom_notify_admin` sería el valor usado en la dirección que va en la cabecera `Header-From`.

Todo lo antes dicho acerca de las notificaciones de virus es igualmente valido para el spam, archivos prohibidos y correos con cabeceras mal formateadas. La única acepción a esto es cuando se procesa un spam, con este también se tiene la capacidad de modificar la línea "`Subject`" del mensaje, y agregar banderas (por ejemplo **SPAM**) en la cabecera del correo.

Recuerde leer la documentación del archivo `/etc/amavisd.conf` para más detalles.

5.5 Listas de correo electrónico con Mailman

Es probable que alguna vez hayamos recibido algún correo con boletines electrónicos de anuncios sobre algún producto, o incluso quizás tratando algún tema de discusión de deportes, política o entretenimiento. Cada uno de estos ejemplos tiene algo en común, aparte de ser un correo electrónico, posiblemente sea una lista de correo a la cual nos hemos subscritos de manera involuntaria o con nuestra previa autorización.

Es fácil enviar un correo electrónico a varias direcciones con las opciones Para, CC y BCC de los clientes de correo, sin embargo cuando la cantidad de destinatarios supera los cientos se vuelve una tarea muy tediosa. La solución a esto es una lista de correo electrónico, con una lista de correo podemos dirigirnos a miles de destinatarios con solo enviar el mensaje a la dirección que identifica la lista.

Mailman es un software que ayuda a gestionar listas de discusión por correo electrónico. Aunque existen otros programas con el objetivo de manejar listas (como mayordomo y

smartlist) utilizaremos Mailman, ya que este se encuentra preconfigurado para ser usado con Postfix y el servidor Web Apache2.

Mailman también integra la mayoría de las cosas que el administrador de la lista desearía que hiciera, esto incluye el manejo de archivos adjuntos, correo con portales de noticias, bajas masivas e individuales, prevención de spam, administración vía comandos, manejo de dominios virtuales y muchas otras.

La lista de funciones de Mailman es extensa e incluye las siguientes:

- Integración con la Web, la mayoría de listas de correo pueden gestionarse con ina intuitiva interfaz Web.
- Soporte de múltiples idiomas.
- Archivos de lista de correos, cada tema de discusión es archivado para ser revisado como un historial.
- Pagina web personalizada, cada lista de correo tiene su propia página web personalizada.

5.5.1 Configuración de Mailman

En la *sección 5.2* se observó como modificar el archivo de configuración `/etc/sysconfig/mailman` para lograr integrar el sistema de listas Mailman con el servicio SMTP Postfix. Ahí mismo vimos otro parámetro que le indica al servicio Mailman cuál es el grupo del servicio Web (parámetro `MAILMAN_CGI_GID`), este le permite a Mailman unirse al grupo del servidor Apache2 y generar las configuración deseadas para su integración.

Asegúrese nuevamente del valor de los parámetros en el archivo de configuración /etc/sysconfig/mailman:

```
MAILMAN_CGI_GID="www"
MAILMAN_MTA="Postfix"
MAILMAN_SMTPHOST="localhost"
MAILMAN_DEFAULT_EMAIL_HOST="unan.edu.ni"
MAILMAN_DEFAULT_URL_HOST="unan.edu.ni"
```

Al script `/sbin/SuSEconfig` le indicamos que reconfigure el servicio Mailman:

```
mail-ha1:~ # /sbin/SuSEconfig -module mailman
```

A continuación proceda a llamar el script `/usr/lib/mailman/bin/mmsitepass` como usuario root para establecer la clave del sitio master:

```
mail-ha1:~ # /sbin/SuSEconfig -module mailman
```

En el archivo de configuración del servidor SMTP `/etc/postfix/main.cf` agregue la siguiente línea, para indicarle a Postfix donde está la lista de alias mailman:

```
alias_maps= hash:/var/lib/mailman/data/aliases
```

Con las configuraciones anteriores ya puede crear la primer lista de correo, esta siempre deberser la del administrador de listas (lista de nombre mailman), para crearla ejecute como root:

```
mail-ha1:/etc/postfix # /usr/lib/mailman/bin/newlist mailman
Enter the email of the person running the list: edison@unan.edu.ni
Initial mailman password:
Hit enter to notify mailman owner...
```

El comando anterior solicita la dirección del administrador de la lista, así como su clave. Note que el servicio SMTP debe estar ejecutándose para que se envíe la primera notificación al administrador Mailman.

Finalmente para integrar Mailman al servidor Web Apache2, debe agregar la bandera `MAILMAN` al archivo de configuración `/etc/sysconfig/apache2` con el comando:

```
mail-ha1:~ # a2enflag MAILMAN
```

5.5.2 Inicio y Prueba del servicio.

Para integrar Mailman al servidor Web y al servidor SMTP es necesario reiniciar los demonios master de apache2 y Postfix:

```
mail-ha1:~ # rcapache2 restart
mail-ha1:~ # rcpostfix restart
```

El servicio Mailman tiene el script de arranque `/etc/init.d/mailman`, establecer los niveles de arranque e iniciar el servicio con:

```
mail-ha1:~ # chkconfig --level 345 mailman on
mail-ha1:~ # rcmailman start
```

La lista mailman no debe ser utilizada para suscripción de usuarios, esta es creada solamente con fines administrativos de configuración del servicio. Si lo que desea es crear una nueva lista para tratar un tema de discusión utilice el script de creación de listas `/usr/lib/mailman/bin/newlist`.

Para probar el servicio se carga en el navegador web el URL de la única lista creada hasta el momento (***http://unan.edu.ni/mailman/admin/mailman***), para hacer login utilice la clave ingresada cuando se creó la lista, la *figura 5.2* muestra la interfaz de administración de la lista de correo ***mailman@unan.edu.ni:***

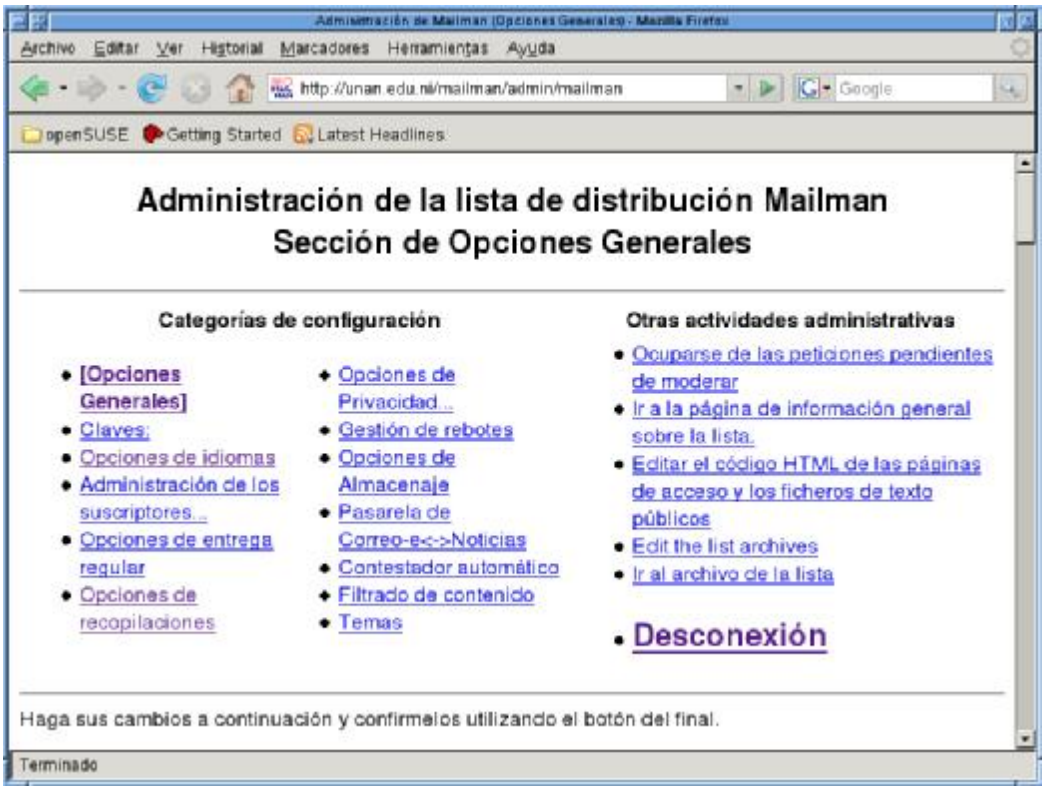


Figura 5.2 *Interfaz de administración mailman*

5.6 Correo Web

El acceso web, es un servicio que siempre debe estar disponible en una solución de correo electrónico, sobre todo cuando la cantidad de usuarios móviles es grande. De no contar con el servicio de correo web un usuario podría verse obligado a utilizar otro servicio (servicios de correo gratuitos) cuando esté fuera de su organización. Aunque POP3 e IMAP son los protocolos de facto para acceso remoto, ambos requieren de un cliente local (MUA) y de configuraciones adicionales no muy sencillas, por lo que lo más cómodo para el Administrador seria publicar una URL en donde los usuarios puedan ingresar al servicio.

Usuarios que están en constante movimiento podrían encontrarse en situaciones en las que no tienen el control del equipo (por ejemplo cybercafés), esto les impediría realizar configuraciones de acceso remoto con los protocolos POP3 e IMAP. La situación puede complicarse aun más en redes que bloquean el tráfico SMTP saliente y el POP3 e IMAP entrante. Suponiendo que los protocolos no estén filtrados se podría encontrar otro problema, el cliente no tiene los permisos para enviar correos utilizando el servidor SMTP del proveedor del servicio, ya que lo más seguro es que el servidor SMTP se encuentre configurado para no permitir el relay desde fuera de la organización.

Brindar el servicio de correo web significa contar con un acceso casi universal, la mayoría de los clientes poseen un navegador web y casi con seguridad el servicio HTTP disponible.

Con una aplicación de correo web los usuarios podrán enviar, recibir, leer y organizar sus mensajes desde cualquier red que permita el trafico HTTP. En esta sección se instalará y configurará la solución de correo web *Horde Groupware Webmail Edition*.

5.6.1 Horde Groupware Webmail Edition

Horde no es solamente una aplicación de correo Web, sino una gama de proyectos que en su conjunto forman una solución Groupware[24]. En si el sub proyecto IMP es el que brinda la solución de correo Web, siendo este el primero y más importante de todos.

Entre los elementos más importantes del Horde Groupware encontramos:

- *Horde:* se trata del Framework sobre el que trabajan el resto de aplicaciones.
- *IMP:* sistema de correo web que permite el acceso a los buzones.
- *Ingo:* sistema de gestión y aplicación de reglas de filtrado de correos.

[24] Groupware, se refiere a una solución de trabajo colaborativo, en la que las herramientas informáticas son aprovechadas para trabajar concurrentemente en un proyecto en común. Ejemplos de ellos son Lotus Notes, Microsoft Exchange Server y OpenXchange Server de Novel.

- *Sork:* conjunto de utilidades para el sistema de correo web que brinda funciones de cambio de claves, redirecciones, respuestas automáticas, etc.
- *Turba:* Libreta de direcciones.
- *Nag:* Administrador de tareas
- *Mnemo:* Administrador de notas.
- *Kronolith:* Gestion de agendas y calendarios, el corazón del sistema de trabajo en grupo.

Horde Groupware Webmail Edition es la aplicación que incorpora muchas de las características antes mencionadas de una solución de trabajo en grupo. Los usuarios podrán leer, enviar y gestionar los mensajes de correo electrónico; ademas gestionar y compartir calendarios, contactos, tareas y notas a través de los componentes IMP, Ingo, Kronolith, Turba, Nag y Mnemo.

Entre las características adicionales podemos encontrar:

- Cliente de correo web IMAP y POP3.
- Filtrado de mensajes.
- Búsqueda de mensajes.
- Composición de mensajes en HTML con editor WYSIWIG (ver lo que escribes).
- Corrector ortográfico.
- Archivos adjuntos.
- Soporte de cuotas de correo.
- Cambio de claves.

5.6.1.1 Instalación y configuración de Horde

Es muy probable que el software necesario para la instalación de Horde ya este disponible en el sistema, como sistema base debe tener instalado y corriendo un servidor LAMP. Para descargar el paquete puede hacerlo desde el sitio ftp o http de descargas de Horde o de la lista de sitios espejos, en cualquier caso ejecutar:

```
mail-ha1: # cd /srv/www/htdocs
fmail-ha1:/srv/www/htdocs # wget ftp://ftp.horde.org/pub/horde-webmail/horde-
webmail-1.1.1.tar.gz
```

Una vez obtenido el software debe descomprimirlo en el directorio raíz de documentos html de Apache2, por lo general ubicado en el direcotrio `/srv/www/htdocs/`:

```
mail-ha1:/srv/www/htdocs # tar xvzf horde-webmail-1.1.1.tar.gz
mail-ha1:/srv/www/htdocs # mv horde-webmail-1.0.6 horde
```

La configuración de Horde se facilita con un script PHP disponible en el directorio `/srv/www/htdocs/horde/scripts`:

```
mail-ha1:/srv/www/htdocs # ./horde/scripts/setup.php
What is the web root path on your web server for this installation, i.e. the
path of the address you use to access Horde Groupware Webmail Edition in your
browser? [/horde]
Configuration Menu
    (0) Exit
    (1) Configure database settings
    (2) Create database or tables
    (3) Configure administrator settings

Type your choice:
```

La ejecución del script anterior solicitará la ubicación del directorio web de la aplicación, a lo que deberá de dar enter. En el menú mostrado podrá configurar parámetros como:

- Base de datos utilizada.
- Parámetros de conexión con la base de datos (servidor, usuario, clave, nombre de base de datos).
- Parámetros de creación de base de tablas (usuario, clave de acceso).
- Creación de tablas o base de datos, recuerde que la base de datos utilizada es la creada en la configuración del administrador Web Web-cyradm (ver *sección 5.3*).
- Dirección de correo del administrador del servicio (cuenta POP3/IMAP creada previamente).

Horde incorpora un test de finalización de la instalación que nos informa si la instalación realizada ha sido exitosa, desde el navegador web ingresar el URL: *http://unan.edu.ni/horde/test.php*.

5.6.1.2 Inicio y prueba del correo Web

Horde como aplicación cliente del servidor LAMP, requiere del reinicio de los servicios Apache2 y MySQL:

```
mail-ha:~ # rcmysql restart
mail-ha:~ # rcmysql restart
```

No debe olvidar que Horde es también un cliente de los servicios de correo electrónico POP3, IMAP y SMTP, por lo tanto debe asegurarse de la ejecución de los proceso máster de Cyrus-IMAP y Postfix.

La *figura 5.3* muestra la interfaz de usuario del cliente de correo Web Horde.

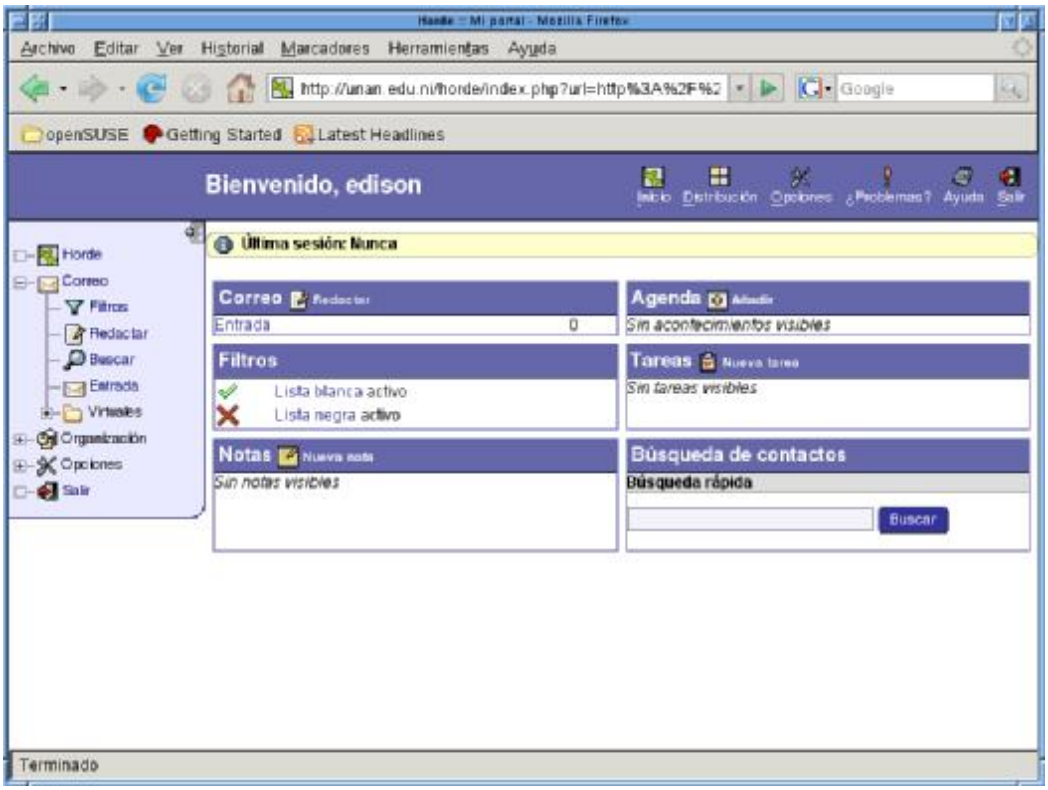


Figura 5.3 *Interfaz de correo web Horde.*

6. SOLUCIÓN DE CORREO ELECTRÓNICO DE ALTA DISPONIBILIDAD

6.1 Introducción a los Sistemas de Alta Disponibilidad

Linux es considerado como un sistema operativo estable; la problemática se genera cuando el hardware falla. En la mayoría de los casos cuando el sistema se cae es debido a un fallo del hardware o algún fallo humano (debido a un error en la administración del sistema).

Con el suficiente presupuesto, en sistemas altamente críticos pueden implementarse *sistemas tolerantes a fallos (foult tolerant ó FT*) en el cual el servicio siempre estará disponible. El problema de estas soluciones es que son altamente costosas. Además suelen ser soluciones cerradas, completamente dependientes de la marca del hardware adquirido. Por lo general estos consisten en un conjunto de equipos con diferentes funciones:

- Por lo menos un servidor tolerante a fallos con varias interfaces de red.
- Sistemas Ininterrumpidos de Energía (UPS) redundantes.
- Tomas especiales redundantes.
- Sistemas de climatización especial

Sin embargo cuando los recursos económicos no sobran, la solución son los *sistemas de alta disponibilidad (high availability ó HA)*. Con los *sistemas de alta disponibilidad* se intentan obtener prestaciones cercanas a las obtenidas en los sistemas tolerante a fallos, pero a un precio mucho más bajo.

La alta disponibilidad está basada en la replicación de elementos, mucho más baratos que un solo elemento tolerante a fallos. Evidentemente, si hablamos de replicar servidores, hablaremos de un *clúster*[25] de alta disponibilidad.

6.1.1 Sistemas de alta disponibilidad y sistemas tolerantes a fallos

En un sistema tolerante a fallos, cuando se produce un fallo hardware, el hardware asociado a este tipo de sistema es capaz de detectar el subsistema que falla y obrar en consecuencia para restablecer el servicio en segundos (o incluso décimas de segundo). El cliente del servicio no notará ningún tiempo de fuera de servicio. En los sistemas de alta disponibilidad existen los tiempos de fuera de servicio; son mínimos pero existen, van desde 1 minuto o menos hasta 5 o 10 minutos, según sea el caso. En teoría esta es la única diferencia entre ambos, pero en los últimos años, se ha ido acercando la idea de alta disponibilidad a la idea de tolerancia a fallos, debido al abaratamiento de hardware, y de ciertas tecnologías que han ido surgiendo. Estas tecnologías han evolucionado de tal forma, que han logrando que subsistemas donde había que recurrir a la alta disponibilidad ahora, se puede lograr tolerancia a fallos a bajo precio. De todos modos en un sistema (como veremos en la siguiente sección) hay muchos elementos y subsistemas, y algunos subsistemas tolerantes a fallos siguen siendo demasiados caros.

En la mayoría de los análisis, que se hacen de un sistema de servicio, si la aplicación puede estar un mínimo tiempo fuera de servicio, y podemos permitir que el cliente pierda la sesión o la conexión, temporalmente, la alta disponibilidad es una opción muy apropiada. Hay soluciones de alta disponibilidad en las cuales las conexiones se mantienen y las sesiones se recuperan.

[25] Un clúster de alta disponibilidad es un conjunto de dos o más nodos (equipos de cómputo) que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizados entre sí.

6.1.2 Dinámica de alta disponibilidad

Se considera dinámica alta disponibilidad (dinámica HA) a todas las reconfiguraciones del clúster que garanticen la máxima disponibilidad del servicio de datos. Esta dinámica está orientada a los nodos integrantes del clúster y la forma en la cual el clúster responde. La dinámica está definida en los siguientes términos:

- ***Failover:*** Es un término genérico que se usa cuando un nodo debe asumir la responsabilidad de otro nodo, importar sus recursos y levantar el servicio de datos. Se ha de entender que una situación de *failover* es una situación excepcional, para cual la alta disponibilidad ha sido concebida, el fallo de un nodo. Si sólo queda un nodo en el clúster, tras los fallos de los demás, estaremos en un SPOF[26] hasta que el administrador del sistema, verifique y restaure el clúster. También se ha de entender que el servicio de datos sigue levantado, que es el objetivo de la alta disponibilidad.

- ***Takeover:*** Es un *failover* automático se produce cuando un nodo nota un fallo en el servicio de datos. Para ello debe haber cierta monitorización con respecto al servicio de datos. El nodo que se declara fallido es forzado a ceder el servicio y recursos, o simplemente eliminado.

- ***Switchover o Giveaway:*** Es un *failover* manual, consiste en ceder los recursos de un servicio de datos y este mismo, a otro nodo del clúster, mientras se realizan ciertas tareas administrativas. A este procedimiento se le denomina “Node outage”.

- ***Splitbrain:*** Para la gestión de un clúster de alta disponibilidad es necesario un mecanismo de comunicación y verificación entre nodos integrantes. Por este

[26] SPOF (Single Point Of Failure ó punto simple de fallo) es un término utilizado para hacer referencia a cualquier elemento no replicado y que puede estar sujeto a fallos; afectando con ello el servicio. Se debe evitar el SPOF en cualquier subsistema dentro de un sistema de alta disponibilidad.

mecanismo, cada nodo debe gestionar los recursos que corresponden a cada uno, según el estado del clúster; a su vez cada nodo debe hacer chequeos o latidos (beats) a sus compañeros.

Un *Splitbrain* (división de cerebros) es un caso especial de *failover*, en el cual falla el mecanismo de comunicación y gestión de un clúster de dos nodos. Es una situación en la cual cada nodo cree que es el único activo, y como no puede saber el estado de su nodo compañero, tomará acciones en consecuencia, forzando un *takeover*.

Esta situación es peligrosa, los dos nodos intentaran apropiarse de todos los recursos, incluyendo el servicio de datos. El peligro aumenta sobre todo cuando tenemos recursos delicados, como recursos de almacenamiento, ya que cada nodo podría tomar y escribir por su cuenta, y quebrar la integridad de los datos.

Para evitar este problema, cada nodo debe actuar de una forma prudente, y utilizar los recursos compartidos como señal de que se está vivo. La forma de proceder de cada nodo es similar, ya que cada uno cree que su compañero ha desaparecido. Después de que un nodo aprecia este problema, tiene indicado que reserve un recurso llamado *quorum*. Un recurso *quorum*, es un recurso compartido, que se ha preestablecido en ambos nodos como tal. Este recurso es un recurso exclusivo, sólo un nodo del clúster puede reservarlo. Como este recurso sólo puede ser reservado por un nodo, el nodo que llegue tarde a la reserva del recurso, entiende que debe abandonar el clúster y ceder todos sus recursos. El quórum es utilizado como simplemente, método de decisión.

Otra forma de evitar esta situación, un poco más violenta, es que un nodo elimine a su compañero; el primero que apague a su compañero se queda con todos los recursos. Es un mecanismo muy brusco, pero muy eficaz. Para este caso existen tarjetas especializadas que realizan esta tarea.

6.2 Alta Disponibilidad en Linux

Linux como sistema operativo debe de ofrecer una serie de facilidades, para que los sistemas de alta disponibilidad puedan integrarse correctamente tal y como se hace en otros entornos. Estas facilidades están relacionados directamente con el entorno hardware y el software para las aplicaciones de alta disponibilidad. En esta sección se comentaran los diversos subsistemas que ayudan a "virtualizar" recursos, así como subsistemas para evitar SPOF.

6.2.1 Sistemas de computación

Se puede afrontar la idea de eliminar SPOF en los sistemas de computación, pero sólo a partir de un nivel que Linux como sistema operativo pueda obrar. Es decir que si consideramos la computación al nivel de nodos y superiores no habrá problemas, pero si consideramos a nivel de procesadores y memorias, es más un problema muy cercano a una solución hardware tolerante a fallos.

6.2.2 Sistemas de red

Linux puede presumir de tener una de las pilas TCP/IP más completas y estables que existen actualmente. La característica de IP Aliasing de Linux, permite asignar varias direcciones IP a una misma interfaz; esto nos permite poder levantar una dirección IP, exclusivamente, en un nodo del clúster. Si algún nodo perece con su IP, cualquier otro nodo del clúster, puede tomar el testigo.

También, Linux es capaz de actualizar sus tablas de rutas dinámicamente, gracias a demonios de enrutamiento tales como ospfd. ospfd es un demonio de enrutamiento que implementa el algoritmo de enrutamiento OSPF. Con estos servicios el clúster puede ser consciente de la topología de la red y reaccionar ante las caídas de líneas de comunicaciones o nodos.

6.2.3 Sistemas de de almacenamiento

En los sistemas de almacenamiento existe una doble problemática

1. SPOF en los elementos de almacenamiento
2. Consistencia ante un crash o caída.

El SPOF se soluciona con sistemas de discos capaces de hacer redundantes los datos, así como de sustituir un disco dañado por otro en reserva.

La consistencia se consigue revisando la integridad de los datos; la consistencia debe ser establecida en el mínimo tiempo posible ya que el servicio de datos depende de él.

6.2.3.1 Sistemas redundantes de disco

Este es un tema que ya fue tratado en la *sección 4.2*, ahí se explicaban los niveles RAID más convenientes para un sistema de correo electrónico. Con RAID el SPOF en los recursos de almacenamiento puede ser eliminado.

Actualmente el kernel de Linux soporta RAID 0, 1 y 5 por software a través del driver MD, además soporta gran número de controladoras SCSI y ATA100 que ofrecen volúmenes de RAID por hardware. Compaq e IBM han colaborado mucho en este aspecto, creando drivers para sus productos, es mas IBM en alianza con novel ha desarrollado una serie de servidores bajo el nombre de OpenPower (basada en Power5) expresamente diseñado para correr el sistema operativo SuSE Linux. También cabe mencionar los sistemas LAN-mirror que son una opción barata para conseguir un medio compartido y eliminación de SPOF replicando por red. Como ejemplos cabe destacar NBD, ENBD y DRBD.

6.2.3.2 Sistemas de ficheros con journal

Journaling es una técnica que nació de las bases de datos y se ha ido incorporando a los sistemas de ficheros. Cuando un sistema de ficheros sufre una caída, dado a un fallo del sistema, este se chequea al completo corrigiendo las inconsistencias. Con journal se lleva

una cuenta de que se ha ido modificando en el sistema de ficheros, ya que a la hora de chequearlo sólo comprobará las inconsistencias de unos pocos ficheros y directorios.
El tiempo de puesta en consistencia de un sistema de ficheros disminuye considerablemente. Además se incorporan técnicas como árboles B y Hashes para un acceso más rápido a ficheros.

A continuación se enumeran los sistemas de ficheros mas familiares y apropiados para la alta disponibilidad en Linux:

- ***ext3:*** Es el clásico ext2 pero con el añadido de una partición de log, para llevar el journal. Por lógica es algo más lento que ext2 en acceso pero más rápido en tiempo de recuperación de consistencia. Lo realmente interesante es que hace journal de datos y metadatos.

- ***reiserfs:*** Es un sistema de ficheros creado desde 0 con la idea de sacar partido a arboles B, hashes y técnicas de journal actuales. Es un sistema de ficheros realmente rápido en lo que a Linux se refiere. Sólo hace journaling de datos. Actualmente sus creadores están desarrollando Reiser4.

- ***XFS:*** Silicon ha sacado releases con calidad de producción bajo GPL. Casi tan rápido como *reiserfs*. Esta muy bien integrado con Linux, teniendo en consideración otros subsistemas como puede ser quota o NFS. Sacan parches cada una o dos versiones del kernel. Una opción muy a considerar, en un futuro.

6.3 Solución de Alta Disponibilidad para Correo Electrónico

El sistema de alta disponibilidad propuesto está compuesto por dos nodos, ambos con características similares, de manera general podemos decir que se cuenta con los siguientes recursos:

- 2 Servidores con Linux corriendo servicios de correo electrónico
- Sistema de almacenamiento de datos redundante con RAID1
- Sistema de ficheros con Jounaling
- Paquete Heartbeat, como servicio de alta disponibilidad
- Paquete DRBD, para replica de discos en red
- Red Ethernet de alta velocidad (FastEthernet)

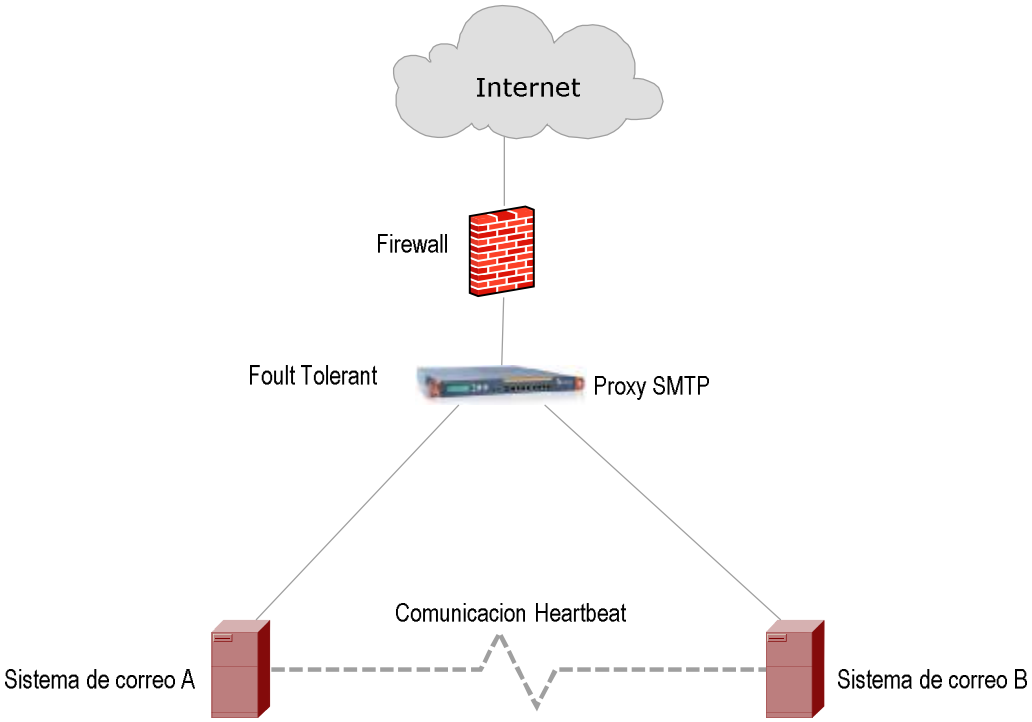


Figura 6.1 *Diseño de alta disponibilidad con Heartbeat*

La infraestructura de correo se ejecuta en 2 Servidores IBM OpenPower que forman los nodos A y B (ver figura 6.1) en modo Activo/Pasivo con una conexión fastethernet para replica de datos y comunicación serial a través del puerto RS-232 y un Apliance Astaro Security Gateway como Proxy SMTP para filtrado de Spam y Virus.

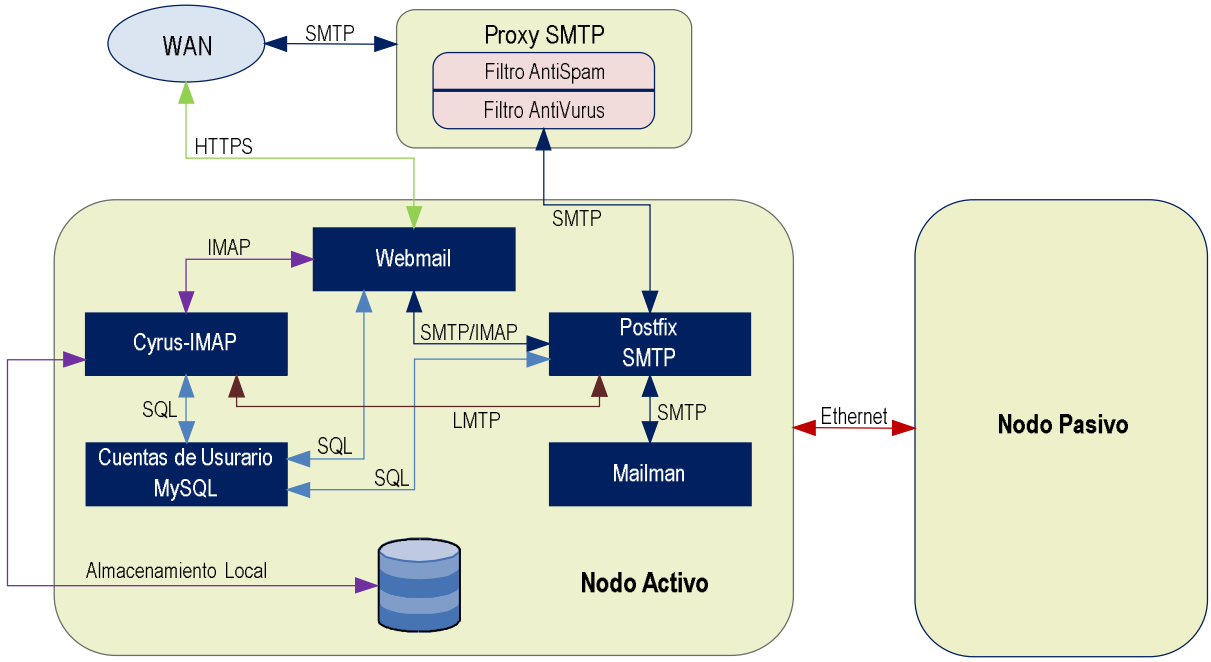


Figura 6.2 *Arquitectura de alta disponibilidad con replicación de nodo*

La complejidad de un sistema de correo electrónico lo convierte en un sistema altamente crítico, más aún cuando un solo equipo hardware aloja todas las aplicaciones implicadas, recuerde que no es solo la ejecución de un solo servicio atendiendo todas las conexiones recibidas, sino más bien son un conjunto de servicios que incluye:

- Servicios SMTP
- Servicios POP/IMAP
- Servicios de autenticación de usuarios
- Servicios de nombres de dominio
- Bases de datos
- Servicios HTTP

La caída de un servicio será motivo suficiente como para declarar el nodo como no disponible y así ceder los recursos al segundo nodo. El tiempo en levantar el o los servicios caídos dependerá de las destrezas del administrador, aunque lo común es que

sean periodos de tiempo no muy altos comparados a los vistos en el escenario descrito a continuación.

Hemos visto el problema desde el punto de vista del software, pero que pasaría en caso de fallo del hardware, ya sea por defectos del mismo o por mala manipulación humana. En esta situación el nodo completo se vendría abajo o por lo menos no funcionaria del todo bien, lo cual es una situación indeseable. El sistema de alta disponibilidad otorga los recursos al segundo nodo, mientras tanto el administrador manualmente tendrá que recuperar el primer nodo instalando un equipo adicional. En este momento el sistema es vulnerable, solo un nodo está funcionando y es el único disponible, ¿qué pasaría si este falla? lógicamente el servicio quedará fuera completamente. Estas son situaciones que deben ser tomadas en cuenta, la recuperación ante un fallo de hardware es mas critica que la recuperación ante un fallo de software, es por eso que no toda la responsabilidad debe caer en la solución de alta disponibilidad, siempre debe contarse con la cantidad y calidad de hardware que cumpla con las necesidades mínimas.

6.3.1 Funcionamiento

Un clúster de alta disponibilidad de dos nodos, requiere:

- Replicación de discos a nivel de nodo.
- Replicación de discos a nivel de clúster.
- Y monitoreo y gestión del clúster.

La replicación de discos a nivel de nodo la conseguimos con un espejo de discos (RAID 1), los datos escritos en un disco se copian en el segundo disco. DRBD es el software utilizado para la réplica a nivel de clúster, funciona de manera similar a un RAID1, sin embargo la copia en vez de ser local, es a través de la red, el dispositivo de bloques `/dev/sdc1` del nodo primario se replica en `/dev/sdc1` del nodo secundario. En la ***figura 6.3*** puede ver los tres discos utilizados en cada nodo, las replicas locales y las replicas en red.

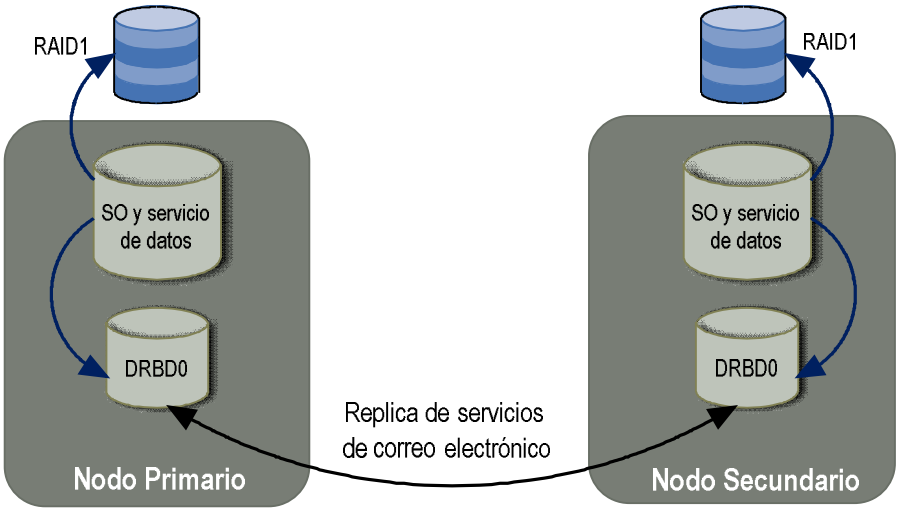


Figura 6.3 *Replicas locales y en red de servicio de datos*

La administración del clúster es llevada a cabo a través de heartbeat, cuando este detecta la muerte de un nodo, primero se asegura de ello y luego le concede los recursos al segundo nodo asumiendo la responsabilidad del clúster.

Se utilizaran dos direcciones IP por servidor, en subredes distintas, en la subred 10.1.120.0/24 se atienden los servicios de correo electrónico y en la subred 192.168.1.0/24 se gestiona el clúster de alta disponibilidad. La *tabla 6.1* detalla el direccionamiento utilizado:

Servidor	Hostname	IP de servicios	IP de clúster
Primario	mail-ha1	10.1.120.3	192.168.1.1
Secundario	mail-ha2	10.1.120.3	192.168.1.2

Tabla 6.1 *Direccionamiento del clúster de alta disponibilidad*

En un modo de funcionamiento maestro/esclavo, el servidor primario mientras tenga el control del clúster tendrá acceso pleno de lectura/escritura a su partición /dev/sdc1, mientras que el servidor secundario no tendrá ningún permiso sobre esta, sin embargo

DRBD estará replicando los datos de la partición /dev/sdc1 del servidor primario en /dev/sdc1 del servidor secundario. En la *tabla 6.2* se aclara esta situación:

Control del clúster	Servidor	/dev/sdc1 primario	/dev/sdc1 secundario	Descripción
Servidor primario	Servidor primario	lectura/escritura	lectura/escritura	El servidor primario tiene el control del clúster, con acceso de lectura/escritura a su partición /dev/sdc1, mientras tanto DRBD replica los datos en la partición /dev/sdc1 del servidor secundario.
	Servidor secundario	Ninguno	Ninguno	El servidor primario tiene el control de clúster, por lo que el servidor secundario no tiene permisos sobre ninguno de los nodos.
Servidor secundario	Servidor primario	Ninguno	Ninguno	El servidor secundario tiene el control del clúster, por lo que el servidor primario no tiene permisos sobre ninguno de los nodos.
	Servidor secundario	lectura/escritura	lectura/escritura	El servidor secundario tiene el control del clúster, con acceso de lectura/escritura a su partición /dev/sdc1. DRBD replica los datos en la partición /dev/sdc1 del primario.

Tabla 6.2 *permisos DRBD sobre dispositivos /dev/sdc1*

Todos los procesos que necesitan acceder a la partición replicada deben correr en el servidor que tiene el control del clúster en ese momento. Si el servidor primario falla, el demonio Hearbeat que corre sobre el servidor secundario le dice a DRBD que él ahora

tiene el control del clúster, por lo que tiene que montar la partición replicada y después iniciar todos los servicios configurados sobre esta.

Aunque se ha considerado que ambas particiones (en los servidores primario y secundario) tienen el mismo tamaño, se aclara que esto no es un requerimiento, bien podría tratarse de particiones distintas, sin embargo los servicios (acceso a buzones, bases de datos, web, etc.) cargados en la partición de mayor tamaño no solo deben de alcanzar en la partición replicada, sino que el espacio reservado debe ser lo suficiente como para evitarse molestias de redimensionamiento de particiones. Por ejemplo el servicio de acceso a buzones POP/IMAP requiere una partición de gran tamaño, es común que los usuarios dejen copia de los correos en el servidor, lo que incrementará constantemente el espacio ocupado en disco.

Es muy importante tener bien claro que la versión de los demonios debe ser la misma en ambos servidores, lo más indicado y fácil seria instalar la misma distribución de Linux en los dos nodos. Igualmente es recomendable que las capacidades del hardware sean iguales, las diferencias de rendimiento de un nodo con respecto al otro deben ser despreciables.

6.3.2 Clúster de Alta Disponibilidad con Heartbeat

Heartbeat es un Software creado por LINUX-HA, con funcionamiento similar al conocido init System V, pero en vez de ejecutarse en una sola maquina pasaría a ejecutar los servicios en los nodos, basándose en que no le llegan respuestas (prueba de echo request con ping) y por pulsaciones del cable serie.

Heartbeat utiliza STONITH (Shoot The Other Node in The Head). STONITH es una técnica que consiste en asegurarse de que un nodo supuestamente muerto no interfiera con el funcionamiento del clúster, en caso de no implementarse bien esta técnica, podría dar lugar a que el clúster no funcione. Básicamente STONITH se basa en la mera

especulación de creer que un nodo esta muerto y asegurarse de ello utilizando recursos para matarlo, entonces el nodo secundario se apodera del control de los recursos tomando el clúster

6.3.2.1 Replica de Servidores con DRBD

DRBD es un dispositivo de bloques que está diseñado para construir *clústeres de alta disponibilidad*. Esto es hecho por medio de una copia idéntica de todo un dispositivo de bloques a través de una red (dedicada). Como se mencionó anteriormente esto podría ser visto como un RAID-1 de red.

DRBD toma los datos, los escribe en el disco local y los envía al otro servidor, entonces este se encarga de escribir los datos en su disco. Un clúster de alta disponibilidad utiliza DRBD como complemento a un software que brinde pertenencia al clúster (heartbeat) y algún tipo de aplicación que trabaje a un nivel más alto que un dispositivo de bloques, por ejemplo:

- Un sistema de archivos y la herramienta fsck.
- Un sistema de archivos con journaling
- Una base de datos con capacidades de recuperación

Cada dispositivo (DRBD provee más de uno de estos dispositivos) tiene un estado, que puede ser *primario* o *secundario*. En el nodo en el que se encuentra el dispositivo primario la aplicación debe correr y acceder al dispositivo (`/dev/drbd0`). Cada operación de escritura es enviada al dispositivo de bloques de más bajo nivel local y al nodo con el dispositivo en estado secundario. El dispositivo secundario simplemente escribe los datos en su dispositivo de bloques de más bajo nivel. Las operaciones de lectura son llevadas a cabo siempre localmente.

Si el nodo primario falla, heartbeat pasa el dispositivo secundario al estado primario e inicia la aplicación ahí. Si se utilizara DRBD con un sistema de archivos sin *journaling* esto implicaría correr la herramienta `fsck`, cada vez que el sistema falle.
Si el nodo que ha fallado se restablece, entra a funcionar como un nuevo nodo secundario y tiene que sincronizar su contenido con el nodo primario. Esto, por supuesto, debería ocurrir sin provocar ninguna interrupción del servicio.

Para instalar DRBD, puede hacerlo desde la herramienta de administración grafica YAST, obtener el paquete del algún repositorio de SuSE o instalarlo manualmente o a través de smart. En cualquier caso se obtendrá un único archivo de configuración (`/etc/drbd.conf`) y el script de arranque del servicio ubicado en `/etc/init.d/drbd`.

Antes de proceder con DRBD recordamos que se cuenta con la estructura de discos mostrada en la *tabla 6.3*:

Disco	Partición	Tipo	Descripción
sda	sda1	Boot	Partición de arranque
	sda2	Linux RAID1	Dispositivo que forma el arreglo redundante (espejo de discos). Aquí se encuentran los datos y servicios del sistema.
sdb	sdb1	Linux SWAP	Partición de intercambio de memoria.
	sdb2	Linux RAID1	Arreglo redundante (espejo de discos) con sda2. Copia de los datos y servicios del sistema local.
sdc	sdc1	drbd	Disco para copia de datos en red a través de DRBD.

Tabla 6.3 *Estructura completa de sistemas de discos*

El archivo `/etc/drbd.conf` debe ser el mismo en ambos nodos, la configuración propuesta es la siguiente:

```
resource "drbd0" {
  protocol C;
  handlers {
    pri-on-incon-degr "echo 'DRBD: primary requested but inconsistent!.' |
wall; /etc/init.d/heartbeat stop"; #"halt -f";
    pri-lost-after-sb "echo 'DRBD: primary requested but lost !' | wall;
/etc/init.d/heartbeat stop"; #"halt -f";
  }

  startup {
    degr-wfc-timeout 120;
  }

  disk {
    on-io-error detach;
  }

  net {
    timeout     60;
    connect-int 10;
    ping-int    10;
    max-buffers 2048;
    max-epoch-size 2048;
    ko-count 30;
    cram-hmac-alg "shal";
    shared-secret "FooFunFactory";
  }

  syncer {
    rate 10M;
    al-extents 257;
  }

  on mail-ha1 {
    device      /dev/drbd0;
    disk        /dev/sda2;
    address     192.168.1.1:7789;
    meta-disk   internal;
  }

  on mail-ha2 {
    device     /dev/drbd0;
    disk       /dev/sda2;
    address    192.168.1.2:7789;
    meta-disk   internal;
  }
}
```

El archivo puede estar compuesto por una o más secciones de recursos (`resource`) que definen el disco a ser replicado. `drbd0` es el nombre que hemos utilizado para referirnos a este dispositivo. Dentro de la sección de recursos `resource drbd0`, se encuentran los siguientes bloques:

- `net:` incluye las configuraciones de red más convenientes para el buen funcionamiento del clúster, intervalos en segundos de pruebas ping y timeout por ejemplo.
- `syncer:` limites de ancho de banda utilizados para re sincronización de los procesos, en una red Fast Ethernet el valor limite seria 12.5M, que equivale a una taza de transferencia total de 100Mbits.
- `on mail-ha1 | on mail-ha2`: en estos bloques se definen las configuraciones para los servidores etiquetados como mail-ha1 y mail-ha2, nombres que deberían estar establecidos en el archivo `/etc/hosts`. El parámetro `device` a través del modulo drbd del kernel crea la partición `/dev/drbd0`, cuando el nodo se encuentra en modo maestro. `disk` establece el nombre del dispositivo, previamente preparado. Con `address` se establecen las direcciones IPs 192.168.1.1 y 192.168.1.2 de los servidores primario y secundario respectivamente.

A continuación se arranca el servicio DRBD, primero en el servidor maestro y después en el esclavo.

```
mail-ha2:~ # modprobe drbd
mail-ha2:~ # rcdrbd start
```

Con esto se creará el dispositivo en bloques para hacer las replicas, cuando ejecute el comando en el servidor maestro el script de arranque indica que se inicie también el servicio DRBD en el servidor secundario.

En el servidor primario ejecutar el comando de administración de consola `drbdadm`, con los parámetros que le indiquen a este que es el servidor maestro.

```
mail-ha1:~ # drbdadm -- --overwrite-data-of-peer primary drbd0
```

Posterior debería de ver una salida como esta en el servidor primario:

```
mail-ha1:~ # rcdrbd status
drbd driver loaded OK; device status:
version: 8.0.6 (api:86/proto:86)
SVN Revision: 3048 build by phil@mescal, 2007-09-03 10:39:27
 0: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
    ns:0 nr:2929556 dw:2929556 dr:0 al:0 bm:179 lo:0 pe:0 ua:0 ap:0
        resync: used:0/31 hits:182919 misses:179 starving:0 dirty:0
changed:179
        act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Mientras que en el servidor secundario la salida seria:

```
mail-ha2:~ # rcdrbd status
drbd driver loaded OK; device status:
version: 8.0.6 (api:86/proto:86)
SVN Revision: 3048 build by phil@mescal, 2007-09-03 10:39:27
 0: cs:Connected st:Secondary/Primary ds:UpToDate/UpToDate C r---
    ns:2929556 nr:0 dw:0 dr:2929556 al:0 bm:179 lo:0 pe:0 ua:0 ap:0
        resync: used:0/31 hits:182919 misses:179 starving:0 dirty:0
changed:179
        act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Lo mostrado en ambos servidores significa que está siendo llevada a cabo una sincronización completa del dispositivo de bloques `/dev/drbd0` del servidor primario al dispositivo `/dev/drbd0` del servidor secundario.

En el servidor maestro crear el sistema de archivos. En esta etapa no es necesario crear también el sistema de archivos en el dispositivo del servidor secundario, ya que cuando se inicie DRBD se hará un espejo de los datos desde el dispositivo maestro al esclavo. También a de montarse manualmente el dispositivo en bloques en el directorio `/replica`:

```
mail-ha1:~ # mkdir /replica
mail-ha1:~ # mkfs.ext3  /dev/drbd0
mail-ha1:~ # mount /dev/drbd0 /replica
```

Las particiones no deben ser montadas automáticamente (en ambos servidores) por el sistema por lo que agregamos al `/etc/fstab` la siguiente línea:

```
/dev/drbd0            /replica              ext3       noauto       0       0
```

6.3.2.2 Replicando servicios:

Para replicar el servidor de correo electrónico debe conocer cuáles son los directorios de datos de los servicios, crear en el directorio `/replica` el conjunto de directorios correspondientes a cada servicio y posteriormente crear un enlace simbólico a los nuevos directorios desde los originales de instalación.

Antes nos aseguraremos que los servicios postfix, saslauthd, cyrus, mysql, apache2, amavis, mailman y named y no sean ejecutados por el init System V.

```
mail-ha1:~ # chkconfig --del postfix saslauthd cyrus mysql apache2 amavis
mailman named
postfix                  0:off  1:off  2:off  3:off  4:off  5:off  6:off
saslauthd                0:off  1:off  2:off  3:off  4:off  5:off  6:off
cyrus                    0:off  1:off  2:off  3:off  4:off  5:off  6:off
mysql                    0:off  1:off  2:off  3:off  4:off  5:off  6:off
apache2                  0:off  1:off  2:off  3:off  4:off  5:off  6:off
amavis                   0:off  1:off  2:off  3:off  4:off  5:off  6:off
mailman                  0:off  1:off  2:off  3:off  4:off  5:off  6:off
named                    0:off  1:off  2:off  3:off  4:off  5:off  6:off
```

Para Postfix se replican los directorios `/etc/postfix` y `/var/spool/postfix`:

```
mail-ha1:/ # mkdir /replica/etc
mail-ha1:/ # mkdir /replica/var /replica/var/spool
mail-ha1:/ # cp -rfp /etc/postfix/ /replica/etc
mail-ha1:/ # cp -rfp /var/spool/postfix /replica/var/spool
mail-ha1:/ # rm -rf /etc/postfix/
mail-ha1:/ # rm -rf /var/spool/postfix/
mail-ha1:/ # ln -s /replica/etc/postfix/ /etc/postfix
mail-ha1:/ # ln -s /replica/var/spool/postfix/ /var/spool/postfix
```

Replicar Cyrus-IMAP requiere mayor atención que Postfix, debe ser cuidadoso en conservar los permisos de los buzones de usuarios, a veces un simple permiso sobre un directorio o un archivo nos puede acarrear problemas (troubleshoting) cuyas detecciones no son del todo intuitivas.

```
mail-ha1:/ # mkdir /replica/var/lib
mail-ha1:/ # cp /etc/cyrus.conf /replica/etc/
mail-ha1:/ # cp /etc/imapd.conf /replica/etc/
mail-ha1:/ # cp -rfp /var/lib/sieve/ /replica/var/lib/
mail-ha1:/ # cp -rfp /var/lib/imap/ /replica/var/lib/
mail-ha1:/ # cp -rfp /var/spool/imap/ /replica/var/spool/
mail-ha1:/ # rm /etc/cyrus.conf /etc/imapd.conf
mail-ha1:/ # rm -rf /var/lib/sieve/
mail-ha1:/ # rm -rf /var/lib/imap/
mail-ha1:/ # rm -rf /var/spool/imap/
mail-ha1:/ # ln -s /replica/etc/cyrus.conf /etc/cyrus.conf
mail-ha1:/ # ln -s /replica/etc/imapd.conf /etc/imapd.conf
mail-ha1:/ # ln -s /replica/var/lib/sieve/ /var/lib/
mail-ha1:/ # ln -s /replica/var/lib/imap/ /var/lib/
mail-ha1:/ # ln -s /replica/var/spool/imap/ /var/spool/
```

Igual atención en cuanto a los permisos de directorios requiere MySQL:

```
mail-ha1:/ # cp /etc/my.cnf /replica/etc/
mail-ha1:/ # cp /etc/mysqlaccess.conf /replica/etc/
mail-ha1:/ # cp -rfp /var/lib/mysql/ /replica/var/lib/
mail-ha1:/ # rm /etc/my.cnf
mail-ha1:/ # rm /etc/mysqlaccess.conf
mail-ha1:/ # rm -rf /var/lib/mysql/
mail-ha1:/ # ln -s /replica/etc/my.cnf /etc/
mail-ha1:/ # ln -s /replica/etc/mysqlaccess.conf /etc/
mail-ha1:/ # ln -s /replica/var/lib/mysql/ /var/lib/
```

Para el servidor Web se replican sus archivos de configuración principal y el `DocumenRoot`, directorio donde Apache2 sirve los documentos web.

```
mail-ha1:/ # cp -rfp /etc/apache2/ /replica/etc/
mail-ha1:/ # cp /etc/mime.types /replica/etc/
mail-ha1:/ # cp -rfp /var/lib/apache2/ /replica/var/lib/
mail-ha1:/ # rm -rf /etc/apache2/
mail-ha1:/ # rm /etc/mime.types
mail-ha1:/ # rm -rf /var/lib/apache2/
mail-ha1:/ # ln -s /replica/etc/apache2/ /etc/
```

```
mail-ha1:/ # ln -s /replica/etc/mime.types /etc/
mail-ha1:/ # ln -s /replica/var/lib/apache2/ /var/lib/
```

El filtro de correo Amavisd-new mantendrá una réplica de los virus y correo spam detectados, aunque no es una mala idea deshacernos de ellos, a veces es bueno conservarlos con fines estadísticos para su posterior estudio, además este es un servicio que debe estar en constante mantenimiento para su mejora.

```
mail-ha1:/ # cp /etc/amavisd.conf /replica/etc/
mail-ha1:/ # cp -rfp /var/spool/amavis/ /replica/var/spool/
mail-ha1:/ # rm /etc/amavisd.conf
mail-ha1:/ # rm -rf /var/spool/amavis/
mail-ha1:/ # ln -s /replica/etc/amavisd.conf /etc/
mail-ha1:/ # ln -s /replica/var/spool/amavis/ /var/spool/
```

Para el servidor de listas de correo mailman los directorios a replicar son `/etc/mailman/` y `/var/lib/Mailman`:

```
mail-ha1:/ # cp -rfp /etc/mailman/ /replica/etc/
mail-ha1:/ # cp -rfp /var/lib/mailman/ /replica/var/lib/
mail-ha1:/ # rm -rf /etc/mailman/
mail-ha1:/ # rm -rf /var/lib/mailman/
mail-ha1:/ # ln -s /replica/etc/mailman/ /etc/
mail-ha1:/ # ln -s /replica/var/lib/mailman/ /var/lib/
```

Aunque la réplica del servicio de nombres de dominio debería ser opcional, como se mencionó antes este es un servicio que para fines de este proyecto ha sido implementado de manera conjunta (mismo sevidor) con los servicios de correo electrónico. Estrictamente para BIND se replican los directorios de archivos de zona `/var/lib/named` y el archivo de configuración principal `/etc/named.conf`:

```
mail-ha1:/ # cp /etc/named.conf /replica/etc/
mail-ha1:/ # cp -rfp /var/lib/named/ /replica/var/lib/
mail-ha1:/ # rm /etc/named.conf
mail-ha1:/ # rm -rf /var/lib/named/
mail-ha1:/ # ln -s /replica/etc/named.conf /etc/
mail-ha1:/ # ln -s /replica/var/lib/named/ /var/lib/
```

En la *figura 6.4* se resume el árbol de directorios replicados.

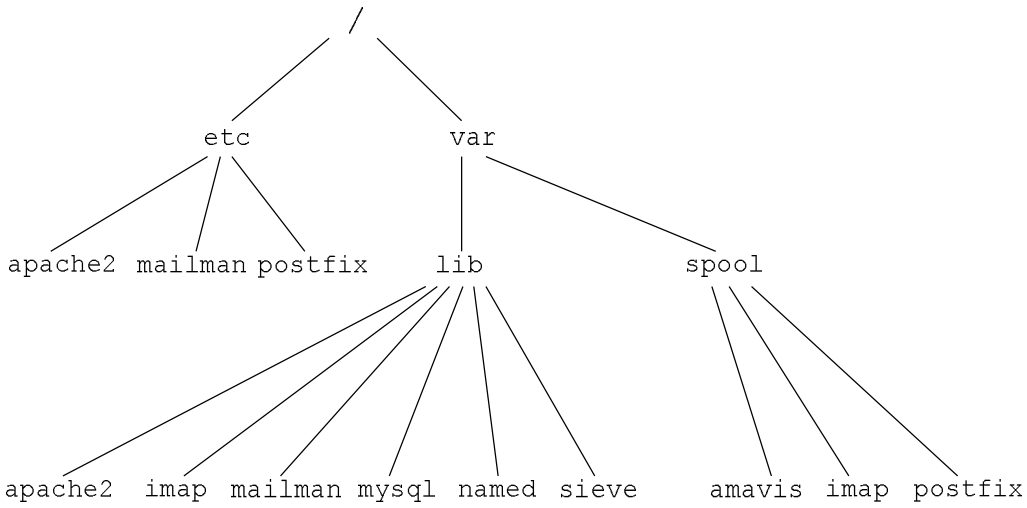


Figura 6.4 *Árbol de directorios replicados*

Al árbol anterior habría que agregar los archivos de configuración de los servicios.

6.3.2.3 Configurando Heartbeat

La función de heartbeat será controlar los servidores, si el servidor primario falla, iniciará los servicios sobre el servidor secundario, lo que lo convertiría en el servidor maestro.

Para configurarlo se le especifica que servidores debe controlar y los servicios que debe iniciar cuando un servidor muere.

Su configuración se divide en los archivos:

- `/etc/ha.d/resource.d`: contiene todos los scripts de arranque de los servicios que heartbeat iniciará. Aquí solo se crean enlaces simbólicos a cada uno de los scripts de arranque de los servicios:

```
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/postfix postfix
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/saslautdh
saslauthd
```

```
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/cyrus cyrus
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/amavis amavis
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/apache2 apache2
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/mysql mysql
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/mailman mailman
mail-ha1:/etc/ha.d/resource.d # ln -s /etc/init.d/named named
```

- `/etc/ha.d/haresources`: archivo de configuración de recursos manejados por heartbeat.

```
mail-ha1 IPaddr2::10.1.120.3/255.255.255.0/eth0:0/10.1.120.255
drbddisk::drbd0 postfix saslauthd cyrus amavis mysql mailman
apache2 named
```

En el primer campo se define cual es el servidor maestro por defecto de estos servicios, con en el segundo (`IPaddr::10.1.120.3/24`) heartbeat arrancará el servidor maestro (IP alias) con la dirección 10.1.120.3 y mascara de subred 255.255.255.0, esta IP será aquella apuntada en el registro `"A"` para el dominio unan.edu.ni. Con `drbddisk::drbd0` le decimos a heartbeat que monte este disco automáticamente sobre el servidor primario, a continuación de este, podemos indicar cada uno de los servicios que serán iniciados y controlados por heartbeat en el nodo local.

Note que el nombre de los servicios coincide con el nombre de los scripts de arranque ubicados en `/etc/init.d/`.

- `/etc/ha.d/ha.conf`: este es el archivo de configuración principal de heartbeat. Los parámetros más importantes a notar aquí son:

  `node`: especifica el nombre de los nodos primario y secundario.

  `deadtime` e `initdead`: estos especifican cuanto tiempo en segundos debería esperar heartbeat al otro nodo, antes de asumir que realmente está muerto y entonces tomar el control del nodo. Un tiempo muy bajo daría

lugar a muchos falsos positivos, lo mejor es siempre hacer pruebas de todo tipo. Este archivo es el mismo en ambos nodos.

```
debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 120
warntime 10
initdead 120
udpport 694
#baud   19200
bcast  eth0            # Linux
auto_failback on
node mail-ha1
node mail-ha2
##IPFAIL
ping 10.1.120.1
respawn hacluster /usr/lib/heartbeat/ipfail
```

Con `auto_failback` se le concede al servidor primario siempre la prioridad de manejar el cluster, es decir si en un momento dado este falla, y los recursos son tomados por el nodo secundario, una vez que el servidor primario es levantado volverá a tomar los recursos del cluster.

Heartbeat puede hacer uso también del Puerto serie conectando con un cable serial cruzado ambos servidores, sin embargo en esta configuración la información de gestión será manejada por broadcast a través de la subred de alta disponibilidad (parámetro `bcast eth0`).

Con ipfail podemos monitorear constantemente la coneccion del servidor virtual (IP alias), siempre es bueno seleccionar una IP a la que el modulo ipfail estará haciendo ping, que se garantice que sea un equipo no se apague accidentalmente.

Finalmente se Inicia heartbeat en ambos servidores. Para probar la funcionalidad de ambos nodos basta con bajar el primario, ver que el secundario toma el control del nodo o viceversa.

```
mail-ha1:/ # rcheartbeat start
```

7. SISTEMAS DE CORREO ELECTRÓNICO CON ESCALABILIDAD Y ALTA DISPONIBILIDAD, CASOS DE ESTUDIO

Hasta el momento se ha visto como desarrollar una solución de correo electrónico de alta disponibilidad para el entorno particular de la red de datos de la Universidad Nacional Autónoma de Nicaragua (UNAN-Managua), esta cuenta con una sola conexión a Internet, con única puerta de enlace, solución Proxy SMTP (Astaro Security Gateway) para el correo entrante, dos servidores con tres dispositivos de almacenamiento (discos duros) cada uno, en fin todo lo detallado en la *sección 6.3*. Sin embargo existen otros ámbitos en los que las necesidades requieren otro tipos de soluciones de alta disonibilidad.

En este capítulo se abordaran otras arquitecturas de alta disponibilidad, en forma de casos de estudio de manera tal que el objetivo final será garantizar la disponibilidad de los servicios de correo electrónico.

7.1 Caso de estudio: Sistema de correo con escalabilidad SMTP y nodo único de almacenamiento.

7.1.1 Historia y análisis previo

En los últimos años el sistema de correo electrónico en la Universidad Nacional Autónoma de Nicaragua se ha convertido en la herramienta telemática más importante, cerca del 90% de los profesores hacen uso intensivo del correo institucional.

La arquitectura anterior (de nodo único) tiene una serie de limitaciones inherentes en diseño, rendimiento, escalabilidad y relación costo-eficiencia. Con el paso del tiempo la infraestructura ha tratado de adaptarse a los nuevos requerimientos, recurriendo a soluciones temporales, por ejemplo aumentando el almacenamiento en disco, se ha incrementado la capacidad de los buzones de 20MB a 100MB; a finales del 2007 se pasó de una configuración de 2 discos duros de 70 GB en RAID0 a 3 discos de 150GB en RAID5, en busca de mayor velocidad de acceso a datos y bajo coste de redundancia. En la actualidad se cuenta con más espacio de almacenamiento y por lo tanto la posibilidad de ofrecer buzones a mas usuarios, sin embargo el aumento significativo de más del 50% de los buzones a incrementado el trafico SMTP, creándose un cuello de botella en una arquitectura sencilla de nodo único (caso de estudio 1).

Algunas evaluaciones de rendimiento del proxy SMTP han revelado no ser el punto crítico. La solución proxy ha demostrado ser robusta, diseñada para procesar 10000 correos diarios y filtrar hasta el 99% del correo spam y almacenarlo por semanas incluso meses para su análisis posterior. Sin embargo para evitar al máximo los falsos positivos, se ha reducido la agresividad antispam en el proxy SMTP, pasando el correo a un segundo filtro antispam ubicado en los servidores SMTP, este los marca y posteriormente los reenvía a los buzones, así se delega al usuario final la tarea de clasificar el correo. En condiciones desiguales de procesamiento (distinto hardware entre

servidor y proxy SMTP) la carga del servidor ha visto un incremento suficiente como para hacer notable una caída en el rendimiento, más conexiones SMTP sumado a las conexiones IMAP, POP3 y HTTP. Consecuentemente un aumento de la cola de correos en el servidor es evidente y sin la posibilidad de cambiar el hardware, la solución inmediata es dividir la carga en equipos de características similares.

La carga típica del proxy SMTP se resume en la *tabla 7.1*.

Trafico SMTP	Diario	Semanal	Mensual	Virus
Correo SMTP procesado (Total)	3200	23400	112000	
Correo SMTP procesado (Tamaño)	250 MB	1.5 GB	7.6 GB	
Tamaño promedio de correo	80 Kb	67 Kb	71 Kb	
Score promedio (Spam	4.5	7.0	12.5	
Correo SMTP con Virus	8	50	90	
Spam	2600	4200	18000	

Tabla 7.1 *Carga típica de Proxy SMTP*

En un ambiente de producción óptimo el Antivirus/Antispam SMTP debería estar configurado en failover en modo activo/activo, balanceo de carga y enlace redundante a Internet, no obstante implicaría un aumento significativo en costos de hardware. Con uno solo, tendremos un punto único de fallo, recuerde nuevamente que lo que se busca en la configuración de alta disponibilidad es reducir al mínimo el SPOOF.

7.1.2 Diseño del sistema de correo

Sin un cambio significativo en la infraestructura se ha propuesto un cambio en la arquitectura de manera tal que brinde escalabilidad del servicio SMTP, manteniéndose inalterado el almacenamiento de los buzones.

Para su diseño se han tomado en cuenta los siguientes factores:

- Costos, alterar en lo mínimo el costo por infraestructura adicional, igualmente seguir utilizando software libre.
- Facilidad de administración, cualquier tarea administrativa agregar o borrar usuarios automatizadamente.
- Sistemas robustos con una madurez ampliamente conocida en otros ámbitos, la mayoría del software utilizado se ha heredado del sistema anterior.
- Seguridad, actualizar los sistemas a sus últimos release implica aumentar la seguridad.
- Escalabilidad en el frontend, agregar la cantidad de MX que se consideren convenientes.

En la nueva arquitectura se mantiene inalterado el software utilizado, con la salvedad de que se han actualizado a sus últimas versiones, de SuSE Enterprise 9.1 a OpenSuSE 11.0 por ejemplo. Se mantiene Postfix como servicio SMTP, configurado para entregar el correo vía LMTP, en realidad cuando nos referimos al frontend deberíamos hablar de proxys LMTP. Cyrus sigue siendo el servidor de Buzones. En esta solución no se contempla la implementación de balanceo de carga[27] o Proxy para los protocolos IMAP, POP3 y HTTP.

[27] Para evitar el spoof en el frontend de la arquitectura del sistema, el balanceador de protocolo HTTP debe estar en alta disponibilidad.

La nueva infraestructura se está ejecutando en 2 servidores Dell PowerEdge 1850 en el frontend (proxys SMTP y correo web en cada uno) y 1 servidor IBM OpenPower 510 en el backend para almacenamientos de buzones de correo (IMAP/POP3). El respaldo de los buzones de correo debería estar garantizado por un sistema avanzado de almacenamiento como una SAN[28] o por lo menos con NAS, no obstante por costos nos hemos limitado a mantener los backups en discos externos, ejecutándose un script en el crontab una vez por semana, considerar que en la actualidad un respaldo completo de los buzones es aproximadamente de 33GB.

La solución propuesta brinda alta disponibilidad con balanceo de carga* del tráfico SMTP y HTTP por round robin y alias DNS, en la *figura 7.1* se ilustra el sistema en cuestión:

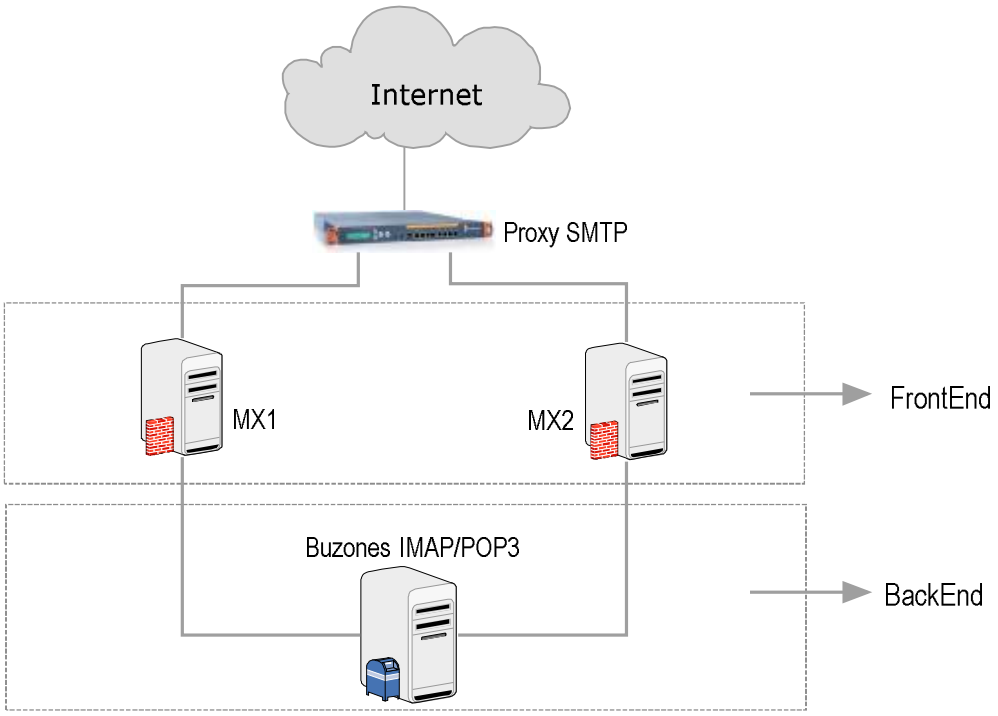


Figura 7.1 *Esquema de alta disponibilidad y balanceo SMTP en el frontend*

[28] Una red de área de almacenamiento, en inglés SAN (Storage Area Network), es una red concebida para conectar servidores, matrices (arrays) de discos y librerías de respaldo principalmente, está basada en tecnología fibre channel y más recientemente en iSCSI. Su función es la de conectar de manera rápida, segura y confiable los distintos elementos que la conforman.

Dividido en dos subsistemas, el frontend de cara a la red externa procesa el tráfico SMTP entrante/saliente, el backend aloja los buzones de usuario ofreciendo los servicios IMAP y POP3.

7.1.3 Arquitectura del nuevo sistema de correo

Son tomar en cuenta aspectos como la réplica de los buzones de usuario, balanceo de carga de tráfico http, ancho de banda y enlaces redundantes a Internet la arquitectura propuesta se describe en la *figura 7.2*.

El frontend está compuesto por dos MX, configurados para procesar el correo entrante y saliente. El balanceo de carga por round robin DNS facilita la escalabilidad con solo agregar más servidores proxy SMTP en paralelo y estableciendo sus respectivos registros MX con igual prioridad en el sistema de nombres de dominio, esta es la manera que los grandes proveedores de correo en Internet procesan el correo electrónico, manteniendo registros MX con igual prioridad se balancea el trafico SMTP y con prioridades distintas se pueden mantener backups SMTP (ver *capitulo 3*) el cual de ser posible ha de mantenerse en ISPs distintos. De igual forma en estos dos equipos se ofrece no de la mejor forma el servicio de correo web (balanceo de carga por round robin y alias DNS), una mejor solución debe incluir un balanceador de carga por protocolo de aplicación, para ello se requeriría de al menos dos* equipos adicionales, una aplicación balanceadora de protocolo HTTP y otra aplicación que controle el clúster.

Postfix puede ser visto como un proxy SMTP/LMTP, está configurado para recibir todos los mensajes de correo electrónico provenientes del proxy (antivirus/antispam), luego lo entrega vía socket local a amavid-new, este le pone la etiqueta **SPAM** en la cabecera SMTP del correo y lo devuelve a Postfix, finalmente es entregarlo al backend vía LMTP. Para asegurarnos de que el correo entregado a un buzón realmente existe, en el frontend se mantiene una lista de los buzones de correo constantemente actualizada con un cron desde el backend.

Un servicio de correo web en cada uno de los servidores del frontend garantiza balanceo de carga. Alojado en un servidor Web Apache2 el cliente de correo Web squirrelmail permite conexiones HTTP y HTTPS y está configurado para autenticarse y servir los buzones ubicados en el backend.

El backend formado por un único nodo almacena los buzones. Las cuentas de usuarios, alias, reenvíos, listas de correo y dominios son almacenadas de manera centralizada en una base de datos MySQL en el mismo equipo que sirve los buzones. La administración de los buzones se realiza a través de una aplicación web desarrollada en PHP (web-cryadm). Se han automatizado tareas en el frontend para llevar los respaldos de la base de datos y los buzones de correo.

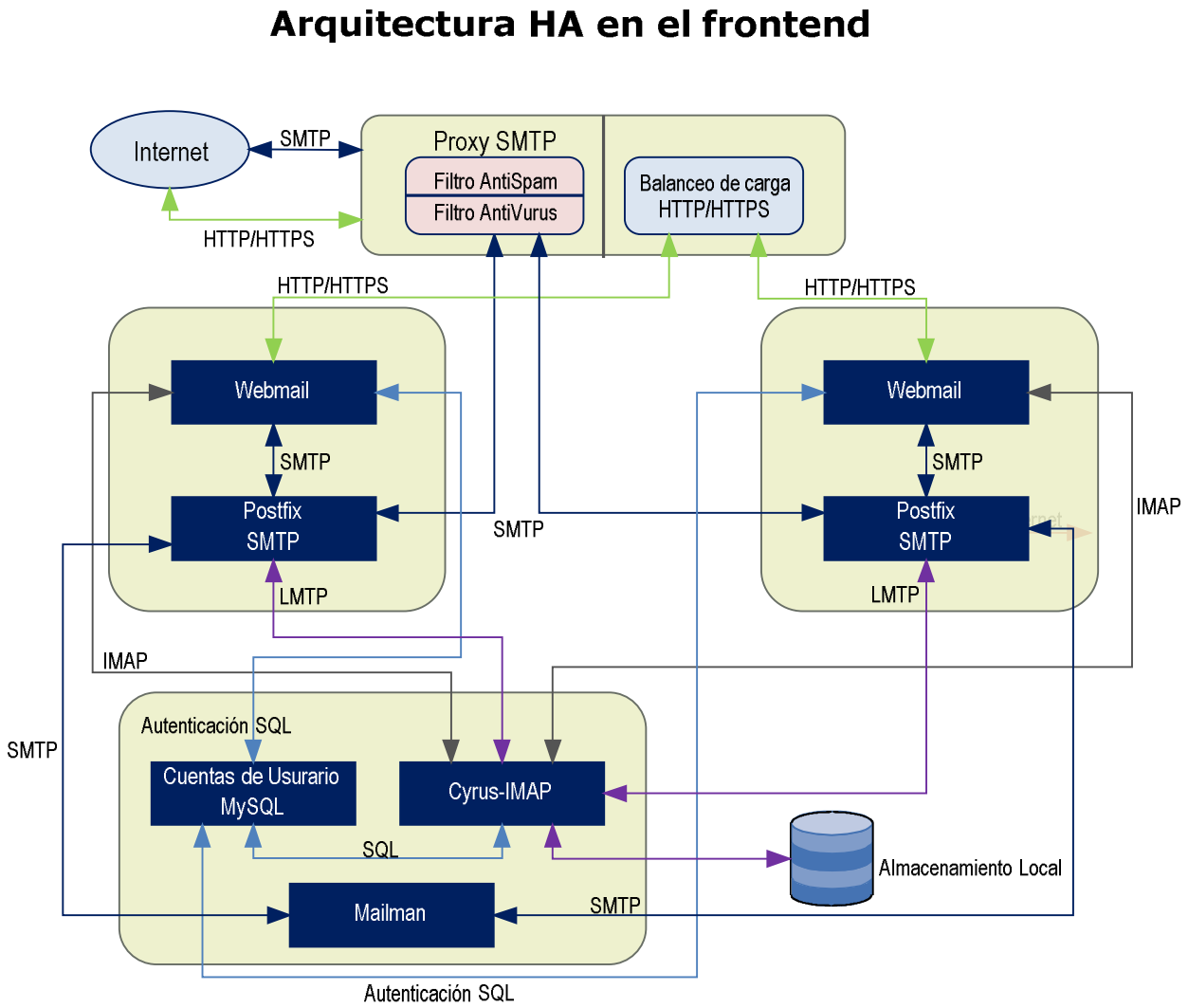


Figura 7.2 *Arquitectura de sistema de alta disponibilidad y escalabilidad en el frontend*

7.1.4 Aspectos claves del DNS

El registro clave del DNS es el MX, él es el responsable del enrutamiento del correo electrónico, el proceso previo a la transferencia de correo electrónico vía protocolo SMTP es el encaminamiento DNS. Cuando un usuario envía un correo electrónico a un dominio dado, no basta con establecer una conexión SMTP, antes necesitará consultar al DNS cual es el registro MX para dicho dominio, por ejemplo para el dominio usl.unan.edu.ni el servicio de nombres mantiene en su base de datos los siguientes registros:

```
Usl.unan.edu.ni.        IN      MX 10           mx1.usl.unan.edu.ni.
usl.unan.edu.ni.        IN      MX 10           mx2.usl.unan.edu.ni.
```

Las líneas anteriores pueden interpretarse de la siguiente manera, todo el correo dirigido al dominio usl.unan.edu.ni, será encaminado primero al servidor SMTP mx1.usl.unan.edu.ni, probablemente en la siguiente consulta el servidor de nombres devolverá como primera salida al servidor mx2.usl.unan.edu.ni por lo que ahora el correo será enrutado a este ultimo servidor. El sistema de nombres almacena dos registros con igual número de prioridad, el servicio de nombres elige que servidor MX procesará el correo electrónico para el dominio usl.unan.edu.ni.

7.1.5 Configuración de Postfix como Proxy SMTP

El objetivo de esta configuración es reenviar los correos electrónicos cuya cabecera ***Mail To*** contengan el dominio ***usl.unan.edu.ni*** al servidor de buzones Cyrus-IMAP ubicado en el backend, cualquier otro dominio debe ser rechazado. Aunque es posible transferir el correo desde el frontend hacia el backend vía SMTP, se recomienda utilizar LMTP, ya que ello significa menos procesamiento del correo y generalmente LMTP es utilizado cuando en el receptor no se dispone de una cola de correos (queue mail), en otras palabras la ventaja de utilizar LMTP es que no requiere el manejo de una cola de correo. En caso de elegir SMTP en la transferencia de los correos será necesario también instalar

otro servidor Postfix en el Backend, finalmente deberá utilizar LMTP para la entrega local al servidor Cyrus-IMAP.

Intentamos tener un firewall SMTP robusto, que deniegue el relay SMTP y evite la entrega local de correo para aumentar la seguridad. Todo el correo debe ser entregado al backend:

```
##Postfix como Proxy SMTP (main.cf)
myorigin = usl.unan.edu.ni
local_recipient_maps =
local_transport = error:local mail delivery is disabled
virtual_alias_maps = hash:/etc/postfix/virtual
mydestination =
```

Con `myorigin` las transferencias de correos al servidor Cyrus son enviadas con la cabecera *`Mail To`* inalterada, de no definir un dominio para este parámetro un mensaje con remitente `usuario@usl.unan.edu.ni` es enviado como `usuario@mx1.usl.unan.edu.ni` o `usuario@mx2.usl.unan.edu.ni`.

Los siguientes parámetros de configuración deshabilitan la entrega local de correos. Con `local_recipient_maps` Postfix valida los recipientes para los que se procesa el correo electrónico, ningún valor significa que se procesará correo electrónico para cualquier dirección, entonces se presenta un problema grave a causa de los spammers, se generará una gran cola de correo del tipo `Undelivered Mail`, notificando que el recipiente no existe. La solución es mantener una lista de los buzones de correo en el servidor Cyrus, dicha lista puede mantenerse actualizada con un script corriendo en un cron, conteniendo las direcciones de correo alojadas en el directorio /var/spool/imap/user del servidor Cyrus-IMAP. `mydestination` mantiene una lista de los dominios que son entregados vía $local_transport, por defecto Postfix entrega los correos locales al demonio master local, por lo que debemos comentar la línea correspondiente en el archivo de configuración del proceso master:

```
##Postfix como Proxy SMTP (master.cf)
# comentar la linea correspondiente al proceso local.
showq     unix  n       -       n       -       -       showq
error     unix  -       -       n       -       -       error
discard   unix  -       -       n       -       -       discard
#local     unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
lmtp      unix  -       -       n       -       -       lmtp
anvil     unix  -       -       n       -       1       anvil
```

7.1.6 Entrega LMTP

Los dos servidores en el frontend se configuran para entregar el correo al backend, para esto Postfix tiene soporte del Protocolo de Transferencia de Correo Local (LMTP) y lo utiliza para comunicarse con el Agente de Entrega Final (MDA) que bien podría estar ejecutándose en el servidor local o en un servidor remoto.

Con LMTP tenemos interesantes posibilidades, un servidor Postfix en el frontend puede manejar múltiples buzones en el backend. Si la carga de correos aumenta, puede aumentar más sistemas Postfix en el frontend y con la opción de agregar más buzones en el backend. De hecho para el autor de Postfix esa era la idea fundamental en su diseño, una arquitectura escalable que permitiera agregar servidores SMTP y servidores de Buzones.

Hay múltiples formas en las que Postfix puede ser configurado para entregar el correo vía LMTP. Los dos métodos básicos de entrega son a través de Sockets Unix y Sockets TCP. Postfix como cliente LMTP sobre un dominio de Socket Unix permite la entrega a servidores no-Postfix corriendo sobre la misma máquina, por otro lado cuando el servidor LMTP no-Postfix corre sobre un equipo remoto dentro de una red local, la entrega se hace por Sockets TCP.

También hay que considerar las opciones disponibles para el enrutamiento del correo local a través de Postfix, el mejor enfoque depende de la disposición de los servidores. Cuando el correo es dado al proceso master local, el cual lo entrega al cliente lmtp para

la entrega final al buzón, es llamado enfoque de “entrega indirecta vía agente de entrega local”. Este enfoque es poco eficiente, sin embargo es capaz de procesar los archivos aliases y forward locales. En un segundo enfoque llamado “entrega directa sin agente de entrega local” el correo es dado directamente al cliente Postfix LMTP, en este enfoque Postfix no invoca al agente de entrega local, razón por la cual no se puede procesar los archivos aliases y forward.

Aquí nos centramos en la entrega desde los servidores Postfix desde el frontend hacia el servidor Cyrus-IMAP en el backend, por lo que la entrega debe ser sobre un Socket TCP, en cuanto a la ruta a tomar a través de postfix de los correos elegimos una “entrega directa sin agente de entrega local”. La entrega local de correos se deshabilita (`local_transport`) para hacer más seguro el servidor.

Por otro lado de haber escogido un enfoque de entrega indirecta, tenemos dos mecanismos disponibles para la entrega del correo al cliente LMTP:

1. Entrega al buzon del sistema UNIX (cuentas del sistema).

2. Entrega al cliente LMTP (aplicable tanto para cuentas Unix como no-Unix).

Entrega LMTP sobre Sockets TCP:

```
##Se entrega el correo al backeed
mynetworks = 127.0.0.0/8 10.1.120.0/24 165.98.8.0/24
relay_domains = usl.unan.edu.ni
parent_domain_matches_subdomains =
        debug_peer_list smtpd_access_maps
smtpd_recipient_restrictions =
        permit_mynetworks reject_unauth_destination
relay_recipient_maps = hash:/etc/postfix/relay_recipients
transport_maps = hash:/etc/postfix/transport

##Interfaz por donde se recibe el correo
inet_interfaces = all
```

Con `mynetworks` se configuran las redes locales de donde aceptaremos correo electrónico, es decir cualquier estación dentro de este rango podrá hacer relay SMTP sobre el servidor Postfix, la utilización de `mynetworks` ha de complementarse con los parámetros `parent_domain_matches_subdomains` y `debug_peer_list`.

En el frontend se mantiene una lista de los recipientes que pueden recibir correo electrónico desde Internet, esta lista debe mantenerse constantemente actualizada, a través de un script que se ejecute periódicamente, lo más indicado es hacer una actualización de la lista cada vez que se agregue, modifique o borre un buzón de correo en el backend. En caso de no poder mantener una lista de los destinatarios validos, el parámetro `relay_recipient_maps` debe quedar vacío o agregarle una línea con el contenido "x@usl.unan.edu.ni" en el archivo `relay_recipients`.
El tipo de tablas especificada depende de lo que el sistema soporte, los opciones son hash o dbm, para saber cual soporta el sistema ejecutar el comando "postconf -d". Comúnmente postfix viene precompilado para soportar tablas hash.

Con `transport_maps` se enruta el correo con dominio destino usl.unan.edu.ni hacia el backend (servidor Cyrus-IMAPD). El contenido del archivo /etc/postfix/transport es el siguiente:

```
## Entrega LMTP al backend
usl.unan.edu.ni lmtp:[10.1.120.3]
```

Arriba se indica que la entrega es vía LMTP y especificamos la dirección IP del servidor de buzones para forzar a que no se use el registro MX en la entrega del correo.
Finalmente cada vez que modifique los archivos `/etc/postfix/transport` y `/etc/postfix/relay_recipients` ejecutamos los comandos "`postmap /etc/postfix/transport`" o "`postmap /etc/postfix/relay_recipients`".

7.1.7 Servidor LMTP en el Backend

Se supone que ya se cuenta con el servidor Cyrus-IMAP configurado para ofrecer y guardar correos con los servicios IMAP y POP3 bajo el `dominio usl.unan.edu.ni.`

Nos encontramos ante una situación de entrega por socket TCP. Cyrus-IMAP puede ser configurado para aceptar correo LMTP desde un servidor Postfix, antes definimos el puerto tcp por el cual escucha el servidor LMTP por lo que editamos el archivo `/etc/services`:

```
lmtp            24/tcp    mail          # Simple Mail Transfer
lmtp            24/tcp    mail          # Simple Mail Transfer
```

Aleatoriamente hemos escogido el Puerto 24. A continuación se configura el servidor Cyrus para abrir un socket en el puerto 24 para atender el servicio LMTP:

```
lmtp          cmd="lmtpd -a" listen="10.1.120.3:lmtp" prefork=1
lmtpunix      cmd="lmtpd" listen="/var/lib/imap/socket/lmtp" prefork=0
```

7.1.8 Enrutamiento del correo externo.

Nos referimos al correo externo, al correo enviado desde cualquier cliente que utilice los servidores Proxy SMTP cuya cabecera RCPT TO del mensaje de correo contenga una dirección de Internet valida y su registro MX apunte a un equipo que acepte conexiones SMTP en el puerto 25.

Tenemos dos tipos de clientes, los que utilizan el servidor SMTP directamente para hacer relay SMTP sobre él y los que lo utilizan indirectamente desde el servicio de webmail.

En el primer caso podrán enviar correos todos aquellos clientes que estén definidos dentro del rago de redes del parámetro `mynteworks`, complementariamente el parámetro `inet_interface` abre el puerto 25 para la interfaz que se le indique, en nuestra

configuración se aceptaran correos tanto para usuarios de la red interna como para usuarios de Internet.

Para clientes del servicio de correo web, la integración del proxy SMTP con el servidor Web, simplifica las configuraciones, al encontrarse alojados ambos servicios en el mismo equipo basta con utilizar los sockets TCP locales smtp, imap o pop3. Para el correo web ubicado en MX1 se hace relay a través del servidor SMTP local, caso similar en MX2.

7.1.9 Consideraciones de rendimiento en el frontend

El diseño de arquitecturas con alta disponibilidad y escalabilidad en el frontend se fundamentan en la utilización de software servidor SMTP cuya arquitectura haya sido pensada para sistemas de este tipo, en este particular Postfix le lleva ventaja a aplicaciones como Sendmail y Exim.

En el frontend todo parece estar bastante claro, ante un aumento de la carga agregando mas servidores SMTP se logra el balanceo de carga y la mejora de la disponibilidad del servicio. Sin embargo existen más factores a considerar en la creación de servicios LMTP, por ejemplo después de la entrega de un mensaje a través de LMTP, Postfix mantendrá abierta la conexión durante algún tiempo, a fin de que pueda ser reutilizada para una entrega posterior. Esto reduce la carga de los servidores LMTP a causa de la creación de procesos nuevas para conexión.

7.1.9 Nodo único de almacenamiento de Buzones

Postfix puede ser personalizado con ciertas características para optimizar la entrega LMTP a un servidor de almacenamiento simple, estos mecanismos siempre entregan los correos al recipiente uno a la vez. De querer implementar esta técnica con la de entrega LMTP por Socket TCP a través de la tabla `transport`, agregar al `/etc/postfix/main.cf`:

```
lmtp1_destination_recipient_limit = 300
lmtp2_destination_recipient_limit = 300
```

Postfix utiliza 50 como valor por defecto, que probablemente sea más que suficiente para obtener el máximo beneficio de una instancia simple de almacenamiento. El numero escogido debe adecuarse a las capacidades del hardware, intentar entregar concurrentemente muchos correos en lugar de acelerar el proceso puede llegar a degradar el rendimiento del servidor. Cuanto más grande sea el número, mayor provecho se obtendrá de la única instancia de almacenamiento. Sin embargo si la carga del servidor LMTP es demasiado grande necesitara más tiempo para entregar los mensajes y Postfix podría experimentar errores de timeout.

7.2 Caso de estudio: Sistema de correo con escalabilidad SMTP y de Buzones IMAP/POP3.

En el caso de estudio anterior se ha abordado desde una perspectiva de servicios SMTP la escalabilidad y disponibilidad en el frontend, sin embargo se ha dejado de un lado la arquitectura del backend.

Se mantiene el diseño del frontend anterior, mejorando la arquitectura del backend en distribución, capacidad y disponibilidad de los buzones.

7.2.1 Consideraciones del backend

La ampliación de un servicio generalmente toma uno de dos caminos: comprar mejores equipos, o distribuir la carga a través de múltiples maquinas. El primer enfoque es evidente y generalmente fácil de aplicar, aunque en algún momento es necesario manejar a pleno el software para aprovechar al máximo las ventajas de los grandes equipos. Sin embargo, si una de estas grandes máquinas falla, entonces el sistema completo colapsa.

El segundo enfoque ya ha sido tratado para la escalabilidad y disponibilidad de frontends con servidores SMTP. Este tiene la ventaja de que ya no existe un único punto de fallo y el costo acumulado de varias máquinas puede ser significativamente inferior al costo de una sola gran máquina, mas aún al ser implementado en sistemas virtualizados brinda ventajas como la consolidación de servidores, flexibilidad en la administración y mayor disponibilidad. Sin embargo, estos sistemas pueden ser más difíciles de aplicar, así como más difíciles de gestionar.

A diferencia del protocolo SMTP, en el que varios servidores pueden ser agrupados y manejados sobre el mismo espacio de nombres, en IMAP no hay un concepto del todo claro de la ubicación de los buzones a través de múltiples servidores, por lo que tratar de distribuir la carga se vuelve una tarea un poco complicada*. Comúnmente los clientes

IMAP o POP dan por hecho que el servidor con el que están hablando aloja el buzón que están buscando.

7.2.2 Balanceo de carga en el backend

Existen mecanismos convencionales con los que se pueden expandir las capacidades del backend, alojando así más buzones. Uno de los métodos más comunes es utilizar el DNS para distribuir los buzones en múltiples equipos, consiste en crear un registro DNS tipo CNAME por cada letra del alfabeto. Después se distribuyen los buzones haciendo corresponder la primera letra del login de usuario con cada servidor, por ejemplo el usuario JUAN será alojado en el servidor IMAP/POP J.USL.UNAN.EDU.NI. Para la mayoría de los ambientes esto es más que suficiente, sin embargo y si el administrador decide hacer una mejor distribución y más ordenada podría configurar un registro CNAME para cada login de usuario, así el usuario JUAN tendría como su servidor IMAP/POP JUAN.USL.UNAN.EDU.NI. Este método tiene la ventaja de que los usuarios no están en un solo servidor (reduciendo el SPOOF) y cualquier cambio no necesitaría reconfiguración de usuario. Sin embargo tiene el inconveniente de las carpetas compartidas, al encontrarse en equipos diferentes dos cuentas no podrán compartir archivos a menos que se utilice un sistema de archivos distribuido como NFS o AFS.

Otro mecanismo de distribución de carga es la multiplexacion IMAP. Este método supone la existencia de múltiples servidores en el frontend y backend. Los servidores en el frontend (proxy IMAP/POP) hacen la búsqueda de los buzones en el backend a través del multiplexor IMAP. El multiplexor ve si el usuario ha sido autenticado, una vez que el usuario se ha autenticado, el frontend hace una búsqueda en los servidores del backend y luego abre una conexión directa con un solo servidor del backend, el servidor que mantiene el buzón buscado. Este sistema proporciona flexibilidad y balanceo deA carga, dando la posibilidad de distribuir los usuarios entre múltiples servidores pero crea el inconveniente de no poder compartir carpetas de usuarios ubicados en distintos servidores del backend.

7.2.3 Solución de equilibrio de carga y HA para el Backend

El enfoque propuesto intenta solucionar los problemas de escalabilidad, disponibilidad y equilibrio inherentes en una arquitectura de nodo único de almacenamiento en el backend. Para ello se ha usado del servicio de agregación de servidores de Cyrus-IMAP (Cyrus Agreggator). Cyrus Agreggator toma un conjunto de servidores Cyrus IMAP y los presenta como un solo servidor IMAP a los clientes, es decir todos los buzones alojados en cada uno de los servidores IMAP/POP son publicados en una sola imagen, lo que lo hace ver como un solo servidor para los clientes.

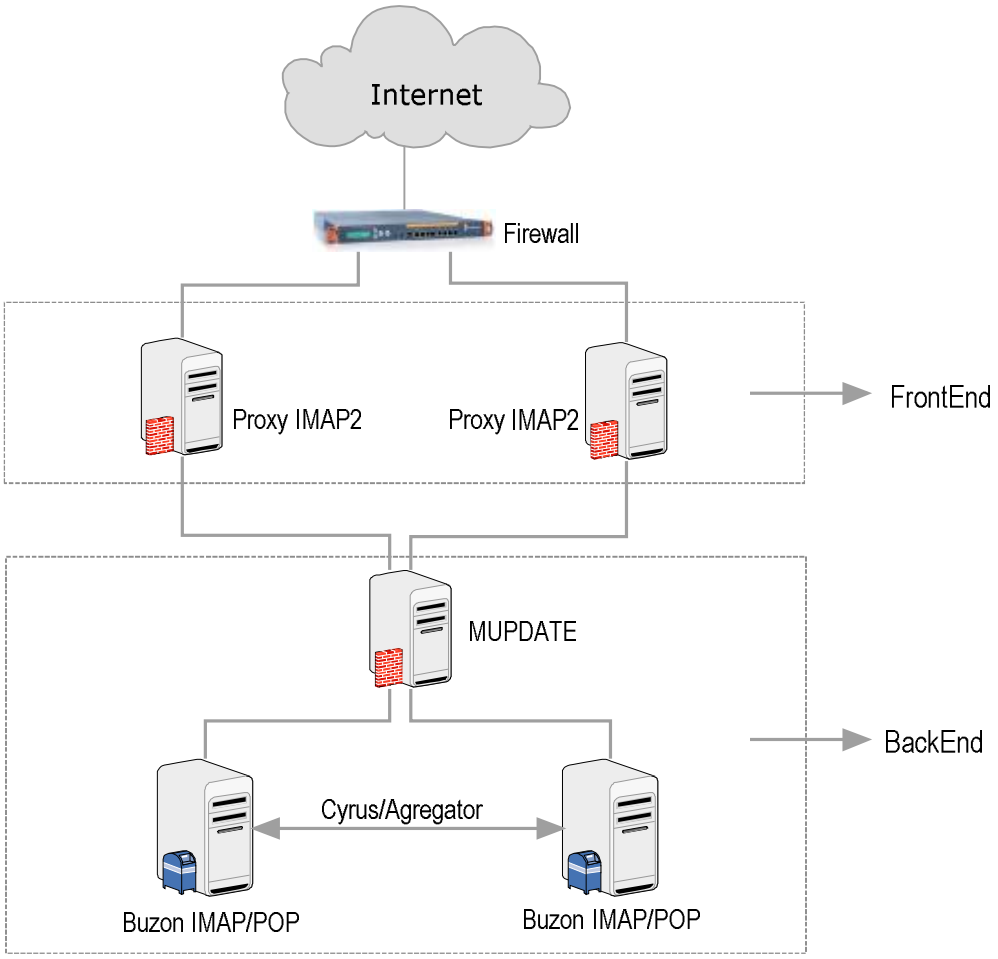


Figura 7.3 *Esquema de alta disponibilidad y balanceo IMAP/POP en el backend*

La arquitectura de Cyrus Agreggator dispone de tres tipos de servidores:

- El IMAP frontend (servidor proxy IMAP).
- El IMAP backend (servidor de buzones).
- Y el MUPDATE.

Los servidores del frontend actúan como el principal punto de comunicación entre los usuarios finales y los servidores del backend. El frontend usa el servidor MUPDATE como una fuente autoritativa para localizar, obtener permisos y nombres de los buzones. Los servidores en el backend almacenan los datos reales y mantienen actualizado al servidor MUTPDATE de los cambios de la lista completa de buzones.

Integrando la arquitectura de Cyrus Agreggator con el sistema de alta disponibilidad para servicios SMTP obtenemos el siguiente esquema:

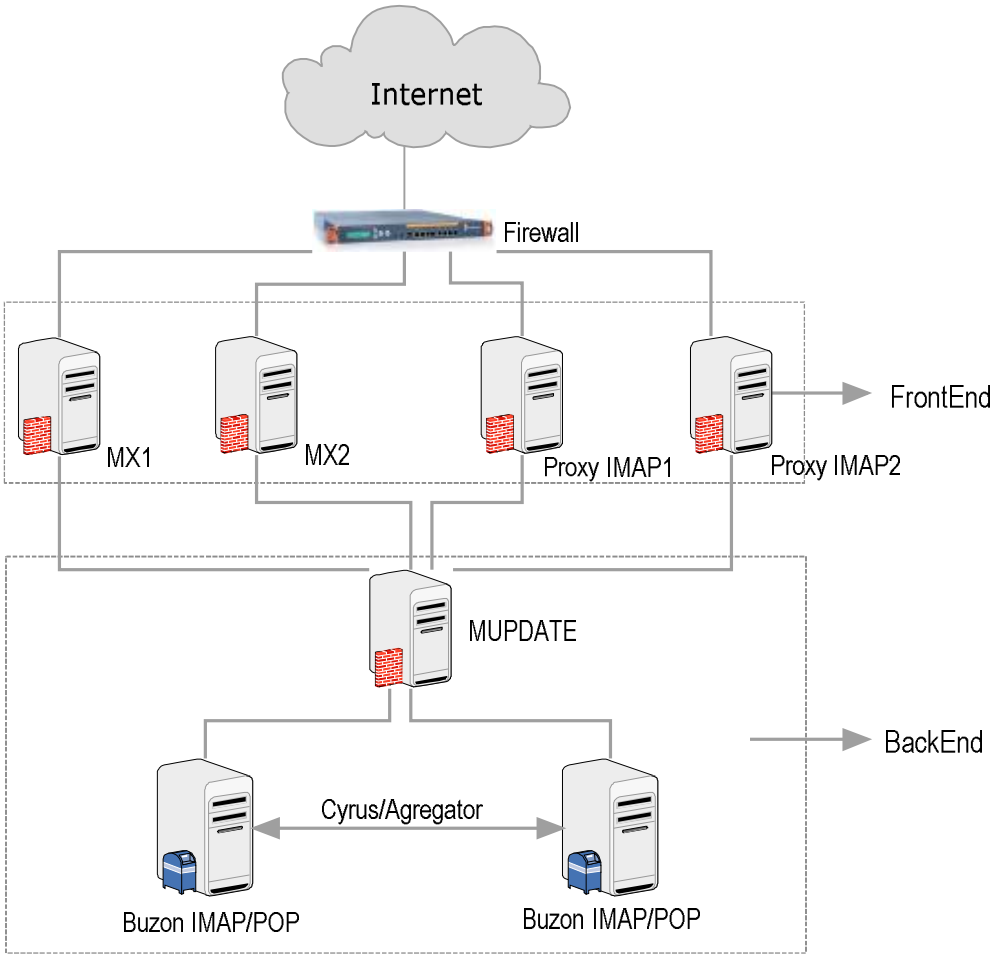


Figura 7.4 *Esquema de alta disponibilidad y balanceo SMTP e IMAP/POP en el backend y frontend*

El esquema completo está compuesto por cuatro servidores en el frontend, un servidor intermedio MUPDATE y dos servidores de buzones dentro del backend. Los cuatro servidores* proxy SMTP e IMAP podrían simplificarse en dos servidores, sin embargo para aumentar la disponibilidad se ha decidido utilizar servidores separados. El servidor MUPDATE (multiplexor IMAP) sirve de interfaz entre los servidores proxy IMAP del frontend con los buzones del backend y tiene el inconveniente de formar un punto único de fallo, una mejor implementación debe asegurar alta disponibilidad con replicas en modo maestro/esclavo del servidor MPUDATE, garantizando de esta manera alta disponibilidad en todo el sistema de correo. Los servidores de almacenamiento de

buzones con Cyrus Agreggator posibilitan la escalabilidad en el backend, es decir la opción de aumentar quotas y cuentas de correo electrónico.

*Aunque se habla de cuatro servidores en el backend, la implementación real es mantenida en 3 equipos. Equipos virtualizados con vmware están organizados de la siguiente manera, en dos servidores se asocian los proxy IMAP y SMTP y en el tercer equipo el servidor MUPDATE.

7.2.4 Arquitectura del nuevo sistema de correo

7.2.4.1 Servidores del frontend

Nuevamente el frontend está compuesto por dos MX, configurados para procesar el correo entrante y saliente haciendo balanceo de carga por round robin DNS, el sistema es escalable ya que permite agregar cualquier cantidad de servidores proxy SMTP en paralelo.

Se han agregado al frontend dos servidores proxy IMAP, que sirven de interfaz entre los clientes y los buzones alojados en el backend. Igual que el caso del servicio SMTP de manera sencilla se intenta balancear el trafico IMAP/POP por round robin DNS, alternativamente se podría haber implementado el mecanismo de distribución por Alias DNS, aunque en realidad cualquier buzón podría ser atendido por cualquiera de los dos servidores proxy IMAP.

Los servidores del frontend, a diferencia de los del backend son completamente intercambiables entre si y puede considerarse que son servidores que no manejan datos; cualquier caída de un proxy no se traduce en la perdida de datos. Los únicos datos (lista de buzones) persistentes que se mantienen son administrados por el servidor maestro MUPDATE. Esta lista se encuentra constantemente sincronizada entre el frontend y el servidor MUPDATE.

La lista de buzones de correo es administrada por el servidor MUPDATE. Note que en esta arquitectura MUPDATE forma un punto único de fallo, por lo que se debe considerar implementar un servidor de replicas.

Para el servicio IMAP en el frontend, hay dos tipos de procesos, el proxyd y el proceso de sincronización mupdate (modo esclavo). El proxyd maneja los periodos de sesiones con los clientes y se basa en una base de datos consistente que refleje el estado de todos los servidores del backend. Este proceso nunca escribe sobre la base de datos de buzones de correo, por el contrario esta base de datos se mantiene sincronizada con el servidor MUPDATE maestro a través del proceso mupdate esclavo.

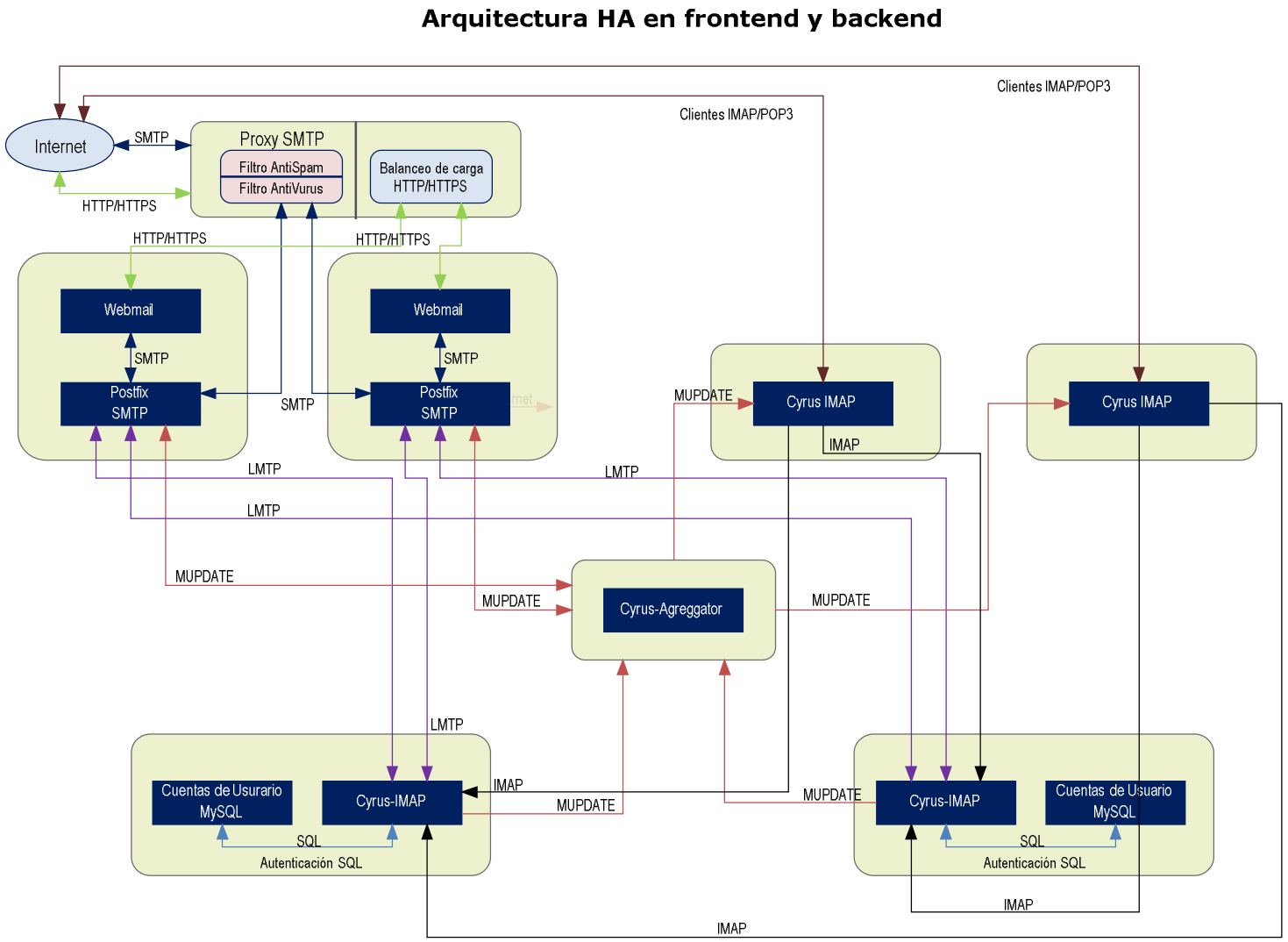


Figura 7.5 *Arquitectura de Alta disponibilida en frontend y backend*

7.2.4.2 Servidores del backend

El diseño del backend depende de las prioridades que se tengan, si desea tener redundancia de datos, es válida la solución presentada en caso de estudio 1 (replica de nodos) no obstante el rendimiento decaerá a medida que crezca la cantidad de usuarios. Otra necesidad podría ser aumentar la capacidad de almacenamiento, bien podría hacerse con una red de área almacenamiento (SAN[29]) o aumentando la cantidad de servidores IMAP/POP3, es conocido que la primera resulta ser una solución bastante costosa, costo retribuido a la simplificación de administración y flexibilidad obtenida. Agregar más servidores de almacenamiento de buzones complica la situación, brindar los buzones de correo bajo el mismo espacio de nombres es más complejo que unificar servidores SMTP con el Sistema de Nombres de Dominio, se necesita de un servidor centralizado que conozca en todo momento la distribución y localización de los buzones en todos los servidores. Notar que bien podría tratarse de una combinación de ambas, agregación de servidores IMAP/POP3 con almacenamiento SAN.

Los servidores del backend almacenan los datos reales, son completamente funcionales y sirven los buzones guardados en el propio servidor. Cada servidor del backend mantiene una base de datos de los buzones locales y lista que buzones están disponibles sobre él. Los procesos imapd sobre los servidores son completamente independientes, de manera que cada servidor IMAP puede ser utilizado de manera aislada, la caída de uno de ellos solo afectara los buzones locales. Sin embargo están configurados de tal modo que no se permite realizar cualquier operación sobre ellos (CREATE, DELETE, RENAME, SETACL, etc) a menos que el servidor MUPDATE haya sido contactado y autorice la transacción.

En este modo (transacción autorizado por el MUPDATE maestro) el proceso impad actualiza los buzones locales. Por ejemplo, cuando se atiende una sentencia CREATE se

[29] Una red de área de almacenamiento, en inglés SAN (Storage Area Network), es una red concebida para conectar servidores, matrices (arrays) de discos y librerías de respaldo principalmente, está basada en tecnología fibre channel y más recientemente en iSCSI. Su función es la de conectar de manera rápida, segura y confiable los distintos elementos que la conformanan.

necesita reservar un lugar en el servidor maestro MUPDATE para asegurar que ningún otro servidor del backend intente crear el mismo buzón de correo. Una vez que se ha completado la creación del buzón, le corresponde su activación al servidor MUPDATE, y es considerado como disponible para ser accesado por cualquier cliente a través de los proxys IMAP del frontend.

En la *figura 7.4* se observa detalladamente una interacción de cada uno de los componentes de la arquitectura.

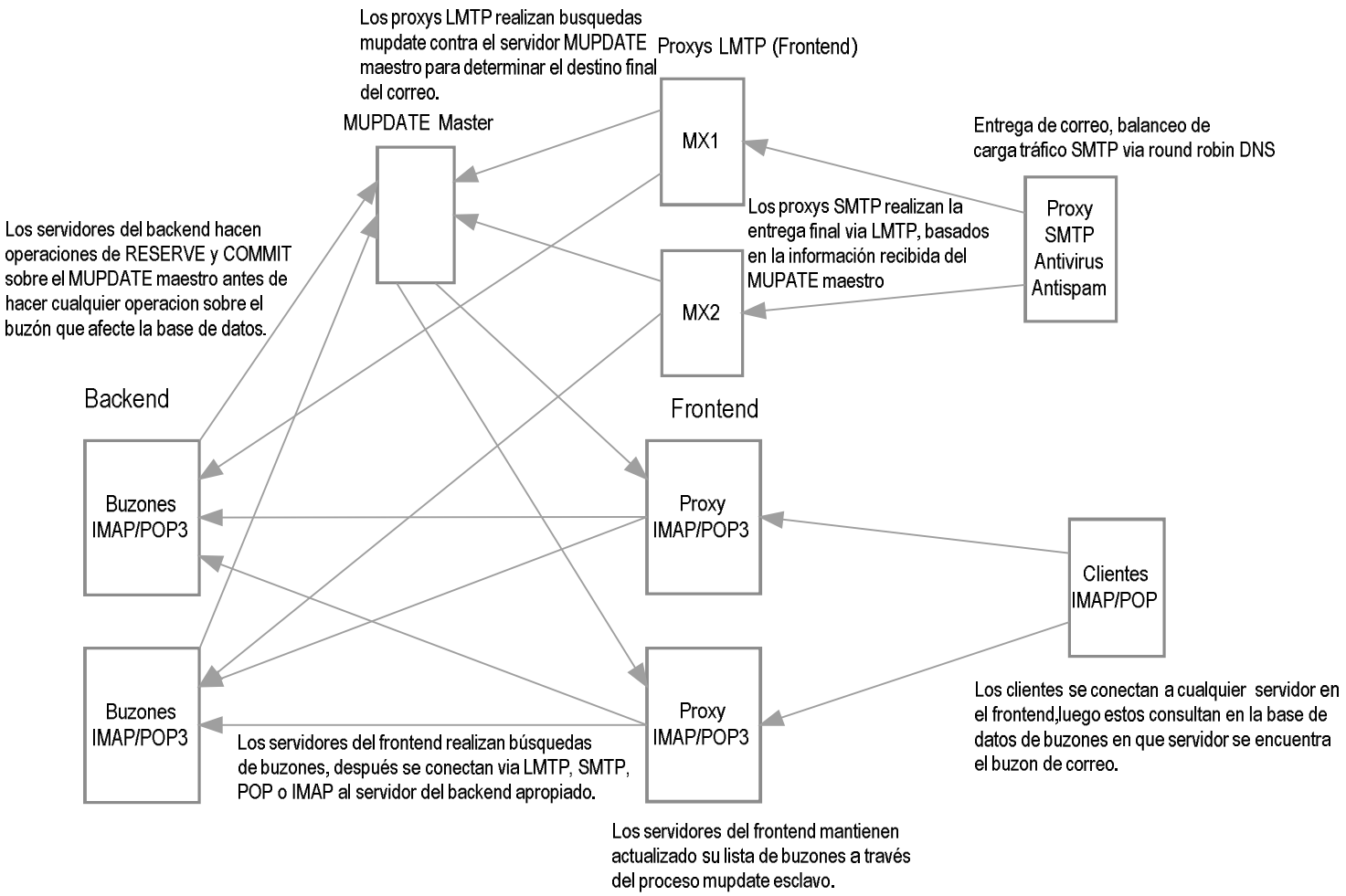


Figura 7.6 *Interacción de componentes de la arquitectura*

La entrega de correo final desde los servidores proxy SMTP del frontend hacia los servidores de buzones del backend es vía LMTP. Los proxy LMTP ejecutándose en el frontend utilizan el servidor MUPDATE maestro para determinar la ubicación exacta del buzón destino y a continuación transfieren vía LMTP el mensaje al servidor apropiado del backend. Si el servidor del backend no está disponible, entonces el proxy LMTP devuelve un error al servidor MTA.
El proceso de autenticación del usuario es a través del servidor proxy del frontend vía cualquier mecanismo SASL que soporte texto plano. Si la autenticación es satisfactoria, el servidor del frontend se autenticara sobre el servidor de backend usando TEXT PLAIN como mecanismo de autenticación.

Para ayudar a proteger al backend de los compromisos a los que puede estar expuesto a causa del frontend, todas las tareas administrativas (creación de usuarios, buzones, cambios de cuotas, etc) deben hacerse directamente desde el cliente hacia el backend, a los proxys del frontend no se les concede ningún permiso administrativo.

7.2.5 Ejemplo de una operación administrativa

Las tareas administrativas son coordinadas directamente por el servidor MUPDATE maestro. Comandos como CREATE, DELETE, RENAME, SETACL son ejecutados por los servidores del backend bajo la autorización del servidor MUPDATE, después los cambios son transferidos al frontend vía protocolo MUPDATE[30].

Por ejemplo el comando CREATE crea un buzón de correo sobre el servidor del backend sobre el que se ejecuta y lo establece como padre del buzón. Si la carpeta padre existe sobre múltiples servidores del backend, entonces el servidor responde con una etiqueta NO al servidor MUPDATE indicando que no se puede crear el buzón.

Cuando esto sucede el administrador tiene dos opciones. Se puede conectar directamente al servidor del backend y ejecutar el comando CREATE sobre este. Por

[30] El protocolo MUPDATE es descrito en el RFC 3656, y una descripción breve puede encontrala en http://cyrusimap.web.cmu.edu/archives/mupdate.html.

otra parte, se puede enviar un segundo argumento después de crear el buzón con CREATE. Este argumento especifica los nombres de los host sobre los cuales se ha creado el buzón de correo.

En el frontend ocurren la siguientes operaciones al ejecutar CREATE:

- `proxyd`: verifica que el buzón no existe en la lista de buzones MUPDATE.
- nivel superior no pueden ser creados por los proxys).
- `proxyd`: En el backend se duplica el comando CREATE y verifica que no se ha creado ninguna inconsistencia en la lista de buzones (si el nombre de la carpeta aún es único).

En el backend se producen las siguientes operaciones:

- `imapd`: verifica la listas de control de acceso
- `imapd`: inicia la transacción de buzones.
- `imapd`: es posible que tenga que abrir una conexión MUPDATE si el buzón ya existe.
- `imapd`: MUPDATE establece usl.unan.edu.ni como reservado.
- `imapd`: crea el dominio padre `usl.unan.edu.ni` en el directorio `/var/spool/imap`.
- `imapd`: añadir los buzones de datos al directorio padre usl.unan.edu.ni.
- `imapd`: commit de transacción.
- `imapd`: MUPDATE establece a usl.unan.edu.ni como activo.

7.2.6 Configuración de servidores Cyrus IMAP

Se da por hecho que los servidores de almacenamiento del backend ya están configurados para guardar los mensajes de correo, así como servir los buzones con los protocolos IMAP y POP3.

La arquitectura de Cyrus Agreggator define tres servidores, por lo que para cada uno de ellos se procederá a configurar de la siguiente manera:

7.2.6.1 Servidores del backend

Los servidores del backend son configurados para enviar los cambios (creación, modificación, elminacion de buzones) al servidor MUPDATE maestro. Igualmente deben asociarse para formar parte del Cyrus Murder (Conjunto de servidores Cyrus), para ello se establece el parámetro mupdate_server en el archivo /etc/imapd.conf:

```
##  The mupdate server for the Cyrus Murder
mupdate_server: mupdate.rd.unan ## fqdn interno de servidor MUPDATE
proxyservers: murder
```

Con `proxyservers` se definen los usuarios o grupos que son permitidos para tener pleno control administrativo sobre el servidor del backend. Se ha creado el usuario murder y se debe estar seguro de la autenticación satisfactoria sobre cualquiera de los servidores del backend.

Finalmente dependiendo de qué mecanismos de autenticación se estén utilizando pueden también establecerse algunos de los siguientes valores:

- `mupdate_username:`
- `mupdate_authname`
- `mupdate_realm`
- `mupdate_password`

Una vez ajustados los parámetros necesarios, cualquier operación sobre los buzones del backend será enviada al servidor MUPDATE maestro para su confirmación y entrada en la base de datos mupdate.

7.2.6.2 Servidor maestro MUDPATE

Un servidor MUPDATE maestro no es más que un servicio Cyrus IMAP corriendo en modo maestro el protocolo MUPDATE.

Para configurar el servidor MUPDATE maestro, se agregará la siguiente línea en el archivo de configuración `/etc/cyrus.conf`.

```
## Cyrus-IMAP como servidor MUPDATE maestro
mupdate       cmd="/usr/lib/cyrus/bin/mupdate -m" listen=3905 prefork=1
```

Con la opcion “-m” se le indica a mupdate que debe arrancar en modo maestro.

A continuación tendrá que configurar al menos un esbozo del archivo `/etc/imapd.conf` en con el que el pueda autenticarse el servidor.

```
## Configuracion de parametros en /etc/imapd.conf
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
sievedir: /var/lib/sieve
admins: cyrus
allowanonymouslogin: no
```

Puede dejar algunos de los parámetros por defecto del archivo, para propósitos del servidor MUPDATE son irrelevantes.

7.2.6.3 Importación de base de datos

El servidor MUPDATE debe importar la base de datos de buzones de correo mantenida en los servidores del backend. Para hacer la primera sincronización, simplemente se cambia al usuario Cyrus y se ejecuta el comando ctl_mboxist –M:

```
##  Importacion de la base de datos en MUPDATE
su cyrus
cyrus@mupdate:/root> /usr/lib/cyrus/bin/ctl_mboxlist –m
```

Tenga en cuenta que puede ejecutar el comando “`ctl_mboxlist -mw`” para ver por pantalla todas las operaciones que se están realizando.

Si todo marcha bien la base de datos generada por la ejecución de `ctl_mboxlist` almacenara la lista de buzones mupdate master, de presentarse problemas la causa mas probable es errores en los ajustes de autenticación, o que el servidor MUPDATE maestro realmente no se esté ejecutando como master. La herramienta `mupdatetest` puede ser utilizada para depurar errores de configuración.

También suele ser útil hacer que los servidores del backend se re-sincronicen automáticamente con el servidor MUPDATE maestro, para ello se agrega el siguiente parámetro al archivo de configuración `/etc/cyrus.conf`.

```
##  Sincronizar automáticamente, archive /etc/cyrus.conf
mupdatepush cmd = "ctl_mboxlist-m"
```

Con esto se ejecutará una sincronización automática con el mupdate master cada vez que un servidor del backend se reinicie, con lo que se mantiene actualizada la base de dato del backend.

7.2.6.4 Servidores del frontend

La configuración de un servidor del frontend se puede resumir en:

- Se establece el `mupdate_server` y se agrega al conjunto (cyrus murder). Sin embargo debido a que el backend hablara con el servidor MUPDATE maestro a través de un proceso mupdate esclavo que se ejecute en la maquina local, se configura un slave mupdate en la sección SERVICES del archivo `/etc/cyrus.conf`:

```
##  Servicio de base de datos mupdate – prefork debe establecerse a 1
  mupdate       cmd="mupdate" listen=3905 prefork=1
```

  `Prefork` se establece a 1 a fin de que la base de datos se pueda sincronizar al inicio. De lo contrario, el servicio no comenzará a correr hasta después de recibir una conexión de uno de los clientes mupdate.

- También se requiere cambiar todas las entradas `imapd` para ser ahora `proxyd`, igualmente para `lmtpd` que pasan a ser lmtpproxyd, al final la sección SERVICES luce de la siguiente manera:

```
##  Servicio de base de datos mupdate – prefork debe establecerse a 1
  mupdate       cmd="mupdate" listen=3905 prefork=1

##  Servicios proxy
  imap          cmd="proxyd" listen="imap" prefork=5
  pop3          cmd="pop3d" listen="pop3" prefork=0
  lmtp          cmd="lmtpproxyd" listen="/var/lib/imap/socket/lmtp" prefork=0
```

Si necesita configurar entradas para indicar al proxy auth (en el backend) el login y la clave de autenticación (en caso de utilizar SASL) puede utilizar los parámetros `mail_password` y `proxy_authname`. Por ejemplo en este caso que se utilizan dos servidores de backend con fqdn mail-store1.rd.unan y mail-store2.rd.unan, con claves “mail1” y “mail2”, y cuyo nombre del conjunto (Cyrus Agreggator) es murder, la configuración de autenticación sería:

```
##  proxy auth, autenticación del backend
mail-store1_password: mail1
mail2-store2_password: mail2
proxy_authname: murder
```

7.2.7 Entrega del correo electrónico al backend

Para entregar el correo al servidor de buzones adecuado, se mantiene la configuración de entrega LMTP convencional, sin embargo en lugar de conectar directamente a lmtpd, la conexión debe hacerse a lmtpproxyd.

Los proxys SMTP del frontend en caso de entregar los correos via transport_maps, utilizando sockets TCP, la configuracion de Postfix solo requiere modificar el socket de entrega:

```
##  Socket de entrega lmtpproxyd.
#        postmap /etc/postfix/transport
usl.unan.edu.ni lmtpproxyd
```

7.2.8 Aspectos administrativos de la arquitectura

Es de suma importancia mantener la consistencia de la base de datos en el conjunto de almacenamiento, el servidor MUPDATE maestro se encarga de ello manteniendo actualizado el estado de los servidores del backend y transfiriendo la base de datos (cada vez que se requiera) a los servidores del frontend. Los servidores del frontend re-sincronizan sus entradas cada vez que se inician, la caída de un proxy IMAP o SMTP no debe representar un problema. Normalmente el frontend está recibiendo

actualizaciones de la base de datos, cuando se pierda la conexión con el master mupdate, automáticamente se intenta reiniciar la conexión y re-sincronizar la base de datos.

Cada vez que el espacio de nombre sea modificado, es recomendable hacer una re-sincronización manual con el servidor MUPDATE maestro ejecutando "`ctl_mboxlist -m`". Si dos servidores tienen el mismo buzón, se necesitará resolver este problema antes de que la consistencia de los datos sea comprobada.

La escalabilidad del backend se logra fácilmente, basta con configurarle al servidor que se desee agregar el parámetro "`mupdate_server`" y luego crear los nuevos buzones a como se haría con cualquiera de los otros servidores.

Mantener réplicas de los servidores, interesa más en el backend y no tanto en el frontend, ya que prácticamente el frontend carece del manejo de datos, salvo la lista de buzones que realmente es actualizada desde el servidor MUPDATE maestro. Las replicas de los servidores de almacenamiento duplica los costos para el backend, lo más recomendable sería mantener respaldo en discos externos (NAS) de los buzones y la base de datos de usuarios. Por tratarse de una arquitectura de buzones distribuidos en el backend, la caída de uno de los servidores representará una pérdida parcial del sistema, dando facilidades en el tiempo de respuesta para la recuperación del servicio de buzones de los usuarios afectados.

El servidor MUPDATE no necesita respaldo de los datos por dos razones. Primero que gracias al manejo de replicadas a través de un servidor MUDATE esclavo se podría configurar el servicio en alta disponibilidad. La segunda razón es que realmente no se necesita mantener copia de los servidores (MUPDATE maestro y esclavo), ya que estos datos son generados por los servidores del backend.

8. CONCLUSIONES Y RECOMENDACIONES

8.1 Conclusiones

En el presente documento se ha planteado la necesidad de resolver una de las necesidades más críticas en los sistemas de correo electrónico, la disponibilidad. Dado que en la actualidad el sistema de correo de la UNAN-Managua presenta constantes fallas inherentes a su arquitectura, se ha intentado hacer el diseño e implementación de un servicio de alta disponibilidad, tratando de aprovechar al máximo los recursos disponibles. Además se dejan algunas propuestas de arquitecturas que ayuden a resolver los problemas de disponibilidad de servicios, redundancia de datos y rendimiento.

En base a los objetivos propuestos se ha llegado a las siguientes conclusiones:

- Las tecnologías actualmente utilizadas en el sistema de correo electrónico de la UNAN-Managua son seguras y escalables por lo que su uso sigue vigente.
- Se analizaron distintos mecanismos o tecnologías que intentan brindar servicios de alta disponibilidad para el correo electrónico.
- Se Probaron tanto en ambiente de laboratorio como de producción las arquitecturas de alta disponibilidad desarrolladas.
- Se realizo un proceso de evaluación de las distintas soluciones para determinar cuál es la que mejor se adapta a las necesidades y requerimientos del servicio brindado por la UNAN-Managua.

8.2 Recomendaciones

Debido a que los servicios de correo electrónico, implican una combinación de distintas y muchas tecnologías, la implementación de una solución uniforme de alta disponibilidad no es un tema resuelto por completo, consecuentemente las necesidades por resolver aún son muchas.

Particularmente para la arquitectura implementada en la UNAN-Managua, se recomienda lo siguiente:

- Los servidores que no manejan datos, por ejemplo los proxys SMTP e IMAP del frontend se sugiere sean instalados en maquina virtual.
- Para el frontend de cualquiera de las arquitecturas se recomienda implementar balanceo de carga a nivel de protocolo, para HTTP e IMAP.
- Se recomienda configurar al menos un servidor SMTP como backup MX fuera de la Universidad, se sugiere alguno de los centros de regionales.
- Para los volúmenes de buzones de correo actualmente manejados se recomienda la adquisición de un sistema de almacenamiento en red que de más fiabilidad, capacidad y mayores tasas de transferencia, por ejemplo una SAN.
- Se recomienda hacer una separación proporcional de la carga (equipos utilizados en la arquitectura) en dos Sistemas Ininterrumpidos de Energía (UPS) completamente independientes.
- En equipos de red como switches y routers se recomienda tolerancia a fallos, son recursos que están fuera de la arquitectura.
- Se sugiere la adquisición de un enlace más a Internet, para tener redundancia en las conexiones externas.

BIBLIOGRAFIA

Robert H'obbes' Zakon http://www.zakon.org/robert/internet/timeline/
http://www.ietf.org/rfc/rfc2235.txt (RFC Historico)

Dent, Kyle D.(2003). Postfix The Definitive Guide, O'Reilly

Hildelbrandt, Ralf. Koetter, Patrick. (2005). Postfix State of the art message tranposrt, no Starch Press.

Mullet, Dianna. Mullet Kevin.(2000). Managing IMAP, O'Reilly

Piedad, Floyd. Hawkins Michael. (2001). High Availibity: Design, Techniques, and Processes, Prentice Hall PTR.

Petersen, Richard. (2007). Fedora Core 7 & Red Hat Enterprise Linux: The Complete Reference, McGraw-Hill Professional.

Paredes, Juan Pedro. Alta disponibilidad para Linux, juanpe@retemail.es

Postfix Documentation. De www.postfix.org

Web-Cyradm Project, Documentation. De www.web-cyradm.org

Project Cyrus, Computing Services, Carnegie Mellon University. De www. cyrusimap.web.cmu.edu

RFC Editor Webpage, IETF. De www.ietf.org/rfc.html

Linux-HA Web Site: www.linux-ha.org

Event Helix Studo, www.eventhelix.com

ANEXOS

Lista de Tablas

Lista de Figuras

Lista de Acrónimos

ACL: *Listas de control de acceso o access control list.*

AFS: *Andrew File Sharing*

ASCII: *acrónimo inglés de American Standard Code for Information Interchange*

BIND: *Berkeley Internet Name Domain, Servicio DNS de la Universidad de Berkeley, el más utilizado en Internet.*

DNS: *Domain Name System*

DoS: *Deny of Service, ataque por denegación de servicios*

FQDN: *Fully Qualified Domain Name, Nombre de Domino Completamente Calificado, se refiere al nombre completo de un host (nombre de host + dominio).*

FT: *Tolerancia a fallos o foult tolerant,*

GPL: *General Public License*

IETF: *Internet Engineering Task Force, Grupo de Trabajo en Ingeniería de Internet*

IMAP: acrónimo inglés de Internet Message Access Protocol

IP: *Protocolo de Internet*

LDAP: *Lightweight Directory Access Protocol, Protocolo Ligero de Acceso a Directorios.*

MDA: *Agente de Entrega de Correo*

MIME: *Multipurpose Internet Mail Extensions, Extensiones de Correo Internet Multipropósito*

MTA: *Agente de Transferencia de Correo*

MUA: *Agente Usuario de Correo*

MX: *Mail Exchange, véase Registros DNS*

NFS: *Network Fie Sharing*

PAM*: Pluggable Authentication Module, véase Servicios de Autenticacion de Usuarios.*

POP: *Post Office Protocol, Protocolo de Oficina de Correos*

RAID: *Arreglo Redundante de Discos Independientes*

RBL: *Listas negras en tiempo real o realtime blakhole list.*

RFC: *Request For Comments*

RR: *Registro de Recursos, estructura de datos que almacena informacion del DNS*

SAN: *Storage Area Network o Red de Area de Almacenamiento.*

SASL: *Simple Authentication and Security Layer. Estructura que provee autenticación y servicios de seguridad de datos en protocolos orientados a conexión. Brinda la interfaz de una estructura entre los protocolos y los mecanismos, permite a los nuevos protocolos el rehúso de mecanismos ya existentes y a los viejos protocolos el uso de los nuevos mecanismos.*

SMTP: *Protocolo Simple de Transferencia de Correos*

SPOF: *Véase introducción a los sistemas de alta disponibilidad*

STONITH: *Shoot The Other Node In The Head o Pegale un Tiro en la Cabeza al otro Nodo*

TCP: *Transfer Control Protocol*

TTL: *Time To Live, véase El Sistema de Nombres de Dominio*

UPS: *Sistemas ininterrumpidos de energía o*

W3C: *World Wide Web Consoritium*