4 Solución para la ESI.

4.1 Dimensionado y propuesta de la solución.

Teniendo en cuenta el punto 2.1 dónde se describe la infraestructura de correo actual de la ESI dentro de la US, se propone como solución corporativa de correo electrónico un sistema sin módulo relay, y con el resto de módulos dobles, es decir, un total de 8 nodos de la siguiente forma:

- 2 Servidores buzón: buzon01 y buzon02
- 2 Servidores directorio: ldap01 y ldap02
- 2 Servidores balanceadores: lvs01 y lvs02
- 2 Servidores almacén: almacen01 y almacen02

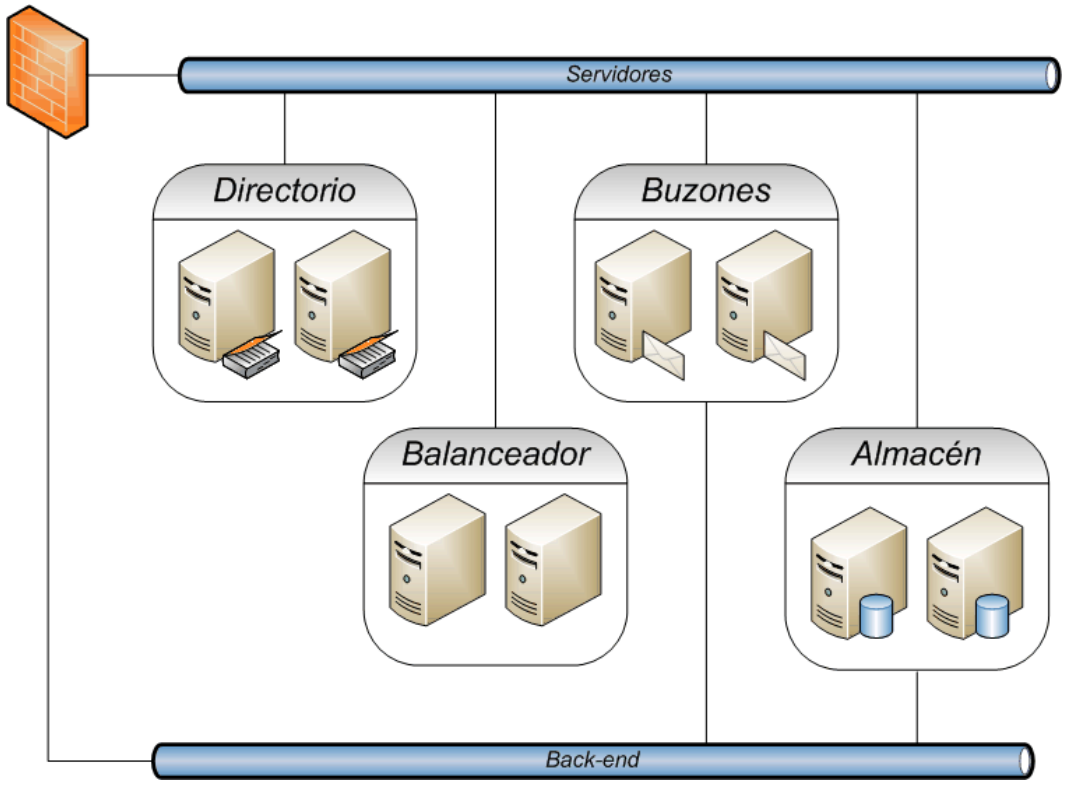

Ilustración 32: Solución propuesta.

Este sistema está dimensionado para 10.000 usuarios con una carga de trabajo administrativa diaria estándar, ya que cada buzón está dimensionado para aproximadamente 5000 usuarios. Podría decirse que el sistema está sobredimensionado, incluso si se incluyen a los alumnos como usuarios (5500) además de PAS/PDI (500 usuarios).

De todas maneras, al implementar un sistema con semejante robustez, 10000 es el número mínimo de usuarios para el que está calculado, por lo que en el caso de PDI/PAS el sistema estará funcionando al 10%, y en el caso de incluir a los alumnos, aproximadamente al 40% (la carga de los alumnos no será tan importante como la de PDI/PAS, y por tanto le aplicamos un factor corrector de 0.7).

También hay que tener en cuenta que en este tipo de sistemas no conviene superar el 50% de carga, ya que en el caso de que cayera un buzón, la carga se transformaría automáticamente en 20% para PDI/PAS, y 80% para PDI/PAS/Alumnos.

En cuanto a la cuota de usuario proponemos aumentarla hasta 1GB para PDI/PAS y 250MB para alumnos, lo cual se traduce en una necesidad de aproximadamente 2TB para almacenamiento.

También se propone la utilización de redes de comunicaciones independientes para el tráfico de correo (SMTP, IMAP, POP, HTTP) y tráfico de almacenamiento (NFS). Se utilizará electrónica Ethernet Gigabit con cableado de categoría 6, aprovechando la infraestructura actual de la ESI.

Asimismo se utilizarán los 2 servidores actuales para el nuevo sistema, y se necesitarán 6 servidores más de potencia similar, es decir, 8 en total. Los buzones deberán tener 4 interfaces de red ethernet; los almacenes 3 y el resto únicamente 2 interfaces. El motivo es que se implementará teaming/bonding para aumentar aún más si cabe la disponibilidad. En el caso de los almacenes, sólo es interesante hacer el bonding en la red backend, ya que la conexión a la red de servidores es indiferente para el funcionamiento del sistema.

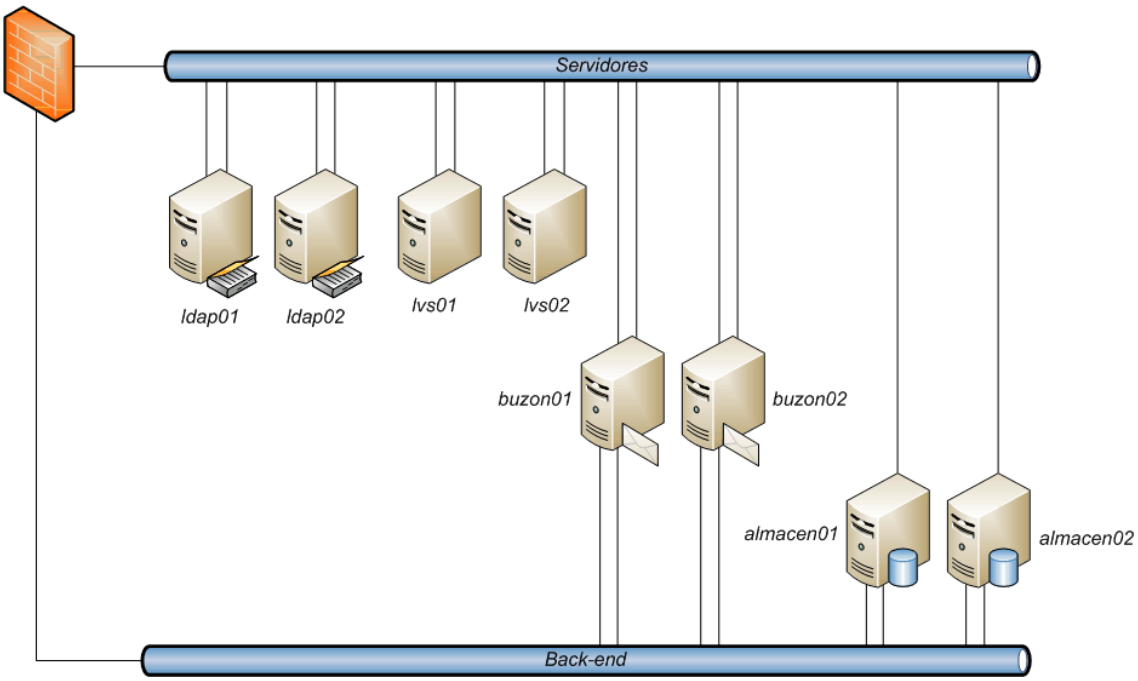

Ilustración 33: Servidores del sistema.

Por último, la asignación de direcciones IP a los servidores será la siguiente:

Servidor	Interfaz	Red	Dirección IP	Nombre
ldap01	1	Servidores	1.1.1.22	ldap01.esi.us.es
	2			
ldap02	1	Servidores	1.1.1.23	ldap02.esi.us.es
	2			
lvs01	1	Servidores	1.1.1.2	lvs01.esi.us.es
	2			

lvs02	1	Servidores	1.1.1.3	lvs02.esi.us.es
	2			
buzon01	1	Servidores	1.1.1.12	buzon01.esi.us.es
	2			
	3	Back-end	192.168.1.12	bbuzon01.esi.us.es
	4			
buzon02	1	Servidores	1.1.1.13	buzon02.esi.us.es
	2			
	3	Back-end	192.168.1.13	bbuzon02.esi.us.es
	4			
almacen01	1	Back-end	192.168.1.253	balmacen01.esi.us.es
	2			
	3	Servidores	1.1.1.253	almacen01.esi.us.es
almacen02	1	Back-end	192.168.1.254	balmacen02.esi.us.es
	2			
	3	Servidores	1.1.1.254	almacen02.esi.us.es

Tabla 4: Asignación de direcciones IP

Y los parámetros de las redes son los siguientes:

Red	Servidores	Back-end
Dirección de red	1.1.1.0	192.168.1.0
Máscara de red	255.255.255.0	255.255.255.0
Dirección de broadcast	1.1.1.255	192.168.1.255
Puerta de enlace	1.1.1.1	192.168.1.1
Rango	1.1.1.1 – 1.1.1.254	192.168.1.1-192.168.1.254

Tabla 5: Parámetros de redes.

4.2 Arquitectura propuesta.

Una vez definida la solución, describiremos con cierta profundidad la implementación y las principales características de cada uno de los 4 módulos del sistema.

4.2.1 Nodos Buzón.

El buzón, también llamado estafeta de correo, es el nodo principal del sistema de correo, y se encarga de procesar (recibir, enviar, almacenar) los mensajes de correo electrónico. El componente principal del mismo es el servidor de correo (SMTP, POP e IMAP), el cual se apoya en otras herramientas para el filtrado de correo y el interfaz con el usuario.

4.2.1.1 Exim.

El servidor SMTP empleado es exim 4, el cual se encargará de dirigir debidamente los correos electrónicos a su destino, ya sea al exterior o a los buzones determinados. Exim es un MTA (Mail Transfer Agent) para servidores tipo unix diseñado para funcionar continuamente en servidores que dan servicio de forma ininterrumpida y con un elevado flujo de mensajes. Maneja los mensajes de correo electrónico según la RFC 2822, y se distribuye históricamente con la distribución debian de Linux, aunque puede ser compilado para la mayoría de los sistemas operativos tipo unix.

La configuración de exim se basa en un fichero de texto de configuración, dividido en varias secciones que comentaremos más adelante. Se ha adoptado el interfaz de línea de comandos de sendmail, por lo que el comando /usr/bin/exim puede ser utilizado en vez de /usr/bin/sendmail con idéntica sintaxis.

Cabe destacar que exim soporta conexiones SMTP encriptadas con TLS/SSL, utilizando el comando STARTTLS en SMTP según RFC 2487.

4.2.1.1.1 Terminología.

Parte local (local part): se define en la RFC 2822, se refiere a la parte de la dirección de correo anterior a la arroba (@).

Dominio (domain): es la parte de la dirección de correo que va después de la arroba (@).

Cabecera (header): es la primera parte del mensaje, que contiene los campos “From”, “To”, “Subject”, etc.

Cuerpo (body): son los datos que el remitente quiere transmitir. Es la última parte del mensaje, y esta separada de la cabecera por una línea en blanco.

Rebote (bounce): ocurre cuando un mensaje no se puede entregar y normalmente se devuelve al remitente en un “informe de no entrega”

Diferido (defer): ocurre cuando un mensaje no se puede entregar en un determinado momento, por lo que se espera para entregarlo pasado un tiempo.

Sobre (envelope): un mensaje tiene un sobre asociado, además de una cabecera y un cuerpo. El sobre contiene la dirección del remitente y la(s) dirección(es) de los destinatarios. Estas son las direcciones usadas por el MTA para la entrega o el rebote de mensajes, en vez de las que aparecen en la cabecera.

Cola (queue): conjunto de mensajes que esperan a ser entregados.

Spool : directorio dónde se mantienen los mensajes de la cola.

4.2.1.1.2 Funcionamiento.

La filosofía de exim es trabajar eficientemente en servidores de correo de forma continuada, manejando todo tipo de mensajes de correo electrónico. En dichas circunstancias, la mayoría de los mensajes pueden ser entregados en el momento, y el resto son encolados en una única cola, independientemente del dominio y el host de destino.

Política de control.

Hoy en día uno de los aspectos más importantes de un MTA es la política de control, que evite que el servidor actúe como un “open relay”, y pueda ser utilizado por terceras personas para el envío de correo.

Este control se realiza mediante listas de control de acceso (ACLs, Acess Control Lists), en las cuales se especifica, mediante ciertas condiciones que especifica el administrador, si se permite o se deniega el acceso.

Por otra parte, tanto el administrador como cada usuario pueden definir una serie de filtros para correos electrónicos.

Identificación de mensajes.

A cada correo que exim maneja, se le adjudica un "message id". Se trata de un número de 16 cifras en base 62, separado en 3 partes por guiones, por ejemplo: `16VDhn-0001bo-D3`. Estas 3 partes son:

- Los primeros seis caracteres representan la hora a la cual se recibe el mensaje. Concretamente es el número de segundos desde el comienzo de la época.
- Los segundos seis caracteres especifican el id del proceso que recibe el mensaje.
- Los últimos dos caracteres representan una fracción del tiempo de recepción, con la idea de que dos mensajes recibidos en el mismo segundo, no tengan el mismo "message id".

Procesado de mensajes.

Exim recibe correos mediante el protocolo SMTP sobre una conexión TCP/IP, y por defecto comienza con el reparto en el momento que el correo llega, aunque puede ser configurado de otra manera, lo cual puede ser interesante si la carga es muy alta.

Una vez que exim acepta el mensaje, se escriben dos ficheros en el directorio de spool. Seguimos con el ejemplo:

- `16VDhn-0001bo-D3-H` **:** contiene la cabecera y el sobre.
- `16VDhn-0001bo-D3-D` **:** contiene los datos (cuerpo).

El correo permanece en el spool hasta que es entregado bien a su destinatario o bien a su remitente como rebote. Si no es posible ninguna de las dos opciones, el correo se marca como congelado, y no se realizan más intentos de entrega. Al mismo tiempo, exim escribe en su fichero de bitácora (log) cada intento de entrega.

Los procesos de entrega en exim se denominan drivers, y existen dos tipos: routers y transports.

- El router se encarga de determinar como se debe entregar un determinado mensaje. Puede enviar el mensaje a un transport, convertir una dirección a otra(s) o incluso rebotar el mensaje.
- El transport se encarga de copiar el mensaje del spool a su destino correspondiente.

En el fichero de configuración se definen una serie de routers, y el mensaje recorre dicha lista de forma secuencial y ordenada, hasta que uno de ellos procesa el mensaje (si cumple sus precondiciones). Si ninguno de ellos es capaz de procesarlo, el mensaje se reintentará según la política configurada.

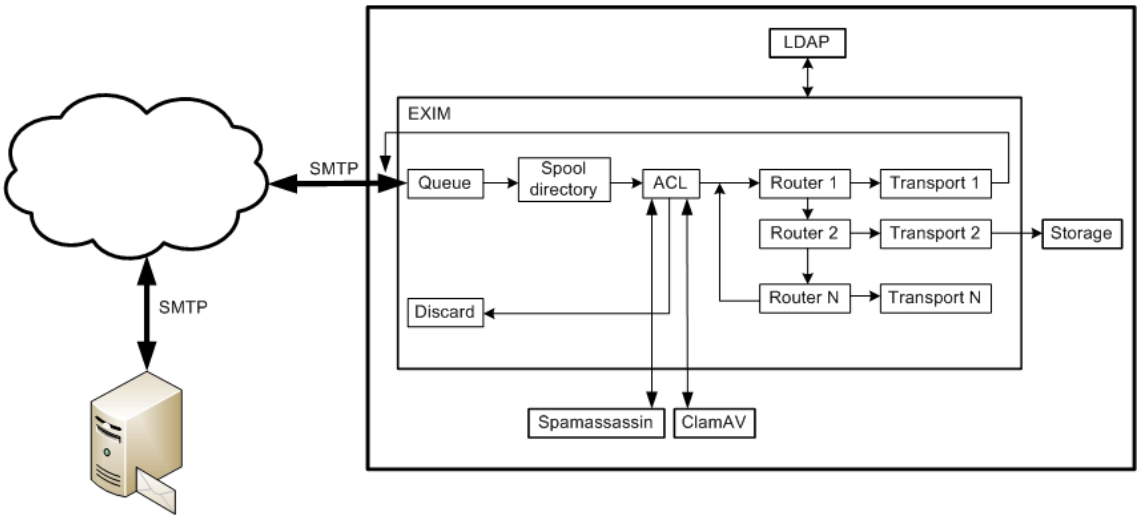

Ilustración 34: Procesado de mensajes en exim.

Durante todo el proceso está disponible la integración con LDAP, que puede ser utilizada para la comprobación de la existencia del usuario y para la búsqueda del directorio dónde almacenar los mensajes de correo, así como la integración con otras herramientas para filtrado como antivirus y antispam.

4.2.1.1.3 Configuración.

Exim utiliza un único fichero de texto para su configuración, lo cual simplifica en gran medida dichas tareas. Este fichero está dividido en varias partes, y las opciones generales deben estar siempre al principio del fichero. Un ejemplo de varias opciones generales son las siguientes:

```
domainlist local_domains = @      # dominios de entrega local
domainlist relay_to_domains =     # dominios para reenvío
hostlist relay_from_hosts = 127.0.0.1   # hosts de reenvio permitido

acl_smtp_rcpt = acl_check_rcpt    # se van a usar ACLs
acl_smtp_data = acl_check_data

ignore_bounce_errors_after = 2d   # tiempo para descartar un rebote
timeout_frozen_after = 7d     # tiempo para descartar un congelado
```

A continuación, aparecen el resto de las partes, las cuales son opcionales. Veremos un ejemplo de las más importantes:

- ACL: listas de control de acceso para el correo entrante.

```
begin acl

deny message = Restricted characters in address
domains = +local_domains
local_parts = ^[.] : ^.*[@%!/|]
```

En este ejemplo, se deniega la entrada de un mensaje con los caracteres @, %, !, / ó | en el local part.

- Authenticators: opciones de configuración para los drivers de autenticación.
- Routers: opciones de configuración para los drivers de enrutado, los cuales procesan el mensaje y deciden como son entregados.

```
begin routers

localuser:
driver = accept
check_local_user
transport = local_delivery
```

Aquí, el router acepta mensajes para usuarios locales, y se los adjudica al transport "local_delivery".

- Transports: opciones de configuración para los drivers de transporte, los cuales definen mecanismos para copiar los mensajes a sus destinos correspondientes.

```
begin transports

local_delivery:
driver = appendfile
file = /var/mail/$local_part
delivery_date_add
envelope_to_add
return_path_add
```

Este transport copia el mensaje al directorio de correo del usuario correspondiente, el cual se halla en /var/mail.

- Retry: reglas de reintentos para mensajes que no se pueden entregar en un primer momento. Si no existe esta sección, no se reintentará ninguna entrega.

```
begin retry

* * F,2h,15m; G,16h,1h,1.5; F,4d,6h
```

Después de un fallo en la entrega, se volverá a intentar cada 15 minutos durante las dos próximas horas. Posteriormente, a intervalos de 1 hora incrementándose con factor 1.5 durante las siguientes 16 horas. Finalmente, cada 6 horas durante 4 días. Si después de todo ello no se entrega, el mensaje se rebota.

- Rewrite: reglas para la traducción de mensajes entrantes.

4.2.1.1.4 Implementación.

Aunque exim está incluido en la instalación básica de debian, el demonio que incluye dicha instalación es el denominado "light", pero nosotros vamos a necesitar el demonio pesado ("heavy") con lo cual tenemos que instalar dicho paquete:

```
#aptitude install exim4-daemon-heavy
```

Una vez instalado, sólo tenemos que modificar un único fichero de configuración: /etc/exim4/exim4.conf. En el anexo 6.1 se muestra el fichero completo, pero a continuación vamos a analizar las líneas más importantes. En las opciones generales tenemos:

```
domainlist local_domains = lsearch;/etc/exim4/dominiosESI

ldap_default_servers = ldap.esi.us.es

av_scanner = clamd:127.0.0.1 3310

spamd_address = 127.0.0.1 783

message_size_limit = 6M
recipients_max = 70
```

La primera línea define los dominios que son locales, para realizar la entrega local. Esto permite el uso de dominios virtuales, introducimos todos los dominios locales (por ejemplo esi.us.es, alumnos.es.us.es, pdi.us.es, pda.esi.us.es, listas.esi.us.es,...) en el fichero /etc/exim4/dominiosESI.

En las siguientes líneas definimos los parámetros para acceder al servicio LDAP, el servicio antivirus (clam av) y el servicio antispam (spamassassin). En el primer caso se trata de los servidores de ldap, y en los otros dos, servicios locales en el propio servidor por los puertos 3310 y 783 respectivamente.

El resto de líneas limita el tamaño del mensaje en 6MB, y el número de destinatarios de un único mensaje a 70.

Ahora pasamos al apartado de listas de control de acceso (ACL).

```
acl_check_data:
  discard message = This message contains virus ($malware_name).
          malware = *
  warn    message = X-Spam-Score: $spam_score ($spam_bar)
          condition = ${if <{$message_size}{32k}{1}{0}}
          spam              =              ${lookup              ldap
{ldap:///dc=esi,dc=us,dc=es?uid?sub?(uid=${lc:${extract{1}{=@}{$rec
ipients}{$value}{mail}}})}}:true
  warn    message = Subject: -SPAM- $h_Subject
          condition = ${if <{$message_size}{32k}{1}{0}}
          spam              =              ${lookup              ldap
{ldap:///dc=esi,dc=us,dc=es?uid?sub?(uid=${lc:${extract{1}{=@}{$rec
ipients}{$value}{mail}}})}}
          condition = ${if <{$spam_score_int}{10}{1}{0}}
  deny    message = This message scored $spam_score spam points
          spam = nobody
          condition = ${if >{$spam_score_int}{10}{1}{0}}
  accept
```

Aquí se comprueban los datos del mensaje. En primer lugar se descarte el mensaje si clamav ha detectado virus. A continuación se analiza el spam; spamassassin ha puntuado

el mensaje y ahora tenemos que decidir qué hacer con él en función de dicha puntuación, la cual es incluida en la cabecera con el atributo "X-Spam-Score".
En primer lugar, sólo se tienen en cuenta mensajes de tamaño 32KB o menor, ya que la gran mayoría de correos spam son de pequeño tamaño, y de esta forma evitamos analizar muchos correos de tamaño mayor que a priori no son sospechosos.

En nuestro caso se han tomado dos políticas: advertir en aquellos correos con una puntuación menor que 10, incluyendo el texto "-SPAM-" en el asunto; y descartar directamente los mensajes con una puntuación mayor que 10.

A continuación vemos el apartado de routers.

```
externos:
  debug_print = "ROUTER: ESI externos -> $local_part@$domain"
  driver = manualroute
  domains = !+local_domains
  route_list = * correo.us.es
  transport = remote_smtp
  no_more
```

Este router se encarga de enviar correos a usuarios externos (los que no pertenecen a los dominios locales). Para ello utilizará el transport smtp hacia el servidor correo.us.es, que será la puerta de salida de correos electrónicos para nuestro sistema.

```
entrega:
  debug_print = "ROUTER: ESI entrega -> $local_part@$domain"
  driver = accept
  domains = +local_domains
  condition               =               ${lookup               ldap
{ldap:///dc=esi,dc=us,dc=es?mail?sub?(mail=$local_part@$domain)}}
  transport = maildir_home
```

El router "entrega" se encarga de repartir el correo a los usuarios locales. Para ello comprueba mediante LDAP que el usuario existe en el sistema, y llama al transport de maildir para la propia entrega.

```
alternativa:
  debug_print = "ROUTER: ESI alternativa -> $local_part@$domain"
  driver = accept
  domains = +local_domains
  condition               =               ${lookup               ldap
{ldap:///dc=esi,dc=us,dc=es?mailAlternateAddress?sub?(mailAlternate
Address=$local_part@$domain)}}
  transport = alternativa_home
```

Este router es idéntico al anterior, con la salvedad de que comprueba los alias de correo. A cada usuario se le pueden definir varios alias, de forma que diferentes direcciones de correo electrónico pertenezcan a la misma cuenta. Como veremos cuando definamos el directorio LDAP, hay un atributo específico para este cometido, que se traduce en la diferencia entre este router y el anterior (mailAlternateAddress).

También se han definido routers para filtros de administrador y de usuario, los que permiten realizar ciertas acciones a los correos mediante determinadas condiciones. Un ejemplo podría ser clasificar correos de un determinado remitente en una carpeta concreta.

Ahora pasamos a repasar los transports.

```
maildir_home:
  debug_print     =     "TRANSPORT:     ESI     maildir_home     ->
$local_part@$domain"
  driver = appendfile
  delivery_date_add
  envelope_to_add
  return_path_add
  maildir_format
  maildir_tag = ,S=$message_size
  maildir_use_size_file = true
  maildir_quota_directory_regex = ^(?:cur|new|\..*)$
  quota_size_regex = ,S=(\d+)
  quota                =                ${lookup                ldap
{ldap:///dc=esi,dc=us,dc=es?mailQuotaSize?sub?(mail=$local_part@$do
main)}}
  quota_warn_threshold = 90%
  quota_warn_message = "\
  To: $local_part@$domain\n\
  Subject: Su buzón de correo esta al límite de su capacidad\n\n\
  Este mensaje ha sido creado automáticamente \
  por el sistema de Correo de la ESI.\n\n\
  La capacidad de su buzón de correo está llegando al límite \
  establecido por su administrador de correo.\n\n \
  Si se alcanza este límite no podrá recibir nuevos correos hasta
\n \
  que libere espacio en su buzón.\n\n"
  directory              =              ${lookup              ldap
{ldap:///dc=esi,dc=us,dc=es?mailMessageStore?sub?(mail=$local_part@
$domain)}}/Maildir
  create_directory
  mode = 0600
```

Se trata del transport que realiza la entrega local en el maildir del usuario. En primer lugar formatea el mensaje para almacenarlo, y comprueba la ocupación de la cuota del usuario. Si la ocupación es mayor al 90%, manda un mensaje al usuario advirtiéndole de tal situación, y si la ocupación alcanza el 100% el mensaje no sería entregado.

En última instancia, se busca mediante LDAP cual es el directorio del usuario dónde almacenar el mensaje, y si el directorio no existe, se crea para poder guardar el correo.

Según el formato maildir, dentro del directorio del usuario se crean 3 subdirectorios: tmp, cur y new. En el momento en que llega un nuevo correo, se guarda en un fichero con un nombre único (que incluye la hora, el nombre de equipo, el id del proceso y números aleatorios) en el directorio `tmp`.

Cuando se acaba de almacenar el mensaje en `tmp`, se enlaza desde el directorio `new` y después se desenlaza el fichero en `tmp`. Así se evita que el cliente de correo electrónico lea un mensaje parcial mientras se está repartiendo.

Un proceso similar ocurre cuando el cliente de correo electrónico encuentra un mensaje en el directorio `new`: lo mueve a `cur` (usando la misma estrategia de enlazar y desenlazar) y le añade al nombre del fichero un sufijo informativo antes de leerlo. El sufijo informativo consiste en dos puntos (para separar el nombre único del fichero de la información siguiente), un '2', una coma y varios indicadores.

4.2.1.2 Courier.

Los servidores IMAP y POP permiten al usuario acceder a su buzón de correo y en consecuencia a los mensajes almacenados en él. Básicamente la diferencia entre ellos reside en que mediante IMAP hacemos una conexión con el servidor, que nos transmite una lista con los mensajes que hay en el buzón, y por demanda podemos acceder a cualquiera y ver su contenido. En cambio, mediante POP se realiza una descarga de los mensajes de correos, a los que se accede de forma local, mientras que en el caso IMAP es una conexión remota.

Courier es uno de los servidores POP e IMAP más populares para Linux, con múltiples posibilidades de autenticación, y disponibilidad de conexiones seguras SSL. Son necesarios en total 4 demonios que escucharán por diferentes puertos TCP.

- imapd: TCP 143.
- imapd-ssl: TCP 993.
- pop3d: TCP 110.
- pop3d-ssl: TCP 995.

Además, es necesario un demonio de autenticación para poder establecer las conexiones, que en nuestro caso haremos mediante consulta al servidor LDAP. La instalación necesita por lo tanto, los siguientes paquetes:

```
# aptitude install courier-authdaemon courier-base courier-imap
courier-imap-ssl
# aptitude install courier-pop courier-pop-ssl courier-ssl
```

En cuanto a la configuración, vamos a centrarnos en los casos de imapd y pop3d, obviando las versiones ssl de ambos servicios, cuya principal diferencia radica en la expedición de certificados X.509. El fichero de configuración para imapd es /etc/courier/imapd, del cual destacamos lo más importante:

```
PORT=143
MAXDAEMONS=4000
MAXPERIP=4
AUTHMODULES="authdaemon"
IMAP_IDLE_TIMEOUT=60
MAILDIRPATH=Maildir
```

Se especifica el puerto dónde va a escuchar el servidor; el número máximo de demonios disponibles y el número máximo de ellos para cada dirección IP. También se especifica el módulo de autenticación que se va a utilizar, el tiempo de espera de la conexión IMAP y el nombre de la carpeta maildir dentro de la carpeta de cada usuario. En el caso de pop tenemos el fichero /etc/courier/pop3d:

```
MAXDAEMONS=40
MAXPERIP=4
AUTHMODULES="authdaemon"
PORT=110
MAILDIRPATH=Maildir
```

Que son análogos al caso de imapd. Para la autenticación, tenemos dos ficheros: el principal es /etc/courier/authdaemonrc, dónde destacamos:

```
authmodulelist="authldap"
daemons=5
```

Aquí simplemente se define un único módulo de autenticación disponible (LDAP), y se limita el número de demonios a 5. Por otra parte se encuentra el fichero /etc/courier/authldaprc :

```
LDAP_SERVER       ldap.correo.esi.us.es
LDAP_PORT         389
LDAP_PROTOCOL_VERSION   3
LDAP_BASEDN       dc=esi,dc=us,dc=es
LDAP_MAIL         uid
LDAP_HOMEDIR      mailMessageStore
LDAP_MAILDIRQUOTA mailQuotaSize
LDAP_FULLNAME     cn
LDAP_CRYPTPW      userPassword
```

En este fichero se definen los parámetros de conexión al servidor ldap (nombre, puerto, versión, raíz) y los atributos que se van a utilizar para el funcionamiento (dirección de correo, directorio de usuario, tamaño de la cuota de usuario, nombre completo y contraseña).

4.2.1.3 ClamAV.

El análisis de correo electrónico es una tarea de vital importancia para cualquier corporación. No sólo es necesario poseer una herramienta eficiente para ello, sino que además es imperativo que dicha herramienta sea actualizada constantemente, debido al dinamismo en el “mercado” de los virus informáticos.

Por lo tanto, en el despliegue de Clam AV debemos cubrir dos objetivos: proporcionar a exim una herramienta de análisis de virus, y establecer un mecanismo de actualización eficiente para hacer efectivo el análisis.

Se va a incluir un nuevo demonio “clamd” que utiliza el puerto TCP 3310 al que exim accede como servicio local. La instalación requiere el siguiente comando:

```
# aptitude install clamav-daemon clamav-freshclam
```

El fichero de configuración de la herramienta es /etc/clamav/clamd.conf en el cual podemos destacar las siguientes líneas:

```
LogFile /var/log/clamav/clamd.log
DatabaseDirectory /var/lib/clamav
TCPSocket 3310
```

Aquí se define el fichero de log, el directorio dónde se almacena la base de datos de virus y el puerto TCP que se va a utilizar para el demonio de Clam AV.

Nos resta configurar la utilidad de actualización, denominada freshclam, lo cual hacemos en el fichero /etc/clamav/freshclam.conf, y dónde comentamos las siguientes líneas:

```
DatabaseDirectory /var/lib/clamav
UpdateLogFile /var/log/clamav/freshclam.log
DatabaseMirror database.clamav.net
MaxAttempts 5
Checks 24
```

Se vuelve a definir el directorio de la base de datos y el fichero de log. Se indica el servidor al cual se le va a hacer la petición de actualización, el número de reintentos por cada conexión, y el número de actualizaciones que se hacen cada día. Con este último parámetro puesto a 24, haremos una actualización de la base de datos cada hora.

4.2.1.4 Spamassassin.

Una vez aceptado que el spam es un problema muy importante que afrontar a la hora de construir un sistema de correo electrónico, hay que pasar a la acción y desarrollar mecanismos para evitarlo.

Spamassassin se encarga de analizar el mensaje de correo electrónico y asignarle una puntuación en función de múltiples factores. Esta puntuación será utilizada por exim en sus ACL (listas de control de acceso) para desestimar o no la entrega del correo. Concretamente, las pruebas disponibles son las siguientes:

- Análisis de la cabecera.
- Identificación de palabras en el cuerpo.
- Filtros bayesianos. Reconoce patrones negativos o positivos desde una base de datos para clasificar el correo.
- Listas blancas y negras automáticas.
- Listas blancas y negras manuales.
- Bases de datos públicas de spam.
- RBL (Real-time Blackhole List): lista de agujeros negros. Comprueban el servidor de correo del que proviene el mensaje.
- Conjunto de caracteres. Revisa el juego de caracteres usado, que puede ser occidentales u oriental (chino, japonés, coreano,...).

Cada una de dichas pruebas asigna una puntuación positiva o negativa. Una vez realizadas todas las pruebas, se obtiene una puntuación definitiva y el correo se califica como spam o como ham según un umbral determinado.

La integración total de spamassassin con exim nos simplificará el despliegue de esta herramienta y de configurarla de manera rápida y eficaz.

Se incluye un nuevo demonio "spamd", el cual utiliza el puerto TCP 783 al que exim accede como servicio local. La instalación se reduce, como de costumbre, a:

```
# aptitude install spamassassin
```

Y la configuración también es relativamente sencilla, está recogida en un fichero.

```
# /etc/spamassassin/user_prefs

required_score    5
use_bayes         1

whitelist_from    *@*.us.es
blacklist_from    *@deal-seeker.com

score SYMBOLIC_TEST_NAME n.nn
```

En la primera línea establecemos el umbral de puntuación para considerar un mensaje como spam. En la segunda línea estamos incluyendo el análisis bayesiano. A continuación estamos definiendo una lista blanca manual (para todos los dominios *.us.es) y una lista negra manual (para el dominio deal-seeker.com).

Con la última línea (línea de ejemplo) asignamos una puntuación determinada a una prueba determinada, lo que nos ofrece mayor granularidad a la hora de establecer el análisis anti-spam.

Por otra parte, conviene asegurarse que spamassassin está habilitado, para lo cual en el fichero /etc/default/spamassassin debería estar la siguiente línea:

```
ENABLED=1
```

4.2.1.5 Squirrelmail.

Squirrelmail es un webmail basado en php que se conecta al servidor IMAP para acceder a los buzones de correo. Asimismo, usa un servidor web (normalmente apache) para su publicación como página web. No vamos a comentar nada relativo a la configuración de apache ya que se sale del alcance de este proyecto. La instalación de squirrelmail en debian es tal que:

```
# aptitude install squirrelmail
```

Y la configuración básica del mismo se hace en el fichero /etc/squirrelmail/config.php, dónde destacamos lo siguiente:

```
$smtpServerAddress = 'smtp.correo.esi.us.es';
$smtpPort = 25;
$sendmail_path = '/usr/sbin/exim4';

$imapServerAddress = 'imap.correo.esi.us.es';
$imapPort = 143;
$imap_server_type = 'courier';

$trash_folder = 'INBOX.Papelera';
$sent_folder  = 'INBOX.Enviados';
$draft_folder = 'INBOX.Borradores';

$ldap_server[0] = Array(
      'host' => 'ldap.esi.us.es',
      'base' => 'dc=esi,dc=us,dc=es',
      'name' => 'Directorio de la ESI'
);
```

En las primeras líneas definimos el servidor SMTP, y en las siguientes se hace lo propio con el servidor IMAP. A continuación se definen los nombres de las carpetas papelera, enviados y borradores. Por último definimos los parámetros del servidor LDAP.

4.2.2 Nodos Directorio.

Un directorio es una base de datos, pero en general contiene información más descriptiva y basada en atributos, la cual normalmente se lee mucho más de lo que se escribe. El objetivo principal de los directorios es proporcionar una respuesta rápida a operaciones de búsqueda o consulta.

4.2.2.1 LDAP.

LDAP (Lightweight Directory Acces Protocol, Protocolo Ligero de Acceso a Directorios) es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio. LDAP es un tipo de base de datos, pero no es una base de datos relacional, ya que no está diseñada para procesar cientos o miles de cambios por minuto, sino para realizar lecturas de datos de forma muy eficiente.

El servicio de directorio LDAP se basa en un modelo cliente-servidor, en el que uno o más servidores LDAP contienen los datos que conforman el árbol de directorio LDAP o base de datos troncal. El cliente LDAP se conecta con el servidor LDAP y le hace una consulta. Posteriormente el servidor contesta con la respuesta
Las principales características de un directorio LDAP son:

- Es muy rápido en la lectura de registros.
- Permite replicar el servidor de forma muy sencilla y económica.
- Muchas aplicaciones de todo tipo tienen interfaces de conexión a LDAP y se pueden integrar fácilmente.
- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
- Usa un sistema jerárquico de almacenamiento de información.
- Permite múltiples directorios independientes.
- Funciona sobre TCP/IP y SSL/TLS.

- La mayoría de aplicaciones disponen de soporte para LDAP.
- La mayoría de servidores LDAP son fáciles de instalar, mantener y optimizar.

Dadas las características de LDAP sus usos más comunes son:

- Directorios de información. Por ejemplo bases de datos de empleados organizados por departamentos (siguiendo la estructura organizativa de la empresa) ó cualquier tipo de páginas amarillas.
- Sistemas de autenticación/autorización centralizada. Grandes sistemas donde se guarda gran cantidad de registros y se requiere un uso constante de los mismos.
- Sistemas de correo electrónico. Grandes sistemas formados por más de un servidor que accedan a un repositorio de datos común.
- Grandes sistemas de autenticación basados en RADIUS, para el control de accesos de los usuarios a una red de conexión o ISP.
- Servidores de certificados públicos y llaves de seguridad.
- Autenticación única ó "single sign-on" para la personalización de aplicaciones.

4.2.2.2 Estructura de un directorio LDAP.

Tradicionalmente se han usado las estructuras de árbol para jerarquizar la información contenida en un medio. El ejemplo más claro es la estructura de carpetas (directorios) de un sistema operativo, que nos permite ordenar la información en subdirectorios que contienen información muy específica.

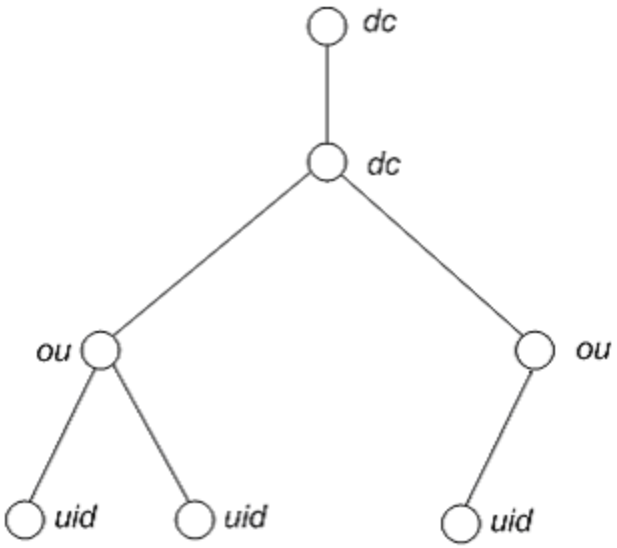

Ilustración 35: Estructura de un directorio LDAP.

En este momento es conveniente aclarar la terminología empleada en LDAP:

- Entry (Entrada). Una entrada es una colección de atributos que tienen un único y global DN (Distinguished Name, Nombre Distintivo). En el diagrama anterior, los círculos son entradas.
- Atribute (Atributo). Los datos del directorio se representan mediante pares de atributo y su valor. Cada entrada que se introduzca en el directorio se define mediante la colección de atributos que determina la clase de objetos. Cada atributo tiene la definición de sintaxis que le corresponde, la cual describe el tipo de información que proporciona ese atributo.

- ObjectClass (Clase de objetos). Una clase de objetos define la colección de atributos que pueden usarse para definir una entrada.
- LDIF (LDAP Data Interchange Format, Formato de Intercambio de LDAP). Se trata de un fichero que se utiliza para importar y exportar información de un directorio, o para describir una serie de cambios que han de aplicarse al mismo. Este fichero almacena información en jerarquías de entradas orientadas a objeto.

Concretamente en el diagrama de ejemplo, observamos que hay entradas de tipo dc (domain controller), ou (organizational unit) y uid (user identification), las cuales se organizan en una estructura jerárquica. Por lo tanto, para obtener el nombre distintivo de una entrada (dn), hay que recorrer el árbol subiendo a niveles superiores hasta llegar a la entrada raíz.

4.2.2.3 Implementación de openldap.

En primer lugar, debemos instalar los paquetes necesarios en nuestro sistema:

```
# aptitude install slapd ldap-utils db4.2-util
```

A continuación, debemos configurar openldap con nuestros requisitos. Para ello hay que editar el fichero /etc/ldap/slapd.conf y modificar los siguientes parámetros:

```
suffix          "dc=esi,dc=us,dc=es"

rootdn "cn=Manager,o=administradores,dc=esi,dc=us,dc=es"
rootpw esi
updatedn "cn=Manager,o=administradores,dc=esi,dc=us,dc=es"

access to attrs=userPassword
        by      dn="cn=Manager,o=administradores,dc=esi,dc=us,dc=es"
write
        by sockname=127.0.0.1 read
        by peername=127.0.0.1 read
        by anonymous auth
        by self write
        by * none

access to dn.base="" by * read

access to *
        by      dn="cn=Manager,o=administradores,dc=esi,dc=us,dc=es"
write
        by * read
```

Y de esta forma concluye el despliegue de openldap. El demonio para iniciar y parar los servicios es /etc/init.d/slapd.

Para la administración del directorio hay soluciones en dos vertientes. En primer lugar, en modo texto consola, el paquete ldap-utils nos proporciona una serie de comandos. La otra posibilidad es utilizar herramientas gráficas, que simplifican la tarea del administrador; como ejemplo existen algunas basadas en software libre: ldapEditor ó J-explorer. En el anexo 6.6 se amplía la información sobre todas las herramientas de administración.

4.2.2.4 Replicación del directorio.

Para conseguir la alta disponibilidad en el servicio LDAP, es necesario ejecutar más de una instancia del demonio servidor openldap (slapd); es decir, tener un maestro y al menos un esclavo. Este arreglo provee una manera simple y efectiva de incrementar la capacidad y disponibilidad del servicio. El demonio de replicación asíncrona 'slurpd' provee la capacidad para que un maestro slapd propague los cambios al resto de las instancias de los esclavos slapd, implementando la replicación maestro/esclavo descrito. 'slurpd' corre en el mismo host que el maestro slapd, y utiliza el protocolo LDAP para actualizar la BD del esclavo. Un simple ejemplo de replicación, sería el siguiente:

1. El cliente LDAP realiza la petición para modificar al maestro slapd.
2. El maestro slapd realiza la operación, escribe los cambios en el archivo log de replicación y retorna éxito al cliente. Este archivo tiene formato LDIF.
3. 'slurpd' nota que una nueva entrada ha sido adjuntada al archivo log de replicación, lee este archivo y envía los cambios al esclavo slapd vía LDAP.
4. El esclavo slapd realiza la operación de modificación y retorna éxito al proceso 'slurpd'.

Por lo tanto, la arquitectura de directorio LDAP comprende un maestro y un esclavo. Para operaciones de lectura se debe atacar a la IP virtual del balanceo. Sin embargo, para operaciones de escritura se debe atacar exclusivamente al maestro. Esto es debido a que la replicación se realiza en un único sentido, desde el maestro al esclavo, y toda escritura en el esclavo no se replicaría en el maestro.

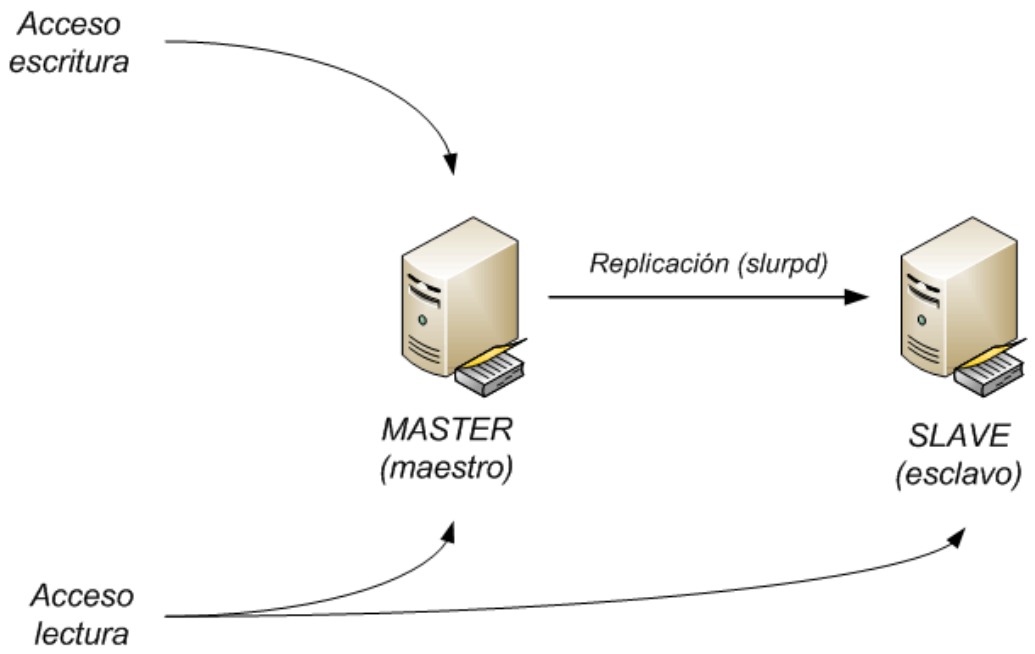

Ilustración 36: Replicación del directorio LDAP.

Se ha optado por balancear la carga mediante el algoritmo round robin las operaciones de lectura para aprovechar al máximo los recursos hardware disponibles en la infraestructura del directorio corporativo. Los nodos balanceadores se encargarán de esta tarea. La implementación se realiza concretamente en los siguientes pasos:

1. Configurar el maestro, en /etc/ldap/slapd.conf:

```
replogfile      /var/tmp/ldap/slurp.log
replica uri=ldap://ldap02.esi.us.es:389
binddn="cn=Manager,dc=esi,dc=us,dc=es"
bindmethod=simple credentials=esi
```

2. Configurar el esclavo, en /etc/ldap/slapd.conf, idéntico al maestro excepto los cambios del paso anterior.
3. Detener openldap en maestro, copiar la base de datos al esclavo (directorio /var/lib/ldap).
4. Arrancar slapd y slurpd en el maestro.

Es de esperar que en el futuro openldap implemente un mecanismo multimaster para mejorar el rendimiento y simplificar el funcionamiento de esta solución.

4.2.2.5 Formato de directorio empleado.

La utilización de un directorio LDAP en un sistema de correo electrónico requiere un diseño específico tanto de la estructura del mismo como de los atributos y las clases de objeto.

Como vemos en el siguiente diagrama, el atributo raíz será dc=esi (dc=us,dc=es), del que colgarán distintas unidades organizativas (ou) para los distintos usuarios: pas, pdi y alumnos. En el ejemplo, se han colgado los usuarios (uid) directamente de los anteriores ou, pero es posible (y quizás recomendable, debido al número de usuarios) crear unidades organizativas de nivel inferior (por ejemplo, departamentos) para una mejor administración de los usuarios.

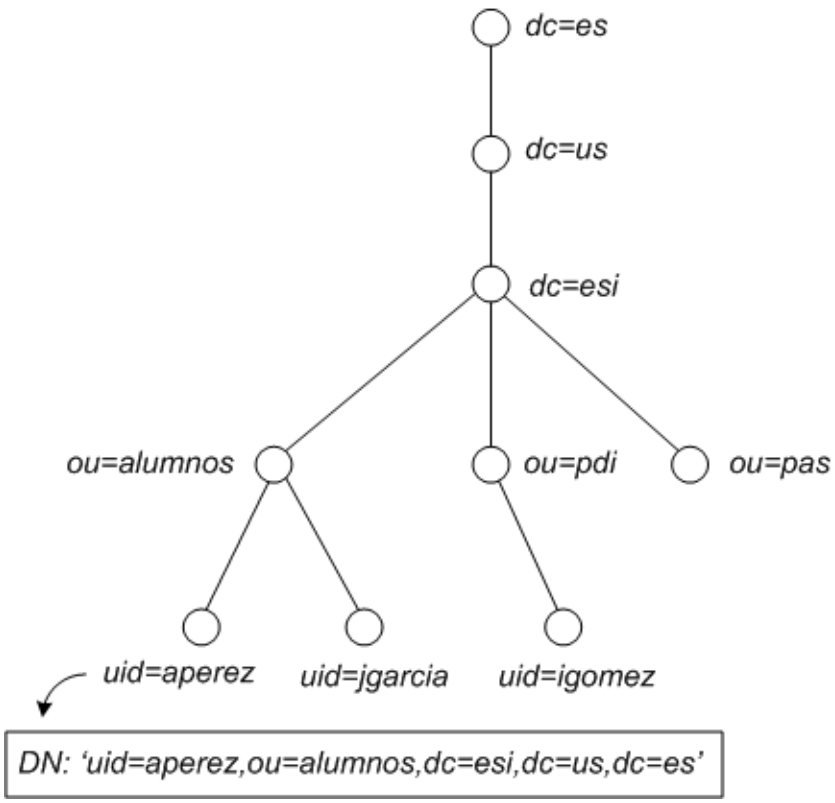

Ilustración 37: Estructura LDAP para la ESI.

En cuanto a la entrada de los usuarios, cada uno se identifica por su dirección de correo electrónico. Los atributos importantes de cada usuario son su contraseña, su límite de almacenamiento (quota), la ruta donde se almacenan los mensajes (buzón) y otros atributos adicionales. Como se puede verse esta entrada LDAP puede llegar a recibir cientos de consultas cada día (una por cada email recibido y una cada vez que el usuario se conecta mediante POP3 o webmail). No obstante el número de modificaciones diarias es muy bajo, ya que solo se puede cambiar la contraseña o dar de baja al usuario, operaciones ambas que no se producen de forma frecuente. A continuación vemos los atributos que componen la entrada tipo de cada usuario.

```
dn: uid=igomez,o=pdi,dc=esi,dc=us,c=es
mailMessageStore: /correo/i/igomez
sn: GOMEZ RUIZ
mailQuotaSize: 10000000
userPassword: {crypt}aanwIfEjVwE.A
mailAlternateAddress: igr@esi.us.es
mailAlternateAddress: __ALIAS2__
mailAlternateAddress: __ALIAS3__
mailAlternateAddress: __ALIAS4__
mailAlternateAddress: __ALIAS5__
mailReplyText: "Actualmente no estoy disponible"
mail: igomez@esi.us.es
deliveryMode: normal
mailQuotaCount: 2000
objectClass: top
objectClass: person
objectClass: qmailUser
objectClass: organizationalUnit
uid: igomez
mailHost: correo.esi.us.es
cn: IGNACIO GOMEZ RUIZ
```

De todos los atributos, destacamos:

mailMessageStore: ruta dónde se encuentra el maildir.
mailQuotaSize: tamaño máximo del buzón (bytes).
userPassword: contraseña.
mailAlternateaddress: alias de correo (dirección alternativa).
mail: dirección de correo.
mailquotacount: número máximo de elementos en el buzón.
mailhost: servidor de correo electrónico.

4.2.3 Nodos Balanceador.

4.2.3.1 Balanceo de carga.

El balanceo de carga es una técnica para distribuir la carga de trabajo entre diferentes subsistemas; este reparto se realiza mediante unos algoritmos que dividen el trabajo en función de ciertos factores. En nuestro caso esto se va a utilizar para los siguientes servicios:

- SMTP en buzon01 y buzon02.
- IMAP en buzon01 y buzon02.
- POP en buzon01 y buzon02.
- HTTPS en buzon01 y buzon02.
- LDAP en ldap01 ldap02.

Las direcciones IP virtuales son las siguientes:

- 1.1.1.11 para buzon01 y buzon02.
- 1.1.1.21 para ldap01 y ldap02.
- 1.1.1.4 para lvs01 y lvs02

4.2.3.1.1 Funcionamiento del balanceo de carga.

Es necesario definir una serie de conceptos previos:

- Director. Es el servidor que realiza el reenvío de tráfico.
- Real Server. Es el servidor que presta el servicio que se balancea.
- VIP (Virtual IP). Dirección IP virtual que proporciona el balanceador, a la que van destinadas las peticiones del cliente.
- RIP (Real IP). Dirección IP del real server.
- Algoritmos. En la forma en la que el tráfico se asigna entre todos los real servers. En nuestro caso vemos los 3 fundamentales.
    - Round robin. Se reparten equitativamente las peticiones entre todos.
    - Weighted round robin. Se reparten las peticiones entre todos mediante una proporción numérica.
    - Least connection. Se envía la petición siguiente al real server con menos conexiones.

4.2.3.1.2 LVS. Tipos de balanceo.

Al utilizar LVS como sistema de balanceo, surgen tres formas diferentes de implementación. Son las siguientes:

- NAT. Método que manipula la dirección de los paquetes origen y destino. A los paquetes recibidos desde el cliente se les cambia la IP de destino por las del servidor real elegido. Los paquetes de vuelta pasan de nuevo a través del director que hace el mapeo inverso.
- Direct Routing. Los paquetes que llegan desde el cliente se redirigen directamente al real server. El paquete IP no sufre ninguna modificación, por lo que los real servers tienen que configurarse para aceptar tráfico dirigido a la VIP (el director sí modifica la dirección MAC destino). El real server debe mandar las respuestas directamente al usuario final, por lo que el Director no está en el camino de vuelta.
- IPTunnel. Permite que los paquetes que van a una determinada IP puedan ser redirigidos a otra, que normalmente está en otra red. Su comportamiento es muy similar al Direct Routing, excepto que para redirigirlos son encapsulados en nuevos paquetes IP. La principal ventaja es que los real servers pueden estar en distintas redes.

En nuestro caso el método elegido es NAT. Aunque es el método que va a suponer más carga de trabajo para el servidor (es el único método en el que los paquetes de vuelta pasan también por el director), permite balancear aplicaciones de cualquier sistema operativo, y sobre todo permite utilizar más de un director, lo cual es esencial para este proyecto.

4.2.3.1.3 Ejemplo de LVS NAT.

Vamos a desarrollar un ejemplo de cómo funciona realmente LVS-NAT.

En esta situación, todo el tráfico que venga desde Internet hacia la dirección 1.1.1.1:80 será balanceada a las direcciones 1.1.1.2:80 y 1.1.1.3:8080, mientras que todo el tráfico que venga desde Internet hacia la dirección 1.1.1.1:21 será balanceada a las direcciones 1.1.1.3:21 y 1.1.1.4:2121. La re-escritura de paquetes funcionará de la siguiente manera:

El paquete de entrada tendrá como origen y destino las siguientes direcciones:

Origen 192.168.1.1:3456 Destino 1.1.1.1:80

Ahora el balanceador de carga (director) elegirá de acuerdo con su algoritmo de balanceo a un servidor real, por ejemplo 1.1.1.3:8080. Entonces el paquete será re-escrito y enviado (forwarded) al servidor como:

Origen 192.168.1.1:3456 Destino 1.1.1.3:8080

Y la respuesta que tendrá el balanceador de carga será:

Origen 1.1.1.3:8080 Destino 192.168.1.1:3456

Finalmente el paquete será re-escrito de nuevo antes de ser devuelto al cliente, y quedará de la siguiente forma:

Origen 1.1.1.1:80 Destino 192.168.1.1:3456

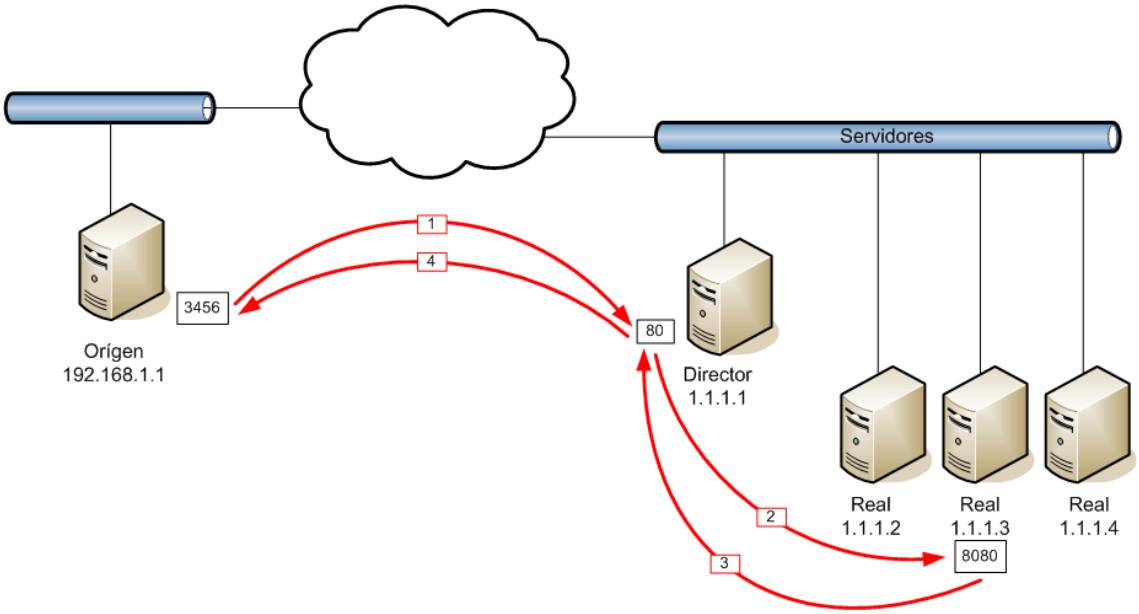

Ilustración 38: Ejemplo de LVS NAT.

4.2.3.1.4 Implementación de LVS.

En primer lugar, debemos cargar el módulo del kernel ipvs, que nos proporciona el balanceo de carga. A partir de la versión 2.4.24 del kernel de linux se incluye el soporte para ipvs.

```
# modprobe ip_vs
```

A continuación instalamos el paquete necesario:

```
# aptitude install ldirectord
```

Hay que modificar configuración de red tanto en el director (balanceador) como en los real servers. El asunto es que el director debe poder reenviar paquetes, y los real servers deben tener como puerta de enlace por defecto el propio director, para que se haga el reenvío de vuelta al cliente.

En el Director, hay que incluir lo siguiente en el fichero /etc/sysctl.conf:

```
net.ipv4.ip_forward = 1
```

Posteriormente hay que ejecutar sysctl –p para que se haga efectivo, o reiniciar la máquina. Por otra parte, en el director también debemos incluir la VIP como interfaz virtual en /etc/network/interfaces:

```
auto eth0:0
iface eth0:0 inet static
        address 1.1.1.2
        netmask 255.255.255.0
        network 1.1.1.0
        broadcast 1.1.1.255
        gateway 1.1.1.1
```

En el real server, hay que cambiar la puerta de enlace por defecto de la RIP en el fichero /etc/network/interfaces:

gateway 1.1.1.4

Por último, editamos y modificamos el fichero /etc/ha.d/conf/ldirectord.cf:

```
# Virtual Server for SMTP
virtual=1.1.1.11:25          # VIP
real=1.1.1.12:25 gate        # RIP 1
real=1.1.1.13:25 gate        # RIP 2
service=smtp                 # servicio definido
scheduler=rr                 # algoritmo de balanceo (rr=round robin)
protocol=tcp                 # protocolo tcp
checktype=negotiate
persistent=0                 # la persistencia es el tiempo que se define para conservar
                             #las conexiones con el mismo servidor real.
```

En el caso de la persistencia, para servicio de correo no tiene sentido, pero sí lo tiene para servicio http ó https. Por ejemplo, al definir una persistencia de 600 segundos, si un usuario inicia una conexión, en el momento que lleguen nuevas peticiones del mismo usuario antes de que pasen 600 segundos el contador se reinicia y estas peticiones se mandan al mismo servidor. Cuando se cumple este tiempo una nueva conexión de este usuario se puede redirigir a otro servidor.

Hemos especificado un único virtual server a modo de ejemplo. El fichero de configuración completo con todos los virtual servers del sistema se puede consultar en el anexo 7.7.

4.2.3.2 Clúster de alta disponibilidad.

En la actualidad linux es considerado un sistema operativo estable, pero el sistema operativo no es el único eslavón en la cadena de los sistemas de información. En la mayoría de los casos, los fallos en un sistema son provocados por un error de hardware o por un fallo humano.

Para evitar los fallos hardware, se implementa hardware con tolerancia a fallos (FT, Fault Tolerant), que tienen todos y cada uno de sus componentes redundados. El problema es que el coste de estos equipos es elevadísimo. Ahí es dónde surge el concepto de alta disponibilidad (HA, High Availability), que obtiene prestaciones cercanas al FT, pero con un coste inferior.

El fundamento de HA es utilizar diferentes subsistemas, en lugar de la utilización de un único sistema con FT. En el caso que nos ocupa, la HA se provee mediante un clúster de servidores. El quid de la cuestión es que resulta mucho más económico implementar dos servidores sin FT, que un único servidor con FT. Aún así, los resultados no son exactamente los mismos, ya que el tiempo fuera de servicio (provocado por un fallo) suele ser sensiblemente mayor en sistemas HA con respecto a sistemas FT.

Para construir un sistema HA, la regla básica es considerar los posibles puntos únicos de fallo (SPOF, Single Point Of Failure). Un SPOF puede ser, por ejemplo, una tarjeta de red, una controladora SCSI ó un servidor. Se deben evitar todos los SPOF posibles, si bien habrá que olvidarlos en un nivel superior, debido al excesivo coste que puede suponer; por ejemplo, si tenemos todos los equipos en el mismo edificio, el propio edificio es un SPOF, y la solución sería replicar todos los sistemas en otro edificio, a poder ser alejado geográficamente del primero.

4.2.3.2.1 Clúster HA de 2 servidores.

Un clúster HA es un conjunto formado por más de 2 servidores que mantienen una serie de servicios compartidos y que se monitorizan de forma constante entre si, con el objetivo de garantizar el funcionamiento ininterrumpido de ciertas aplicaciones. Los componentes de un clúster HA son:

- Nodos. Se trata de los equipos físicos que forman el sistema: los servidores.
- Red de monitorización. Es la red de datos que los nodos utilizan para comprobar continuamente el estado de todos los nodos.
- Red de servicio. Es la red de datos que los nodos utilizan para ofrecer los servicios a los que están destinados.
- Recursos compartidos. Es el conjunto de entidades virtuales que no residen exclusivamente en ninguno de los nodos, y que forman parte del servicio prestado. Por ejemplo: dirección IP, servicios, almacenamiento,…

4.2.3.2.2 Funcionamiento del clúster HA.

El objetivo del clúster es garantizar la máxima disponibilidad de los servicios. La dinámica del clúster se encarga de reconfigurar el clúster de manera automática, para ello se definen los siguientes términos:

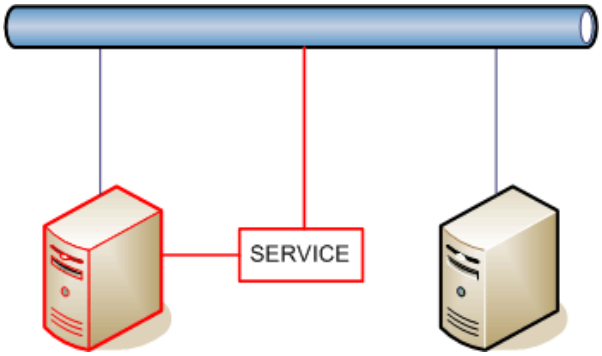

Ilustración 39: Funcionamiento normal del clúster.

Failover. Es un evento que sucede cuando un nodo asume la responsabilidad de otro, importa sus recursos y levanta el servicio correspondiente. Se trata de una situación excepcional, tras el fallo de un nodo.

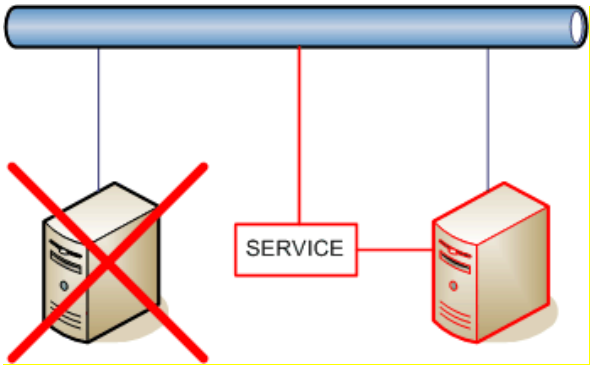

Ilustración 40: Failover.

Failback. Es una política de distribución de recursos y servicios, tras la recuperación de un nodo y su nueva unión al clúster, posterior a un failover. El failback obliga a que los recursos y servicios que estuvieran inicialmente en el nodo antes del failover, vuelvan a ser asumidos por él mismo.

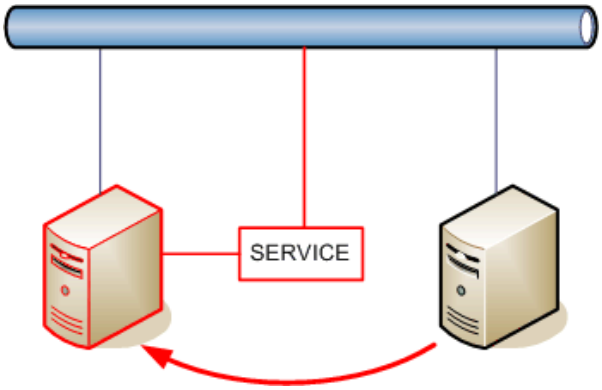

Ilustración 41: Failback.

Takeover. Es un failover automático, que se produce cuando un nodo nota un fallo en el servicio. El nodo que se declara fallido es forzado a ceder el servicio y recursos, o simplemente eliminado.

Splitbrain (división de cerebros). Es un caso especial de failover, en el cual falla el mecanismo de comunicación 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.

4.2.3.2.3 Implementación de heartbeat.

En primer lugar instalamos los paquetes necesarios:

```
# aptitude install heartbeat ntpdate
```

A continuación editamos los ficheros de configuración para adecuar los parámetros a nuestros requerimientos. En el fichero /etc/ha.d/ha.cf:

```
logfacility daemon                      # Logeo en el syslog.
node balanceador1 balanceador2          # Nombre de los nodos del clúster.
keepalive 1                             # Manda "latidos" cada 1 segundo.
deadtime 10                     # Declara nodo muerto tras 10 segundos.
bcast eth0                      # Utiliza interfax eth0 para broadcast.
ping 1.1.1.1                    # Ping para monitorizar la conectividad.
auto_failback no                # No hacer failback automáticamente.
respawn hacluster /usr/lib/heartbeat/ipfail # Hacer failover con fallos de red
```

En el fichero /etc/ha.d/haresources se incluye el nodo preferido, la VIP y ldirectord para que arranque ese servicio.

```
lvs01 1.1.1.4 ldirectord
```

Es conveniente eliminar del arranque automático al demonio de ldirectord:

```
# /usr/sbin/update-rc.d -f ldirectord remove
```

Sí habría que incluir al heartbeat en el arranque automático (si es que no está ya de la instalación).

4.2.4 Nodos Almacén.

4.2.4.1 Network File System (NFS).

El sistema NFS (Network File Sistem) fue desarrollado por Sun Microsystems en 1984 para permitir montar una partición perteneciente a una máquina remota como si fuera una partición local. Nos proporciona, por tanto, un método rápido y eficaz de compartir almacenamiento en el ámbito de una red de área local. Esto nos permite centralizar archivos en una localización, mientras se permite su acceso continuo a los usuarios autorizados.

Los RFCs implicados son:

- RFC 1094 (NFSv2)
- RFC 1813 (NFSv3)
- RFC 2203 (RPCSEC_GSS)
- RFC 2623 (seguridad NFSv2 y NFSv3)
- RFC 3530 (NFSv4)

El soporte NFS lo proporciona el kernel de Linux. Se usan RPCs (Remote Procedure Calls, Llamadas a procedimiento remoto) para enrutar peticiones entre clientes y servidores, implicando que el servicio `portmap` debe estar disponible y activo en los niveles de ejecución adecuados para que la comunicación NFS funcione. Los procesos más importantes que intervienen en NFS son:

- `rpc.mountd` : El proceso que recibe la petición de montaje desde un cliente NFS y chequea para mirar si coincide con un sistema de archivos actualmente exportado.
- `rpc.nfsd` : El proceso que implementa los componentes del espacio del usuario del servicio NFS. Trabaja con el kernel Linux para satisfacer las demandas dinámicas de clientes NFS, tales como proporcionar procesos adicionales del servidor para que los clientes NFS lo utilicen.

Con NFS, la autentificación sólo se produce cuando el cliente intenta montar un sistema de archivos remoto. Para limitar el acceso, el servidor NFS utiliza en primer lugar envolturas TCP (TCP wrappers). Estas envolturas leen los archivos /etc/hosts.allow y /etc/hosts.deny para determinar si a un cliente particular le debe ser explícitamente permitido o denegado su acceso al NFS. Después de que al cliente se le permite acceso a una envoltura TCP, el servidor NFS recurre a su archivo de configuración, /etc/exports, para determinar si el cliente tiene suficientes privilegios para montar alguno de los sistemas de archivos exportados. Después de permitir el acceso, cualquier operación de archivos y directorios es mandada al servidor usando llamadas de procedimiento remotas. Los privilegios de montajes NFS son permitidos específicamente a clientes, no a usuarios. Los sistemas de archivos exportados pueden ser accedidos por cualquier usuario en la máquina remota.

4.2.4.1.1 Implementación de NFS.

Cada Sistema de archivos que se exporta a usuarios remotos vía NFS, así como los derechos de acceso relativos a ellos, están localizados en el archivo `/etc/exports`. Este archivo es leído por el comando `exportfs` para dar a `rpc.mountd` y a `rpc.nfsd` la información necesaria para permitir el montaje remoto de un sistema de archivos por una máquina autorizada.

El comando `exportfs` permite a root exportar directorios concretos sin reiniciar los servicios NFS. Cuando se le pasan las opciones apropiadas a `exportfs`, el sistema de archivos a exportar es incluido en `/var/lib/nfs/xtab`. El demonio `rpc.mountd` vigila el archivo `xtab` continuamente para realizar la exportación.

Hay varias opciones disponibles cuando usamos `exportfs`:

- `-r` : Provoca que todos los directorios listados en `/etc/exports` sean exportados construyendo una nueva lista de exportación en `/etc/lib/nfs/xtab`.
- `-a` : Provoca que todos los directorios sean exportados o no, dependiendo de otras opciones.

- `-o opciones` : Permite al usuario especificar directorios a exportar que no estén listados en `/etc/exports`. Estos sistemas de archivos adicionales compartidos deben ser escritos de la misma forma que son especificados en `/etc/exports`.
- `-i` : Ignora `/etc/exports`; sólo las opciones dadas desde la línea de comandos son usadas para definir los sistemas de archivos exportados.
- `-u` : Termina de exportar directorios que puedan ser montados por usuarios remotos.
- `-v` : Operación descriptiva, se muestra en gran detalle la ejecución del comando.

Si no se pasan opciones al comando `exportfs`, mostrará una lista de los sistemas de archivos actualmente exportados.

El archivo `/etc/exports` controla qué sistemas de archivos son exportados a las máquinas remotas y especifica opciones particulares para ello. Cada sistema de archivos exportado debe tener su propia línea, y la lista de máquinas autorizadas colocada después, debe estar separada por un espacio. Las opciones para cada uno de las máquinas deben ser colocadas entre paréntesis directamente detrás del identificador de la máquina, sin ningún espacio de separación entre la máquina y el primer paréntesis. Las opciones predeterminadas son:

- `ro` : Sólo lectura (read only). Las máquinas que monten este sistema de archivos no podrán cambiarlo. Para permitir los cambios, se debe especificar la opción `rw` (lectura-escritura, read-write).
- `async` : Permite al servidor escribir los datos en el disco cuando lo crea conveniente. Mientras que esto no tiene importancia en un sistema de sólo lectura, si una máquina hace cambios en un sistema de archivos de lectura-escritura y el servidor se cae o se apaga, se pueden perder datos. Especificando la opción `sync`, todas las escrituras en el disco deben hacerse antes de devolver el control al cliente. Esto disminuye el rendimiento pero aumenta la disponibilidad.
- `wdelay` : Provoca que el servidor NFS retrase el escribir a disco si sospecha que otra petición de escritura es inminente. Esto puede mejorar el rendimiento reduciendo las veces que se debe acceder al disco por comandos de escritura separados. Use `no wdelay` para desactivar esta opción, la cual sólo funciona si está usando la opción `sync`.
- `root_squash` — Previene a los usuarios root conectados remotamente de tener privilegios como root asignándole el userid de 'nobody'. Esto reconvierte el poder del usuario root remoto al de usuario local más bajo, previniendo que los usuarios root remotos puedan convertirse en usuarios root en el sistema local. Alternativamente, la opción `no root squash` lo desactiva. Para reconvertir a todos los usuarios, incluyendo a root, use la opción `all squash`.

Para saltarse estas opciones predeterminadas, debe especificar una opción que tome su lugar. Por ejemplo, si no especifica la opción `rw`, entonces se exportará en sólo lectura. Cada opción predeterminada para cada sistema de archivos exportado, debe ser explícitamente ignorada. Adicionalmente, existen otras opciones que están disponibles que no tienen especificado un valor predeterminado. Estas incluyen desactivar el navegar por subdirectorios, permitir el acceso a puertos inseguros, y permitir bloquear archivos inseguros.

La manera en que el archivo `/etc/exports` está organizado es muy importante, particularmente lo que concierne a los espacios en blanco. Recuerde separar siempre los sistemas de archivos exportados de una máquina a la otra, con un espacio. Sin embargo, no debería haber otros espacios en el archivo a menos que se usen en líneas comentadas. Por ejemplo, las siguientes dos líneas significan cosas distintas:

```
/correo bbuzon01.esi.us.es(rw)
/correo bbuzon01.esi.us.es (rw)
```

La primera línea permite sólo a los usuarios de correo1.esi.us.es acceder en modo de lectura-escritura al directorio /buzones. La segunda línea permite a los usuarios de buzon01.esi.us.es montar el directorio de solo lectura (el predeterminado), pero el resto del mundo puede instalarlo como lectura-escritura.

En el lado cliente, cualquier compartición NFS puesta a disposición por un servidor puede ser montada usando varios métodos. La compartición puede ser montada manualmente, usando el comando mount. Sin embargo, esto requiere que el usuario root teclee el comando mount cada vez que el sistema reinicie. Los dos métodos de configurar las comparticiones NFS para que sean montadas automáticamente en el momento de arranque incluyen la modificación de /etc/fstab o el uso del servicio autofs.

Colocando una línea adecuadamente formada en el archivo /etc/fstab tiene el mismo efecto que el montaje manual del sistema de archivos exportado. El archivo /etc/fstab es leído por el script /etc/rc.d/init.d/netfs cuando arranca el sistema y cualquier compartición NFS listada será montada.

Un ejemplo de línea `/etc/fstab` para montar un NFS exportado será parecida a:

```
<server-host>:</path/of/dir> </local/mnt/point> nfs <options> 0 0
```

La opción `<server-host>` tiene que ver con el nombre de la máquina, dirección IP o nombre de dominio totalmente cualificado del servidor que exporta el sistema de archivos.

La opción `</path/of/directory>` es la ruta al directorio exportado.

La opción `</local/mount/point>` especifica dónde montar en el sistema de archivos local el directorio exportado. Este punto de montaje debe existir antes de que `/etc/fstab` sea leído o el montaje fallará.

La opción nfs especifica el tipo de sistema de archivos que esta siendo montado.

El área `<options>` especifica como el sistema de archivos es montado. Por ejemplo, si las opciones indican rw,suid, el sistema de archivos exportado será montado en modo de lectura-escritura y los ID de usuario y grupo puestos por el servidor serán usados. Aquí no se usan paréntesis.

Una desventaja de usar /etc/fstab es que, sin tener en cuenta con que frecuencia se use este sistema de archivos montado, su sistema debe dedicar recursos para mantener este montaje en su sitio. Esto no es un problema con uno o dos montajes, pero cuando su sistema está manteniendo montajes a una docena de sistemas al mismo tiempo, el rendimiento global puede decaer. Una alternativa a /etc/fstab es usar la utilidad basada en el kernel automount, la cual monta y desmonta sistemas de archivos NFS automáticamente, ahorrando recursos.

4.2.4.2 Clúster de alta disponibilidad.

La implementación de NFS en alta disponibilidad presenta cierta dificultad, ya que queremos conseguir a la vez tres objetivos:

1. Almacenamiento compartido, para proveer de datos a todos los servidores buzones.
2. Redundancia en los datos, para evitar que fallos de disco interrumpan el servicio.
3. Alta disponibilidad en el servicio NFS, para evitar que un fallo en un servidor NFS interrumpa el servicio.

Por lo tanto en los servidores de almacén debemos implementar varios componentes que nos proporcionen dicha funcionalidad, los cuales son:

1. NFS, para el almacenamiento compartido.
2. RAID hardware y DRBD para la redundancia de datos.
3. Heartbeat para la alta disponibilidad.

Hemos hablado de NFS en el apartado anterior (4.2.4.1), y de heartbeat en el apartado 4.2.3.1, pero nos queda comentar cómo vamos a conseguir la redundancia con RAID hardware y DRBD.

DRBD (Distributed Replicated Block Device, Dispositivo de replicación distribuida de bloques) permite construir un mirror remoto en tiempo real. Para ello, DRBD crea un dispositivo de bloques drbd0 accesible desde ambos servidores. El servidor primario es el que tiene acceso rw en el dispositivo drbd0: cada vez que escribe algo en drbd0 lo escribe en la partición física y esos mismos datos se envían por TCP/IP al servidor secundario (que sólo tiene acceso ro) consiguiendo que ambas particiones físicas estén sincronizadas.

Si además conseguimos que ambas particiones físicas estén compuestas por un RAID 1 a nivel hardware, la redundancia de datos será doble.

4.2.4.2.1 Implementación de NFS+DRBD+heartbeat.

Lo primero que debemos hacer es instalar los paquetes necesarios.

```
# aptitude install heartbeat ntpdate nfs-kernel-server
# aptitude install kernel-headers-2.6.8-2-386
# aptitude install drbd0.7-module-source drbd0.7-utils
# cd /usr/src/
# tar xvfz drbd0.7.tar.gz
# cd modules/drbd/drbd
# make
# make install
```

Y entonces, cargar el módulo DRDB.

```
# modprobe drbd
```

Queremos implementar esta solución para el directorio /correo de los servidores almacen01 y almacen02. En primer lugar, debemos asegurarnos que tienen disponibles los siguientes dispositivos.

`/dev/sda7`: 150 MB, unmounted, lógica, ext3 (contendrá los meta-datos de DRBD).
`/dev/sda8`: unmounted, lógica, ext3 (contendrá el directorio /correo).

Y también debemos comprobar en ambos servidores que `/dev/sda7` y `/dev/sda8` no aparecen en /etc/fstab.

A continuación hay que modificar el fichero /ect/drdb.conf, incluyendo los datos de los servidores y los discos con los que vamos a trabajar. El fichero completo lo podemos encontrar en el anexo 6.7. A continuación hay que ejecutar:

```
# drbdadm up all
# drbdadm -- --do-what-I-say primary all
# drbdadm -- connect all
```

También modificamos el fichero de exportación NFS (etc/exports) en cada servidor para indicar los clientes buzon01 y buzon02:

```
/correo bbuzon01(rw,no_root_squash,sync) bbuzon02(rw,no_root_squash,sync)
```

NFS almacena información importante (locks, etc.) en /var/lib/nfs. Si almacen01 falla almacen02 le reemplazará pero la información que almacen02 tiene en /var/lib/nfs será diferente de la que tenía almacen01. Para resolver este problema vamos a almacenar la información de /var/lib/nfs en la partición /dev/sda8 que está sincronizada mediante DRBD entre almacen01 y almacen02. Así, si almacen01 falla almacen02 dispondrá de toda su información. En almacen01:

```
# mkdir /data
# mount -t ext3 /dev/drbd0 /data
# mv /var/lib/nfs /data
# ln -s /data/nfs /var/lib/nfs
# mkdir /data/export
# umount /data
```

Y en almacen02:

```
# mkdir /data
# rm -fr /var/lib/nfs
# ln -s /data/nfs /var/lib/nfs
```

Ahora nos resta configurar la alta disponibilidad. En primer lugar creamos los ficheros /etc/heartbeat/ha.cf, /etc/heartbeat/haresources y /etc/heartbeat/authkeys idénticos en ambos servidores.

/etc/heartbeat/ha.cf

```
logfacility     local0
bcast   eth0
```

```
keepalive 2
deadtime 10
node    balmacen01
node    balmacen02
```

/etc/heartbeat/haresources

```
balmacen01  IPaddr::192.168.1.253/24/eth0  drbddisk::r0
Filesystem::/dev/drbd0::/data::ext3  nfs-kernel-server
```

/etc/heartbeat/authkeys

```
auth 3
3 md5 mi_password
```

Ya conocemos del apartado 4.2.3.1 la utilidad de los dos primeros ficheros. En el tercero, definimos el mecanismo de autentificación (md5) y el password para que los dos demonios heartbeat de los servidores se autentifiquen uno contra el otro (mi_password). Sólo root debe tener permisos de lectura sobre /etc/heartbeat/authkeys por lo que haremos:

```
# chmod 600 /etc/heartbeat/authkeys
```

En este momento ya hemos realizado toda la configuración. Nos queda arrancar todos los servicios.

Arrancamos DRBD en ambos servidores:

```
# /etc/init.d/drbd start
```

Arrancamos HeartBeat en ambos servidores:

```
# /etc/init.d/heartbeat start
```

Por último, nos falta automatizar el montaje del dispositivo en los clientes. Para ello editamos /etc/fstab y añadimos una entrada:

```
# File_system          Mount_point Type Options Dump Pass
192.168.1.12:/correo   /correo     nfs  rw      0    0
```

4.3 Banco de pruebas.

En este apartado definimos un conjunto de pruebas para comprobar que el sistema cumple correctamente con los requisitos propuestos. Con este cometido, separamos las pruebas en tres grupos:

- Pruebas funcionales: se comprobará que el sistema es capaz de hacer lo que debe (mandar y recibir correos, etc.).
- Pruebas de disponibilidad: se comprobará que el sistema es robusto frente a fallos hardware/software.
- Pruebas de carga: se someterá el sistema a una simulación de carga de trabajo estándar y límite, para comprobar su estabilidad.

4.3.1 Pruebas funcionales.

Durante estas pruebas debemos comprobar el correcto funcionamiento de cada uno de los componentes del sistema. Definimos un total de 6 conjuntos de pruebas.

4.3.1.1 Envío y recepción de correo electrónico.

Se deben realizar envíos y recepciones de correo electrónico con origen y destino tanto externo como interno. Además, hay que tener en cuenta como destino las direcciones alternativas (alias) de los usuarios.

Obviando el caso de origen y destino externo, tendremos que realizar un total de 5 pruebas distintas.

Destino / Origen	Interno principal	Interno alternativa	Externo
Interno	**1**	**2**	**3**
Externo	**4**	**5**	**X**

Tabla 6: Pruebas de envío y recepción.

4.3.1.2 Conexiones POP e IMAP.

Hay que realizar conexiones POP e IMAP al servidor, realizar la autenticación correctamente con un usuario, y poder ver o descargar el correo electrónico que tenga en su buzón.

Para ello se puede utilizar cualquier cliente de correo con capacidad para establecer conexiones POP e IMAP, como por ejemplo Mozilla Thunderbird.

4.3.1.3 Webmail.

Se debe poder establecer una conexión con el webmail vía navegador web. Una vez realizada la conexión, hay que comprobar que la aplicación funciona correctamente en todas sus opciones, incluyendo pruebas de envío y recepción de correo.

4.3.1.4 Antivirus.

Para esta prueba se utilizará un virus de prueba denominado eicar (http://www.eicar.com). Si enviamos un correo con este fichero adjunto, el sistema debería rechazarlo.

4.3.1.5 Antispam.

En esta prueba utilizaremos una base de datos de spam para elegir varios correos al azar y enviarlos a un usuario del sistema. Los correos deberían ser rechazados.

4.3.1.6 Balanceo de carga.

El objetivo de esta prueba es comprobar que las conexiones se realizan hacia los dos servidores del mismo servicio de forma equitativa. Para ello realizaremos varias pruebas consecutivas de los servicios afectados, y deberemos comprobar que el balanceo funciona con la configuración establecida. Los servicios a prueba son:

- SMTP en buzon01 y buzon02.
- IMAP en buzon01 y buzon02.
- POP en buzon01 y buzon02.
- HTTPS en buzon01 y buzon02.
- LDAP en ldap01 ldap02.

4.3.2 Pruebas de disponibilidad.

El objetivo de estas pruebas es asegurar que el sistema continua funcionando de manera completamente operativa tras la pérdida de servidores (incluyendo sus componentes internos), o de electrónica de red. Simularemos por lo tanto dos tipos de fallos:

- Fallo hardware: suponemos que falla cualquier componente de un servidor o el servidor completo, por lo que lo simulamos apagando el propio servidor.
- Fallo de red: suponemos que perdemos una conexión de red, causada por un fallo en el switch o por un fallo en el cableado. Lo simularemos desconectando uno de los cables.

4.3.2.1 Fallo hardware.

Tenemos que tener en cuenta toda la casuística, que implica:

- Por cada módulo, 3 casos posibles:
  - 0: los dos nodos arrancados.
  - 1: apagado el nodo 01.
  - 2: apagado el nodo 02.

- 4 módulos diferentes:
  - buzon
  - ldap
  - lvs
  - almacen

Esto supone un total de 3^4=81 casos posibles. Obviando el caso en el que todos los servidores están arrancados, hay un total de 80 pruebas, que aunque serían necesarias de realizar para comprobar exhaustivamente el correcto funcionamiento de la alta disponibilidad, es una tarea algo complicada de realizar. En el anexo 6.8 se encuentra la tabla de pruebas completas.

Para simplificar, vamos a suponer que los nodos dentro del mismo módulo son idénticos a efectos de esta prueba, lo cual implica:

- Por cada módulo, 2 casos posibles:
  - 0: los dos nodos arrancados.
  - 1: uno de los nodos apagados.

- 4 módulos diferentes:
  - buzon
  - ldap
  - lvs
  - almacen

Lo cual nos reduce el número de pruebas a 2^4=16. Obviando el caso en el que todos los servidores están encendidos, nos quedamos con 15 pruebas, que esquematizamos en la siguiente tabla:

Prueba	buzon	ldap	lvs	almacen
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1
10	0	1	1	0

11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1

Tabla 7: Pruebas de fallo hardware.

4.3.2.2 Fallo de red.

En este caso tenemos que simular el fallo de una de las dos interfaces que componen cada uno de los teaming/bonding existentes. Esto significa:

- Hay dos casos posibles:
  - 1: desconectar el cable número 1
  - 2: desconectar el cable número 2

- Existen un total de 10 bondings:
  - buzon01_servidores
  - buzon01_backend
  - buzon02_servidores
  - buzon02_backend
  - ldap01
  - ldap02
  - lvs01
  - lvs02
  - almacen01
  - almacen02

En este caso lo más sencillo es definir un procedimiento y repetirlo para cada uno de los bonding, con lo cual tenemos un total de 10 pruebas. El procedimiento sería el siguiente:

1. Comprobar la conectividad de red.
2. Desconectar el cable número 1.
3. Conectar el cable número 1.
4. Desconectar el cable número 2.
5. Conectar el cable número 2.

La prueba se da por superada si la conectividad de red no se pierde en ningún momento de la prueba. Es conveniente dejar pasar unos 10 segundos entre un paso y el siguiente.

4.3.3 Pruebas de carga.

Una vez comprobado que el sistema funciona de manera correcta, debemos asegurar que sigue haciéndolo con la carga de trabajo que debe soportar de forma continua.

Para ello utilizaremos herramientas de envío masivo de correos electrónicos, y lanzar determinados envíos con el fin de medir tiempos y comprobar la estabilidad del sistema. Se van a definir 2 pruebas distintas.

1. Envío de 500 correos electrónicos a 500 usuarios diferentes en el mismo momento. Debería ser capaz de despacharlos en aproximadamente 5 minutos, sin comprometer la estabilidad del sistema.

2. Envío de 1000 correos electrónicos al mismo usuario en el mismo momento. Debería ser capaz de despacharlos en menos de 10 minutos, sin comprometer la estabilidad del sistema.

Sería conveniente repetir estas pruebas 5 veces cada una, dejando un tiempo entre pruebas equivalente al tiempo empleado en la prueba anterior.

4.4 Planificación temporal.

Se va a esbozar una planificación temporal, ilustrándose con el correspondiente diagrama de Gantt. Se van a definir las siguientes tareas:

	Nombre de tarea	Duración
1	Inicio de Proyecto	0 días
2	Adquisición del hardware	15 días
3	Reutilización de hardware	7 días
4	Integración de sistemas	21 días
5	Pruebas	5 días
6	Fin de proyecto	0 días
7	Soporte y mantenimiento	260 días

Ilustración 42: Tareas de proyecto.

1. Inicio de proyecto.

2. Adquisición del hardware. Se trata del pedido del nuevo hardware necesario, el cual se espera que sea servido en 15 días laborables.

3. Reutilización del hardware. Se trata de la adecuación del hardware existente para reutilizarlo en el nuevo sistema. Se estiman 7 días laborables, ejecutándose en paralelo a la adquisición.

4. Integración de sistemas. Es la implantación de los sistemas operativos y todas las herramientas conforme a las instrucciones dadas en este proyecto. Se estiman 21 días laborables.

5. Pruebas. Batería de pruebas especificadas en el proyecto. Se estiman 5 días laborables.

6. Fin de proyecto. Hito de aceptación de proyecto.

7. Soporte y mantenimiento. Durante 1 año natural a partir del fin de proyecto, incluye el soporte del software y hardware, así como una administración y mantenimiento que se especificará en la memoria económica.

A continuación vemos el diagrama de Gantt correspondiente.

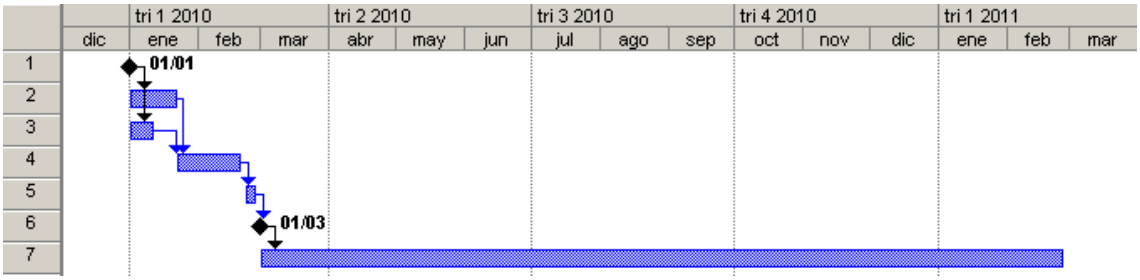

Ilustración 43: Diagrama de Gantt.

4.5 Memoria económica.

En este apartado vamos a realizar una estimación de costes incurridos en la implantación del sistema de correo. Para ello tendremos en cuenta tres ámbitos principales:

- Adquisición del hardware.
- Integración del sistema.
- Soporte y mantenimiento.

A continuación se desglosa una estimación de costes (sin IVA) teniendo en cuenta los tres conceptos anteriores.
En cuanto a la adquisición del hardware, se han aprovechado los 2 servidores existentes, con lo cual para cubrir la necesidad total de 8 servidores, es necesario el suministro de 6 nuevos equipos. Se ha optado por el modelo siguiente, con un coste de 2.500 euros.

HP Proliant DL360 G6 (504634-421)

Ilustración 44: HP Proliant Dl360 G6.

Tipo de procesador	Procesador Intel® Xeon® E5540 Quad Core a 2,53 GHz
Velocidad del procesador	2,53 GHz
Número de procesadores	1 procesador
Actualización del procesador	Se admiten hasta 2 procesadores
Núcleo de procesador disponible	Quad
Memoria caché interna	8 MB (1 x 8 MB) de caché de nivel 3
Chipset	Chipset Intel® 5520
Memoria	DDR3 registrada (RDIMM)
Memoria de serie	6 GB (3 x 2 GB) de memoria estándar
Bus frontal del procesador	Bus frontal a 1.066 MHz
Memoria máxima	192 GB (12 x 16 GB)
Ranuras de memoria	18 ranuras DIMM

Ilustración 45: Especificaciones técnicas HP Proliant DL360 G6.

Para el resto de costes se han determinado tres perfiles de trabajo:

- Jefe de proyecto. Su tarea es la planificación y la gestión durante todo el ciclo de vida del proyecto. Coordinará los trabajos y asistirá a las reuniones de seguimiento del proyecto. Su coste estimado es de 50 euros/hora.
- Ingeniero de sistemas. Se trata de una persona con un perfil eminentemente técnico y con experiencia en el despliegues de sistemas GNU/Linux. Se encargará del diseño del sistema y su configuración, así como del período de pruebas. Se tendrá en cuenta un coste de 35 euros/hora.
- Técnico de sistemas. Su tarea es básicamente el despliegue del hardware y la instalación de los sistemas operativos y paquetes software. Su coste estimado es de 20 euros/hora.

Estimando el número de horas especificado en la tabla 8, el coste estimado de la integración del sistema es de 7.000 euros, y el coste del mantenimiento de 9.000 euros.

Adquisición del hardware	Cantidad	Euros/unidad	Subtotal
Servidores disponibles	2	0	0
Servidores necesarios	6	2500	15000
TOTAL hardware			**15000**

Integración del sistema.	Horas	Euros/hora	Subtotal
Jefe de proyecto	50	50	2500
Ingeniero de sistemas	100	35	3500
Técnico de sistemas	50	20	1000
TOTAL integración			**7000**

Soporte y mantenimiento	Horas	Euros/hora	Subtotal
Jefe de proyecto	20	50	1000
Ingeniero de sistemas	200	35	7000
Técnico de sistemas	50	20	1000
TOTAL mantenimiento			**9000**

TOTAL	**31000**

Tabla 8: Estimación económica.

La estimación del coste total, por lo tanto, asciende a 31.000 euros.