Proyecto Integrado:

"Firewall en cluster de alta disponibilidad"

Administración de Sistemas Informáticos
Ciclo Formativo Grado Superior
IES Gonzalo Nazareno

Arturo Borrero González
arturo.borrero.glez@gmail.com
Abril-Junio de 2011

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

Este es el documento central del Proyecto Integrado para el Ciclo Superior de Formación Profesional (cfgs) de Administración de Sistemas Informáticos (asi) en el curso 2010/2011.

Durante esta memoria se desarrollan asuntos relacionados con la implantación de un cluster de alta disponibilidad cuyo servicio principal es un firewall.

Objetivos

Este Proyecto Integrado se basa en una serie de objetivos principales, entre los que cabe destacar los siguientes:

1. Comprender y entender toda la tecnología, el software y las metodologías de trabajo implicadas en el mundo del clustering de alta disponibilidad.
2. Desarrollar una implantación de un firewall sobre un cluster de alta disponibilidad en modo Activo/Pasivo.

Sobre el clustering

El mundo del clustering es muy amplio. Para situarnos en una primera posición e intentar acceder a él, deben hacerse dos separaciones:

- Clustering de supercomputación: Agrupaciones de máquinas que compartirán ciertos recursos (generalmente procesador y memoria) para desarrollar cálculos de gran coste y complejidad.
- Clustering de alta disponibilidad: Agrupaciones de máquinas que comparten ciertos recursos (cualesquiera) para presentar al cliente un único sistema con una alta tolerancia a fallos.

Aunque compartan la palabra "cluster" y aunque compartan algunas herramientas para su creación/administración, realmente la función de cada tipo de cluster está bien diferenciada.

Del clustering, se esperan las siguientes características a un coste relativamente bajo:

- Alto rendimiento.
- Alta disponibilidad.
- Alta eficiencia.
- Gran escalabilidad.

Existen varias configuraciones posibles, que están relacionadas con el reparto de recursos y con la

función específica de cada nodo. En este proyecto son tratadas dos, que son:

- **Activo/Pasivo →** Cluster donde un nodo funciona como "Master" y otro como "Slave". Esto quiere decir que en el nodo principal estarán todos los recursos desplegados, y el nodo secundario solo entrará en juego cuando se produzca un fallo en el nodo primario.

  En cualquier caso, el cliente solo interactúa con un nodo (Fig. 1).

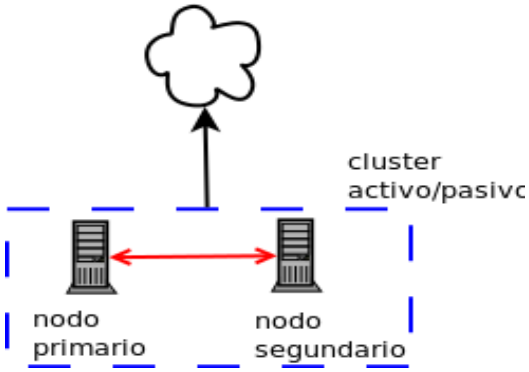


Fig. 1: sólo un nodo interactúa.

- **Activo/Activo →** Los recursos se han repartido por ambos nodos, de modo que ambos pueden responder a las peticiones entrantes en "igualdad" de condiciones.

  El uso principal para este tipo de clusters es el balanceo de carga. El cliente interactúa con más de un nodo, aunque este proceso es transparente (Fig. 2).

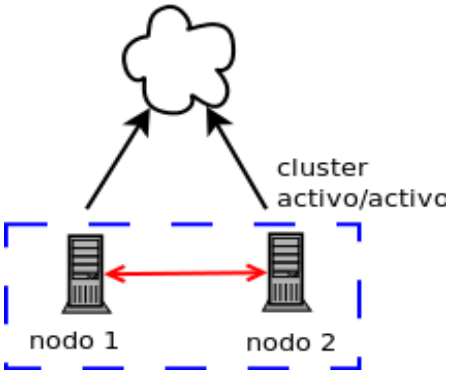


Fig. 2: ambos nodos interactúan.

Además existen otras posibilidades muy avanzadas para la construcción de clusters de alta disponibilidad, y que son:

- **N+1** -> En configuraciones tipo activo-pasivo, para reducir el coste de un nodo de backup que no hace nada, se pueden crear configuraciones en las que el nodo de backup es compartido por varios otros nodos principales (Fig. 3). El sistema sigue siendo de alta disponibilidad, dado que existe tolerancia a fallo en los nodos primarios.

  El cliente interactúa con varias IPVs, según la propia configuración del cluster.

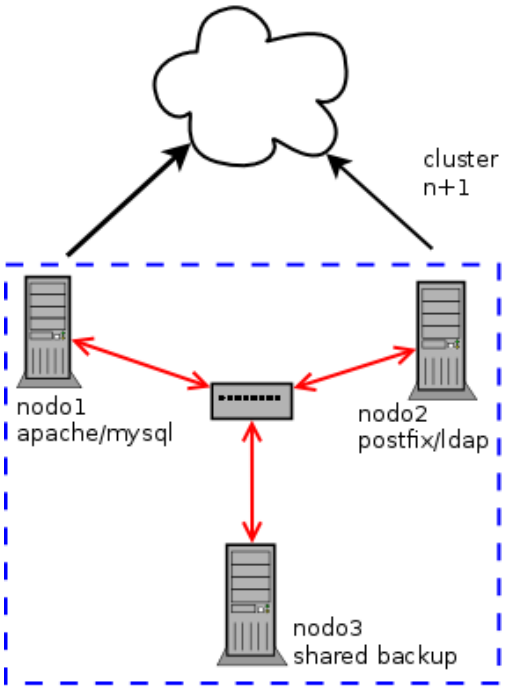


Fig. 3: Se interactua con cada nodo primario.

- **N-to-N** -> Se combinan recursos en alta disponibilidad en múltiples nodos (Fig. 4), incluyendo balanceos de carga, donde cada nodo puede contener cualquier servicio gracias a sistemas de almacenamiento compartido (SAN, etc...).

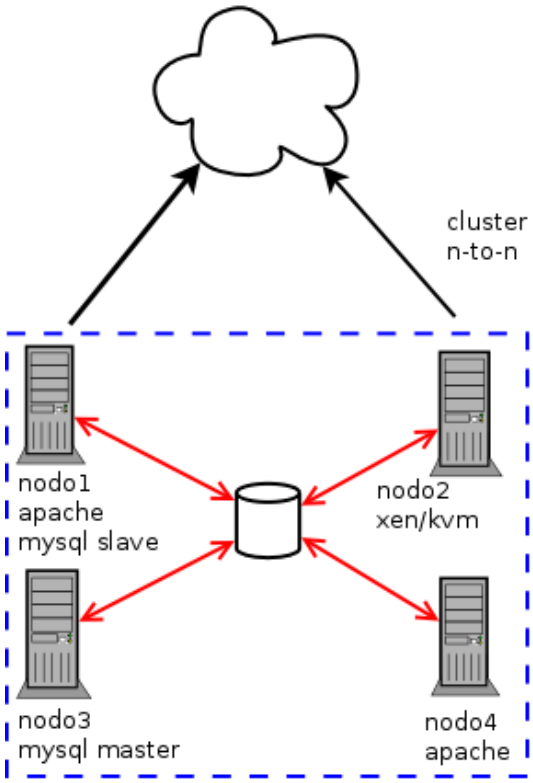


Fig. 4: Múltiples recursos repartidos por todo el cluster.

Concepto importante: recurso

Lo primero que vamos a encontrar al trabajar con un cluster de alta disponibilidad es un recurso. Es fundamental comprender el concepto de recurso, porque nos abrirá la puerta a entendernos con lo que hay detrás y con las posibilidades de configuración

Un recurso es una interpretación de un servicio (a veces, incluso un conjunto de servicios o demonios) que será puesto en alta disponibilidad. Para cada recurso hay un script que sabe cómo manejar los servicios que corresponden, controlar los estados, parar, arrancar, etc...

A priori, un recurso se corresponde con un servicio. Esto quiere decir lo siguiente:

-> Servicio http (apache) -> Recurso http (apache)
-> Servicio sgbd (mysql) -> Recurso sgbd (mysql)
[etc...]

Una vez comprendida e interiorizada esta correspondencia podemos decir que un recurso puede ser algo mucho más completo que un servicio y que tiene características añadidas:

- Movilidad: Un servicio pertenece a una máquina. Un recurso no pertenece a una máquina concreta.
- Integración: Podemos conseguir un grado muy alto de integración entre distintos recursos; relaciones de localización, relaciones de orden de arranque y parada, etc...
- Agrupación: Podríamos decir que un recurso en algunos casos está compuesto de varios servicios, e incluso de otros recursos.

Así que podemos resumir que cuando hablamos de un cluster de alta disponibilidad, nos referimos que se ofrecen recursos y servicios.

Firewall clustering

Dentro del mundo del software libre (GNU/Linux), hablar de técnicas de firewall significa hacer una referencia directa a la herramienta "iptables", del proyecto Netfilter. Puede considerarse la base desde la que surgen o donde se apoyan la mayoría del resto de herramientas para construcción de cortafuegos.

Debido a la naturaleza de este Proyecto Integrado (esto no es un documento científico de investigación y/o análisis), no se tendrán en cuenta herramientas privativas ni firewalls basados en hardware.

Cuando hablamos de "firewall clustering", las referencias a herramientas para construir el cluster son triviales, y por lo general sólo se prestará atención al funcionamiento del conjunto de firewall, a los propio mecanismos del firewall.

En la actualidad pueden encontrarse documentos que describen y valoran con gran precisión

las posibilidades y opciones del firewall clustering, entre las que destacan:

- Firewall en cluster activo/pasivo: Sin balanceo de carga. Un nodo principal filtra todo el tráfico y un nodo de *backup* espera a entrar en acción en caso de fallo del primero. La principal consecuencia es el "desaprovechamiento" de los recursos del nodo de *backup*, que en circunstancias normales no hará nada. (Fig. 5)

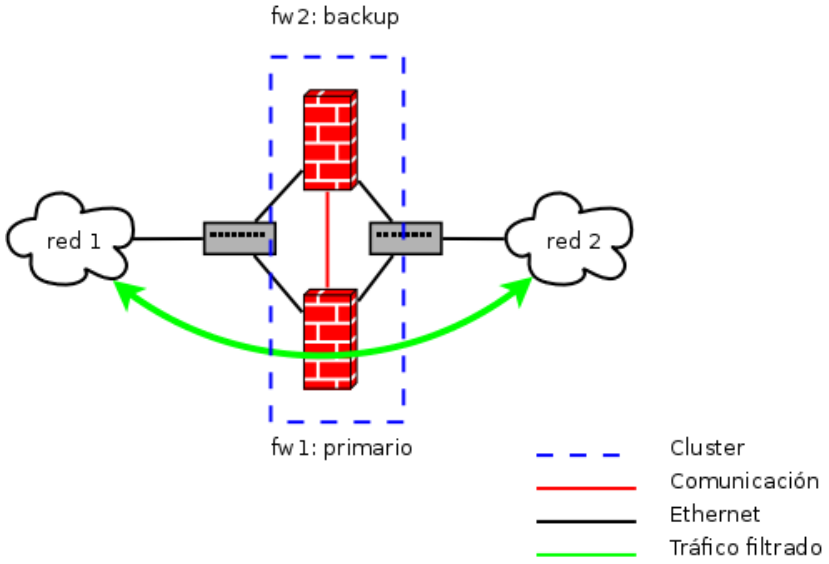


Fig. 5: Firewall en cluster activo/pasivo.

- Firewall en cluster activo/activo con multiruta: Relativo balanceo de carga. Todos los nodos del cluster filtran tráfico, cada nodo trabajando para una ruta o *sentido* de la conexión (por ejemplo, fw1 filtrando el tráfico saliente de la red y fw2 el tráfico entrante). Inviable para firewalls basados en filtros por estados, debido a la necesidad de sincronización instantánea de estados de conexiones.

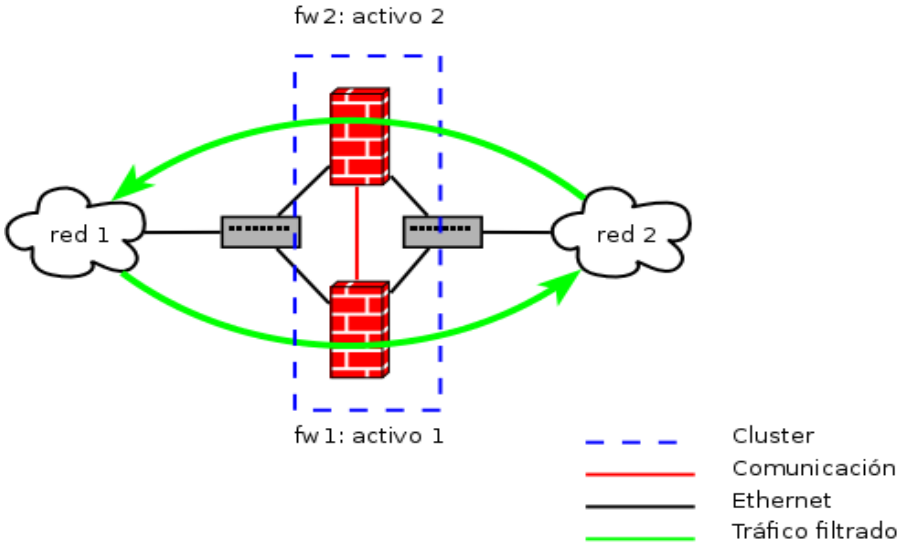


Fig. 6: Firewall en cluster activo/activo con multiruta.

- Firewall en cluster activo/activo con balanceo externo (sandwich): Balanceo de carga provisto por un sistema externo al cluster del firewall. Cada nodo del cluster filtrará el tráfico que le haya sido asignado. Opción con una relación "coste-complejidad-resultados" poco satisfactoria, debido que para un sistema real de alta disponibilidad (Fig. 7) el número de máquinas y servidores a configurar es muy alto (balanceadores de carga redundantes por cada "pata" del firewall, además de los dos servidores para el propio cluster del firewall).

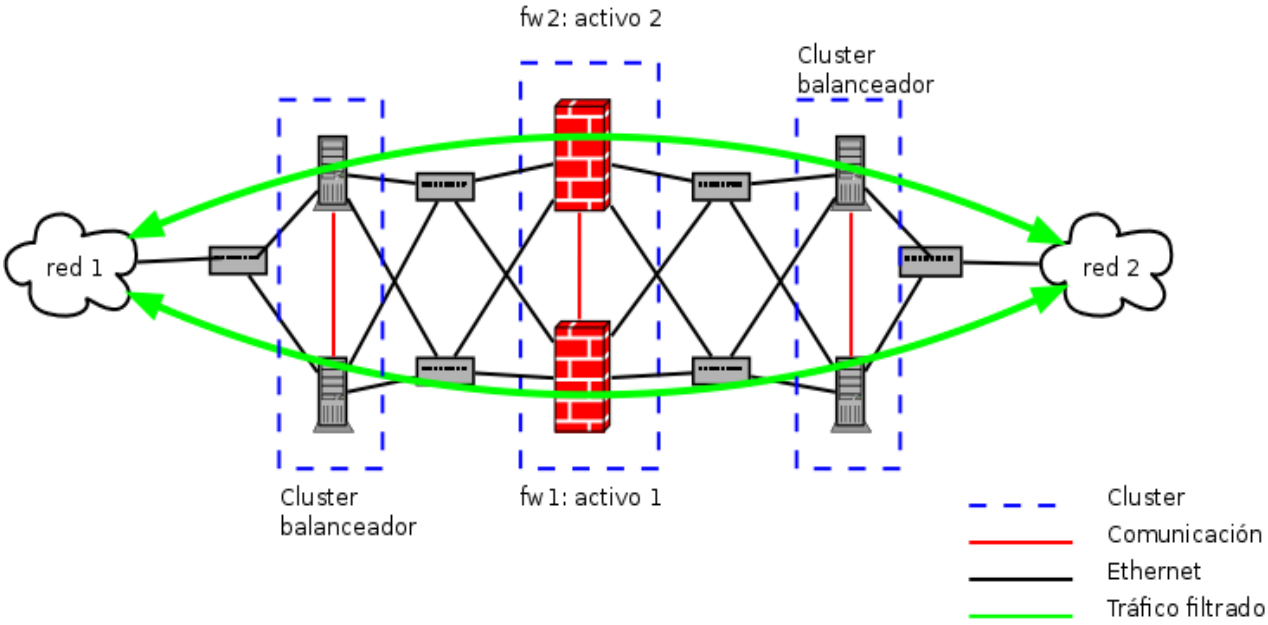


Fig. 7: Cluster firewall activo/activo (sandwich).

- Firewall en cluster activo/activo basados en hash: Con un balanceo de carga interno. Todos los nodos recibirán el mismo tráfico y haciendo operaciones matemáticas sobre el hash de algunos datos (puerto, IP, protocolo, etc..) de cada paquete, cada nodo puede distinguir si es él mismo el encargado de realizar el filtrado (Fig. 8). Válido para cortafuegos de abundante tráfico sobre gran cantidad de reglas de iptables.

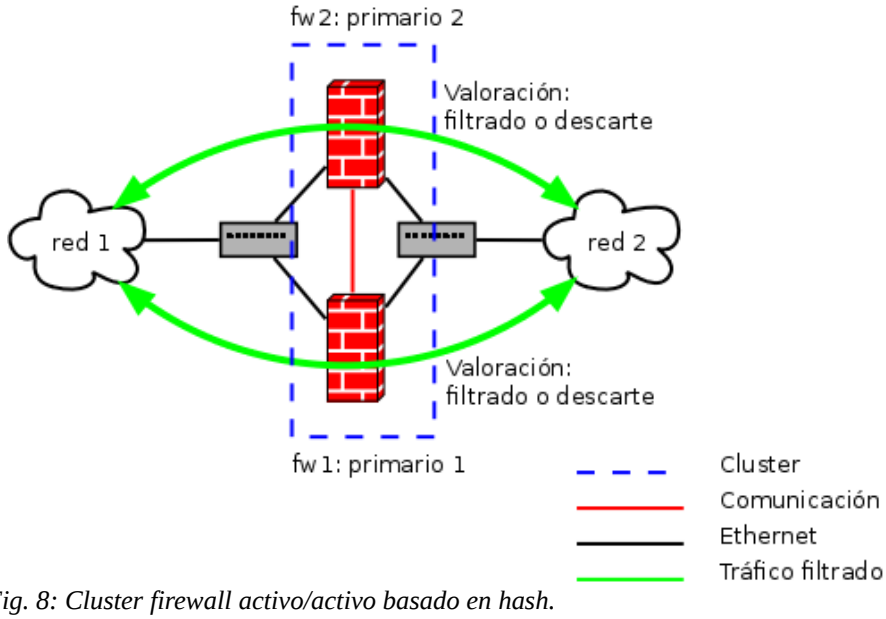


Fig. 8: Cluster firewall activo/activo basado en hash.

PREPARACIÓN

Antes de empezar a montar el cluster, es fundamental tener controlado el entorno de trabajo y los recursos disponibles/necesarios.

Hardware necesario

La cantidad, calidad y capacidad del hardware estará íntegramente relacionada con la dimensión del cluster a construir.

Para un firewall, es de suponer que los elementos principales de cada máquina será la RAM y las interfaces de red. Otras características como el disco duro no toman tanta importancia.

De manera genérica, para un firewall montado en un cluster necesitaremos las siguientes interfaces de red:

- Interfaz dedicada para la comunicación entre nodos. Recomendable usar más de una; el sistema será más completo mientras más redundancia haya.
- Interfaz dedicada a cada zona. Si estamos construyendo un firewall de tres patas, necesitaremos 3 interfaces dedicadas en cada nodo.
- En caso de bonding entre interfaces, el número se incrementa considerablemente, dado que también habrá que hacer el bonding en cada uno de los nodos del cluster.

Entornos virtualizados

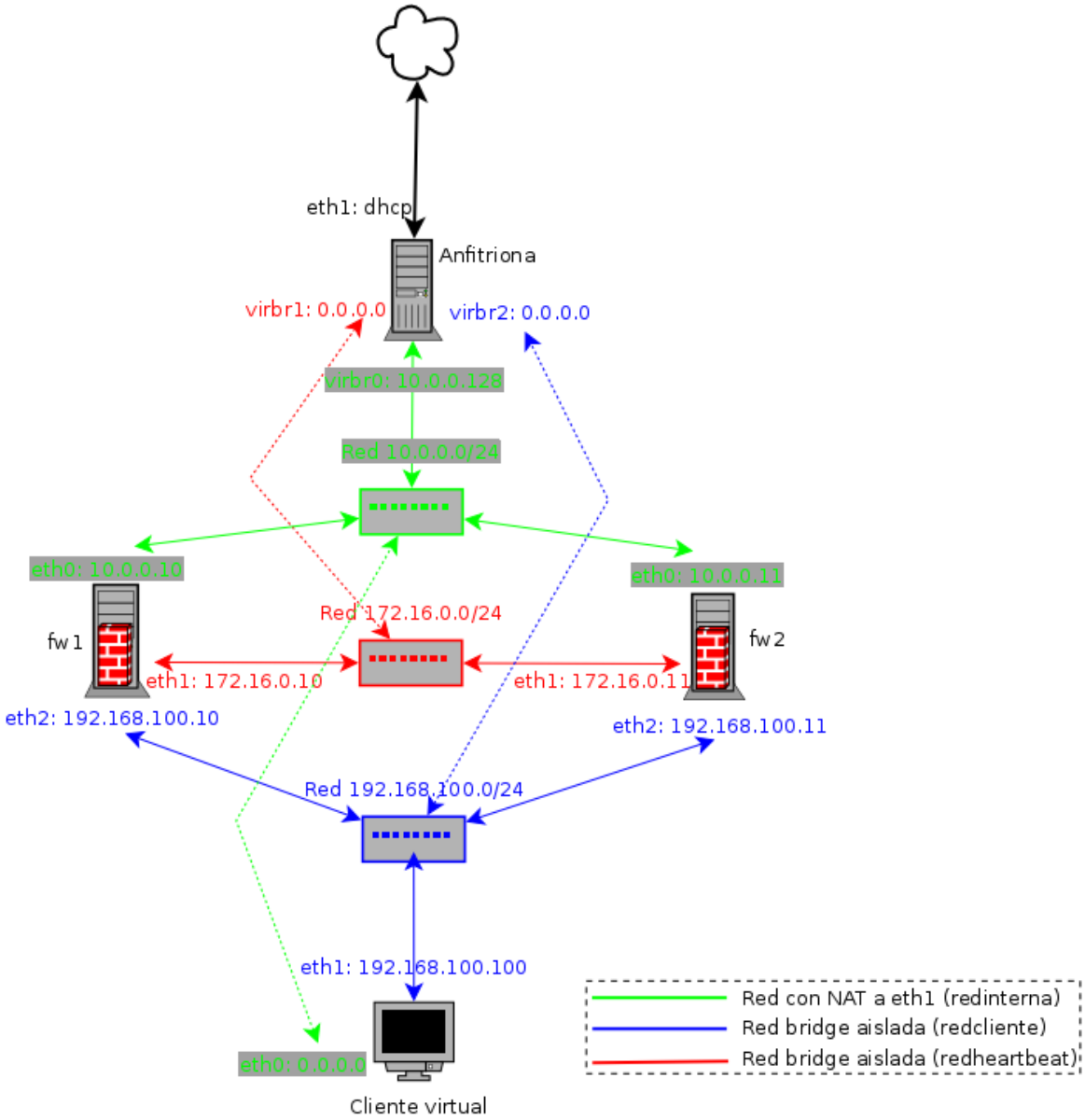
Cuando se construye un cluster en entornos virtualizados (por ejemplo, KVM), es importante tener en cuenta el conexionado de red, que adquiere una dimensión un poco más compleja de lo habitual:

- Por cada "switch" que construiríamos en la realidad, debemos crear una red virtual, a base de interfaces virtuales en modo "bridge".
- Es importante vigilar la tabla de encaminamiento de la máquina anfitriona para obligar a que el tráfico se mueva por donde está planeado.
- A cada máquina virtual hay que definirle sus interfaces, especificando a que red estan "enganchadas".

Los ficheros de configuración para cada red de KVM son los siguientes:

```
root@nostromo:~# cat /etc/libvirt/qemu/networks/redinterna.xml
<network>
  <name>redinterna</name>
  <uuid>a73ee72d-bfe1-fb3c-e682-b5602e42a0a3</uuid>
  <forward dev='eth1' mode='nat'/>
  <bridge name='virbr0' stp='on' delay='0' />
```

```
  <ip address='10.0.0.128' netmask='255.255.255.0'>
  </ip>
</network>
root@nostromo:~# cat /etc/libvirt/qemu/networks/redcliente.xml
<network>
  <name>redcliente</name>
  <uuid>68d8cc18-fe22-2153-6de5-f5dbd79c9292</uuid>
  <bridge name='virbr2' stp='on' delay='0' />
  <ip address='192.168.100.1' netmask='255.255.255.0'>
  </ip>
</network>
root@nostromo:~# cat /etc/libvirt/qemu/networks/redheartbeat.xml
<network>
  <name>redheartbeat</name>
  <uuid>266fa738-4e21-f27d-f5a0-c851df74b4f2</uuid>
  <bridge name='virbr1' stp='on' delay='0' />
  <ip address='172.16.0.1' netmask='255.255.255.0'>
  </ip>
</network>
```



Sofware a usar

Cabe destacar que el software usado es 100% libre y gratuito. Todo el software gira en torno al movimiento GNU/Linux, y es probable en otros entornos tipo UNIX también puedan desarrollarse técnicas de clustering con configuraciones/herramientas similares.

Un cluster básico se compone (como se ha descrito anteriormente) de dos elementos fundamentales, que se corresponden con la gestión de recursos (servicios) y a la comunicación entre nodos para la determinación de estados.
Dependiendo del sistema a montar, trabajaremos con:

- Keepalived (v 1.1.20-1) -> Control de los recursos IPV y manejo de Conntrackd

o bien:

- Corosync (v.1.2.1-4) -> Comunicación entre nodos.
- Pacemaker (v.1.0.9.1+hg15626-1) -> Control de recursos desplegados en alta disponibilidad.

Además, para este cluster que realiza funciones de firewall, se han hecho necesarias otras herramientas:

- Rsync (v. 3.0.7-2) -> Sincronización de ficheros de configuración entre nodos.
- Conntrackd (v. 0.9.14-2) -> Sincronización del estado de las conexiones TCP/IP que son valoradas por el firewall.

Debian

El Sistema Operativo donde se van a realizar todas las pruebas y todo el montaje es Debian GNU/Linux en su versión 6.0 (Debian Squeeze). Es probable que en otras distribuciones de Linux, siendo igualmente válidas, tengan diferentes versiones de los paquetes a usar o incluso que las características propias del sistema obliguen a realizar ciertas tareas de distinta manera a las descritas aquí.

Para el montaje del cluster, es muy recomendable una instalación de Debian Squeeze en modo "netinstall" instalando únicamente el sistema base. El resto de software se instalará a medida que vaya siendo necesitado, desde los repositorios oficiales (aptitude o apt-get).

Keepalived

Software que realiza una implementación del protocolo VRRP. El protocolo está diseñado (según el RFC3768) para aportar redundancia a nodos de una determinada red mediante la abstracción de una IP virtual, que representará a más de un nodo.

Keepalived puede trabajar solo o mediante la conjunción con LVS (Linux Virtual Server) para aportar servicios de alta redundancia y balanceo de carga, aunque este asunto escapa a los objetivos de este proyecto.

La instalación es desde los repositorios oficiales de Debian:

```
root@fw1:~# aptitude install keepalived
root@fw2:~# aptitude install keepalived
```

Cabe señalar que con la instalación de "conntrack-tools" se incluyen ficheros de configuración específicos para Keepalived que proporcionan la funcionalidad básica para el cluster de cortafuegos.

Corosync

Software que proporciona la comunicación directa de estados entre nodos de un cluster de Alta Disponibilidad. A través de Corosync, todas las máquinas conocerán si algún miembro del cluster se encuentra en "failover" o cualquier otro estado que implique imposibilidad de un funcionamiento normal.

La instalación es desde los repositorios oficiales de debian:

```
root@fw1:~# aptitude install corosync
root@fw2:~# aptitude install corosync
```

Podemos usar la configuración que viene de ejemplo, con lo cual:

```
root@fw1:~# mv /etc/corosync/corosync.conf /etc/corosync/corosync.conf.orig
root@fw1:~# cp /etc/corosync/corosync.conf.example /etc/corosync/corosync.conf
```

En principio, todos los nodos del cluster tendrán la misma configuración para Corosync, así que no es mala idea sincronizar los ficheros. Corosync no proporciona ningún método para que la configuración se auto propague. Es posible que en un montaje avanzado, cada nodo posea una configuración distinta (tiempos de espera, transmisión y recepción de datos, etc..), aunque no es el caso.

Pacemaker

Pacemaker es un conjunto de utilidades y herramientas con base al fichero "cib.xml", que es donde está declarado el cluster, todos sus componentes, todos los recursos y todas las restricciones.

Existen varias interfaces para Pacemaker y para manejar el "cib.xml". Las más extendidas:

- CRM – La más común. Proporciona interfaz de comandos (CLI) con una sintaxis muy simplificada pero con una gran potencia.
- DMC – Herramienta gráfica escrita en Java. Puede simplificar bastante las tareas de mantenimiento de un cluster de alta disponibilidad.
- Cibadmin – Programa CLI de edición directa del "cib.xml" con sintaxis muy compleja.

En este proyecto se usa la herramienta "crm", que puede trabajar en modo interactivo (con su propio prompt) o como un programa normal que recibe parámetros de entrada. En ambos casos la sintaxis es idéntica.

Cuando vamos a hacer modificaciones, es recomendable crear un "cib.xml" de tipo temporal (llamado "shadow cib") donde haremos las modificaciones, las validaremos y finalmente confirmaremos:

```
root@fw1:~# crm
crm(live)# cib
crm(live)cib# new
usage: new <shadow_cib> [withstatus] [force]
crm(live)cib# new cfg
INFO: cfg shadow CIB created
crm(live)cib# use cfg
crm(cfg)cib# end
crm(cfg)# configure
crm(cfg)configure# show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="openais"
crm(cfg)configure# show xml
<?xml version="1.0" ?>
<cib admin_epoch="0" cib-last-written="Tue Apr 19 15:06:47 2011" crm_feature_set="3.0.1" dc-uuid="705d40f1-22ea-4be0-a5b7-fff0123809ad" epoch="6" have-quorum="1" num_updates="2" validate-with="pacemaker-1.0">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
                        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-infrastructure" value="openais"/>
      </cluster_property_set>
    </crm_config>
    <rsc_defaults/>
    <op_defaults/>
    <nodes>
      <node id="705d40f1-22ea-4be0-a5b7-fff0123809ad" type="normal" uname="fw1"/>
      <node id="a20e072f-ce5b-41ec-873a-d998180ca5ce" type="normal" uname="fw2"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
</cib>
```

Cabe destacar que el "cib.xml" es propagado automáticamente entre todos los nodos del cluster, por lo que las instrucciones de Pacemaker pueden ejecutarse en cualquier máquina del conjunto.

Configuración de recursos en el cluster

Es posible escribir nuestros propios scripts de control (Resource Agents) para Pacemaker, aunque existen un buen número de scripts ya desarrollados para los recursos más comunes (Bds, IPVs, web, sistemas de ficheros compartidos, etc...).

Si fuera necesario listar los scripts disponibles para recursos, crm nos lo pone fácil:

```
root@fw1:~# crm ra  classes
heartbeat
lsb
ocf / heartbeat pacemaker
stonith

root@fw1:~# crm ra list ocf heartbeat
AoEtarget           MailTo              WAS                 mysql-proxy
AudibleAlarm        ManageRAID          WAS6                nfsserver
CTDB                ManageVE            WinPopup            oracle
ClusterMon          Pure-FTPd           Xen                 oralsnr
Delay               Raid1               Xinetd              pgsql
Dummy               Route               anything            pingd
EvmsSCC             SAPDatabase         apache              portblock
Evmsd               SAPInstance         db2                 postfix
Filesystem          SendArp             drbd                proftpd
ICP                 ServeRAID           eDir88              rsyncd
IPaddr              SphinxSearchDaemon  iSCSILogicalUnit    scsi2reservation
IPaddr2             Squid               iSCSITarget         sfex
IPsrcaddr           Stateful            ids                 syslog-ng
IPv6addr            SysInfo             iscsi               tomcat
LVM                 VIPArip             ldirectord          vmware
LinuxSCSI           VirtualDomain       mysql
```

Podemos ver una descripción del "Resource Agent" que se vaya a usar a continuación:

```
root@fw1:~# crm ra meta ocf:IPaddr
Manages virtual IPv4 addresses (portable version) (ocf:heartbeat:IPaddr)

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Parameters (* denotes required, [] the default):

ip* (string): IPv4 address
    The IPv4 address to be configured in dotted quad notation, for example
    "192.168.1.1".

nic (string, [eth0]): Network interface
    The base network interface on which the IP address will be brought
    online.

    If left empty, the script will try and determine this from the
    routing table.

    Do NOT specify an alias interface in the form eth0:1 or anything here;
    rather, specify the base interface only.

cidr_netmask (string): Netmask
    The netmask for the interface in CIDR format. (ie, 24), or in
    dotted quad notation  255.255.255.0).

    If unspecified, the script will also try to determine this from the
    routing table.
```

```
broadcast (string): Broadcast address
    Broadcast address associated with the IP. If left empty, the script will
    determine this from the netmask.

iflabel (string): Interface label
    You can specify an additional label for your IP address here.

lvs_support (boolean, [false]): Enable support for LVS DR
    Enable support for LVS Direct Routing configurations. In case a IP
    address is stopped, only move it to the loopback device to allow the
    local node to continue to service requests, but no longer advertise it
    on the network.

local_stop_script (string):
    Script called when the IP is released

local_start_script (string):
    Script called when the IP is added

ARP_INTERVAL_MS (integer, [500]): milliseconds between gratuitous ARPs
    milliseconds between ARPs

ARP_REPEAT (integer, [10]): repeat count
    How many gratuitous ARPs to send out when bringing up a new address

ARP_BACKGROUND (boolean, [yes]): run in background
    run in background (no longer any reason to do this)

ARP_NETMASK (string, [ffffffffffff]): netmask for ARP
    netmask for ARP - in nonstandard hexadecimal format.

Operations' defaults (advisory minimum):

    start         timeout=20s
    stop          timeout=20s
    monitor_0     interval=5s timeout=20s
```

Para modificar los recursos, tenemos que entrar en el modo configuración de crm, y tocar los parámetros en función del script que vayamos a usar. Debemos especificar un nombre identificativo para el recurso, que nos servirá después para editarlo, borrarlo, etc..
Esta es la declaración de una IP virtual en la interfaz eth0 del cluster, donde se usa el script "ocf:heartbeat:IPaddr" con parámetros para su configuración:

```
crm(cfg)configure# primitive IPV-inet ocf:heartbeat:IPaddr  \
                        params ip=10.0.0.12 nic=eth0 cidr_netmask=24 \
                        op monitor interval=5s
crm(cfg)configure# show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
primitive IPV-inet ocf:heartbeat:IPaddr \
            params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
            op monitor interval="5s"
property $id="cib-bootstrap-options" \
            dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
```

```
        cluster-infrastructure="openais"
crm(cfg)configure# commit
```

Otra IP virtual en la interfaz (eth2) de la red interna (LAN):

```
crm(cfg)configure# primitive IPV-lan ocf:heartbeat:IPaddr \
                params ip=192.168.100.3 nic=eth2 cidr_netmask=24 \
                op monitor interval=5s
crm(cfg)configure# show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
primitive IPV-inet ocf:heartbeat:IPaddr \
        params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
        op monitor interval="5s"
primitive IPV-lan ocf:heartbeat:IPaddr \
        params ip="192.168.100.3" nic="eth2" cidr_netmask="24" \
        op monitor interval="5s"
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="Heartbeat"
crm(cfg)configure# commit
crm(cfg)configure# exit
bye
root@fw1:~# crm_mon
============
Last updated: Sun Apr 23 11:33:19 2011
Stack: Heartbeat
Current DC: fw1 (a20e072f-ce5b-41ec-873a-d998180ca5ce) - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ fw1 fw2 ]

IPV-inet        (ocf::heartbeat:IPaddr):        Started fw1
IPV-lan (ocf::heartbeat:IPaddr):        Started fw2
```

Cuando los recursos hayan terminado de definirse y estemos trabajando con un cib shadow, es necesario hacer un commit para que se guarden al sistema. Es posible que haya algunos fallos si STONITH no está configurado:

```
crm(cfg)configure# verify
crm_verify[3534]: 2011/04/19_19:18:13 ERROR: unpack_resources: Resource start-up disabled since no STONITH resources have been defined
crm_verify[3534]: 2011/04/19_19:18:13 ERROR: unpack_resources: Either configure some or disable STONITH with the stonith-enabled option
crm_verify[3534]: 2011/04/19_19:18:13 ERROR: unpack_resources: NOTE: Clusters with shared data need STONITH to ensure data integrity
Errors found during check: config not valid
crm(cfg)configure# commit
crm_verify[3550]: 2011/04/19_19:18:17 ERROR: unpack_resources: Resource start-up disabled since no STONITH resources have been defined
crm_verify[3550]: 2011/04/19_19:18:17 ERROR: unpack_resources: Either configure some or disable STONITH with
```

```
the stonith-enabled option
crm_verify[3550]: 2011/04/19_19:18:17 ERROR: unpack_resources: NOTE: Clusters with shared data need
STONITH to ensure data integrity
Errors found during check: config not valid
Do you still want to commit? Yes
crm(cfg)configure# exit
```

Para avisar al cluster de que de momento trabajaremos sin STONITH, usamos lo siguiente:

```
root@fw1:~# crm configure property stonith-enabled=false
root@fw1:~# crm_verify -L
```

Además, como es el caso de un cluster de 2 nodos, es necesario especificar que se trabajará ignorando el "quorum". De lo contrario, en caso del apagado repentino de un nodo, Pacemaker no será capaz de volver a repartir los recursos debido a que considerará que el cluster está incapacitado.

```
root@fw1:~# crm configure property  no-quorum-policy=ignore
```

Según el caso, quizás sea necesario especificar que los recursos del cluster tienen que correr en el mismo nodo. El comportamiento por defecto de Pacemaker es intentar repartir los recursos entre diferentes nodos, buscando equilibrar cargas de trabajo. En el *argot* del clustering, esto se llama "colocación":

```
root@fw1:~# crm configure colocation IPVs INFINITY: IPV-inet IPV-lan
root@fw2:~# crm configure show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
primitive IPV-inet ocf:heartbeat:IPaddr \
        params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
        op monitor interval="5s"
primitive IPV-lan ocf:heartbeat:IPaddr \
        params ip="192.168.100.12" nic="eth2" cidr_netmask="24" \
        op monitor interval="5s"
colocation IPVs inf: IPV-inet IPV-lan
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="Heartbeat" \
        stonith-enabled="false" \
        no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
        resource-stickiness="100"
root@fw1:~# crm_mon
============
Last updated: Sun Apr 24 13:03:59 2011
Stack: Heartbeat
Current DC: fw2 (a20e072f-ce5b-41ec-873a-d998180ca5ce) - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
2 Resources configured.
============
```

```
Online: [ fw1 fw2 ]

IPV-inet       (ocf::heartbeat:IPaddr):        Started fw2
IPV-lan (ocf::heartbeat:IPaddr):       Started fw2
```

Si tubiesemos que ordenar el arranque de recursos (unos antes que otros), se haría con la directiva "mandatory". Algo así especificaría al cluster que el recurso "IPV-inet" debe arrancarse antes que "IPV-lan":

```
root@fw2:~# crm configure order IPV-inet-antes-IPV-lan \
        mandatory: IPV-inet IPV-lan
root@fw2:~# crm configure show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
primitive IPV-inet ocf:heartbeat:IPaddr \
        params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
        op monitor interval="5s"
primitive IPV-lan ocf:heartbeat:IPaddr \
        params ip="192.168.100.12" nic="eth2" cidr_netmask="24" \
        op monitor interval="5s"
colocation IPVs inf: IPV-inet IPV-lan
order IPV-inet-antes-IPV-lan inf: IPV-inet IPV-lan
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="Heartbeat" \
        stonith-enabled="false" \
        no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
        resource-stickiness="100"
```

No son necesarias restricciones de orden de arranque en todos los casos. Están pensadas para grandes combinaciones de recursos, donde (por ejemplo) una base de datos requiere un sistema de ficheros compartidos, etc.. Aquí solo se ha puesto a modo de ejemplo.
Podemos especificar el nivel de preferencia de un recurso a correr en un nodo concreto del cluster. Esto significa que en la medida de lo posible, el recurso se quedará en un nodo específico:

```
root@fw1:~# crm configure location prefer-fw1-IPV-inet IPV-inet 50: fw1
root@fw1:~# crm configure location prefer-fw1-IPV-lan IPV-lan 50: fw1
root@fw2:~# crm configure show
node $id="705d40f1-22ea-4be0-a5b7-fff0123809ad" fw1
node $id="a20e072f-ce5b-41ec-873a-d998180ca5ce" fw2
primitive IPV-inet ocf:heartbeat:IPaddr \
        params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
        op monitor interval="5s"
primitive IPV-lan ocf:heartbeat:IPaddr \
        params ip="192.168.100.12" nic="eth2" cidr_netmask="24" \
        op monitor interval="5s"
location prefer-fw1-IPV-inet IPV-inet 50: fw1
location prefer-fw1-IPV-lan IPV-lan 50: fw1
colocation IPVs inf: IPV-inet IPV-lan
order IPV-inet-antes-IPV-lan inf: IPV-inet IPV-lan
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
```

```
        cluster-infrastructure="Heartbeat" \
        stonith-enabled="false" \
        no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
        resource-stickiness="100"
```

El número especificado está relacionado con el parámetro "resource-stickness", que indica la preferencia de cualquier recurso a quedarse en el nodo que está antes de cambiar de nodo luego de una caida. Esto está pensado para servicios pesados (como Oracle DB), cuyo tiempo de arranque y parada puede ser considerable:

```
root@fw1:~# crm configure rsc_defaults resource-stickiness=100
```

Podemos usar la herramienta "ptest" para visualizar las puntuaciones de los recursos con respecto a cada nodo:

```
root@fw2:~# ptest -sL
Allocation scores:
native_color: IPV-lan allocation score on fw1: 100
native_color: IPV-lan allocation score on fw2: 200
native_color: IPV-inet allocation score on fw1: -1000000
native_color: IPV-inet allocation score on fw2: 100
```

Para indicarle a Pacemaker que debe mover inmediatamente un recurso de un nodo a otro del cluster, podemos usar la siguiente instrucción:

```
root@fw1:~# crm resource move IPV-lan fw1
```

No obstante, esto le quitará el control de los movimientos de recursos a Pacemaker. Para devolver el cluster a la normalidad, moviendose de nuevo los recursos con los parámetros y puntuaciones asignados previamente:

```
root@fw1:~# crm resource unmove IPV-lan
```

Cuando estamos trabajando con nodos, haciendo ajustes a Pacemaker y al resto de configuración podemos "separar" momentaneamente a un nodo del cluster, con las opciones "standby" y "online". Un nodo en modo "standby" no podrá poseer ningún recurso:

```
root@fw2:~# crm node standby
root@fw2:~# crm_mon
============
Last updated: Sun Apr 24 13:44:02 2011
Stack: Heartbeat
Current DC: fw2 (a20e072f-ce5b-41ec-873a-d998180ca5ce) - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
2 Resources configured.
============
```

```
Node fw2 (a20e072f-ce5b-41ec-873a-d998180ca5ce): standby
Online: [ fw1 ]

IPV-inet        (ocf::heartbeat:IPaddr):        Started fw1
IPV-lan (ocf::heartbeat:IPaddr):        Started fw1
root@fw2:~# crm node online
root@fw2:~# crm_mon
============
Last updated: Sun Apr 24 13:46:10 2011
Stack: Heartbeat
Current DC: fw2 (a20e072f-ce5b-41ec-873a-d998180ca5ce) - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, unknown expected votes
2 Resources configured.
============

Online: [ fw1 fw2 ]

IPV-inet        (ocf::heartbeat:IPaddr):        Started fw1
IPV-lan (ocf::heartbeat:IPaddr):        Started fw1
```

En el resto del documento se detalla la construcción del cluster para el firewall usando Pacemaker.

Rsync

Para trabajar con ficheros de configuración genéricos que deban ser iguales en todos los nodos del cluster, Rsync es la herramienta.

De forma genérica, algunos ficheros que será interesante mantener sincronizados:

- El propio script "/usr/local/bin/cluster/sync_files.sh", además del fichero que contiene la lista de archivos a sincronizar ("/etc/rsync.d/sync_files.conf").
- Los ficheros de iptables (para el firewall), en el directorio "/etc/firewall.d/" y también el script LSB de inicio, en "/etc/init.d/firewall".
- Ficheros de configuración relacionados con el entorno de red, como por ejemplo "/etc/resolv.conf" y "/etc/sysctl.conf".
- En el caso de máquinas idénticas en cuanto a hardware, es posible que sea interesante sincronizar la configuración específica de los módulos del sistema, con los ficheros "/etc/modules" y el directorio "/etc/modprobe.d/"
- Ficheros de configuración de Corosync, idénticos en todos los nodos, "/etc/corosync/corosync.conf", "/etc/corosync/service.d/pcmk", "/etc/default/corosync".

Rsync debe ser usado de manera que solo sincronice los ficheros más nuevos. Esto significa que mediante Rsync propagaremos por todos los nodos el fichero de configuración más reciente que se haya encontrado.
Para ello, se usa un pequeño script que se ejecutará cada 30 segundos. El script es de desarrollo propio y está en el fichero "/usr/local/bin/cluster/sync_files.sh":

```
#!/bin/bash
```

```
# Este script hace uso de rsync para sincronizar
# ficheros entre los nodos de un cluster.

# El script esta pensado para ejecutarse varias veces por minuto. (crontab)

# Todas las maquinas que participen deben intercambiar las claves
# para la conexion ssh:
#                       ssh-keygen
#                       ssh-agent $SHELL
#                       ssh-add
#                       ssh-copy-id

# Ademas, es muy importante que las horas esten sincronizadas.
# En caso contrario, el comportamiento es ciertamente problematico.

# Los nombres de cada maquina deben ajustarse a `uname -n`
# No importa de donde venga la lista de nombres mientras se cumpla.
# Debe haber un espacio en blanco entre cada nombre de maquina.

# Se comprueba un lock file para que no se superpongan ejecuciones
# de este script

#########################
# Variables             #
#########################
#LISTA_NODOS=`cat /etc/heartbeat/ha.cf | grep node | awk -F' ' '{print $2}'`
LISTA_NODOS="fw1 fw2"
THIS=`uname -n`
LISTA_FICHEROS="/etc/rsync.d/sync_files.conf"
THIS_PID=$$
LOCK_FILE="/var/run/sync_files.sh.LOCK"

#########################
# Programa              #
#########################

# Validando ejecucion
if [ -e $LOCK_FILE ]
then
        if [ "`cat $LOCK_FILE`" != $THIS_PID ]
        then
                exit 1
        fi
else
        echo "$THIS_PID" > $LOCK_FILE
fi

# Sincronia
for nodo in $LISTA_NODOS
do
    # Si el nodo de la lista no es el propio nodo ejecutando este script
        # se ejecuta la instruccion de rsync
    if [ "$nodo" != "$THIS" ]
    then
         # Se sincronizan los ficheros especificados en $LISTA_FICHEROS
                # Usando como copia maestra el mas reciente.
```

```
            rsync -urv --files-from=$LISTA_FICHEROS $nodo:/ /
        fi
done

# Finalizando ejecucion
rm -f $LOCK_FILE
exit
```

Permisos adecuados para el script (cron puede dar problemas):

```
root@fw1~# chmod ug+x /usr/local/bin/cluster/sync_files.sh
```

Se ha añadido a "/etc/crontab" de la siguiente manera:

```
* *    * * *   root    /usr/local/bin/cluster/sync_files.sh
* *    * * *   root    sleep 30 && /usr/local/bin/cluster/sync_files.sh
```

Paso previo para el correcto funcionamiento de este sistema, deben intercambiarse las claves de ssh de todos los nodos, de manera que el proceso de autenticación sea totalmente transparente y automático.
En todo este proyecto se usan los mismos nodos, con lo cual el intercambio de claves es el siguiente:

- Desde el nodo 1 (fw1)

```
root@fw1:~# ssh-keygen
root@fw1:~# ssh-copy-id 172.16.0.2
```

- Y desde el nodo 2 (fw2):

```
root@fw2:~# ssh-keygen
root@fw2:~# ssh-copy-id 172.16.0.1
```

Para añadir un nuevo fichero que sea necesario sincronizar se debe seguir el siguiente procedimiento:

1. Crear el fichero, con su contenido. En cualquier nodo.
2. Añadir la linea correspondiente al fichero "/etc/rsync.d/rsync_files.conf" del nodo donde primero se agregó el fichero.

Para borrar un fichero que estaba siendo sincronizado, el procedimiento es el siguiente:

1. Borrar el fichero de la lista en "/etc/rsync.d/rsync_files.conf" de cualquier nodo.
2. Borrar manualmente el fichero en el resto de nodos.

Conntrackd

Conntrackd es una herramienta (demonio) del proyecto Netfilter que proporciona transmisión y recepción de los estados de las conexiones valorados por el kernel de Linux.

Se puede encajar dentro de las "conntrack-tools", que incluye al citado Conntrackd y también la herramienta Conntrack, para interactuar "manualmente" contra el estado de las conexiones mantenidas por el kernel.

Entendiendo el funcionamiento

En algunos tipos de clusters es necesario sincronizar discos duros o determinados procesos, pero para el caso concreto de este firewall en alta disponibilidad, tenemos que sincronizar una parte de la memoria RAM de cada máquina, de modo que cada nodo pueda ser "consciente" del estado de las conexiones del otro nodo.

Esto lo hace Conntrackd siguiendo el siguiente esquema de funcionamiento (Fig. 9):

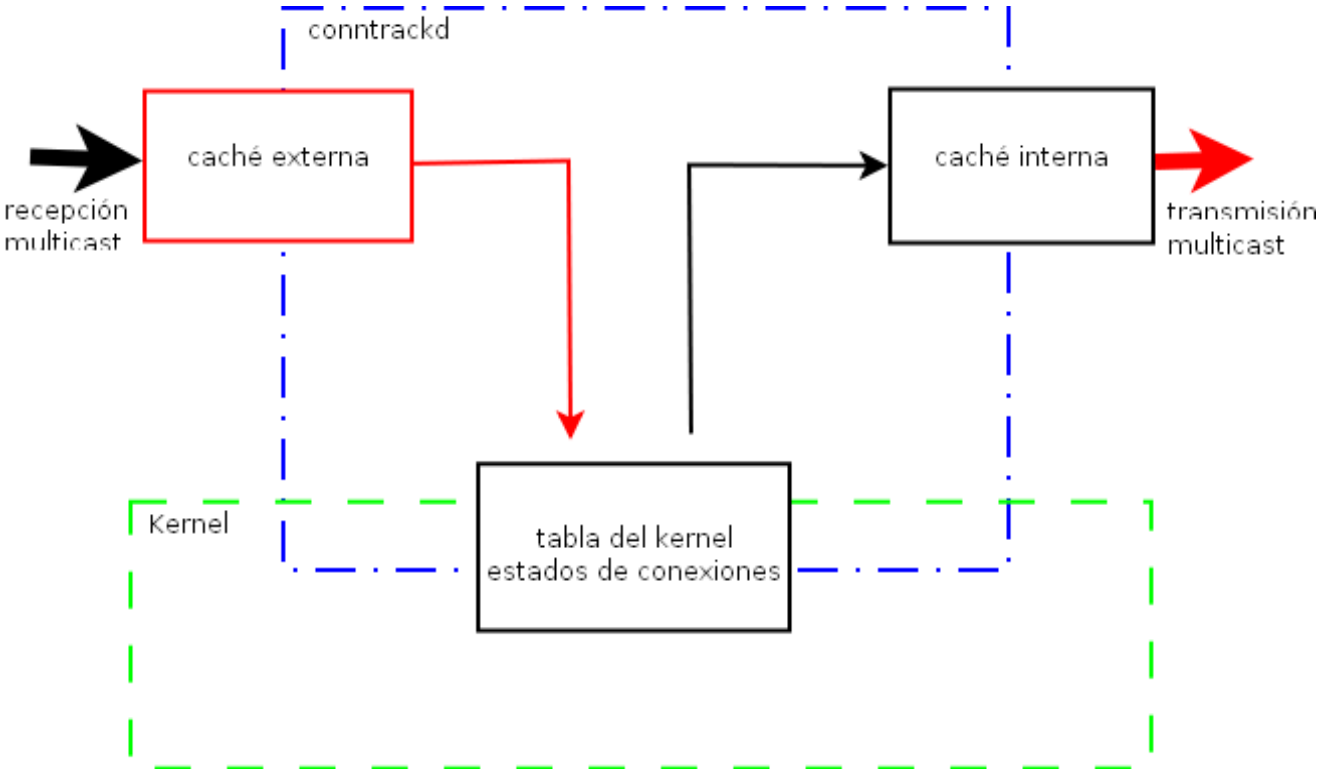


Fig. 9: flujo de datos de Connntrackd

Conntrackd se puede usar en varias configuraciones distintas. Aquí se usará con sincronización periódica (no realtime), que soporta los siguientes modos:

- ***notrack***: (no-seguimiento) Muy simple, envía y recibe información sobre los estados de las conexiones sin realizar ningún checkeo adicional específico.
- ***ft-fw***: (fault-tolerant firewall) Se basa en un protocolo que realiza seguimiento de mensajes, con lo que conntrack puede recuperarse de pérdidas, reordenaciones y corrupción de datos.
- ***alarm***: Este enfoque requiere grandes prestaciones de red. Periódicamente reenvía la caché interna de estado de las conexiones a los nodos de backup, resolviendo problemas de sincronización rápidamente.

Se usará el modo "alarm", que requiere una configuración especial del fichero "/etc/conntrackd/conntrackd.conf" y que será detallado en siguientes secciones.

Requisitos previos

Según puede leerse en la documentación oficial, para trabajar con las "conntrack-tools" necesitamos tener un kernel de Linux de versión superior a la "2.6.18", compilado con soporte para lo siguiente:

- Sistema de seguimiento de conexiones:
  - CONFIG_NF_CONNTRACK=m
  - CONFIG_NF_CONNTRACK_IPV4=m
  - CONFIG_NF_CONNTRACK_IPV6=m (si procediera)
- nfnetlink, intefaz genérica de mensajes para Netfilter:
  - CONFIG_NETFILTER_NETLINK=m
- nf_conntrack_netlink: interfaz de mensajes para el sistema de seguimiento de conexiones:
  - CONFIG_NF_CT_NETLINK=m
- n API: La interfaz de notificaciones basada en tráfico.
  - CONFIG_NF_CONNTRACK_EVENTS=y

Es necesario cargar algunos módulos en el sistema, si no lo estuvieran ya previamente:

- nf_conntrack
- nf_conntrack_ipv4
- nf_conntrack_ipv6 (si procediera)
- nf_conntrack_netlink
- nfnetlink
- nf_defrag_ipv4

Podemos cargar los módulos automáticamente en el arranque del sistema de cada nodo, metiéndolos en el fichero "/etc/modules":

/etc/modules
[...] nf_conntrack nf_conntrack_ipv4 nf_conntrack_netlink nfnetlink nf_defrag_ipv4 [...]

Además, es necesaria la configuración de un parámetro que resultará fundamental en el caso de que se produzca un failover/failback real del cluster y el firewall tenga que valorar el estado de conexiones recibidas mediante Conntrackd.

/etc/sysctl.conf
[...] net.netfilter.nf_conntrack_tcp_loose = 0 [...]

La instalación básica de Debian Squeeze incluye (seguramente) todas las opciones referentes al kernel, así que trabajamos desde los repositorios oficiales. Se instala en todas las máquinas del cluster que queramos que sirvan como firewall:

```
root@fw1:~# aptitude install conntrackd conntrack
[...]
root@fw2:~# aptitude install conntrackd conntrack
```

Compilación (si fuera necesaria)

Si necesitáramos compilar las "conntrack-tools", es necesario seguir este procedimiento:

1. Obtener los paquetes de la web oficial de netfilter.org
   · conntrack-tools-x.x.x.tar.bz2
   · libnfnetlink-x.x.x.tar.bz2
   · libnetfilter-conntrack.x.x.x.tar.bz2
2. Descompresión de los paquetes, con tar:
   # tar xfj *fichero.tar.bz2*
3. Instalación de algunos paquetes necesarios para la compilación, según el sistema operativo:
   # aptitude install gcc flex bison pkg-config make
4. Para compilar las librerias:
   # ./configure && make && make install
5. Para compilar conntrack-tools:
   # ./configure –prefix=/usr && make && make install

Configuración de Conntrackd

Los paquetes instalados vienen con ficheros de configuración con modelos sobre cada modos de funcionamiento. Se pueden encontrar en "/usr/share/doc/conntrackd/examples/".

Como se usará el modo "alarm", para cualquier configuración siguiente es necesario realizar las siguientes acciones:

```
root@fw1:/usr/share/doc/conntrackd/examples/sync/alarm# zcat conntrackd.conf.gz > conntrackd.conf
root@fw1:/usr/share/doc/conntrackd/examples/sync/alarm# cp conntrackd.conf /etc/conntrackd/conntrackd.conf
```

Resource Agent (controlador del recurso)

Como ya se ha visto antes, para manejar un recurso necesitamos un Resource Agent. El de Conntrackd no viene por defecto con Pacemaker (en la versión usada de Debian) ni con otra herramienta propia de clustering y hay que crearlo a mano siguiendo los estándares OCF o buscarlo online. Es posible que en futuras versiones de Pacemaker (>1.0) este Resource Agent se incluya en los paquetes.

El Resource Agent se ha descargado de internet y se ha metido en el sistema en el fichero "/usr/lib/ocf/resource.d/heartbeat/conntrackd". Además, es necesario dar permisos de ejecución a

todos los usuarios.

```
#!/bin/bash
#
#
#       An OCF RA for conntrackd
#       http://conntrack-tools.netfilter.org/
#
# Copyright (c) 2011 Dominik Klein
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of version 2 of the GNU General Public License as
# published by the Free Software Foundation.
#
# This program is distributed in the hope that it would be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
#
# Further, this software is distributed without any warranty that it is
# free of the rightful claim of any third person regarding infringement
# or the like.  Any license provided herein, whether implied or
# otherwise, applies only to this software file.  Patent licenses, if
# any, provided herein do not apply to combinations of this program with
# other software, or any other product whatsoever.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write the Free Software Foundation,
# Inc., 59 Temple Place - Suite 330, Boston MA 02111-1307, USA.
#

#######################################################################
# Initialization:

: ${OCF_FUNCTIONS_DIR=${OCF_ROOT}/resource.d/heartbeat}
. ${OCF_FUNCTIONS_DIR}/.ocf-shellfuncs

#######################################################################

OCF_RESKEY_binary_default=/usr/sbin/conntrackd
OCF_RESKEY_config_default=/etc/conntrackd/conntrackd.conf
: ${OCF_RESKEY_binary=${OCF_RESKEY_binary_default}}
: ${OCF_RESKEY_config=${OCF_RESKEY_config_default}}

meta_data() {
        cat <<END
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="conntrackd">
<version>1.1</version>

<longdesc lang="en">
Master/Slave OCF Resource Agent for conntrackd
</longdesc>

<shortdesc lang="en">This resource agent manages conntrackd</shortdesc>
```

```
<parameters>
<parameter name="conntrackd">
<longdesc lang="en">Name of the conntrackd executable.
If conntrackd is installed and available in the default PATH, it is sufficient to configure the name of the binary
For example "my-conntrackd-binary-version-0.9.14"
If conntrackd is installed somehwere else, you may also give a full path
For example "/packages/conntrackd-0.9.14/sbin/conntrackd"
</longdesc>
<shortdesc lang="en">Name of the conntrackd executable</shortdesc>
<content type="string" default="$OCF_RESKEY_binary_default"/>
</parameter>

<parameter name="config">
<longdesc lang="en">Full path to the conntrackd.conf file.
For example "/packages/conntrackd-0.9.4/etc/conntrackd/conntrackd.conf"</longdesc>
<shortdesc lang="en">Path to conntrackd.conf</shortdesc>
<content type="string" default="$OCF_RESKEY_config_default"/>
</parameter>
</parameters>

<actions>
<action name="start"   timeout="30" />
<action name="promote"   timeout="30" />
<action name="demote"   timeout="30" />
<action name="notify"   timeout="30" />
<action name="stop"    timeout="30" />
<action name="monitor" depth="0"  timeout="20" interval="20" role="Slave" />
<action name="monitor" depth="0"  timeout="20" interval="10" role="Master" />
<action name="meta-data"  timeout="5" />
<action name="validate-all"  timeout="30" />
</actions>
</resource-agent>
END
}

meta_expect()
{
        local what=$1 whatvar=OCF_RESKEY_CRM_meta_${1//-/_} op=$2 expect=$3
        local val=${!whatvar}
        if [[ -n $val ]]; then
                # [, not [[, or it won't work ;)
                [ $val $op $expect ] && return
        fi
        ocf_log err "meta parameter misconfigured, expected $what $op $expect, but found ${val:-unset}."
        exit $OCF_ERR_CONFIGURED
}

conntrackd_is_master() {
        # You can't query conntrackd whether it is master or slave. It can be both at the same time.
        # This RA creates a statefile during promote and enforces master-max=1 and clone-node-max=1
        ha_pseudo_resource $statefile monitor
}

conntrackd_set_master_score() {
        ${HA_SBIN_DIR}/crm_master -Q -l reboot -v $1
}
```

```
conntrackd_monitor() {
	rc=$OCF_NOT_RUNNING
	# It does not write a PID file, so check with pgrep
	pgrep -f $OCF_RESKEY_binary && rc=$OCF_SUCCESS
	if [ "$rc" -eq "$OCF_SUCCESS" ]; then
		# conntrackd is running
		# now see if it acceppts queries
		if ! $OCF_RESKEY_binary -C $OCF_RESKEY_config -s > /dev/null 2>&1; then
			rc=$OCF_ERR_GENERIC
			ocf_log err "conntrackd is running but not responding to queries"
		fi
		if conntrackd_is_master; then
			rc=$OCF_RUNNING_MASTER
			# Restore master setting on probes
			if [ $OCF_RESKEY_CRM_meta_interval -eq 0 ]; then
				conntrackd_set_master_score $master_score
			fi
		else
			# Restore master setting on probes
			if [ $OCF_RESKEY_CRM_meta_interval -eq 0 ]; then
				conntrackd_set_master_score $slave_score
			fi
		fi
	fi
	return $rc
}

conntrackd_start() {
	rc=$OCF_ERR_GENERIC

	# Keep trying to start the resource;
	# wait for the CRM to time us out if this fails
	while :; do
		conntrackd_monitor
		status=$?
		case "$status" in
		$OCF_SUCCESS)
			rc=$OCF_SUCCESS
			conntrackd_set_master_score $slave_score
			break
			;;
		$OCF_NOT_RUNNING)
			ocf_log info "Starting conntrackd"
			$OCF_RESKEY_binary -C $OCF_RESKEY_config -d
			;;
		$OCF_RUNNING_MASTER)
			ocf_log warn "conntrackd already in master mode, demoting."
			ha_pseudo_resource $statefile stop
			;;
		$OCF_ERR_GENERIC)
			ocf_log err "conntrackd start failed"
			rc=$OCF_ERR_GENERIC
			break
			;;
		esac
```

```
        done
        return $rc
}

conntrackd_stop() {
        rc=$OCF_ERR_GENERIC

        # Keep trying to bring down the resource;
        # wait for the CRM to time us out if this fails
        while :; do
                conntrackd_monitor
                status=$?
                case "$status" in
                $OCF_SUCCESS)
                        ocf_log info "Stopping conntrackd"
                        $OCF_RESKEY_binary -C $OCF_RESKEY_config -k
                        ;;
                $OCF_NOT_RUNNING)
                        rc=$OCF_SUCCESS
                        break
                        ;;
                $OCF_RUNNING_MASTER)
                        ocf_log warn "conntrackd still master"
                        ;;
                esac
        done
        return $rc

}

conntrackd_validate_all() {
        check_binary "$OCF_RESKEY_binary"
        if ! [ -e "$OCF_RESKEY_config" ]; then
                ocf_log err "Config FILE $OCF_RESKEY_config does not exist"
                return $OCF_ERR_INSTALLED
        fi
        meta_expect master-node-max = 1
        meta_expect master-max = 1
        meta_expect clone-node-max = 1
        meta_expect clone-max = 2

        return $OCF_SUCCESS
}

conntrackd_promote() {
        rc=$OCF_SUCCESS
        if ! conntrackd_is_master; then
                # -c = Commit the external cache to the kernel
                # -f = Flush internal and external cache
                # -R = resync with the kernel table
                # -B = send a bulk update on the line
                for parm in c f R B; do
                        if ! $OCF_RESKEY_binary -C $OCF_RESKEY_config -$parm; then
                                ocf_log err "$OCF_RESKEY_binary -C $OCF_RESKEY_config -$parm failed during promote."
                                rc=$OCF_ERR_GENERIC
                                break
```

```
                fi
        done
        ha_pseudo_resource $statefile start
        conntrackd_set_master_score $master_score
    fi
    return $rc
}

conntrackd_demote() {
    rc=$OCF_SUCCESS
    if conntrackd_is_master; then
        # -t = shorten kernel timers to remove zombies
        # -n = request a resync from the others
        for parm in t n; do
            if ! $OCF_RESKEY_binary -C $OCF_RESKEY_config -$parm; then
                ocf_log err "$OCF_RESKEY_binary -C $OCF_RESKEY_config -$parm failed during demote."
                rc=$OCF_ERR_GENERIC
                break
            fi
        done
        ha_pseudo_resource $statefile stop
        conntrackd_set_master_score $slave_score
    fi
    return $rc
}

conntrackd_notify() {
    hostname=$(hostname)
    # OCF_RESKEY_CRM_meta_notify_master_uname is a whitespace separated list of master hostnames
    for master in $OCF_RESKEY_CRM_meta_notify_master_uname; do
        # if we are the master and an instance was just started on another node:
        # send a bulk update to allow failback
                    if [ "$hostname" = "$master" -a "$OCF_RESKEY_CRM_meta_notify_type" = "post" -a
"$OCF_RESKEY_CRM_meta_notify_operation" = "start" -a "$OCF_RESKEY_CRM_meta_notify_start_uname" !=
"$hostname" ]; then
            ocf_log info "Sending bulk update in post start to peers to allow failback"
            $OCF_RESKEY_binary -C $OCF_RESKEY_config -B
        fi
    done
    for tobepromoted in $OCF_RESKEY_CRM_meta_notify_promote_uname; do
        # if there is a promote action to be executed on another node:
        # send a bulk update to allow failback
                if [ "$hostname" != "$tobepromoted" -a "$OCF_RESKEY_CRM_meta_notify_type" = "pre" -a
"$OCF_RESKEY_CRM_meta_notify_operation" = "promote" ]; then
            ocf_log info "Sending bulk update in pre promote to peers to allow failback"
            $OCF_RESKEY_binary -C $OCF_RESKEY_config -B
        fi
    done
}

conntrackd_usage() {
    cat <<EOF
usage: $0 {start|stop|promote|demote|monitor|validate-all|meta-data}
Expects to have a fully populated OCF RA-compliant environment set.
EOF
}
```

```
statefile=conntrackd.${OCF_RESOURCE_INSTANCE}.master

master_score=1000
slave_score=100

if [ $# -ne 1 ]; then
        conntrackd_usage
        exit $OCF_ERR_ARGS
fi

case $__OCF_ACTION in
meta-data)
        meta_data
        exit $OCF_SUCCESS
        ;;
usage)
        conntrackd_usage
        exit $OCF_SUCCESS
esac

# Everything except usage and meta-data must pass the validate test
conntrackd_validate_all || exit

case $__OCF_ACTION in
start)
        conntrackd_start
        ;;
stop)
        conntrackd_stop
        ;;
promote)
        conntrackd_promote
        ;;
demote)
        conntrackd_demote
        ;;
status|monitor)
        conntrackd_monitor
        ;;
notify)
        conntrackd_notify
        ;;
validate-all)
        ;;
*)
        conntrackd_usage
        exit $OCF_ERR_UNIMPLEMENTED
esac
# exit code is the exit code (return code) of the last command (shell function)
```

Puesta en marcha

Como el demonio tiene un script de control en "/etc/init.d" no es necesario arrancarlo manualmente al inicio del sistema: de forma predeterminada Debian lo hará, y si no, Pacemaker.

En cualquier caso:

```
root@fw1:/etc/conntrackd# conntrackd -d
root@fw1:/etc/conntrackd# ps aux | grep conntrackd
root       979  0.0  0.1  12616   764 ?        Ss   Apr27   0:00 /usr/sbin/conntrackd -d
```

Si el script LSB de /etc/init.d no fuera incluido, podemos usar el siguiente:

/etc/init.d/conntrackd

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:            conntrackd
# Required-Start:
# Required-Stop:
# X-Start-Before:corosync
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 6
# Short-Description:    Script de inicio de Conntrackd
# Description:         Script de inicio de Conntrackd creado por Arturo Junio 2011.
### END INIT INFO

###################
# Variables
NAME="conntrackd"
CONNTRACKD_PATH="/usr/sbin/conntrackd"
CONNTRACKD_CONFIG="/etc/conntrackd/conntrackd.conf"
CONNTRACKD_CTLFILE="/var/run/conntrackd.ctl"
CONNTRACKD_LOCKFILE="/var/lock/conntrack.lock"
return_code="0"

###################
# Validaciones
if [ ! -x $CONNTRACKD_PATH ]; then
       echo "ERROR. Expected binary not found at \"`echo -n $CONNTRACKD_PATH`\"."
       exit 1
fi
if [ ! -f $CONNTRACKD_CONFIG ]; then
       echo "ERROR: Expected conntrackd config file at \"`echo -n $CONNTRACKD_CONFIG`\"."
       exit 1
fi

####################
# Ejecucion
case "$1" in
            start)
            $CONNTRACKD_PATH -d
              if [ $? -ne 0 ]
              then
                        echo "ERROR. Failed to start \"`echo -n $CONNTRACKD_PATH`\"."
                    return_code=1
              fi
              ;;
       stop)
```

```
        $CONNTRACKD_PATH -k 2> /dev/null
        if [ $? -ne 0 ]
        then
                echo "ERROR. Failed to stop \"`echo -n $CONNTRACKD_PATH`\". Trying harder..."
            killall "$NAME -d" 2> /dev/null
            if [ $? -ne 0 ]
            then
                echo "ERROR. Failed to stop \"`echo -n $CONNTRACKD_PATH`\" even with \"killall\"."
                  return_code=1
            else
                return_code=0
            fi
        else
                return_code=0
        fi
        ;;
    status)
                if [ ! -e $CONNTRACKD_LOCKFILE ]
        then
                echo "$NAME is not running."
            return_code=0
        else
                pid=`ps aux | grep $NAME | grep "\-d" | grep -v grep | awk -F' ' '{print $2}'`
            echo "$NAME is running, with PID $pid"
            if [ $? != 0 ]
            then
                return_code=1
            else
                return_code=0
            fi
        fi
        ;;
    restart)
        sh /etc/init.d/conntrackd stop
        if [ $? != 0 ]
        then
                return_code=$?
        fi
        sh /etc/init.d/conntrackd start
        if [ $? != 0 ]
        then
                return_code=$?
        fi
        if [ $return_code != 0 ]
        then
                echo "Something went wrong when restarting."
        fi
        ;;
    *)
        echo "Usage: /etc/init.d/conntrackd {start|stop|restart|status}"
        return_code=1
        ;;
esac

exit $return_code
```

CLUSTER KEEPALIVED (activo-pasivo)

Haremos uso del protocolo VRRP implementado en Keepalived para hacer el despliegue de los recursos IPVs. Esta arquitectura sólo soporta un cluster de dos nodos.

Arquitectura

Esquema:

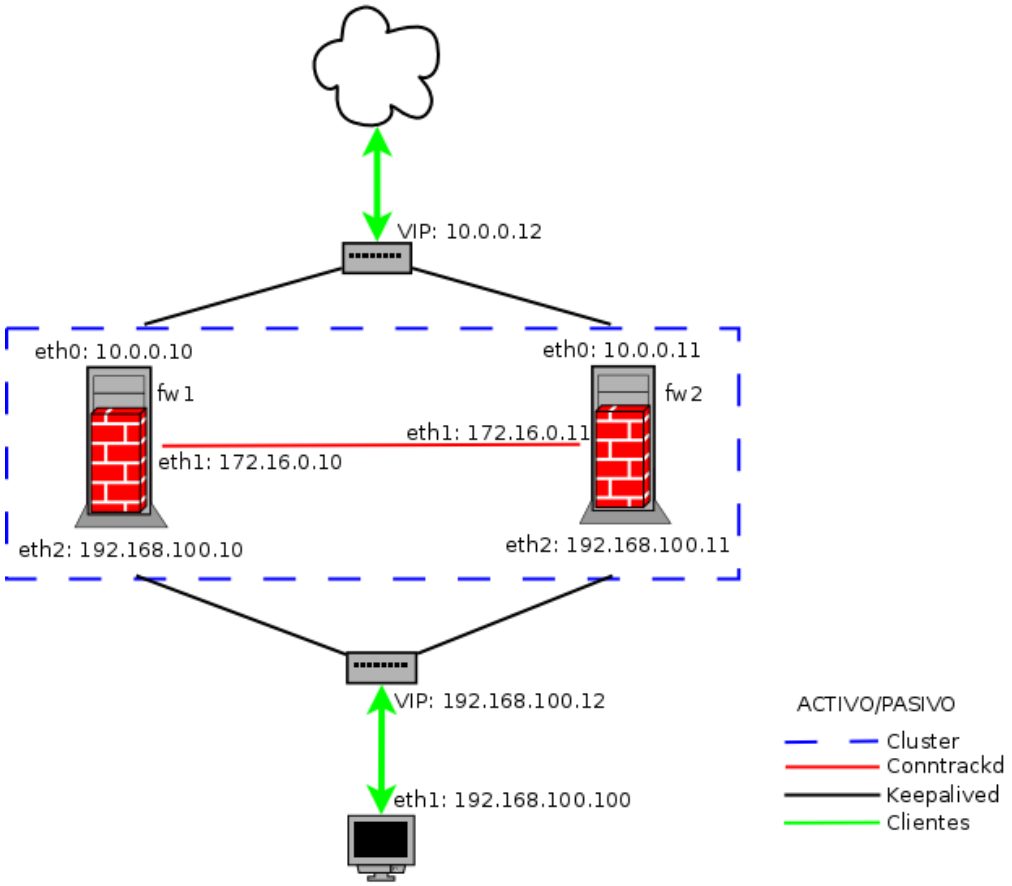


Fig. 10: Cluster Activo/Pasivo con Keepalived

Las interfaces internas (red 172.16.0.0) sólo serán usadas por Conntrackd, dado que Keepalived gestionará la alta disponibilidad en las interfaces públicas mediante anuncios multicast.

Configuración

En este montaje solo es necesario configurar Keepalived y Conntrackd, con lo que la construcción es bastante sencilla.

Keepalived

Fichero en el nodo primario, copiado de "/usr/share/doc/conntrackd/examples/sync/keepalived".

/etc/keepalived/keepalived.conf

```
vrrp_sync_group G1 {
  group {
    VIP-inet
    VIP-lan
  }
  notify_master "/etc/conntrackd/primary-backup.sh primary"
  notify_backup "/etc/conntrackd/primary-backup.sh backup"
  notify_fault "/etc/conntrackd/primary-backup.sh fault"
}
vrrp_instance VIP-inet {
    interface eth0
    state MASTER
    virtual_router_id 61
    priority 80
    advert_int 3
    authentication {
      auth_type PASS
      auth_pass clu$t3r
    }
    virtual_ipaddress {
        10.0.0.12   # default CIDR mask is /32
    }
}
vrrp_instance VIP-lan {
    interface eth2
    state MASTER
    virtual_router_id 62
    priority 80
    advert_int 3
    authentication {
      auth_type PASS
      auth_pass clu$t3r
    }
    virtual_ipaddress {
        192.168.100.12
    }
}
```

Fichero en el nodo segundario:

/etc/keepalived/keepalived.conf

```
vrrp_sync_group G1 {    # must be before vrrp_instance declaration
  group {
    VIP-inet
    VIP-lan
  }
  notify_master "/etc/conntrackd/primary-backup.sh primary"
  notify_backup "/etc/conntrackd/primary-backup.sh backup"
```

```
  notify_fault "/etc/conntrackd/primary-backup.sh fault"
}

vrrp_instance VIP-inet {
    interface eth0
    state BACKUP
    virtual_router_id 61
    priority 80
    advert_int 1
    authentication {
      auth_type PASS
      auth_pass clu$t3r
    }
    virtual_ipaddress {
        10.0.0.12   # default CIDR mask is /32
    }
}

vrrp_instance VIP-lan {
    interface eth2
    state BACKUP
    virtual_router_id 62
    priority 80
    advert_int 1
    authentication {
      auth_type PASS
      auth_pass clu$t3r
    }
    virtual_ipaddress {
        192.168.100.12
    }
}
```

Podemos añadir alguna opción de configuración adicional. Por ejemplo, podemos indicar a Keepalived que monitorice alguna interfaz concreta y que entre en failover si alguna falla:

```
# optional, monitor these as well.
# go to FAULT state if any of these go down.
track_interface {
   eth0
   eth2
}
```

Conntrackd

Para un cluster Activo/Pasivo (este caso), es necesario un script que interactúa con el demonio, y que también viene incluido en la documentación. No es necesaria ninguna modificación al script.

```
root@fw1:/usr/share/doc/conntrackd/examples/sync/# cp primary-backup.sh /etc/conntrackd/
```

Conntrackd necesita un tratamiento especial para funcionar en este tipo de cluster. Es necesario que en caso de caída del un nodo máster, el nuevo nodo que pasa a ser el firewall principal vuelque la caché externa a su tabla del kernel. A partir de este momento, llenará y transmitirá periódicamente su caché interna con los datos de la propia tabla del kernel.

Esta es la configuración final usada para Conntrackd:

```
root@fw1:/etc/conntrackd# cat conntrackd.conf | grep -v ^# | grep -v ^$ | grep -v "^[[:space:]]*#"
Sync {
        Mode ALARM {
                RefreshTime 15

                CacheTimeout 180
        }
        Multicast {
                IPv4_address 225.0.0.50
                Group 3780
                IPv4_interface 172.16.0.10
                Interface eth1
                SndSocketBuffer 1249280
                RcvSocketBuffer 1249280
                Checksum on
        }
}
General {
        Nice -20
        HashSize 32768
        HashLimit 131072
        LogFile on
        LockFile /var/lock/conntrack.lock
        UNIX {
                Path /var/run/conntrackd.ctl
                Backlog 20
        }
        NetlinkBufferSize 2097152
        NetlinkBufferSizeMaxGrowth 8388608
        Filter From Userspace {
                Protocol Accept {
                        TCP
                        SCTP
                        DCCP
                }
                Address Ignore {
                        IPv4_address 127.0.0.1 # loopback
                        IPv4_address 10.0.0.12 # VIP-inet
                        IPv4_address 10.0.0.10
                        IPv4_address 10.0.0.11
                        IPv4_address 192.168.100.12 # VIP-lan
                        IPv4_address 192.168.100.10
                        IPv4_address 192.168.100.11
                        IPv4_address 172.16.0.0./24. # dedicated link ip
                }
        }
}
```

Hay que prestar especial atención a las direcciones IP que se especifican, así como a las interfaces que se usarán para la sincronización de datos entre los nodos. Se indican distintas IPs locales del cluster cuyos estados de los paquetes con ese destino no interesa sincronizar: no sincronizaremos el estado del tráfico dirigido a cada nodo concreto.

Importante señalar que necesitaremos reglas de iptables en el firewall que permitan las transmisiones multicast definidas arriba, así que como mínimo:

```
iptables -I INPUT -d 225.0.0.50 -j ACCEPT
iptables -I OUTPUT -d 225.0.0.50 -j ACCEPT
```

Una configuración correcta de Keepalived debe mostrar mensajes del protocolo VRRP en las redes afectadas (Fig. 11):

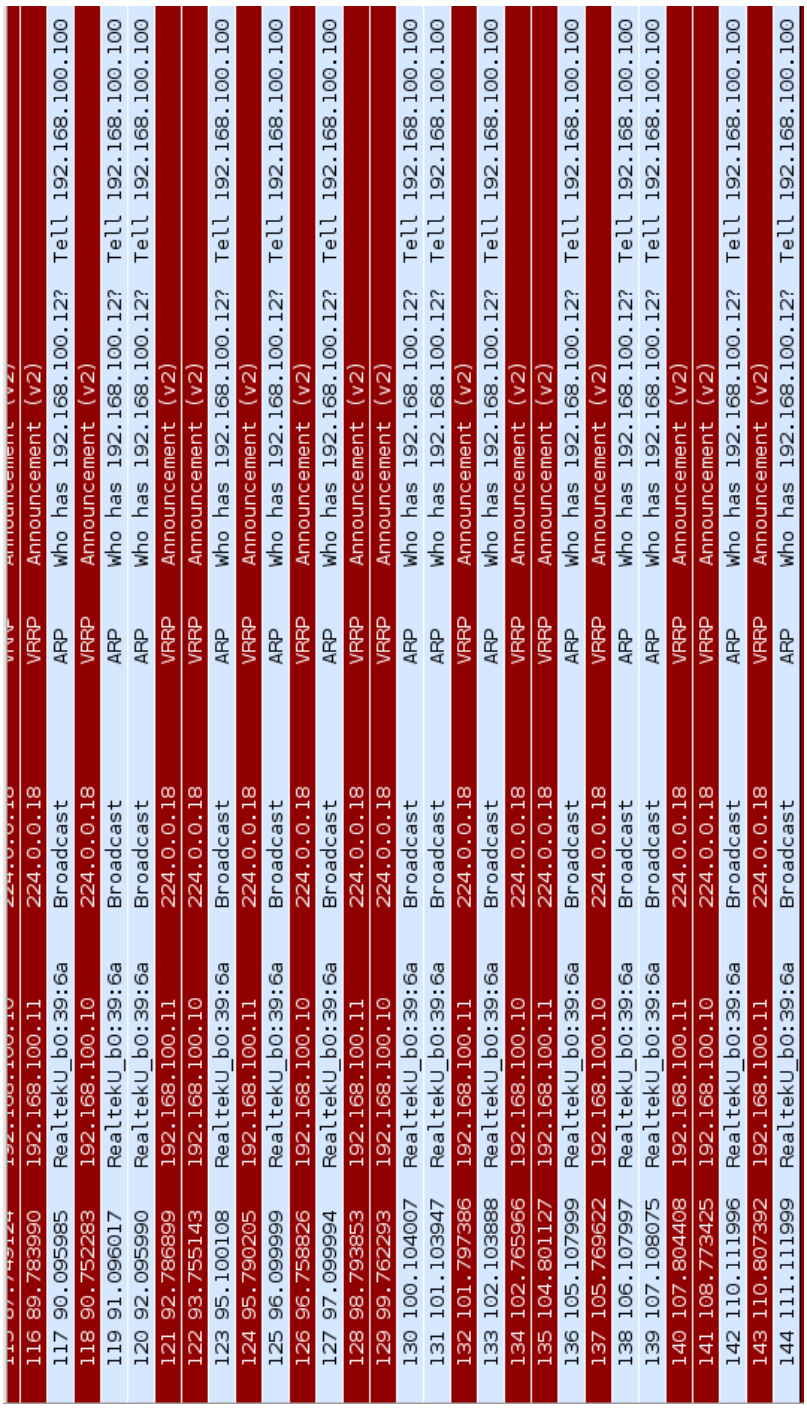


Fig. 11: Tráfico VRRP

CLUSTER CON COROSYNC+PACEMAKER

(activo-pasivo)

Se detalla el funcionamiento, construcción y pruebas para un firewall montado sobre un cluster Activo/Pasivo haciendo uso de la combinación de Corosync y Pacemaker.

Arquitectura

El siguiente esquema (Fig. 12) representa la arquitectura de conexionado y direcciones IPs del cluster Activo/Pasivo.

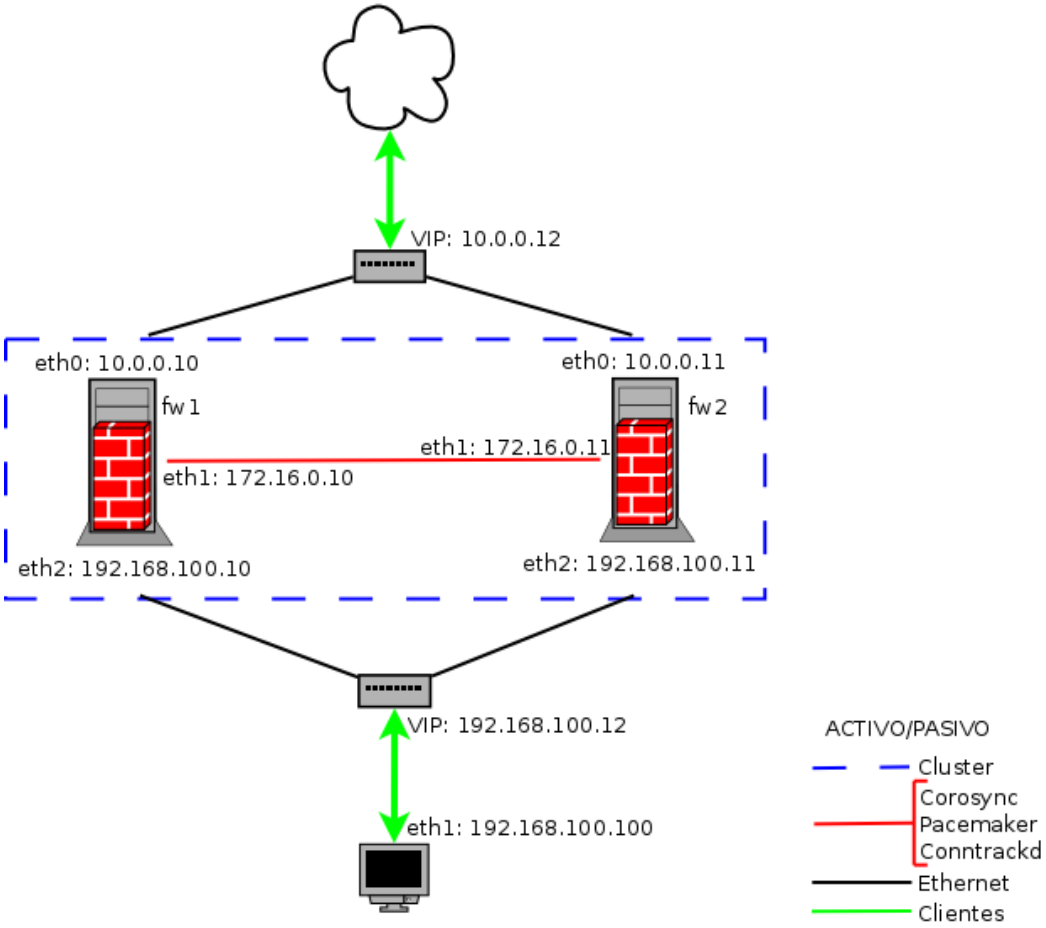


Fig. 12: Cluster Activo/Pasivo con Corosync+Pacemaker

En este caso, por las interfaces internas (red 172.16.0.0) se comunicarán Corosync, Pacemaker y Conntrackd, con lo que el tráfico es mayor. En situaciones de especiales exigencias de disponibilidad, es recomendable usar varias interfaces internas y/o bondings.

Configuración

A continuación se detalla la configuración necesaria para la construcción del cluster Activo/Pasivo.

Corosync

Necesitamos especificar a Corosync que tiene que levantar a Pacemaker como servicio, dado que en esta versión del software (< 1.2), no se proporciona ni siquiera un script LSB tipo "/etc/init.d/" para Pacemaker.
El fichero de configuración principal queda de la siguiente manera:

/etc/corosync/corosync.conf

```
compatibility: whitetank
totem {
        version: 2
        secauth: off
        threads: 0
        interface {
                ringnumber: 0
                bindnetaddr: 172.16.0.0
                mcastaddr: 226.94.1.1
                mcastport: 5405
        }
}
logging {
        fileline: off
        to_stderr: yes
        to_logfile: yes
        to_syslog: yes
        logfile: /tmp/corosync.log
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off
        }
}
amf {
        mode: disabled
}
```

Fichero donde declaramos a Pacemaker, y donde especificamos que es Corosync el que debe levantarlo (en versiones más nuevas de Pacemaker, el valor de "ver" variará):

/etc/corosync/service.d/pcmk

```
service {
      name: pacemaker
      ver:  0
}
```

Y otra importante directiva en otro fichero, sin el cual el servicio de Corosync no arrancará bajo ninguna circunstancia:

/etc/default/corosync

```
START=yes
```

Conntrackd

La configuración final usada para Conntrackd es la misma que en el caso del cluster montado con Keepalived:

```
root@fw1:/etc/conntrackd# cat conntrackd.conf | grep -v ^# | grep -v ^$ | grep -v "^[[:space:]]*#"
Sync {
        Mode ALARM {
                RefreshTime 15

                CacheTimeout 180
        }
        Multicast {
                IPv4_address 225.0.0.50
                Group 3780
                IPv4_interface 172.16.0.10
                Interface eth1
                SndSocketBuffer 1249280
                RcvSocketBuffer 1249280
                Checksum on
        }
}
General {
        Nice -20
        HashSize 32768
        HashLimit 131072
        LogFile on
        LockFile /var/lock/conntrack.lock
        UNIX {
                Path /var/run/conntrackd.ctl
                Backlog 20
        }
        NetlinkBufferSize 2097152
        NetlinkBufferSizeMaxGrowth 8388608
        Filter From Userspace {
                Protocol Accept {
                        TCP
                        SCTP
                        DCCP
                }
                Address Ignore {
                        IPv4_address 127.0.0.1 # loopback
                        IPv4_address 10.0.0.12 # VIP-inet
                        IPv4_address 10.0.0.10
                        IPv4_address 10.0.0.11
                        IPv4_address 192.168.100.12 # VIP-lan
                        IPv4_address 192.168.100.10
```

```
                        IPv4_address 192.168.100.11
                        IPv4_address 172.16.0.0./24. # dedicated link ip
                }
        }
}
```

Importante señalar que necesitaremos reglas de iptables en el firewall que permitan las transmisiones multicast definidas arriba, así que como mínimo:

```
iptables -I INPUT -d 225.0.0.50 -j ACCEPT
iptables -I OUTPUT -d 225.0.0.50 -j ACCEPT
```

Recurso en el cluster

Añadimos el recurso de Conntrackd, llamado "p_conntrackd", especificando que el script del Resource Agent es de Heartbeat (osea, buscará en "/usr/lib/ocf/resource.d/heartbeat/") y añadimos algunas opciones de configuración. Es posible que después de añadir el recurso debamos reiniciar las máquinas implicadas o al menos todos los servicios implicados, de modo que Pacemaker tome el control de los demonios necesarios.

```
root@fw1:~# crm configure primitive p_conntrackd ocf:heartbeat:conntrackd \
        op monitor timeout="20" interval="20" role="Slave" \
        op monitor timeout="20" interval="10" role="Master"
```

El recurso de Conntrackd tiene que estar configurado en modo "master-slave"; implícito en un cluster activo-pasivo. Esto significa que será arrancado en un nodo en modo "master" y otro nodo en modo "slave".
Se hace con la siguiente directiva:

```
root@fw1:~# crm configure ms ms_conntrackd p_conntrackd meta notify="true" interleave="true"
```

Además, necesitamos que el demonio de Conntrack que esté con funciones de maestro corra en el mismo nodo que tiene las IPVs. Pacemaker pondrá automáticamente el recurso esclavo en otro nodo (por eso de la tendencia a repartir las cargas de trabajo).

```
root@fw1:~# crm configure colocation master-conntrackd-ipv-inet \
        inf: ms_conntrackd:Master IPV-inet
```

Lo que cuando generamos tráfico, resulta en la siguiente comunicación entre demonios de Conntrack:

```
root@fw1:~# conntrackd -i
tcp      6 ESTABLISHED src=10.0.0.128 dst=192.168.100.100 sport=48139 dport=22 src=192.168.100.100 dst=10.0.0.128 sport=22 dport=48139 [ASSURED] [active since 4465s]
root@fw1:~# conntrackd -e
root@fw1:~#
root@fw2:~# conntrackd -e
tcp      6 ESTABLISHED src=10.0.0.128 dst=192.168.100.100 sport=48139 dport=22 [ASSURED] [expires in 163s]
```

```
[active since 4484s]
root@fw2:~# conntrackd -i
root@fw2:~#
```

En el nodo que está siendo usado como primario (fw1), hay conexiones en la tabla del kernel, pero no en la caché externa. En el nodo de backup, hay conexiones en la caché externa y no en la caché interna (que refleja la tabla del kernel).

Pacemaker

Necesitamos 3 recursos en Pacemaker, y que son:

- IP virtual de internet "10.0.0.12" en la interfaz eth0. Recurso llamado "IPV-inet".
- IP virtual de la red local "192.168.100.12" en la interfaz eth2. Recurso llamado "IPV-lan".
- Recurso de Conntrack, llamado "p_conntrackd", configurado como Maestro/Esclavo.

Los recursos deben cumplir los siguientes requisitos:

- El nodo preferido para los recursos de IPs virtuales es: fw1 (nodo principal).
- Los recursos "IPV-lan" e "IPV-inet" deben estar en el mismo nodo.
- El recurso "ms_conntrackd" debe estar en modo "máster" en el mismo nodo donde esté el recurso "IPV-inet".

Además, se aplican las siguientes directivas de configuración globales:

- STONITH desabilitado.
- Se ignoran las mediciones de quorum.
- Los recursos se quedarán en el nodo en el que están tras un failover/failback.

Esto genera los siguientes comandos:

```
root@fw1:~# primitive IPV-inet ocf:heartbeat:IPaddr  \
        params ip=10.0.0.12 nic=eth0 cidr_netmask=24 \
        op monitor interval=5s
root@fw1:~# primitive IPV-lan ocf:heartbeat:IPaddr  \
        params ip=192.168.100.12 nic=eth2 cidr_netmask=24 \
        op monitor interval=5s
root@fw1:~# crm configure primitive p_conntrackd ocf:heartbeat:conntrackd \
        op monitor timeout="20" interval="20" role="Slave" \
        op monitor timeout="20" interval="10" role="Master"
root@fw1:~# crm configure ms ms_conntrackd p_conntrackd meta notify="true" interleave="true"
root@fw1:~# crm configure location prefer-fw1-IPV-inet IPV-inet 50: fw1
root@fw1:~# crm configure location prefer-fw1-IPV-lan IPV-lan 50: fw1
root@fw1:~# crm configure colocation IPVs INFINITY: IPV-inet IPV-lan
root@fw1:~# crm configure colocation master-conntrackd-ipv-inet \
        inf: ms_conntrackd:Master IPV-inet
root@fw1:~# crm configure property stonith-enabled=false
root@fw1:~# crm configure property  no-quorum-policy=ignore
root@fw1:~# crm configure rsc_defaults resource-stickiness=100
```

La configuración final de objetos es la siguiente:

```
root@fw2:~# crm configure show
node fw1 \
        attributes standby="off"
node fw2 \
        attributes standby="off"
primitive IPV-inet ocf:heartbeat:IPaddr \
        params ip="10.0.0.12" nic="eth0" cidr_netmask="24" \
        op monitor interval="5s"
primitive IPV-lan ocf:heartbeat:IPaddr \
        params ip="192.168.100.12" nic="eth2" cidr_netmask="24" \
        op monitor interval="5s"
primitive p_conntrackd ocf:heartbeat:conntrackd \
        op monitor interval="20" role="Slave" timeout="20" \
        op monitor interval="10" role="Master" timeout="20" \
        meta target-role="started"
ms ms_conntrackd p_conntrackd \
        meta notify="true" interleave="true"
location prefer-fw1-IPV-inet IPV-inet 50: fw1
location prefer-fw1-IPV-lan IPV-lan 50: fw1
colocation IPVs inf: IPV-inet IPV-lan
colocation master-conntrackd-ipv-inet inf: ms_conntrackd:Master IPV-inet
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="openais" \
        expected-quorum-votes="2" \
        stonith-enabled="false" \
        last-lrm-refresh="1304665751" \
        no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
        resource-stickiness="100"
```

Que ofrece un status como el siguiente:

```
root@fw1:~# crm status
============
Last updated: Sun May  8 22:33:07 2011
Stack: openais
Current DC: fw2 - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, 2 expected votes
3 Resources configured.
============

Online: [ fw2 fw1 ]

 IPV-inet        (ocf::heartbeat:IPaddr):     Started fw1
 IPV-lan (ocf::heartbeat:IPaddr):      Started fw1
 Master/Slave Set: ms_conntrackd
     Masters: [ fw1 ]
     Slaves: [ fw2 ]
```

Grupo de recursos

Es posible combinar varios recursos en uno solo mediante la creación de un grupo. Esto es muy útil para este cluster, dado que podemos agrupar los recursos de IPVs en un único grupo y menejar éste en vez de cada IPV individualmente.

Crear un grupo es extremadamente sencillo:

```
root@fw1:~# crm configure primitive group grupo-IPVs IPV-inet IPV-lan
```

Pacemaker también tratará a un grupo como otro recurso, de manera que es posible realizar agrupaciones de grupos de manera recursiva sin límites (en principio).

Pingd

Pacemaker nos ofrece varias herramientas que podemos usar en cualquier cluster para mejorar o aumentar el nivel de fiabilidad con el que los recursos son manejados.

Una de estas herramientas es "pingd". A priori, pingd solo provee de un método para lanzar automáticamente paquetes ICMP tipo 8 (ping request). Detrás del telón, se está manejando la herramienta standar en entornos GNU/Linux "ping", pero usada y controlada como demonio mediante un Resource Agent.

La mecánica es simple:

1. Creamos un recurso pingd que comprobará la conectividad local contra otra IP de la red.
2. El recurso es clonado a todos los nodos del cluster que queramos que valoren su conectividad.
3. Creamos una *constraint* de tipo *location*, donde especificamos que un recurso **no** puede permanecer en un nodo donde pingd no funcione, esto es, no tenga conectividad contra la IP seleccionada.

Para este cluster en concreto, el uso de la herramienta "pingd" se haría de la siguiente manera:

```
root@fw1:~# crm configure primitive pingd-192.168.0.1 ocf:pacemaker:pingd \
      params host_list=192.168.0.1 multiplier=50 \
      op monitor interval=15 timeout=40s
root@fw1:~# crm configure clone clon_pingd-192.168.0.1 pingd_192.168.0.1 meta globally-unique=false
root@fw1:~# crm configure location l_pingd-192.168.0.1_grupo-IPVs grupo-IPVs \
      rule -inf: not_defined pingd-192.168.0.1 or pingd-192.168.0.1 lte 0
```

De esta manera, los recursos IPVs (y por la configuración del resto del cluster, todos los recursos) sólo serán desplegados en aquel nodo que tenga conexión con 192.168.0.1.

No obstante, **es necesario** tener en cuenta algunas consideraciones al trabajar con pingd:

- La red estará "ensuciada" por tráfico ICMP constante, multiplicado por nodo y por IP con la que queramos valorar la conectividad.
- La caída de una IP con la que se estaba comprobando conectividad puede suponer la caída de recursos asociados a pingd si la configuración del cluster no lo evita. De esta manera, se hace muy recomendable el comprobar conectividad contra IPs en alta disponibilidad

PRUEBAS DE FUNCIONAMIENTO

Existen algunas pruebas que ayudarán a determinar que la configuración del cluster es definitivamente correcta.

Es importante valorar el estado de la configuración de Keepalived/Corosync/Pacemaker (recursos, etc..) y también la configuración propia de las herramientas que están interviniendo (por ejemplo, Conntrackd).

Como estamos hablando de un cluster de alta disponibilidad, lo realmente aconsejable es hacer pruebas de failover/failback "neuróticas" y valorar en profundidad hasta qué punto nuestro cluster soporta un fallo. Esto significa probar los casos más descabellados y todas las distintas posibilidades:

- Apagado de un nodo del cluster mediante las instrucciónes "halt", "reboot" y "shutdown".
- Apagado de un nodo del cluster "agresivamente": tirando del cable, botón de apagado, etc..
- Desconexión de cables de red en distintos puntos de la arquitectura.
- Borrado o cambio de permisos de algún fichero de configuración.
- Cualquier otra cosa que se nos ocurra...

Keepalived y Corosync

La prueba más inmediata a hacer con Keepalived y con Corosync es provocar un fallo de red y ver la reacción del software. Esto incluye desconexión de interfaces, sacar módulos de la memoria, desconexión física de la red, etc..

Además, nos ayudará a hacernos una idea de los tiempos de respuesta ante failover que tiene Keepalived/Corosync, facilitando que podamos hacer un ajuste fino.

En el caso de Keepalived, podemos tocar la directiva "***advert_int***" y probar distintos valores para adaptar los tiempos de respuesta a una situación concreta.

Con Corosync, podemos modificar dos directivas interesantes:

- **secauth** {off | on} -> Cuando está marcado en "on", Corosync trabajará con operaciones en modo seguridad, con un rendimiento operacional y de red menor que con la directiva en "off".

- **threads** {0 | n_cpus-1 } -> Podemos definir entre 0 y el número de cpus de la máquina -1 para el número de hilos que Corosync puede manejar concurrentemente.

- **interface** -> Usando varias veces esta directiva, indicamos a Corosync que tenemos disponibles varias interfaces de red para la comunicación entre nodos (mientras más redundancia, mejor para Corosync, mejor para Conntrackd y mejor para el cluster).

Ajuste de recursos en Pacemaker

Para probar que efectivamente, cuando las IPVs se mueven, el recurso **máster** de Conntrackd se mueve con ellas, podemos probar unos cuantos comandos y ver el resultado con "crm_mon":

```
root@fw1:~# crm resource move IPV-inet fw2
root@fw1:~# crm resource unmove IPV-inet
root@fw1:~# crm resource move IPV-inet fw1
root@fw1:~# crm resource unmove IPV-inet
```

Como ya se ha señalado en secciones anteriores, hay que especificar a Pacemaker que el propio cluster debe volver a tomar el control de la posición del recurso.

Si estamos usando un grupo de recursos:

```
root@fw1:~# crm resource move grupo-IPVs fw2
root@fw1:~# crm resource unmove grupo-IPVs
```

Si la configuración es correcta, el recurso máster de Conntrackd habrá saltado de un nodo a otro.

Los intervalos de monitorización usados por Pacemaker como ejemplo en los recursos son los recomendados por los desarrolladores, y no es aconsejable cambiarlos salvo configuraciones extremadamente personalizadas y avanzadas (sabiendo siempre lo que se está haciendo).

Tráfico

Es fundamental que el firewall filtre tráfico, y para ello debemos "medir" la cantidad datos que es capaz de manejar el cluster.

Cluster al límite: Iperf y AB

Usando la herramienta Iperf con un cliente y un servidor que pasen a través del cluster, podremos determinar el ancho de banda disponible a través del firewall.

Iperf se inicia en modo servidor en una máquina y en modo cliente en otra:

```
root@server:~# iperf -s
usuario@cliente:~$ iperf -C server
```

Una vez pasado el tiempo, Iperf mostrará por pantalla el ancho de banda usado con efectividad durante el test. Usando Iperf podemos hacernos una idea del ancho de banda absoluto que puede manejar el cluster. Si la configuración de la red implicada un firewall de varias "patas" es conveniente hacer los test de ancho de banda entre todas las patas.

Otro test interesante es medir la cantidad de conexiones distintas que puede soportar el cluster. Para

ello, podemos usar la herramienta AB (originalmente diseñada para hacer benchmarking del servidor web apache) para valorar la cantidad de sesiones NEW y ESTABLISHED que puede manejar el cluster.

Monitorizando la carga del sistema y las interfaces de red (tcpdump, top, ps, etc..), lanzamos los siguientes comandos contra un servidor web que esté localizado atravesando el cluster.
Es recomendable ir ascendiendo progresivamente e ir analizando los resultados:

```
(prompt):$ ab -n 10 -c 10 webserver.acrossfw.example.com/
(prompt):$ ab -n 1000 -c 100 webserver.acrossfw.example.com/
(prompt):$ ab -n 100000 -c 1000 webserver.acrossfw.example.com/
```

AB enviará el número de peticiones (-n xxx) HTTP GET "/" indicadas, usando además muchas de ellas de forma concurrente (-c xxx)

Conntrackd sufrirá bastante estrés conforme la cantidad de peticiones vaya aumentando, lo que nos servirá para determinar su funcionamiento en situaciones que se aproximen a la realidad.

```
root@fw1:~# conntrackd -i
[…]
[…]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=60903 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=60903 [ASSURED] [active since 11s]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=59211 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=59211 [ASSURED] [active since 12s]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=58761 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=58761 [ASSURED] [active since 13s]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=58314 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=58314 [ASSURED] [active since 13s]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=58308 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=58308 [ASSURED] [active since 13s]
tcp      6 SYN_SENT src=10.0.0.128 dst=192.168.100.100 sport=56076 dport=80 [UNREPLIED] src=192.168.100.100 dst=10.0.0.128 sport=80 dport=56076 [active since 16s]
tcp      6 TIME_WAIT src=10.0.0.128 dst=192.168.100.100 sport=60679 dport=80 src=192.168.100.100 dst=10.0.0.128 sport=80 dport=60679 [ASSURED] [active since 11s]
[…]
[…]
```

Para valorar el funcionamiento de Conntrackd, que es la misión principal del cluster, es conveniente revisar las estadísticas producidas por la propia herramienta. Podemos ver los datos de la conexión entre nodos, los eventos creados/filtrados/transmitidos, cache, etc..

```
root@fw1:~# conntrackd -s cache
cache:internal      active objects:                 0
          active/total entries:                     0/        0
          creation OK/failed:                  164802/        0
                    no memory available:            0
                    no space left in cache:         0
          update OK/failed:                    639181/        0
                    entry not found:                0
          deletion created/failed:             164802/        0
```

```
                    entry not found:                0

cache:external      active objects:                 0
            active/total entries:                   0/          0
            creation OK/failed:                 93265/          0
                    no memory available:            0
                    no space left in cache:         0
            update OK/failed:                   113969/         0
                    entry not found:                0
            deletion created/failed:            93265/          0
                    entry not found:                0

root@fw1:~# conntrackd -s link
multicast traffic device=eth1 status=RUNNING role=ACTIVE:
          129201844 Bytes sent          14505808 Bytes recv
            946235 Pckts sent           146295 Pckts recv
                0 Error send            0 Error recv

root@fw2:~# conntrackd -s
cache internal:
current active connections:             0
connections created:                    77938       failed:         0
connections updated:                    128412      failed:         0
connections destroyed:                  77938       failed:         0

cache external:
current active connections:             0
connections created:                    622102      failed:         0
connections updated:                    2029619     failed:         0
connections destroyed:                  622102      failed:         0

traffic processed:
            819956 Bytes                6360 Pckts

multicast traffic (active device=eth1):
          35216936 Bytes sent           196431744 Bytes recv
            223255 Pckts sent           1380037 Pckts recv
                0 Error send            0 Error recv

message sequence tracking:
                0 Msgs mfrm             0 Msgs lost
```

Conntrackd

Para probar que Conntrackd realmente está cumpliendo con su función, el test descrito a continuación es bastante recomendable:

Caso A) no Conntrackd activo

1. Cluster funcionando en su plenitud.
2. Matar todos los procesos de Conntrackd en los nodos del cluster.
3. Efectuar una transferencia de un fichero desde una máquina a un lado del firewall a otro. Por

ejemplo, un fichero grande usando SCP (puerto tcp 22).
4. Durante el tiempo de la transferencia, se realiza failover manual del nodo primario o del nodo que esté manteniendo la conexión.
5. La transferencia debe interrumpirse indefinidamente dado que el firewall en el nodo que pasa a filtrar la conexión no reconoce el estado de la misma como válido.

Caso B) si Conntrackd activo

1. Cluster funcionando en su plenitud.
2. Ejecutar transferencia de un fichero desde una máquina a un lado del firewall al otro. Por ejemplo, un fichero grande usando SCP (puerto tcp 22).
3. Durante el tiempo de transferencia, se realiza failover manual del nodo primario o del que esté manteniendo la conexión en ese momento.
4. La transferencia debe sufrir una pequeña relentización momentánea durante la transición de recursos del nodo en failover al nodo que queda activo, pero la conexión es tomada como válida gracias a Conntrackd.
5. Con un funcionamiento normal, la transferencia continua.

Queda señalar que todo el funcionamiento del cluster está supeditado a una configuración de iptables correcta. Si no hay ninguna regla de iptables definida en el sistema, Conntrackd no funcionará dado que no habrá estados de conexiones que valorar.

Además, si el firewall usa reglas “stateless” (sin valorar el estado de las conexiones) el uso de Conntrackd es inútil, y por tanto la mitad de la configuración del cluster sobra.

Cabe destacar que desde un tiempo a esta parte, el proyecto Netfilter no se refiere a estados de conexiones con referencia a paquetes concretos, si no a flujos de conexión completos.

CONCLUSIONES

A modo de resumen y de consideraciones finales, pueden encontrarse algunas referencias de interés que un administrador interesado en el tema encontrará útiles a la hora de enfrentarse a casos en los que intervengan herramientas similares.

Valoración personal

Tengo que reconocer que el concepto de cluster siempre me ha llamado la atención. Las técnicas que implican que varias máquinas se pongan de acuerdo para funcionar como una sola, me resultan interesantes. Por otra parte, el diseño, creación y gestión de cortafuegos también me gusta bastante.

Combinando ambos aspectos, se entiende rápidamente el por qué de la temática de este Proyecto Integrado.

Introducirme en el mundo de la alta disponibilidad ha sido una tarea apasionante, sobre todo el acercamiento a Pacemaker.

Al principio he estado muy confundido con la cantidad de herramientas disponibles para la construcción de clusters de alta disponibilidad, y por la falta de documentación reciente para algunas de ellas.

Gracias a los canales de chat del IRC, las listas de correo, correos electrónicos personales a algunas personalidades implicadas con las herramientas aquí tratadas y extensas sesiones de lectura de documentación, he sido capaz de situarme dentro del panorama de la alta disponibilidad y entender por donde me movía.

Después de haber aprendido lo que considero básico para trabajar en alta disponibilidad me doy cuenta de que todo este mundillo me gusta todavía más de lo que pensaba y que lo me queda por aprender (mucho) me resulta muy atractivo como meta a medio/largo plazo.

Personalmente, he quedado fascinado con Pacemaker y la gestión de los recursos que hace. Verdaderamente, las posibilidades son inmensas y las opciones de configuración casi infinitas. Un acercamiento profundo y un trabajo "real" con la tecnología requiere conocimientos muy extensos y un buen entendimiento de todos los factores implicados en los recursos a desplegar en alta disponibilidad (colocación, arranques, restricciones y agrupaciones, etc..).

Durante el desarrollo de este proyecto, contaba con lo que creo que es un ventaja previa, y es que ya conocía la herramienta iptables, incluso en entornos en producción. Considero que estaba familiarizada con ella, lo que ha facilitado sobremanera todo el trabajo con Conntrackd y con el propio firewall. Evidentemente, en este aspecto, cuento con que aún me falta mucho por aprender y rendimiento que sacarle a las herramientas del proyecto Netfilter.

Me hubiera gustado desarrollar/implantar un **cluster activo/activo** con **balanceo de carga**, pero lo he encontrado fuera de mis posibilidades debido a la falta de documentación a tal respecto y a que en la realidad, un hardware de bajo coste puede filtrar grandes cantidades de tráfico ¿quizás por eso no hay un gran desarrollo de herramientas diseñadas para ello?

¿Keepalived o Corosync+Pacemaker?

A nivel de firewall y a nivel de cluster, pueden distinguirse sutiles diferencias en el uso de Keepalived o Corosync+Pacemaker.

Aunque en ambos casos el uso de una u otra herramienta es transparente para el cliente, se detectan una serie de ventajas e inconvenientes:

	Keepalived	Corosync+Pacemaker
Comunicación entre nodos.	Interfaces públicas de la red. Mucho "ruido" derivado del funcionamiento de VRRP. En cualquier caso, será necesaria una interfaz dedicada para Conntrackd.	Interfaz dedicada, privada. Tráfico interno y comunicado de forma segura.
Ancho de banda necesario.	Menor que Corosync + Pacemaker.	Mayor que Keepalived. Interfaz dedicada.
Ajustes de tiempos de sincronía, failover, etc..	Grandes posibilidad de ajuste, (ajuste de Keepalived y de Conntrackd)	Extremas posibilidades de ajuste (ajuste de Corosync, ajuste de Pacemaker y ajuste de Conntrackd).
Detección de failover (destrucción nodo máster)	~ 10 segundos. *	<= 2 segundos. *
Detección de failover (error de red)	~ 10 segundos. *	~ 15 segundos. *
Integración con más recursos en el cluster (otros servicios).	Muy complejo.	Total. Diseñado para ello.
Escalabilidad (más nodos).	Posible, aunque hay algunas pegas, como el "ruido" excesivo de las transmisiones multicast en las interfaces de red públicas.	Total.
Capacidad de trabajo (cantidad de recursos, por ejemplo IPVs)	Se advierte que con más de 200 IPVs, el rendimiento varía y hay que hacer ajustes concretos.	Sin límite conocido o publicado oficialmente.
Facilidad de montaje, administración y mantenimiento.	Muy fácil.	Mayor complejidad que con Keepalived, aunque también sencillo.
Integración con Conntrackd.	Muy buena. Los desarrolladores de Conntrackd recomiendan su uso.	En la versión de Pacemaker que ofrece Debian Squeeze **no** se incluye soporte para Conntrackd y hay que buscar el Resource Agent online.
Control global del cluster. Visualización de estado de recursos y manejo de los mismos.	Nulo.	Excelente.
Soporte para activo/activo con balanceo de carga.	Posible, sin necesidad de software externo. (Solo Keepalived+Conntrackd) Aunque es un balanceo de carga parcial.	Posible, con necesidad de balanceadores de carga externos (ej, LVS) en una estructura de red bastante compleja.
Valoración personal final.	En según qué circunstancias, opción válida.	**Muy recomendable.**

[*] Los test realizados no cubren, evidentemente, todas las posibilidades y circunstancias que pueden rodear a un cluster en implantación/producción en el momento del failover.

Problemas encontrados

A continuación se resumen los problemas más relevantes que se han encontrado durante el desarrollo de esta proyecto.

➔ **Pacemaker no arranca con Corosync.**

Durante las pruebas con Corosync+Pacemaker, me encuentro con que Pacemaker no arranca y que por tanto, no puedo seguir desarrollando el cluster. Esto es debido a que en la versión de Pacemaker con la que se ha trabajado, no se incluye un script LSB en "/etc/init.d/". Corosync es el encargado de levantar el servicio de Pacemaker, y esto lo hace mediante la correcta declaración en el fichero "/etc/corosync/service.d/pcmk", que se ha descrito en este documento.

➔ **Conntrackd no arranca. Pacemaker no es capaz de arrancarlo.**

Este problema está relacionado con un socket no abierto, debido a una IP e interfaz que no se ha especificado bien en el fichero de configuración de Conntrackd "/etc/conntrackd/conntrackd.conf".

➔ **En caso de failover, conexiones ESTABLISHED son asumidas como válidas sin el uso de Conntrackd en el nodo que pasa a ser primario**.

Debido a que no se han cargado todos los módulos necesarios al kernel ni se ha modificado el sysctl añadiendo "*net.netfilter.nf_conntrack_tcp_loose = 0*".

➔ **En caso de failover, los recursos no son repartidos en los nodos restantes.**

Esto es debido al problema del Quórum en clusters con dos nodos. Pacemaker entenderá que el cluster **no** tiene quórum cuando sólo existe un único nodo activo (momento de failback, y por tanto hay que especificar que se trabajará sin atender las mediciones de Quórum (ver sección de configuración correspondiente).

➔ **Keepalived no se pone de acuerdo en qué nodo toma los recursos.**

Esto es debido probablemente a que los paquetes de anuncio de recursos y de membresía a los grupos multicast no están llegando correctamente a cada nodo, o bien que están siendo filtrados por el firewall.

Como se ha descrito durante todo el documento, es muy importante ajustar el firewall para que permita las conexiones multicast que van a ser necesarias tanto por Keepalived, Corosync, Pacemaker y Conntrackd.

Terminología

Durante todo el documento se hace uso de cierta terminología específica. Aquí un breve resumen de la nomenclatura más relevante:

- Corosync -> Proporciona servicios de infraestructura de clustering (comunicación y pertenencia) a sus clientes. Esto permite a los clientes conocer la disponibilidad y los

procesos de los demás nodos del cluster.

- Keepalived -> Software que implementa el protocolo VRRPv2 y que haciendo uso del mismo, permite desplegar ciertos recursos en alta disponibilidad.
- OCF -> (Open Cluster Framework). Estándar internacional para el trabajo con recursos en clusters de alta disponibilidad. Mediante parámetros muy concretos, diferencia totalmente de un servicio parado de otro, por ejemplo, con un comportamiento indeterminado.
- CRM -> (Cluster Resource Manager). Gestiona recursos de alta disponibilidad que están desplegados en el cluster. Pacemaker es el CRM usado.
- CIB -> (Cluster Information Base). Fichero XML donde declaramos y definimos el cluster y los recursos repartidos o compartidos entre nodos. No se edita directamente, sino mediante otras herramientas como "cibadmin" o "crm".
- CCM -> (Cluster Consensus Membership). Provee de métodos para asegurar que cada nodo de un cluster puede comunicarse con los demás nodos del mismo cluster.
- VRRP -> (Virtual Router Redundancy Protocol). Protocolo de redundancia diseñado para aumentar la disponibilidad de un nodo de red mediante la creación, gestión y abstracción de una dirección de red virtual (IPV). Está basado en el RFC-3768.
- Failover -> Momento de fallo de un nodo de un cluster, provocado por cualquier circunstancia que impida el normal y correcto funcionamiento previamente planificado.
- Failback -> Momento de vuelta a la normalidad de un nodo de un cluster que previamente ha estado en modo "failover". Es muy importante analizar y prever el comportamiento del cluster en los momentos de failover y failback.

Referencias

Parte fundamental en el desarrollo de este proyecto, aquí se detallan algunas referencias usadas y que serán de gran interés para alguien con inquietud en los temas que se han tratado:

- Documentación oficial del proyecto *Pacemaker*.:
  http://www.clusterlabs.org/wiki/Example_configurations

  http://www.clusterlabs.org/doc/
- Documentación oficial de *Netfilter/Conntrack*:
  http://conntrack-tools.netfilter.org/manual.html

  http://conntrack-tools.netfilter.org/testcase.html
- "*Demystifying cluster-based fault-tolerant Firewalls*", de Pablo Neira, R.M. Gasca y L. Lefèvre, 2009:
  http://1984.lsi.us.es/~pablo/docs/intcomp09.pdf
- Documentación sobre iptables / firewalling
  http://www.faqs.org/docs/iptables/index.html
- Blog de Rhommel Lamas llamado "techironic.com":

http://techironic.com/index.php/2010/07/02/heartbeat-modo-crm-monitor/

- Foros de la web oficial del proyecto *LVS*, en "www.linuxvirtualserver.org":
  http://archive.linuxvirtualserver.org/html/lvs-users/2007-01/msg00152.html
- Foros oficiales de la empresa Gossamer Threads:
  http://www.gossamer-threads.com/lists/drbd/users/20972?do=post_view_threaded#20972
- Recursos varios en la web oficial del proyecto "*Linux High Avalilability*":
  http://hg.linux-ha.org/agents/file/tip/heartbeat/conntrackd
  http://linux-ha.org/wiki/Pacemaker
  http://linux-ha.org/doc/man-pages/re-ra-IPaddr.html
  http://www.linux-ha.org/wiki/Resource_Agents
- Documentación sobre la herramienta *iptables* de los profesores Jesús Moreno León y Alberto Molina Coballes para la asignatura "Redes de Área Local" en el IES Gonzalo Nazareno, año 2009-2010.

  http://ral-arturo.blogspot.com/p/documentacion.html
- Building Debian packages:
  http://www.ibm.com/developerworks/linux/library/l-debpkg/index.html#resources
- Wikipedia, artículos varios:
  http://es.wikipedia.org/
- Listas de correo de Clusterlabs, LVS y Netfilter.

  Clusterlabs: http://oss.clusterlabs.org/pipermail/pacemaker/

  LVS: http://lists.graemef.net/pipermail/lvs-users/

  Netfilter: http://marc.info/?l=netfilter
- Wackamole (HA manager), actualizado por última vez en 2007.
  http://www.backhand.org/wackamole/
- Página web oficial de Keepalived (implementación de VRRP2)
  http://www.keepalived.org/index.html
- Documentación oficial del proyecto Corosync/OpenAIS

  http://www.corosync.org/doku.php?id=faq:configure_openai
- Revista “Linux Journal”, Mayo de 2011, Issue 205, "Build a better Firewall" de Mike Horn.