PROYECTO FINAL DE MÁSTER

Implantación de sistema VoIP
en alta disponibilidad y distribución de carga

Autor: José M. Callejón Sánchez
Tutor: Miguel Martín Mateo

Màster en Programari Lliure
Universitat Oberta de Catalunya
Curso 2011-2012

Índex de continguts

1 RESUMEN

El desarrollo de este proyecto como “Proyecto Final de Máster” nace de una necesidad real, la migración de la infraestructura de telefonía actual de un organismo público (en concreto un ayuntamiento) a una solución de telefonía IP basada en software libre.

Fruto de esta necesidad se planteo un reto. El reto de construir un sistema que fuera fiable y eficiente, y que para ello garantizara aspectos como la disponibilidad, el alto rendimiento, o la escalabilidad.

La solución que se describe en este proyecto se sostenta básicamente en cuatro pilares:

- **SIP**: Un protocolo de señalización de sesiones interactivas.
- **Asterisk**: Software que proporciona funcionalidades de centralita telefónica
- **Kamailio**: Proxy SIP capaz de manejar cientos de llamadas por segundo.
- **Pacemaker**: Software para la gestión de servicios clúster.

Con el desarrollo de este proyecto se pretenden conseguir los siguientes objetivos:

Por un lado y desde el punto de vista del cliente, ofrecer la posibilidad de migrar de la actual infraestructura de telefonía totalmente desfasada por una innovadora solución capaz de proporcionar infinidad de posibilidades que hasta ahora eran impensables se pudieran relacionar con la telefonía, consiguiendo además una notable reducción de costes a medio y largo plazo gracias principalmente a tres factores:

- Ahorro en licencias por la utilización de software libre.
- La reutilización de la infraestructura de red actual y la conectividad entre sus diversas sedes para el servicio de voz.
- La substitución de las actuales lineas de telefonía RTB,RDSI y E1 por canales SIP.

Por otro lado, y ya a titulo personal, la principal motivación al desarrollar este proyecto como PFM es la de poner en práctica los conocimientos adquiridos durante la realización de este Máster y conseguir profundizar en el conocimiento de un tema de tanta trascendencia actual como es la VoIP.

Por último, y aunque no menos destacable, con el desarrollo de este proyecto y la posible posterior implantación de este sistema, basado íntegramente en software libre, se pretende contribuir al fomento de este movimiento especialmente en entes públicos en los que el uso de software libre y estándares abiertos debería ser algo común pero en los que lamentablemente aún no lo es.

2 ESTADO DEL ARTE

El ayuntamiento de cierta localidad ha renovado recientemente su infraestructura de red, logrando interconectar todas sus sedes.
Las sedes más próximas al edificio central han sido interconectadas gracias a enlaces redundados de fibra óptica.
El resto de sedes, más distantes al ayuntamiento, han logrado ser conectadas con dicho edificio a través de enlaces wifi en la frecuencia de los 5Ghz obteniendo velocidades de transferencia similares a las de una LAN Ethernet.

Con objeto de rentabilizar los costes de dicha inversión y aprovechar la infraestructura recientemente renovada y después de oír hablar de las ventajas que presentan la VoIP y la utilización de software libre, el departamento de TI del ayuntamiento, tras un estudio de viabilidad técnico y económico, ha decidido migrar su actual sistema de telefonía ha esta tecnología eliminando todas las centralitas de sus sedes y centralizando los servicios en el edificio principal.

Actualmente el ayuntamiento consta de unas 20 sedes repartidas por todo el municipio, se calcula que entre estas sedes hay cerca de 500 extensiones además de una treintena de faxes repartidos entre un total de 15 centralitas.

El grueso de llamadas suele producirse entre las 09:00 y las 12:00, y se estima que ocasionalmente podrían llegarse a producir entorno a 150 llamadas de forma simultanea entre llamadas internas y externas.

En función del tamaño de cada una de estas sedes son diversas las lineas de telefonía que las centralitas actuales tienen conectadas, desde enlaces primario tanto de telefonía fija como móvil, pasando por RDSI hasta lineas analógicas convencionales y enlaces GSM.

Uno de los objetivos principales por parte del ayuntamiento a la hora de implantar el sistema VoIP es conseguir reducir notablemente los gastos correspondientes a la facturación de telefonía, por ello se propone eliminar las centralitas actuales de todas sus sedes y unificar el servicio de voz en una único sistema.

El ayuntamiento dispone además en su edificio central de una importante infraestructura virtual sobre la que que desearía poder integrar el sistema de telefonía en la medida de lo posible.

El ayuntamiento cuenta además con una conexión redundada de fibra óptica de 100Mbps simétricos con su ISP.

3 DISEÑO DEL SISTEMA

A la hora de determinar que sistema es el más adecuado cuando se trata de implementar una solución VoIP son varios los factores que hay tener en cuenta, entre estos podríamos destacar los siguientes:

- Número de usuarios o extensiones
- Número de llamadas concurrentes esperadas
- Transcodificaciones que serán necesarias
- Protocolos empleados
- Funcionalidades avanzadas deseadas
- etc ...

Basándonos principalmente en los requisitos detallados anteriormente, y teniendo en cuenta el resto, recogidos durante la fase inicial de este proyecto, la solución que a continuación se presenta, se estima como la más adecuada con el fin de garantizar un servicio fiable y de calidad.

La siguiente imagen define a grandes rasgos el diseño de la solución propuesta, que posteriormente sera justificado.

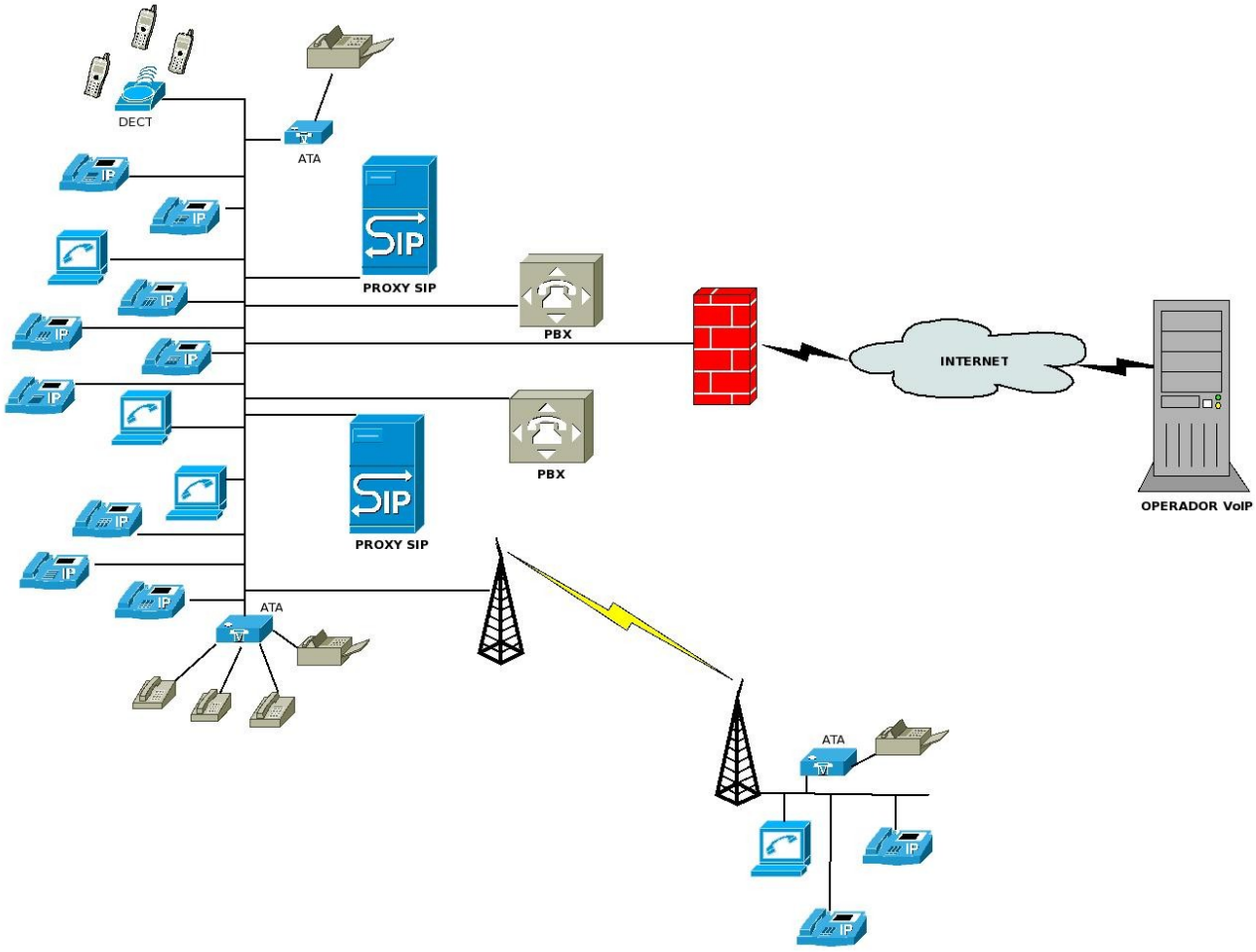


Figura 3.1 Solución diseñada

Para entender el funcionamiento del sistema que se ha desarrollado en este proyecto es necesario conocer previamente los “miembros” que componen generalmente toda arquitectura VoIP, a continuación los definiremos brevemente.

- **Teléfonos IP:** Equipos similares a los tradicionales teléfonos pero adaptados para ser utilizados en entornos IP (Wired o Wireless). Encargados de codificar y decodificar señales de audio y vídeo Son ya considerados pequeños ordenadores (poseen procesador,memoria,sistema operativo...) Ofrecen gran variedad de funcionalidades avanzadas.

- **Softphone**: Software que permite emular el comportamiento de los teléfonos IP ejecutándose directamente desde un ordenador sin necesidad de hardware.

- **Adaptadores ATA**: Dispositivos que permiten conectar teléfonos y faxes convencionales a una infraestructura IP transformando la señal.

- **SIP**: Session Initiation Protocol, protocolo de señalización para VoIP de sintaxis similar al HTTP permite mediante el envío de mensajes, el inicio, la modificación o la finalización de sesiones interactivas, como la voz. Existen otros protocolos como IAX2 (utilizado por Asterisk y desarrollado por su mismo creador) o H.323. Este proyecto se basa íntegramente en SIP.

- **RTP**: Real Time Protocol, protocolo utilizado para la transmisión de audio y vídeo en tiempo real. Utilizado frecuentemente junto a SIP.

- **Usuarios SIP**: Cualquier dispositivo (Teléfono IP, ATA) o software (Softphone) compatible con SIP. Capaz de registrarse en un Servidor SIP.
- **Servidor SIP**: Software o dispositivo que permite crear y gestionar cuentas SIP permitiendo el registro de usuario, guardando la información referente a estos para localizarlo en caso de producirse alguna petición.
- **Proxy SIP**: Software que actúa como intermediario entre dos usuarios SIP cuando estos quieren establecer una conversación poniéndolos en contacto. Para establecer una conversación entre dos usuarios SIP son necesarios, un Servidor SIP donde registrar los usuarios y un Proxy SIP que ponga en contacto a los dos usuarios. Posteriormente el intercambio de audio/vídeo (a través de RTP p.e) se realizará directamente entre los dos usuarios sin necesidad de que el resto de elementos intervengan. Un proxy SIP únicamente gestiona el estado de una llamada cuando se realiza.
- **B2BUA**: Back 2 Back User Agent, aplicación para la gestión de llamadas entre usuarios SIP que a diferencia de un Proxy SIP mantiene el estado de las llamadas, lo cual puede ser útil en aspectos como para el control de la facturación. Asterisk actúa como B2BUA pero ofreciendo funcionalidades adicionales como veremos posteriormente.

Tras analizar los requisitos del proyecto y especialmente los aspectos que conciernen al número total de extensiones y de llamadas concurrentes, junto con la posibilidad de utilizar la infraestructura virtual del cliente, desde un principio se considero que la carga de trabajo estimada podría ser excesiva para ser soportada por un único equipo, y que lógicamente este debía de estar redundado puesto que un fallo de este podía comportar el paro completo del servicio de telefonía de toda la organización.

La solución finalmente propuesta consta de los siguientes elementos principales:

- 2 Servidores[1] virtuales Proxy SIP en alta disponibilidad formando un clúster activo-pasivo actuarán de intermediarios reenviando las peticiones SIP.
  Proporcionaran también balanceo de carga distribuyendo las llamadas uniformemente entre los 2 servidores Asterisk. Paralelamente ofrecerán servicios adicionales como DHCP,TFTP,NTP.
  Características generales de cada equipo:
  - 2 vCPU
  - 2GB RAM
  - 4vNIC
  - 10GB HDD

- 2 Servidores virtuales Asterisk en alta disponibilidad (activo-activo). Actuarán como B2BUA ofreciendo funcionalidades avanzadas. Como IVR, Ringgrup, Colas, Voicemail...
  La conexión con el exterior se realizará a través de 2 “trunks” SIP con un operador de VoIP de Internet.
  Características generales de cada equipo:
  - 2 vCPU
  - 2GB RAM
  - 4vNIC[2]
  - 150GB HDD[3]

El ayuntamiento ha proporcionado 2 vLANs (para la gestión de la VoIP y la gestión del Clúster) con las siguientes características:

- VLAN ID=99
  NOMBRE=VoIP
  DIRECCIONAMIENTO IP=192.168.0.0/22

- VLAN ID=100
  NOMBRE=CLUSTER
  DIRECCIONAMIENTO IP=172.16.1.0/24

[1]*Se considera que los sistemas correrán sobre diferentes hosts físicos*
[2]*Se asignan 2 tarjetas de red a cada vlan formando un bounding*
[3]*La reserva de 150GB de disco se justifica frente a la posibilidad de realizar grabaciones eventuales de llamadas.*

El proyecto se ha centrado en la implementación de un sistema distribuido fiable y eficiente que garantizara aspectos como la alta disponibilidad, la escalabilidad, o la distribución de carga, lejos de profundizar en otros aspectos más específicos relacionados con las diversas soluciones empleadas.
El sistema que a continuación se presenta se basa principalmente en las siguientes soluciones:

- **Debian 6.0.0.4 Squeeze:** Como sistema operativo.
- **Asterisk 1.8.11.0**: Como PBX SERVER.
- **Kamailio 3.1.5**: Como PROXY SIP SERVER.
- **Pacemaker 1.0.9.1 + Corosync 1.2.1-4**: Como gestor del clúster.
- **DRBD 8.3.2**: Como sistema de replicación de datos en red.
- **MySQL 5.1.61**: Como sistema gestor de bases de datos.

La figura 3.2 muestra el diseño definitivo de la solución propuesta:

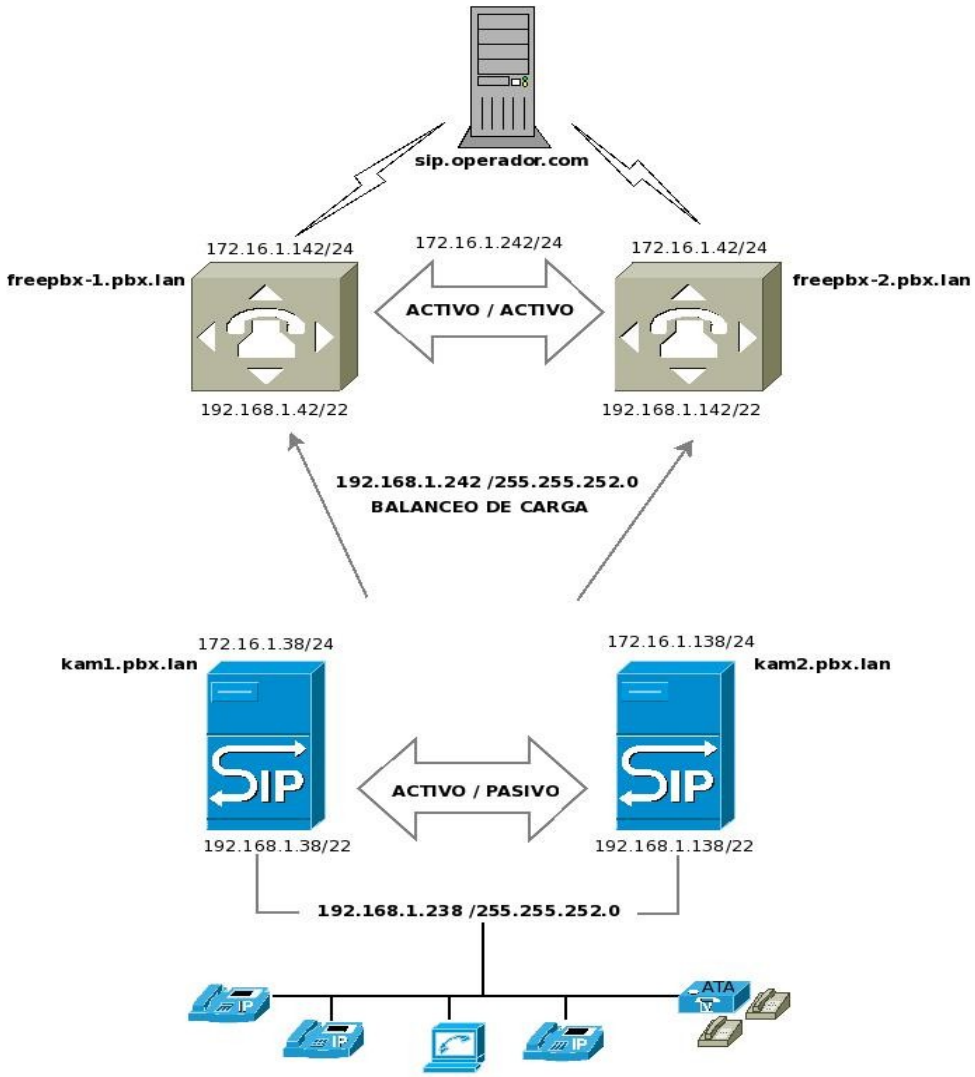


Figura 3.2 Diseño final

4 IMPLEMENTACIÓN DEL SISTEMA

4.1 ASTERISK

Asterisk es un aplicativo open source que proporciona funcionalidades de PBX avanzada. Licenciado bajo licencia dual, GNU/GPL y propietaria, Asterisk empezó a desarrollarse el año 1999 de la mano de su creador Mark Spencer y fundador de Diguim (principal empresa desarrolladora de Asterisk) quién tras intentar adquirir una solución privada decidió programar su propia centralita por el elevado precio de estas.

Asterisk presenta una arquitectura modular que permite activar y desactivar módulos específicos en función de la necesidad. Estos módulos se dividen en siete categorías:

- Core: Núcleo de Asterisk que posibilita la carga del resto de módulos
- Recursos: Aportan diversas funcionalidades a Asterisk
- Canales: Permiten a Asterisk interactuar con dispositivos de diversas tecnologías (SIP, IAX, DAHDI)
- Aplicaciones y funciones: Proporcionan herramientas para configurar el sistema.
- CDR: Control del registro de llamadas (ver 3.1)
- Codecs: Gracias a estos Asterisk puede actuar como codificador y descodificador de audio y vídeo.
- Formatos: Posibilitan a Asterisk la gestión de ficheros de diversos formatos (wav,mp3,ulaw...)

Asterisk es capaz de trabajar con prácticamente todos los estándares de telefonía tradicional (lineas analógicas , digitales: E1/T1) además de soportar la mayoría de protocolos VoIP(SIP,IAX2 ,MGCP ,Cisco Skinny).

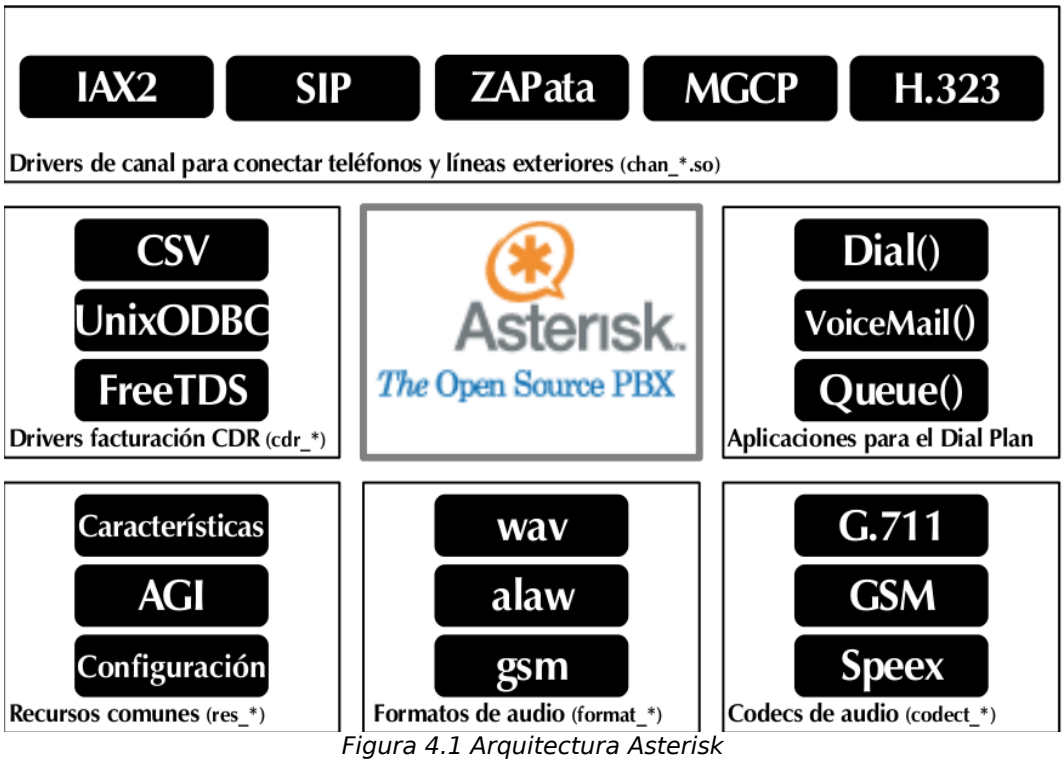


Figura 4.1 Arquitectura Asterisk

Asterisk proporciona funcionalidades básicas y avanzadas frente a otras soluciones propietarias, entre estas:

Funcionalidades Básicas	Funcionalidades Avanzada
Llamada en espera Transferencia Identificado de llamada Bloqueo de Caller ID Timbres distintivos Música en espera Música en transferencia Salas de Conferencia Buzón de Voz personal Colas de llamada Colas con prioridad Registro de llamadas en BD Buzón de Voz por Mail Pickup de llamadas Desvío si ocupado Desvío si no responde	IVR: Interactive Voice Response (Gestión de llamadas a través menús interactivos.) LCR: Least Cost Routing, encaminamiento de llamadas por el proveedor más económico. AGI: Asterisk Gateway Interface (integración con todo tipo de aplicaciones externas.) AMI: Asterisk Management Interface, gestión y control remoto de Asterisk. Configuración en BD: usuarios, extensiones ...

Asterisk puede ser instalado sobre sistemas GNU/Linux, BSD y MacOSX. Está disponible en los repositorios de las distribuciones más comunes, sin embargo su instalación suele realizarse a partir del código fuente.
La configuración de Asterisk suele realizarse a través de ficheros de texto.

4.1.1 Instalación de Asterisk

La versión de Asterisk escogida en este proyecto es 1.8.11.0 la última versión estable al iniciar el proyecto.
El proceso de instalación que se describe a continuación afecta solo a uno de los nodos, el proceso que se debe seguir es idéntico para el segundo de los servidores PBX descrito en el esquema inicial.

Existe una versión disponible de Asterisk en los repositorios oficiales de Debian, sin embargo es preferible realizar la instalación desde el propio código fuente, de esta manera podremos compilar aquellos módulos que necesitemos.

En primer lugar descargaremos el software necesario:

```
root@freepbx-1:~# mkdir /usr/src/voip

root@freepbx-1:~# cd  /usr/src/voip

root@freepbx-1:/usr/src/voip# wget  http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-1.8.11.0.tar.gz

root@freepbx-1:/usr/src/voip# wget  http://downloads.asterisk.org/pub/telephony/dahdi-linux-complete/releases/dahdi-linux-complete-2.6.0+2.6.0.tar.gz
```

DAHDI (Digium Asterisk Hardware Device Interface) proporciona los módulos necesarios para poder utilizar tarjetas de comunicaciones (ISDN,TDM).

Pese a que en este proyecto no se va a utilizar ningún de estas tarjetas al tratarse de un esquema completamente SIP, la instalación de DAHDI es recomendable ya que este es necesario para poder hacer uso de algunas aplicaciones como MeetMe().

Adicionalmente, si se requiriera conectarse a una linea de primario (E1/T1) sería necesario instalar la librería LibPRI, disponible en:
http://downloads.asterisk.org/pub/telephony/libpri/releases/libpri-1.4.12.tar.gz

La instalación de Asterisk requiere de ciertas dependencias que deberían instalarse previamente:

```
root@freepbx-1:~# apt-get install build-essential libncurses5-dev libssl-dev libxml2-dev linux-headers-`uname -r`
```

Una vez instaladas las dependencias necesarias podemos empezar la instalación de en primer lugar de DAHDI:

```
root@freepbx-1:/usr/src/voip# tar -zxvf dahdi-linux-complete-2.6.0+2.6.0 .tar.gz

root@freepbx-1:/usr/src/voip# cd dahdi-linux-complete-2.6.0+2.6.0

root@freepbx-1:usr/src/voip/dahdi-linux-complete-2.6.0+2.6.0# make

root@freepbx-1:usr/src/voip/dahdi-linux-complete-2.6.0+2.6.0# make install

root@freepbx-1:usr/src/voip/dahdi-linux-complete-2.6.0+2.6.0# make config
```

Finalizada la instalación de DAHDI, el siguiente paso es el de compilar Asterisk:

```
root@freepbx-1:/usr/src/voip# tar -zxvf asterisk-1.8.11.0.tar.gz

root@freepbx-1:/usr/src/voip# cd asterisk-1.8.11.0

root@freepbx-1:usr/src/voip/asterisk-1.8.11.0# ./configure

root@freepbx-1:usr/src/voip/asterisk-1.8.11.0# make

root@freepbx-1:usr/src/voip/asterisk-1.8.11.0# make install

root@freepbx-1:usr/src/voip/asterisk-1.8.11.0# make config

root@freepbx-1:usr/src/voip/asterisk-1.8.11.0# make samples
```

Mediante la orden “make samples” se generan archivos de configuración de ejemplo.

Con el fin de hacer más seguro el sistema el servicio de Asterisk será ejecutado por el un nuevo usuario:

```
root@freepbx-1:/usr/src/voip# groupadd asterisk

root@freepbx-1:/usr/src/voip# useradd -g asterisk -d /var/lib/asterisk -s /bin/bash asterisk

root@freepbx-1:/usr/src/voip# chown -R asterisk: /usr/lib/asterisk
```

```
root@freepbx-1:/usr/src/voip# chown -R asterisk: /var/lib/asterisk

root@freepbx-1:/usr/src/voip# chown -R asterisk: /var/spool/asterisk

root@freepbx-1:/usr/src/voip# chown -R asterisk: /var/log/asterisk

root@freepbx-1:/usr/src/voip# chown -R asterisk: /var/run/asterisk

root@freepbx-1:/usr/src/voip# chown -R asterisk: /usr/sbin/asterisk
```

Para que DAHDI también se ejecute con este nuevo usuario deberemos indicarlo en el fichero /etc/udev/rules.d/dahdi.rules

```
root@freepbx-1:/usr/src/voip# tail -n 2 /etc/udev/rules.d/dahdi.rules
# DAHDI devices with ownership/permissions for running as non-root
SUBSYSTEM=="dahdi",  OWNER="asterisk", GROUP="asterisk", MODE="0660"
```

Con estos pasos el sistema ya estaría completamente instalado pero aún no sería operativo ya que faltaría configurar aspectos tan importantes como la definición de canales (SIP en nuestro caso) o el Dialplan (que hacer con las llamadas).

Llegados a este punto, se plantearon dos alternativas, la primera era llevar una configuración manual de los ficheros que Asterisk utiliza para la configuración de sus módulos, lo cual presentaba la ventaja de tener un sistema completamente seguro y gestionado exclusivamente por los administradores del sistemam, pero por otro lado presentaba la desventaja de que el personal del Ayuntamiento tuviera que recurrir a la empresa encargada del mantenimiento para modificar aspectos tan simples como el nombre de una extensión.
Ante esta tesitura se decidió por instalar una interficie GUI que permitiera administrar con facilidad los aspectos más simples de la centralita, recurriendo a la configuración costumizada solo en aquellas situaciones en las que fuera necesario.

4.2 FreePBX

FreePBX es una interficie GUI HTML amigable que permite interactuar con Asterisk, ofreciendo simplicidad a la hora de configurar la PBX.
FreePBX está desarrollado por una comunidad de voluntarios como tantos otros proyectos open source, y se distribuye bajo licencia GNU/GPL.
FreePBX debe ser ejecutado sobre un entorno LAMP (Linux, Apache, MySQL, PHP).
Entre otras, ofrece facilidades para configurar las siguientes funcionalidades:

- Gestión de extensiones
- Gestión de trunks SIP, IAX, DAHDI
- Gestión de enrutamiento de llamadas (entrantes y salientes)
- Operadora Virtual (IVR)
- Ring-Groups
- Colas
- Horarios
- Voicemail
- Informes de llamadas
- etc...

4.2.1 Instalación de FreePBX

Como se ha comentado la instalación de FreePBX requiere de un entorno LAMP, con lo que en primer lugar será necesario preparar este entorno:

```
root@freepbx-1:/usr/src/voip# apt-get  install apache2 php5 mysql-server  php5-cli php-pear php-db php5-mysql libmysqlclient15-dev sox mpg123
```

Es necesario realizar algunas modificaciones en el servidor Apache:

```
root@freepbx-1:/usr/src/voip# perl -pi -e 's/upload_max_filesize = 2M/upload_max_filesize = 40M/g'

root@freepbx-1:/usr/src/voip# /etc/php5/apache2/php.ini

root@freepbx-1:/usr/src/voip#  perl -pi -e 's/max_execution_time = 30/max_execution_time = 120/g' /etc/php5/apache2/php.ini

root@freepbx-1:/usr/src/voip# perl -pi -e 's/max_input_time = 60/max_input_time = 120/g' /etc/php5/apache2/php.ini

root@freepbx-1:/usr/src/voip# perl -pi -e 's/magic_quotes_gpc = On/magic_quotes_gpc = Off/g' /etc/php5/apache2/php.ini

root@freepbx-1:/usr/src/voip# perl -pi -e 's/www-data:x:33:/www-data:x:33:asterisk/g' /etc/group

root@freepbx-1:/usr/src/voip# perl -pi -e 's/User \${APACHE_RUN_USER}/User asterisk/g' /etc/apache2/apache2.conf

root@freepbx-1:/usr/src/voip# perl -pi -e 's/Group \${APACHE_RUN_GROUP}/Group asterisk/g' /etc/apache2/apache2.conf
```

Descargamos y descomprimimos FreePBX desde la página oficial:

```
root@freepbx-1:/usr/src/voip# wget http://mirror.freepbx.org/freepbx-2.9.0.tar.gz

root@freepbx-1:/usr/src/voip# tar -zxvf freepbx-2.9.0.tar.gz

root@freepbx-1:/usr/src/voip# cd  freepbx-2.9.0
```

En primer lugar será necesario generar las BBDD que FreePBX utilizará para almacenar las configuraciones:

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql --defaults-file=/etc/mysql/debian.cnf -e "CREATE DATABASE asteriskcdrdb;"

root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql --defaults-file=/etc/mysql/debian.cnf asteriskcdrdb < SQL/cdr_mysql_table.sql

root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql --defaults-file=/etc/mysql/debian.cnf -e "CREATE DATABASE asterisk;"
```

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql --defaults-file=/etc/mysql/debian.cnf asterisk < SQL/newinstall.sql
```

Asignamos los permisos necesarios sobre las BBDD:

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql -uroot -p -e "GRANT ALL PRIVILEGES ON asterisk.* TO asterisk@localhost IDENTIFIED BY 'asterisk10'; FLUSH PRIVILEGES;"

root@freepbx-1:/usr/src/voip/freepbx-2.9.0# mysql -uroot -p -e "GRANT ALL PRIVILEGES ON asteriskcdrdb.* TO asterisk@localhost IDENTIFIED BY 'asterisk10'; FLUSH PRIVILEGES;"
```

Para instalar FreePBX, la propia aplicación ofrece un script de instalación, el único requisito antes de iniciar la instalación es que asterisk este corriendo:

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# asterisk start

root@freepbx-1:/usr/src/voip/freepbx-2.9.0# ./install_amp --username=asterisk --password=asterisk10
```

Finalizada la instalación copiaremos el script de inicio en el directorio /etc/init.d y actualizaremos los runleves necesarios para que el servicio arranque y se detenga de forma ordenada.

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# cp start_asterisk /etc/init.d/amportal
```

Para que el script cumpla los requisitos LSB (Linux Standrards Base) previamente es necesario añadir la siguiente información al inicio del script.

```
root@freepbx-2:/usr/src/voip# head /etc/init.d/amportal
#!/usr/bin/env bash
### BEGIN INIT INFO
# Provides:        amportal
# Required-Start:  $local_fs $remote_fs dahdi
# Required-Stop:   $local_fs $remote_fs dahdi
# Should-Start:    $network $syslog
# Should-Stop:     $network $syslog
# Default-Start:   2 3 4 5
# Default-Stop:    0 1 6
# Description:     amportal by jmc
### END INIT INFO
```

Finalmente podremos añadir el script a los runlevels e inciar el servicio:

```
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# insserv amportal
root@freepbx-1:/usr/src/voip/freepbx-2.9.0# /etc/init.d/amportal start
```

Accediendo mediante el navegador a la dirección configurada podremos visualizar la página principal de FreePBX.

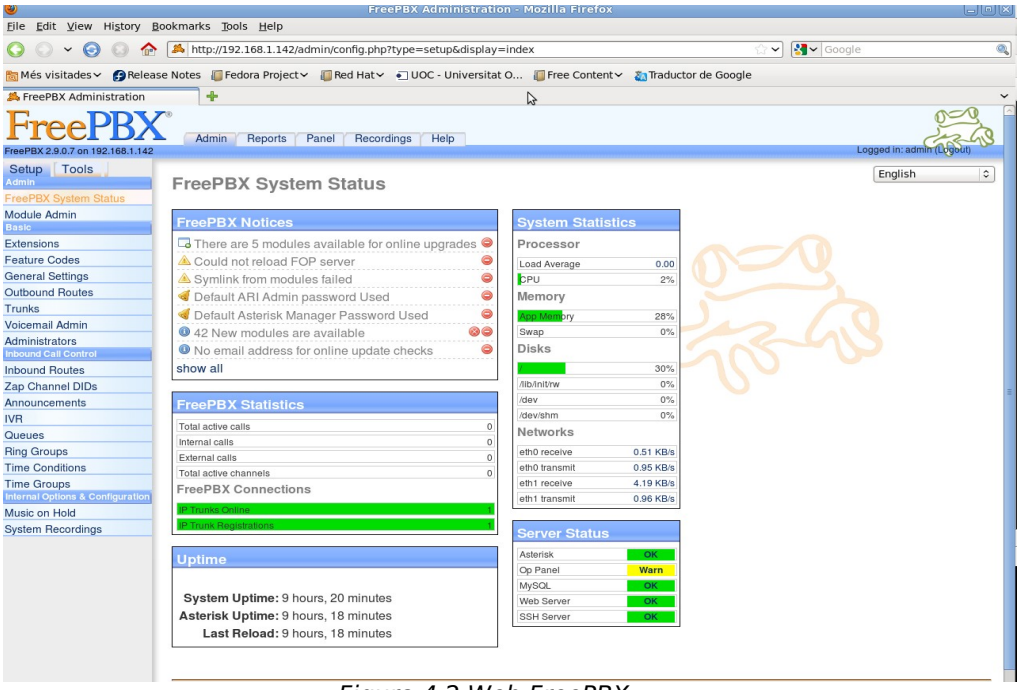


Figura 4.2 Web FreePBX

4.3 Kamailio

Kamailio también conocido como SIP Router y anteriormente al año 2008 como OpenSER, es un un servidor proxy SIP open source, licenciado bajo GNU/GPL.
Se caracteriza por su capacidad de gestionar cientos de llamadas por segundo.
Kamailio puede ser utilizado para construir grandes plataformas de servicios de VoIP o para ampliar pasarelas SIP o media servers como Asterisk o FreeSWITCH .
La aplicación está escrita en C para plataformas Linux/UNIX y se centra en el rendimiento, la flexibilidad y la seguridad. Además de C, en algunas funcionalidades también se emplean lenguajes como Lua, Perl o Python.

Al igual que Asterisk Kamailio tiene una arquitectura modular que permite adaptar su configuración en función de las características que se necesiten de el.
Algunas de las características que proporcionan los alrededor de 150 módulos disponibles son las siguientes:

- Diversos protocolos: TCP, UDP y SCTP
- Comunicaciones segura a través de TLS
- IPV4 y IPV6
- SIMPLE (Session Inititation Protocol for Instant Messaging and Presence Leveraging Extensions)
- ENUM (mapeo de número telefónico)
- Enrutamiento de menor costo
- Balanceo de carga,
- Fail-over
- Autenticación y autorización a través de MySQL, PostgreSQL, Oracle, RADIUS, LDAP
- Monitorización SNMP

La Figura 4.3.2 describe la arquitectura de kamailio dividida en 2 categorias

principales, Core que proporciona las funcionalidades de más bajo nivel y los módulos, componentes que proporcionan el resto de funcionalidades.

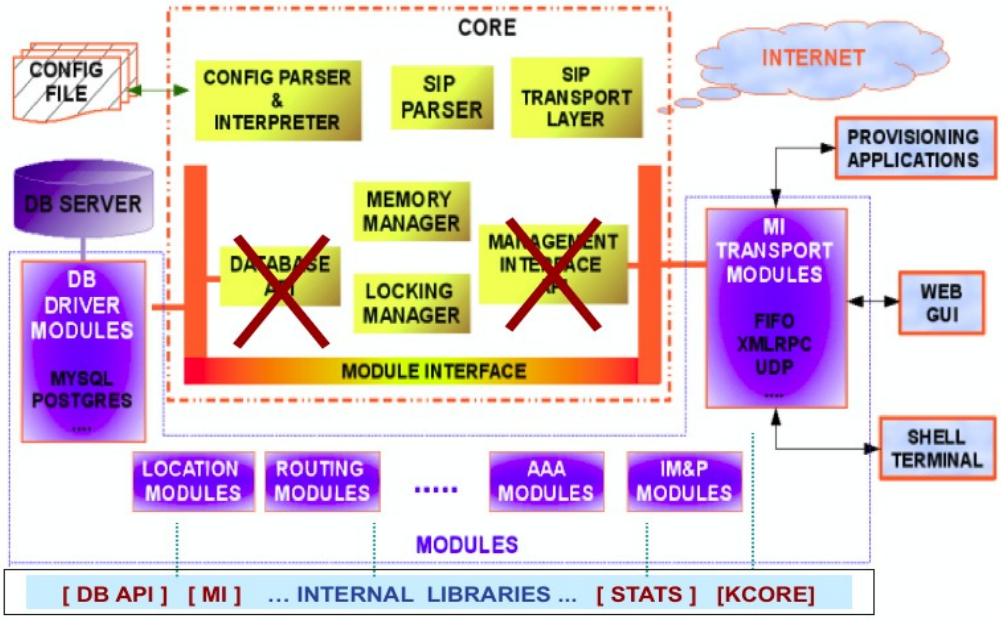


Figura 4.3 Arquitectura Kamailio 3.X

4.3.1 Instalación de Kamailio

La versión de Kamailio utilizada en este proyecto es la 3.1.5. La instalación puede realizarse tanto a través de GIT como desde el propio código fuente.
En este caso se ha realizado a través del código fuente con lo que en primer lugar será necesario descargar y descomprimir el páquete que contiene dicho código.

```
root@kam1:~# cd /usr/src/
root@kam1:/usr/src# wget http://www.kamailio.org/pub/kamailio/3.1.5/src/kamailio-3.1.5_src.tar.gz
root@kam1:/usr/src# tar -zxvf kamailio-3.1.5_src.tar.gz
root@kam1:/usr/src# cd kamailio-3.1.5
```

Una vez descomprimido si accedemos al nuevo directorio y observamos el fichero INSTALL veremos que son necesarias algunas dependencias en función de los módulos que vayan a utilizarse. En este caso las dependencias necesarias son las siguientes:

```
root@kam1:/usr/src/kamailio-3.1.5# apt-get install build-essential flex bison libmysqlclient15-dev
```

Instaladas las dependencias podemos proceder a compilar e instalar Kamailio. El módulo de MySQL será necesario para poder realizar la integración con Asterisk.

```
root@kam1:/usr/src/kamailio-3.1.5# make FLAVOUR=kamailio  include_modules="db_mysql dialplan" \
cfg
root@kam1:/usr/src/kamailio-3.1.5# make all
root@kam1:/usr/src/kamailio-3.1.5# make install
```

La instalación nos crea los siguientes ejecutables en /usr/local/sbin*:

- *kamailio* - Binario de Kamailio

- *kamdbctl* - Script para crear y gestionar bases de datos.
- *kamctl* - Script para la gestión y el control de Kamailio
- *sercmd* - Linea de comandos para interactuar con Kamailio

Los módulos requeridos por Kamailio son instalados en*:

- */usr/local/lib/kamailio/modules/* - Módulos generales
- */usr/local/lib/kamailio/modules_k/* - Módulos específicos de Kamailio
- */usr/local/lib/kamailio/modules_s/* - Módulos específicos de SER

**En sistemas de 64bits se econtran en /usr/loca/lib64/...*

El directorio de configuración de Kamailio se encuentra en */usr/local/etc/kamailio/* .

Alternativamente podemos modificar estos directorios indicando en la compilación y la instalación el directorio raíz:

```
root@kam1:/usr/src/kamailio-3.1.5# make prefix=/ all
root@kam1:/usr/src/kamailio-3.1.5# make prefix=/ install
```

El siguiente paso consiste en crear la base de datos. Pese a que la autenticación de usuarios se realizará a través de la integración con Asterisk, la creación de esta BD es necesaria para almacenar los registros correspondientes a la localización de los UAC. p.ej:

```
mysql> select contact,user_agent from location limit 2;
+-------------------------------------------------------------+--------------------------------------+
| contact                                                     | user_agent                           |
+-------------------------------------------------------------+--------------------------------------+
| sip:103@192.168.1.50:5060;line=e29bf3037d2d69f | Linphone/3.2.1 (eXosip2/3.1.0) |
| sip:102@192.168.1.54:5060                                   | Linksys/SPA922-5.2.8                 |
+-------------------------------------------------------------+--------------------------------------+
2 rows in set (0.00 sec)
```

La instalación de la BD puede realizarse con uno de los scripts proporcionados (kamdbctl), antes de ejecutarlo es necesario indicar en el fichero kamctlrc que BD se va a utilizar:

```
root@kam1:~# cat /usr/local/etc/kamailio/kamctlrc | grep -v "^#"
DBENGINE=MYSQL
root@kam1:~# kamdbctl create
```

4.3.2 Integración Asterisk-Kamailio

Como se comentara al principio, el sistema que se define en este proyecto utiliza dos elementos principales . Por un lado Kamailio encargado del registro y la gestión de las llamadas SIP, y por otro Asterisk que realizará las funciones de media-server (con aplicaciones como voicemail, IVR ...) y gateway (conectando con el exterior a través de trunks SIP).

Asterisk Realtime Architecture (ARA) es una funcionalidad de Asterisk que permite

almacenar configuraciones (que normalmente estarían en /etc/asterisk/) en bases de datos.

Existen dos métodos para implementar esta arquitectura en tiempo real, estático y dinámico. El método estático es similar al método tradicional de lectura de ficheros de configuración, excepto porque los datos son leídos desde una base de datos. En el caso del método dinámico la información se carga y se actualiza en función de las necesidades. Este método es utilizado por ejemplo para definir usuarios SIP o buzones de voz.
Mientras que al aplicar un cambio con el método estático requiere hacer una recarga del módulo correspondiente, como si se hiciera el cambio en fichero de texto del sistema, en el método dinámico estos cambios se aplican de forma automática.
El fichero de configuración de la funcionalidad “Realtime” es **extconfig.conf**, generalmente dentro del directorio /etc/asterisk/. Este archivo indica a Asterisk que datos deben cargarse desde la base de datos y a que corresponden. Este método es totalmente compatible con la configuración estándar

El proceso de integración de ambos componentes que a continuación se describe se realiza a través de MySQL.
Para que Asterisk pueda comunicarse con MySQL una de las formas posibles de realizarlo es a través de la librería ODBC de Unix y el conector para MySQL así que en primer lugar será necesario instalar dichas dependencias.

A modo recordatorio decir que hasta ahora Asterisk no estaba vinculado de ninguna manera con MySQL sino que era FreePBX quien lo hacía para guardar las configuraciones que luego este mismo presentaba en formato de fichero de texto a Asterisk.
**El proceso que a continuación se describe debe realizarse también sobre los nodos secundarios*

```
root@freepbx-1:~# apt-get install unixodbc-dev libmyodbc
```

Instaladas las dependencias el siguiente paso es crear la BBDD donde se almacenarán los registros que Kamailio y las correspondientes tablas. Para ello generaremos un script SQL de nombre p.ej asterisktmp.sql con el siguiente contenido:

```
CREATE DATABASE `asterisktmp` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `asterisktmp`;

CREATE TABLE IF NOT EXISTS `sipregs` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(80) NOT NULL DEFAULT '',
  `fullcontact` varchar(80) NOT NULL DEFAULT '',
  `ipaddr` varchar(15) NOT NULL DEFAULT '',
  `port` mediumint(5) unsigned NOT NULL DEFAULT '0',
  `username` varchar(80) NOT NULL DEFAULT '',
  `regserver` varchar(100) DEFAULT NULL,
  `regseconds` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

CREATE TABLE IF NOT EXISTS `sipusers` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(80) NOT NULL DEFAULT '',
  `host` varchar(31) NOT NULL DEFAULT '',
```

```
  `nat` varchar(5) NOT NULL DEFAULT 'no',
  `type` enum('user','peer','friend') NOT NULL DEFAULT 'friend',
  `accountcode` varchar(20) DEFAULT NULL,
  `amaflags` varchar(13) DEFAULT NULL,
  `call-limit` smallint(5) unsigned DEFAULT NULL,
  `callgroup` varchar(10) DEFAULT NULL,
  `callerid` varchar(80) DEFAULT NULL,
  `cancallforward` char(3) DEFAULT 'yes',
  `canreinvite` char(3) DEFAULT 'yes',
  `context` varchar(80) DEFAULT NULL,
  `defaultip` varchar(15) DEFAULT NULL,
  `dtmfmode` varchar(7) DEFAULT NULL,
  `fromuser` varchar(80) DEFAULT NULL,
  `fromdomain` varchar(80) DEFAULT NULL,
  `insecure` varchar(25) DEFAULT NULL,
  `language` char(2) DEFAULT NULL,
  `mailbox` varchar(50) DEFAULT NULL,
  `md5secret` varchar(80) DEFAULT NULL,
  `deny` varchar(95) DEFAULT NULL,
  `permit` varchar(95) DEFAULT NULL,
  `mask` varchar(95) DEFAULT NULL,
  `musiconhold` varchar(100) DEFAULT NULL,
  `pickupgroup` varchar(10) DEFAULT NULL,
  `qualify` char(3) DEFAULT NULL,
  `regexten` varchar(80) DEFAULT NULL,
  `restrictcid` char(3) DEFAULT NULL,
  `rtptimeout` char(3) DEFAULT NULL,
  `rtpholdtimeout` char(3) DEFAULT NULL,
  `secret` varchar(80) DEFAULT NULL,
  `setvar` varchar(100) DEFAULT NULL,
  `disallow` varchar(100) DEFAULT NULL,
  `allow` varchar(100) DEFAULT NULL,
  `fullcontact` varchar(80) NOT NULL DEFAULT '',
  `ipaddr` varchar(15) NOT NULL DEFAULT '',
  `port` mediumint(5) unsigned NOT NULL DEFAULT '0',
  `regserver` varchar(100) DEFAULT NULL,
  `regseconds` int(11) NOT NULL DEFAULT '0',
  `lastms` int(11) NOT NULL DEFAULT '0',
  `username` varchar(80) NOT NULL DEFAULT '',
  `defaultuser` varchar(80) NOT NULL DEFAULT '',
  `subscribecontext` varchar(80) DEFAULT NULL,
  `useragent` varchar(20) DEFAULT NULL,
  `sippasswd` varchar(80) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name_uk` (`name`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

CREATE TABLE IF NOT EXISTS `version` (
  `table_name` varchar(32) NOT NULL,
  `table_version` int(10) unsigned NOT NULL DEFAULT '0'
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
INSERT INTO version (table_name, table_version) VALUES ('sipusers','6');
```

Debido a que la instalación de FreePBX ya nos generó una BBDD de nombre "*asterisk*" la nueva BBDD deberá tener un nombre diferente.

Creado el fichero podemos ejecutar su contenido gracias a las herramientas que proporciona MySQL:

```
root@freepbx-1:~# mysql -u root -p <asterisktmp.sql
```

Asignamos los permisos necesarios para poder acceder desde los hosts donde está instalado Kamailio:

```
root@freepbx-1:~# mysql -u root -p
mysql> GRANT ALL ON asterisktmp.* TO 'asterisk'@'192.168.1.%' IDENTIFIED BY 's3cr3t0';
Query OK, 0 rows affected (0.00 sec)
```

Las tablas generadas corresponden a:

- *sipusers* - Tabla que almacenará los perfiles de los usuarios SIP.
- *sipregs* - Tabla que almacenará los registros SIP producidos.
- *version* – Tabla que Kamailio utiliza para verificar la estructura de la BD.

El siguiente paso será crear algún usuario en la tabla *sipusers* para ello copiaremos las extensiones creadas con FreePBX p.ej:

```
INSERT INTO `sipusers` (`id`, `name`, `host`, `nat`, `type`, `accountcode`, `amaflags`, `call-limit`,
`callgroup`, `callerid`, `cancallforward`, `canreinvite`, `context`, `defaultip`, `dtmfmode`, `fromuser`,
`fromdomain`, `insecure`, `language`, `mailbox`, `md5secret`, `deny`, `permit`, `mask`, `musiconhold`,
`pickupgroup`, `qualify`, `regexten`, `restrictcid`, `rtptimeout`, `rtpholdtimeout`, `secret`, `setvar`,
`disallow`, `allow`, `fullcontact`, `ipaddr`, `port`, `regserver`, `regseconds`, `lastms`, `username`,
`defaultuser`, `subscribecontext`, `useragent`, `sippasswd`) VALUES
(1, '101', 'dynamic', 'no', 'friend', NULL, NULL, NULL, NULL, 'device <101>', 'yes', 'no', 'from-internal',
NULL, 'rfc2833', '101', NULL, NULL, NULL, '101@device', NULL, '0.0.0.0/0.0.0.0', '0.0.0.0/0.0.0.0',
NULL, NULL, NULL, 'yes', NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, '', '', 0, '', 0, 0,
'101', '', NULL, '', 'secreto123');
```

A continuación será necesario configurar la funcionalidad ARA comentada anteriormete, para ello en primer lugar definiremos el driver, generando el fichero **/etc/odbcinst.ini** con el siguiente contenido:

```
root@freepbx-1:~# cat /etc/odbcinst.ini
[MySQL]
Description = MySQL driver
Driver = /usr/lib/odbc/libmyodbc.so
Setup = /usr/lib/odbc/libodbcmyS.so
CPTimeout =
CPReuse =
UsageCount = 1
```

El siguiente paso es crear el conector para la BBDD editando el fichero **/etc/odbc.ini** : La dirección del servidor corresponde a la IP de clúster como veremos posteriormente en la sección de “Alta disponibilidad”:

```
root@freepbx-1:~# cat /etc/odbc.ini
[MySQL-asterisktmp]
Description = MySQL Asterisk database
```

```
Trace = Off
TraceFile = stderr
Driver = MySQL
SERVER = 172.16.1.242
USER = asterisk
PASSWORD = s3cr3t0
PORT = 3306
DATABASE = asterisktmp
```

Finamente configuramos Asterisk para que implemente la funcionalidad ARA. Para ello en primer lugar será necesario editar el fichero **/etc/asterisk/res_odbc.conf** correspondiente a la configuración del módulo ODBC:

```
root@freepbx-1:~# cat /etc/asterisk/res_odbc.conf | grep . |grep -v '^;'
[ENV]
[asterisktmp]
enabled => yes
dsn => MySQL-asterisktmp
username => asterisk
password => s3cr3t0
pre-connect => yes
```

Editamos el fichero **/etc/asterisk/extconfig.conf** para indicar a Asterisk que información es la que debe obtener de la BBDD:

```
root@freepbx-1:~# cat /etc/asterisk/extconfig.conf | grep . |grep -v '^;'
[settings]
sipusers => odbc,asterisktmp,sipusers
sippeers => odbc,asterisktmp,sipusers
```

Por último activaremos **rtcachefriends=yes** en fichero **/etc/asterisk/sip.conf** para que al producirse el registro de una extensión Asterisk lo guarde en su caché. Ya que cuando Asterisk se inicie, las extensiones no se cargaran en la memoria de manera automática sino cuando estas contacten con el servidor.

```
root@freepbx-1:~# cat /etc/asterisk/sip.conf | grep rtc
rtcachefriends=yes
```

4.3.3 Configuración de Kamailio

Como se comentara anteriormente el directorio de configuración de Kamailio (según la instalación realizada en este documento) se encuentra en **/usr/local/etc/kamailio**.
La configuración de Kamailio se realiza principalmente a través de 2 ficheros que encontraremos en este directorio, el primero de ellos es **kamctlrc**, (el cual ya editamos anteriormente para indicar el motor de BD que utilizaríamos) que es utilizado por la herramienta *kamctl* para la gestión de BD , el segundo y de más relevancia es el fichero **kamailio.cfg**, adicionalmente y como veremos posteriormente otros ficheros de configuración especifica de algunos módulos puede ser especificados.
Como decíamos, el fichero de configuración principal es **kamailio.cfg**, este, se divide principalmente en 3 secciones:

- *Configuración Global*- En la que se establecen las variables generales como el modo de ejecución, la IP y puerto de escucha, el nivel de log...

- *Definición y Configuración de módulos* - En esta sección, como su nombre indica se definen los módulos utilizados por Kamailio y los parámetros de configuración de estos.

- *Definición de rutas* – En esta última sección se define como procesar cada una de las peticiones SIP que Kamailio recibe.

La instalación de Kamailio genera por defecto un fichero de configuración funcional pero a la vez complejo, incluyendo algunos módulos que pese a ser interesantes se consideran prescindibles en la arquitectura definida en este proyecto, como podrían ser los módulos de NAT o TLS.
A continuación analizaremos el fichero de configuración kamailio.cfg y comentando cada detalle de la configuración personalizada para el proyecto.

Sección 1 Configuración Global

La primera de las secciones que compone este fichero como ya se ha comentado, corresponde a los parámetros de configuración general del servicio. Al estilo del lenguaje “C”, el fichero de configuración permite definir directivas que determinarán que parte del código se ejecutará asegurando un menor consumo de memória y mayor rapidez en la ejecución. Definimos la variable DBURL que utilizaremos posteriormente para la conexión a la BBDD local y la variable DBASTURL para la conexión a la BBDD del clúster de servidores Asterisk que utilizaremos para la autenticación y el registro de los usuarios SIP.

Fichero kamailio.cfg

```
#!KAMAILIO

#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_ASTERISK
####### Defined Values #########

# *** Value defines - IDs used later in config
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!define DBURL "mysql://openser:openserrw@localhost/openser"
#!define DBASTURL "mysql://asterisk:s3cr3t0@192.168.1.242/asterisktmp"
#!endif

# - flags
#   FLT_ - per transaction (message) flags
#       FLB_ - per branch flags
#!define FLT_ACC 1
#!define FLT_ACCMISSED 2
#!define FLT_ACCFAILED 3
...
```

A continuación se definen las parámetros globales que afectan a la configuración del

servidor, como por ejemplo la IP y puerto de escucha del servicio, el nivel de log o el tipo de ejecución (background fork=yes).
En este caso se ha establecido que Kamailio escuche las peticiones SIP en la dirección udp:192.168.1.238:5060, correspondiente a la IP del clúster de servidores Kamailio.
En esta sección definiremos también las variables que identificarán a los servidores Asterisk y Kamailio para posteriormente poder referenciarlas.

Fichero kamailio.cfg

```
...
####### Global Parameters #########
#!ifdef WITH_DEBUG
debug=4
log_stderror=yes
#!else
debug=2
log_stderror=no
#!endif
memdbg=5
memlog=5
log_facility=LOG_LOCAL0
fork=yes
children=4
disable_tcp=yes
auto_aliases=no
listen=192.168.1.238
port=5060

####### Custom Parameters #########
# These parameters can be modified runtime via RPC interface
# - see the documentation of 'cfg_rpc' module.
# Format: group.id = value 'desc' description
# Access: $sel(cfg_get.group.id) or @cfg_get.group.id

#!ifdef WITH_ASTERISK
asterisk.bindip = "192.168.1.42" desc "Asterisk IP Address"
asterisk.bindport = "5060" desc "Asterisk Port"
asterisk2.bindip = "192.168.1.142" desc "Asterisk IP Address"
asterisk2.bindport = "5060" desc "Asterisk Port"
kamailio.bindip = "192.168.1.238" desc "Kamailio IP Address"
kamailio.bindport = "5060" desc "Kamailio Port"
#!endif
...
```

Sección 2 Definición y configuración de modulos

En la segunda de las secciones se definen aquellos módulos externos que deben cargarse para ser utilizados algunos de estos módulos dependen de otros así que es importante el orden en que estos son cargados.

Fichero kamailio.cfg

```
...
#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif

loadmodule "mi_fifo.so"
```

```
loadmodule "kex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
loadmodule "sl.so"
loadmodule "rr.so"
loadmodule "pv.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "siputils.so"
loadmodule "xlog.so"
loadmodule "sanity.so"
loadmodule "ctl.so"
loadmodule "mi_rpc.so"
loadmodule "acc.so"

#!ifdef WITH_AUTH
loadmodule "auth.so"
loadmodule "auth_db.so"
#!endif

#!ifdef WITH_ASTERISK
loadmodule "uac.so"
#!endif
loadmodule "dispatcher.so"
...
```

Una vez definidos los módulos que serán cargado si los módulos lo requieren se configurará el comportamiento de estos.

El módulo **mi_fifo** proporciona una capa intermedia que permite al programa kamctl interactuar con el servició a través de comandos externos.

Fichero kamailio.cfg

```
...
# ----------------- setting module-specific parameters ---------------
# ----- mi_fifo params -----
modparam("mi_fifo", "fifo_name", "/tmp/kamailio_fifo")
...
```

El módulo **tm** permite el proceso de transacciones SIP en stateful. Pese a que el procesamiento stateful requiere de más recursos (al mantener el estado de las peticiones SIP), este módulo es necesario para uno de los aspectos más importantes de este proyecto como es la distribución de carga. Posteriormente comentaremos el módulo encargado de la distribución de carga (**dispatcher**), pero este requiere del módulo **tm.** El parámetro *fr_timer* establece el tiempo límite (en ms) para determinar como proceder con una petición de la que no se ha recibido respuesta.
Por su parte el parametro *fr_inv_timer* establece el tiempo límite (en ms) si no se ha recibido respuesta a una petición INVITE de la que se recibió un mensaje provisional.

Fichero kamailio.cfg

```
...
# default retransmission timeout: 30sec
modparam("tm", "fr_timer", 3000)
```

```
# default invite retransmission timeout after 1xx: 120sec
modparam("tm", "fr_inv_timer", 40000)
...
```

El módulo **rr** Record-Route se utilizada para resolver problemas que afectan a las cabeceras de las peticiones, provocadas normalmente cuando hay algún NAT de por medio.

Fichero kamailio.cfg

```
...
# ----- rr params -----
# add value to ;lr param to cope with most of the UAs
modparam("rr", "enable_full_lr", 1)
# do not append from tag to the RR (no need for this script)
#!ifdef WITH_ASTERISK
modparam("rr", "append_fromtag", 1)
#!else
modparam("rr", "append_fromtag", 0)
#!endif
...
```

El módulo **usrloc** mantiene una tabla actualizada con la localización de cada usuario (URI SIP) en este caso esta información se almacena en la BBDD como detallamos durante el proceso de instalación, así Kamailio es capaz de poner en contacto a 2 usuarios para que establezcan conversación.

Fichero kamailio.cfg

```
...
# ----- usrloc params -----
/* enable DB persistency for location entries */
#!ifdef WITH_USRLOCDB
modparam("usrloc", "db_url", DBURL)
modparam("usrloc", "db_mode", 2)
#!endif
...
```

El módulo **auth_db** contiene todas las funciones relacionadas con el acceso a bases de datos, como dijimos, la interacción con Asterisk se realiza a través de una BBDD MySQL.

Fichero kamailio.cfg

```
...
# ----- auth_db params -----
#!ifdef WITH_AUTH
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "load_credentials", "")
#!ifdef WITH_ASTERISK
modparam("auth_db", "user_column", "username")
modparam("auth_db", "password_column", "sippasswd")
modparam("auth_db", "db_url", DBASTURL)
#!else
modparam("auth_db", "db_url", DBURL)
modparam("auth_db", "password_column", "password")
#!endif
#!endif
```

```
...
```

El último de los módulos que encontramos en el fichero es **dispatcher**. Este módulo como comentamos anteriormente es de especial relevancia en la arquitectura definida en este proyecto puesto que es el encargado de distribuir la carga entre los 2 servidores Asterisk.
El módulo dispatcher permite varios algoritmos para poder balancear el tráfico. En este caso el algoritmo elegido es **Round-Robin**.
El módulo requiere de el comentado módulo **tm**.
Debido a la relavancia de este módulo se comenta en el propio código la funcionalidad de cada uno de los parametros de configuración.

Fichero kamailio.cfg

```
...
# ----- dispatcher params -----
/*Sobreescribir la dirección de destino(outbound proxy). De esta forma en caso de caída de uno de los Asterisk  la petición puede reenviarse sin perderse */
modparam("dispatcher", "force_dst", 1)
/*Si el flag se establece a 3 se habilita el soporte failover. Las funciones exportadas por el módulo almacenarán el resto de las direcciones de destino AVPs que se utilizarán para contactar con la siguiente dirección en caso de fallida. */
modparam("dispatcher", "flags", 3)
/*Los AVP (attribute-value-pair) son variables que en este caso almacenarán una lista con los destinos entre los que se deberá balancear según el algoritmo escogido*/
modparam("dispatcher", "dst_avp", "$avp(dsdst)")
modparam("dispatcher", "grp_avp", "$avp(dsgrp)")
modparam("dispatcher", "cnt_avp", "$avp(dscnt)")
modparam("dispatcher", "attrs_avp", "$avp(dsattrs)")
/*Parametros para la  monitorización de los destinos*/
#decidir si el destino esta inactivo tras 5 intentos
modparam("dispatcher", "ds_probing_threshhold", 5)
#monitorizar los destinos para activarlos/desactivarlos
modparam("dispatcher", "ds_probing_mode", 1)
#metodo de monitorizacion solicitando peticion SIP OPTIONS
modparam("dispatcher", "ds_ping_method", "OPTIONS")
modparam("dispatcher", "ds_ping_from", "sip:kamailio@192.168.1.238")
#Intervalo de monitorizacion cada 15 segundos
modparam("dispatcher", "ds_ping_interval", 15)
#---------- end dispatcher
...
```

Por defecto el módulo dispatcher buscará dentro de su directorio de configuración un fichero llamado **dispather.list** que contendrá un listado con los servidores a los que se quiere balancear el tráfico. El listado de servidores puede realizare en cualquier otra ubicación que debería indicarse con el parámetro modparam("dispatcher", "list_file", "/path/to/dispatcher.list"). Alternativamente también puede guardarse en BBDD.

```
root@kam1:/replica/kamailio# cat dispatcher.list
#ASTERISK
1 sip:192.168.1.42:5060
1 sip:192.168.1.142:5060
```

Sección 3 Lógica de enrutamiento

La útlima de las secciones que componen el fichero kamailio.cfg es la que corresponde a la lógica que debe realizar cualquier petición SIP que sea recibida por el proxy. Sin duda es la parte más compleja donde cualquier error de configuración puede provocar un funcionamiento inadecuado del sistema.
La sección de rutas suele dividirse en 3 secciones más, la primera incluye la ruta principal que procesará toda petición SIP y que tendrá llamadas a las rutas secundarias.
La segunda esta compuesta por rutas secundarias que son referenciadas desde la ruta principal como si se tratara de rutinas, haciendo de esta forma más legible el código.
La última sección esta compuestas por rutas que sirven para el control frente a posibles fallos producidos al no procesarse de la forma esperadas las anteriores rutinas.
Continuando con el fichero de configuración kamailio.cfg el contenido correspondiente a las rutas utilizadas en este proyecto es el siguiente:

Fichero kamailio.cfg

```
...
####### Routing Logic ########
# Main SIP request routing logic
# - processing of any incoming SIP request starts with this route
route {

	# per request initial checks
	route(REQINIT);

	# handle requests within SIP dialogs
	route(WITHINDLG);

	### only initial requests (no To tag)

	# CANCEL processing
	if (is_method("CANCEL"))
	{
		if (t_check_trans())
			t_relay();
		exit;
	}

	t_check_trans();

	# authentication
	route(AUTH);

	# record routing for dialog forming requests (in case they are routed)
	# - remove preloaded route headers
	remove_hf("Route");
	if (is_method("INVITE|SUBSCRIBE"))
		record_route();
	# account only INVITEs
	if (is_method("INVITE"))
	{
		setflag(FLT_ACC); # do accounting
	}
```

```
        # handle presence related requests
        route(PRESENCE);

        # handle registrations
        route(REGISTRAR);

        if ($rU==$null)
        {     # request with no Username in RURI
              sl_send_reply("484","Address Incomplete");
              exit;
        }

       # user location service
        route(LOCATION);
        route(RELAY);
} # end route 0
...
```

Como puede observarse la ruta principal (o también 0), contienen principamente llamadas a rutas secundarias que procesan el tratamiento de cada de petición SIP. Detallaremos a continuación cada una de estas rutas secundarias en el mismo orden en que son referenciadas en la ruta principal. Estas dos primeras rutas son utlizadas en primer lugar para verificar la integridad de la petición SIP, y en segundo para verificar si la petición pertenece a un dialogo ya establecido.

Fichero kamailio.cfg

```
...
route[REQINIT] {
        if (!mf_process_maxfwd_header("10")) {
              sl_send_reply("483","Too Many Hops");
              exit;
        }

        if(!sanity_check("1511", "7"))
        {
              xlog("Malformed SIP message from $si:$sp\n");
              exit;
        }
}

# Handle requests within SIP dialogs
route[WITHINDLG] {
        if (has_totag()) {
              # sequential request withing a dialog should
              # take the path determined by record-routing
              if (loose_route()) {
                    if (is_method("BYE")) {
                          setflag(1); # do accounting ...
                          setflag(3); # ... even if the transaction fails
                    }
                    route(RELAY);
              } else {
                    if (is_method("SUBSCRIBE") && uri == myself) {
```

```
                        # in-dialog subscribe requests
                        route(PRESENCE);
                        exit;
                }
                if ( is_method("ACK") ) {
                        if ( t_check_trans() ) {
                                # non loose-route, but stateful ACK;
                                # must be ACK after a 487 or e.g. 404 from upstream server
                                t_relay();
                                exit;
                        } else {
                                # ACK without matching transaction ... ignore and discard.
                                exit;
                        }
                }
                sl_send_reply("404","Not here");
        }
        exit;
    }
}
...
```

En la ruta AUTH se procesan todas las autenticaciones de los usuarios, que como ya hemos comentado se realizan a través de la BBDD definida en el módulo **auth_db**. Correspondiente al clúster de servidores Asterisk.

Fichero kamailio.cfg

```
...
# Authentication route
route[AUTH] {
#!ifdef WITH_AUTH
#!ifdef WITH_ASTERISK
    # do not auth traffic from Asterisk - trusted!
    if(route(FROMASTERISK))
        return;
#!endif
    if (is_method("REGISTER"))
    {
        # authenticate the REGISTER requests (uncomment to enable auth)
#!ifdef WITH_ASTERISK
        if (!www_authorize("$td", "sipusers"))
#!else
        if (!www_authorize("$td", "subscriber"))
#!endif
        {
            www_challenge("$td", "0");
            exit;
        }

        if ($au!=$tU)
        {
            sl_send_reply("403","Forbidden auth ID");
            exit;
        }
    } else {
      # authenticate if from local subscriber
```

```
            if (from_uri==myself)
            {
#!ifdef WITH_ASTERISK
                if (!proxy_authorize("$fd", "sipusers")) {
#!else
                if (!proxy_authorize("$fd", "subscriber")) {
#!endif
                    proxy_challenge("$fd", "0");
                    exit;
                }
                if (is_method("PUBLISH"))
                {
                    if ($au!=$tU) {
                        sl_send_reply("403","Forbidden auth ID");
                        exit;
                    }
                } else {
                    if ($au!=$fU) {
                        sl_send_reply("403","Forbidden auth ID");
                        exit;
                    }
                }
                consume_credentials();
                # caller authenticated
            } else {
                # caller is not local subscriber, then check if it calls
                # a local destination, otherwise deny, not an open relay here
                if (!uri==myself)
                {
                    sl_send_reply("403","Not relaying");
                    exit;
                }
            }
        }
#!endif
        return;
}
...
```

La siguiente ruta se utiliza para generar mensajes de notificación a solicitudes *PUBLISH* o *SUBSCRIBE*.

Fichero kamailio.cfg

```
...
# Presence server route
route[PRESENCE] {

        if(!is_method("PUBLISH|SUBSCRIBE"))
                return;

        sl_send_reply("404", "Not here");
        exit;

}
...
```

Las peticiones de registro de los usuarios se procesan en la ruta *REGISTRAR*. Esta a su vez reenvía dichas peticiones a la ruta *REGFWD*.

Fichero kamailio.cfg

```
...
# Handle SIP registrations
route[REGISTRAR] {
	if (is_method("REGISTER"))
	{
		if(isflagset(FLT_NATS))
		{
			setbflag(FLB_NATB);
			# uncomment next line to do SIP NAT pinging
			## setbflag(FLB_NATSIPPING);
		}
		if (!save("location"))
			sl_reply_error();

#!ifdef WITH_ASTERISK
		route(REGFWD);
#!endif

		exit;
	}
}
...
```

La ruta REGFWD, reenvía todas las peticiones de registro a los servidores Asterisk, mediante las variables que definimos en la primera sección. Registrando los usuarios en ambos servidores se conseguirá como veremos a continuación, distribuir las llamadas entre estos.

Fichero kamailio.cfg

```
...
# Presence server route
# Forward REGISTER to Asterisk
route[REGFWD] {
	if(!is_method("REGISTER"))
	{
		return;
	}
	$var(rip) = $sel(cfg_get.asterisk.bindip);
	$uac_req(method)="REGISTER";
	$uac_req(ruri)="sip:" + $var(rip) + ":" + $sel(cfg_get.asterisk.bindport);
	$uac_req(furi)="sip:" + $au + "@" + $var(rip);
	$uac_req(turi)="sip:" + $au + "@" + $var(rip);
	$uac_req(hdrs)="Contact: <sip:" + $au + "@"
			+ $sel(cfg_get.kamailio.bindip)
			+ ":" + $sel(cfg_get.kamailio.bindport) + ">\r\n";
	if($sel(contact.expires) != $null)
		$uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $sel(contact.expires) + "\r\n";
	else
		$uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $hdr(Expires) + "\r\n";
	uac_req_send();
```

```
	#Registro Asterisk 2
	$var(rip) = $sel(cfg_get.asterisk2.bindip);
	$uac_req(method)="REGISTER";
	$uac_req(ruri)="sip:" + $var(rip) + ":" + $sel(cfg_get.asterisk2.bindport);
	$uac_req(furi)="sip:" + $au + "@" + $var(rip);
	$uac_req(turi)="sip:" + $au + "@" + $var(rip);
	$uac_req(hdrs)="Contact: <sip:" + $au + "@"
			+ $sel(cfg_get.kamailio.bindip)
			+ ":" + $sel(cfg_get.kamailio.bindport) + ">\r\n";
	if($sel(contact.expires) != $null)
		$uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $sel(contact.expires) + "\r\n";
	else
		$uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $hdr(Expires) + "\r\n";
	uac_req_send();

}
#!endif
...
```

En la ruta LOCATION se indica que todas las peticiones SIP que no sean INVITE serán tramitadas por kamailio, mientras que los INVITE se reenviarán al clúster Asterisk.

Fichero kamailio.cfg

```
...
# USER location service
route[LOCATION] {

#!ifdef WITH_ASTERISK
	if(!is_method("INVITE")) {
		# non-INVITE request are routed directly by Kamailio
#!endif
	if (!lookup("location")) {
		switch ($rc) {
			case -1:
			case -3:
				t_newtran();
				t_reply("404", "Not Found");
				exit;
			case -2:
				sl_send_reply("405", "Method Not Allowed");
				exit;
		}
	}
#!ifdef WITH_ASTERISK
	} /* end non-INVITE test */
	# only INVITE from now on
	if(route(FROMASTERISK))
	{

		# coming from Asterisk - do location lookup
		if (!lookup("location")) {
			switch ($rc) {
				case -1:
				case -3:
					t_newtran();
					t_reply("404", "Not Found");
```

```
                        exit;
                    case -2:
                        sl_send_reply("405", "Method Not Allowed");
                        exit;
                }
            }
        } else {
            # new call - send to Asterisk
            route(DISPATCH);
        }
#!endif

        # when routing via usrloc, log the missed calls also
        if (is_method("INVITE"))
        {
            setflag(FLT_ACCMISSED);
        }
}
...
```

La ruta FROMASTERISK se utiliza como verificación para comprobar si la petición ha sido originada desde alguno de los servidores Asterisk del clúster.

Fichero kamailio.cfg

```
...
route[FROMASTERISK] {
        if(($si==$sel(cfg_get.asterisk.bindip)&& $sp==$sel(cfg_get.asterisk.bindport)) |
($si==$sel(cfg_get.asterisk2.bindip)&& $sp==$sel(cfg_get.asterisk2.bindport))){
            return 1;
        }
        return -1;
}
...
```

La ruta Dispatch es la que se encarga de distribuir las llamadas entre los dos servidores Asterisk. El algoritmo empleado es Round-Robin, lo determina la función ds_select_dst.

Fichero kamailio.cfg

```
...
route[DISPATCH] {
        # round robin dispatching on gateways group '1'
        if(!ds_select_dst("1", "4"))
        {
            send_reply("404", "No destination");
            exit;
        }
        sl_send_reply("100","Trying");

        xlog("L_INFO", "ROUND ROBIN-- SCRIPT: going to <$ru> via <$du>\n");
        t_on_failure("RTF_DISPATCH");
        return;
}
...
```

En caso de que la ruta DISPATCH falle la petición se reenvía a la ruta RTF_DISPATCH. En esta se determinará si el error se ha producido a causa del servidor, en función de la respuesta recibida, en caso de serlo se procede a marcar este como inactivo e intentar una nueva petición con el siguiente servidor disponible en la lista AVP comentada anteriormente.

Fichero kamailio.cfg

```
...
failure_route[RTF_DISPATCH] {


	xlog("L_INFO", "FAILURE ROUTE: going to <$ru> via <$du>\n");
	if (t_is_canceled()) {
		exit;
	}
	# next DST - para errores 5XX o local timeout
	if (t_check_status("5[0-9][0-9]")or (t_branch_timeout() and !t_branch_replied()))
	{
		ds_mark_dst("i");
		xlog("L_INFO", "FAIL_ONE:time [$Tf] destination $avp(dsdst) marked as inactive \n");
		if(ds_next_dst())
		{

			xlog("L_INFO", "DS_NEXT_DST: going to <$ru> via <$du>\n");
			t_on_failure("RTF_DISPATCH");
			route(RELAY);
			exit;
		}
		else {
			xlog("L_INFO", "FALLA DS_NEXT_DST\n");
		}

	}
}
...
```

Fichero kamailio.cfg

```
...
failure_route[RTF_DISPATCH] {


	xlog("L_INFO", "FAILURE ROUTE: going to <$ru> via <$du>\n");
	if (t_is_canceled()) {
		exit;
	}
	# next DST - para errores 5XX o local timeout
	if (t_check_status("5[0-9][0-9]")or (t_branch_timeout() and !t_branch_replied()))
	{
		ds_mark_dst("i");
		xlog("L_INFO", "FAIL_ONE:time [$Tf] destination $avp(dsdst) marked as inactive \n");
		if(ds_next_dst())
		{

			xlog("L_INFO", "DS_NEXT_DST: going to <$ru> via <$du>\n");
```

```
                t_on_failure("RTF_DISPATCH");
                route(RELAY);
                exit;
        }
        else {
                xlog("L_INFO", "FALLA DS_NEXT_DST\n");
        }

    }
}
...
```

La última ruta que será procesada sera siempre la ruta RELAY encargada de reenviar cada mensaje a su destino o un error en caso de fallo.

```
...
route[RELAY] {
    if (!t_relay()) {
        sl_reply_error();
     }
     exit;
}
```

4.4 MySQL Server

MySQL es un Sistema Gestor de Bases de Datos Relacional (SGBDR) distribuido bajo licencia GNU/GPL y comercial MySQL AB.

Como se ha descrito durante la instalación de FreePBX, este utiliza MySQL por un lado para guardar las configuraciones que posteriormente generará de Asterisk y por otro para consultar los registros del histórico de llamadas producidas.

Además, como también se ha visto MySQL era necesario para la iteración entre el servidor PROXY SIP (Kamailio) y Asterisk.
Por otro lado al realizar la configuración del clúster era necesario tener un único almacén CDR en el que las dos PBX pudieran guardar su histórico de llamadas.

Ante la necesidad de disponer de un entorno replicado de BD, y con la pecurialidad de que en ambos nodos debía ser posible tanto la consulta como la insercción de registros, MySQL ofrece la posibilidad de configurar un singular entorno de replicación Master-Master, en el que cada nodo actúa a la vez com master y slave del otro, permitiendo de esta forma tener una copia sincronizada de los registros, y posibilitando la continuidad del servicio frente a la caida de uno de los dos nodos.

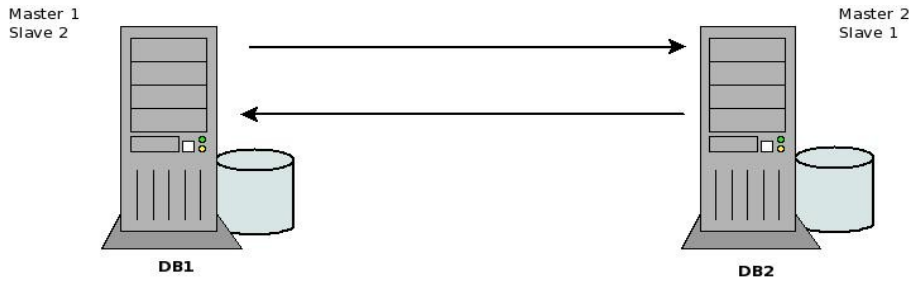


Figura 4.4 MySQL Master-Master Replication.

4.3.1 Configuración Master-Master Replication MySQL

"*Las características de MySQL 5 soportan replicación asíncrona unidireccional: un servidor actúa como maestro y uno o más actúan como esclavos. (Esto contrasta con la replicación síncrona que es una característica de MySQL Cluster — consulte Capítulo 16, MySQL Cluster). El servidor maestro escribe actualizaciones en el fichero de log binario, y mantiene un índice de los ficheros para rastrear las rotaciones de logs. Estos logs sirven como registros de actualizaciones para enviar a los servidores esclavos. Cuando un esclavo se conecta al maestro, informa al maestro de la posición hasta la que el esclavo ha leído los logs en la última actualización satisfactoria. El esclavo recibe cualquier actualización que han tenido lugar desde entonces, y se bloquea y espera para que el master le envíe nuevas actualizaciones.*"

Continuando con el esquema de configuración del clúster de PBX donde las IPs de cada nodo eran:

freepbx-1.pbx.lan IP-CLUSTER: 172.16.1.42
freepbx-2.pbx.lan IP-CLUSTER: 172.16.1.142
failover IP : 192.168.1.242

El primer paso consiste en permitir el acceso desde otros hosts al servicio MySQL, para ello es necesario modificar el valor de la variable "bind-address" del fichero **/etc/mysql/my.cnf** en ambos hosts.

```
bind-address            = 0.0.0.0
```

Posteriormente podriamos, mediante reglas iptables restringuir el acceso al servicio para aquellos hosts que nos interese.

El segundo paso consiste en dar permisos de replicación en ambos nodos.

En freepbx-1.pbx.lan:

```
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT
ON *.*  TO 'slave-fpbx2'@'172.16.1.%' IDENTIFIED BY 's3cr3t0';
FLUSH PRIVILEGES;
mysql> quit;
```

En freepbx-2.pbx.lan:

```
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT
ON *.* TO 'slave-fpbx1'@'192.168.1.% IDENTIFIED BY 's3cr3t0';
FLUSH PRIVILEGES;
mysql> quit;
```

Para que cada uno de los nodos pueda realizar insercciones sobre la misma tabla sin que se produzca ninguna incoherencia es importante establecer el valor de las variables **auto_increment_increment** y **auto_increment_offset** ,destinadas para utilizar en una replicación master-master, estas variables determinan el punto de inicio para el valor de AUTO_INCREMENT. Ambas variables tienen valores globales y de sesión, y cada uno puede asumir un valor entero entre 1 y 65535 inclusive.
En el entorno planteado en este proyecto con 2 nodos, este es por tanto el valor que

se le debe asignar a la variable **auto_increment_increment**, por otro lado el valor del offset (**auto_increment_offset**) debe ser distinto en cada uno de los nodos para que de esta forma cada uno siga su propia secuencia a la hora de insertar registros.

De nuevo será necesario editar el fichero **/etc/mysql/my.cnf** la respectiva configuración para cada uno de los nodos sería la siguiente:

En freepbx-1.pbx.lan:

```
server-id = 1
replicate-same-server-id = 0
auto-increment-increment = 2
auto-increment-offset = 1

master-host = freepbx-2
master-user = slave-fpbx1
master-password = s3cr3t0
master-connect-retry = 60

log-bin=/var/lib/mysql/master-bin.log
log-bin-index=/var/lib/mysql/master-bin.index
log-error=/var/lib/mysql/master-error.log

relay-log = /var/lib/mysql/slave-relay.log
relay-log-index = /var/lib/mysql/slave-relay-log.index

expire_logs_days       = 10

max_binlog_size        = 100M
```

En freepbx-2.pbx.lan:

```
server-id              = 2
replicate-same-server-id = 0
auto-increment-increment = 2
auto-increment-offset = 2

master-host = freepbx-1
master-user = slave-fpbx2
master-password = s3cr3t0
master-connect-retry = 60

log-bin=/var/lib/mysql/master-bin.log
log-bin-index=/var/lib/mysql/master-bin.index
log-error=/var/lib/mysql/master-error.log

relay-log = /var/lib/mysql/slave-relay.log
relay-log-index = /var/lib/mysql/slave-relay-log.index
relay-log-info=/var/lib/mysql/slave-error-info
```

Es importante tener en cuenta que esta replicación a la hora de insertar registros solo puede llevarse a cabo en el supuesto de que la tabla que deseamos actualizar de forma replicada debe tener agún campo con el atributo AUTO_INCREMENT y que este además se clave primaria, para idenficar unequívocamente cada nueva fila insertada.

En el caso que concierne a este proyecto se deseaba tener un única tabla de CDR. Como pudimos ver durante la instalación de FreePBX creamos una BD llamada

asteriskcdrdb, esta contiene una única tabla llamada cdr, que como vimos anteriormente sirve para almacenar los registros de las llamadas producidas en el entorno. Sin embargo esta tabla no tenía la característica de disponer de un primary-key con el atributo auto-increment así, que para poder realizar insercciones en en entorno master-master era importante establecer este nuevo campo.
Para el resto de tabla hemos supuestos que cualquier otra modificación se realizaría exclusivamente desde el nodo master.

En freepbx-1.pbx.lan:

```
mysql> use asteriskcdrdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE `cdr` ADD `id_ha` INT NOT NULL AUTO_INCREMENT FIRST ,
ADD PRIMARY KEY ( `id_ha` ) ;

mysql> desc cdr;
+-------------+--------------+------+-----+---------------------+----------------+
| Field       | Type         | Null | Key | Default             | Extra          |
+-------------+--------------+------+-----+---------------------+----------------+
| id_ha       | int(11)      | NO   | PRI | NULL                | auto_increment |
| calldate    | datetime     | NO   |     | 0000-00-00 00:00:00 |                |
| clid        | varchar(80)  | NO   |     |                     |                |
| src         | varchar(80)  | NO   |     |                     |                |
| dst         | varchar(80)  | NO   |     |                     |                |
| dcontext    | varchar(80)  | NO   |     |                     |                |
| channel     | varchar(80)  | NO   |     |                     |                |
| dstchannel  | varchar(80)  | NO   |     |                     |                |
| lastapp     | varchar(80)  | NO   |     |                     |                |
| lastdata    | varchar(80)  | NO   |     |                     |                |
| duration    | int(11)      | NO   |     | 0                   |                |
| billsec     | int(11)      | NO   |     | 0                   |                |
| disposition | varchar(45)  | NO   |     |                     |                |
| amaflags    | int(11)      | NO   |     | 0                   |                |
| accountcode | varchar(20)  | NO   |     |                     |                |
| uniqueid    | varchar(32)  | NO   |     |                     |                |
| userfield   | varchar(255) | NO   |     |                     |                |
+-------------+--------------+------+-----+---------------------+----------------+
17 rows in set (0.01 sec)

mysql> quit;
```

En el supuesto que hubiera algún registro en la tabla cdr, sería necesario bloquear la lectura antes de exportar a la BD del segundo servidor. Previamente reiniciaremos el servicio en el primer servidor para que se apliquen los nuevos cambios en la configuración.

```
root@freepbx-1:~# /etc/init.d/mysql restart
```

En freepbx-1.pbx.lan:

```
mysql> use asteriskcdrdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql> FLUSH TABLES WITH READ LOCK;
```

Verificamos el estado del Master, y guardamos la información ya que será importante para poder asignar posteriormente el slave de la replicación.

En freepbx-1.pbx.lan:

```
mysql> SHOW MASTER STATUS;
+----------------------+----------+------------------+----------------------+
| File                 | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------------+----------+------------------+----------------------+
| master-bin.000006 |     881   |                      |                          |
+----------------------+----------+------------------+----------------------+
1 row in set (0.01 sec)
```

Sin salir de la consola de mysql (será necesario abrir una nueva shell), puesto que se desbloquearía la BD. Exportamos, en este caso, todas las BD al segundo servidor.

```
root@freepbx-1:~# mysqldump -u root -p --all-databases > /tmp/master.sql
root@freepbx-1:~# scp /tmp/master.sql freepbx-2:/tmp
```

```
root@freepbx-2:~# mysql -u root -p < /tmp/master.sql
root@freepbx-2:~# /etc/init.d/mysql restart
```

Ahora podemos desbloquear la BD.

En freepbx-1.pbx.lan:

```
mysql> UNLOCK TABLES;
```

El siguiente paso es convertir a freepbx-2 en slave de freepbx-1, recuperando la información correspondiente al fichero de log binario que previamente guardamos.

En freepbx-2.pbx.lan:

```
mysql> CHANGE MASTER TO  MASTER_HOST='freepbx-1',  MASTER_USER='slave-fpbx2',
MASTER_PASSWORD='s3cr3t0',  MASTER_LOG_FILE='master-bin.000006',
MASTER_LOG_POS=881;
```

Ahora se puede inciar el servicio de slave sobre freepbx-2 y verificar su estado.

En freepbx-2.pbx.lan:

```
mysql> start slave;
mysql> show slave status\G
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: freepbx-1
                Master_User: slave-fpbx2
                Master_Port: 3306
              Connect_Retry: 60
            Master_Log_File: master-bin.000031
        Read_Master_Log_Pos: 190445
             Relay_Log_File: slave-relay.000081
```

```
            Relay_Log_Pos: 1290
        Relay_Master_Log_File: master-bin.000031
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 190445
              Relay_Log_Space: 188293
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
1 row in set (0.01 sec)
```

Hasta este momento se ha conseguido que el host freepbx-2 sea slave de freepbx-1. Para conseguir el estado master-master, necesitamos convertir a su vez a freepbx-1 como slave de freepbx-2. El procedimiento es identico al que se ha realizado anteriormente.

En freepbx-2.pbx.lan:

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| master-bin.000020 |    10486 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql>
```

En freepbx-1.pbx.lan:

```
mysql> stop slave;
mysql> CHANGE MASTER TO  MASTER_HOST='freepbx-2',  MASTER_USER='slave-fpbx1',
MASTER_PASSWORD='s3cr3t0',  MASTER_LOG_FILE='master-bin.000020',
MASTER_LOG_POS=10486;
```

Iniciamos el servicio de slave sobre freepbx-1 y verificamos su estado.

En freepbx-1.pbx.lan:

```
mysql> start slave;
mysql> show slave status\G
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: freepbx-2
                Master_User: slave-fpbx1
                Master_Port: 3306
              Connect_Retry: 60
            Master_Log_File: master-bin.000020
        Read_Master_Log_Pos: 10486
             Relay_Log_File: slave-relay.000063
              Relay_Log_Pos: 10588
      Relay_Master_Log_File: master-bin.000020
           Slave_IO_Running: Yes
          Slave_SQL_Running: Yes
            Replicate_Do_DB:
        Replicate_Ignore_DB:
         Replicate_Do_Table:
     Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                 Last_Errno: 0
                 Last_Error:
               Skip_Counter: 0
        Exec_Master_Log_Pos: 10486
            Relay_Log_Space: 10885
            Until_Condition: None
             Until_Log_File:
              Until_Log_Pos: 0
         Master_SSL_Allowed: No
         Master_SSL_CA_File:
         Master_SSL_CA_Path:
            Master_SSL_Cert:
          Master_SSL_Cipher:
             Master_SSL_Key:
      Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
             Last_IO_Errno: 0
             Last_IO_Error:
            Last_SQL_Errno: 0
            Last_SQL_Error:
1 row in set (0.00 sec)
```

4.4.2 CDR

Call Detail Records (CDR) es el nombre del sistema que utiliza Asterisk para registrar el histórico de llamadas producidos.
La configuración del modulo CDR se realiza mediante el fichero cdr.conf dentro del directorio de configuración de asterisk (generalmente /etc/asterisk).

```
freepbx-1*CLI> cdr show status
```

```
Call Detail Record (CDR) settings
---------------------------------
  Logging:                 Enabled
  Mode:                    Simple
  Log unanswered calls:    No

* Registered Backends
  -------------------
    sqlite
    csv
    cdr-custom
    Adaptive ODBC
    radius
    ODBC
    res_config_sqlite
```

Por defecto Asterisk almacena estos registros en ficheros csv en el directorio /var/log/asterisk/cdr-csv.
Cada registro está compuesto por una serie de campos separados por coma que corresponden a la siguiente información:

1. **accountcode**: ID de la cuenta. Por defecto vacío.
2. **src**: Número del llamante.
3. **dst**: Extensión de destino.
4. **dcontext**: Contexto de destino.
5. **clid**: Identificación completa del llamante (nombre,número).
6. **channel**: Canal utilizado.
7. **dstchannel**: Canal destino.
8. **lastapp**: Última aplicación del Dialplan ejecutada.
9. **lastdata**: Argumentos enviados a la útlima aplicación.
10. **start**: Fecha y hora del inicio de la llamada.
11. **answer**: Fecha y hora del establecimiento de la llamada.
12. **end**: Fecha y hora del fin de la llamada.
13. **duration**: Número de segundos entre el incio y el fin de la llamada.
14. **billsec**: Segundos entre el “answer” y el “end”. Utilzado para facturación.
15. **disposition**: Que ocurrió con la llamada:ANSWERED,NO ANSWER,BUSY,FAILED
16. **amaflags**: Flag AMA (Automatic Message Acounting) asociado a la llamada.

Asterisk proporciona de forma alternativa una serie de Backends que permiten almacenar estos registros de otra manera.

Gracias por ejemplo al módulo cdr_odbc y mediante la librería unixODBC podemos utilizar la ODBC, una capa de abstracción entre Asterisk (o muchas otras aplicaciones) y la mayoría de las bases de datos.
En este proyecto concreto se ha decido utilizar una base de datos MySQL para almacenar dichos registros.

Cuando se realizó la instalación de FreePBX se creo una BD en MySQL llamada “asteriskcdrdb”.
Esta BD es consultada por FreePBX y mediante la sección de “Reports” podemos recuperar toda la información almacenada pudiendo aplicar filtros como la fecha, el

origen o el destino de una llamada entre otros.
Adicionlmente, estos informes pueden ser exportados a pdf o csv, con los que posteriormente podrían relizarse gráficas.

Aprovechando la BD que generamos durante la instalación de FreePBX y gracias al servicio de replicación Master-Master de MySQL antes descrito, podemos conseguir que ambas centralitas registren sus llamadas en una única BD, facilitando así posterores consultas sobre estos datos.

4.4.2.1 Configuración CDR-ODBC

En primer lugar sera necesario crear un nuevo conector para la BD “asteriskcdrdb” tal y como describiéramos durante el proceso de integración Asterisk-Kamailio.
Así el contenido del fichero /etc/odbc.ini tendrá ahora la siguiente forma en ambos hosts:

```
root@freepbx-2:~# cat /etc/odbc.ini
[MySQL-asterisktmp]
Description = MySQL Asterisk database
Trace = Off
TraceFile = stderr
Driver = MySQL
SERVER = 172.16.1.242
USER = asterisk
PASSWORD = s3cr3t0
PORT = 3306
DATABASE = asterisktmp

[MySQL-asteriskcdrdb]
Description = MySQL Asterisk CDR database
Trace = Off
TraceFile = stderr
Driver = MySQL
SERVER = 172.16.1.242
USER = asterisk
PASSWORD = s3cr3t0
PORT = 3306
DATABASE = asteriskcdrdb
```

Como puede observarse el conector apunta a la dirección IP de failover creada...

Una vez creado el conector, será necesario indicarle Asterisk como conectarse a través de este nuevo conector.
Nuevamente será necesario editar el fichero /etc/asterisk/res_odbc.conf en ambos hosts añadiendo:

```
[asteriskcdrdb]
enabled => yes
dsn => MySQL-asteriskcdrdb
username => asterisk
password => s3cr3t0
pre-connect => yes
```

Finalmente es necesario modificar la configuración del módulo CDR de Asterisk para añadir este nuevo backend.
Para ello editaremos el fichero /etc/asterisk/cdr_odbc.conf que debería tener el siguiente contenido.

```
root@freepbx-2:~# cat /etc/asterisk/cdr_odbc.conf | grep -v '^;'

[global]
dsn => asteriskcdrdb
username => asterisk
password => s3cr3t0
loguniqueid=yes
dispositionstring=yes
table=cdr
```

Captura de pantalla correspondiente a la sección de generación de informes sobre el histórico de llamadas que FreePBX proporciona.

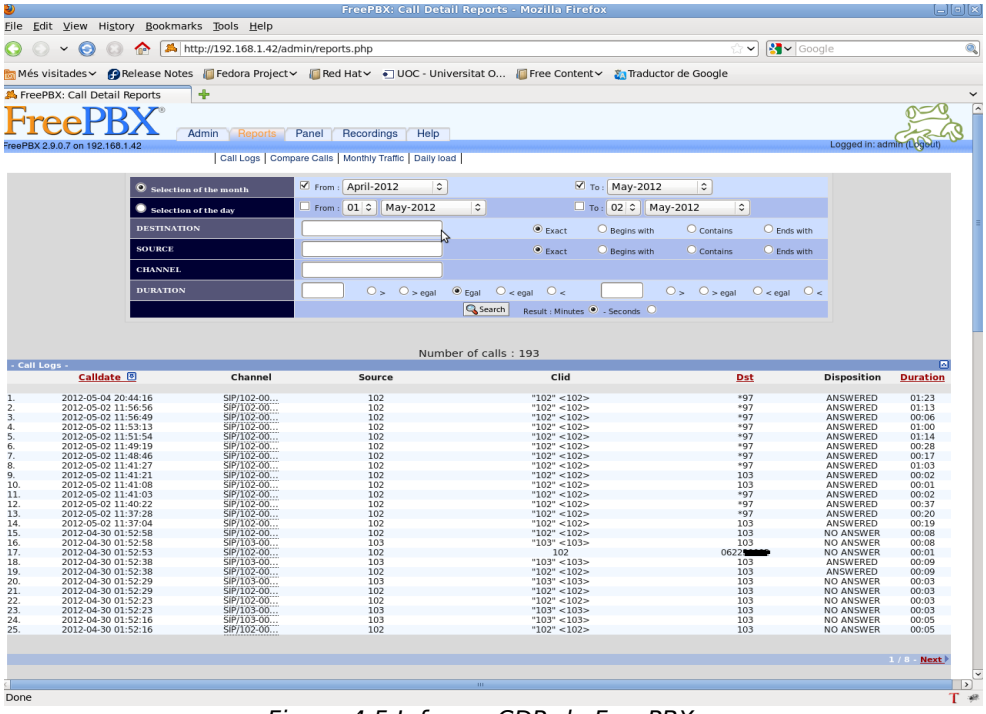


Figura 4.5 Informe CDR de FreePBX

4.5 Servicios adicionales

A continuación se detalla el proceso de instalación y configuración de varios servicios adicionales que se han considerado imprescindibles en la arquitectura diseñada. Estos servicios han sido instalados en los equipos PROXY SIP, aprovechando la configuración de estos en alta disponibilidad.

4.5.1 DHCP Server

La necesidad de disponer de un servicio de DHCP en la vLAN VoIP es vital para que los terminales IP puedan obtener los parámetros de configuración IP de forma automática sin necesidad de tener que asignar manualmente la dirección a cada equipo y evitar el trabajo del control que supondría no duplicar direcciones.

La solución escogida es ISC-DHCP-SERVER, disponible en los repositorios oficiales de Debian, es el servidor DHCP open source más implementado a nivel empresarial.

El servidor permite la opción de configuración de “failover” de esta forma lograremos tener siempre activo el servicio en alguno de los dos servidores.

El proceso de instalación es el siguiente (para ambos hosts):

```
root@kam1:~# apt-get install dhcp3-server
```

El fichero de configuración del servicio se encuentra en /etc/dhcp/dhcpd.conf, dicho fichero deberá tener la siguiete configuración en el servidor principal:

```
authoritative;
ddns-update-style none;

key primaryserver {
        algorithm hmac-md5;
        secret "s3cr3t0";
};

omapi-key primaryserver;
omapi-port 7911;

failover peer "dhcp-failover" {
  primary;
  address 192.168.1.38;
  port 647;
  peer address 192.168.1.138;
  peer port 647;
  max-response-delay 30;
  max-unacked-updates 10;
  load balance max seconds 3;
  mclt 1800;
  split 128;
}


subnet 192.168.0.0 netmask 255.255.252.0 {
  option domain-name "pbx.lan";
  option routers 192.168.1.1;
  option ntp-servers 192.168.1.238;
  option tftp-server-name "192.168.1.238";
  option domain-name-servers 192.168.1.1;
  pool {
```

```
    failover peer "dhcp-failover";
    max-lease-time 7200;
    range 192.168.0.1 192.168.0.255;
    range 192.168.2.0 192.168.3.254;
  }


}
```

En el caso del servidor que actuará como secundario la configuración será la siguiente:

```
authoritative;
ddns-update-style none;

key primaryserver {
        algorithm hmac-md5;
        secret "s3cr3t0";
};

omapi-key primaryserver;
omapi-port 7911;

failover peer "dhcp-failover" {
  secondary; #
  address 192.168.1.138;
  port 647;
  peer address 192.168.1.38;
  peer port 647;
  max-response-delay 30;
  max-unacked-updates 10;
  load balance max seconds 3;
}
subnet 192.168.0.0 netmask 255.255.252.0 {
  option domain-name "pbx.lan";
  option routers 192.168.1.1;
  option ntp-servers 192.168.1.238;
  option tftp-server-name "192.168.1.238";
  option domain-name-servers 192.168.1.1;
  pool {
    failover peer "dhcp-failover";
    max-lease-time 7200;
    range 192.168.0.1 192.168.0.255;
    range 192.168.2.0 192.168.3.254;
  }

}
```

4.5.2 TFTP Server

El servicio de TFTP en una red VoIP tiene el objetivo de facilitar la puesta en marcha de los teléfonos IP.
En entornos donde hay un gran número de terminales como es el caso de este proyecto sería inviable pensar tener que configurar de forma manual, indicando valores como la extensión, la dirección IP propia o la del UAS, cada uno de los terminales.

En este sentido el servidor TFTP nos aportará la funcionalidad de aprovisionar a cada terminal su configuración propia.

Lógicamente los terminales utilizados deben implementar esta funcionalidad.

Mediante la opción 66 del servidor DHCP previamente configurado se le indica a los terminales la dirección donde deberán ir a obtener su configuración.

Para instalar un servidor TFTP en Debian desde sus repositorios oficiales el procedimiento es muy simple (en ambos hosts):

```
root@kam2:~# apt-get install atftpd
```

La configuración del servidor TFTP es muy sencilla, en este caso existe la pecurialidad de que al tratarse de un sistema en alta disponibilidad el servicio ha sido configurado para que en caso de caída del servidor PROXY SIP principal el servicio siga completamente operativo en el servidor secundario.

Sobre el servidor que fuera principal en ese momento (en este caso kam2) crearemos un directorio donde se almacenarán los ficheros de configuración de los terminales, y modificamos los permisos para conceder solo lectura sobre el nuevo directorio.

```
root@kam2:~# mkdir /replica/tftp

root@kam2:~# chmod -R 555 /replica/tftp/

root@kam2:~# chown -R nobody /replica/tftp/
```

Generamos un enlace para que el directorio por defecto que genera la instalación apunte a la nueva ubicación (en ambos hosts).

```
root@kam2:~# rm -r /srv/tftp/

root@kam2:~# ln -s /replica/tftp/ /srv/tftp
```

Por último será necesario modificar la configuración del servicio por defecto para que este no se ejectue con el demonio inetd y lo haga de forma autonoma. De esta forma podremos podremos detener y arrancar el servicio sin problemas cada vez que un host tome el rol principal.

```
root@kam2:~# vi /etc/default/atftpd
USE_INETD=false
```

Finalmente reiniciamos el servicio y desde otro equipo verificamos su funcionamiento.

```
root@kam2:~# /etc/init.d/atftpd restart

root@kam2:~# touch /srv/tftp/test.txt


root@freepbx-1:~# atftp kam2
tftp> put backup.sql
tftp: error received from server <Access violation>
tftp: aborting
tftp> get test.txt
tftp> quit

root@freepbx-1:~# ls
backup.sql  test.txt
```

4.5.3 NFS

NFS (Network File Syste fue desarrollado para permitir a equipos montar particiones sobre un disco remoto como si se tratara de un disco local. Esto permite el intercambio de archivos de forma rápida entre diversos equipos a través de una red TCP/IP.
NFS establece mecanismos de seguridad para limitar el acceso a los recursos compartidos desde el servidor.

Ante la necesidad de tener que acceder a ciertos recursos de forma simultanea desde los dos servidores que actúan como PBX como podían ser:

- **/var/lib/asterisk/sounds/custom**
  Directorio donde FreePBX almacena las grabaciones personalizadas.

- **/var/spool/asterisk/voicemail**
  Directorio donde FreePBX almacena las grabaciones personalizadas.

Se consideró que una posible opción podría ser la de compartir mediante NFS estos datos, de tal forma que ambos hosts pudieran acceder a la misma información de forma simultánea.

Aprovechando el entorno de alta disponibilidad que se ha creado gracias a DRBD y Pacemaker-Corosync, compartiremos mediante NFS una ubicación que estará siempre accesible a través de la IP de failover.
A continuación describiremos el proceso de instalación y configuración que se ha realizado adaptándolo a las necesidades descritas.

En primer lugar instalaremos los paquetes necesarios en los dos nodos:

```
root@freepbx-1:~# apt-get install nfs-kernel-server nfs-comon

root@freepbx-2:~# apt-get install nfs-kernel-server nfs-common
```

Detenemos los servicios iniciados y los quitamos de los runleves, puesto que a partir de ahora el inicio y la parada de los servicios serán gestionados por Pacemaker.

```
root@freepbx-1:~# /etc/init.d/nfs-kernel-server stop
root@freepbx-1:~# /etc/init.d/nfs-common stop
root@freepbx-1:~# insserv -r  nfs-kernel-server
root@freepbx-1:~# insserv -r nfs-common

root@freepbx-2:~# /etc/init.d/nfs-kernel-server stop
root@freepbx-2:~# /etc/init.d/nfs-common stop
root@freepbx-2:~# insserv -r  nfs-kernel-server
root@freepbx-2:~# insserv -r nfs-common
```

Para que el servidor pueda ser montado por ambos servidores es necesario que los datos variables que utiliza NFS esten en una ubicación a la que ambos nodos puedan acceder, esta ubicación será el punto de montaje del dispositivo drbd0. Sin embargo el directorio rpc_pipefs no debería estar en esta ubicación puesto que debe ser legible en todo momento por el servidor.

```
root@freepbx-1:~# mv /var/lib/nfs/rpc_pipefs /var/lib/

root@freepbx-2:~# mv /var/lib/nfs/rpc_pipefs /var/lib/
```

Ahora en el host que tenga en ese instante montado el dispositivo DRBD moveremos deberemos mover los datos a la nueva ubicación. En el segundo host directamente podemos eliminar la ubicación y generar un enlace simbólico que de momento en el nodo secundario permanecerá roto hasta que adquiera el rol de primario.

```
root@freepbx-1:~#  mv /var/lib/nfs/ /replica
root@freepbx-1:~# ln -sn /replica/nfs /var/lib/nfs

root@freepbx-2:~# rm -r /var/lib/nfs/
root@freepbx-2:~# ln -sn /replica/nfs /var/lib/nfs
```

Editaremos el script de inicio del servicio nfs-common (**/etc/init.d/nfs-common**), de ambos nodos para indicar la nueva ubicación de rpc_pipefs.

```
PIPEFS_MOUNTPOINT=/var/lib/rpc_pipefs
```

La misma modificación es necesaria en esta caso en el fichero **/etc/idmapd.conf**

```
Pipefs-Directory = /var/lib/rpc_pipefs
```

Creamos el directorio que compartiremos y asignamos permisos para que el usuario asterisk de ambos nodos pueda acceder, es importante que el usuario asterisk de ambos nodos tengan el mismo UID y GID, ya que sino el usuario asterisk no sería propietario del directorio en el nodo secundario.

```
root@freepbx-1:~#  mkdir /replica/asterisk
root@freepbx-1:~#  chown -R asterisk: /replica/asterisk
```

En el fichero **/etc/exports** indicaremos las opciones del nuevo directorio compartido. Esta operación debe realizarse en ambos nodos.

```
/replica/asterisk 172.16.1.0/255.255.255.0(rw,sync,fsid=0,no_subtree_check)
```

Ahora simplemente necesitamos añadir una entra en el fichero **/etc/fstab** para montar la nueva ubicación de forma automática. De nuevo en ambos nodos.

```
172.16.1.242:/ /nfsasterisk     nfs4    rw,hard,intr    0       0
```

Deberemos crear el directorio que utilizaremos como putno de montaje.

```
root@freepbx-1:~#  mkdir /nfsasterisk
root@freepbx-2:~#  mkdir /nfsasterisk
```

Como puede observarse la dirección del punto de montaje corresponde a la IP de failover de esta forma siempre se podrá acceder al recurso compartido.

4.6 Alta Disponibilidad

Como se comentó en la introducción, el objetivo principal de este proyecto era proporcionar un sistema fiable, por ello desde un principio se pensó en una modelo de alta disponibilidad en el que la caída de alguno de los nodos no supusiera la parada completa del servicio, siendo de esta forma dicho fallo, totalmente transparente para el usuario y permitiendo por otra parte la recuperación del nodo fallido en un segundo plano sin afectar así al servicio.
En esta arquitectura de alta disponibilidad son varios los componentes necesarios que hacen que el sistema sea completamente fiable.

Los componentes que a continuación se describen son por un lado DRBD encargado la replicación de datos, y por otro Pacemaker (CRM) encargado de la gestión de los recursos del clúster.

4.6.1 DRBD

DRBD distribuido bajo GNU/GPL v2, es un sistema de almacenamiento distribuido para sistemas GNU/Linux. Proporciona funcionalidades para emular un RAID1 (espejo de discos) sobre TCP/IP muy útil en entornos de alta disponibilidad.

DRBD permite crear un dispositivo por bloques (drbd0) accesible desde dos servidores. Los datos escritos en el servidor principal son sincronizados en el secundario a través de la red. De esta forma en el instante que el servidor secundario adquiera el rol de principal dispondrá de una copia idéntica de los datos.

Como se ha descrito en la introducción, en el esquema planteado en este proyecto, se dispone de 2 servidores PROXY SIP, en alta disponibilidad. Mientras el nodo principal esté operativo, el secundario no efectuará ninguna acción, sin embargo en el momento que el nodo principal sufra algún fallo, ya sea de software o hardware, y el servidor secundario tome el rol de principal necesitará tener los mismos datos y la misma configuración que el hasta ahora era el servidor principal.
Es en esta situación cuando DRBD, nos es útil permitiéndonos tener los datos totalmente actualizados siempre disponibles en ambos servidores.
En un principio se consideró que esta configuración no sería necesaria para los servidores que actúan como PBX, sin embargo, ante la necesidad de acceder a datos compartidos como se describe en la sección configuración personalizada, se optó por utilizar el mismo sistema de replicación de discos para estos servidores.

4.6.1.1 Instalación de y configuración DRBD

Es importante que los nodos que deben replicarse estén sincronizados, por ello es necesario instalar los paquetes ntp y ntpdate, además como veremos posteriormente estos nos servirán para actuar como servidor de tiempo para los teléfonos IP.

```
root@kam1:~# apt-get install ntp ntpdate
root@kam2:~# apt-get install ntp ntpdate
root@freepbx-1:~# apt-get install ntp ntpdate
root@freepbx-2:~# apt-get install ntp ntpdate
```

Es importante que los dos nodos que formarán cada clúster tengan una partición igual. En este caso y aprovechando la plataforma virtual, se ha presentado un nuevo disco a cada sistema de igual tamaño.
Es importante al crear la nueva partición asignarle el identificador “83 Linux” pero no darle ningún formato.

```
root@freepbx-1:~# fdisk -l /dev/sdb

Disco /dev/sdb: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x5303f71b

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sdb1               1         261     2096451   83  Linux
root@freepbx-1:~#
```

Una vez creada la nueva partición en cada uno de los sistemas, el siguiente paso es la instalación del paquete drbd:

```
root@kam1:~# apt-get install drbd8-utils
root@kam2:~# apt-get install drbd8-utils
root@freepbx-1:~# apt-get install drbd8-utils
root@freepbx-2:~# apt-get install drbd8-utils
```

Las últimas versiones de Debian ya incorporan el módulo drbd compilado en el Kernel con lo que únicamente es necesario cargarlo:

```
root@kam1:~# modprobe drbd
root@kam2:~# modprobe drbds
root@freepbx-1:~#modprobe dbd
root@freepbx-2:~# modprobe drbd
```

El siguiente paso es editar el fichero de configuración de DRBD en cada uno de los equipos, el fichero es **/etc/drbd.conf**. Un aspecto a tener en consideración es los nombres configurados deben coincidir con en nombre completo del host, es recomendable utilizar el comando "*uname -n*" para verificarlo.

En kam1.pbx.lan y kam2.pbx.lan :

```
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
	protocol C;
	startup {
		wfc-timeout  15;
		degr-wfc-timeout 60;
	}
	net {
		cram-hmac-alg sha1;
		shared-secret "Lhy12jkV";
	}
	on kam1.pbx.lan {
		device /dev/drbd0;
		disk /dev/sdb1;
		address 172.16.1.38:7788;
		meta-disk internal;
	}
	on kam2.pbx.lan {
		device /dev/drbd0;
		disk /dev/sdb1;
		address 172.16.1.138:7788;
		meta-disk internal;
	}
}
```

En freepbx-1.pbx.lan y freepbx-2.pbx.lan

```
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
	protocol C;
	startup {
		wfc-timeout  15;
		degr-wfc-timeout 60;
	}
	net {
		cram-hmac-alg sha1;
		shared-secret "Al12pf3D6";
	}
	on freepbx-1.pbx.lan {
		device /dev/drbd0;
```

```
        disk /dev/sdb1;
        address 172.16.1.42:7788;
        meta-disk internal;
    }
    on freepbx-2.pbx.lan {
        device /dev/drbd0;
        disk /dev/sdb1;
        address 172.16.1.142:7788;
        meta-disk internal;
    }
}
```

Finalizada la configuaración es el momento de inicializar el el dispositivo en todos los nodos.

```
root@kam1:~# drbdadm create-md r0
root@kam2:~# drbdadm create-md r0
root@freepbx-1:~# drbdadm create-md r0
root@freepbx-2:~# drbdadm create-md r0
```

Incializamos el servicio drbd.

```
root@kam1:~# /etc/init.d/drbd start
root@kam2:~# /etc/init.d/drbd start
root@freepbx-1:~#/etc/init.d/drbd start
root@freepbx-2:~# /etc/init.d/drbd start
```

El siguiente paso es indicar los nodos que serán los primarios incialmente.

```
root@kam1:~# drbdadm -- --overwrite-data-of-peer primary all
 root@freepbx-1:~# drbdadm -- --overwrite-data-of-peer primary all
```

En el siguiente paso daremos formato al nuevo dispostivo drbd creado.

```
root@kam1:~# mkfs.ext3 /dev/drbd0
root@freepbx-1:~# mkfs.ext3 /dev/drbd0ll
```

A continuación crearemos un punto de montaje en todos los nodos.

```
root@kam1:~# mkdir /replica
root@kam2:~#  mkdir /replica
root@freepbx-1:~# mkdir /replica
root@freepbx-2:~# mkdir /replica
```

Finalmente montaremos el dispositivo drbd en los puntos de montaje que acabamos de crear:

```
root@kam1:~# mount -t ext3 /dev/drbd0 /replica
root@freepbx-1:~# mount -t ext3 /dev/drbd0 /replica
```

Podemos verificar el estado en cada uno de los nodos de la siguiente manera:

```
root@freepbx-1:~# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r----
    ns:20572 nr:0 dw:96 dr:21290 al:8 bm:6 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
```

```
root@kam2:~# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r----
    ns:0 nr:248 dw:248 dr:0 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0
```

DRBD permite que solo uno de los dos nodos (el primario) tenga montado el dispositivo. De esta manera si quisiéramos darles el rol de primario por ejemplo al host kam2, en primer lugar deberíamos desmontar la partición del actual primario.

```
root@kam1:~# umount /replica
```

Luego deberiamos asignar el rol de secundario a este mismo host.

```
root@kam1:~# drbdadm secondary r0
```

El siguiente paso sería asignar el rol primario al nodo kam2.

```
root@kam2:~# drbdadm primary r0
```

Y finalmente montar la partición en el nuevo host primario.

```
root@kam2:~#  mount -t ext3 /dev/drbd0 /replica
```

Este proceso de cambio de roles puede automatizarse gracias a las herramientas que veremos a continuación.

4.6.2 Pacemaker-Corosync

Pacemaker es una solución open source licenciada bajo GNU/GPL v2, para la gestión de recursos clusters.
Pacemaker facilita la recuperación instantánea y de forma automática de los servicios proporcionados, asegurando que estos estarán siempre disponibles en alguno de los nodos que forman el clúster, y asegurando que estos servicios estarán corriendo únicamente en una localización evitando de esta forma posibles corrupciones en los datos.
Para que Pacemaker sea capaz de gestionar los recursos de esta forma, necesita poder comunicarse con el resto de nodos que forman el clúster, y es en esta comunicación donde Pacemaker necesita de otro componente, Corosync.

Corosync Cluster Engine distribuido bajo la licencia New BSD proporciona a Pacemaker una nueva capa que ofrece las siguientes funcionalidades:

- Un mecanismo para enviar de forma fiable mensajes entre los nodos miembros del clúster.
- Notificaciones cuando los nodos aparecen o desaparecen.
- Una lista de los nodos que están operativos para el clúster.

Corosync realiza las mismas y nuevas funcionalidades que proporciona Heartbeat, el cual también puede utilizarse como alternativa a Corosync junto con Pacemaker.
Sin embargo, desde que el proyecto Heartbeat ha quedado definitivamente en mantenimiento (no se desarrollarán nuevas funcionalidades) Corosync está siendo la solución mas implementada.
La siguientes figura nos muestra la arquitectura que proporicionan los componentes Pacemaker y Corosync.

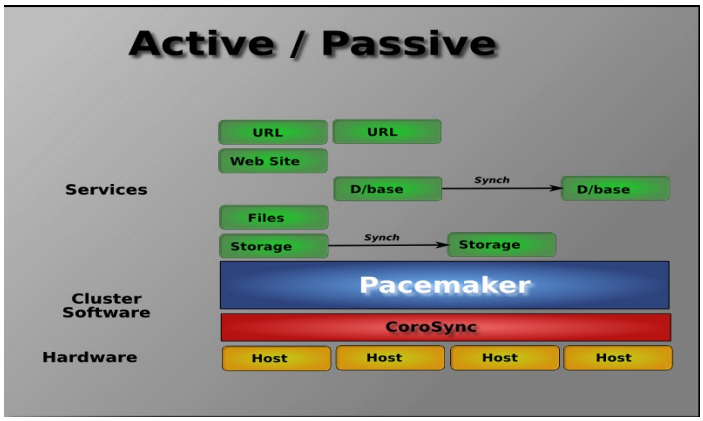


Figura 4.6 Arquitectura Pacemaker-Corosync

4.6.2.1 Instalación y Configuración Pacemaker-Corosync

El paquete pacemaker disponible en los repositorios de Debian instala como dependencia corosync, con lo que para instalar ambos componentes el proceso es sencillo, recordamos que la instalación debe realizarse en los 4 servidores que se describen en el esquema inicial (kam1, kam2, freepbx-1, freepbx-2).

```
root@kam1:~#  apt-get install pacemaker
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  cluster-agents cluster-glue corosync libcluster-glue libcorosync4 libcurl3 libesmtp5 libglib2.0-0 libglib2.0-
data
  libheartbeat2 libnspr4-0d libnss3-1d libopenhpi2 libopenipmi0 libssh2-1 libxslt1.1 openhpid shared-
mime-info
Se instalarán los siguientes paquetes NUEVOS:
  cluster-agents cluster-glue corosync libcluster-glue libcorosync4 libcurl3 libesmtp5 libglib2.0-0 libglib2.0-
data
  libheartbeat2 libnspr4-0d libnss3-1d libopenhpi2 libopenipmi0 libssh2-1 libxslt1.1 openhpid pacemaker
shared-mime-info
0 actualizados, 19 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 7824 kB de archivos.
Se utilizarán 25,0 MB de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]?
```

Una vez instalados los componentes necesarios el siguiente paso será adaptar la configuración a cada uno de los clusters definidos.

Antes de entrar en profundidad acerca del funcionamiento de Pacemaker, configuraremos Corosync, como hemos comentado este componente proporciona una capa de mensajería entre los nodos del clúster.

Para habilitar que Corosync arranque de forma automática al iniciar el sistema en primer lugar será necesario editar el fichero **/etc/defamar/corosync** y modificar el valor de la variable START.
Esta operación debe realizarse en los 4 nodos.

```
root@kam1:~#  cat /etc/default/corosync
# start corosync at boot [yes|no]
START=yes
```

El siguiente paso es configurar los nodos para que se comuniquen de forma segura en trel ellos. Si recordamos el esquema incial el proyecto consta de 4 nodos que forman 2 clústers que pese a interactuar conjuntamente se comportan de manera independiente.
Por un lado el clúster formado por los 2 nodos que actúan como PROXY SIP SERVERS y por otro, el clúster formado por los 2 nodos que actúan como PBX.

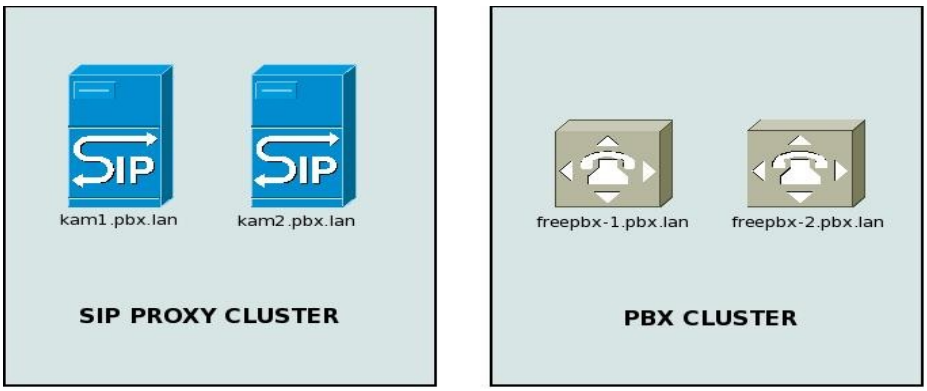


Figura 4.7 Clusters de la arquitectura

Para que Corosync pueda intercambiar mensajes cifrados que aseguren la autenticidad y la privacidad entre los dos nodos de cada clúster en primer lugar será necesario genera una clave privada.
Este proceso podría ser no necesario en un entrono privado como el que se utiliza en en este proyecto, donde se dispone de una vLAN exclusiva, o en entornos en los que ambos nodos estén conectados directamente NIC a NIC.

Para generar la clave privada corosync proporciona la siguiente herramienta que deberemos ejecutar en el nodo principal de cada clúster.

```
root@kam1:~#  corosync-keygen

root@freepbx-1:~#  corosync-keygen
```

Copiaremos la clave generada en cada uno de los nodos secundarios.

```
root@kam1:~# scp /etc/corosync/authkey kam2:/etc/corosync

root@freepbx-1:~# scp /etc/corosync/authkey freepbx-2:/etc/corosync
```

Asignamos los permisos necesarios para que la clave no pueda ser visible por cualquier usuario.

```
root@kam2:~# chown root:root /etc/corosync/authkey
root@kam2:~# chmod 400 /etc/corosync/authkey

root@freepbx-2:~# chown root:root /etc/corosync/authkey
root@freepbx-2:~# chmod 400 /etc/corosync/authkey
```

El fichero de configuración principal de Corosync es **/etc/corosync/corosync.conf**.
Las únicas variables que serán necesarias modificar para que Corosync sea operativo son las que afectan por un lado a la interfacey por el otro al identificador de nodo.
Debido a que ambos clusters compartirán la misma vLAN, las direcciones multicast donde anunciaran los mensajes deben ser diferentes.
Corosync genera gran cantidad de mensajes con lo que podría resultar conveniente conectar cada nodo directamente al otro a través de su interfaz o interfaces de red (Un bounding de 2 tarjetas daría más fiabilidad al sistema).
En este proyecto sin embargo se ha decidido utilizar una vLAN exclusiva para evitar un problema muy común en los clusters (split-brain) como veremos posteriormente en la configuración de Pacemaker.
En este caso las configuaraciones respectivas son las siguientes:

En kam1.pbx.lan:

```
nodeid: 1

        interface {
            # The following values need to be set based on your environment
            ringnumber: 0
            bindnetaddr: 172.16.1.0
            mcastaddr: 226.94.1.1
            mcastport: 5405
    }
```

En kam2.pbx.lan:

```
nodeid: 2

        interface {
            # The following values need to be set based on your environment
            ringnumber: 0
            bindnetaddr: 172.16.1.0
            mcastaddr: 226.94.1.1
            mcastport: 5405
    }
```

En freepbx-1.pbx.lan:

```
nodeid: 1

        interface {
            # The following values need to be set based on your environment
            ringnumber: 0
```

```
        bindnetaddr: 172.16.1.0
        mcastaddr: 226.94.1.2
        mcastport: 5405
    }
```

En freepbx-2.pbx.lan:

```
 nodeid: 2

        interface {
            # The following values need to be set based on your environment
            ringnumber: 0
            bindnetaddr: 172.16.1.0
            mcastaddr: 226.94.1.2
            mcastport: 5405
    }
```

De esta manera ya tendríamos los nodos miembros de cada clúster preparados, podemos verificar el estado del clúster con la utilidad **crm_mon** de Pacemaker.

```
============
Last updated: Sat May  5 13:04:25 2012
Stack: openais
Current DC: freepbx-1.pbx.lan - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ freepbx-1.pbx.lan freepbx-2.pbx.lan ]
```

El siguiente proceso para completar la configuración del clúster correspondería a la gestión de los recursos del clúster, es decir, que hacer por ejemplo cuando un nodo cae o cuando uno de los servicios falla.
Esta tarea como se ha descrito anteriormente corresponde a Pacemaker.

Pacemaker es también conocido por las iniciales **CRM** (Cluster Resource Manager), de hecho no es llamado por este nombre por el gran número términos que son abreviados con estas tres letras.
CRM implementa la configuración del clúster definida en un conjunto de de instrucciones codificadas en XML proporcionadas por los administradores en CIB(Cluster Information Base).
El CIB, que genera estos ficheros XML que serán interpretados por CRM para gestionar los recursos puede ser gestionado de tres formas diferentes:

- A través de una **GUI (hb_gui)**
- **Linea de comandos** ; método empleado en este proyecto
- **cibadmin** ; Permite consultar, modificar o eliminar la información de configuración del cluster a través de ficheros XML.

La linea de comandos (CLI) de CRM (o Pacemaker) proporciona una interfaz que facilita la gestión del CIB, por este motivo es el método escogido en este proyecto par a gestionar la configuración del clúster.

```
root@kam1:~# crm
crm(live)# help

This is the CRM command line interface program.

Available commands:

        cib             manage shadow CIBs
        resource        resources management
        node            nodes management
        options         user preferences
        configure       CRM cluster configuration
        ra              resource agents information center
        status          show cluster status
        quit,bye,exit   exit the program
        help            show help
        end,cd,up       go back one level

crm(live)#
```

Son varios los componentes que participan en la configuración de CRM como puede observarse en la salida correspondiente al comando anterior.
A modo de introducción detallaremos a continuación aquellos que afectaran a este proyecto.

- **node**: A través de esta opción podemos añadir, eliminar nodos al clúster. También podemos verificar el estado de los nodos o por ejemplo parar uno de ellos del clúster para hacer tareas de mantenimiento sobre el.

```
root@kam1:~# crm
crm(live)# node
crm(live)node# standby kam1.pbx.lan
crm(live)node# status
<nodes>
  <node id="kam2.pbx.lan" type="normal" uname="kam2.pbx.lan"/>
  <node id="kam1.pbx.lan" type="normal" uname="kam1.pbx.lan">
    <instance_attributes id="nodes-kam1.pbx.lan">
      <nvpair id="nodes-kam1.pbx.lan-standby" name="standby" value="on"/>
    </instance_attributes>
  </node>
</nodes>
crm(live)node#
```

- **resource**: Esta opción permite gestionar los recursos del cluster (servicios, ip's virutales ...), verificar su estado, detenerlos, eliminarlos ...

```
crm(live)# resource
crm(live)resource# status
 Master/Slave Set: drbd_ms
     Masters: [ kam2.pbx.lan ]
     Stopped: [ drbd:1 ]
 Resource Group: sip_proxy_group
```

```
    drbd_fs        (ocf::heartbeat:Filesystem) Started
    ClusterIP-192          (ocf::heartbeat:IPaddr2) Started
    ClusterIP-172          (ocf::heartbeat:IPaddr2) Started
    mysqld         (lsb:mysql) Started
    Kamailio       (lsb:kamailio) Started
 pingd   (ocf::pacemaker:pingd) Started
crm(live)resource#
```

- **configure**: La opción “configure” nos permite definir el comportamiento tanto de los nodos, como de los recursos así como de las restricciones y propiedades del clúster.

Esta última sección (configure) será por tanto la que nos permitirá definir la lógica del comportamiento del clúster. A continuación veremos la configuración adaptada al esquema de este proyecto tratando de explicar la función de cada componente.

La configuración correspondiente al clúster PROXY SIP:

```
crm(live)configure# show
node kam1.pbx.lan \
        attributes standby="off"
node kam2.pbx.lan
primitive ClusterIP-192 ocf:heartbeat:IPaddr2 \
        params ip="192.168.1.238" cidr_netmask="22" \
        op monitor interval="4s" \
        meta is-managed="true"
primitive Kamailio lsb:kamailio \
        op start interval="0" timeout="30s" \
        op stop interval="0" timeout="30s" \
        op monitor interval="20s"
primitive drbd ocf:linbit:drbd \
        params drbd_resource="r0" \
        op monitor interval="20s"
primitive drbd_fs ocf:heartbeat:Filesystem \
        params device="/dev/drbd0" directory="/replica" fstype="ext3"
primitive mysqld lsb:mysql \
        op start interval="0" timeout="30s" \
        op stop interval="0" timeout="30s" \
        op monitor interval="20s"
primitive pingd ocf:pacemaker:pingd \
        params host_list="192.168.1.35 172.16.1.35" multiplier="100" \
        op monitor interval="15s" timeout="5s" \
        meta target-role="Started"
group sip_proxy_group drbd_fs ClusterIP-192 mysqld Kamailio \
        meta target-role="Started"
ms drbd_ms drbd \
        meta master-max="1" master-node-max="1" clone-max="2" clone-node-max="1" notify="true"
clone pingdclone pingd \
        meta globally-unique="true"
location sip_proxy_group_with_ping sip_proxy_group \
        rule $id="sip_proxy_group_with_ping-rule" -inf: not_defined pingd or pingd lte 100
colocation fs_on_drbd inf: sip_proxy_group drbd_ms:Master
order fs_after_drbd inf: drbd_ms:promote sip_proxy_group:start
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
```

```
        cluster-infrastructure="openais" \
        expected-quorum-votes="2" \
        stonith-enabled="false" \
        no-quorum-policy="ignore" \
        default-resource-stickiness="100"
```

Empezaremos comentando la sección correspondiente a las propiedades del clúster. Al iniciar el servicio de Corosync con ambos nodos ya configurados, las propiedades del clúster que se muestran son establecidas por defecto, en función del número de nodos. Algunas de estas propiedades han sido modificadas para que el comportamiento del clúster fuera el deseado. En concreto estas propiedades son:

- **no-quorum-policy="ignore" ** : Puesto que el clúster está formado únicamente por 2 nodos llegar a un consenso para determinar que nodo/s falla será imposible. Por tanto no tiene ningún significado establecer esta propiedad.

- **stonith-enabled=”false”** : La propiedad stonith (Shot The Other Node In The Head) está relacionado con el Fencing (técnica para aislar del clúster a un nodo conflictivo). Stonith propicia el apagado de dicho nodo conflictivo. Esta propiedad suele emplearse en clúster con almacenamiento compartido, donde fallos constantes podrían producir incoherencia en los datos.
  En el esquema de este proyecto no es una propiedad imprescindible, puesto que el acceso para escritura sera mínimo.

- **Default-resource-stickiness=”100”**: Mediante esta propiedad definimos que en caso de fallida y posterior recuperación del nodo principal, los servicios se mantengan en el secundario, evitando así movimientos de rol innecesarios.

La configuración referente a las propiedades del clúster seria finalmente la siguiente:

```
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="openais" \
        expected-quorum-votes="2" \
        stonith-enabled="false" \
        no-quorum-policy="ignore" \
        default-resource-stickiness="100"
```

La configuración referente a los nodos miembros del clúster es establecida al finalizar la configuración de Corosync como pudimos ver anteriormente. El atributo standby=”on” indica que el nodo esta en “espera” fuera del clúster para hacer por ejemplo tareas de mantenimiento sobre este.

```
node kam1.pbx.lan \
        attributes standby="on"
node kam2.pbx.lan
```

Como vimos anteriormente CRM llama a “resource” a los servicios que este deberá gestionar. Estos recursos son definidos mediante la etiqueta “primitive” seguida de una serie de atributos que definirán el comportamiento de CRM sobre dicho recurso. A continuación detallaremos los recursos definidos para el clúster PROXY-SIP y su comportamiento.

El primer recurso es la definición de una IP de failover. Como vimos en la sección “Servicios Adicionales”, los terminales IP son configurados para registrarse contra esta dirección. Estableciendo esta dirección IP para el clúster en caso de caída del nodo principal dicha dirección IP será asignada al nodo secundario, de esta forma los terminales podrán continuar funcionando.

```
primitive ClusterIP-192 ocf:heartbeat:IPaddr2 \
        params ip="192.168.1.238" cidr_netmask="22" \
        op monitor interval="4s" \
        meta is-managed="true"
```

Los recursos que se muestran a continuación corresponden a los servicios propiamente dicho, que CRM deberá gestionar. En concreto el servicio de Kamailio y y MySQL.
Establecemos un intervalo de monitorización de 20 segundos para verificar el estado de cada uno de los servicio.
En el caso que el servicio se detuviera por algún error CRM trataría de rearmarlo, transcurridos 30 segundos si no consiguiera hacerlo moveria los servicios al nodo secundario.

```
primitive Kamailio lsb:kamailio \
        op start interval="0" timeout="30s" \
        op stop interval="0" timeout="30s" \
        op monitor interval="20s"
primitive mysqld lsb:mysql \
        op start interval="0" timeout="30s" \
        op stop interval="0" timeout="30s" \
        op monitor interval="20s"
```

Para que los servicios tuvieran el mismo comportamiento en ambos nodos es importante que tanto la configuración como los datos que pudieran necesitar fueran los mismos.
Aprovechando la replicación de datos que DRBD proporciona podemos hacer que ambos nodos accedan a los mismos datos y a los mismos ficheros de configuración para que de esta forma los servicios se comporten de la misma manera.
De esta manera era necesario mover tanto los archivos de configuración de Kamailio como los datos de MySQL a la partición común, el proceso realizado fue el siguiente:

```
root@kam1:~# /etc/init.d/kamailio stop
root@kam1:~# /etc/init.d/mysql stop
root@kam1:~# mv /usr/local/etc/kamailio /replica/
root@kam1:~# mv /var/lib/mysql /replica/
root@kam1:~# ln -sn /replica/kamailio /usr/local/etc/kamailio
root@kam1:~# ln -sn /replica/mysql /var/lib/mysql

root@kam2:~# /etc/init.d/kamailio stop
root@kam2:~# /etc/init.d/mysql stop
root@kam2:~# rm -rf /usr/local/etc/kamailio
root@kam2:~# rm -rf /var/lib/mysql
root@kam2:~# ln -sn /replica/kamailio /usr/local/etc/kamailio
root@kam2:~# ln -sn /replica/mysql /var/lib/mysql
```

Continuando la configuración de los recursos de CRM, uno de los principales servicios que este debía gestionar era el de DRBD. La siguiente configuración nos sirve por controlar del servicio drbd como el punto de montaje que definimos durante la instalación de DRBD. La opción “ms” permite definir el comportamiento master-slave de un recurso concreto,en este caso del recurso drbd. Con esta opción indicamos que al sistema que establezca el rol de máster y slave para cada nodo.

```
primitive drbd ocf:linbit:drbd \
        params drbd_resource="r0" \
        op monitor interval="20s"
primitive drbd_fs ocf:heartbeat:Filesystem \
        params device="/dev/drbd0" directory="/replica" fstype="ext3"
ms drbd_ms drbd \
        meta master-max="1" master-node-max="1" clone-max="2" clone-node-max="1" notify="true"
```

El siguiente recurso, corresponde a la monitorización a través de paquetes icmp de un IP concreta. En este caso las direcciones corresponde a un mismo switch de nivel 3, en cada una de las 2 vLAN a las que los nodos tienen acceso.
De esta forma podemos verificar el estado de las NIC de cada nodo en todo momento, y resolver un problema muy común en este tipo de clústers, split-brain.
El split-brain se produce cuando cada uno de los nodos miembros del clúster cree que el otro ha caído. Monitorizando un equipo en alta disponibilidad como es este switch evitamos que este problema se produzca.
El valor del atributo multiplicador indica señala la puntuación que recibe cada nodo por alcanzar (via ping) cada uno de las ip's indicadas.
Así en el supuesto de que falle la NIC de uno de los nodos el otro conseguirá llegar a los 2 destinos indicados en la “host_list”, obteniendo una puntuación de 200 frente a la puntuación 100 que obtendría el nodo al cual le falla una de las NIC.
La constraint “clone” sirve para indicar que dicho recurso se ejectue en los dos nodos tanto en el master como en el slave.

```
primitive pingd ocf:pacemaker:pingd \
        params host_list="192.168.1.35 172.16.1.35" multiplier="100" \
        op monitor interval="20s" timeout="5s" \
        meta target-role="Started"
clone pingdclone pingd \
        meta globally-unique="true"
```

La propiedad “group” sirve para tratar un grupo de recursos como si se tratara de un único recurso. De esta forma se consigue que cuando uno de los servicios falle todos los servicios definidos en dicho grupo sean movidos al nodo secundario.

```
group sip_proxy_group drbd_fs ClusterIP-192 mysqld Kamailio \
        meta target-role="Started"
```

La constraint “location” definida sirve para mover los servicios al nodo secundario en caso de que no se cumpla el recurso ping definido anteriormente.

```
location sip_proxy_group_with_ping sip_proxy_group \
        rule $id="sip_proxy_group_with_ping-rule" -inf: not_defined pingd or pingd lte 100
```

La constraint “colocation” se utiliza para definir que recursos deberían correr en el mismo nodo. En este caso indicamos que el sistema de ficheros DRBD y el grupo de

servicios deben estar en el mismo nodo.

```
colocation fs_on_drbd inf: sip_proxy_group drbd_ms:Master
```

La última de las constraints definidas es “order”, esta como su nombre indica se utiliza para definir el orden en que los recursos deben ser inciados en caso de existir alguna dependencia, como en este caso en el que los servicios Kamailio y MySQL deberán leer la información guardada en el dispositivo DRBD.

```
order fs_after_drbd inf: drbd_ms:promote sip_proxy_group:start
```

Por último mostraremos la configuración correspondiente al clúster formado por los servidores PBX.
En un principio no se considero necesaria la configuración de los servicios de CRM, hablar tema AUTH kamailio...

Como puede observarse la configuración es similar a la establecida en el anterior clúster, las únicas diferencias corresponden al servicio NFS.

```
crm(live)configure# show
node freepbx-1.pbx.lan \
        attributes standby="off"
node freepbx-2.pbx.lan \
        attributes standby="off"
primitive ClusterIP-172 ocf:heartbeat:IPaddr2 \
        params ip="172.16.1.242" cidr_netmask="24" \
        op monitor interval="4s" \
        meta is-managed="true" target-role="Started"
primitive ClusterIP-192 ocf:heartbeat:IPaddr2 \
        params ip="192.168.1.242" cidr_netmask="22" \
        op monitor interval="4s" \
        meta is-managed="true"
primitive drbd ocf:linbit:drbd \
        params drbd_resource="r0" \
        op monitor interval="60s"
primitive drbd_fs ocf:heartbeat:Filesystem \
        params device="/dev/drbd0" directory="/replica" fstype="ext3"
primitive nfs-kernel-server lsb:nfs-kernel-server
primitive pingd ocf:pacemaker:pingd \
        params host_list="192.168.1.35 172.16.1.35" multiplier="100" \
        op monitor interval="15s" timeout="5s"
group pbx_group drbd_fs ClusterIP-192 ClusterIP-172 nfs-kernel-server \
        meta target-role="Started"
ms drbd_ms drbd \
        meta master-max="1" master-node-max="1" clone-max="2" clone-node-max="1" notify="true"
clone pingdclone pingd \
        meta globally-unique="true"
location pbx_group_with_ping pbx_group \
        rule $id="sip_proxy_group_with_ping-rule" -inf: not_defined pingd or pingd lte 100
colocation fs_on_drbd inf: pbx_group drbd_ms:Master
order drbd_fs-before-nfs-kernel-server inf: drbd_fs:start nfs-kernel-server:start
order fs_after_drbd inf: drbd_ms:promote pbx_group:start
property $id="cib-bootstrap-options" \
        dc-version="1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b" \
        cluster-infrastructure="openais" \
```

```
        expected-quorum-votes="2" \
        stonith-enabled="false" \
        no-quorum-policy="ignore" \
        default-resource-stickiness="100"
```

5 TESTING

En esta fase de testing se ha querido determinar por un lado la capacidad de rendimiento de la solución planteada y por otro la tolerancia a fallos que presenta el sistema de alta disponibilidad frente a los problemas más comunes.

Pruebas Alta disponibilidad

Los servidores Kamailio (kam1.pbx.lan y kam2.pbx.lan) forman un clúster activo-pasivo en el que uno de los nodos (generalmente kam1) actúa como máster y el nodo secundario tomará el control de los servicios solo en caso de fallo del nodo principal. A continuación detallaremos el comportamiento del clúster frente a los fallos más comunes que podrían producirse:

1.- Caída del nodo principal

Verificación previas; Como vimos anteriormente la utilidad *crm_mon* sirve para monitorizar el estado del clúster en todo momento. Podemos ver que ambos nodos están “Online” y que el rol master es actualmente de kam1.pbx.lan

```
============
Last updated: Sun May 6 13:55:46 2012
Stack: openais
Current DC: kam1.pbx.lan - partition with quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, 2 expected votes
4 Resources configured.
============

Online: [ kam2.pbx.lan kam1.pbx.lan ]

 Master/Slave Set: drbd_ms
     Masters: [ kam1.pbx.lan ]
     Slaves: [ kam2.pbx.lan ]
 Resource Group: sip_proxy_group
     drbd_fs    (ocf::heartbeat:Filesystem):    Started kam1.pbx.lan
     ClusterIP-192      (ocf::heartbeat:IPaddr2):       Started kam1.pbx.lan
     mysqld     (lsb:mysql):    Started kam1.pbx.lan
     Kamailio   (lsb:kamailio): Started kam1.pbx.lan
 Clone Set: pingdclone (unique)
     pingd:0    (ocf::pacemaker:pingd): Started kam1.pbx.lan
     pingd:1    (ocf::pacemaker:pingd): Started kam2.pbx.lan
```

Al provocar un paro repentino en el servidor kam1.pbx.lan vemos como el rol de máster y el control de los servicios son movidos al nodo kam2.pbx.lan, en un tiempo

de aproximado de 21 segundos el servicio vuelve a estar operativo. Las únicas llamadas afectadas son las que intentaran producirse en ese intervalo de tiempo. Puesto que las llamadas en curso ya son gestionadas a través de Asterisk.

```
============
Last updated: Sun May 6 14:07:02 2012
Stack: openais
Current DC: kam2.pbx.lan - partition WITHOUT quorum
Version: 1.0.9-74392a28b7f31d7ddc86689598bd23114f58978b
2 Nodes configured, 2 expected votes
4 Resources configured.
============

Online: [ kam2.pbx.lan ]
OFFLINE: [ kam1.pbx.lan ]

 Master/Slave Set: drbd_ms
     Masters: [ kam2.pbx.lan ]
     Stopped: [ drbd:0 ]
 Resource Group: sip_proxy_group
     drbd_fs    (ocf::heartbeat:Filesystem):    Started kam2.pbx.lan
     ClusterIP-192      (ocf::heartbeat:IPaddr2):       Started kam2.pbx.lan
     mysqld     (lsb:mysql):    Started kam2.pbx.lan
     Kamailio   (lsb:kamailio): Started kam2.pbx.lan
 Clone Set: pingdclone (unique)
     pingd:0    (ocf::pacemaker:pingd): Stopped
     pingd:1    (ocf::pacemaker:pingd): Started kam2.pbx.lan
```

2.- Fallo de una de las interfaces de red del nodo principal.

Como detallamos al inicio del proyecto cada servidor constará de 4 tarjetas formando un bounding para cada una de las dos vLANs definidas, por lo que la probabilidad de que se produzca este error es minima. Gracias al recurso de monitorizacion “ping” configurado en CRM podemos verificar el estado de cada uno de los bounding. El comportamiento frente a uno fallo de este tipo es idéntico al producido en el fallo anterior. Para simular dicho fallo podemos agregar una regla iptables como la siguiente.

```
root@kam2:~# iptables -A OUTPUT -p icmp  -j DROP
```

3.- Fallo uno de los servidores Asterisk.

La configuración de Kamailio se encarga de distribuir las llamadas entre los nodos Asterisk, al detener el servicio de Asterisk en uno de los nodos. El módulo dispatcher se encargará de “deshabilitar” el nodo fallido transcurridos tantos intentos fallidos y tiempo de inactividad como se definimos en la configuración del módulo. Las llamadas producidas en ese intervalo con destino el nodo con el servicio de Asterisk detenido no podrán efectuarse. Después de que Kamailio marque el nodo no activo las peticiones ya no serán redirigidas a este.
El comando kamctl puede darnos información sobre el estado de los nodos definidos en el fichero dispatcher.list:

```
root@kam1:~#  kamctl 'fifo' ds_list
```

```
SET_NO:: 1
SET:: 1
        URI:: sip:192.168.1.142:5060 flag=A priority=0 attrs=
        URI:: sip:192.168.1.42:5060 flag=P priority=0 attrs=
```

Durante la instalación de el clúster Asterisk formado por freepbx-1.pbx.lan y freepbx-2.pbx.lan pudimos ver que en el diseño de la arquitectura los servidores formaban un cluser activo-activo gracias a aspectos como la replicación master-master formada por los servidores MySQL.
El único punto en común es como también pudimos ver, el recurso NFS, al que ambos servidores acceden para compartir datos como buzones de voz o grabaciones
Con la configuración de CRM vimos como en el caso de fallida de algún nodo este recurso era desplazado al nodo secundario.
Ante las mismas pruebas realizadas sobre el cluster Asterisk, este aseguraba de la misma forma la funcionalidad del sistema.

Pruebas de rendimiento

Para la realización de las pruebas de rendimiento de los servidores, en cuanto a la capacidad para procesar llamadas de cada uno de estos, se ha utilizado la herramienta SIPp un generador de tráfico SIP open source.

Para instalar SIPp simplemente es necesario obtener el código fuente e instalar ciertas dependencias antes de proceder a su instalación. En este caso no se ha instalado SIPp en ninguno de los servidores descritos hasta ahora sino en nuevo servidor.

```
root@pbx1:/usr/src/voip# wget http://downloads.sourceforge.net/project/sipp/sipp/3.2/sipp.svn.tar.gz
root@pbx1:/usr/src/voip# apt-get install gsl-bin libpcap-dev libncurses5-dev
root@pbx1:/usr/src/voip# tar -zxvf sipp.svn.tar.gz
root@pbx1:/usr/src/voip# cd sipp.svn/
root@pbx1:/usr/src/voip/sipp.svn# make pcapplay_ossl
```

Es necesario crear un nuevo usuario en la tabla sipusers esto puede hacerse en cualquiera de los 2 nodos Asterisk gracias al clúster MySQL.

```
INSERT INTO `sipusers` (`name`, `host`, `nat`, `type`, `accountcode`, `amaflags`, `call-limit`, `callgroup`,
`callerid`, `cancallforward`, `canreinvite`, `context`, `disallow`, `allow`, `port) VALUES
('sipp', 'dynamic', 'no', 'friend', 'yes', 'no', 'sipp',  'all', 'alaw,ulaw', 6000);
```

Necesitaremos también crear un contexto nuevo para tratar las llamadas realizadas durante la prueba, para ellos editaremos tanto en los 2 nodos Asterisk el fichero **/etc/asterisk/extensions_custom.conf**

```
[sipp]
exten => 9999,1,Answer()
        same => n,Set(CHANNEL(musicclass)=default)
        same => n,WaitMusicOnHold(40)
        same => n,Hangup()
```

Y aplicar los cambios en el dialplan:

```
freepbx-2*CLI> dialplan reload
```

```
freepbx-1*CLI> dialplan reload
```

Definidos tanto el usuario como el nuevo contexto ya pueden ejecutarse las pruebas desde el host donde se instalo SIPp. Son varias los modos de ejecución que pueden realizarse con SIPp, dependiendo del tipo de mensaje SIP o combinación que se utilicen (OPTIONS, REGISTER, INVITE...).
SIPp utiliza ficheros en formato XML para definir distintos escenarios de prueba.

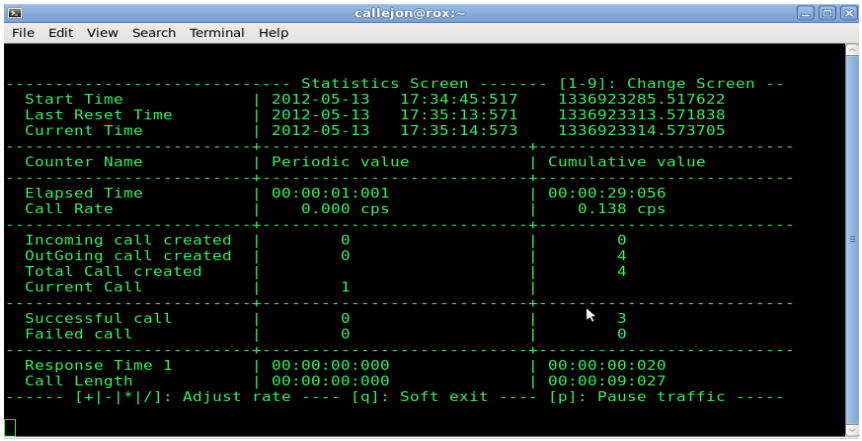
```
root@pbx1:/usr/src/voip# ./sipp -sn uac_pcap -d 30000 -s 9999 freepbx-1 -l 50 -mp 6000 -r 5 -t un -i
192.168.1.22
```

En este caso la prueba se ha realizado con las siguientes opciones:

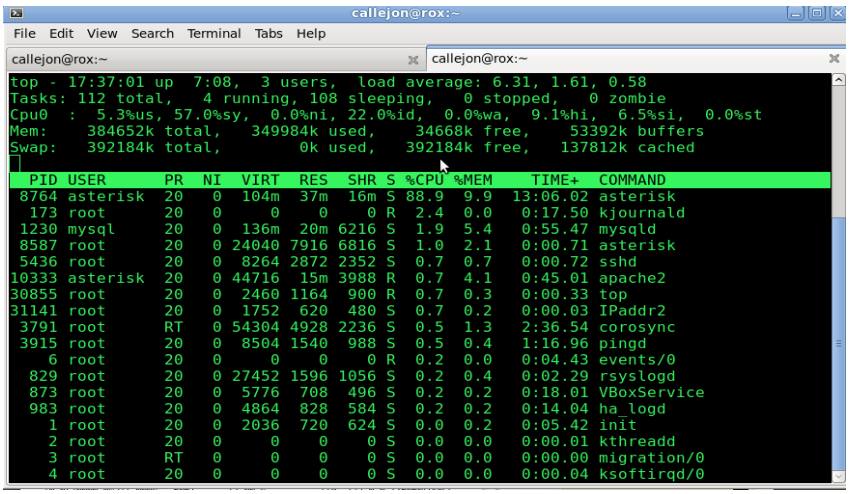
- -d: Duración de cada llamada 30 segundos (en ms)
- -s: Número (9999) en el nodo (freepbx-1) al que se llama (definido en el contexto anterior)
- -l: Cantidad máxima de llamadas simultaneas (50).
- -mp: Puerto de escucha local para el intercambio de dialogo SIP.
- -r: Ratio de llamadas, 5 llamadas nuevas cada segundo
- -t un: Indica que se utilice un único socket UDP por llamada
- -i: Establece la IP origen en las cabeceras de los mensajes (Contact, Via...)

El escenario de esta prueba se realiza íntegramente con los códecs G.711 (Alaw/Ulaw) con lo que si se requiere realizar otras comprobaciones será necesario definir un escenario personalizado.
Estas pruebas son meramente orientativas puesto que se han realizado sobre una plataforma virtual en un laptop con procesador Intel-Core i3 2.10GHz y 8GB de RAM, por lo que los resultados obtenidos no son concluyentes.

Al ejecutar el comando SIPp proporciona una serie de ventanas entre las que se puede desplazar y que muestra información como las estadísticas de la prueba.



De forma paralela podemos utilizar alguna herramienta como TOP, para comprobar el rendimiento del sistema frente la prueba de estress que está sufriendo.



6 CONCLUSIONES

Cuando se inicio este proyecto se plantearon principalmente dos objetivos.
En primer lugar, la posibilidad de profundizar en el conocimiento sobre el mundo de la VoIP a partir de un proyecto real. En segundo lugar, la necesidad de ofrecer al cliente una solución de VoIP que fuera eficiente y fiable.

Una vez finalizado el proyecto se pueden considerar como logrados ambos objetivos. Desde el punto de vista de cliente la solución que aquí se ha diseñado supone la garantía de ofrecer un sistema completamente fiable y eficiente, gracias a los dos ejes de este proyecto, como son la alta disponibilidad y la distribución de carga y que cumple con todos los requisitos especificados por este.

Desde el punto de vista personal el desarrollo de este proyecto ha servido por una parte para poder ampliar conocimientos relacionados con Asterisk.
Ya desde el inicio de este proyecto que se destino principalmente a la formación, con la lectura principalmente del libro *Asterisk The Definitve Guide,* y que ha supuesto una enorme ayuda durante todo el transcurso.
Por otra parte cuando se planteo este proyecto y se planteó la posibilidad de implementar un sistema en el que además de alta disponibilidad existiera la distribución de carga, se hizo tras haber escuchado y leído acerca de las posibilidades que podía ofrecer Kamailio, sin tener ningún conocimiento previo sobre este y que gracias al desarrollo del proyecto se ha podido conocer al detalle.

Las dificultades que han aparecido durante la fase de implementación del proyecto no han sido pocas, provocadas principalmente por el desconocimiento previo de algunas materias, sin embargo la aparición de cada una de estas ha supuesto una nueva oportunidad en la adquisición de conceptos nuevos.

Por último destacar que, con la distribución de este documento bajo la licencia libre, y el uso exclusivo de software libre, se pretende contribuir con la comunidad compartiendo los conocimientos adquiridos durante el transcurrso de este proyecto.

Webgrafía:

- http://www.asterisk.org
- http://www.freepbx.org/
- http://www.asteriskdocs.org/
- http://groups.google.com/group/asterisk-es
- http://www.voip-info.org
- http://www.kamailio.org/dokuwiki/doku.php/core-cookbook:3.1.x
- http://www.asipto.com/pub/kamailio-devel-guide/
- http://lists.sip-router.org
- http://dev.mysql.com/doc/refman/5.0/es/replication-howto.html
- http://www.drbd.org/
- http://www.clusterlabs.org/
- http://www.linux-ha.org/wiki/Main_Page
- http://sipp.sourceforge.net/

7 ANEXO I – Creative Commons BY-NC-SA License

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.
d. **"Original Author"** means the individual or entity who created the Work.
e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.
f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
g. **"License Elements"** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to create and reproduce Derivative Works;
c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;
d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(d), as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any credit as required by clause 4(d), as requested.
b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.5 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.
c. You may not exercise any of the rights granted to You in Section 3 above in any manner that

is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.
e. For the avoidance of doubt, where the Work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation.
f. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at http://creativecommons.org/.