Utilización y Administración avanzadas de sistemas GNU/Linux y aplicaciones Software Libre para estudiantes universitarios

Clustering y Alta Disponibilidad en GNU/Linux

Josá Angel de Bustos Pérez

Utilización y Administración avanzadas de sistemas GNU/Linux y aplicaciones Software Libre para estudiantes universitariosClustering y Alta Disponibilidad en GNU/Linux

por Josá Angel de Bustos Pérez

Tabla de contenidos

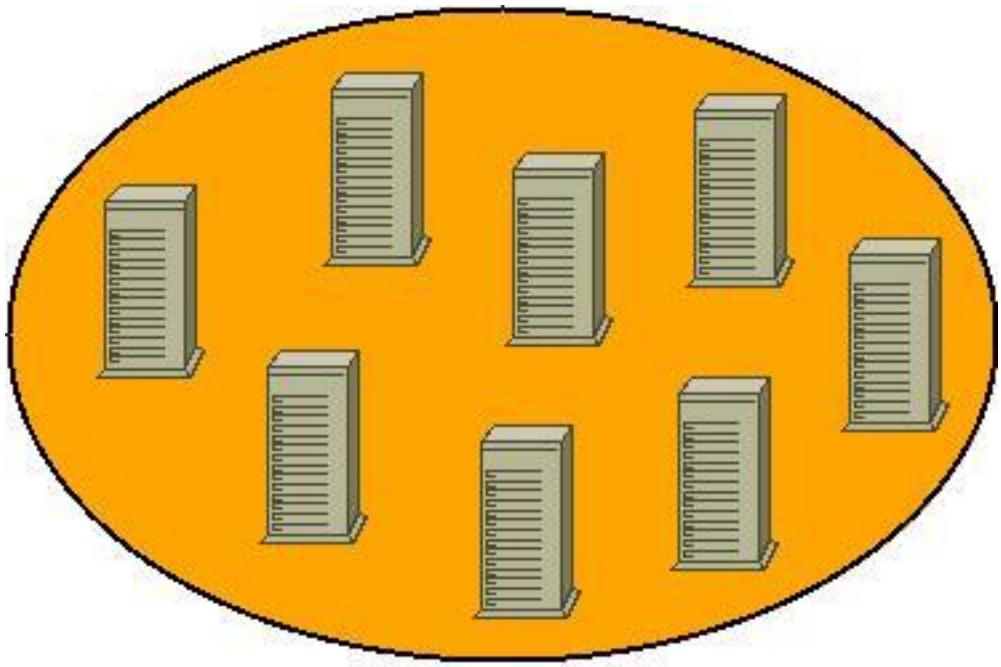
Lista de tablas

Capítulo 1. Clustering

1.1. Que es el clustering?

Clustering es la capacidad de varias máquinas para realizar una tarea como sí de una única máquina se tratase.

Cada una de las máquinas que forman el cluster recibe el nombre de *nodo*.



Cluster como un todo.

1.2. Tipos de cluster

Dependiendo del tipo de solución que busquemos podemos clasificar los clusters en varios tipos:

- *High Performance Cluster*
- *Activo / Pasivo*
- *Activo / Activo*
- *Grid Computing*

1.3. High Performance Cluster

Este tipo de clusters son clusters para cálculo matemático intensivo.

Un ejemplo es el Marenostrum que IBM tiene en España.

1.3.1. Ventajas

- Escalabilidad.

1.3.2. Inconvenientes

- Unica aplicación para cálculo.

1.4. Cluster Activo / Pasivo

En este tipo de cluster solamente hay un nodo que da servicio, el resto estan inactivos.

En el momento que el nodo activo no pueda dar servicio, otro nodo se hará cargo del servicio.

1.4.1. Ventajas

- Cualquier servicio es clusterizable.
- Configuración "sencilla".

1.4.2. Inconvenientes

- Coste (TPC/Euro).
- No aumenta la potencia con el aumento de nodos.

1.5. Cluster Activo / Activo

En este tipo de cluster todos los nodos estan dando servicio.

1.5.1. Ventajas

- Aumento de potencia con el aumento de nodos.
- Coste (TPC/Euro).

1.5.2. Inconvenientes

- No todos los servicios son clusterizables.
- Configuración complicada.

1.6. Grid Computing

Grid Computing se puede pensar cómo una especialización de un cluster activo/activo.

Este tipo de computación distribuida se basa en la utilización de los recursos de diferentes ordenadores, no necesariamente en la misma red. Pudiendo estar geográficamente dispersos.

Este tipo de sistemas están pensados para utilizar los recursos de cientos de ordenadores que no están siendo utilizados.

Por ejemplo, es común que en una empresa o una institución haya multitud de ordenadores encendidos que no están siendo utilizados. A la hora de comer, de los cafés, durante reuniones, ..., los cafés, ...

Se puede utilizar esa potencia para la incrementar la potencia de cálculo de un conjunto de máquinas dedicadas al completo a la realización de una determinada tarea.

Hay diferentes formas de uso:

- Uso de recursos cuando el equipo no está siendo utilizado.
- Uso mínimo de recursos y siempre en función del uso del equipo por parte del usuario.

1.6.1. Ventajas

- Reaprovechamiento de recursos existentes e infrautilizados.

1.6.2. Inconvenientes

- No todos los servicios admiten clustering de este tipo.

Capítulo 2. Arquitectura de Clustering

El propósito de un cluster es:

- Redundancia frente a fallos (alta disponibilidad).
- Aumento de potencia (escalabilidad).

Estos propósitos no son excluyentes.

> **Importante:** A la hora de escoger que tipo de cluster se va a utilizar hay que tener en cuenta las características que nos ofrece cada tipo y cual es el que mas encaja en nuestro entorno.

2.1. Alta disponibilidad

Alta disponibilidad es la capacidad para ofrecer un servicio cuando el servidor que lo ofrece deja de funcionar.

La alta disponibilidad es ofrecida por todos los tipos de cluster.

Garantizar la alta disponibilidad requiere:

- Una correcta configuración del cluster.
- Redundancia de hardware (red, almacenamiento, fuentes de alimentación, ...).
- Redundancia del sistema eléctrico.
- SAIs.
- Nodos en diferentes CPDs.

2.1.1. Los famosos 9s

La disponibilidad de los sistemas se clasifica con la regla de los 9s:

Tabla 2-1. Disponibilidad

Disponibilidad del servicio	Tiempo de caida
99 %	3,7 días
99,9 %	8,8 horas
99,99 %	52,7 minutos
99,999 %	5,3 minutos

Disponibilidad del servicio	Tiempo de caida
99,9999 %	31,5 segundos

2.1.2. Bonding

Bonding es una ténica utilizada en GNU/Linux para utilizar dos tarjetas de red como si fueran una.

Importante: Es necesario que el núcleo esté configurado para ello.

Importante: Los dispositivos de networking tienen que soportarlo.

2.1.3. Almacenamiento

Los datos son críticos ya que sin ellos no se puede ofrecer el servicio. Existen dos técnicas para garantizar la alta disponibilidad en los sistemas de almacenamiento:

- Sistemas de almacenamiento externo.

  Ventajas:

  - Altamente escalable.
  - Rapidez, hasta 4 Gb/s.

  Inconvenientes:

  - Precio.
  - Requiere personal especializado.

- Replicación de datos:

  Ventajas:

  - Barato.

  Inconvenientes:

  - Poco escalable.
  - "Lentitud".

La replicación de datos se puede hacer utilizando slony (http://slony.info/).

> **Importante:** *MySQL* utiliza *NDB* para la replicación de datos.

2.2. Escalabilidad

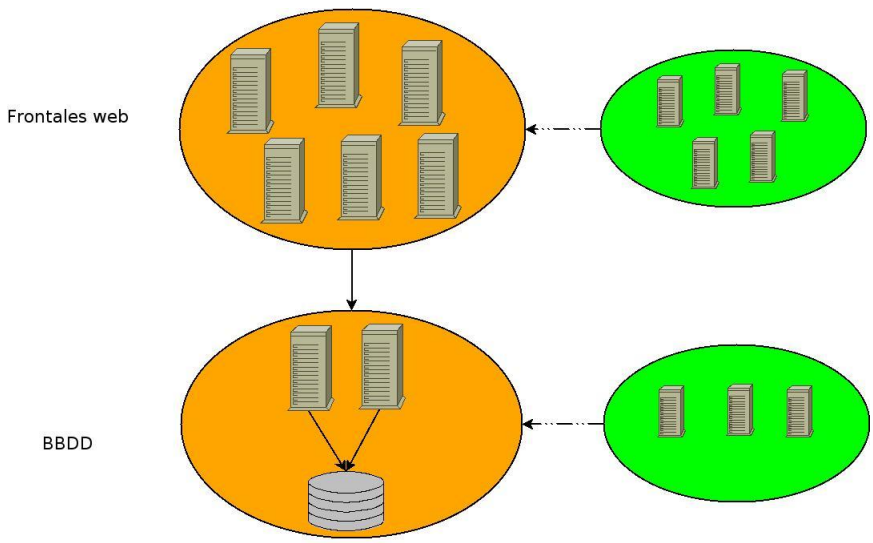
Escalabilidad es la capacidad de crecimiento que tiene un servicio frente a la demanda.

La escalabilidad sólo es ofrecida por:

- High Performance Cluster.
- Cluster Activo / Activo.

La escalabilidad se logra incrementando el número de nodos disponibles en el cluster.

> **Importante:** En un cluster activo / pasivo el incrementar el número de nodos disponibles en el cluster no incrementa la potencia del cluster ya que unicamente un nodo estará ofreciendo el servicio.



Escalabilidad de un cluster.

2.3. Funcionamiento de un cluster

Un servicio de cluster consta de:

- Balanceador de carga.
- Sistema para la detección de fallos en los nodos del cluster.
- Servicio a clusterizar.
- Recursos del cluster.
- Fencing.

Arquitectura básica de un cluster.

2.3.1. Balanceador de carga

Sólo son necesarios en aquellas configuraciones que sean Activo / Activo y/o balanceo de carga.

La función de los balanceadores, también conocidos como *network dispatcher* es redireccionar las peticiones a los servidores que las están atendiendo.

2.3.2. Sistema para la detección de fallos en los nodos del cluster

Es necesario un sistema que detecte cuando existen nodos en el cluster que no están disponibles para ofrecer el servicio. En este caso no se enviarán peticiones para ser atendidas si el cluster es Activo / Activo o no se balanceará el servicio sobre ellos si es Activo / Pasivo.

Para detectar esta situación se utilizan dos técnicas:

1. Heartbeat.
2. Disco de quorum.

2.3.3. Servicio a clusterizar

Es el servicio que se quiere clusterizar.

2.3.4. Recursos del cluster

Son todos aquellos recursos necesarios para el servicio:

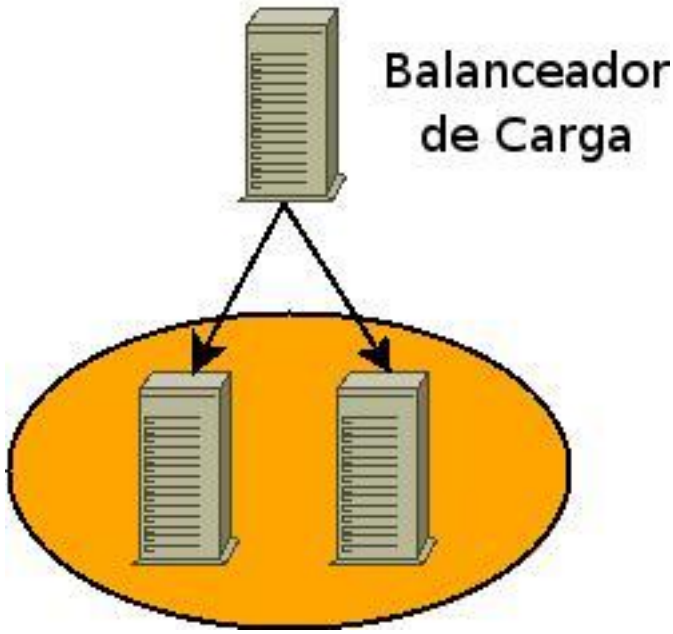
- IP de servicio.
- Filesystems.
- Scripts para arrancar el servicio, ...

2.3.5. Fencing

Cuando un nodo falla y no es capaz de ofrecer el servicio es necesario disponer un medio para reiniciarlo y que libere los recursos que pudiera tener ocupados.

Capítulo 3. Balanceo de Carga

Se puede definir el balanceo de carga como la habilidad que tiene un servicio para repartir el trabajo a realizar entre varias máquinas.



Arquitectura básica de un cluster con balanceo de carga.

Los balancearores pueden ser tanto hardware como software.

3.1. Balanceadores hardware

Los balancedares hardware son máquinas con un propósito específico y solo son útiles para el balanceo de carga.

3.1.1. Ventajas

- Potencia.
- Estabilidad.
- Escalabilidad.

3.1.2. Inconvenientes

- Precio (equipo, mantenimiento, técnicos).
- Sólo sirve para balanceo.

3.2. Balanceadores software

Los balanceadores software son servidores configurados para hacer balanceo.

3.2.1. Ventajas

- Precio.
- Una vez no sea necesario el balanceador o se requiera uno más potente se puede reciclar para otras tareas.

3.2.2. Inconvenientes

- Mayor tiempo de mantenimiento.
- Menor potencia.

3.3. Balanceo en DNS

A partir de la version 9 de BIND nos bastaria simplemte con añadir varias entradas "IN A", una para cada servidor, que apunten al mismo nombre, el del servidor que queremos balancear:

```
www IN A 192.168.0.100
www IN A 192.168.0.101
www IN A 192.168.0.102
```

Utilizando el comando "host" varias veces podemos ver como se balancean las peticiones dns:

```
[jadebustos@dedalo ~]$ host www.servidorbalanceado.es
www.servidorbalanceado.es has address 192.168.0.100
www.servidorbalanceado.es has address 192.168.0.101
www.servidorbalanceado.es has address 192.168.0.102
[jadebustos@dedalo ~]$ host www.servidorbalanceado.es
www.servidorbalanceado.es has address 192.168.0.101
www.servidorbalanceado.es has address 192.168.0.102
www.servidorbalanceado.es has address 192.168.0.100
```

```
[jadebustos@dedalo ~]$ host www.servidorbalanceado.es
www.servidorbalanceado.es has address 192.168.0.100
www.servidorbalanceado.es has address 192.168.0.101
www.servidorbalanceado.es has address 192.168.0.102
[jadebustos@dedalo ~]$
```

De esta forma cada vez que se haga una solicitud DNS para www.servidorbalanceado.es se resolverá con una IP diferente, irá rotando.

Este sistema tiene varios inconvenientes:

1. El primero es que si un servidor cae el DNS no será consciente de ello y el cliente que obtenga la IP del servidor caido como respuesta a su solicitud encontrará que el servicio no esta disponible.
2. El segundo inconveniente es que con las caches de DNS se cacheará permanente el nombre *www.servidorbalanceado.es* con una IP especifica, con lo cual todas las peticiones iran a esa IP y no habrá balanceo, pudiendo de esta forma estar uno o varios servidores sobrecargados y el resto muy ligeros.

3.4. Linux Virtual Server - LVS

Linux Virtual Server (http://www.linuxvirtualserver.org) es una solución para poder implementar un servidor virtual altamente escalable y en alta disponibilidad.

Esta solución consiste en un balanceador de carga, también conocido como director, que será la máquina que será accesible directamente para los clientes y luego tendremos los servidores que serán aquellos que recibiran las peticiones de los clientes, vía el balanceador de carga, y responderán a las peticiones.

> **Importante:** Los servidores podrán estar o bien en la misma red física o en redes diferentes lo que permitirá el tener servidores en granjas distribuidas geográficamente.
>
> Esta solución nos permitirá tener el servicio funcionando casi continuamente ya que no se verá afectado por posibles caídas de las máquinas debido a fallos en el suministro eléctrico o bien cortes en el ISP de una determinada granja. Cualquiera de estos fallos, u otros que pudieran ocurrir, afectarán a una o varias granjas, pero nunca a todas con lo cual el servicio seguiráa funcionando aunque los clientes podráan experimentar cierta demora en el servicio.

Para los clientes existirá un único servidor (el balanceador) que se encargará de distribuir la carga entre los servidores reales.

La escalabilidad en el servicio la conseguiremos añadiendo nodos, mientras que la disponibilidad se logrará identificando el nodo o el balanceador que no funciona y reconfigurando el sistema de tal forma que el servicio no se vea interrumpido. Es decir no enviando peticiones a un nodo que no pudiera dar servicio en ese momento.

El balanceo lo podemos hacer de tres formas:

- Mediante *NAT*
- *IP Tunneling*
- *Direct Routing*

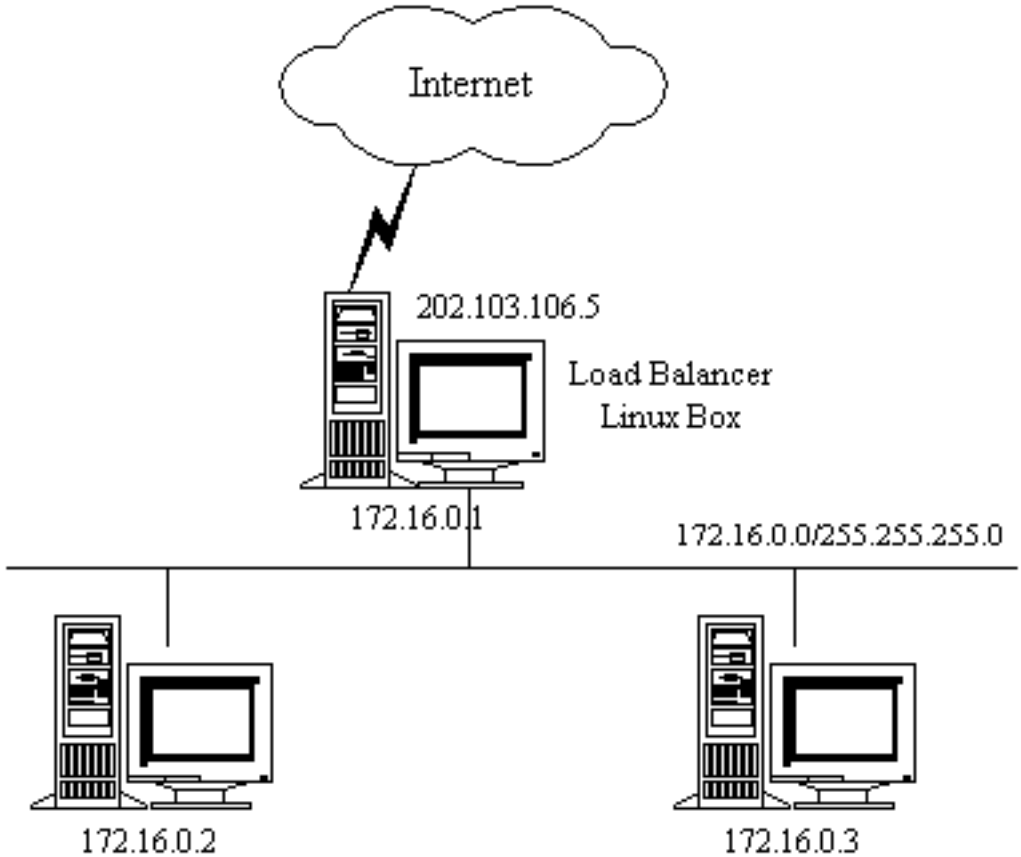
3.4.1. Virtual Server mediante *NAT*

NAT (*N*etwork *A*ddress *T*ranslation) es una técnica utilizada para que una máquina reciba información dirigida a otra y esta pueda reenviarla a quien la solicitó inicialmente.

Para ello la máquina que recibe la información, en forma de paquetes, deberá reescribir los paquetes sustituyendo su propia dirección con la de la máquina que realizó la petición (nos referimos a direcciones tanto físicas, MAC, como lógicas, IP) una vez reescrito el paquete de la forma correcta el balanceador se encargará de enviar los paquetes por la interface adecuada para que le lleguen al destino verdadero del paquete.

Cuando el balanceador reciba peticiones este sobreescribirá el paquete y pondrá la dirección de un servidor real, gracias a esto los servidores reales podrán estar ejecutando cualquier sistema operativo que sea accesible vía TCP/IP.

Cuando el servidor responda lo hará al balanceador y este reescribirá el paquete, otra vez, poniendo en los paquetes la dirección del cliente que solicitó la información.



LVS - NAT.

El balanceador guardará los datos de todas las conexiones que balancee para luego devolver la respuesta al cliente adecuado.

Pero no todo son ventajas, esta sobreescritura de los paquetes trae consigo una carga de CPU que puede llegar a ser un cuello de botella. Además tendremos que tener en cuenta cual es el ancho real de nuestra interface de red y tener presente que por el balanceador van a pasar tanto las peticiones hacia los servidores, como las respuestas de los servidores hacia los clientes.

Todos estos paquetes tendrán que ser reescritos por el balanceador y aunque aumentemos la memoria o las capacidades de procesamiento del balanceador todavía estaremos limitados por el ancho real de la interface ya que las respuestas de los servidores ocuparán mas ancho de banda que las peticiones.

> **Sugerencia:** El problema del ancho de banda se podrá paliar utilizando las capacidades de Bonding del núcleo de Linux.

No suele ser muy recomendable esta opción ya que los costes necesarios para desplegar la infraestructura suelen ser mayores que los de implementar LVS con IP Tunneling o Direct Routing.

El balanceador deberá tener dos IP una de cara a los posibles clientes (DIP) y otra en la red de los servidores (VIP), es decir que el balanceador deberá hacer funciones de enrutado con lo cual el núcleo deberá estar configurado para ello y tendremos que tener el enrutado habilitado y el núcleo tendrá que estar compilado para poder sobreescribir paquetes.

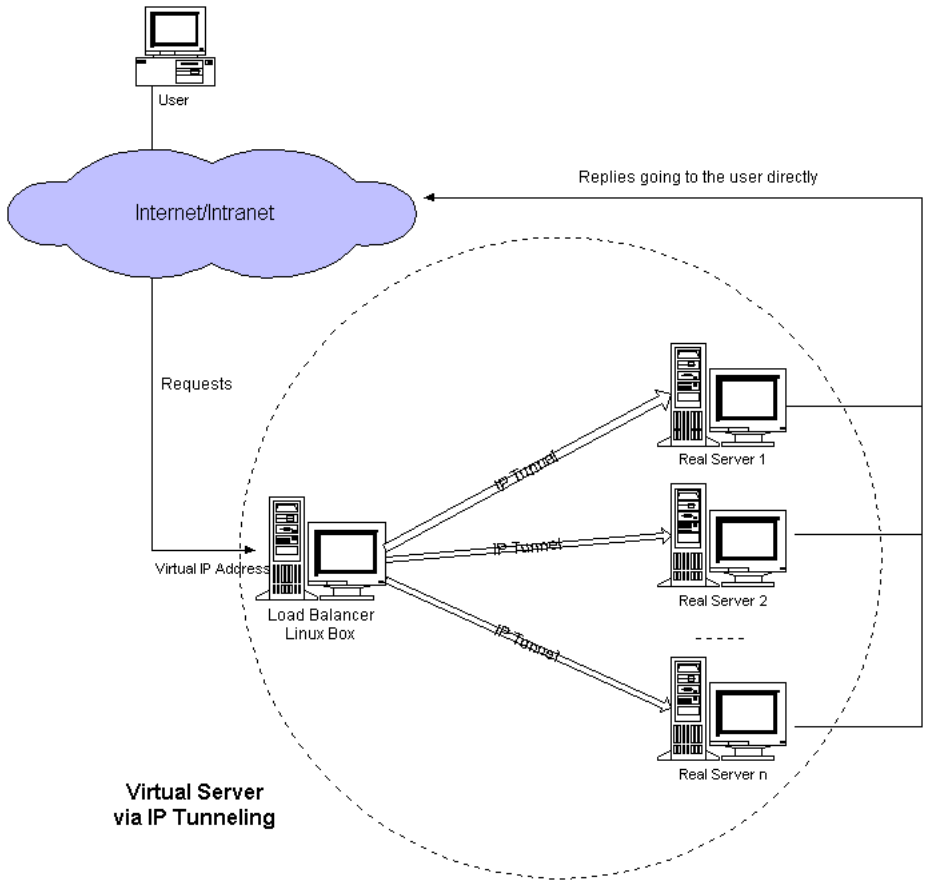
> **Importante:** Para tener habilitado el enrutado es necesario que el núcleo esté configurado para ello y necesitaremos que el fichero `/proc/sys/net/ipv4/ip_forward` si utilizamos *ipv4* o `/proc/sys/net/ipv6/conf/all/forwarding` estén a *1*.

> **Importante:** Los servidores en este caso estarán en la misma red de *VIP* y tendrán como gateway al balanceador de carga.

3.4.2. Virtual Server mediante *IP Tunneling*

Utilizando *NAT* teníamos un cuello de botella en el balanceador ya que tiene que reescribir y distribuir los paquetes del cliente al servidor y viceversa.

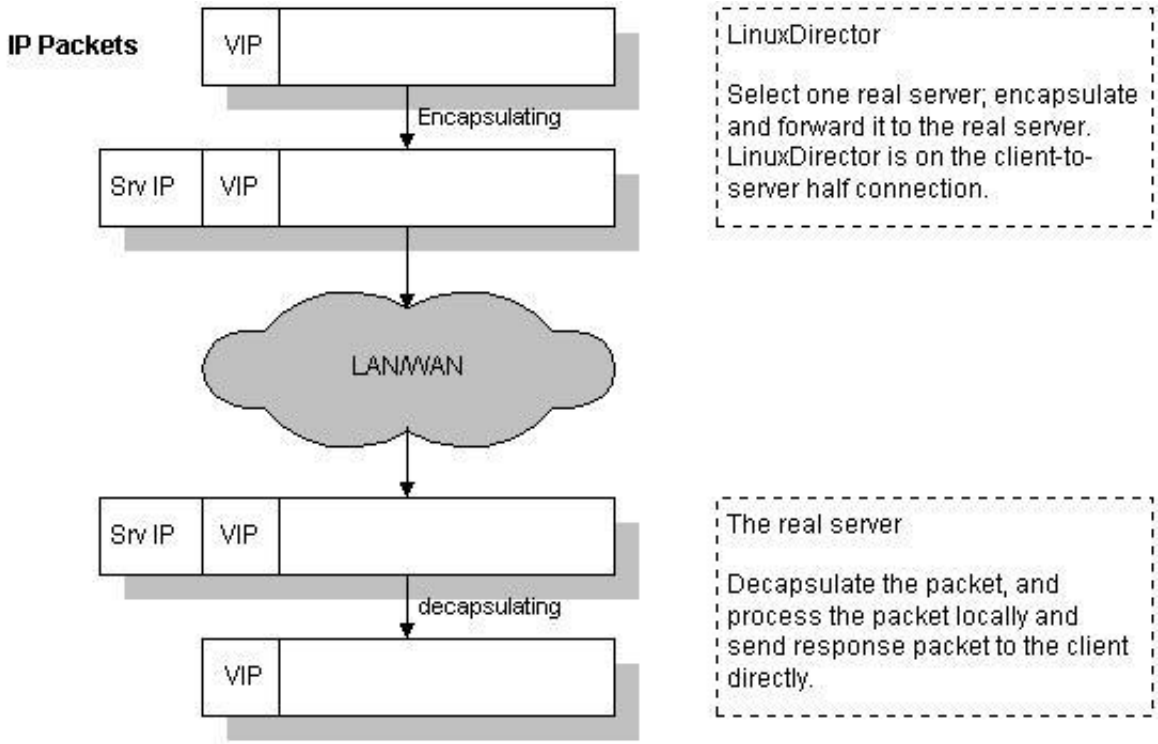
Utilizando *IP Tunneling* el balanceador únicamente tendrá que hacerse cargo de las peticiones de los clientes y enviarlas a los servidores, siendo estos mismos los que responderán a los clientes. De esta forma el balanceador de carga puede manejar mas nodos, es decir el servicio es mas escalable.



LVS - IP Tunneling.

Una vez el balanceador de carga tiene el paquete determina si pertenece a uno de los servicios que tiene balanceados. De ser así encapsula el paquete en otro paquete y se lo envía al servidor de destino. Es este el que se encarga de responder al cliente directametne sin pasar por el balanceador.

El balanceador guarda una tabla de conexiones y cuando le llega un paquete determina si ya existe una conexión abierta y de ser así que servidor real es el que está sirviendola para enviarle el paquete.



Encapsulamiento IP en IP-Tunneling.

Los servidores deberán estar configurados para trabajar con *IP Tunneling (encapsulation)* ya que cuando el balanceador recibe un paquete para uno de los servidores este lo encapsula en un datagrama IP y lo manda a uno de los servidores. Cuando el servidor lo recibe tendá que desencapsularlo y responderá directamente al cliente sin pasar por el balanceador con lo cual los servidores tendrán que estar conectados tanto al balanceador como a los clientes (en NAT sólo con el balanceador).

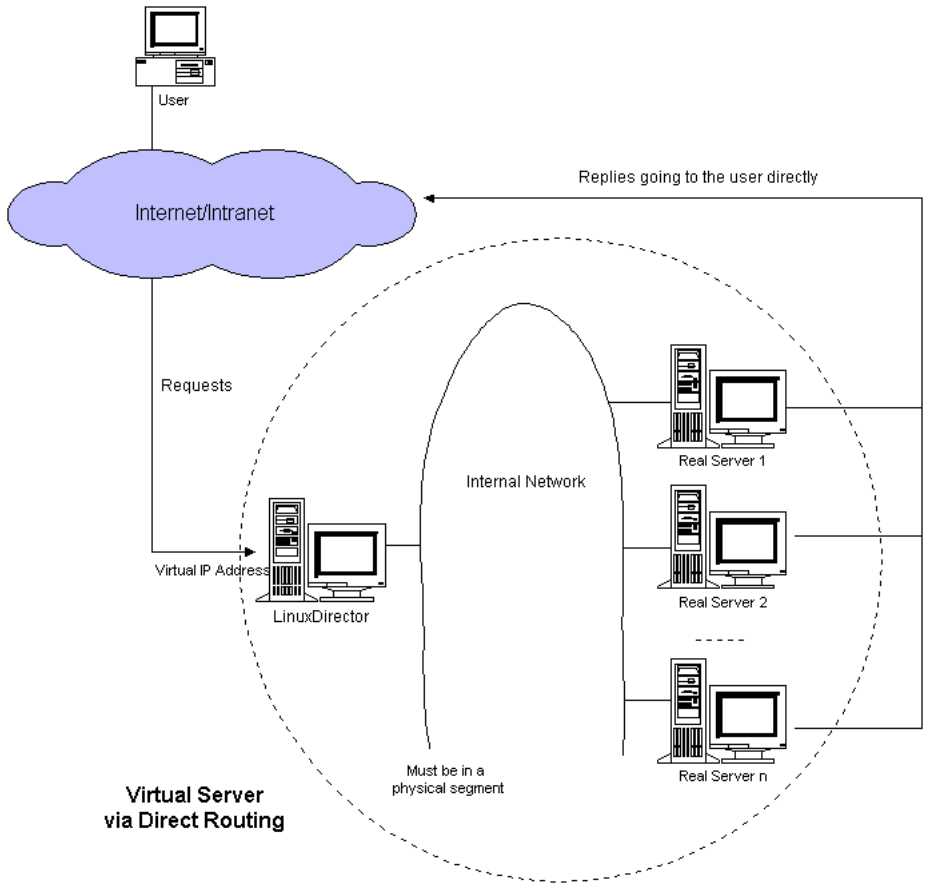
> **Sugerencia:** No es necesario que los servidores estén en la misma red, pueden estar geográficamente distribuidos.

> **Importante:** En esta configuración surge el problema de ARP.

3.4.3. Virtual Server mediante *Direct Routing*

Al igual que en *IP Tunneling* el balanceador sólo gestionará las peticiones del cliente hacía el servidor con lo cual es una solución altamente escalable.

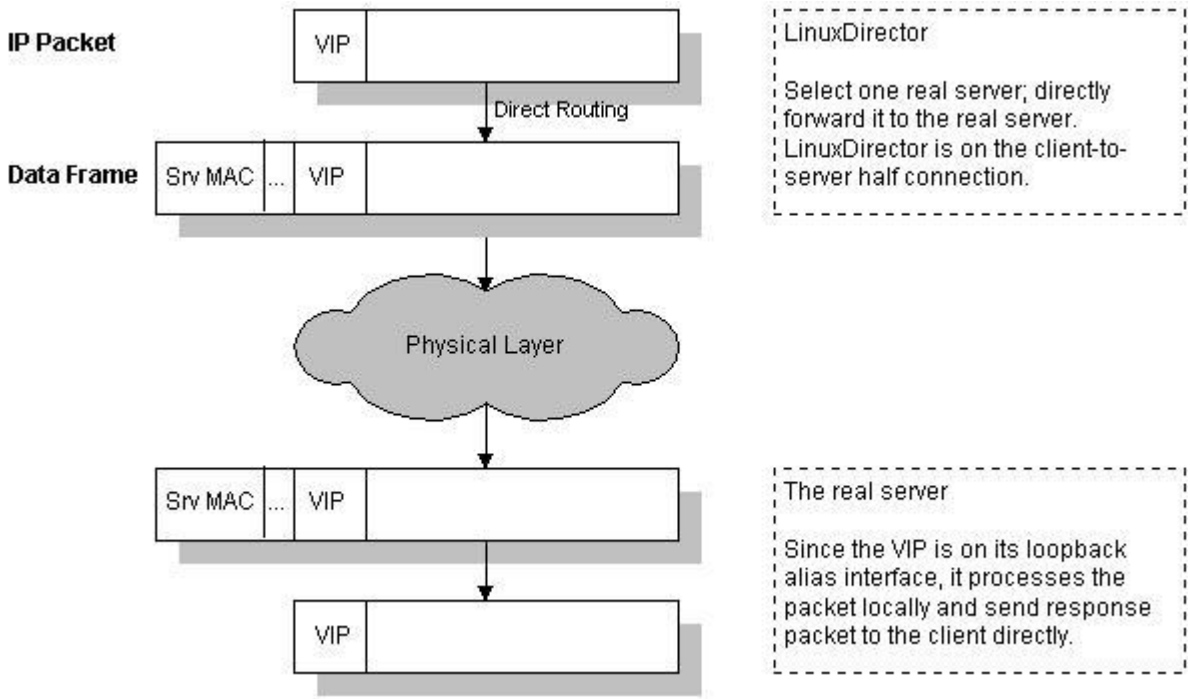
La dirección virtual (VIP) es compartida por el balanceador y los servidores. De esta manera el balanceador recibe las peticiones y las envía a los servidores que procesan las peticiones y dan servicio directamente a los clientes.



LVS - Direct Routing.

En esta solución es necesario que una de las interfaces del balanceador y los servidores están en el mismo segmento físico de red ya que el balanceador de carga cambiará su dirección física, MAC, en la trama por la dirección física de uno de los servidores que tendrá un alias con la dirección VIP.

El balanceador guarda una tabla de conexiones y cuando le llega un paquete determina si ya existe una conexión abierta y de ser así que servidor real es el que está sirviendola para enviarle el paquete.



LVS - Direct Routing.

Importante: El alias se acostumbra a poner en el dispositivo de loopback.

Importante: En esta configuración surge el problema de ARP.

3.4.4. El problema del *ARP*

Cuando utilizamos *Tunneling* o *Direct Routing* tanto el balanceador como los servidores comparten una dirección IP (VIP) y esto puede traer consigo problemas si los balanceadores y los servidores están en la misma red.

El modelo *OSI* de la *ISO* consta de 7 capas, las tres primeras son la capa física, la capa de enlace de datos y la capa de red.

Cuando un ordenador transmite datos empieza a encapsular en la capa de aplicación, séptima capa. Cuando llega a la capa de red añade la información de red, direcciones IP y direcciones lógicas (origen y destino). Acto seguido añade su dirección física o MAC que es una dirección única que tiene cada tarjeta

de red o NIC y que consta de dos partes una que identifica al fabricante de la tarjeta y la otra parte identifica a la tarjeta.

Después en la capa física se traduce toda la información a señales eléctricas y se transmite por el medio. Además de sus direcciones propias IP y MAC se añaden las direcciones del destinatario y si el paquete fuera destinado a una red diferente de la de partida se pondría en el campo de la MAC de destino la MAC de gateway o del router por defecto y este se iría encargando de enrutar el paquete hasta que llegará al último router o gateway el cual cambiaría su MAC por la MAC del equipo que tuviera la IP de destino del paquete. Este último router mandaría el paquete por la interface correspondiente y todos los equipos en esa red desencapsularín el paquete hasta la segunda capa y sólo aquel cuya MAC este en esa trama, como destino, tomará el paquete y lo desencapsulara entero para hacer uso de el.

Cuando utilizamos *Tunneling* o *Direct Routing* tenemos que tener en cuenta que los clientes hacen las peticiones al balanceador, pero sin embargo las respuestas las reciben de los servidores.

> **Importante:** Tanto el balanceador de carga como los servidores comparten una IP (VIP), cuando un cliente solicita una conexión con VIP la petición se debe de hacer al balanceador, no a los clientes.

Cuando llega una petición de un cliente para el servicio bajo *LVS* esta llegará desde fuera de la red, con lo cual el router de esa red hará una petición ARP para obtener la MAC de la máquina con la IP VIP.

En esa red podría haber varias máquinas con la IP VIP (el balanceador y los servidores comparten dicha IP) con lo cual cualquiera de ellas podría, y de hecho lo hará, responder a la petición. Pero el paquete deberá ir destinado al balanceador no a los servidores.

El balanceador registrará en sus tablas a que servidor le manda las peticiones y consecuentemente todas las peticiones de ese cliente irán al mismo servidor, bajo la conexión ya establecida. Si uno de los servidores respondiera a la petición ARP el router tendría en su tabla ARP la dirección física del servidor y todos los paquetes se los enviará directamente al servidor sin utilizar el balanceador.

Si en algún momento se cambiará la entrada en la tabla ARP y el router actualizará con la MAC de otra máquina (el balanceador y el resto de servidores tienen una interface o alias con la IP VIP) entonces las peticiones de ese cliente iran a otro servidor en lugar de al servidor que originariamente estaban yendo. Si esto pasa dentro de una misma conexión cuando un servidor empiece a recibir las solicitudes de una conexión que el no ha iniciado (la realizó el servidor que primero respondió a la petición ARP) la conexión se cerrará y habrá que volver a negociarla.

Este problema se presenta con núcleos a partir de la serie 2.2.x y se soluciona haciendo que la interface que tiene la IP VIP no responda a peticiones ARP en los servidores y si en el balanceador de carga. De esta forma nos aseguramos que cuando el router haga una petición ARP para la VIP la única máquina que responda sea el balanceador y de esta forma todos los paquetes para el LVS serán enviados al balanceador y este hará su trabajo.

3.4.5. Algoritmos de planificación en *LVS*

Hemos estado haciendo referencia a que el balanceador distribuirá las peticiones entre los servidores. Pero para que esta distribución sea efectiva ha de ser planificada de alguna forma. A la hora de compilar el núcleo en el balanceador tendremos que escoger que algoritmos vamos a utilizar para hacer el balanceo de carga.

Los algoritmos más interesantes son los siguientes:

- *Round-Robin*. Este algoritmo es el más simple y lo que hace es distribuir las peticiones entre los servidores de tal manera que si hay 5 servidores y 100 peticiones cada servidor atenderá a 20 peticiones.

  El orden de distribución de la carga será secuencial, primera petición hacía el primer servidor, segunda al segundo, ..., quinta al quinto, sexta al primero, ...

  Esta distribución es muy sencilla pero presupone que todas las peticiones van a ser equivalentes, en términos de carga, para el servidor, algo que en la realidad dista mucho de ser cierto. O que la capacidad de procesamiento de los servidores es la misma.

  Podría darse el caso de que haya servidores atendiendo varias peticiones y otros esperando o que los servidores más lentos estuvieran muy sobrecargados mientras que los más potentes estuvieran más desahogados.

- *Weighted Round-Robin*. Este algoritmo permite un aprovechamiento mejor del cluster cuando hay máquinas con diferentes capacidades de procesamiento, de esta forma a las máquinas con mayor capacidad de procesamiento se les dará una mayor prioridad (weight) para responder a las peticiones de los clientes y el balanceador distribuirá la carga entre los servidores teniendo en cuenta su prioridad.

  En realidad el Round-Robin Scheduling es un Weighted Round-Robin Scheduling y todas las prioridades son iguales para los servidores.

- *Least-Connection*. Con este algoritmo las peticiones se enviaran al servidor que menos conexiones este sirviendo en ese momento.

  Si la capacidad de procesamiento de los servidores es similar este algoritmo distribuirá la carga de forma óptima entre todas las máquinas del cluster. Sin embargo si las capacidades de procesamiento varían mucho la carga no sera repartida de forma ecuánime ya que la carga se repartirá según el número de conexiones abiertas en ese momento y no sobre la carga real de cada máquina.

- *Weighted Least-Connection*. Este algoritmo es al *Least-Connection Scheduling* lo que el *Weighted Round-Robin Scheduling* es al *Round-Robin Scheduling*.

A cada servidor se le asigna una prioridad según su capacidad de procesamiento y aquellos que mas prioridad tengan asignada atenderán más peticiones, es decir tendrún mús conexiones abiertas.

3.4.6. ipvsadm

ipvsadm es una herramienta en espacio de usuario para interactuar con *LVS*.

Podemos ver el listado de conexiones:

```
[jadebustos@dedalo ~]# ipvsadm -Ln
IP Virtual Server version 1.2.0 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  172.16.0.207:443 wlc persistent 3600
  -> 172.16.0.199:443             Route   1      5          0
  -> 172.16.0.198:443             Route   1      2          0
TCP  172.16.0.205:9085 wlc
  -> 172.16.0.200:9085            Route   1      2          0
  -> 172.16.0.201:9085            Route   1      2          0
TCP  172.16.0.206:80 wlc
  -> 172.16.0.197:80              Route   1      1          0
  -> 172.16.0.195:80              Route   1      4          0
  -> 172.16.0.196:80              Route   1      2          0
TCP  172.16.0.207:80 wlc persistent 3600
  -> 172.16.0.199:80              Route   1      1          0
  -> 172.16.0.198:80              Route   1      1          0
TCP  172.16.0.205:80 wlc
  -> 172.16.0.200:80              Route   1      1          0
  -> 172.16.0.201:80              Route   1      1          0
[jadebustos@dedalo ~]#
```

Podemos sacar estadísticas:

```
[jadebustos@dedalo ~]# ipvsadm --list --stats --numeric
IP Virtual Server version 1.2.0 (size=4096)
Prot LocalAddress:Port               Conns   InPkts  OutPkts  InBytes OutBytes
  -> RemoteAddress:Port
TCP  172.16.0.207:443                    7      169      292        x        y
  -> 172.16.0.199:443                    5      113      214        x        y
  -> 172.16.0.198:443                    2       56       78        x        y
TCP  172.16.0.205:9085                   4       67       99        x        y
  -> 172.16.0.200:9085                   2       34       45        x        y
  -> 172.16.0.201:9085                   2       33       44        x        y
TCP  172.16.0.206:80                     7      109      178        x        y
  -> 172.16.0.197:80                     1       12       23        x        y
  -> 172.16.0.195:80                     4       67      100        x        y
  -> 172.16.0.196:80                     2       30       45        x        y
TCP  172.16.0.207:80                     2       30       58        x        y
```

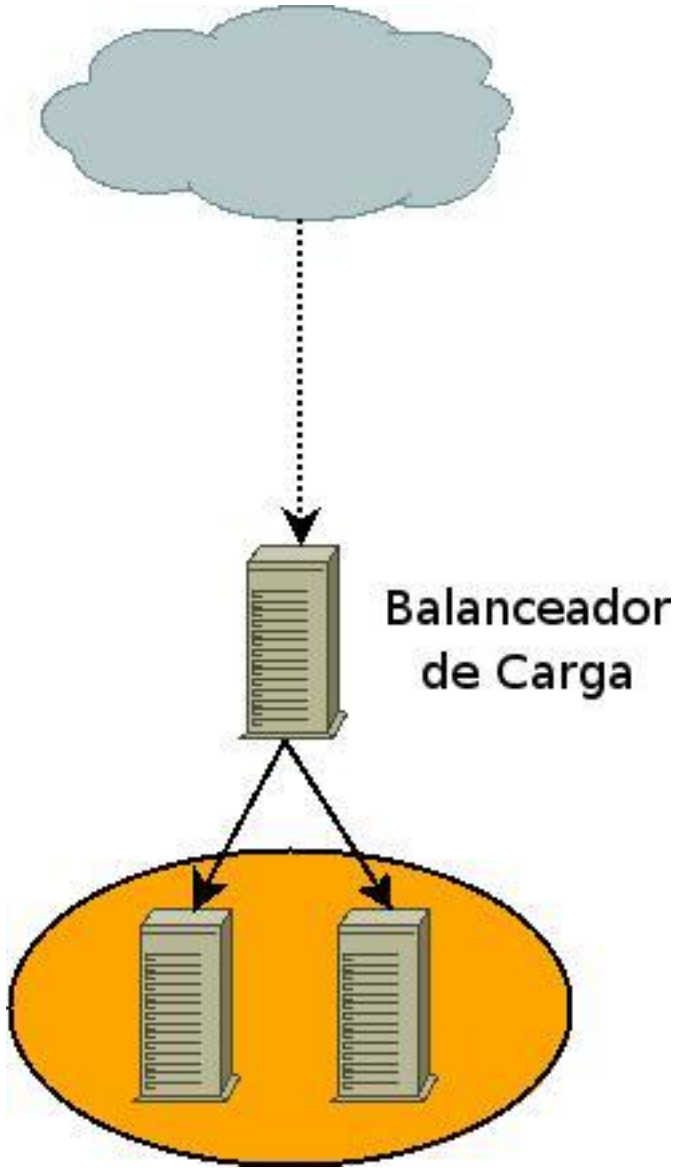
```
  -> 172.16.0.199:80                  1          13         25         x          y
  -> 172.16.0.198:80                  1          17         33         x          y
TCP  172.16.0.205:80                  2          27         50         x          y
  -> 172.16.0.200:80                  1          15         27         x          y
  -> 172.16.0.201:80                  1          12         23         x          y
[jadebustos@dedalo ~]#
```

Otras opciones que nos permite:

- Añadir, modificar y borrar servicios.
- Añadir, modificar y borrar servidores.
- Modificar los parámetros de configuración de los servicios balanceados.

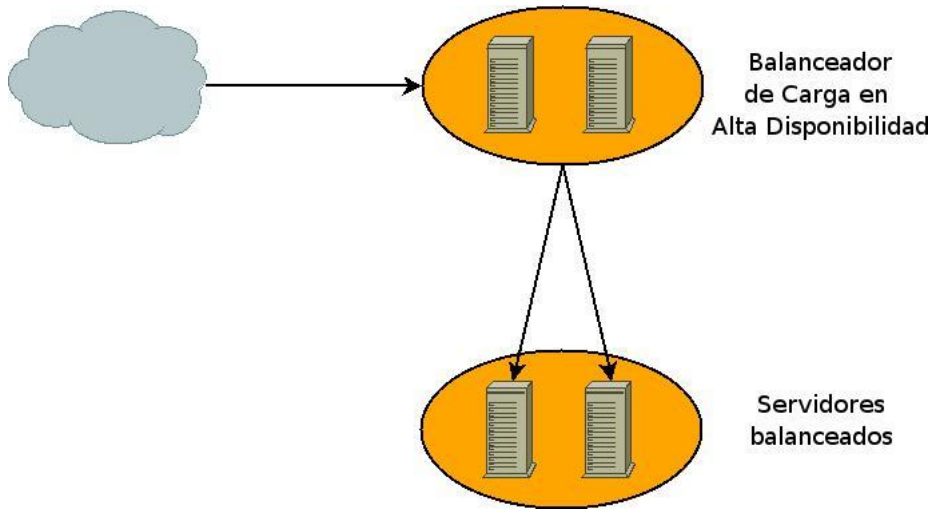
3.4.7. Alta disponibilidad en los balanceadores con *keepalived*

El balanceador es un punto de fallo crítico. Si el balanceador no está disponible no habrá servicio aunque las máquinas que den el servicio estén en perfecto estado de funcionamiento.



Fallo en balanceador.

La alta disponibilidad se lográ garantizando que siempre haya un balanceador funcionando, resumiendo el servicio siempre tiene que estar disponible.



Balanceadores en Alta Disponibilidad.

Keepalived (http://www.keepalived.org) es un demonio que se encarga de que siempre haya un balanceador balanceanado el servicio (*failover*). En realidad es un interface a *LVS*.

Además también se encarga de que no se envíen peticiones a servidores que en ese momento no puedan atenderlas (*health-checking*).

Keepalived se instala en los balanceadores. Uno de ellos será el *MASTER* y el resto serán denominados de *BACKUP*. Cuando el *MASTER* deje de estar operativo entrará en funcionamiento el balanceador *BACKUP* de más alta prioridad y continuará balanceando hasta que el *MASTER* vuelva a estar operativo o hasta que tenga algún problema, entonces lo sustituirá otro balanceador de *BACKUP*, el siguiente de más alta prioridad. De esta forma siempre estará balanceando el balanceador de más alta prioridad. En el momento que un balanceador de más alta prioridad que el activo vuelva al servicio asumirá el balanceo.

Keepalived utiliza el protocolo *VRRP* (rfc 2338). *Keepalived* formará un balanceador virtual formado por un balanceador *MASTER* y varios balanceadores *BACKUP* funcionando de la manera antés descrita.

> **Importante:** Los balanceadores informarán al resto de su disponibilidad utilizando el protocolo *VRRP*.

Para el *health-checking* de los servicios balanceados *keepalived* puede realizar la comprobación de varias formas:

- *TCP_CHECK* se hará una petición TCP y si no es respondida se eliminará el servidor de la lista de servidores activos.

- *HTTP_GET* Se solicitará una página del servidor y se comprobará con una que es la correcta mediante un hashing MD5, previamente calculado, en caso de no coincidir el servidor será eliminado de la lista de servidores activos.

  Importante: Para general el hashing MD5 utilizaremos la utilidad **genhash** que nos permitirá generar el hashing directamente desde el servidor.

- *SSL_GET* igual que *HTTP_GET* pero la página será solicitada bajo una conexión SSL por el puerto 443 (https).
- *MISC_CHECK* esta opción nos permitirá comprobar la disponibilidad de los servidores mediante un script creado por nosotros. Si la situación lo requiere podremos comprobar la disponibilidad de los servidores de forma personalizada.

3.4.8. Configuración de *keepalived*

El fichero de configuración de *keepalived* normalmente en `/etc/keepalived.conf` consta de tres partes:

1. *Zona de configuración global* donde se especificará la forma en la que *keepalived* avisará a los administradores de un fallo en el servidor virtual (dirección de correo, servidor SMTP, ...)
2. *Configuración de VRRP* donde indicaremos si el balanceador es *MASTER* o *BACKUP*, la prioridad, cual será la interface por la que tiene acceso al servidor virtual, la IP virtual (VIP) ...
3. *Configuración del servidor virtual* indicando la IP (VIP) y el puerto, protocolo, ... y una entrada real server con los parámetros de cada servidor real, IP (real), puerto, método de health-checking, ...

   Importante: Este fichero será exactamente igual en el *MASTER* y en los *BACKUPS* salvo que en los *BACKUPS* el parámetro *state* en la configuración de *VRRP* será *BACKUP* y en el *MASTER* será *MASTER* como cabría esperar.

Un ejemplo típico del fichero `/etc/keepalived.conf` para una configuración *Direct Routing*:

```
# Configuration File for keepalived

# Configuracion Global

global_defs {
   notification_email {
     alertas@midominio.com
   }
   notification_email_from balanceador1@midominio.com
   smtp_server 192.168.1.10
   smtp_connect_timeout 30
```

```
   lvs_id LVS_DEVEL
}

vrrp_instance wasIntranet {
    state MASTER
    interface eth1
    virtual_router_id 50
    priority 100
    advert_int 1
    wdog-vrrp 1
    virtual_ipaddress {
        172.16.0.205
 172.16.0.206
 172.16.0.207
    }
}

virtual_server 172.16.0.205 80{
    delay_loop 6
    lb_algo wlc
    lb_kind DR
    persistence_timeout 3600
    protocol TCP

    real_server 172.16.0.200 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
    real_server 172.16.0.201 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
}

virtual_server 172.16.0.205 9085 {
    delay_loop 6
    lb_algo wlc
    lb_kind DR
    persistence_timeout 3600
    protocol TCP

    real_server 172.16.0.200 9085 {
        weight 1
        TCP_CHECK {
           connect_port 9085
```

```
            connect_timeout 30
        }
        delay_before_retry 3
    }
    real_server 172.16.0.201 9085 {
        weight 1
        TCP_CHECK {
            connect_port 9085
            connect_timeout 30
        }
        delay_before_retry 3
    }
}

virtual_server 172.16.0.206 80 {
    delay_loop 6
    lb_algo wlc
    lb_kind DR
    persistence_timeout 3600
    protocol TCP

    real_server 172.16.0.195 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
    real_server 172.16.0.196 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
    real_server 172.16.0.197 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
}

virtual_server 172.16.0.207 80 {
    delay_loop 6
    lb_algo wlc
    lb_kind DR
    persistence_timeout 3600
    protocol TCP
```

```
    real_server 172.16.0.198 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
    real_server 172.16.0.199 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 30
        }
        delay_before_retry 3
    }
}

virtual_server 172.16.0.207 443 {
    delay_loop 6
    lb_algo wlc
    lb_kind DR
    persistence_timeout 3600
    protocol TCP

    real_server 172.16.0.198 443 {
         weight 1
         TCP_CHECK {
            connect_port 443
            connect_timeout 30
         }
         delay_before_retry 3
    }
    real_server 172.16.0.199 443 {
         weight 1
         TCP_CHECK {
             connect_port 443
             connect_timeout 30
         }
         delay_before_retry 3
    }
}
```

Capítulo 4. Detección de fallos en los nodos del cluster.

Un cluster debe conocer cuando algún nodo no está disponible para no enviale peticiones.

Esto se hace de dos formas:

- *Heartbeat* es la técnica más habitual. Consiste en comunicarse o bien a través de una interface de red o puerto serie cada cierto tiempo. Si se pierde la comunicación durante un determinado tiempo se supone que el nodo ha caido.
- *Disco de quorum* es una ténica complementaria que lo que se hace es que todos los nodos del cluster escriben su estado en un disco y de esa forma se determina quien está disponible para dar el servicio.

4.1. Heartbeat

Para implementar está ténica los nodos tiene líneas dedicadas, bien sea por red o conexiones serie por las que se comunican de forma continua para verificar el correcto funcionamiento de los nodos.

> **Sugerencia:** Es recomendable utilizar varias líneas de *heartbeat*.

4.2. Disco de quorum

Si un nodo tiene problemas de red y no llega a los otros nodos pensará que los otros nodos no están disponibles e intentará hacerse con el servicio.

Si disponemos de dispositivos serie para el *heartbeat* entonces dispondremos de una forma de comprobar que los otros nodos están funcionando correctamente y el problema es nuestro. De no disponerlo se asumirá de forma incorrecta que los otros nodos han fallado, intentando asumir el control del cluster y produciendo un *Split Brain*.

Para evitar esto se utiliza el disco de quorum. Cada nodo escribe de forma continua su estado en el disco y de esta forma se puede verificar que nodos están disponibles para hacerse con el servicio en el cluster.

> **Importante:** El *disco de quorum* no es una técnica que sustituya al *heartbeat* es una ténica que debe usarse como complemento al *heartbeat*.

Capítulo 5. Fencing

Fencing es la capacidad de un cluster para hacer que uno de sus nodos libere los recursos que tiene ocupados.

También es posible encontrarnos con el témino *STONITH* que es un acrónimo para *S*hoot *T*he *O*ther *N*ode *I*nto *T*he *H*ead (Dispara al otro nodo en la cabeza).

Cuando un nodo no dectecta a otro, mediante *heartbeat* o *disco de quorum*, asume que el otro nodo no está disponible. Si este nodo está en estado pasivo intentará levantar el servicio.

STONITH es una modalidad de *fencing* que consiste en apagar o reiniciar a un nodo para asegurarse que libera los recursos que tiene asignados.

Cuando se tiene alguna duda al respecto se hace *fencing*. Hay diferentes tipos de *fencing*:

- *Power fencing*, consiste en utilizar un dispositivo que pueda apagar o reiniciar el servidor:
  - Tarjetas *RSA* de IBM.
  - *Management Module* en los Blade Centers de IBM.
  - Tarjetas *ILO* de HP.

- *Fabric fencing*, consiste en suprimir el acceso a los discos. Por ejemplo utilizando *SCSI reservations*.

Capítulo 6. Otras soluciones libres para Clustering

6.1. Heartbeat

Heartbeat (http://linux-ha.org) diseñado para detectar cuando se ha caído un servicio en una máquina y administración de cluster.

Heartbeat puede utilizar dispositivos *ethernet* y *de serie* para comunicarse con los otros nodos del cluster. También permite el uso de *discos de quorum* mediante el plugin ipfail (http://www.ultramonkey.org/3/ipfail.html).

6.2. Mon

Mon (http://mon.sourceforge.net) está diseñado para monitorizar la disponibilidad de servicios y lanzar alertas cuando detecte un servicio caído.

6.3. Fake

Fake (http://www.vergenet.net/linux/fake) ha sido diseñado para cambiar un servicio de servidor.

Cuando se detecta que un servicio ha caido en una máquina se utiliza *Fake* para levantar el servicio en otra máquina.

6.4. ldirectord

ldirectord (http://www.ultramonkey.org/3/ldirectord.html) es un demonio que se utilizar para monitorizar servicios y detectar la caída de los mismos..

6.5. Kimberlite

Kimberlite (http://www.linux.org/apps/AppId_7953.html) es una solución de cluster para clusters de dos nodos y permite la conexión a un disco compartido de *quorum* vía *SCSI* o por fibra (*SAN*).

6.6. Ultramonkey

Ultramonkey (http://www.ultramonkey.org) es una solución para balanceo de carga y alta disponibilidad basada en *LVS*.

Ultramonkey utiliza:

- *Heartbeat* en los balanceadores para detectar cuando ha caído un servicio, monitorizar los servidores a balancear mediante *ldirectord* y controlar la IP del servicio balanceado.
- *ldirectord* en los balanceadores para realizar el *health-checking* de los servidores balanceados.

6.7. Red Hat Cluster Suite

Red Hat Cluster Suite (http://www.redhat.com/software/rha/cluster/) es la solución de *Red Hat* para clustering y alta disponibilidad.

Esta solución consta de:

- *GFS* sistema de ficheros de acceso concurrente.
- *CLVM* o *C*lustered *LVM*.
- *Piranha* interface gráfico para *LVS*.
- *system-config-cluster* herramienta gráfica para la configuración del cluster.
- *ccsd* demonio que hace accesible la configuración del cluster por otros nodos.
- *cman* demonio para la administración del cluster.
- *qdiskd* demonio para el manejo de discos de quorum..
- *fenced* demonio para el fencing de dispositivos.
- *rgmanager*.

Importante: *Red Hat Cluster* está disponible en *Debian Etch*.

Apéndice A. GNU Free Documentation License

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of

Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and

3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

GNU FDL Modification Conditions

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is

included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or

rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Sample Invariant Sections list

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

Sample Invariant Sections list

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public

License, to permit their use in free software.