Tesina de Grado para la obtención del título de
"Ingeniería en Sistemas"

"REDISEÑO E IMPLEMENTACIÓN DE MEJORAS DE DISPONIBILIDAD Y ESCALABILIDAD A UNA CENTRAL TELEFÓNICA VOIP"

Presentada por

WALS OCHOA, Lucas Germán
LABAYÉN, Franco Sahid

Dirigida por

Ing. CRESPO, Aldo Abel

General Pico, La Pampa, Argentina
2016

Agradecimientos

La realización del presente trabajo significo un gran esfuerzo personal que hubiera sido muy difícil de superar sin el apoyo de personas que estuvieron presentes a lo largo de este recorrido. Espero encontrar las palabras correctas para expresar mi gratitud.

Agradezco a mi director de tesis, Ing. Crespo Abel, por darme la oportunidad de participar en este trabajo y guiarme por el camino correcto para alcanzar satisfactoriamente mi meta. Los conocimientos y experiencias compartidas me hacen hoy mejor persona y profesional, gracias por su entrega al proyecto, sus consejos y críticas siempre bien fundadas.

A mi compañero de tesis, Wals Lucas, que desde el primero momento tomó la decisión de enfrentar este camino juntos. Su predisposición y entusiasmo son los que siempre nos permiten ver un futuro mejor y seguir adelante sin importar la cantidad de obstáculos que debamos enfrentar.

A mis progenitores quienes me motivan constantemente para alcanzar mis anhelos y me enseñaron lo importante que es el saber.

Finalmente, a mi pareja quien me acompaño casi a lo largo de toda mi carrera universitaria, supo comprenderme y me brindó su apoyo de manera incondicional.

Labayén, Franco Sahid

Agradecimientos

En primer lugar, quiero agradecerle al Ingeniero Abel Crespo, por su apoyo y consejos constantes, no solo en el ámbito profesional sino también respecto a experiencias de vida. También agradecerle por ofrecerme y recomendarme realizar mi trabajo de tesis en el área de la telefonía VoIP.

A mis padres, que a pesar de las diferencias que podamos tener, siempre velaron por mi bienestar y felicidad, me enseñaron lo importante que son los estudios y me dieron la posibilidad de realizar un estudio universitario.

A mi compañero de tesis, Franco Labayén, por querer realizar otro proyecto más conmigo, un proyecto mucho más grande que cualquiera que hayamos hecho anteriormente, y en un campo en el cual ambos empezábamos de cero. Espero que podamos llevar esta sociedad a más y mayores proyectos en el futuro.

Al Ingeniero Ricardo Furch, por darme la posibilidad de tener mi primera experiencia en el ámbito laboral. También agradecerle por la ayuda brindada en diferentes momentos de la realización del proyecto y los consejos en cuanto a mi futuro.

Por último, quiero agradecer a esa persona especial que me enseño lo importante que es el esfuerzo, que ante cualquier caída siempre hay que levantarse y seguir luchando por los objetivos y sueños que uno anhela, ya que la única persona que puede decir que uno está equivocado, es uno mismo.

Lucas Germán Wals Ochoa

Resumen

En la actualidad, los servicios orientados a las redes convergieron hacia una red de propósito general o "Red de Servicios Convergentes". Ello significa que ya no se requiere de una red para cada servicio y la infraestructura de Internet soporta los servicios de televisión, radiofonía, telefonía e Internet. En el caso de los sistemas telefónicos, es un hecho irreversible que VoIP (voz sobre el protocolo de Internet) se impuso sobre los sistemas tradicionales de telefonía.

Son cada vez menos las empresas telefónicas que utilizan costosas centrales de conmutación analógicas propietarias, cuya ampliación y/o actualización requiere de elevadas erogaciones económicas. Los sistemas operativos de licencia gratuita tal como aquellos basados en GNU/Linux, de software de código abierto y las comunidades de usuarios que apoyan estas iniciativas, permitieron el desarrollo de software para la implementación de centrales telefónicas VoIP de bajo costo y han revolucionado al sector de las grandes empresas productoras de tecnologías para la implementación de las redes telefónicas.

La principal ventaja que VoIP brinda con respecto al servicio tradicional de telefonía, es sin lugar a dudas el menor costo (medido en unidades de abonados) y la simplicidad a la hora de la implementación. Adicionalmente las comunicaciones telefónicas de larga distancia no difieren en costos con respecto a las llamadas locales. Las empresas que ofrecen servicios telefónicos basados en VoIP, a su vez son proveedores de servicios de Internet, y ello les posibilita una gran flexibilidad respecto de opciones en lo referido a planes para atraer nuevos usuarios y ser competitivos en un mercado muy exigente. La tecnología VoIP se caracteriza por ofrecer una gran variedad de funcionalidades que son costosas y/o difíciles de implementar en los sistemas de telefonía convencionales. Algunos ejemplos al respecto son servicios de: identificación de llamadas, desvío de llamadas, correo de voz, conferencia entre tres o más usuarios, etc.

En nuestro país, a partir de la desregulación de los servicios de telefonía e Internet, diversos sectores, predominantemente del ámbito cooperativista, vislumbraron la posibilidad de prestar servicios de telefonía e Internet. El interés se transformó rápidamente en realidad y fue la apertura de oportunidades para el desarrollo de centrales telefónicas. En el marco descrito, el área de Redes y Comunicaciones de la Facultad de Ingeniería decidió investigar y generar conocimiento en el tópico VoIP, y como resultado directo se plantearon las tesinas de grado referenciadas como (2) y (3), y un proyecto API (Áreas Prioritarias de Investigación) que fue seleccionado como de interés, por la Secretaría de Ciencia y Técnica de la UNLPam en conjunto con el Ministerio de la Producción de la Provincia de La Pampa, entidad que lo financió.

Todo lo descrito en el párrafo previo, fue vital para incursionar en el diseño e implementación de centrales telefónicas VoIP. Se avanzó desde una perspectiva de crecimiento paulatino respecto al nivel de complejidad, hacia un objetivo final muy ambicioso, que contempló el diseño e implementación de una central telefónica, de alta disponibilidad en todos sus componentes (software y hardware), que pudiera escalar fácilmente respecto del número de abonados a servir, con un alto desempeño en lo que

se refiere a consumo de recursos tales como, la utilización de memoria RAM y el porcentaje de uso de núcleos de procesamiento. En el marco señalado, este trabajo final es una continuación de los trabajos previos en la prosecución del objetivo final, por lo tanto se lo debe interpretar como un "rediseño". Por "rediseño" (respecto de los trabajos previos) se entiende, la actualización de diversos módulos de software que componen la central telefónica, el remplazo de componentes obsoletos, la introducción de nuevos módulos destinados a fortalecer el desempeño y proveer condiciones de alta disponibilidad en los servicios que se brindan a los usuarios finales.

Es importante destacar que una implementación de (3) fue presentada en la Cooperativa de Obras y Servicios Públicos de Rio Tercero (provincia de Córdoba) y comparada con una solución propietaria en un ambiente de producción. En esa oportunidad se entabló un dialogo constructivo con los ingenieros responsables del sector telecomunicaciones de la cooperativa, que derivó en sugerencias concretas sobre puntos a fortalecer en la implementación. El intercambio de puntos de vista sobre componentes de tecnologías de software resultó extremadamente útil y fue el punto de partida para el "rediseño" de la central telefónica VoIP.

Este trabajo está estructurado en cinco capítulos. En el Capítulo 1 se introducen aspectos generales sobre telefonía, posteriormente se realiza un análisis general de los trabajos (2) y (3), luego; considerando el análisis previo, se argumentará y justificará sobre aspectos puntuales de la arquitectura que deben ser rediseñados o remplazados, y finalmente se presentaran los objetivos finales del trabajo. En el Capítulo 2 presentará los conceptos teóricos sobre Sistemas Distribuidos utilizados para la creación de la central telefónica VoIP. En el Capítulo 3 se abordarán conceptos teóricos sobre los protocolos utilizados en la implementación de la central telefónica. En el Capítulo 4 se describirán los componentes de software seleccionados para el desarrollo de la central telefónica, y se explicaran distintas funcionalidades que responden a distintas configuraciones. Luego, en el Capítulo 5, se presentará el diseño de la central telefónica y se implementará un caso práctico que será sometido a pruebas de disponibilidad y rendimiento. Finalmente, en el Capítulo 6 se escribirán las conclusiones de este trabajo.

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1
"Introducción y Objetivos"

1.0 Introducción

Desde la creación del teléfono por parte de Antonio Meucci (aproximadamente en 1854), las redes telefónicas evolucionaron constantemente con el propósito de atender la demanda creciente de nuevos usuarios que requerían del servicio de voz. La infraestructura de red utilizada para cursar las comunicaciones telefónicas se soportaba sobre redes dedicadas al tráfico de voz y bajo estas circunstancias evolucionaron las redes orientadas a la conexión, capaces de garantizar las restricciones que imponía el tráfico de voz caracterizado como de tiempo real. En redes orientadas a la conexión, para cursar una llamada de voz, previamente se debe establecer un circuito entre el abonado originador y el abonado receptor de la llamada, y posteriormente la red proporciona un canal con un ancho de banda que garantiza la comunicación. Al finalizar la llamada, la red libera los recursos que utilizó para la conexión, y estos, pueden ser empleados por otros abonados al sistema.

Como contrapartida, las redes no orientadas a la conexión, o redes de conmutación de paquetes, originariamente, fueron utilizadas exclusivamente para el transporte de datos, y su diseño no respondía a tipos de tráfico de tiempo real. Sin embargo, al principio de la década de 1990, distintos grupos de investigación, se mostraron interesados en el estudio del transporte de voz y video sobre redes de conmutación de paquetes basadas en el protocolo de Internet (redes IP). Las distintas investigaciones realizadas dieron origen a lo que hoy se conoce con el acrónimo de VoIP (Voz sobre el Protocolo de Internet), que consiste en dividir el audio en pequeños fragmentos, transmitir los fragmentos a través de la red IP, para finalmente ensamblarlos en el destino final.

En 1995, una pequeña compañía llamada VocalTec anunció el lanzamiento del primer software para comunicaciones entre computadoras a través de Internet. A tal fin se necesitaban utilizar diversos componentes de hardware tales como micrófonos, altavoces, tarjetas de sonido, tarjetas de red y módem´s. El funcionamiento de esta aplicación, en la actualidad sigue siendo básicamente igual; se debe transformar la señal de voz en muestras binarias, luego procesarlas para su compresión y finalmente generar datagramas IP para su transmisión. Sin embargo, la primera alternativa a la comunicación telefónica tradicional fue un fracaso comercial por el motivo de que las conexiones a Internet de las que se disponía en ese entonces carecían del ancho de banda necesario para garantizar alguna calidad de servicio para las comunicaciones de voz.

Durante los años siguientes, las tecnologías asociadas a las redes de datos y las comunicaciones continuaron innovándose y fue en 1998, cuando definitivamente se dieron los primeros pasos desde una perspectiva comercial. Diversas compañías lanzaron al mercado adaptadores que permitían utilizar los teléfonos tradicionales en entornos de VoIP. Ello facilitó el acercamiento de los usuarios a la tecnología VoIP, por lo que algunas importantes empresas lanzaron al mercado productos y servicios relacionados con la nueva tecnología. A finales de la década de 1990, la tecnología

VoIP alcanzaba el 1% del total del tráfico de voz.

En 1999, compañías dedicadas a la producción de tecnologías para la interconexión de infraestructuras de red, tales como Cisco, crearon sus primeras plataformas para cursar tráfico VoIP y sus clientes eran empresas y organizaciones de alta envergadura en lo que se refiere al tamaño. Este hecho impulsó la tecnología VoIP, y la consecuencia directa fue que VoIP alcanzó en el año 2000, más del 3% del total del tráfico de voz.

A partir del año 2000, se produjo una revolución tecnológica en todos los campos de las comunicaciones digitales, las redes de área local habían escalado desde los primitivos 10 Mbps a 100 Mbps y posteriormente a 1 Gbps. Las redes de acceso, aquellas que vinculan al usuario final de la comunicación con el proveedor de servicios de Internet, (ISP); escalaron desde los primitivos 56 kbps al orden de algunos Mbps merced a las tecnologías ADSL *(Asymmetric Digital Subscriber Line)* y estas últimas tecnologías están siendo reemplazadas en la actualidad por FTTH *(Fiber to the Home)* que escalan en varios ordenes de magnitud a las velocidades de ADSL. Las redes de transporte, (WAN´s) basadas en SONET o SDH (orientadas a la conexión) fueron reemplazadas por tecnologías Ethernet MPLS sobre enlaces de fibra óptica que pueden alcanzar velocidades de 40 Gbps o aún 100 Gbps. Las tecnologías de WDM permiten multiplexar hasta 160 canales Ethernet sobre un simple enlace de fibra óptica mono-modo. Los avances en tecnologías de microprocesadores permitió construir productos que evolucionan en velocidad (frecuencia de reloj en que las computadoras personales basan sus ciclos para la ejecución de su conjunto de instrucciones) y en lo que se refiere a múltiples núcleos *(multicore)* de procesamiento trabajando en forma colaborativa.

Como consecuencia de todo lo expresado en el párrafo anterior, existe en la actualidad una sobre demanda de recursos de ancho de banda, y las instalaciones telefónicas basadas en la antigua red telefónica pública conmutada (Public Switched Telephone Network – PSTN); van cediendo el lugar a sistemas de telefonía basados en VoIP.

Con respecto a la situación de la tecnología VoIP en Argentina, en el año 2007, la consultora “Prince & Cooke” presento un informe (1) sobre el panorama del mercado de las comunicaciones IP en Argentina, utilizando como muestra un grupo de más de 100 empresas de diversas partes del país. La Fig. 1.1 muestra el grado de adopción de la tecnología VoIP en el segmento de las grandes empresas. Con el gran avance que ha vislumbrado la tecnología VoIP en los últimos años, no es de extrañar que estos números se hayan incrementado, y el uso de la tecnología se haya extendido incluso fuera del ámbito interno de una empresa.

El área de Redes y Comunicaciones de la Facultad de Ingeniería de la Universidad de La Pampa, vislumbró desde hace años las posibilidades de generar soluciones basadas en VoIP, y en esa dirección se plantearon y finalizaron dos tesinas de grado, y un proyecto API (Áreas Prioritarias de Investigación) que fue seleccionado como de interés, por la Secretaría de Ciencia y Técnica de la UNLPam, y que fuera financiado por el Ministerio de la Producción de la Provincia de La Pampa.

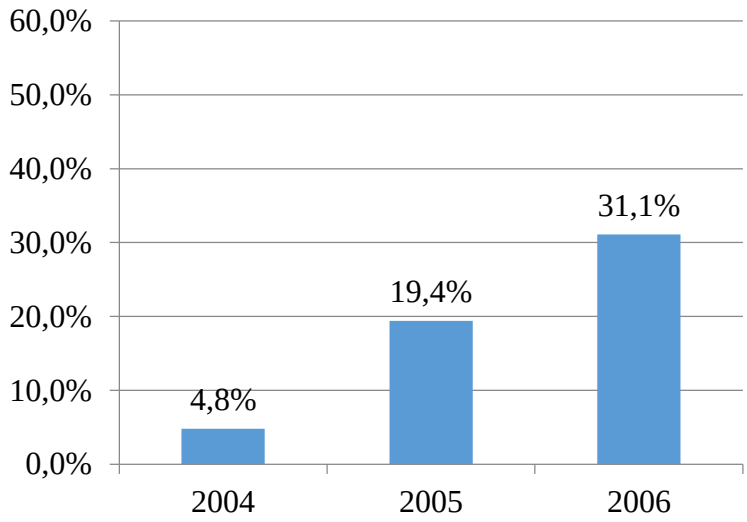

Fig. 1.1 – Adopción de VoIP en una muestra de grandes empresas en Argentina (2004-2006)

En orden cronológico, la primer tesina de grado “Implementación de Servicios de VoIP utilizando Asterisk” (2) se desarrolló durante el año 2009. La segunda tesina de grado, “Implementación de una Central Telefónica VoIP de Alta Disponibilidad en un ambiente Distribuido y de Calidad” (3) coincidió en su ejecución con el proyecto API “Sistema de Telefonía para Entidades Cooperativas de la Región” y culminó en el año 2012. El objetivo de los trabajos mencionados fue incursionar en soluciones VoIP, desde una perspectiva de crecimiento lento respecto al nivel de complejidad relacionado con los objetivos finales perseguidos.

A continuación, en la sección 1.1 se abordarán los alcances de los trabajos previos, mencionados en el párrafo anterior. Posteriormente, en la sección 1.2, se describirán los detalles a considerar, respecto de los trabajos (2) y (3), que motivaron la realización de este proyecto. Finalmente, en la sección 1.3, se expondrán los objetivos a cumplir, derivados de la sección 1.2.

1.1 Alcance de los trabajos previos

En el año 2009, se presenta en (2) el primer trabajo relacionado a VoIP. El objetivo perseguido fue el estudio de protocolos para VoIP, el diseño y la implementación de una central telefónica sobre una plataforma de Sistema Operativo basada en GNU/Linux utilizando Asterisk (4), una aplicación de software libre bajo licencia GPL, que proporciona funcionalidades de una PBX *(Private Branch Exchange)* o de una central telefónica privada secundaria (tal como lo es la que presta servicios de telefonía en la Facultad de Ingeniería). La solución desarrollada combina numerosas tecnologías de telefonía, tanto en lo referido a las tradicionales (analógicas) como en las basadas en VoIP, respondiendo a una arquitectura modular que la convierte en una solución flexible comparada con las centrales de conmutación de circuitos.

El trabajo en (2) comienza describiendo los conceptos básicos de la telefonía convencional, y dedica un apartado a la telefonía VoIP, destacando los principales protocolos que se pueden utilizar para el establecimiento de las llamadas. En el tercer capítulo de (2) se aborda Asterisk y un conjunto de herramientas de software que lo complementan. Finalmente se enumeran un conjunto de funcionalidades que una central

VoIP puede realizar y se exhibe una guía de pasos para la instalación de Asterisk. En las siguientes secciones del trabajo (2), se presenta un escenario de implementación junto a la configuración requerida para su ejecución. El escenario responde a la topología de la Fig. 1.2, en la que se emula una empresa con distintas áreas: gerencia, oficinas y una operadora. Se construye una red Ethernet que incluye teléfonos IP, teléfonos convencionales con adaptadores a VoIP *(Analog Telephony Adapter – ATA), notebooks* y computadoras de escritorio con aplicaciones de software para VoIP *(softphone)*. El sistema descrito en la Fig. 1.2 posee una interfaz que le permite al servidor Asterisk conectarse e interactuar con la central PBX analógica que brinda el servicio de telefonía en la Facultad de Ingeniería.

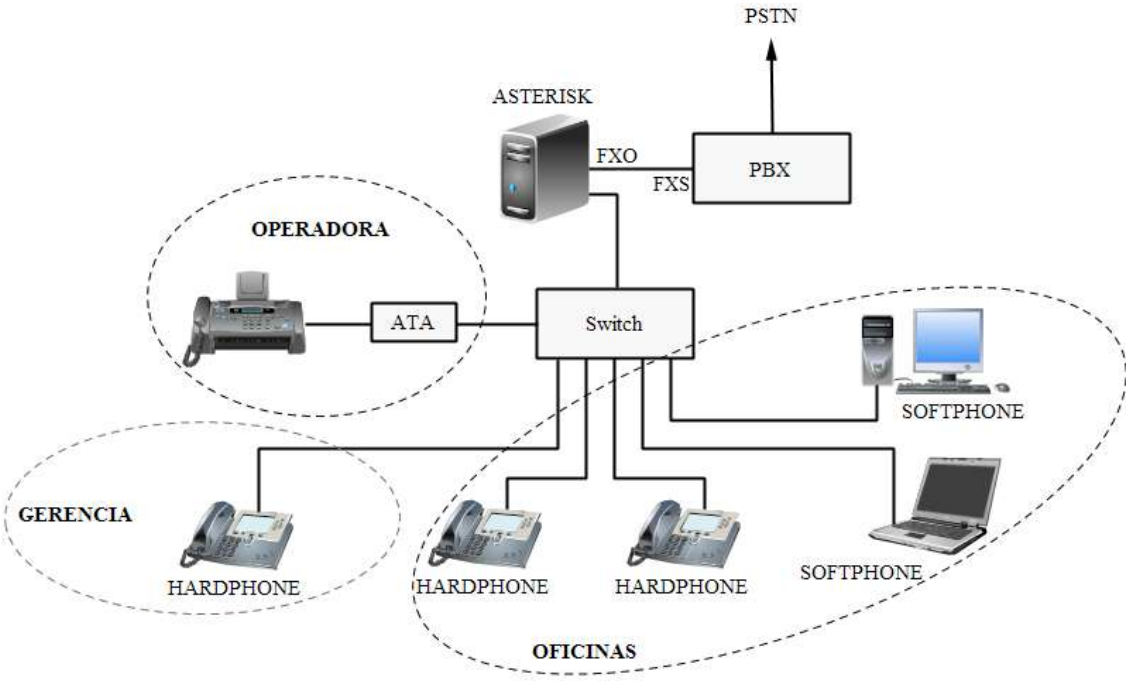

Fig. 1.2 – Topología de experimentación tesina (2)

Posteriormente en (2) se explican algunos de los servicios que se pueden ejecutar sobre la plataforma de la Fig. 1.2, tales como el correo de voz, transferencias o desvío de llamadas, salas de conferencias entre tres o más usuarios, y respuesta de voz interactiva *(Interactive Voice Response - IVR)*. El trabajo incluye la configuración de los módulos para el soporte de las opciones descritas al inicio de este párrafo. Finalmente, el trabajo trata sobre aspectos de configuración tendientes a realizar el “plan de marcación”. Se entiende por “plan de marcación” al sistema que permite a los usuarios llamarse unos a otros por medio de números telefónicos.

En el año 2011 se presentó el trabajo de tesina (3), cuyo objetivo central fue conformar un sistema distribuido constituido por dos servidores Asterisk, con posibilidades de escalar para adaptarse a escenarios que requieran mayor número de abonados al servicio telefónico, con soporte de calidad de servicio (QoS), y de alta disponibilidad, resultando en una solución superadora respecto de lo realizado en (2). El trabajo (3) aborda minuciosamente conceptos relacionados a calidad de servicio para la priorización del tráfico de tiempo real en una intranet. Presenta los conceptos claves de sistemas distribuidos utilizados en el diseño de la central telefónica, y las herramientas de software utilizadas para su implementación y ejecución. Sus autores, escogieron un

sistema distribuido en forma de clúster conformado por un conjunto de computadoras interconectadas mediante una tecnología de red de alta velocidad, en la cual los componentes de software que conforman la central telefónica se comunican y coordinan sus acciones a través del intercambio de mensajes. Concretamente, la central telefónica diseñada e implementada en (3) respondió a la topología mostrada en la Fig. 1.3. En la ilustración se puede observar que los usuarios finales del sistema interactúan con el modulo para el balanceo de tráfico encargado de distribuir equitativamente las peticiones de los usuarios, entre los servidores de registro, utilizando para ello una política de balanceo de tráfico de menor peso *(Weighted Least Connection)*. De este modo, las peticiones de los usuarios VoIP, se envían al servidor de registro que posea la menor cantidad de conexiones en curso. Opcionalmente es posible configurar distintos pesos, cuando los servidores de registro posean una marcada diferencia en lo que respecta a la capacidad de procesamiento. En el ámbito de (3) se optó por asignar el mismo peso dado a que ambos servidores de registro se implementaron sobre plataformas de hardware idénticas.

La función de los servidores de registro es vincular direcciones lógicas de usuarios con direcciones de red (IP). Cada usuario tiene una dirección lógica que es invariable respecto de la ubicación física del usuario. Una dirección lógica es de la forma:

```
usuario@dominio
```

La dirección de red, denominada “dirección de contacto", es dependiente del lugar desde donde el usuario se conecta (dirección IP). Para comunicarse con los servidores de registro, los balanceadores encapsulan la petición del cliente en un nuevo datagrama IP antes de enviarla. De esta manera, el servidor de registro puede contactarse directamente con el cliente sin necesidad recurrir al módulo balanceador, evitando de este modo sobrecargas bajo situaciones de alta demanda de tráfico.

Cuando un servidor de registro debe ubicar a un usuario, primero realiza una búsqueda local y, en caso de no encontrarlo, acudirá a un servidor en el módulo de localización *(lookup)* para conocer su ubicación (observar la Fig. 1.3). Los servidores de localización *(lookup)* son los encargados de intermediar en la búsqueda de los usuarios dentro del sistema telefónico VoIP. Para ello utilizan el protocolo DUNDi, que permite el intercambio de información relativa al plan de marcado; de este modo, pueden localizar clientes VoIP para satisfacer la demanda del módulo servidores de registro. A los servidores de localización, se les instaló una base de datos configurada en un modo de replicación maestro-maestro; que les permite a los servidores de registro acceder a consultas de variada información, tales como, usuarios configurados y cuentas de correo de voz, entre otras opciones.

El servidor de acceso a la PSTN es el encargado de conectar la infraestructura ilustrada en la Fig. 1.3, basada en VoIP, con una central analógica de la telefonía tradicional. Para ello se utilizó la central desarrollada en (2).

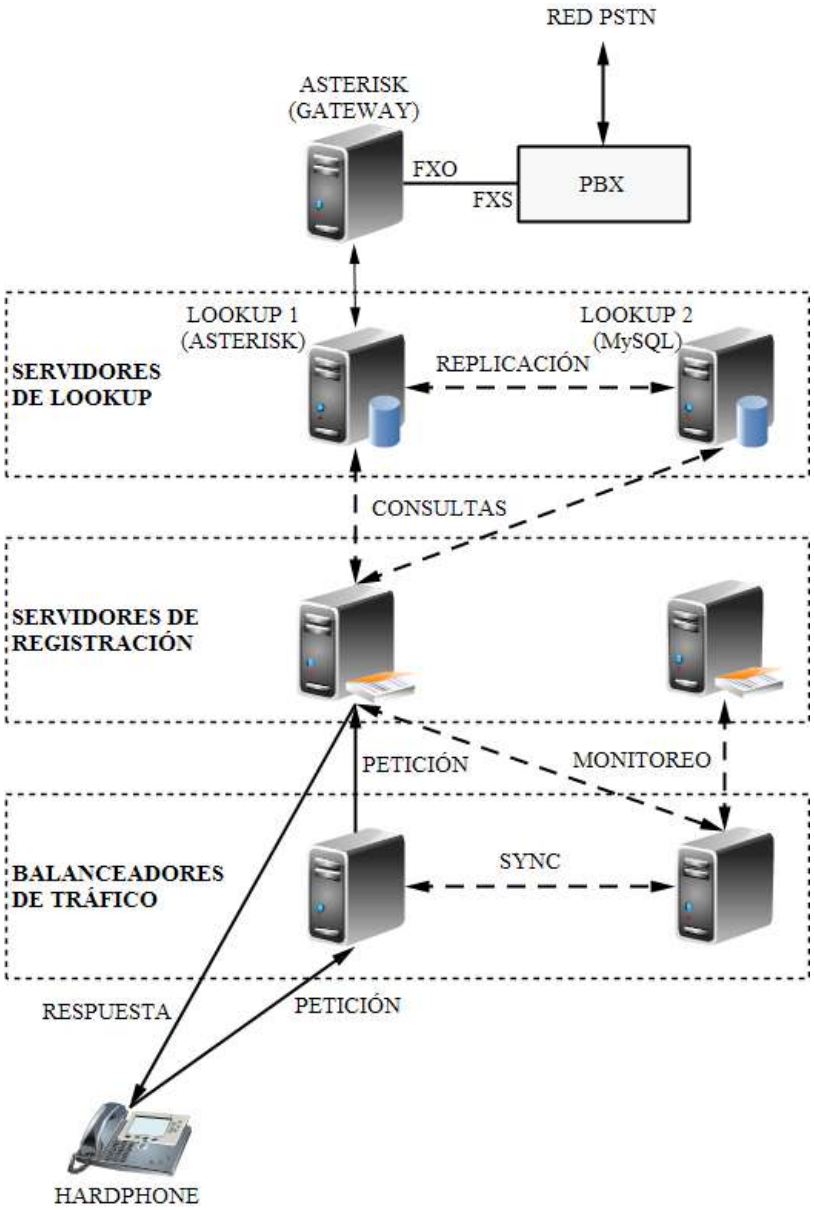

Fig. 1.3 – Topología de experimentación tesina (3)

El concepto de alta disponibilidad fue aplicado a todos los servidores, exceptuando la pasarela o *gateway* PSTN. En el caso de los balanceadores de tráfico, fueron configurados en el modo activo/activo, que permite que ambos servidores funcionen en el mismo nivel de jerarquía. En el caso de que uno de los servidores falle, el restante tomará el control de los recursos previamente administrados en el servidor fuera de servicio. Esto se logró mediante la asignación de una dirección IP a cada servidor, a través de la cual, los clientes pueden acceder al sistema. Ante un fallo en uno de los servidores, su dirección IP será migrada al que permanece activo. Es importante destacar que una falla en un servidor de balanceo, no afecta las comunicaciones en curso, pues como se explicó en esta sección, una vez establecida una comunicación de voz, el cliente se comunica directamente con el modulo servidor de registro.

Los servidores de registro trabajan en un modo activo/activo, y ante la falla de uno de ellos, los clientes que se encontraban registrados en él, serán registrados en el servidor activo tras un intervalo de tiempo de 60 segundos (intervalo de registración configurado

en los teléfonos VoIP utilizados). Si existía alguna llamada en curso al momento de la falla, esta culmina de manera inmediata, ya que el servidor de registro participa activamente en ellas.

Los servidores de localización *(lookup)* y de bases de datos fueron configurados en modo activo-pasivo. En este tipo de configuración, el servidor activo realiza toda la tarea, mientras que el servidor pasivo, en caso de que ocurra una falla (en el activo), tomará control de los recursos para asegurar la continuidad del servicio. Para optimizar los recursos de hardware, el servidor “*lookup.1*” (Fig. 1.3) se lo configuró para ese objetivo, mientras que al servidor “*lookup.2*” se lo configuró como servidor de base de datos, pero ante el desperfecto de alguno de ellos, el servicio es migrado al servidor activo.

En el contexto señalado en el párrafo anterior, una falla en el servidor de base de datos no afecta a la integridad de los datos de los usuarios, puesto a que la base de datos se encuentra replicada, lo único que ocurrirá es que el servicio se ejecutará en el servidor activo. Cuando el servidor se reestablezca volverá a adquirir todos los datos desde su par. Si se produce una falla en uno de los servidores de localización, el servicio será reestablecido de manera inmediata vía el servidor restante.

1.2 Consideraciones de los trabajos previos

Los trabajos sintéticamente descritos en la sección anterior han permitido incursionar en la temática VoIP, a la vez que ayudaron a comprender la evolución de la tecnología en forma progresiva, y posteriormente a implementarla. Luego de un exhaustivo estudio de (2) y (3) se concluyó en que existen numerosos aspectos en los que es posible contribuir a una solución superadora y actualizada, apta para ser implementada en ambientes de producción de cualquier envergadura. A continuación, se describirán aspectos de los trabajos previos que presentan problemas bajo ciertas circunstancias de operatividad.

En general, considerando la alta dinámica científica y tecnológica del área abordada, existen muchos componentes de software empleados en los trabajos descritos en la sección previa que en la actualidad son obsoletos, o que, en el mejor de los casos, necesitan ser actualizados. Para el primero de los casos no queda otra opción que la de reemplazar componentes de software y, para el último caso, se requiere de una actualización de componentes de software que por lo general incluye nuevas funcionalidades, que se traducen en nuevas posibilidades de diseño (rediseño).

En relación al trabajo (2), en que se implementó una central VoIP con funcionalidades que incluyeron salas de conferencias, correos de voz y notificación de llamadas perdidas, por nombrar algunas; el principal problema es que el límite de peticiones simultáneas que el sistema puede atender, depende de la capacidad de procesamiento del hardware utilizado. En ámbitos de producción que demanden una utilización intensa de peticiones que supere el límite antes descrito, no queda otra opción que la de migrar la central VoIP a un hardware con mayores prestaciones, en lo relativo a medidas de desempeño. Si bien la central VoIP diseñada trabaja correctamente, una falla en el hardware, o en alguno de los componentes de software que la conforman provoca una interrupción del servicio, con las implicancias que ello genera. No obstante, es un producto cuyo ámbito de aplicación está perfilado a instituciones u organizaciones de pequeña envergadura.

Del análisis exhaustivo del trabajo en (3) y de la posterior prueba operativa de esa versión de en un ambiente de producción (Cooperativa Popular de Obras y Servicios Públicos de Río Tercero) se concluyó en que la implementación presenta una serie de aspectos a resolver. A continuación los detalles:

- Se utilizan dos bases de datos MySQL con replicación maestro-maestro lograda a través de una doble replicación maestro-esclavo. Este modo anula cualquier opción de crecimiento; puesto que, un servidor solo puede ser esclavo de un único maestro. La configuración no puede escalar a un nivel superior a dos servidores en lo relativo al servicio de base de datos.
- Un problema análogo al anterior sucede con el módulo balanceo de tráfico que fue configurado en modo, maestro-esclavo. El servicio no puede escalar en un nivel superior a dos servidores.
- En el caso de los servidores de bases de datos y los servidores de localización *(lookup)*, tal como se mencionó anteriormente, operan con una configuración de alta disponibilidad denominada activo-pasivo. El problema de este esquema es que ante una sobredemanda de procesamiento, el servidor activo quedará fuera de servicio, cuando el servidor pasivo migre al rol activo le ocurrirá exactamente lo mismo, y ello se traducirá en la pérdida del módulo de localización, que comparte los componentes de hardware con el servicio de base de datos.
- El diseño del módulo “servidores de registro” tiene inconvenientes de disponibilidad. Cuando un servidor de registro recibe una petición de un cliente para comunicarse con otro usuario del servicio, el servidor de registro realiza una búsqueda local, y en caso de que arroje un resultado negativo, recurrirá al servidor de localización. Dadas las circunstancias descritas en el punto anterior, la llamada no se podrá concretar. Adicionalmente cuando ocurra una falla en uno de los servidores de registro, y se desee llamar a un usuario que se encuentre registrado en él, la llamada no podrá concretarse (el servicio de localización no podrá hallar el cliente).
- Un problema, no menor, es la elevada complejidad de diseño, que incluye un número elevado de componentes de software que, si bien se complementan, se configuran por separado, aspecto que se traduce en un trabajo tedioso de configuración.
- En lo que respecta al funcionamiento de la central VoIP, existe un exceso de complejidad en el número máximo de transacciones para el establecimiento de una llamada. En el peor de los casos, para establecer una comunicación entre clientes que se encuentran registrados en diferentes servidores de registro, los mensajes atravesarán tres servidores. El usuario que inicia la llamada enviará un mensaje que arribará a uno de los balanceadores de tráfico, este lo encapsulará dentro de un nuevo datagrama IP y posteriormente lo enviará al servidor de registro que posea la menor cantidad de conexiones. El servidor de registro al recibir el datagrama, comprobará que el cliente al que se desea llamar se encuentra registrado en el otro servidor de registro, quien finalmente será el responsable de conectar al cliente destinatario del llamado.
- Otro aspecto no deseable en el diseño es que el módulo para el balanceo de carga se trata de un *router,* y no de balanceadores tal como lo sugiere la Fig. 1.3.

- El problema de mayor envergadura radica en que la información de los clientes se encuentra distribuida entre los módulos de “servidores de registro” y “servidores de localización”. Estos datos carecen de replicación, lo que implica que, ante una eventual falla en cualquiera de los componentes involucrados, se perderá el registro de los clientes vinculados con el servicio afectado, y el proceso de recuperación demandará una mayor capacidad de procesamiento y recursos de red.

El presente trabajo resolverá todas las falencias descritas, empleando metodologías que deriven en un rediseño de (3) que permita simplicidad de configuración, compatibilidad de los componentes de software utilizados, y optimización en el uso de los recursos de hardware. En la próxima sección se plantean los objetivos de este trabajo, que resultaran en un rediseño del trabajo (3).

1.3 Objetivos

La motivación de este trabajo final es el rediseño e implementación de una central telefónica VoIP de bajo costo, utilizando componentes de software de fuentes abiertas y libres para la implementación, que responda a solucionar los problemas detectados en el trabajo (3). Concretamente, se pretenden alcanzar los siguientes objetivos:

- Diseñar una central de telefonía VoIP con capacidad para adaptarse a distintos ámbitos de aplicación. Desde pequeñas organizaciones hasta ciudades de cualquier envergadura. La central a diseñar deberá poder escalar a un número creciente de clientes (abonados al servicio) mediante el simple agregado de servidores, con un mínimo esfuerzo de configuración en los componentes de software.
- La implementación tiene como premisa fundamental aportar una solución de bajo costo, que provea como mínimo los mismos servicios que una central analógica convencional.
- Se considerará en el diseño el concepto de alta disponibilidad. Se pretende que los servicios ofrecidos por el sistema estén disponibles para el usuario final, aun cuando ocurra cualquier falla en los múltiples componentes de software.
- Se modificará el diseño de los módulos de la Fig. 1.3 para evitar las limitaciones descritas en la sección 1.2 de este capítulo.

Capítulo 2
"Sistemas Distribuidos"

2.0 Introducción

El desarrollo y la evolución de los sistemas distribuidos se produjeron por distintos factores. El primero de ellos está relacionado con el crecimiento exponencial en la tasa de transmisión de las redes de área local, de área amplia, y de las redes de acceso. Otro factor, no menos importante, fue el desarrollo de servidores de altas prestaciones cuyas arquitecturas incluyen tecnologías de microprocesadores multi-núcleos operando a velocidades de hasta 3,7 GHz, con tecnologías de redundancia incorporadas en todos los componentes que conforman el sistema. El desarrollo de memorias RAM alcanzó el orden de los 64 GB por zócalo, y las tecnologías de discos rígidos el orden de los TB. Todo lo descrito se traduce en productos aptos para altas prestaciones en lo relativo a la disponibilidad y a la capacidad de procesamiento.

El crecimiento en la demanda de sistemas distribuidos impulsó el desarrollo de software específico para que los servidores que conforman un sistema coordinen sus actividades y compartan los recursos distribuidos. En el contexto de este trabajo, cuando se emplee el término "recurso", se lo interpretará como el uso compartido de dispositivos de hardware o de software entre los miembros y/o clientes de un sistema distribuido.

A continuación, en la sección 2.1, se presentarán algunas definiciones conceptuales de un sistema distribuido, junto a las características que surgen del análisis de las definiciones. Posteriormente, en la sección 2.2 se introducirán los tipos de sistemas distribuidos, y se efectuará una elección para el diseño de la central telefónica. Luego, en la sección 2.3, se presentarán los conceptos que caracterizan a los clústeres, y como se desarrollaron en la central telefónica. Finalmente, en las secciones 2.4 y 2.5, se explicarán los conceptos de alta disponibilidad y escalabilidad, los cuales son objetivos a alcanzar en el presente trabajo.

2.1 Definición de Sistema Distribuido

Distintos autores han definido conceptualmente el significado del término "sistema distribuido", algunas de ellas se expresan a continuación:

- Colouris en (5), expresa que un sistema distribuido es aquel en el cual los componentes de hardware y software ubicados en computadoras conectadas en red, coordinan sus acciones mediante el intercambio de mensajes.
- Tanembaun en (6), se refiere a un sistema distribuido como una colección de computadoras independientes cuya apariencia a los usuarios finales es la de un sistema coherente y único.
- Lamport en (7) define un sistema distribuido, como aquel que ante una falla en uno o más dispositivos que lo conforman, (que el usuario desconoce respecto de su existencia) los trabajos continúan ejecutándose normalmente.

De las definiciones previas, se deducen algunas características que debería poseer todo

sistema distribuido. La primera de ellas es que un sistema distribuido, desde la perspectiva de usuario, aparenta ser un sistema único y centralizado. Otra característica destacable hace referencia a que cada computadora o servidor en el sistema distribuido puede trabajar, internamente, de manera diferente a las demás, pero al colaborar en una misma tarea debe estar interconectada a una red para comunicar y sincronizar sus avances. Finalmente, se deduce que los componentes de un sistema distribuido deben coordinar sus acciones para que en todo momento cada componente del sistema conozca fehacientemente el estado de sus pares.

2.2 Tipos de Sistemas Distribuidos

Existen numerosos tipos de sistemas distribuidos, clasificables y perfilados para aplicaciones específicas. Una breve síntesis respecto de su clasificación se brinda a continuación. Sistemas de computación distribuida basada en grid (6) o en clúster (6). Sistema de información distribuida para el procesamiento de transacciones o integración de aplicaciones en empresas. Sistemas distribuidos empotrados o ubicuos, aplicables a redes de sensores, a sistemas hogareños, a sistemas electrónicos para el cuidado de la salud, etc.

Como el objetivo del presente trabajo es desarrollar una central telefónica escalable (en lo que se refiere al número de clientes a servir), la elección respecto del tipo de sistema distribuido a utilizar se orienta hacia aquellos que provean una capacidad de procesamiento adaptable a cada circunstancia. Expresada la finalidad, y de acuerdo a la clasificación del párrafo previo, se opta por un sistema de computación distribuido.

Un sistema de computación distribuido basado en grid (6) se orienta a organizaciones que necesitan compartir recursos geográficamente distribuidos para un objetivo común, sin una locación central, ni control central, en una relación de mutua confianza. Claramente este tipo de sistema no se adapta a los fines de este trabajo.

Un sistema de computación basado en clúster (6) está conformado por un grupo de computadoras interconectadas por una red de área local (LAN) de alta velocidad, que lógicamente se comportan como una computadora. Se utilizan para mejorar el rendimiento y/o disponibilidad y constituye una atractiva solución desde la perspectiva económica respecto de una computadora individual de velocidad y disponibilidad comparable a la de un clúster.

En el contexto del trabajo, la Fig. 2.1 ilustra el concepto de clúster. Un usuario del servicio brindado por la central telefónica VoIP, abstrae su visión a la de un sistema de alta disponibilidad sin conocer la existencia o dimensión del clúster y menos aún sobre los componentes que esporádicamente pueden estar activos o bien fuera de servicio.

La Fig. 2.1 es un ejemplo representativo que involucra un número de servidores, en este caso seis, pero que puede extenderse recurriendo a tecnologías de red de mayor velocidad y a servidores con mayores prestaciones en lo relativo a capacidad de procesamiento.

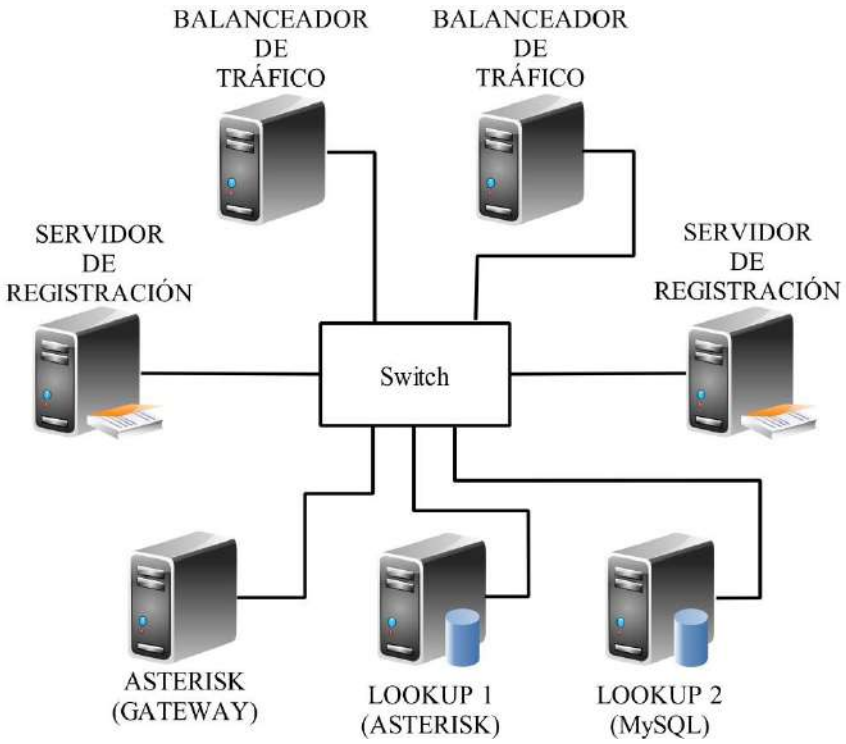

Fig. 2.1 – Clúster de Computadoras para Central VoIP

En la siguiente sección se abordará desde una perspectiva conceptual las características típicas de un clúster.

2.3 Características de un Clúster

Los conceptos que permiten caracterizar a un clúster están referidos al acoplamiento del software, el control y la homogeneidad del clúster (6).

El acoplamiento de software dependerá del tipo de software utilizado, y el término hace referencia a la integración que posea todo el software existente en cada nodo del clúster. Se definen dos tipos de acoplamiento catalogados como “fuerte” y “débil”. El acoplamiento fuerte se produce en componentes de software distribuido que se vinculan persistentemente para la concreción de un objetivo común. Un ejemplo claro de acoplamiento fuerte es la imagen del núcleo de un sistema operativo, distribuido entre los nodos que conforman el clúster. Su contraparte, el acoplamiento débil, permite que las computadoras del clúster sean independientes entre sí, se las pueda diferenciar en forma univoca, y en caso de ser necesario, podrán interactuar mediante el uso de bibliotecas desarrolladas especialmente para sistemas distribuidos.

En el caso de la central telefónica que se busca diseñar e implementar, el software presentará un acoplamiento débil. Cada una de las computadoras que intervienen en el sistema distribuido es fácilmente diferenciable, inclusive, entre miembros de un mismo módulo, como por ejemplo, el de balanceo o el de registración.

El término control del clúster hace referencia a la forma en la que se lo administra y configura. En tal sentido, se identifican dos modelos de control, denominados “centralizado” y “descentralizado”. En el modelo centralizado, un nodo maestro es el

que tiene la capacidad para configurar el clúster como un todo. La ventaja de este modo, es la facilidad para administrar y configurar el sistema, y la desventaja inherente es que ante una falla en el nodo maestro resultará imposible el control y la administración del sistema. Por su parte, en el modelo de control descentralizado, cada nodo del sistema se administra y se gestiona en forma aislada, aunque se pueden utilizar aplicaciones de alto nivel centralizadas para su gestión. La ventaja del modo de control descentralizado es que el sistema presenta una mayor tolerancia a fallas a nivel general, y la desventaja es que insume un mayor tiempo para la gestión y administración de los equipos en el clúster.

En lo referente a la característica de control del clúster, la central telefónica se diseñará de acuerdo con el modo de control descentralizado. A los efectos de atenuar las desventajas de este modo, se recurrirá a configuraciones idénticas en cada módulo que conforman la arquitectura de la central telefónica VoIP.

El último concepto para caracterizar un clúster hace referencia a la “paridad” o “disparidad” en las computadoras que lo componen. Se dice que un clúster es homogéneo cuando todos los nodos poseen una arquitectura y recursos idénticos (o muy similares), de forma tal que no existan considerables diferencias entre ellos. Su contraparte son los clústeres heterogéneos, caracterizados por los nodos que pueden poseer arquitecturas, sistemas operativos o incluso tiempos de acceso disímiles entre sí. Generalmente, los sistemas tipo clúster presentan un modelo homogéneo.

A nivel de hardware, las computadoras que conformarán la central telefónica presentarán una arquitectura con un cierto grado de heterogeneidad, mientras que, a nivel de software, la pertenencia a uno de estos modelos dependerá del modo en que se la observe. Si se observa cada módulo por separado, se concluirá que las computadoras que lo conforman son homogéneas, pero si se observa la central de forma global, se concluirá en que hay un cierto grado de heterogeneidad.

Los clústeres se clasifican de acuerdo a la funcionalidad u objetivos que se pretenden alcanzar. En el caso concreto de la central telefónica a diseñar, se construirá un clúster que ofrecerá servicios de alta disponibilidad, servicio de balanceo de carga, y servicio de escalabilidad respecto del número de usuarios a servir.

A continuación, en la sección 2.4 se presentará, desde lo conceptual, lo referido al concepto de alta disponibilidad.

2.4 Alta Disponibilidad

En la actualidad las empresas que brindan servicios de telefonía o de IPTV (Televisión sobre el protocolo de Internet), por nombrar algunos de ellos, dependen de sus sistemas de información, y resulta obvio que el objetivo principal es que los servicios permanezcan disponibles el mayor tiempo posible.

Para una empresa, una interrupción en el sistema supone un grave problema por las implicancias derivadas que se traducen en mayores costos asociados a la reparación del sistema, horas de trabajo adicionales para el departamento encargado de las reparaciones, pérdida de productividad de los usuarios que dependen del sistema para efectuar sus trabajos, insatisfacción de los usuarios que se traduce en una pérdida de

reputación y mala publicidad para la empresa prestataria del servicio.

Un sistema de alta disponibilidad o HA *(High Availability)* (8) consiste en que el servicio que se brinda, este lo suficientemente redundado como para que no sufra interrupciones, aun cuando existieran incidencias. Para dotar a un sistema con esta característica, se debe utilizar tanto soluciones a nivel software como a nivel hardware.

Respecto al nivel de hardware, los servidores que conforman el clúster deberían poseer fuentes de alimentación redundantes con componentes de respaldo respecto de su energización. A nivel del hardware de red, cada interfaz debería duplicarse de tal forma que si se produce una falla en la interfaz activa, la redundancia aseguraría condiciones de operatividad garantizada. El mismo concepto de redundancia se aplica a los discos rígidos, un sistema RAID (*Redundant Array of Independent Disks*) garantizaría la operatividad de un componente servidor aun cuando se produzca una falla en algunos de los discos que conforman el arreglo (RAID). Otro aspecto importante que merece ser destacado es relativo a proveer redundancia en la infraestructura de red que comunica a los nodos en el clúster. Todo lo descrito en este párrafo asegurara una alta disponibilidad en lo relativo a los componentes de hardware utilizados y a la infraestructura de red propiamente dicha, pero obligatoriamente debe diseñarse un sistema de alta disponibilidad en lo que se refiere a los componentes de software distribuidos en el clúster.

2.4.1 Configuraciones de Redundancia

La cantidad mínima de nodos para proveer un clúster de alta disponibilidad es dos, aunque no es lo recomendado. Según los requerimientos iniciales, el diseño del sistema distribuido puede pertenecer a alguno de los siguientes modelos de redundancia (8):

Modelo Activo/Pasivo: Uno de los nodos es el maestro responsable de la ejecución de una determinada tarea. El resto de los nodos redundantes para la tarea específica, o nodos esclavos, permanecerán en el modo pasivo a menos que se produzca un incidente en el nodo maestro.

Modelo Activo/Activo: En esta configuración todos los nodos que poseen el mismo software para la ejecución de una determinada tarea, trabajan en un modo colaborativo mediante algún algoritmo de planificación que equilibre la carga total entre los nodos participantes. Cuando uno o más nodos fallen, la carga se deberá repartir entre los nodos activos.

Modelo N+1: Esta configuración posee N nodos activos ejecutando una o más tareas y un nodo pasivo. Cuando se produce una falla en uno de los N nodos, el nodo pasivo procede a ocupar su lugar, y cuando el afectado se recupera se convierte en el nodo pasivo. Es evidente que en este esquema de redundancia solo puede fallar uno de los nodos activos, y que todos deben tener la misma disposición.

Modelo N+M: Existen M nodos en estado pasivo para N nodos en estado activo. Cuando se produce una falla en uno de los N nodos activos, uno de los M nodos pasivos se convertirá en activo. Este modelo es una extensión del N+1.

Modelo N a 1: A diferencia del modelo N a 1, cuando uno de los N nodos activos falle, el nodo pasivo toma su lugar hasta que el nodo se recupere, entonces retorna al modo pasivo.

Modelo N a N: Es una combinación de las configuraciones Activo/Activo y N+M. Cuando un nodo falla, las tareas del mismo son redistribuidas sobre el resto de los nodos activos. Esta configuración elimina la necesidad de nodos pasivos, sin embargo, necesita que cada miembro tenga capacidad extra.

Con el fin de aprovechar al máximo la capacidad de cada nodo que conformará la central telefónica VoIP, el modelo de redundancia utilizado para el diseño será el de Activo/Activo.

2.4.2 Medición de la Disponibilidad

Existen varias métricas para el cálculo de la disponibilidad (9), y cada una de ellas sirve para calcular diferentes tipos de disponibilidad que conducen a distintas conclusiones respecto del sistema distribuido. Por lo tanto, la elección sobre cuál de ellas utilizar es crítica. Para el caso de una central telefónica, la métrica que ofrece una buena conclusión respecto de la disponibilidad del sistema es la denominada “disponibilidad operacional” (9). La disponibilidad operacional es una medida “real” de la disponibilidad sobre un período de tiempo dado. Incluye todos los tipos de causales de tiempos de inactividad del sistema, que afectan en forma directa a los usuarios del sistema. Esta métrica es la razón entre el tiempo de actividad del sistema, respecto del tiempo total a evaluar. Matemáticamente, se representa por la siguiente expresión:

$$D_O = \frac{Tiempo\ de\ Actividad}{Tiempo\ Total}$$

A modo de ejemplo, se expone el siguiente caso: se desea conocer la disponibilidad de un el sistema en un año no bisiesto. El “tiempo total” es de 8760 horas (un año). Se comprobó que el sistema tuvo 20 horas de inactividad en el periodo considerado. Según los datos brindados la disponibilidad operacional será $D_0 = 0{,}9977$. Ello significa que el sistema estuvo disponible el 99,77% del año. La disponibilidad suele medirse en “nueves” (8). En el ejemplo expuesto anteriormente, se puede concluir que la disponibilidad del sistema fue de “dos nueves”. En la Tabla 2.1, se expone la relación entre distintos niveles de disponibilidad junto al tiempo de inactividad que representan.

Disponibilidad (%)	Inactividad por Año	Inactividad por Mes
90%	36,5 días	72 horas
95%	18,25 días	36 horas
99%	3,65 días	7,2 horas
99,5%	1,83 días	3,6 horas
99,9%	17,52 horas	43,8 minutos
99,99%	52,56 minutos	4,32 minutos
99,999%	5,26 minutos	25,9 segundos

Tabla 2.1 – Niveles de Disponibilidad

2.5 Escalabilidad

El término escalabilidad (6) hace referencia a la capacidad de un sistema para adaptarse a distintos ambientes de aplicación, contemplando el factor crecimiento y la no limitación respecto del número máximo de usuarios a servir.

El sistema distribuido que proveerá el servicio de una central telefónica será escalable. Ello significa que la solución a implementar se adaptará a cualquier ámbito de aplicación respecto del número de abonados a servir. En caso de crecimiento bastara con añadir nuevos recursos en forma modular.

2.5.1 Tipos de Escalabilidad

Se distinguen dos tipos de escalabilidad: la escalabilidad vertical y la horizontal. El sistema escalará verticalmente si al añadir o reemplazar recursos en uno de sus nodos, el desempeño del sistema mejora como un todo. Un ejemplo representativo de escalabilidad vertical, es el añadir memoria extra a uno de los nodos, o reemplazar el disco por uno de mayor velocidad en lo que se refiere a revoluciones por minuto (rpm). El aspecto negativo de este modo de escalabilidad se vincula a mayores costos de inversión para la adquisición de hardware de altas prestaciones. Si estos costos son amortizables en un periodo razonable, esta opción es atractiva pues no incide a nivel arquitectónico del sistema.

El escalado horizontal hace referencia al agregado de más nodos para aumentar el rendimiento y escalabilidad del sistema. Se pueden añadir nodos extras al sistema, y distribuir equitativamente la carga según la tarea que ejecuten. En principio, no existen limitaciones de crecimiento para una escalabilidad horizontal, pero ello dependerá de un buen diseño. En el ámbito de este trabajo se optará por diseñar un sistema de escalabilidad horizontal.

Capítulo 3
"Protocolos de Señalización y Multimedia para VoIP"

3.0 Introducción

Tal como se mencionó en la sección 1.0, las redes de conmutación de paquetes en su diseño original no fueron concebidas para el transporte de tráfico multimedia, que exhiba características de tiempo real. Aunque el estándar que define IPv4 (RFC 791) describe el campo ToS (*Type of Service*); para brindar un tratamiento diferenciado de acuerdo al tipo de tráfico en *buffers* de los *routers*, en la práctica el campo ToS nunca fue utilizado y los datagramas IP que ingresan a un dispositivo de capa 3 son atendidos de acuerdo a la política FCFS (*Firts-Come, First-Served*). Ello significa que un *router* en Internet procesa a los datagramas que arriban por sus interfaces de acuerdo al orden de llegada sin importar cuál es el protocolo que encapsulan los datagramas IP. Un problema adicional al descripto en este párrafo era el escaso ancho de banda de la infraestructura de Internet, que impedía a los usuarios mantener un diálogo aceptable comparado con el que proveía la telefonía convencional.

A mediados de la primera década del siglo XXI la situación cambió, y en la actualidad existe una sobre oferta de ancho de banda. Los proveedores de servicios de Internet (*ISP, Internet Service Provider*) pueden adquirir ancho de banda según sus propias estimaciones, en lo relativo al tráfico, para brindar servicios y obtener un alto grado de satisfacción de sus abonados. En este contexto, se suma el aporte de herramientas de software tales como la que brinda el paquete "*iproute2*", que entre otras funcionalidades permite crear canales lógicos con un ancho de banda garantizado, aspecto no menor pues desde la perspectiva de un prestador de telefonía VoIP, el tráfico de voz se puede derivar por un canal exclusivo, diseñado en lo relativo a su capacidad, de acuerdo a las características cuantitativas de los abonados a atender.

Así como la telefonía convencional necesita de un protocolo de señalización (SS7) para establecer, mantener y finalizar una comunicación de voz, los sistemas de telefonía basados en VoIP requieren de protocolos para la señalización y el transporte de datos multimedia.

A continuación, en la sección 3.1 se explicará el protocolo de señalización más utilizado en implementaciones VoIP, el protocolo SIP (*Session Initiation Protocol*), posteriormente, en la sección 3.2; se describirán las características principales del protocolo RTP *(Real Time Protocol)*, para el transporte de los datos multimedia. Finalmente, en la sección 3.3 se presentarán ejemplos en lo que se refiere a la operación y complementación de ambos protocolos.

3.1 Protocolo SIP

El Protocolo SIP, es un protocolo de señalización desarrollado por la IETF *(Internet Engineering Task Force)* y su última versión es la 2 (10). SIP es un protocolo que opera a nivel de capa de sesión respecto del modelo de referencia OSI, y permite crear, modificar y finalizar sesiones multimedia entre uno o más participantes. Utiliza según

las opciones de configuración los protocolos de transporte UDP o TCP, en el puerto bien conocido 5060.

SIP se encarga de la señalización, y una vez que establece una sesión, los datos multimedia se transmiten en forma independiente, a través de los protocolos RTP *(Real-time Protocol)* y RTCP *(Real-Time Control Protocol)*.

Una característica importante del protocolo SIP es que se trata de un protocolo transaccional, en que la interacción entre los componentes surge de un intercambio de mensajes basado en el modelo cliente-servidor (5). Una transacción consta de todos los mensajes desde la primera petición realizada por el cliente hasta la última respuesta del servidor, que no sea provisional. Estos mensajes serán explicados en la sección 3.1.2. Otro concepto importante dentro del ámbito del protocolo SIP es el de diálogo, el cual refiere a la relación entre dos componentes que perdura en el tiempo, y se compone de una o más transacciones.

3.1.1 Identificador de Recursos Uniforme

Una de las facetas en el establecimiento de una sesión es la localización de los usuarios, y para poder cumplir con esta funcionalidad, el protocolo utiliza un Identificador de Recursos Uniforme o URI (*Uniform Resource Identifier*). El URI provee un modo simple y extensible de identificar un recurso, ya sea abstracto o físico. La sintaxis genérica de un URI se encuentra especificada en (11).

En el ámbito del protocolo SIP, el URI SIP permite identificar a los recursos de comunicación, y posee la suficiente información para iniciar y mantener una comunicación de voz entre estos recursos involucrados. Ejemplos de recursos de comunicación son usuarios de un servicio en línea, casillas de correo de un servidor de mensajería o un número telefónico de la red analógica.

El formato de un URI SIP posee una sintaxis muy similar al de una dirección de correo electrónico. A continuación se expone un URI SIP genérico (10).

```
sip:usuario[:clave]@host[:puerto]
```

Los campos `clave` y `puerto` son opcionales, y por lo general no se utilizan. Si no se especifica el `puerto`, se utilizará el puerto asignado por defecto al protocolo SIP. No se recomienda el uso del campo `clave` pues, dentro del mensaje SIP, el URI se encapsula como texto plano, y este aspecto revelaría datos de autenticación de los usuarios. El campo `host` asumirá como valores válidos, un nombre de dominio o una dirección IPv4 o IPv6. Finalmente, el campo `usuario` permite identificar el recurso dentro del dominio especificado en el campo `host`, y se conforma por una combinación de letras y números.

En el caso de la central VoIP a implementar, se opta por utilizar un URI sencillo, que consta del número telefónico del abonado para el campo `usuario`, y la dirección IPv4 para el campo `host`. Notar que la opción escogida no afecta el concepto de movilidad, pues si un usuario cambia su teléfono a otra subred IP, deberá registrarse en el módulo servicio de registro y en este proceso notificará su nueva dirección IPv4.

3.1.2 Mensajes SIP

El protocolo SIP opera valiéndose de mensajes de petición y respuesta. La fase de inicio de una sesión comienza cuando un cliente SIP envía una petición a su par servidor quien luego de procesarla le responde al cliente.

Conforme a (10), ambas partes del servicio SIP utilizan un formato de mensaje general definido en (12), aunque también existen implementaciones que utilizan campos en el encabezado que no se corresponden con el estándar RFC 5322.

El encabezado del protocolo SIP se compone de un campo inicial que indica el tipo de mensaje, uno o más campos de encabezado, un campo nulo para indicar el final del encabezado, y un campo opcional. Estos mensajes están basados en texto plano, y utilizan el conjunto de caracteres UTF-8. La Fig. 3.1 ilustra un mensaje SIP conteniendo los campos descritos en este párrafo.

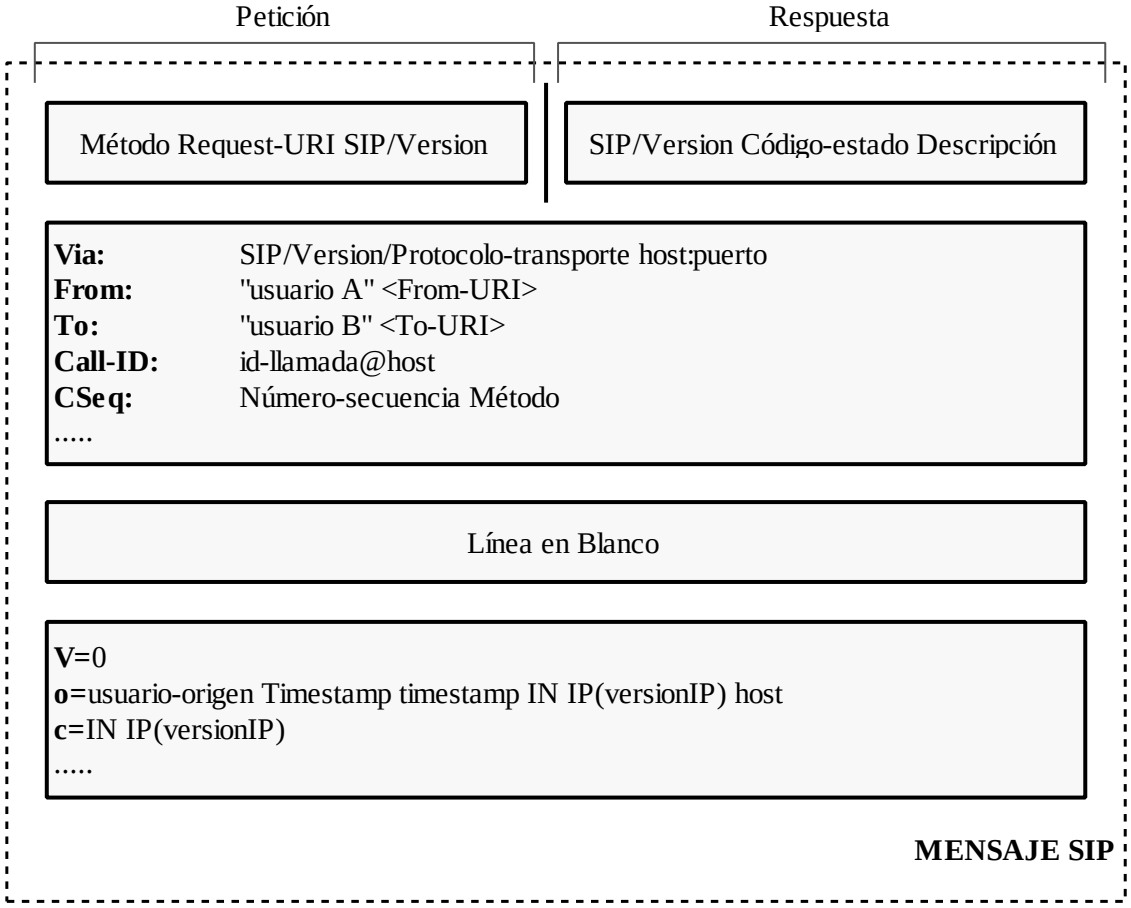

Fig. 3.1 – Estructura genérica del mensaje SIP

3.1.2.1 Peticiones y Respuestas SIP

Las peticiones, al igual que las respuestas, se tipifican en el primer campo del mensaje SIP (ver Fig. 3.1). En el caso de la petición mostrada en la Fig. 3.1, contiene el nombre del tipo de solicitud, un *Request-URI*, que sirve para indicar el destino de la llamada; e

incluye la versión del protocolo SIP del usuario que inicia la sesión. Existen seis modos de petición definidos en el estándar (10):

- *Register*: Utilizado para registrar la información de contacto del usuario, que posibilitará localizarlo al efecto de establecer sesiones.
- *Invite*: Se utiliza para invitar a un usuario o servicio a participar en una sesión, o bien, para modificar los parámetros de una sesión existente; en cuyo caso, se denomina a la petición, Re-Invite.
- *Cancel*: Utilizado para cancelar una petición pendiente.
- *Bye*: Finaliza una sesión.
- *Ack*: Utilizado para reconocer la recepción correcta de un mensaje. No genera respuestas.
- *Option*: Su finalidad es la de consultar a los servidores o clientes acerca de las capacidades que soportan. Ejemplos de capacidades son los *codecs* soportados por el cliente, o los métodos que este puede invocar.

Los mensajes de respuesta se distinguen por el primer campo en el encabezado SIP y contiene la versión del protocolo seguido de un código de estado que incluye una breve descripción textual, destinada al usuario para que interprete la respuesta.

El código de estado está compuesto por tres dígitos enteros, que indican la salida correspondiente al procesamiento de una petición (10). Como ilustra la Tabla 3.1, el primer dígito define la clase o tipo de respuesta, mientras que los dígitos restantes representan algún motivo particular dentro de la categorización. Algunos ejemplos de códigos son: 100 (TRYING), 180 (RINGING), 200 (OK), 301 (MOVED PERMANENTLY), 401 (UNAUTHORIZED) y 503 (SERVICE UNAVAILABLE).

Código	Descripción
1XX	Provisional. La petición fue recibida y se continúa con su procesamiento.
2XX	Éxito. La petición se recibió, se entendió y fue aceptada.
3XX	Redirección. Se necesitaran de más acciones para completar la petición.
4XX	Falla del cliente. La petición tiene una mala sintaxis o no se puede completar en este servidor.
5XX	Falla del servidor. El servidor no puede completar una petición, aparentemente, válida.
6XX	Falla global. Ningún servidor puede completar la petición.

Tabla 3.1 – Códigos de Estado del protocolo SIP

3.1.3 Campos del encabezado SIP

Los campos del encabezado SIP son muy similares a los del protocolo HTTP, tanto sintáctica como semánticamente (10). Cada uno de los campos del encabezado se compone de un nombre de campo seguido de dos puntos “:” y el valor del campo. Los campos mínimos que debe contener un mensaje SIP son los siguientes:

- *Via*: Indica el servidor por el cual ha pasado la petición. Cada servidor agregará un nuevo campo *Via* con su dirección al principio del encabezado. Al procesar las respuestas, los mensajes pasarán por los mismos servidores en orden inverso. El campo agregado por un servidor se remueve antes de enviar el mensaje al siguiente servidor.
- *To*: Indica el destinatario “lógico” de una petición, que puede ser el destinatario final de la petición, o un destinatario intermedio que permite alcanzar el usuario o servicio objetivo de la petición.
- *From*: Indica la identidad “lógica” del usuario que originó la petición.
- *Call-ID*: Es un identificador único que sirve para agrupar los mensajes pertenecientes a un diálogo.
- *CSeq*: Es un identificador del orden de los mensajes dentro de una transacción.

Cuando se habla de “lógico”, se hace referencia a alguno de los componentes que conforman la estructura del protocolo SIP, que se tratarán en la sección 3.1.5 de este capítulo.

Además de los campos mencionados en párrafos previos, las peticiones poseen el campo *Max-Forwards*, que sirve para limitar el número de saltos por los que puede transitar un mensaje antes de ser descartado (similar al campo *Time to Live* en el encabezado de IPv4).

A la mayoría de los campos también se les puede asociar parámetros, que aparecen después del valor del campo, siendo separados por el carácter punto y coma “;”. Los parámetros se componen de un nombre de parámetro y un valor para ese parámetro, siendo estos separados por el símbolo igual “=”.

3.1.4 Cuerpo del mensaje SIP

Como se mencionó anteriormente, el cuerpo del mensaje SIP es opcional. Cuando este se encuentre en un mensaje, su interpretación dependerá del tipo de mensaje (10). Si se trata de un mensaje de petición, la interpretación dependerá del modo utilizado; mientras que, si se trata de un mensaje de respuesta, dependerá del par conformado por el código de estado y el modo de la petición. A modo de ejemplo, si arriba un mensaje de petición INVITE, la información del cuerpo se interpretará como los parámetros que soporta el remitente para el establecimiento de la comunicación. Si arriba un mensaje de respuesta de código 200 (OK) generado por una petición INVITE, el cuerpo se interpretará como los parámetros de comunicación que soporta el cliente al que se está llamando.

Normalmente, en el cuerpo de un mensaje SIP se encapsula el protocolo SDP (*Session Description Protocol*) con el propósito de incluir una descripción de datos de la sesión.

3.1.4.1 Protocolo SDP

El protocolo SDP es utilizado para describir sesiones en tiempo real y su especificación se puede leer de (13). SDP provee una representación estándar para transmitir metadatos de la sesión a los participantes correspondientes. Fue diseñado originalmente para anunciar la información necesaria respecto de la sesión a los participantes. Actualmente, se lo utiliza no solo para el anuncio, sino también para la negociación de los parámetros

de la sesión multimedia, como pueden ser el tipo de dato multimedia a transportar (audio, video), el protocolo a través del cual se transportaran los datos (RTP/UDP, RTP/TCP, H.320), el formato del dato transportado (audio GSM, video MPEG) y la dirección y puerto a la cual se enviarán esos datos.
Al igual que SIP, los mensajes SDP se basan en texto plano utilizando el conjunto de caracteres UTF-8. Estos se componen de uno o más campos. Cada uno de ellos posee un nombre abreviado por una única letra, seguido del signo igual “=” y el valor del campo. Los campos que acepta el protocolo se ilustran en la Tabla 3.2 – Campos del mensaje SDPTabla 3.2 (13).

Descripción de la Sesión	
v=	Versión del protocolo.
o=	Origen e identificador de sesión.
s=	Nombre de sesión.
i=*	Información de la sesión.
u=*	URI de descripción.
e=*	Correo electrónico.
p=*	Número telefónico.
c=*	Información de conexión.
b=*	Cero o más líneas de información de ancho de banda.
Una o más líneas de descripción de tiempo (“t=” y “r=”)	
z=*	Ajuste de zona horaria.
k=*	Clave de encriptación.
a=*	Cero o más líneas de atributos de sesión.
Cero o más líneas de descripción de medios.	
Descripción de Tiempo	
t=	Tiempo durante el cual la sesión estará activa.
r=*	Cero o más veces de repetición
Descripción de Medios, si está presente	
m=	Nombre de medio y dirección de transporte
i=*	Título.
c=*	Información de conexión.
b=*	Cero o más líneas de información de ancho de banda.
k=*	Clave de encriptación.
a=*	Cero o más líneas de atributos de sesión

*Campos opcionales

Tabla 3.2 – Campos del mensaje SDP

3.1.5 Componentes

Dentro de la arquitectura del protocolo, existen elementos que soportan las funcionalidades mencionadas e interactúan entre sí para poder llevarlas a cabo. Estos

elementos se los puede clasificar en dos tipos: físicos y lógicos. Los elementos lógicos son aquellos que pueden comportarse como servidor, cliente o ambos simultáneamente; mientras que los elementos físicos implementan uno o más elementos lógicos.

El protocolo dispone de dos elementos fundamentales (10), ambos entes lógicos; los Agentes de Usuario (User Agent o UA) y los Servidores.

Los Agentes de Usuario o UA (*User Agent*) son elementos ubicados en los extremos del protocolo. Estos pueden actuar como clientes, denominados UAC (*User Agent Client*); o como servidores, denominados UAS (*User Agent Server*). Los UAC son los que generan las peticiones SIP, mientras que los UAS son aquellos que generan respuestas a esas peticiones.

Existe un tercer tipo de UA, denominado B2BUA (*Back-to-Back User Agent*), los cuales reciben las peticiones y las procesan como un UAS, pero, para determinar cómo se debe responder, se comportan como un UAC y generan nuevas peticiones. El uso más común para este tipo de UA es para funciones de control, como puede ser el caso del saldo del usuario, o el tiempo restante de una llamada en un teléfono público.

En cuanto a los servidores, (10) define tres tipos a implementar. El primero de ellos es el Servidor de Redirección (*Redirect Server*), que se encarga de encaminar las peticiones al destino correcto, mediante las diferentes respuestas de redirección. Luego, se define el Servidor de Registro (*Registrar Server*), el cual se encarga de aceptar las peticiones de registro de los usuarios, almacenando la información de contacto contenida en estos mensajes para poder localizarlos. Finalmente, se define el Servidor Proxy (*Proxy Server*), cuyo objetivo es retransmitir las peticiones y respuestas hasta otro componente cercano al destino final, modificando los campos de los mensajes en caso de ser necesario. Este es un elemento intermedio que puede actuar como cliente y servidor. Existen dos tipos de este servidor: el Proxy con Estado (*Statefull Proxy*) el cual mantiene el estado de las transacciones durante el procesamiento de las peticiones; y el Proxy sin Estado (*Stateless Proxy*), el cual no mantiene el estado de las transacciones.

3.2 Protocolo RTP

El Protocolo de Transporte en Tiempo Real o RTP es un protocolo de red para el envío de datos en tiempo real; como pueden serlo el audio y el video, y análogamente al protocolo SIP opera a nivel de capa de sesión respecto del modelo de referencia OSI. Fue desarrollado por la IETF, y actualmente se encuentra en vigencia la versión 2 (14).

Se utiliza en conjunto con el Protocolo de Control de Transporte en Tiempo Real o RTCP. Mientras RTP se encarga del envío de flujos multimedia, RTCP monitorea y realiza estadísticas de la transmisión respecto a la calidad del servicio.

El protocolo RTP puede ser transportado indistintamente mediante los protocolos TCP o UDP, aunque es más utilizado en conjunto con UDP, ya que los requerimientos temporales de las transmisiones en tiempo real no aceptan los retardos causados por el establecimiento y finalización de una conexión. RTP no posee un número de puerto bien conocido, en su lugar, utiliza puertos no privilegiados (1024 a 65535). Conforme al RFC 3550 (14), y para el protocolo UDP, se recomienda que RTP utilice un número de

puerto par, mientras que RTCP debería utilizar el siguiente mayor número de puerto.

Una sesión RTP consiste de todos los flujos de datos multimedia de todos los participantes que se comunican a través del protocolo. Cada sesión se identifica mediante la dirección de transporte, que consta de una dirección de red y un par de puertos RTP y RTCP respectivamente.

3.2.1 Encabezado RTP

El encabezado RTP posee un formato fijo (Fig. 3.2), aunque existe la posibilidad de extenderlo con otros campos en caso de que sea necesario enviar información adicional. No se hará referencia a esta extensión, ya que en el presente trabajo no fue necesaria su implementación.

Bit Offset	**0-1**	**2**	**3**	**4-7**	**8**	**9-15**	**16-31**
0	V	P	X	CC	M	PT	N° Secuencia
32	Timestamp						
64	Identificador SSRC						
96	Identificador CSRC						

Fig. 3.2 – Encabezado fijo del datagrama RTP

Los campos CC y CSRC son utilizados únicamente cuando se multiplexan múltiples flujos, funcionalidad no implementada en el trabajo de tesis. A continuación se explica el significado de los campos restantes:

- *Versión (V)*: Este campo identifica la versión de RTP. Conforme (14), el valor del campo será 2.
- *Relleno (P)*: Bit que indica que al final del paquete se encuentran cero o más octetos de *relleno* que no forman parte del cuerpo.
- *Extensión (X)*: Este bit, si se encuentra activado, indica que luego del encabezado fijo debe ir exactamente un encabezado de extensión.
- *Marker (M)*: Campo de un bit que indica que el dato transportado tiene una relevancia especial para la aplicación. Como ejemplo, el primer paquete del flujo RTP o el último.
- *Tipo de Payload (PT)*: Identifica el tipo de dato transportado (audio, video, texto) y el formato (GSM, MPEG) del cuerpo del paquete, además de determinar la interpretación que la aplicación dará al cuerpo.
- *Número de Secuencia*: El número de secuencia aumenta por cada paquete RTP enviado, y el receptor puede utilizarlo para detectar la pérdida de paquetes y para restaurar el orden de los mismos. Por cuestiones de seguridad, el valor inicial del número de secuencia se elegirá aleatoriamente.
- *Timestamp*: Este campo refleja el instante de la muestra del primer octeto en el

paquete. Ese instante debe ser calculado por un reloj que se incrementa monótonamente y linealmente en el tiempo, para permitir la sincronización y el cálculo del jitter.

- *Identificador SSRC*: Campo utilizado para identificar, de forma única, la fuente de un flujo dentro de una sesión. Este identificador es elegido al azar, con la intención de que el valor sea único entre todas las fuentes que conforman la sesión.

3.3 Intercambio de mensajes

Como la central VoIP a desarrollar trabajará con SIP como el protocolo para el establecimiento de las sesiones, se cree conveniente el uso de ejemplos para afianzar el conocimiento expuesto anteriormente.

Los ejemplos mostrarán un escenario con un único servidor Proxy, de manera que el flujo de mensajes quede lo más sencillo posible. Este Proxy se representará mediante una nube, ya que puede tratarse de un conjunto de servidores, los cuales los usuarios verán como si fuese uno solo.

3.3.1 Registración de un usuario

En la Fig. 3.3, se ilustra el intercambio de mensajes involucrados en el proceso de registración del usuario, compuesto por dos transacciones.

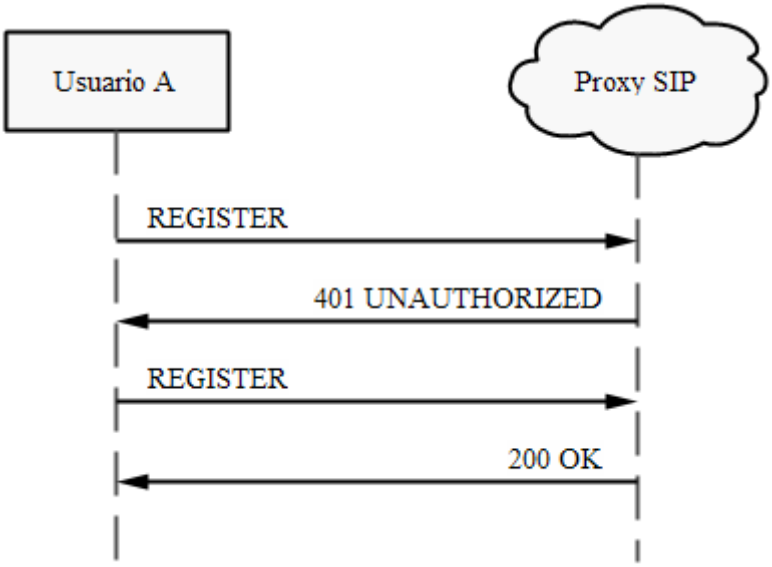

Fig. 3.3 – Proceso de registración de un usuario

En la primer transacción, el Usuario A le envía una petición REGISTER al servidor. Para poder registrarlo, el servidor necesita información de autenticación para comprobar la identidad del usuario, pero este no ha enviado dicha información. Por esta razón, el servidor responde con UNAUTHORIZED (código 401); el cual posee un campo denominado *WWW-Authenticate* que contiene un desafío solicitando la información para poder autenticarle. En este intercambio de mensajes finaliza la primera transacción.

Luego, en la segunda transacción, el Usuario A construye su información de autenticación y responde al desafío con un nuevo mensaje REGISTER, el cual contendrá el campo denominado *Authorization* en el que se almacena la información para poder autenticarse. Con la información necesaria, el servidor autentica al usuario, y responde con un mensaje de OK (código 200).

3.3.2 Llamada exitosa

En el caso de una llamada que se ha concretado exitosamente (Fig. 3.4), esta se conforma por cuatro transacciones.

La primera transacción corresponde al establecimiento de la sesión. El Usuario A envía una petición INVITE al Proxy, solicitando comunicación con el Usuario B. El Proxy responde, de manera análoga al ejemplo de la Fig. 3.3, con un PROXY AUTHENTICATION REQUIRED (código 407) informándole al Usuario A que falta la información para poder comprobar su identidad. Este mensaje contiene un campo denominado *Proxy-Authenticate*, el cual incluye un desafío solicitando la información necesaria para poder autenticarle. El Usuario A responde con un ACK informando que recibió el mensaje, y finaliza la transacción.

Luego, en la segunda transacción, el usuario A vuelve a enviar un INVITE pero con un nuevo campo, denominado *Proxy-Authorization*. Este campo contiene la información de autenticación del usuario. Inmediatamente, el Proxy responde con un TRYING (código 100) indicando que se ha recibido el mensaje y se comienza su procesamiento, además de evitar las retransmisiones por parte del Usuario A. Posteriormente, envía la petición al Usuario B.

El Usuario B puede, aunque no de forma obligatoria, responder con un TRYING al Proxy por el mismo motivo mencionado en el párrafo anterior. Luego, envía un RINGING (código 180) al Proxy una vez que el teléfono empieza a sonar, y este último lo reenvía al Usuario A. Cuando el Usuario B atiende el teléfono, este le envía un OK (código 200) al Proxy, el cual a su vez envía al Usuario A finalizando la segunda transacción.

Finalmente, el Usuario A inicia una pequeña transacción enviando un ACK al Proxy, el cual emula el comportamiento y envía un ACK al Usuario B, finalizando la transacción e iniciando la llamada. En esta transacción es cuando se negocian los parámetros de la sesión multimedia mencionados en la sección 3.1.4.1.

Una vez establecida la llamada, empieza el intercambio de audio mediante el protocolo RTP, utilizando los diferentes parámetros (direcciones, puertos, codecs) negociados a través del protocolo SDP.

La última transacción corresponde a la finalización de la sesión, en este caso, por parte del Usuario A. Ambos usuarios están habilitados para empezar esta transacción. El Usuario A envía un mensaje BYE al Proxy, el cual reenvía al Usuario B. Este responde con un OK, indicando la recepción correcta del mensaje, y finalizando así el diálogo.

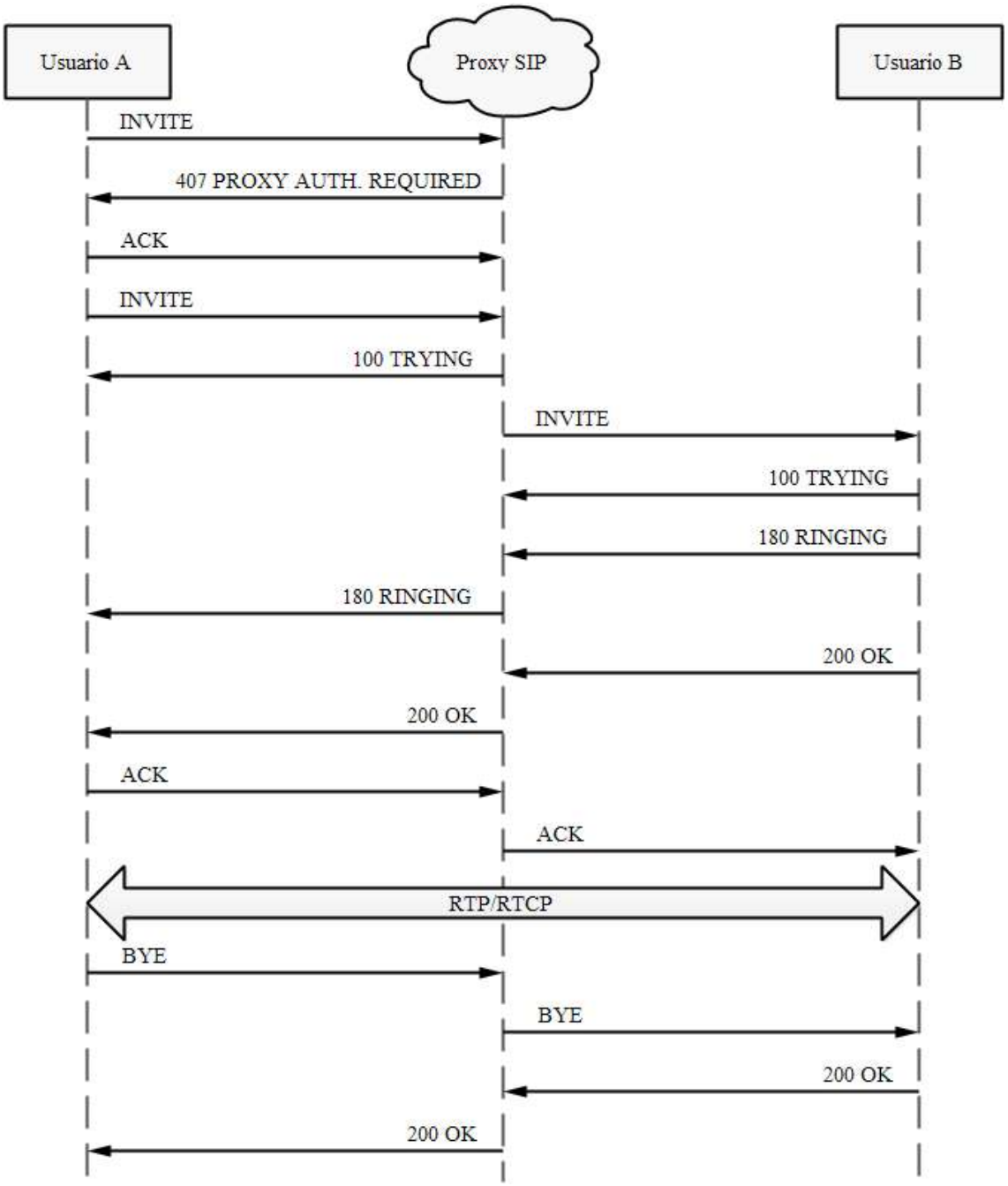

Fig. 3.4 – Llamada exitosa entre usuarios

Capítulo 4
"Tecnologías"

4.0 Introducción

Como se mencionó en la sección 1.2, muchos componentes de software empleados en trabajos previos relativos al diseño e implementación de una central telefónica VoIP en ambientes de producción, quedaron obsoletos o sufrieron actualizaciones para obtener mejoras en rendimiento y nuevas funcionalidades. Todo lo descrito conlleva a un exhaustivo rediseño respecto de las tecnologías de software que se utilizarán.

En este capítulo se presentarán las aplicaciones de software escogidas para el desarrollo de la central VoIP, que incluyen un conjunto de las tecnologías de software actualizadas respecto de los componentes utilizados en (2) y (3), y otras que constituyen nuevas tecnologías de software para agregar nuevas funcionalidades a la central telefónica a diseñar.

A continuación, en las primeras tres secciones se presentarán las aplicaciones de software utilizadas para el módulo de bases de datos. En la sección 4.1; se presentará MariaDB, un sistema gestor de bases de datos, utilizado para almacenar información de configuración de las demás tecnologías de software, en reemplazo de MySQL utilizado en (3). Posteriormente, en la sección 4.2; se describirá la aplicación Galera Cluster, que permitirá la creación de clústeres de bases de datos multi-maestro con redundancia activo/activo, permitiendo resolver los problemas descritos en la sección 1.2 en lo que se refiere al módulo de bases de datos (3). Luego, en la sección 4.3 se explicará el software Galera Load Balancer, un simple y efectivo balanceador de conexiones TCP, utilizado para implementar un esquema de redundancia activo/activo ofrecido por Galera Cluster y así evitar el problema de sobrecarga de la implementación (3).

En la sección 4.4, se presenta Asterisk, una aplicación con funcionalidades de PBX, que de igual manera que (2) y (3), proveerá las principales funcionalidades de la central telefónica, al conformar los módulos de registro y localización.

En las siguientes dos secciones se presentarán las tecnologías de software empleadas en el módulo de balanceo de carga. Debido a que en (3) se utilizó un balanceador de conexiones TCP/UDP genérico que no ha sido actualizado en los últimos años, se decidió reemplazarlo por dos aplicaciones capaces de manejar el tráfico de los protocolos mencionados en el Capítulo 3. En la sección 4.5, se describirá la herramienta de software Kamailio, un servidor del protocolo SIP que ofrece gran variedad de funcionalidades para el control y manejo de los mensajes SIP, entre ellas, el balanceo de carga. Luego, la sección 4.6 tratará sobre la aplicación RTPProxy, utilizado para el manejo del flujo de mensajes RTP, la cual trabajará en conjunto con Kamailio en el tratamiento del tráfico entrante de la central telefónica.

Finalmente, en la sección 4.7 se presentará Pacemaker, la herramienta de software utilizada para dotar de alta disponibilidad a la central telefónica, asegurando el continuo

funcionamiento de las aplicaciones de software mencionadas anteriormente ante la ocurrencia de fallas.

4.1 MariaDB

MariaDB (15) es un SGBD (*Sistema de Gestión de Bases de Datos*) bajo licencia GPL, presentado como el sustituto mejorado de su progenitor, MySQL. Ambos proyectos fueron generados por Michael Widenius, quien decidió introducir la nueva variante luego de que los derechos de su primera creación fueran adquiridos por la compañía Oracle. Widenius desarrolló MariaDB cuando Oracle adquirió los derechos de MySQL y presintió que la empresa tenía intenciones de reducir la competencia para imponer sus productos propietarios.

Un SGBD es una colección de aplicaciones de software que interactúan con los usuarios y otras aplicaciones para definir, construir y manipular una Base de Datos, asegurando integridad, confiabilidad y seguridad.

4.1.1 Características

MariaDB ha conseguido un gran reconocimiento en un breve período de tiempo, esto se debe, en gran medida a que es esencialmente un reemplazo binario de MySQL. La compatibilidad entre ambas tecnologías es tal que permite a los usuarios migrar de software de un modo transparente. MariaDB incorpora nuevas características respecto de MySQL que es necesario destacar.

El primer aspecto fundamental es relativo a la seguridad. Los desarrolladores de MariaDB producen sus propias actualizaciones de seguridad, que incluyen un análisis exhaustivo de las actualizaciones de MySQL a los efectos de mejorarlas en caso de ser necesario. Cuando se descubre una vulnerabilidad crítica, los desarrolladores modifican el código fuente y lo distribuyen inmediatamente para eliminarla.

Como se mencionó a principio de esta sección, la compatibilidad es otro aspecto esencial a destacar. MariaDB contiene los comandos, interfaces, bibliotecas y APIs utilizados por MySQL; a la vez que incorpora nuevas características no contempladas en su antecesor (MySQL).

MariaDB ofrece tres versiones distintas; una estable, ideal para sistemas de producción, una de prueba *(testing)* y otra para desarrolladores. Cada uno de las versiones es supervisado por una comunidad autorizada que brinda distintos niveles de soporte.

Además de lo mencionado, al tratarse de un SGBD, MariaDB provee otros aspectos y funcionalidades que se pueden encontrar en cualquiera de estos sistemas. Un SGBD ahorra a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario; definiéndose varios niveles de abstracción. También genera independencia de los datos, lo cual consiste en la capacidad de modificar el esquema, físico o lógico; de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella. Finalmente mantiene la consistencia de los datos al monitorear que cualquier modificación introducida se actualice de forma coherente, en todas las réplicas utilizadas.

Por otra parte, la base de datos representa una realidad que tiene determinadas condiciones. Por ejemplo, en un sistema sobre licencias de conducir que utilice una base de datos, el sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.

La seguridad también se ve reflejada en un SGBD, ya que la información almacenada en una base de datos puede llegar a tener un gran valor. Estos sistemas garantizan que la información se encuentra segura mediante la aplicación de permisos a usuarios y grupos, generando diversas categorías de privilegios. Además, brindan el soporte de transacciones, que consisten en un programa que se ejecuta como una sola operación, es decir, si esa ejecución por cualquier motivo falla, el resultado que se obtendrá es el mismo que si no se hubiera ejecutado.

Finalmente, un SGBD provee distintos mecanismos que facilitan la manipulación de los datos, y procura con ellos brindar tiempo de respuestas mínimos, ante las distintas solicitudes que puede recibir el sistema.

4.2 Galera Cluster

Galera Cluster (16) es un clúster multi-maestro síncrono para sistemas de base de datos. En un alto nivel de abstracción, consiste en un servidor de base de datos distribuido que expande los módulos de replicación provistos por motores como MariaDB o MySQL, con el fin de proporcionar toda la información necesaria para garantizar la replicación síncrona a los miembros del clúster, permitiendo el almacenamiento de datos en cualquier nodo miembro. Esta API extendida es llamada WSREP (*Write-Set Replication*).

Cuando se utiliza esta tecnología, los usuarios y aplicaciones pueden leer o escribir en cualquier nodo indiferentemente, e incluso garantiza que sus operaciones no se verán interrumpidas por fallas individuales en el sistema. Sólo se encuentra disponible para distribuciones Linux, y soporta únicamente los mecanismos de almacenamiento XtraDB e InnoDB (17).

4.2.1 Características

Galera Cluster permite replicación síncrona, topología multi-maestro con redundancia activo/activo, control de membresía automático, replicación paralela a nivel de fila, conexiones directas con los clientes, reducción de latencia (retardos), escalabilidad, pérdidas nulas de transacciones y esclavos sin retraso.

4.2.2 Funcionamiento

Una operación de lectura requiere que todos los nodos mantengan consistencia respecto a los datos almacenados. Por este motivo cada vez que ocurre un evento de escritura, se dispara automáticamente una transacción replicada en forma simultánea al conjunto de servidores que componen la base distribuida. La Fig. 4.1, ilustra el mencionado comportamiento.

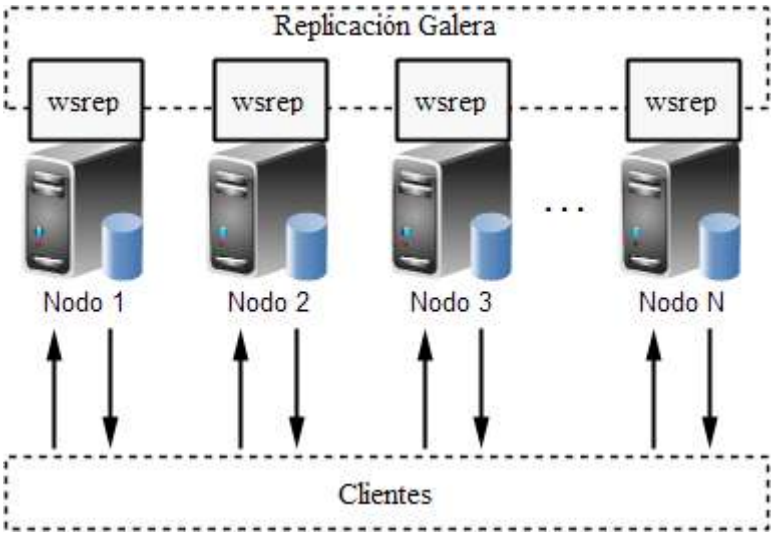

Fig. 4.1 – Replicación de Galera Cluster

Si, por cualquier razón, alguno de los nodos no puede completar el proceso, el clúster evalúa la situación y ejecuta las acciones adecuadas. Por ejemplo, el sistema puede aislar a un nodo que experimenta fallas, y permitir que la transacción permanezca activa para que el servicio funcione sin interrupciones. En el peor caso, bajo los fundamentos de un mecanismo de quorum administrable en su número, cuando una falla afecte a un conjunto de nodos que impida llegar al nivel de quorum previsto, el clúster no aceptará peticiones adicionales de escritura hasta que no alcanzar la mayoría de votos estipulados.

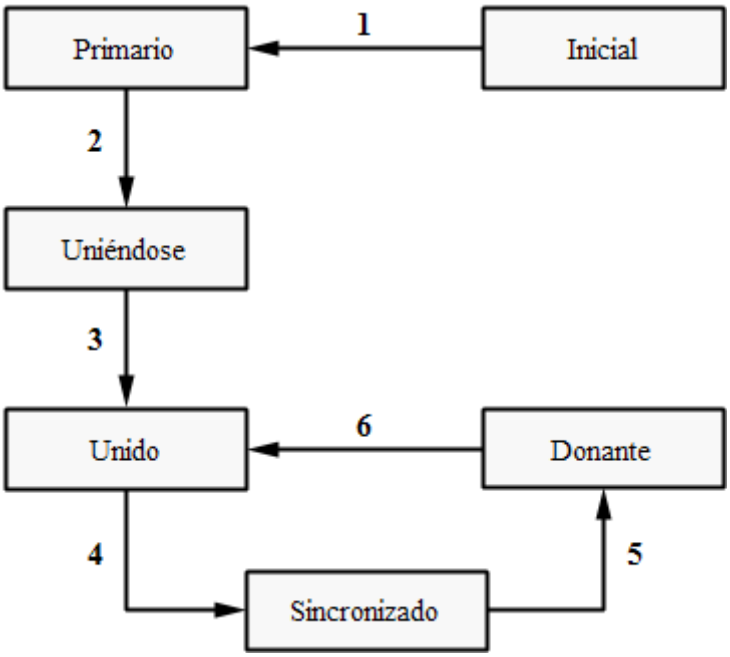

Fig. 4.2 – Máquina de Estados de Galera Cluster

En un sistema conformado con Galera Cluster, cada uno de sus miembros adopta un rol que determinará la función que desempeñará en el clúster. La Fig. 4.2 ilustra la máquina de estados en que basa su funcionamiento. Cuando un nodo inicia su actividad,

establece una conexión con el componente primario (conjunto de nodos capaces de comunicarse entre sí, y está conformado por la mayoría de los nodos). En el momento en el que el nodo recibe una respuesta de aceptación desde el nodo primario, comienza el proceso de unión al clúster, y almacena las peticiones de escritura recibidas, pero no las aplica.

Una vez el clúster entregue una imagen del estado del sistema, el nodo tendrá toda la información necesaria para unirse al grupo. En este punto se habilita un mecanismo de retroalimentación, denominado “flujo de control”, que permite que el emisor de la imagen detenga su ejecución para no generar retrasos en las transacciones. Terminada la transferencia, el nuevo integrante comenzará a aplicar las transacciones almacenadas y tendrá los mismos datos que el resto de los nodos participantes. Se convertirá en un nodo sincronizado, habilitado para participar como otro miembro en las operaciones de escritura.

Eventualmente, cualquier nodo sincronizado puede recibir una consulta de transferencia de estado que lo convertirá en un nodo donante que almacenara las transacciones pero no las aplicará hasta no terminar la transferencia. Una vez completada la transferencia vuelve a unirse al clúster, y finalmente administrará las transacciones para volver al estado sincronizado.

4.3 Galera Load Balancer

Abreviado GLB (18), Galera Load Balancer es un simple balanceador de carga, diseñado específicamente para trabajar con sistemas que implementen Galera Cluster. Su diseño está inspirado en el proyecto Pen, pero a diferencia de este, su funcionalidad está limitada a balancear conexiones TCP genéricas.

4.3.1 Características

GLB es un balanceador de carga sencillo, capaz de brindar un pequeño conjunto de funcionalidades que repercuten diminutamente en el consumo de recursos de hardware. Entre ellas, se distingue la posibilidad de configurar en tiempo de ejecución la lista de servidores de base de datos involucrados en el balanceo de carga y el soporte para la anulación de los servidores. También permite el drenado de las conexiones de un servidor, que consiste en negar la asignación de nuevas conexiones sobre alguna de las base de datos, y esperar a que concluyan las conexiones ya existentes.

GLB admite distintas políticas de balanceo, que individualmente pueden ser personalizadas mediante la asignación de prioridades sobre los distintos puntos de balanceo. Por defecto, el nivel de preferencia es equivalente en todos los nodos involucrados. Además, provee un módulo opcional para el monitoreo de los destinos y el ajuste automático de la lista de servidores.

4.3.2 Funcionamiento

El balanceo de carga sobre un sistema Galera Cluster, varía de acuerdo a las necesidades de las aplicaciones y/o usuarios que requieran de sus servicios. GLB, se ajusta a las distintas configuraciones ofreciendo un variado conjunto de políticas de balanceo de carga.

Por defecto, ofrece el mecanismo “*least connected*” (Fig. 4.3) en el cual las solicitudes de conexión se envían al servidor que posee menor cantidad de vínculos establecidos. Adicionalmente brinda otros mecanismos como el métodos “*single*”, en el que todas las solicitudes de conexiones se envían al servidor de mayor prioridad; “*Round-Robin*” en el que cada solicitud de conexión entrante es asignada al próximo servidor en una lista circular; un mecanismo “*random*”, en que una solicitud de conexión entrante es asignada en forma aleatoria entre los servidores activos; y por último el mecanismo “*source tracking*”, en que las peticiones originadas desde un mismo origen son dirigidas al mismo servidor.

Otro mecanismo de balanceo interesante es “*top*”, el cual restringe las conexiones a servidores de idéntica prioridad, y se utiliza en conjunto con una de las políticas mencionadas en párrafos previos. Como ejemplo de su funcionamiento, si se tuviesen servidores agrupados en prioridades de 1 y 2, todo el balanceo se realizaría sobre el conjunto de nodos cuya preferencia es 2, y solamente se considerarían los destinos de prioridad 1 cuando no exista disponibilidad de servidores con prioridad 2.

GLB trabaja por defecto en el puerto 8010. Cuando recibe una petición de un cliente, la reenvía (de acuerdo a la política seleccionada) hacia uno de los servidores de base de datos previamente configurados. Una vez procesada la consulta, la respuesta es entregada al balanceador para que este la reenvíe al cliente correspondiente.

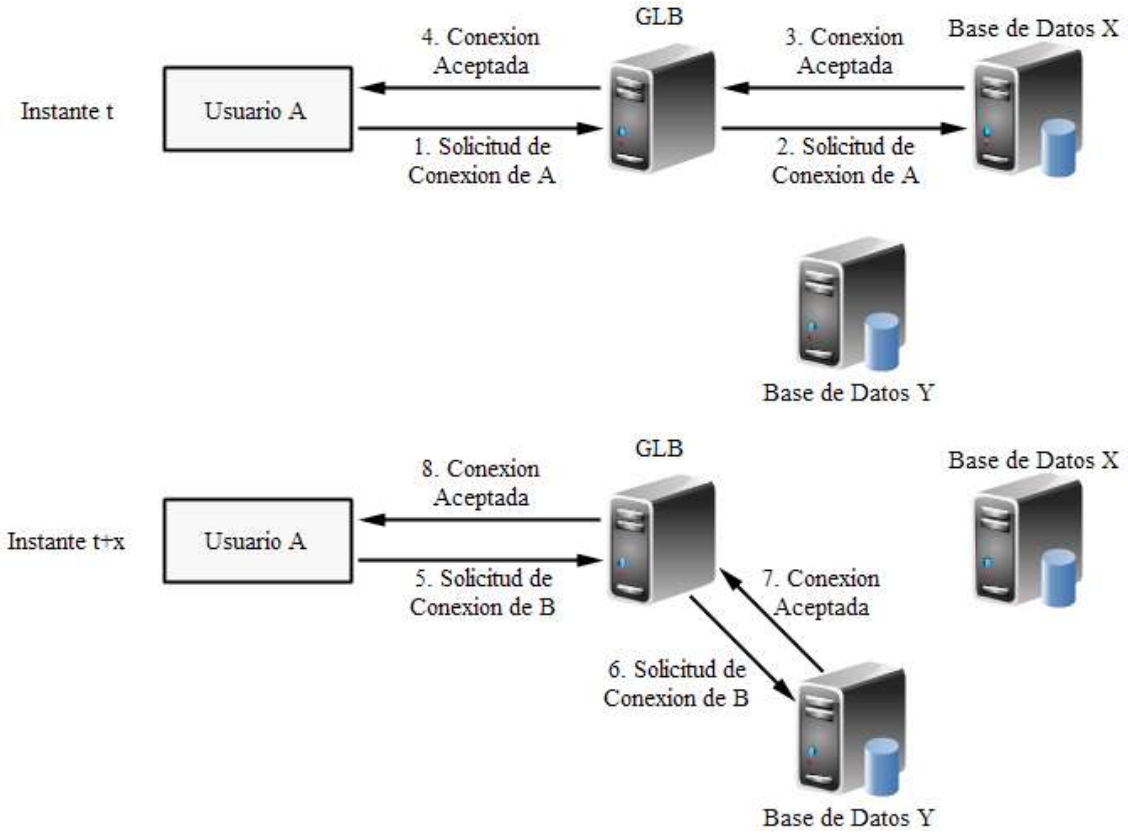

Fig. 4.3 – Funcionamiento de GLB utilizando la política de balanceo *“least connected”*

4.4 Asterisk

Mark Spencer, fundador de la empresa Digium, creó inicialmente Asterisk (4) y actualmente es su principal desarrollador junto con otros programadores que contribuyen a actualizar el producto para mejorar el desempeño y añadir nuevas funcionalidades. Digium define a Asterisk como “una caja de Legos para aquellas personas que quieren crear aplicaciones de comunicación”, debido a que permite construir una solución que se ajuste a las necesidades de cada situación particular.

Asterisk es un software bajo licencia GPL, con la capacidad de brindar y superar los servicios que anteriormente sólo estaban disponibles en costosas centrales telefónicas propietarias. Los usuarios pueden ampliar sus características escribiendo un plan de marcado en el lenguaje script característico de Asterisk, o añadiendo módulos de programación que sean soportados en GNU/Linux. Uno de los puntos fuertes de esta aplicación es que permite unificar las tecnologías de VoIP, GSM (*Global System for Mobile communication*) y PSTN.

4.4.1 Funcionalidades

Asterisk incorpora todas las funcionalidades que caracterizan a una central telefónica analógica convencional y opciones avanzadas que tendrían un elevado costo en sistemas tradicionales propietarios. Las diferentes funciones están constituidas como módulos de software que se pueden activar o desactivar, según las necesidades planificadas en la etapa de diseño. En esta sección se hace uso del concepto de extensión, que en el contexto de un sistema telefónico, hace referencia a un identificador numérico que al ser marcado, provocará una colección de acciones tales como la de establecer una llamada a la extensión marcada En las secciones 4.4.1.1 y 4.4.1.2 se nombran las funcionalidades más relevantes, agrupadas de acuerdo a su uso y complejidad.

4.4.1.1 Funcionalidades básicas

Las funcionalidades básicas de Asterisk refieren a aquellos aspectos de la central que suelen ser de uso común y que consecuentemente se encuentran implementadas en la mayoría de los servidores. Dentro de este conjunto se pueden encontrar el soporte para transferencias, que consisten en redirigir una llamada en curso hacia otra extensión y que pueden ser de dos tipos; atendidas, en que se consulta al nuevo destinatario si desea recibir la llamada; o a ciegas, donde se redirige la llamada sin notificar al nuevo destinatario. Una variante de esta características son los desvíos de llamadas, que permiten la transferencia automática de una llamada entrante hacia una extensión explicita cuando se cumplen determinadas condiciones, por ejemplo, si el destino objeto de la llamada está ocupado o no contesta.

Asterisk soporta capturas de llamadas, que consiste en tomar una llamada que se está recibiendo en una extensión desde otra extensión distinta. Esta característica puede ser individual, o pueden definirse grupos de manera que un miembro del grupo puede tomar cualquier llamada dirigida al grupo. Del mismo modo, el concepto de grupo es utilizado en muchas otras situaciones como pueden ser las conferencias para establecer comunicaciones entre múltiples usuarios, o la redirección de llamadas entrantes a un grupo de extensiones de acuerdo a una estrategia previamente configurada.

4.4.1.2 Funcionalidades avanzadas

Las funcionalidades avanzadas de Asterisk refieren a aquellos aspectos de la telefonía que suelen ser de uso opcional o que requieren de varias configuraciones adicionales, que generalmente no se encuentran o son de un elevado costo en las centrales analógicas.

Dentro de este conjunto, el más reconocido es el correo de voz, el cual consiste en permitirle al usuario que no se ha podido comunicar con otro usuario, dejarle un mensaje de voz para indicarle el motivo de la llamada. Un atractivo que incorpora Asterisk a esta funcionalidad es, la posibilidad de enviar un correo electrónico con el mensaje de voz adjunto al usuario que ha sido llamado.

Otras opciones de amplio uso provistas por Asterisk son la música en espera y la operadora automática en complemento con las extensiones DISA (*Direct Inward System Access*). La música en espera permite introducir categorías de sonidos en formato “wav” y “mp3”, de manera que se puedan seleccionar y reproducir cuando una llamada ingrese al sistema en modo de espera. La operadora automática permite al sistema interactuar con los usuarios que realizan una llamada, de forma que estos puedan seleccionar entre opciones previamente configuradas y acceder automáticamente a los destinos programados. En complemento con la operadora, DISA posibilita configurar opciones de post marcación para determinadas llamadas entrantes, de forma que una vez establecida la comunicación con la central, permite el enrutamiento a un nuevo destino de forma sencilla y automática.

Adicionalmente, algunos servicios interesantes que no son de uso frecuente son la gestión de llamadas entrantes con condiciones de tiempo, que consiste en definir un horario y calendario que permita hacer un tratamiento diferenciado de las llamadas entrantes; la llamada automática de respuesta a una llamada perdida, que posibilita que el sistema declare como perdida una llamada entrante, y consecuentemente establezca otra con destino al número que solicitó la comunicación con el fin de centralizar costos; y por último la retro llamada, que permite que si se realiza una llamada a una extensión y esta no contesta por encontrarse ocupada o ausente, se genera una notificación al usuario que inició la llamada cuando el destino solicitado esté disponible.

Asterisk también brinda otros mecanismos que extienden las capacidades de algunas de sus funciones básicas. Estos son la cola de llamada, que distribuye las mismas entre un grupo específico de agentes de acuerdo a una determinada estrategia, y que permite programar su transferencia a otro destino en caso de que alguna no sea atendida; y las salas de audio/video conferencias, que posibilitan conectar múltiples usuarios en una misma conversación, donde los interesados pueden acceder a la sala desde una extensión interna o bien desde el exterior.

Finalmente, esta tecnología proporciona herramientas para la generación de informes con detalles de llamadas realizadas y/o recibidas por extensión o cliente, útiles por ejemplo, para facturación de servicios. Además, es compatible con sistemas informatizados, permitiendo por ejemplo, ejecutar una llamada desde una computadora o bien recibir información sobre la misma en pantalla.

4.5 Kamailio

Kamailio (19) es un servidor SIP de código abierto, sucesor de OpenSER y SER (*SIP Express Router*), con la habilidad de gestionar miles de llamadas por segundo. Puede ser utilizado para construir grandes plataformas para comunicaciones en tiempo real y VoIP. Consigue escalar fácilmente sistemas con puertas de enlace SIP-PSTN, PBX o servidores como Asterisk.

4.5.1 Características y Funcionalidades

Kamailio es capaz de adaptarse a casi cualquier esquema de necesidades gracias a su infraestructura modular y al gran número de características y funcionalidades que ofrece. El poder de esta herramienta se centra en un archivo binario muy pequeño, al punto de ser adecuado para dispositivos embebidos; y en su estructura que permite extender su potencial mediante librerías internas y una interfaz *“plug and play”*, que brinda la posibilidad de añadir una gran variedad de módulos, sin las complicaciones que implica modificar el núcleo de la aplicación. Simultáneamente, permite el desarrollo de nuevas aplicaciones con una diminuta curva de aprendizaje, utilizando para ello sus interfaces de programación con los lenguajes C, Java, Perl, Python y Lua. Debido a su interfaz con Servlets Java, también es posible extender e integrar los servicios de VoIP con otros servicios web.

Kamailio posee un fichero de configuración con una sintaxis muy similar a los lenguajes de scripting, lo que dota de una gran flexibilidad y potencia a la hora de desplegar soluciones SIP personalizadas. Sólo es necesario modificar el archivo para adaptar el servidor a nuevas aplicaciones o funcionalidades. El fichero cuenta con un sistema de pseudo-variables para acceder y gestionar parte de los mensajes SIP, y atributos específicos de los usuarios y el servidor. También admite la creación de variables personalizadas por parte de los usuarios para adaptarse a las necesidades que posean.

Una de las principales características de Kamailio es su robustez y estandarización frente al protocolo SIP, la cual le posibilita encaminar paquetes en un entorno VoIP y al mismo tiempo adoptar todas las entidades lógicas conocidas en este contexto como Servidor de Registro, Servidor de Redirección, Servidor Proxy y demás componentes. Está preparado para procesar tráfico SIP en modo “sin estado”, es decir, sin conservar información de los mensajes que gestiona, lo cual puede ser muy útil, por ejemplo, para el balanceo de carga. Así mismo, soporta el procesamiento de paquetes “con estado”, que guarda el curso de las transacciones SIP que atraviesan el servidor, lo cual permite desplegar gran cantidad de servicios, además de posibilitar el registro de detalles de llamadas y gran cantidad de estadísticas, que le permiten proveer un sistema de conteo de eventos y llamadas altamente configurable, que constituyen la base para el diseño de un sistema de facturación.

Bajo otro punto de vista, Kamailio es fácilmente desplegable en modo redundante, pudiendo ser configurado en diversos modos, como los descritos en la sección 2.4.1, o mediante balanceo de tráfico. Todos los datos de suscriptores, configuración de rutas e información de facturación pueden ser almacenados en bases de datos externas a los servidores, para una mayor seguridad.

En cuanto al balanceo de tráfico SIP, puede realizarse con muchos algoritmos de distribución distintos, lo que admite adaptarse a gran número de escenarios de manera eficiente y efectiva. El mecanismo de LCR (*Least Cost Routing*) implementado por Kamailio permite escoger la ruta óptima para cada destino en función del destino de una llamada. Esta configuración puede ser reemplazada en tiempo de ejecución en la base de datos, siendo aplicada de inmediato al procesamiento del tráfico.

Adicionalmente, incorpora un protocolo estandarizado de mensajería instantánea basado en SIP, llamado SIMPLE (*Session initiation protocol for Instant Messaging and Presence Leveraging Extensions*), que transmite la voluntad de una persona de entablar comunicación con otras. Esta información es reconocida en la actualidad como el “estado” de clientes en distintas herramientas de software tales como Skype o Google Talk. Para posibilitar la interacción con aplicaciones que usen este tipo de servicios, Kamailio es capaz de gestionar el tráfico SIP relativo a suscripciones y notificaciones, y puede proporcionar servicios de mensajería instantánea compatible con la gran mayoría de clientes VoIP, sin la necesidad de ningún servidor o software adicional. Complementariamente, Kamailio ofrece funcionalidades de cliente y servidor XCAP (*XML Configuration Access Protocol*), un servicio que facilita la recuperación y manipulación de las listas de contactos y recursos de usuarios; así como también compatibilidad con el protocolo MSRP (*Message Session Relay Protocol*), para transmitir series de mensajes instantáneos en un contexto de sesiones de comunicación. Además de los mencionados, propone interfaces capaces de comunicarse con servicios de mensajería instantánea como SMS y XMPP (*Extensible Messaging and Presence Protocol*).

Kamailio soporta los protocolos de transporte UDP, TCP, TLS (*Transport Layer Security*) y SCTP (*Stream Control Transmission Protocol*) sobre IPv4 e IPv6, lo que lo hace totalmente compatible con cualquier aplicación y facilita las futuras actualizaciones de la red. Además, Kamailio es compatible con WebRTC (*Web Real-Time Communication*); una API que está siendo elaborada por la W3C (*World Wide Web Consortium*) para permitir a las aplicaciones del navegador realizar llamadas de voz, chat de vídeo y uso compartido de archivos P2P sin módulos adicionales.

De forma similar a muchas aplicaciones de software en la actualidad, Kamailio es capaz de trabajar en conjunto con bases de datos. Si bien admite el almacenamiento de información en archivos de texto, también es capaz de conectarse con las bases de datos de MySQL, MariaDB, PostgreSQL, SQLite, BerkeleyDB, Oracle y cualquier otra compatible con controladores unixODBC. Todas estas interacciones pueden darse simultáneamente cuando se está funcionando con más de una fuente de información a la vez. Los datos de localización de usuarios, definición de rutas de tráfico, dominios y demás información necesaria para la operación del servicio, se pueden almacenar en bases de datos, lo que facilita en gran medida la implementación del servicio en alta disponibilidad y la integración con otros sistemas existentes.

Esta tecnología ofrece soporte para AAA (*Authentication, Authorization and Accounting*) a través de bases de datos y los protocolos DIAMETER y RADIUS. DIAMETER es un protocolo de red para la autenticación de los usuarios que se conectan remotamente a Internet a través de la conexión por línea conmutada o RTC (*Red Telefónica Conmutada*); también provee de servicios de autorización y auditoría para aplicaciones tales como acceso de red o movilidad IP. El concepto básico de esta

tecnología, cuyo desarrollo se ha basado en RADIUS; es de proporcionar una base que pueda ser extendida para proporcionar servicios de autenticación, autorización y auditoría, a nuevas tecnologías de acceso. Está diseñado para trabajar tanto de una manera local como en un estado de alerta, sondeo y captura, que le permite ofrecer servicios sumamente móviles, dinámicos, flexibles y versátiles.

Otro aspecto a destacar es la compatibilidad e interoperabilidad que Kamailio ofrece. Brinda módulos que facilitan la conexión y trabajo en conjunto, mediante puertas de enlace; con cualquier red PSTN. También es compatible con sistemas que brindan servicios sobre VoIP, como Asterisk o FreeSWITCH; y con todos los dispositivos y softwares asociados al mismo; tales como *hardphones*, *softphones*, entre otras herramientas. Así mismo, hace uso del servicio ENUM (*Electronic Number Mapping System*) para asociar números de teléfono de la red PSTN con URIs SIP, lo cual permite que los usuarios SIP puedan ser llamados desde la red analógica marcando un número de teléfono convencional. Esta prestación se implementa mediante registros especiales en servidores DNS, que ya se encuentra operativa en muchos países. Aumentando aún más la compatibilidad, permite dar servicio a diferentes dominios de usuarios en una misma máquina, pudiendo aplicar distintas reglas por cada grupo.

Respecto al apartado de seguridad, Kamailio provee bloqueo de direcciones a nivel IP, permitiendo restringir los paquetes que arriban al sistema, lo cual limita que gran cantidad de tráfico no deseado sea procesado, al mismo tiempo que reduce la vulnerabilidad ante ataques a la infraestructura. La capacidad de ocultar la topología, mediante la manipulación del encabezado SIP; también es otro punto fuerte relativo a la seguridad, ya que imposibilita a los usuarios finales conocer la topología de la red SIP del operador telefónico.

Cuenta con un módulo que hace de interfaz con el protocolo SNMP (*Simple Network Management Protocol*) entre el servidor y los servicios de gestión de la organización, lo que facilita la integración con cualquier software de monitoreo. Los problemas existentes en la configuración de Kamailio pueden ser fácilmente depurados gracias a las opciones de *debugging* y al archivo de mensajes logs que ofrece. Dispone de una interfaz de gestión implementada mediante ficheros FIFO y sockets Unix que permite la introducción de comandos en consola para consultar el estado del servicio, modificar todo tipo de parámetros y analizar estadísticas en tiempo real. Asimismo, Kamailio permite integrar su funcionalidad con múltiples aplicaciones, gracias a su entendimiento con el protocolo de RPC (*Remote Procedure Call*), a través de XMLRPC, JSONRPC, UDP o TCP.

Finalmente, además de la gran variedad de funcionalidades descritas en párrafos anteriores; otro gran atractivo para incorporar Kamailio en el desarrollo de una central VoIP es su alto nivel de escalabilidad. Gracias a muchos de los aspectos nombrados anteriormente, Kamailio puede ejecutarse en sistemas con bajos recursos, realizar balanceo de carga mediante distintos algoritmos de distribución, minimizar costos de ruteo adaptándose a casi cualquier escenario y escalar fácilmente un sistema solo añadiendo un mayor número de servidores, sin la necesidad de realizar cambios en la configuración existente. Complementariamente, provee mecanismos que facilitan la creación de ambientes de información redundante y servidores de respaldo, y opera sin mayores complicaciones sobre plataformas de VoIP distribuidas geográficamente.

4.6 RTPProxy

RTPProxy (20) es un software de alto rendimiento pensado para gestionar flujos de mensajes RTP. Originalmente fue diseñado como un intermediario que permite la realización de llamadas VoIP entre SIP UAs que se encuentren detrás de NAT. Con el transcurso del tiempo fue ganando reconocimiento y añadiendo nuevas particularidades.

En conjunto con un Proxy SIP, esta tecnología permite construir redes de VoIP complejas, optimizar el flujo de tráfico y recolectar información sobre la calidad de voz, entre otras características. Adicionalmente, brinda capacidades como grabación de llamadas, reproducción de audio pre-codificado y copia de flujos en tiempo real.

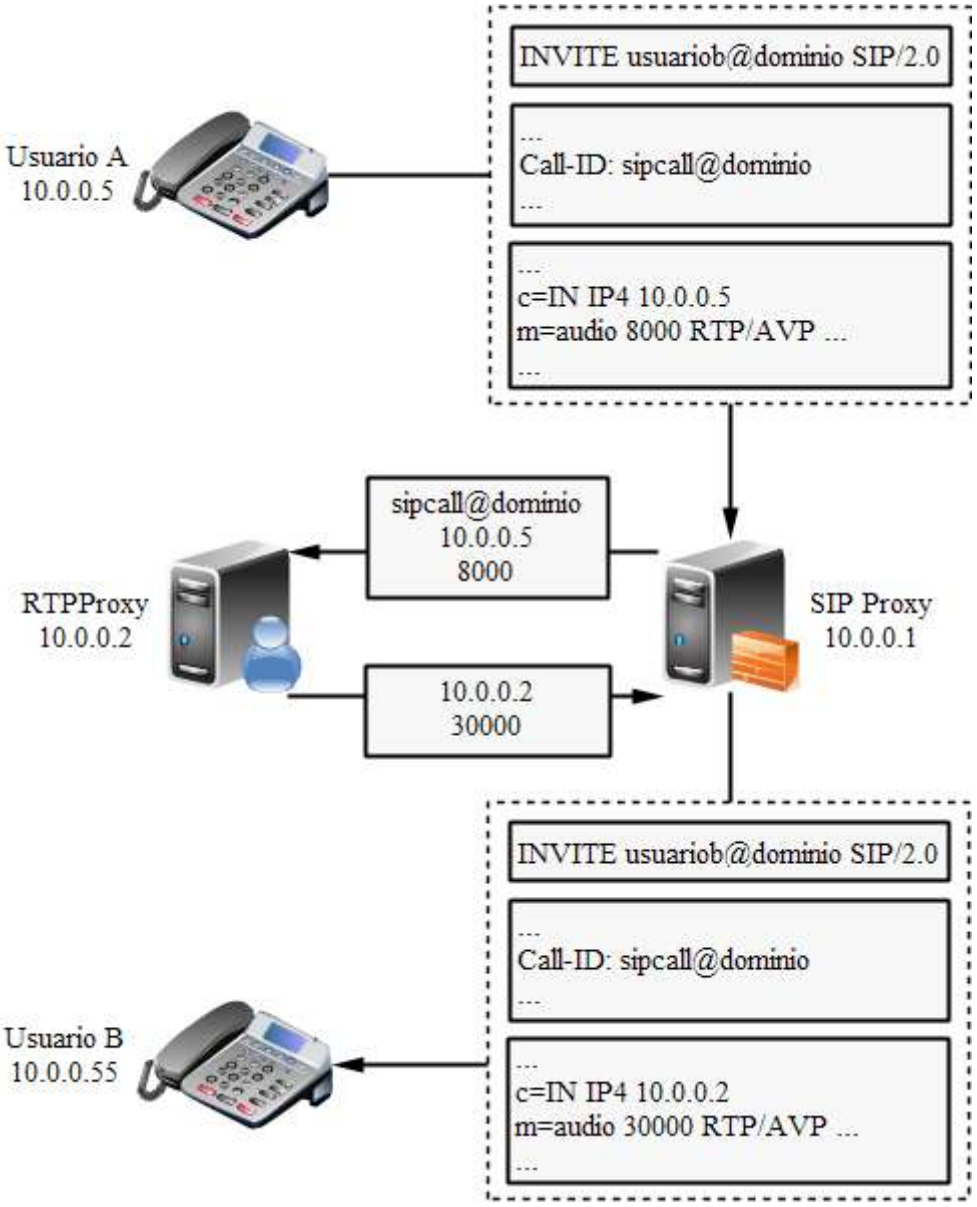

Fig. 4.4 – Interacción entre RTPProxy y un Proxy SIP ante una petición INVITE

4.6.1 Funcionamiento

Al trabajar en conjunto con un proxy SIP, RTPProxy puede encontrarse dentro del mismo componente hardware o no. Como se observa en la Fig. 4.4, cuando el Proxy SIP recibe una petición INVITE, extrae el valor del campo *Call-ID* del encabezado y el valor de los campos “m” y “c” del mensaje SDP, y se lo envía a RTPProxy mediante un socket Unix. RTPProxy comprueba si existe alguna sesión que posea ese id. En caso de que exista, retorna su dirección IP y el número del puerto UDP que está siendo utilizado para el usuario receptor de la petición; en caso contrario, se enlaza a un puerto UDP libre elegido al azar y retorna el número de ese puerto, además de su dirección IP. Tras recibir la respuesta por parte de RTPProxy, el Proxy SIP reemplaza en el mensaje SDP la dirección IP y el puerto para que apunte al RTPProxy, posteriormente, envía la petición a donde corresponda.

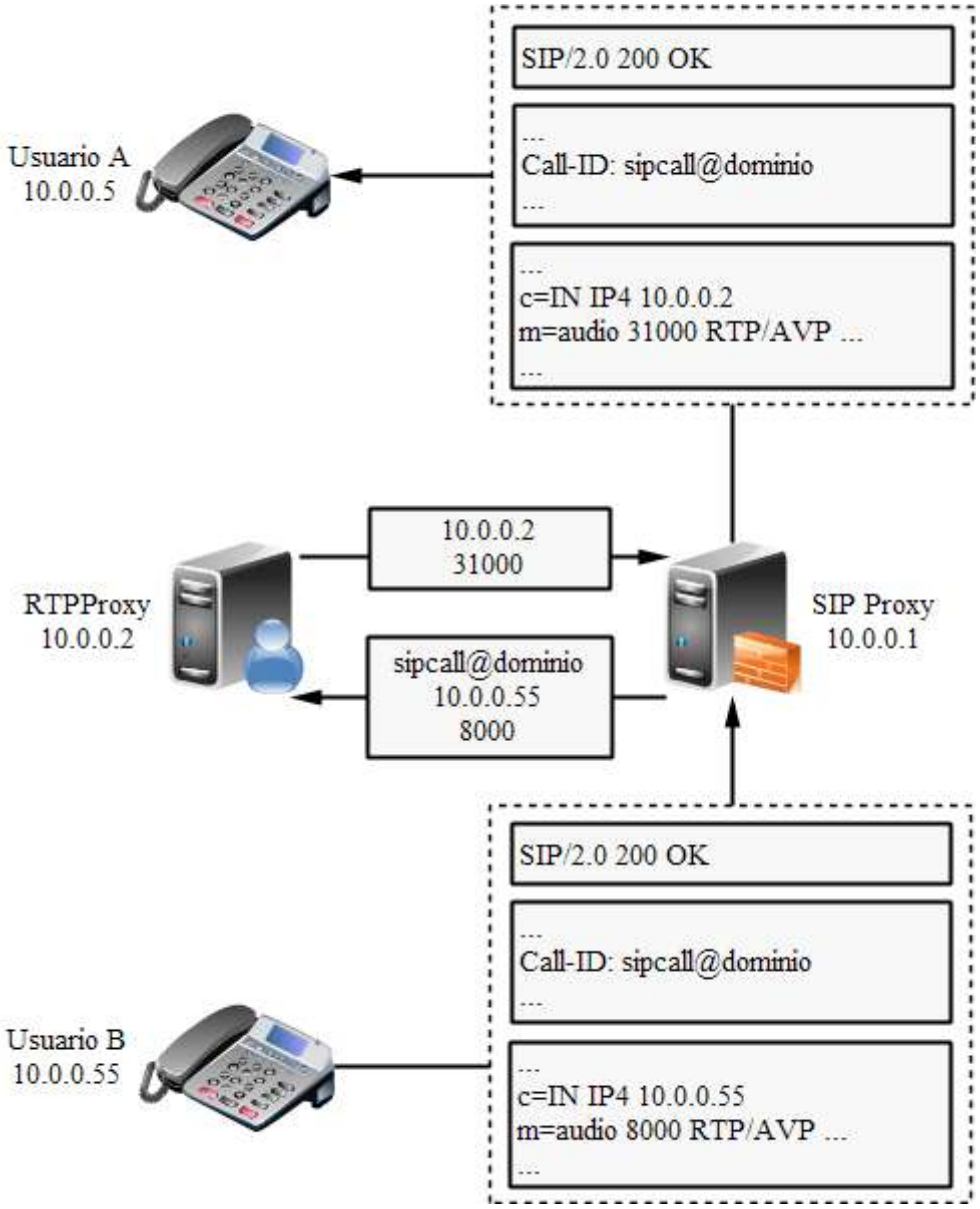

Fig. 4.5 – Interacción entre RTPProxy y un Proxy SIP ante una respuesta afirmativa

Cuando el Proxy SIP recibe una respuesta no negativa que contenga un mensaje SDP, como en la Fig. 4.5, nuevamente extraerá el valor del campo *Call-ID* y los campos “m” y “c”, y se los comunicará a RTPProxy.

En este caso, RTPProxy realiza una búsqueda entre todas las sesiones existentes. Si existe una sesión con ese id retorna, junto a su dirección IP, el número del puerto UDP que usará el receptor de la respuesta. En caso contrario, retorna un código de error indicando que no existe ninguna sesión con ese id.

Tras recibir la respuesta por parte de RTPProxy, el Proxy SIP reemplaza en el mensaje SDP la dirección IP y el puerto para direccionar el flujo RTP hacia RTPProxy, luego, envía la respuesta a donde corresponda. Una vez creada la sesión, RTPProxy escucha en el puerto asignado a esa sesión y espera recibir al menos un paquete UDP de cada uno de los participantes de la llamada. Una vez recibidos esos paquetes, el proxy almacena la dirección ip y el puerto de origen de los paquetes en una estructura asociada a cada flujo, y empieza a retransmitir los paquetes entre los participantes.

4.7 Pacemaker

Pacemaker (21) es un gestor de recursos de clúster de código abierto, que brinda las herramientas para lograr la máxima disponibilidad de los servicios de un clúster, a los que denomina recursos. Su objetivo principal es detectar y recuperar las fallas de un sistema, tanto a nivel de nodos como aplicaciones, haciendo uso de las capacidades de mensajería y membresía provistas por la infraestructura del clúster.

Las infraestructuras capaces de trabajar con esta tecnología son las provistas por los proyectos Heartbeat (22) y CoroSync (23). A los efectos de este proyecto, y con el fundamento de que Pacemaker impulsó el desarrollo de la segunda (en base a lo expresado en su documentación) (21), se optó por la utilización del motor de clúster CoroSync, definido como un sistema de comunicación de grupo con características adicionales para implementar alta disponibilidad entre aplicaciones.

4.7.1 Características y Funcionalidades

Pacemaker es una tecnología muy compleja, que provee diversas herramientas para la configuración y administración de un clúster y al mismo tiempo resuelve muchas de las problemáticas que este proceso implica. Gracias a esta tecnología, un clúster puede detectar y recuperarse de fallas a nivel de nodo y servicio, y además replicar automáticamente la configuración de estos últimos y modificarla desde cualquier miembro.

Para que un servicio pueda ser gestionado por el clúster debe poder ser administrado por medio de un script, característica que admite convertirlo en un recurso de Pacemaker. En el contexto de Pacemaker, se denomina recurso a cualquier servicio que se requiera que posea una alta disponibilidad.

Además, permite especificar el orden de los recursos en el sistema, es decir, en que miembro o miembros se ejecutarán los servicios y en qué orden deberían hacerlo, además de cómo deben este trasladarse por los distintos integrantes en base a distintas condiciones, como por ejemplo la falla o aislamiento de un nodo. Adicionalmente

provee herramientas de gestión, encriptación y unificación que facilitan este este tipo de labor.

Esta tecnología es capaz de ajustarse a casi cualquier tipo de clúster. Soporta grandes y pequeños grupos, gestiones por recursos y/o quórum y prácticamente cualquier configuración de redundancia (descritas en la sección 2.4.1). Al mismo tiempo, dota a estos sistemas de soporte para un almacenamiento agnóstico, de manera que no necesiten ningún requerimiento adicional para lograr un almacenamiento compartido; y “fencing”, que consiste en aislar un nodo del clúster en caso de mal funcionamiento, con el fin de asegurar la integridad de los recursos. El mecanismo de “fencing” provisto por Pacemaker se denomina STONITH (*Shoot The Other Node In The Head)*.

4.7.2 Arquitectura

En un alto nivel de abstracción, un clúster gestionado mediante Pacemaker está conformado por tres grandes bloques. En el primer nivel, están las componentes que no son conscientes del clúster, dentro de los que se incluyen los propios recursos; los scripts que los inician, paran y monitorean; y un demonio local, denominado agente de recurso, que enmascara las diferencias entre los distintos estándares que estos implementan.

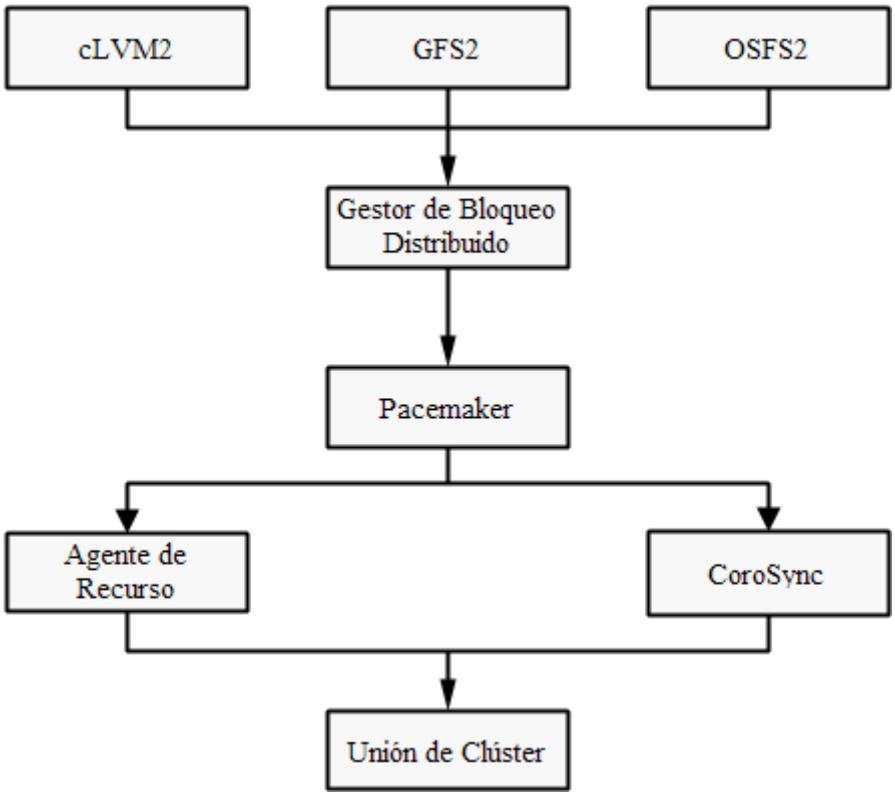

Fig. 4.6 – Dependencias de un clúster gestionado mediante Pacemaker en combinación con una infraestructura CoroSync

En el segundo nivel se encuentra el Gestor de Recursos, definido como el cerebro que procesa y reacciona a los eventos respecto al clúster. Estos incluyen la membresía de nodos; eventos de recursos causados por fallas, mantenimiento y actividades programadas; y otras acciones administrativas. Luego de la ocurrencia de cualquiera de

estos sucesos, Pacemaker computa el estado ideal para el clúster y traza un camino para alcanzarlo. Este proceso puede incluir movimientos de recurso, anulación de nodos o incluso forzar el apagado o reinicio de un servidor con interruptores de poder remotos.

En el tercer nivel se alberga la infraestructura, es decir, proyectos como CoroSync y Heartbeat que proveen información confiable de mensajería, membresía y quorum respecto al clúster.

Cuando se combina con CoroSync, Pacemaker es capaz de soportar sistemas de archivos populares de código abierto. Estos sistemas hacen uso de un gestor de bloqueos distribuido, que a su vez utiliza el motor de CoroSync, por sus capacidades de mensajería y membresía, y Pacemaker por sus servicios de “fencing”. En la Fig. 4.6 se ilustran las dependencias planteadas.

Capítulo 5
"Diseño e Implementación de la Central Telefónica"

5.0 Introducción

El objetivo de este capítulo es exponer todo lo hasta aquí expresado de una manera práctica en lo que se refiere al rediseño y la puesta en marcha de una central de telefonía VoIP, cumpliendo con los objetivos planteados en la sección 1.3.

En la sección 5.1 se presentará el rediseño de la central telefónica, los módulos que la conforman y como fueron distribuidas los diferentes componentes de software presentados en el Capítulo 4. La sección 5.2 describirá el objetivo y el funcionamiento de cada uno de los módulos diseñados, dedicando una subsección al componente de software Pacemaker. Luego, en la sección 5.3 se presentará la central telefónica implementada para la realización de pruebas, utilizando los recursos disponibles en el Laboratorio de Redes de la Facultad de Ingeniería. A continuación, la sección 5.4 retoma los intercambios de mensajes presentados en la sección 3.3, pero ahora desde el punto de vista de la central telefónica implementada. Finalmente, las secciones 5.5 y 5.6 presentaran las pruebas realizadas y los resultados obtenidos acerca de la disponibilidad y el rendimiento de la central telefónica, respectivamente.

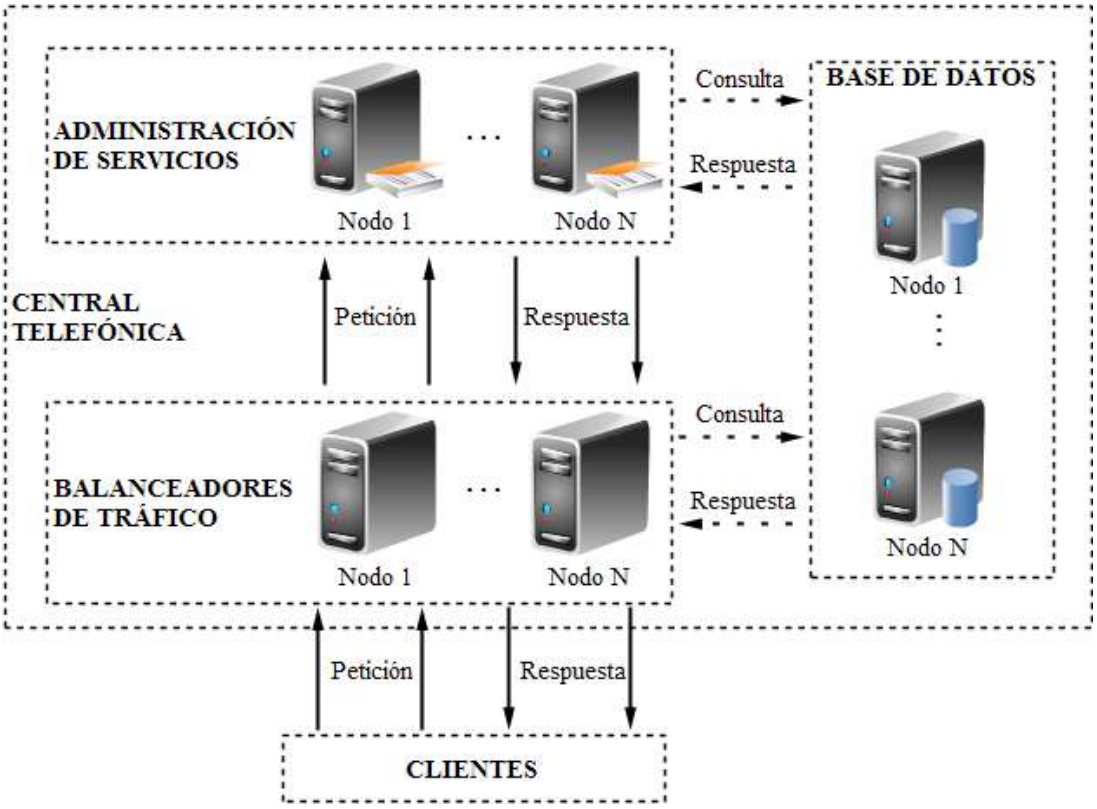

Fig. 5.1 – Rediseño de la central telefónica

5.1 Diseño de la Central Telefónica

Tras el análisis de los diseños de (2) y (3) desarrollado en la sección 1.2, se arribó a la conclusión que se debía realizar un rediseño de central telefónica para superar los problemas encontrados. Con los conceptos teóricos expuestos en el Capítulo 2 y en el Capítulo 3 y la actualización de las tecnologías de software a utilizar, descritas en el Capítulo 4; el rediseño de la central telefónica VoIP constará de tres módulos (Fig. 5.1), un módulo de base de datos en donde se almacenará información importante para el funcionamiento de la central, un módulo de administración de servicios que se encargará de atender las solicitudes de los clientes, y un módulo balanceador de tráfico que distribuirá la carga entre los nodos del módulo mencionado anteriormente.

Las tecnologías de software descritas en el Capítulo 4 fueron distribuidas en cada uno de los módulos como se explica a continuación. El módulo de base de datos estará conformado por MariaDB, Galera Cluster y GLB. El módulo de administración de servicios estará compuesto por Asterisk, y el módulo de balanceo de tráfico será conformado por Kamailio y RTPProxy. El componente de software Pacemaker participará en cada uno de los módulos para dotarlos de alta disponibilidad.

La principal diferencia respecto del diseño de (3) es la separación física del servicio de localización de las bases de datos, uniéndolo al servicio de registración para conformar el módulo administrador de servicios. El rediseño presentado resulta flexible respecto de varios aspectos:

- Dependiendo de los requerimientos respecto de la cantidad de llamadas simultáneas a soportar; la central telefónica podría ser desplegada en varios servidores, o ser virtualizada en uno o más de los componentes físicos mencionados.
- Se considera la comunicación entre los módulos utilizando diferentes redes, para el tránsito de mensajes de los protocolos principales de cada módulo. A modo de ejemplo, suponiendo que el mayor consumo de ancho de banda provenga de las peticiones de los clientes, se podría desplegar una red para atender las peticiones de los clientes (SIP y RTP) y otra red para los mensajes de Pacemaker, realizar consultas a base de datos y demás servicios tales como SSH (*Secure SHell*).

5.2 Funcionamiento

En la presente sección se explica el objetivo de cada módulo, la funcionalidad ejercida por cada uno de los componentes de software que los componen, como trabajan conjuntamente para alcanzar el objetivo del módulo, y las situaciones en las cuales se comunican con los demás módulos. Además, se explicará cómo interviene la tecnología Pacemaker para proveer de alta disponibilidad a cada uno de los módulos.

5.2.1 Módulo de Base de Datos

El módulo de base de datos se encargará de proveer el acceso y almacenamiento de algunas opciones de configuración de los demás módulos de la central, además de almacenar información de los usuarios del servicio. Como se mencionó en la sección 4.0, MariaDB será el encargado de almacenar la información mencionada.

Los servidores de los módulos de balanceo y de administración de servicios no se contactarán directamente con ninguna base de datos, en su lugar, se comunicarán con GLB, que fue configurado para aplicar una política de balanceo dependiendo del estado del servidor MariaDB que se encuentra en el mismo nodo que GLB. La política de balanceo elegida, descrita en la sección 4.3.2; es “*least connected*”, a la cual se modificó su comportamiento mediante la opción “top”. Cada GLB fue configurado de forma tal que asigne una prioridad de 2 a la base de datos que se encuentre en el mismo nodo, y una prioridad de 1 al resto.

Si el servidor de base de datos perteneciente al mismo nodo que GLB se encuentra funcionando correctamente, todas las conexiones serán dirigidas a él, ya que es el único servidor de prioridad 2 que se encuentra activo. En caso contrario, las conexiones se repartirán equitativamente entre las demás bases de datos.

Respecto a cuáles GLB serán contactados por los nodos de los demás módulos, el diseño de central telefónica propuesta contempla una distribución equitativa respecto de la cantidad de nodos que posean los módulos. Es decir, si se tuviese una central con cuatro nodos de base de datos, y cuatro nodos balanceadores de tráfico, se configuraría una relación unívoca entre los nodos de cada módulo. Si en la misma central se tuviesen cinco nodos en el módulo de administración de servicios, se podrían asignar tres de los nodos a tres instancias de GLB distintas, y los nodos restantes podrían ser asignados a la instancia de GLB presente en el servidor más potente, en caso de que los nodos del módulo de base de datos no fueran homogéneos.

Dado que la información almacenada en este módulo es crítica para el funcionamiento de la central, y a diferencia de (3), se utilizó Galera Cluster para proveer una topología multi-maestro con redundancia activo/activo, permitiendo la lectura y escritura en cada nodo, además de facilitar la escalabilidad del módulo mediante el agregado de nodos con un esfuerzo mínimo en su configuración.

5.2.2 Módulo de Administración de Servicios

Los nodos pertenecientes a este módulo son los encargados de atender todas las solicitudes de los clientes, las cuales fueron re-direccionadas por los servidores del módulo de balanceo de tráfico.

Cuando los servidores reciben una petición de registración de un usuario, la aceptan ya que el usuario fue autenticado por el módulo de balanceo, y almacenan la información de contacto en base de datos para que esta pueda ser utilizada por los otros nodos del módulo. A su vez, almacena la información en memoria RAM el tiempo especificado por el campo “expire” del encabezado SIP, para evitar tener que consultar la base de datos cada vez que se solicite establecer una comunicación con ese usuario. La información almacenada por Asterisk es similar a la información almacenada por Kamailio, con la diferencia de que Asterisk también almacena la dirección IP del nodo balanceador que le ha enviado la petición de registro del usuario, la cual será utilizada para poder comunicarse con los usuarios. Además de almacenar información relevante para el uso de los diferentes servicios ofrecidos por Asterisk.

En el caso en el que algún nodo del módulo reciba una petición de un usuario para comunicarse con otro, primero buscará de forma local en su memoria RAM, y si

encuentra la información de contacto del usuario, iniciará una transacción para poder llevar a cabo la comunicación. En caso de que no posea la información del usuario en memoria, Asterisk realizará una consulta a la base de datos y luego iniciará una transacción para contactarse con ese usuario, además de guardar en memoria un breve periodo de tiempo la información de contacto del usuario, nuevamente para evitar realizar consultas a la base de datos.

5.2.3 Módulo de Balanceo de Tráfico

La función del módulo de balanceo de tráfico es la de distribuir de forma equitativa las conexiones que se reciben por parte de los clientes hacia el módulo de administración de servicios. Los servidores que pertenecen a este módulo serán entrada y salida de la central telefónica, por lo que, además de distribuir las conexiones entrantes, también se encargarán de autenticar a los usuarios en todas las acciones que emprendan.

Cuando un usuario intenta registrarse o realizar una llamada a otro usuario o servicio de la central, antes de elegir que nodo del módulo de administración de servicios será el encargado de atenderla, Kamailio se encargará de verificar la identidad del usuario. El proceso de autenticación para cada caso (registros y llamadas a un servicio o cliente) es idéntico al reflejado en los ejemplos de la sección 3.3, lo único en lo que difiere es que Kamailio debe realizar una consulta a base de datos luego de recibir la información de autenticación del usuario para poder autenticarlo, ya que no posee dicha información en memoria. Para el caso de una petición de registro, Kamailio almacenará en memoria, y posteriormente en base de datos, la información de contacto del usuario, ya que es el componente encargado de encaminar los mensajes entre los nodos Asterisk y los usuarios de la central telefónica.

Para el tratamiento del tráfico SIP, Kamailio fue configurado para hacer uso de dos políticas de balanceo, de acuerdo a la acción realizada por el cliente. Cuando un usuario inicia el proceso de registración, se aplica una política de balanceo que consiste en aplicar una función hash sobre el “From-URI” de la petición SIP, permitiendo así que el usuario se registre siempre en el mismo servidor administrador de servicios, en tanto este se encuentre funcionando. La elección de esta política está sujeta a evitar una inconsistencia que puede ocurrir en los nodos del módulo de administración de servicios, en la cual los servidores Asterisk podrían ver un estado distinto respecto del mismo usuario. Esta inconsistencia será explicada en la sección 5.4.1. Para el caso en que un usuario solicite comunicarse con un servicio de la central u otro usuario, se aplica la política de balanceo conocida como “Round-Robin”, la cual irá asignando cada llamada entrante de forma equitativa a cada servidor Asterisk.

A diferencia de (3), no es necesario aplicar ninguna política de marcado de paquetes para asegurarse que los mensajes pertenecientes a un mismo diálogo SIP sean encaminados al mismo servidor Asterisk. Esto se debe a que el tráfico es balanceado a nivel del protocolo SIP, el cual incluye un mecanismo que permite identificar todos los mensajes que pertenecen a un mismo diálogo. Los mensajes son agrupados mediante un campo del encabezado del mensaje SIP denominado *Call-ID*, junto al parámetro “tag” de los campos *From* y *To*. De esta manera, las políticas de balanceo mencionadas son aplicadas únicamente cuando se recibe el primer mensaje de un diálogo SIP. Los mensajes subsequentes serán reconocidos mediante la combinación mencionada y serán encaminados según corresponda.

El componente de software restante que conforma este módulo, RTPProxy, se encargará de encaminar el flujo RTP entre los usuarios y los servidores Asterisk pertenecientes al módulo de administración de servicios que fueron elegidos para encargarse de su consulta. Su funcionamiento será idéntico al explicado en la sección 4.6.1.

5.2.4 Alta Disponibilidad con Pacemaker

Los módulos mencionados en las secciones anteriores conforman un sistema perfectamente funcional con la capacidad de proveer los servicios de una central telefónica, sin embargo, no cubren totalmente los aspectos de disponibilidad y escalabilidad perseguidos en este proyecto. Ante la presencia de alguna falla que implique la caída de un nodo o un servicio, el sistema no dejará de funcionar pero presentará algunos inconvenientes, como la incapacidad de recuperar automáticamente las partes involucradas en el incidente. Además, cada vez que el sistema requiera expandirse mediante el agregado de un nuevo nodo, este y sus pares en el módulo al que pertenezca deben ser configurados manualmente para operar de manera conjunta.

Para resolver estas problemáticas se utiliza Pacemaker, quien se encarga principalmente de monitorear cada módulo del sistema para detectar fallas a nivel de software y a nivel de nodos, e intentar recuperar al conjunto para que se mantenga en su estado ideal. Así mismo, se hace uso de una de las características de esta tecnología que permite anexar fácilmente nuevos nodos al sistema gracias a un mecanismo de replicación de configuración. La capacidad de almacenamiento compartido explicado en la sección 4.7 no se implementa en este proyecto debido a que el funcionamiento de las tecnologías y estructuras involucradas favorecen el uso de una base de datos distribuida.

Pacemaker monitorea los distintos servicios que corren dentro del sistema gestionándolos como recursos. Para lograr esto, cada proceso que se desee sea gestionado por esta tecnología debe tener asociado lo que se denomina un “agente de recurso”, el cual es un script que mínimamente permite iniciar, detener y verificar el estado del servicio. Estos agentes pueden ser de dos tipos; los más simples son los basados en estándares Linux (LSB) que generalmente son provistos por las distintas distribuciones de código abierto y suelen ser de fácil entendimiento y modificación. A su vez, existen los scripts denominados OCF, que son vistos como scripts que extienden la capacidad de los agentes LSB y permiten administrar procesos de mayor complejidad, como aquellos que cuentan con estados especiales como Galera Cluster; o que aceptan parámetros de inicialización o finalización.

Una vez que se cuenta con todos los agentes necesarios para gestionar los recursos que conforman al clúster, se deben configurar los mismos para que cada servicio se comporte de la manera deseada. Las variables fundamentales que se deben especificar en casi cualquier estructura de clúster son costos o pesos, orden, dependencias y cantidad de instancias de cada proceso que se quieran ejecutar simultáneamente. Un costo es una relación entre un proceso y un nodo determinados, de manera que Pacemaker pueda determinar cuál es el esfuerzo que implica mantener activo un servicio en cada nodo y consecuentemente encontrar el esquema de relaciones que involucre el menor costo. Con el orden se define la disposición de un servicio a iniciarse luego de que otro u otros se encuentren activos; mientras que las dependencias describen la voluntad de un proceso de mantenerse activo o no, de acuerdo al estado en

el que se encuentren los demás procesos. Finalmente, el número de instancias por proceso permite detallar la cantidad de instancias de un servicio que se ejecutaran simultáneamente en un clúster. Su valor no puede superar al número total de nodos, y resulta primordial en la configuración de un clúster con una configuración de redundancia activo/activo.

Las siguientes subsecciones detallarán la configuración y el comportamiento de la herramienta Pacemaker en los módulos explicados en secciones anteriores.

5.2.4.1 Pacemaker en el Módulo de Base de Datos

En este módulo se posee una excepción, y es que se utilizan dos agentes de recursos para gestionar los tres procesos. Esto se debe a que Galera Clúster puede verse como una extensión de las funcionalidades de MariaDB, lo que para Pacemaker representa un único espacio de control.

Bajo esta convicción, se implementó un script OCF con el objeto de administrar el sistema de base de datos distribuido. En este caso, no se puede utilizar un elemento LSB debido a que Galera compone un conjunto de instancias de MariaDB. Se deduce que para iniciar un nodo es necesario conocer el estado del resto de los nodos, y posteriormente ejecutar el comando de inicio necesario dependiendo de la existencia del componente primario. Si bien se requiere de un agente de recurso con características específicas, su configuración es relativamente sencilla debido a que Galera implementa en sí mismo un clúster, resolviendo muchas de las problemáticas en cuanto a la comunicación y control de procesos, es decir, de instancias MariaDB. La labor de Pacemaker se resume en complementar estos mecanismos con la habilidad de iniciar, detener y monitorizar cada una de estas instancias de MaríaDB. Al crease el recurso sólo se indica al sistema cuántos y cuáles nodos conforman la base de datos distribuida, dejando que el script OCF opere el resto de los parámetros.

En la Fig. 5.2 se puede observar el trabajo en conjunto que realizan Pacemaker y Galera Clúster ante la ocurrencia de una falla en una de las instancias de MariaDB. En un instante T1, Galera detecta la caída de la instancia y la aísla del sistema de base de datos (instante T2). Seguidamente, en un instante T3, Pacemaker recupera la instancia caída, por ejemplo, reiniciando el proceso, dejándola en una situación donde Galera pueda reinsertarla en su labor original (instante T4).

GLB no requiere de controles especiales, por lo que es gestionado por el script LSB provisto por la distribución de Linux utilizada. La configuración del mismo se divide en dos etapas. Inicialmente, se debe crear el agente indicándole al sistema cuanto tiempo requiere GLB para ser iniciado, detenido y monitorizado, cada cuanto tiempo debe ser inspeccionado, y la acción a realizar cuando se detecta una falla. Particularmente, en este proyecto se dispone de 60 segundos para las operaciones de inicio y detención, con un intervalo de monitorización de 10 segundos, con otros 10 segundos de tolerancia para recibir la respuesta del estado del proceso. Estos tiempos son el resultado de pruebas realizadas sobre el caso práctico (sección 5.3) para determinar los tiempos adecuados de respuesta para cada situación, de acuerdo a las necesidades de la implementación práctica. En caso de que el monitoreo devuelva algún tipo de error, o bien se incumpla con el lapso de tolerancia del mismo, Pacemaker asumirá que sucedió una falla y realizará las acciones pertinentes para reiniciar el servicio. Finalmente,

debido a que el clúster que presenta este trabajo es activo/activo, se indica a Pacemaker que una instancia de GLB debe estar presente en cada nodo que conforma el módulo de base de datos.

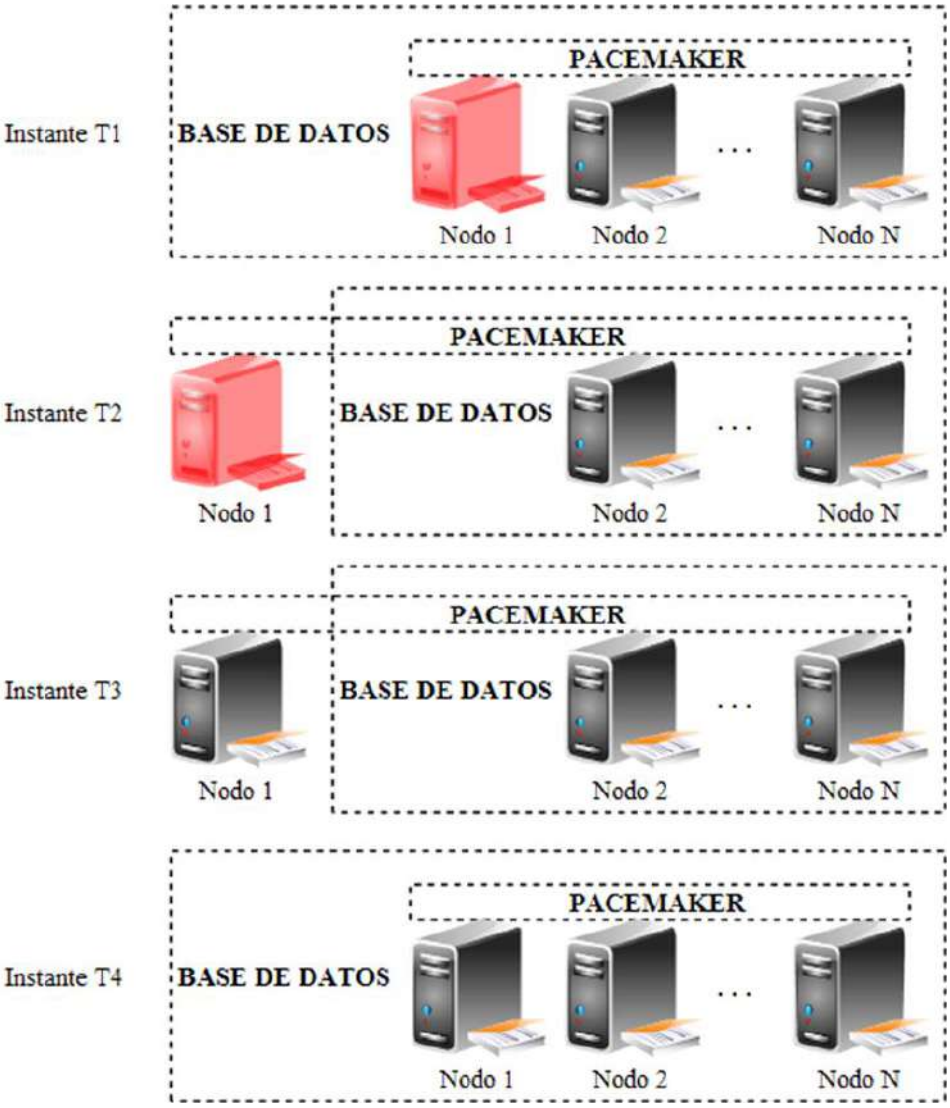

Fig. 5.2 – Funcionamiento de Pacemaker ante el fallo de una instancia de MariaDB en Galera Clúster

Si un nodo de la base de datos distribuida se encuentra inaccesible, GLB se encarga de redirigir las peticiones hacia un punto donde puedan ser resueltas, sin embargo, sí lo mismo sucede con una instancia de GLB, los clientes vinculados con el perderán el acceso a la base de datos. Para resolver esta problemática se hace uso de un agente OCF provisto por Pacemaker, el cual permite manipular IPs virtuales como recursos. En este caso, se crean tantas IPs como instancias de GLB halla en el módulo de base de datos, donde a cada una se le asigna un peso sobre un nodo y otro menor en el resto, de manera que al fallar una instancia del balanceador, la IP relacionada se traslade a cualquier otro miembro con la capacidad de resolver las consultas dirigidas al nodo donde se encuentra la instancia inaccesible. Para que esto tenga sentido, se indica a Pacemaker que una IP sólo se puede ejecutar en un nodo, sólo si este contiene una instancia de GLB activa.

En la Fig. 5.3 se muestra el comportamiento de Pacemaker ante la caída de una instancia de GLB. Cuando se detecta un error (instante T1), lo primero que ocurre es la detención del recurso que corresponde a la IP virtual en el nodo involucrado, de manera que la misma pueda ser asignada en otro nodo sin entrar en ningún tipo de conflicto. Una vez la IP se encuentre accesible nuevamente (instante T2), Pacemaker se encarga de recuperar la instancia caída, para que finalmente todos los recursos se reposicionen en su lugar original (instante T3).

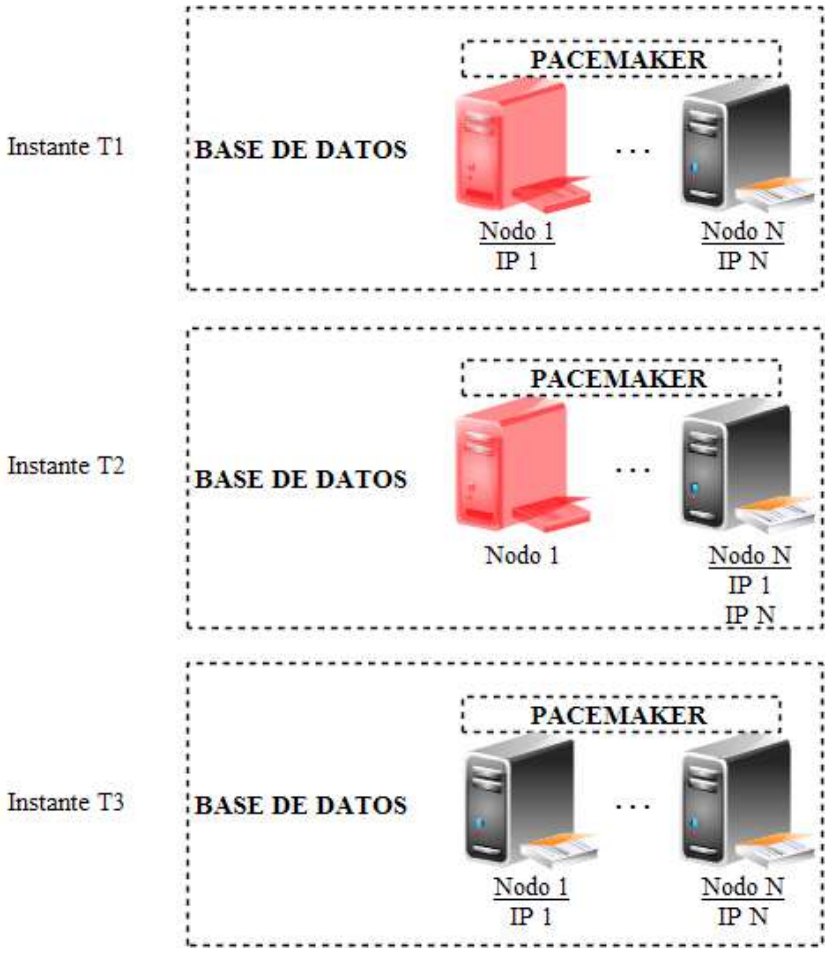

Fig. 5.3 – Funcionamiento de Pacemaker ante el fallo de una instancia de GLB.

5.2.4.2 Pacemaker en el Módulo de Administración de Servicios

La labor de Pacemaker se resume a la gestión del proceso Asterisk con el objeto de que el mismo se encuentre activo, siempre y cuando sea posible, en todos los nodos que conforman al conjunto.

El agente de recurso utilizado para esta tarea es el script LSB suministrado por la distribución Linux. La configuración del mismo es muy similar a la del recurso de GLB especificada en la sección anterior. La programación de los tiempos de inicio, parada y monitoreo son exactamente los mismos, así como la acción de reiniciar un proceso ante la detección de una falla. Además, se configuro al recurso para que se encuentre presente en todos los nodos del módulo.

Cada instancia de Asterisk requiere de una IP para poder ser alcanzada por los distintos servicios con los que interactúa. Por este motivo, se utiliza el mismo agente tipo OCF utilizado para el manejo de las IPs de GLB para crear y gestionar las direcciones IPs utilizadas por Asterisk. En este caso, se creó una IP virtual para cada proceso, donde a cada una se le asignó un peso muy grande sobre un único nodo y otro infinitamente pequeño en el resto, de manera que sólo se pueda establecer en un único lugar. Posteriormente, se le indica al sistema que cada IP debe ser ejecutada en vinculación con una instancia de Asterisk. Se debe destacar que no es necesario que las IPs virtuales se muevan a través de los distintos miembros del conjunto, ya que el módulo de Balanceo se encarga de monitorear el estado de los distintos nodos de Asterisk, equilibrando la carga sólo sobre aquellos que se encuentren accesibles.

En la Fig. 5.4 se puede visualizar la labor de Pacemaker ante la caída de una instancia de Asterisk. En el momento en que se detecta una falla (instante T1), se intenta reiniciar el proceso comprometido, esto implica detenerlo para luego volverlo a iniciar (instante T2). Si alguna de estas dos acciones no puede concretarse Pacemaker debe recurrir a medidas más rigurosas, como puede ser el uso de mecanismos STONITH. Estos últimos requieren de hardware adicional para su óptimo funcionamiento por lo que escapan del análisis de este proyecto.

Fig. 5.4 – Funcionamiento de Pacemaker ante el fallo de una instancia de Asterisk

5.2.4.3 Pacemaker en el Módulo de Balanceo de Tráfico

RTPProxy requiere de un agente de recurso con capacidades específicas. Tomando en cuenta que esta tecnología no es multihilo, y con el objeto de mejorar la performance de la misma se creó un script OCF que permite ejecutar varias instancias que son vistas como un único proceso por Pacemaker. Al momento de crear el agente, se indica al sistema la cantidad de instancias que se ejecutarán, y el tiempo de tolerancia que se le dará al recurso para iniciar todos ellos. También admite como parámetro definir un “puente” que realiza el flujo RTP entre interfaces, en caso de que el mismo deba transitar por IPs que pertenezcan a distintas redes. Consecuentemente, se definió tantos recursos como nodos hay en la topología, indicando a cada uno su correspondiente “puente”. La cantidad de instancias por recurso creado puede ser variable dependiendo de los requerimientos de llamadas simultáneas a soportar por la central. En base a lo

mencionado, se deduce que cada recurso sólo puede ser ejecutado en un único nodo, motivo por el cual se deben indicar pesos infinitamente pequeños para aquellos nodos donde no se deba ejecutar la tecnología.

En la Fig. 5.5 se ilustra el comportamiento de Pacemaker ante la falla de uno de las instancias de ejecución de RTPProxy.

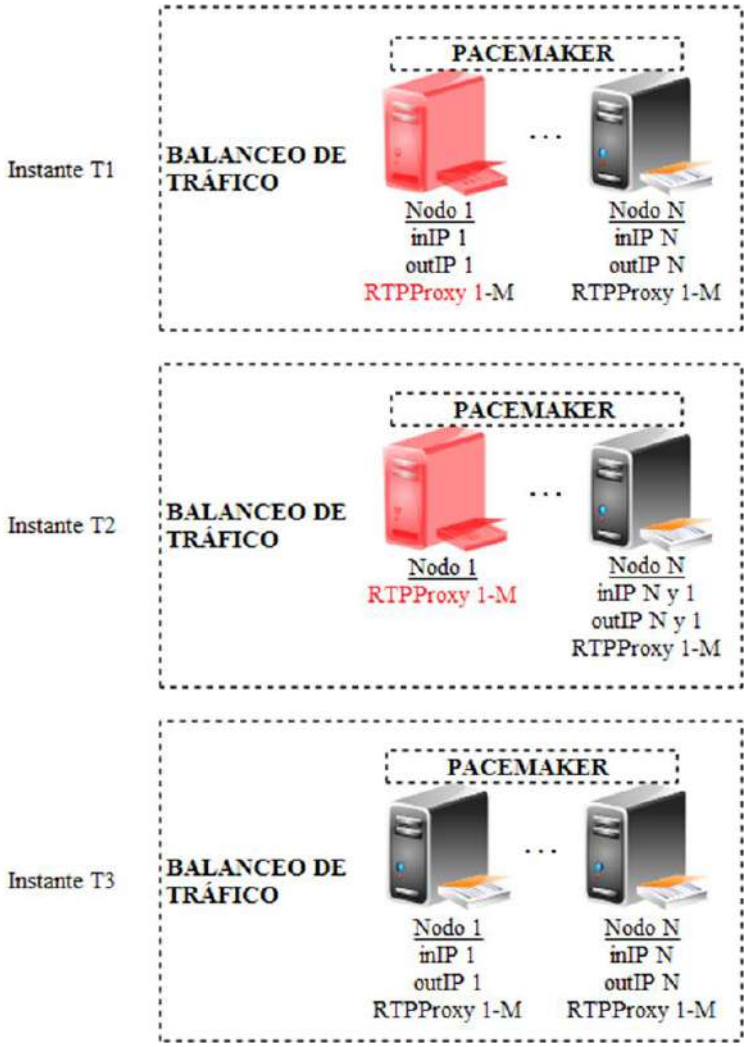

Fig. 5.5 – Funcionamiento de Pacemaker ante el fallo de una instancia de RTPProxy

Un error en este nivel implica la inaccesibilidad al servicio de una porción de los clientes que se comunican por medio del nodo afectado (instante T1). La gravedad de la situación obliga a Pacemaker a tomar la decisión (instante T2) de detener el proceso de todas las instancias que se ejecutan en el nodo, y migrar las IPs de acceso hacia otro nodo con la capacidad de atender las peticiones. Cuando la falla es solucionada y se vuelve a ejecutar todas las instancias de RTPProxy (instante T3), las direcciones IP son devueltas a su posición original.

Dentro del mismo módulo, Kamailio es gestionado con el script LSB provisto por la distribución Linux utilizada. La configuración del mismo es muy similar a la del recurso Asterisk explicada en la sección anterior. Inicialmente, se indicó a Pacemaker que se debe dispone de 120 segundos para las operaciones de inicio y detención, con un

intervalo de monitoreo de 10 segundos, con otros 10 segundos de tolerancia para recibir la respuesta del estado del proceso. De manera similar a lo sucedido con GLB, estos tiempos son el resultado de pruebas realizadas de acuerdo a las necesidades de la implementación práctica (sección 5.3). Además, se configuro al agente para que esté presente en todos los nodos del módulo.

La inaccesibilidad a cualquiera de los servicios que componen el módulo de balaceo de carga, deja al nodo víctima de la falla inutilizable, generando que los clientes vinculados con el mismo no dispongan del servicio telefónico. Para solucionar esta problemática se hace uso de las IPs virtuales, gestionadas por el agente provisto por Pacemaker. Se definen dos IPs por cada nodo, una para recibir las peticiones de los clientes y otra para comunicarse con el módulo de administración de servicios. Se indica al sistema la relación de dependencia que existe entre estas direcciones, y los recursos de RTPProxy y Kamailio, de manera que al detectar una falla se detengan todos los recursos del acceso que se encuentre no disponible, y se trasladen las IPs del mismo a otro miembro que pueda resolver las consultas.

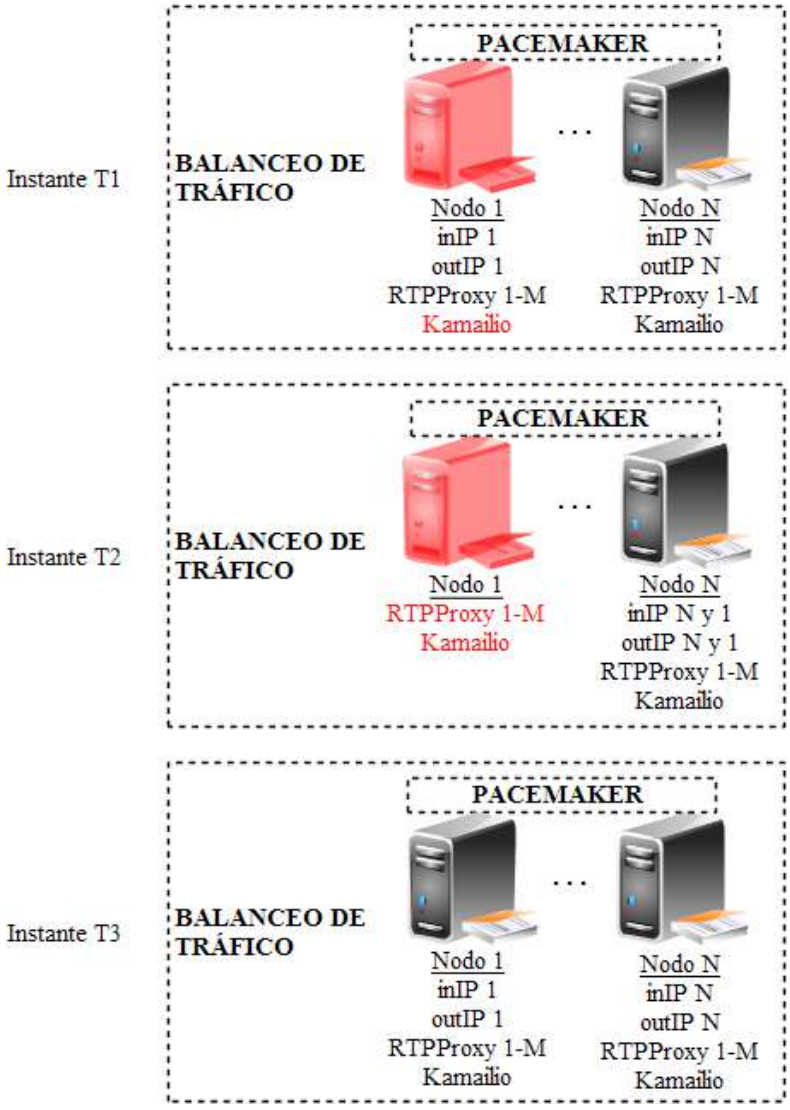

Fig. 5.6 – Funcionamiento de Pacemaker ante el fallo de una instancia de Kamailio

La Fig. 5.6 ilustra la caída de la tecnología Kamailio y las acciones de Pacemaker para volver al estado ideal del módulo. Cuando Kamailio deja de responder con normalidad (instante T1), inmediatamente se procede a detener el recurso de RTPProxy y las IPs virtuales en el nodo comprometido, de manera que las mismas puedan transitar hacia otro nodo sin generar conflictos (instante T2). Al momento en que se recupera la instancia caída, se restablecen todos los recursos restableciendo el orden óptimo (instante T3).

5.3 Implementación de un caso práctico

Debido a que el proyecto fue desarrollado en el ámbito del Laboratorio de Redes de la Facultad de Ingeniería de la UNLPam, fue necesario adaptar el diseño realizado a los componentes de hardware disponibles. Esta restricción permitirá demostrar la flexibilidad del diseño mencionada a comienzos del presente capítulo.

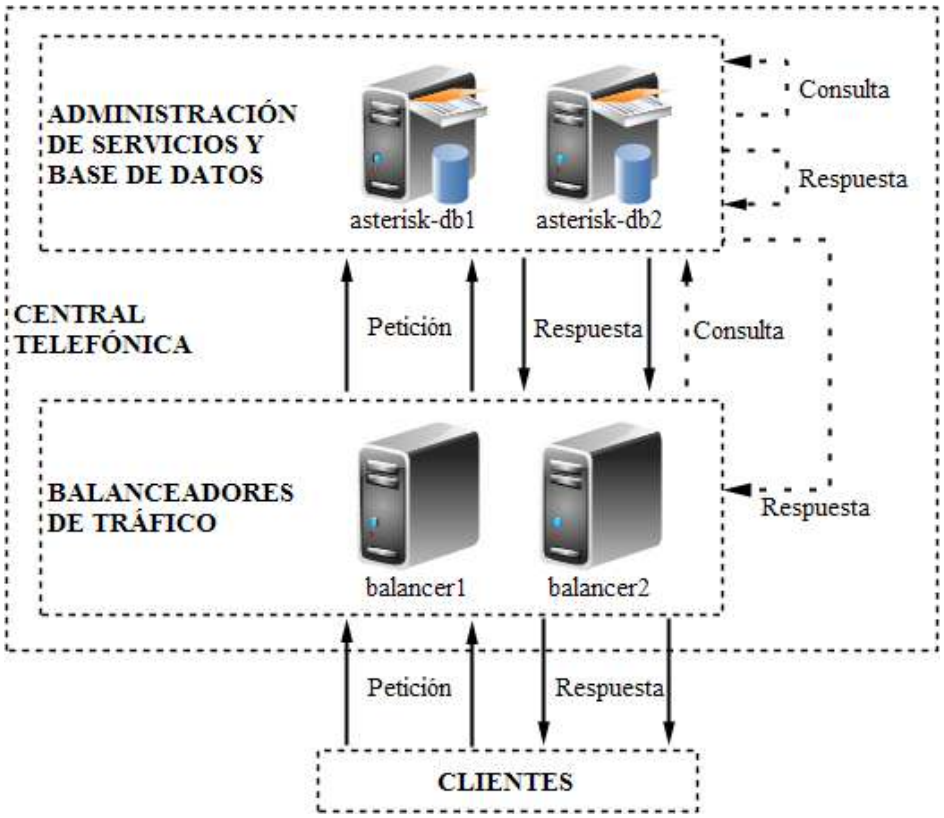

Fig. 5.7 – Topología de la central telefónica implementada

Al solo disponer de cuatro computadoras para la implementación de la central, se decidió la instalación de los módulos de base de datos y administración de servicios en un mismo componente físico. El diseño definido para la implementación del caso práctico se ve reflejado en la Fig. 5.7. Se destaca el hecho de que al realizar una implementación en un ambiente controlado es posible utilizar Galera Cluster únicamente con dos nodos de base de datos. En un sistema de producción, lo recomendable es desplegar un número impar de nodos de base de datos (24), siendo tres el número mínimo, a fin de evitar que un fallo en la red produzca la separación de los nodos en partes iguales, derivando en el bloqueo de las operaciones escritura en todos

los nodos hasta que alguna de las partes obtenga la mayoría de votos, tal como se mencionó en la sección 4.2.2.

A continuación, la Tabla 5.1 presenta las características técnicas de cada nodo, información relevante para la sección 5.6.

Nombre	Procesador (8 núcleos)	Memoria	Disco Rígido	Ifaces. de Red
asterisk-db1	AMD FX-8120	8 GB	500 GB	x1 GbEth
asterisk-db2	AMD FX-8120	8 GB	500 GB	x1 GbEth
balancer1	AMD FX-8120	8 GB	500 GB	x2 GbEth
balancer2	AMD FX-8120	8 GB	500 GB	x2 GbEth

Tabla 5.1 – Prestaciones de los componentes de hardware de los nodos servidores

En cuanto a la infraestructura de conexión, se contó con un switch Linksys SRW224G4 (25) que permitió aislar todos los componentes de hardware utilizados del resto de equipamiento del Laboratorio de Redes. Debido a que el switch únicamente posee cuatro puertos capaces de soportar Gigabit Ethernet, dos de las interfaces de red tuvieron que utilizar los puertos Fast Ethernet, lo que terminó limitando la capacidad de sesiones simultáneas soportada. Además, se realizó una división lógica a nivel de capa de red en el modelo OSI, es decir, a nivel del protocolo IP; para simular una división en diversas redes y a su vez facilitar la comprensión de las capturas de paquetes realizadas. De esta manera, los nodos de administración y base de datos utilizaran una dirección IP de la red interna de la central, inalcanzable directamente por los clientes; mientras que los nodos balanceadores fueron configurados con una dirección IP interna y otra externa, una por interface de red. Los clientes utilizaran la dirección IP externa para acceder a los servicios de la central.

También se contó con tres hardphone Grandstream GXP280, utilizados en (2), y el softphone Zoiper, el cual fue instalado en dos PCs. Estos teléfonos IP fueron utilizados para realizar pruebas básicas que permitiesen corroborar la correcta configuración de cada una de las tecnologías utilizadas.

5.4 Intercambio de mensajes

En la sección 3.3 se presentó un ejemplo de intercambio de mensajes para el proceso de registración de un usuario, y otro ejemplo para la concreción de una llamada. Ambos ejemplos serán retomados en esta sección, pero ahora desde el punto de vista de la topología que se ha diseñado e implementado.

5.4.1 Registración de un usuario

La Fig. 5.8 ilustra el intercambio de mensajes involucrados en el proceso de registración del usuario. En este caso en particular, el usuario A se pone en contacto con el nodo de balanceo “balancer1”.

La primera transacción (mensajes 1 y 2) corresponde a un intento fallido de registración por parte del usuario A, debido a que no ha proporcionado la información necesaria para

autenticarle. Al iniciar la segunda transacción, el usuario envía la información de autenticación en un nuevo mensaje REGISTER.

Al no poseer la información de los usuarios en memoria, Kamailio, luego de recibir la petición de registro, deberá consultar una base de datos para poder autenticar al usuario. Debido a la configuración explicada en la sección 5.2.3, Kamailio realiza una consulta al GLB del nodo “asterisk-db1”. Luego, GLB derivará la consulta al servidor MariaDB existente en el mismo nodo. MariaDB procesará la consulta, y enviará la respuesta a GLB para que la redireccione a Kamailio.

Con la información necesaria, Kamailio corroborará la identidad del usuario A, y continuará con el proceso de registración. Para ello, Kamailio enviará el mensaje REGISTER recibido por parte del usuario hacia uno de los servidores Asterisk, de acuerdo al resultado de la función hash aplicada al “From-URI” de la petición de registro. Una vez que Asterisk almacene la información de contacto del usuario A en memoria y en la base de datos, responderá con un mensaje OK a Kamailio, el cual reenviará al usuario para terminar el proceso de registración.

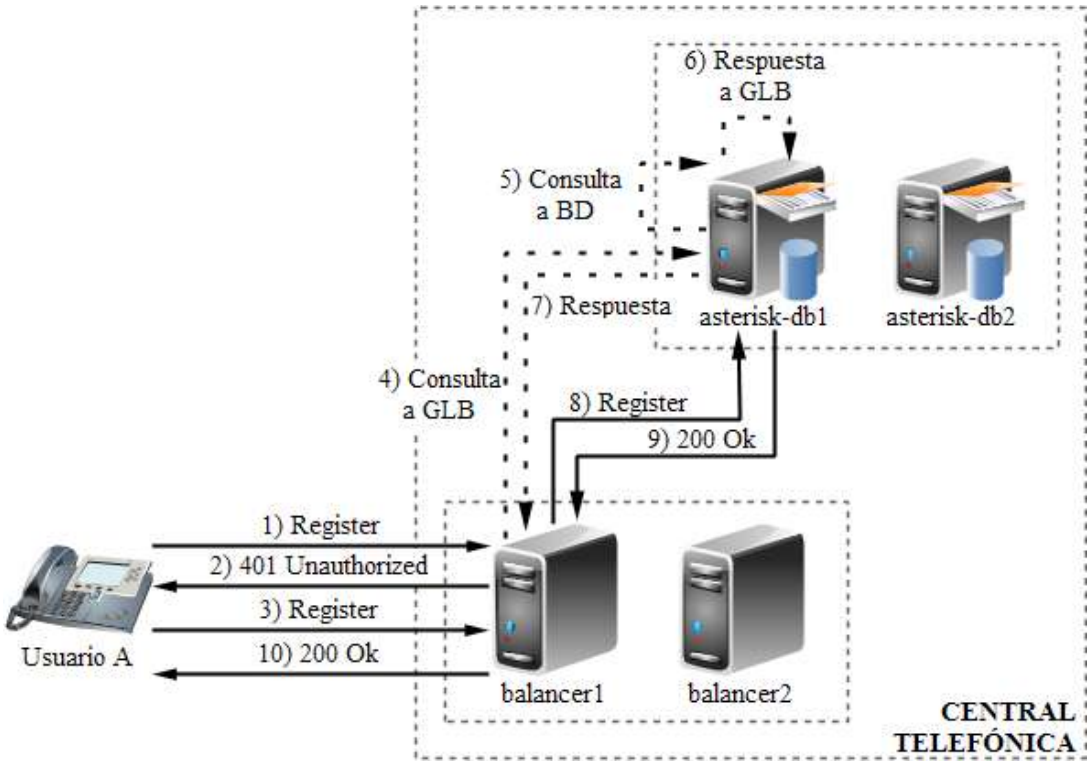

Fig. 5.8 – Proceso de registración en la topología desarrollada

Si bien en la sección 5.2.3 se explicó que Kamailio también almacena información de contacto del usuario en base de datos, el intercambio de mensajes correspondiente a este proceso no se ve reflejado en la Fig. 5.8 debido a como fue configurado Kamailio para tratar con el almacenamiento de la información. Kamailio almacena la información de contacto en memoria, y la sincronización con la base de datos se realiza en un intervalo de tiempo definido por un proceso “timer” ejecutado por Kamailio. De esta manera, se puede procesar las peticiones de registro lo más rápido posible.

Si no se utilizara la técnica de balanceo mediante la función hash sobre el “From-URI”, se podría dar el caso de que uno de los servidores Asterisk vea al usuario como “disponible”, mientras que el servidor restante lo vería “no disponible”. Supongamos que la técnica de balanceo utilizada fuese “Round-Robin”. Un usuario X contacta al nodo “balancer1” para registrarse, y la petición es balanceada al nodo “asterisk-db1”. Cuando la registración del usuario expira, el usuario X puede enviar una petición REGISTER para registrarse nuevamente, la cual sería balanceada al nodo “asterisk-db2”, y no produciría problema alguno si es que la información de contacto anterior ha sido eliminada de la memoria del nodo “asterisk-db1”. No se puede confiar en que la nueva información de registro del usuario siempre arribará luego de haber expirado la anterior, razón por la cual se optó por la técnica de balanceo mencionada.

5.4.2 Llamada exitosa

En la presente sección se muestra como se comunican los nodos de la central telefónica para poder concretar una llamada. En el ejemplo aquí expuesto, un usuario A desea comunicarse con un usuario B. El usuario A tiene acceso a los servicios de la central a través del nodo “balancer1”, y ha sido registrado en el nodo “asterisk-db1”, mientras que el usuario B posee acceso a través del nodo “balancer2” y ha sido registrado en el nodo “asterisk-db2”. Con las consideraciones anteriores, el intercambio de mensajes para el correcto establecimiento de la llamada se refleja en la Fig. 5.9.

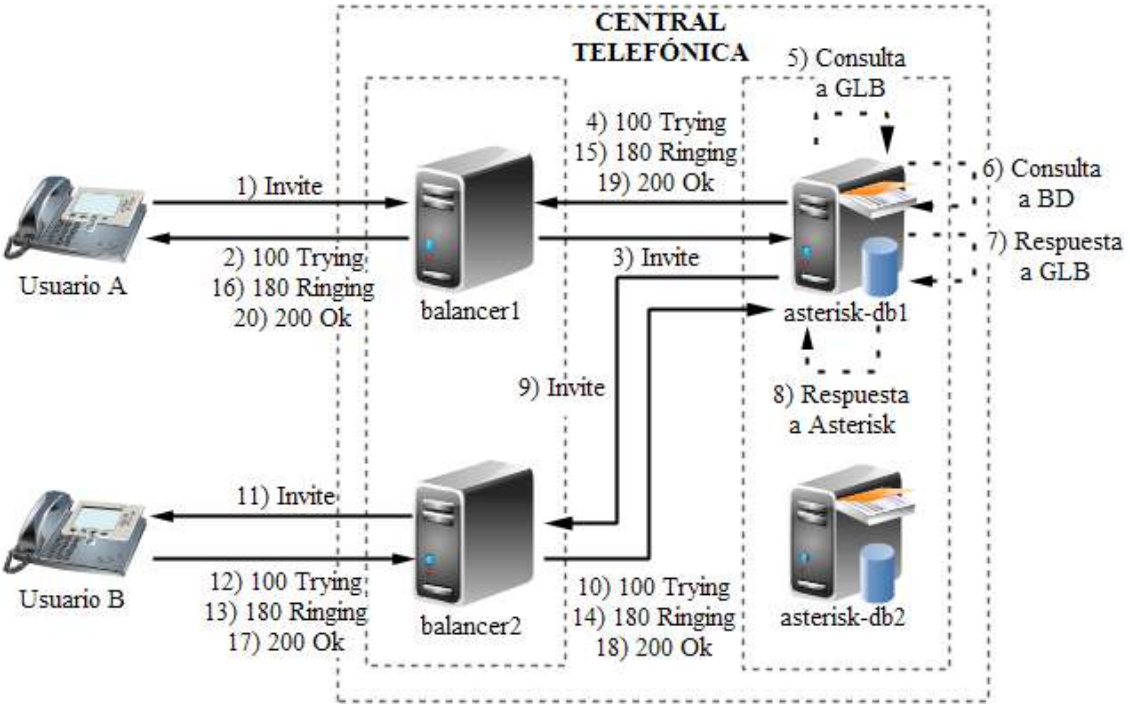

Fig. 5.9 – Establecimiento de una llamada en la topología desarrollada

Como se explicó anteriormente en la sección 5.2.3, los nodos balanceadores de tráfico son los encargados de autenticar a los usuarios, pero este proceso es idéntico para todas las acciones emprendidas por los usuarios, razón por la cual en la Fig. 5.9 se obvio el intercambio de mensajes necesario para la autenticación.

Una vez autenticado el usuario A, Kamailio aplica la política de balanceo “Round-Robin” y envía la petición INVITE al nodo seleccionado. En el ejemplo expuesto, el nodo elegido para atender la petición del usuario fue “asterisk-db1”. Al recibir la petición INVITE del nodo “balancer1”, Asterisk buscará en memoria al usuario objeto del mensaje. Como el usuario B fue registrado en el nodo “asterisk-db2”, Asterisk no poseerá su información de contacto, por lo que deberá realizar una consulta a la base de datos. Una vez obtenida la información necesaria, Asterisk crea un nuevo mensaje INVITE y lo envía al nodo “balancer2”, ya que es el nodo a través del cual el usuario B utiliza los servicios de la central telefónica.

En el proceso explicado en el párrafo anterior radica una de las principales diferencias respecto de (3) referente al tratamiento de las peticiones de los usuarios, ya que en el caso de que los usuarios participantes de una llamada se encuentren registrados en nodos distintos, no es necesaria la intervención de ambos nodos en el establecimiento de la sesión.

El nodo “balancer2” comenzará a intermediar entre el servidor Asterisk del nodo “asterisk-db1” y el usuario B reenviando los mensajes entre ambos extremos. Al recibir los mensajes del usuario B, Asterisk responderá al usuario A a través del nodo “balancer1”, hasta el momento en que el mensaje OK del usuario B arribe al usuario A, finalizando el intercambio de mensajes SIP e iniciando la comunicación a través del intercambio de mensajes RTP mediante RTPProxy, como se observa en la Fig. 5.10.

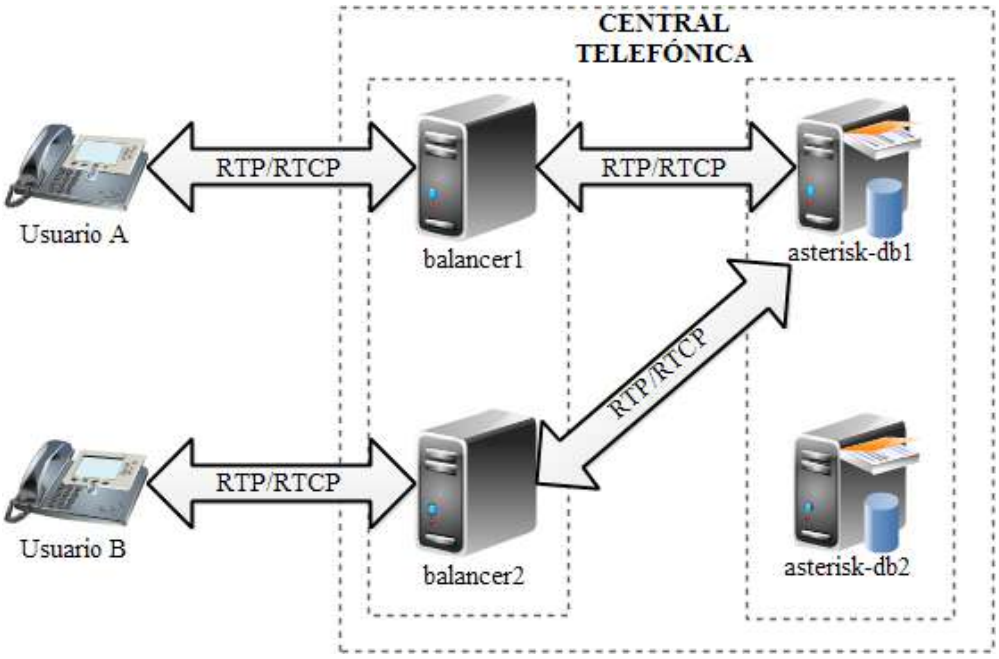

Fig. 5.10 – Flujo RTP en la topología desarrollada

Cuando cualquiera de las partes desee finalizar la llamada, enviará el mensaje BYE al nodo balanceador que corresponda, y este se encargará junto al servidor “asterisk-db1” de comunicar la finalización de la sesión al otro usuario.

5.5 Pruebas de Disponibilidad

Debido a que uno de los objetivos mencionados en la sección 1.3 consiste en dotar a la central telefónica diseñada del concepto de alta disponibilidad, resulta coherente realizar las pruebas pertinentes a fin de demostrar que el objetivo ha sido alcanzado.

Mediante las pruebas se comprobó el correcto funcionamiento de Pacemaker, realizando todas las acciones mencionadas en la sección 5.2.4 ante la caída de cualquiera de los componentes de software, garantizando que al menos siempre haya una instancia de software para resolver las peticiones, e intentando recuperar los componentes caídos siempre que fuese posible.

En lo referente al servicio de telefonía, se garantiza que la caída de cualquier de los servidores Asterisk no influye en la tarea de localización de los usuarios, ya que los datos necesarios pueden ser obtenidos por cualquier otro nodo Asterisk mediante una consulta a base de datos.

La caída de un Asterisk durante el transcurso de una llamada resulta en la pérdida inevitable de esta, debiendo ambos extremos de la llamada finalizar la sesión por su cuenta. Esto se debe a que una vez ha sido establecida una sesión, se definió que todos los mensajes deben pasar por el servidor Asterisk que ha atendido la solicitud del cliente. Si el componente que sufre de una falla durante el transcurso de una llamada es Kamailio, la llamada también se pierde por las mismas razones explicadas anteriormente, con la diferencia de que ahora si un extremo decide finalizar la llamada, el otro extremo recibirá el mensaje para acabar la llamada. Si durante la llamada el componente que refleja fallas es una de las instancias de RTPProxy, por lo explicado en la sección 5.2.4.3, la llamada en curso será interrumpida. La caída de los componentes mencionados es totalmente transparente para los usuarios cuando no se encuentran llamadas en curso.

5.6 Pruebas de Rendimiento

Otro de los objetivos a alcanzar en la realización de este proyecto es lograr un diseño capaz de adaptarse a diferentes ámbitos de aplicación. La implementación presentada en la sección 5.3 también fue sometida a pruebas de rendimiento. Concretamente, se realizaron pruebas respecto del número de llamadas por segundo recibidas y el número de llamadas simultáneas que la infraestructura es capaz de soportar, comprobando el consumo de ancho de banda, porcentaje de utilización de CPU y utilización de la memoria por parte de los principales componentes de software utilizados. Las pruebas fueron realizadas utilizando dos de las codificaciones más utilizadas en implementaciones de VoIP: GSM (26) y G.711-Ley A (27).

Las pruebas fueron realizadas utilizando el software SIPp (28). SIPp es una herramienta capaz de generar tráfico SIP, la cual fue optimizada para pruebas de tráfico y rendimiento. Soporta todos los métodos definidos en (10), y soporta una gran variedad de protocolos de transporte tales como UDP, TCP, TLS y SCTP. Las últimas versiones de esta herramienta permiten el envío y recepción de flujo RTP, transportando audio o video indistintamente.

El primer paso en la realización de las pruebas se basó en comprobar el límite teórico de llamadas simultáneas capaz de soportar los enlaces de la central telefónica, respecto del consumo de ancho de banda de una llamada codificada en GSM y G.711-Ley A. Si bien el consumo de ancho de banda depende en gran medida de la codificación utilizada, también existen otras variables que dificultan el cálculo correcto del ancho de banda utilizado. Como se explicó en la sección 5.3, el *switch* utilizado únicamente posee cuatro puertos (25) capaces de soportar la velocidad de transmisión máxima de las interfaces de red de los nodos servidores, lo que conlleva a tener que utilizar dos puertos de 100 Mbit para dos de las interfaces de red disponibles, limitando la capacidad de llamadas de la central a la capacidad que puedan soportar esos enlaces de 100 Mbit/s.

Una de las variables mencionada en el párrafo previo es el “*overhead*” producido por el agregado de encabezados de los protocolos que transportan el audio (en el caso del presente trabajo, UDP sobre IPv4 sobre Ethernet). Otro factor a destacar es que en una comunicación normal incluye un porcentaje elevado de silencios, que no deberían producir ningún envío de mensajes. Esto significa que aunque se inicie una llamada con dos flujos RTP de 64 Kbps más el “*overhead*” acarreado por los protocolos de transporte, no todo el tiempo se estaría utilizando el ancho de banda total. La Tabla 5.2 muestra una comparativa entre las codificaciones GSM y G.711, en la cual se incluye el consumo de ancho de banda base de la codificación y el consumo pon “*overhead*” del protocolo de transporte (29). Los valores de la Tabla 5.2 corresponden a un único flujo de audio, mientras que una comunicación VoIP se encuentra compuesta por un flujo por cada participante.

Códec	Calidad	Utilización de CPU	Consumo Base	Consumo “overhead”
G.711	Buena	Muy baja	64 kbps	95.2 Kbps
GSM	Aceptable	Media	13 Kbps	44.2 Kbps

Tabla 5.2 – Comparativa de los códec G.711 y GSM

Como ejemplo, el verdadero consumo de ancho de banda para una llamada entre dos usuarios consumirá el doble de lo reflejado en la Tabla 5.2, es decir, 190.4 Kbps para G.711 y 88.4 Kbps para GSM. Debido al diseño realizado respecto al manejo de los flujos RTP, representado en la Fig. 5.10, una llamada entre dos usuarios podría ser vista como dos llamadas entre un usuario y el correspondiente servidor Asterisk. Esto sugiere que una llamada en la central telefónica desarrollada se compone por cuatro flujos RTP, resultando en un consumo de ancho de banda de 380.8 Kbps para G.711 y 176.8 Kbps para GSM.

Con la información presentada en párrafos anteriores, se puede calcular finalmente la capacidad de llamadas teórica de la central. En un enlace de 100 Mbps y sin la utilización de ningún mecanismo de supresión de silencio, utilizando G.711 se podrían concretar simultáneamente 262 llamadas, mientras que si se utilizase GSM se concretarían simultáneamente 565 llamadas. Por lo tanto, la capacidad máxima de llamadas simultaneas a soportar sería el doble de lo mencionado anteriormente, es decir, 524 llamadas utilizando G.711 y 1130 utilizando GSM. Estos valores son posibles debido a que las interfaces conectadas a los puertos Fast Ethernet fueron las interfaces de los nodos Asterisk.

Debido a que el mismo enlace de red es utilizado por otros protocolos y aplicaciones, no fue posible alcanzar la cantidad de llamadas simultáneas, mencionadas anteriormente, pero los resultados fueron cercanos. Para la codificación G.711, se pudo mantener un funcionamiento correcto de la central con 500 llamadas simultáneas, mientras que utilizando la codificación GSM, se logró alcanzar las 1100 llamadas simultáneas.

Las pruebas consistieron en la simulación de llamadas mediante el establecimiento de una sesión SIP, durante la cual se intercambiará un flujo de mensajes RTP, entre dos instancias del software SIPp. Estas instancias se contactaban con la central a través de nodos balanceadores diferentes. A continuación, se analizará el impacto en el consumo de recursos de hardware, concretamente utilización de CPU y memoria RAM, de los componentes involucrados principalmente en la concreción de llamadas, Kamailio, RTPProxy y Asterisk.

Kamailio trabaja únicamente con el protocolo SIP, por lo que el momento de mayor utilización de CPU será durante el establecimiento y finalización de una llamada. Como las pruebas fueron desarrolladas en un ambiente cercano y aislado del resto de equipamientos del Laboratorio de Redes, los tiempos para el establecimiento y finalización de una sesión serán mínimos, resultando en un uso de CPU por muy cortos períodos de tiempo. La Tabla 5.3 muestra el promedio de utilización de CPU de cada proceso hijo ante el arribo de diferentes cantidades de llamadas por segundo, destacando que Kamailio fue configurado para ejecutar cuatro procesos hijos por dirección IP en la cual escuchan, resultando en un total de 8 procesos hijos.

Llamadas/s	CPU utilizada (%)
25	1,0
50	1,3
75	2
100	2,7

Tabla 5.3 – Porcentaje de utilización de CPU aproximada de cada proceso hijo de Kamailio

A diferencia de Kamailio, Asterisk y RTPProxy harán uso de CPU durante toda la duración de llamada, ya que son los encargados de encaminar el flujo de audio entre los extremos de la llamada. Por lo tanto, para obtener una buena muestra de datos se debería ver la utilización de CPU de ambos software durante un periodo de tiempo prolongado. Las Fig. 5.11 y Fig. 5.12 ilustran los valores mínimos y máximos aproximados de utilización de CPU de RTPProxy y Asterisk, para diferentes cantidades de llamadas simultáneas.

Los valores mostrados corresponden a un período de 50 segundos durante el cual la central atendió de forma concurrente la cantidad especificada de llamadas. Como se mencionó anteriormente, la duración de las llamadas de prueba era de un minuto, por lo que se decidió ir realizando llamadas de forma paulatina hasta lograr alcanzar la cantidad correspondiente a los 10 segundos de empezada la prueba. Es decir, se dividió la cantidad de llamadas objetivo por 10 para determinar la cantidad de llamadas por segundo a iniciar. Es decir, para alcanzar las 60 llamadas concurrentes, se realizaron 6 llamadas por segundos; para arribar a 120 llamadas se iniciaban 12 llamadas por segundo y así de manera sucesiva.

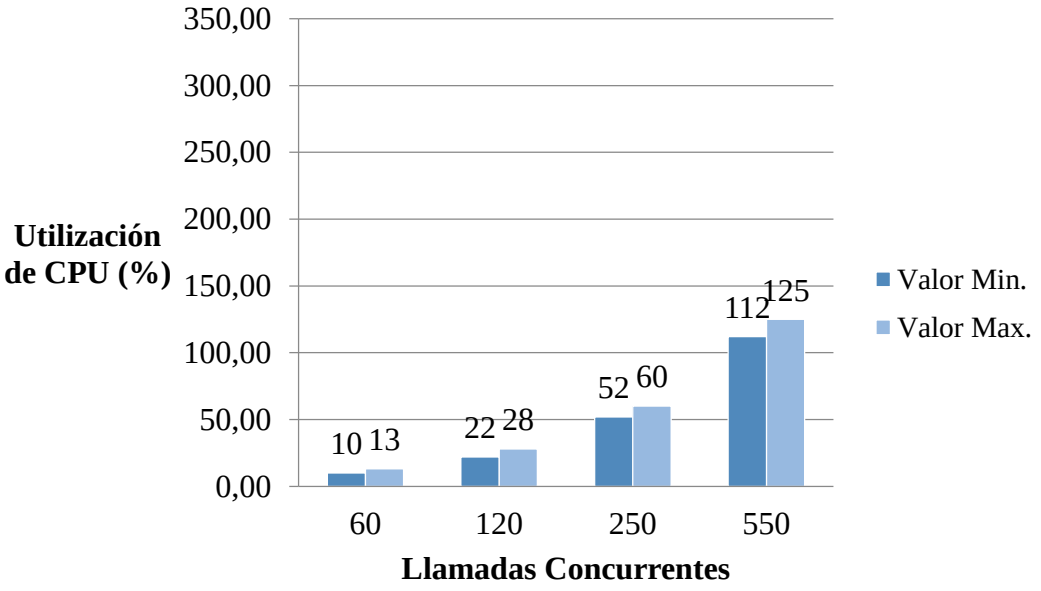

Fig. 5.11 – Porcentaje de utilización de CPU del software RTPProxy

Analizando los valores obtenidos, se observó que el consumo de CPU es muy similar para ambas codificaciones utilizadas. Este comportamiento se ve en ambos software. Esto se debe a que ambos extremos de la comunicación acordaron previamente la utilización de la misma codificación. Si esto no hubiese sucedido, los nodos Asterisk sufrirían un fuerte impacto en la utilización de CPU, ya que deberían recodificar el flujo antes de encaminarlo al cliente. Respecto al porque Asterisk llega a utilizar el doble de CPU o más comparado con RTPProxy, se debe a que Asterisk realiza un manejo más complejo y completo del flujo de audio que RTPProxy, que solo se encarga de retransmitir los paquetes.

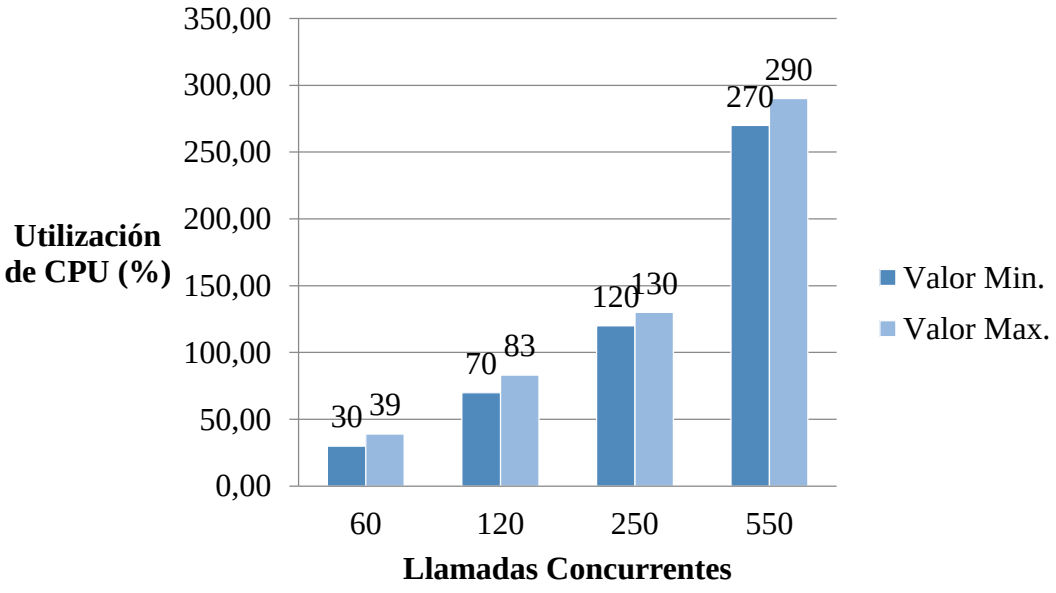

Fig. 5.12 – Porcentaje de utilización de CPU del software Asterisk

En cuanto al impacto en el uso de memoria RAM, se observó una utilización muy baja respecto de la capacidad máxima de los nodos servidores. Un detalle a tener en cuenta fue que para la realización de estas pruebas, se eliminó la restricción respecto de la cantidad de llamadas máxima que podría realizar un usuario, de forma tal que todas las llamadas tuviesen el mismo receptor y emisor. Esto fue realizado para evitar que el consumo de memoria por mantener información de diferentes usuarios no permitiese ver correctamente el consumo de memoria por almacenar información de las llamadas.

Para el software Kamailio, la memoria utilizada al inicio puede ser muy variable, dependiendo de las funcionalidades que se hayan implementado. En el caso de la central implementada, Kamailio fue configurado para ejecutar cuatro procesos hijos por dirección IP en la cual escuchan. Al iniciar el software, se observó un consumo inicial por proceso hijo de 0,1% (8 MB aproximadamente) de la memoria total. Al encontrarse 500 llamadas activas en simultáneo, se notó una utilización del 0,2% de la memoria total por proceso hijo. Para 1100 llamadas simultáneas, se observó que algunos proceso hijos alcanzaban el 0,4% de consumo mientras que otros utilizaban un 0,3% de la memoria total. A diferencia de la prueba de utilización de CPU, no se tuvo en cuenta la cantidad de llamadas por segundo recibida porque Kamailio mantiene en memoria la información de las sesiones activas para poder identificar rápidamente si el arribo de un paquete SIP se corresponde con una sesión ya establecida o no.

Respecto de RTPProxy, al iniciar su consumo de memoria es de apenas 2 MB. Como la información que almacena en memoria RTPProxy se encuentra ligada únicamente a la dirección a la cual enviar los paquetes RTP, el consumo fue similar para todas las pruebas realizadas. Al alcanzar las 500 llamadas concurrentes, se notó un incremento de apenas el 0,1% de la memoria total, mientras que al procesar 1100 llamadas simultáneas, la memoria utilizada llego al 0,2%.

Para el software Asterisk, análogamente a Kamailio, la memoria utilizada al inicio puede variar dependiendo de las funcionalidades implementadas. Para la central desarrollada, el consumo inicial es del 2,1% (172 MB aproximadamente). Al alcanzar las 500 llamadas concurrentes, se observó un consumo de 3,2% de la memoria total. Para las pruebas utilizando la codificación GSM, se notó que al alcanzar las 1100 llamadas simultáneas, el consumo de memoria había alcanzado un 4,6% del total disponible.

Otra prueba realizada respecto del consumo de memoria de los nodos servidores, se basó en la carga de información de contacto de 10 mil usuarios en un nodo Asterisk y en un nodo Kamailio. Las pruebas fueron realizadas luego de reiniciar ambos nodos para poder comprobar correctamente la memoria utilizada. En el caso del nodo Asterisk el uso de memoria fue de 6,2% (516 MB aproximadamente) del total disponible, es decir, el 4,1% (344 MB aproximadamente) fue utilizado para la información de los usuarios. Para el nodo balanceador, los procesos hijos escuchando en la dirección IP de entrada a la central apenas utilizaron un 0,46% (38 MB aproximadamente) de la memoria total, donde 0,36% (30 MB aproximadamente) se corresponden al almacenamiento de la información de los usuarios.

Capítulo 6
"Conclusiones"

La motivación del presente trabajo fue continuar la línea de los trabajos previos en la prosecución del objetivo de diseñar e implementar una central telefónica que presentara una alta disponibilidad en todos sus componentes (software y hardware), y que pudiera escalar fácilmente respecto del número de abonados a servir, con un alto desempeño en lo que se refiere al consumo de recursos. Primeramente, se estudió (2) para comprender el alcance de una central VoIP y sus beneficios. Luego, en la Cooperativa de Obras y Servicios Públicos de Rio Tercero, se realizó la presentación de la implementación de (3), a fin de compararla con una solución propietaria en un ambiente de producción. Las sugerencias derivadas de la presentación y el diálogo con los ingenieros a cargo del sector de telecomunicaciones de la cooperativa fueron el punto de partida para el rediseño de central telefónica presentado en el presente trabajo.

Se rediseñó (3) por las falencias halladas en su diseño, descritas en la sección 1.2; que no permitían arribar al objetivo deseado. El modo de replicación (maestro-esclavo) utilizado en los módulos de base de datos y balanceo de tráfico limitaba la capacidad de crecimiento de ambos módulos. La configuración de alta disponibilidad (activo-pasivo) de los módulos de localización y de base de datos no contempla la distribución de la carga de trabajo y pueden sufrir una sobredemanda de procesamiento. Además, la información de los clientes, que se encuentra distribuida entre los módulos de registro y de localización; carece de replicación, lo cual implica que ante un eventual falla en cualquiera de los componentes en los módulos, se perderá la información de registro de los clientes.

El aporte de este trabajo para solucionar las fallas mencionadas consistió en rediseñar la central en un esquema totalmente modular, en el cual cada módulo tuviese limitado su alcance en cuanto a funcionalidad a realizar, además de evitar que compartieran el mismo componente de hardware. De esta forma, y con respecto al diseño de (3), se definió separar el servicio de base de datos del módulo de localización para crear el módulo de base de datos. La capacidad de localizar usuarios fue re asignada a los servidores del módulo de registro, el cual fue renombrado a módulo de administración de servicios, ya que los nodos que lo conforman son los encargados de servir todas las funcionalidades de la central.

El módulo de base de datos fue rediseñado utilizando un esquema de replicación maestro-maestro con una configuración de disponibilidad activo-activo, permitiendo que el módulo pueda escalar a más de dos servidores, además de permitir operaciones de lectura y escritura en cualquier nodo miembro del módulo. El módulo de administración de servicios también fue configurado para trabajar de forma activo-activo. Además se permitió que la información de registro de los usuarios estuviese disponible para cada nodo miembro de este módulo mediante su almacenamiento en los nodos de base de datos, evitando de esta forma el problema de la pérdida de esos datos en casos de falla de un nodo. A nivel funcional, el módulo de balanceo de tráfico adquirió la capacidad de verificar la identidad de los clientes, con el fin de que solo sean atendidas las peticiones de clientes válidos.

Otro aporte realizado por el presente trabajo ha sido la adhesión de nuevos componentes de software y el reemplazo de otros porque se encontraban obsoletos o porque no ofrecían las funcionalidades necesarias para los módulos definidos. En el caso del módulo de base de datos, se reemplazó MySQL por un derivado del mismo, MariaDB, y se incorporó los componentes Galera Cluster, para lograr la replicación maestro-maestro y permitir la escalabilidad del módulo; y Galera Load Balancer, para balancear la carga entre los nodos del módulo. En el módulo de balanceo de tráfico, se reemplazó el balanceador genérico por Kamailio, un router del protocolo SIP; con el fin de poder controlar mejor el tráfico entrante para la utilización del sistema telefónica, y se agregó el componente RTPProxy, el cual se encarga de gestionar el flujo multimedia entre los clientes y la central telefónica. El módulo de administración de servicios, al igual que los trabajos anteriores, siguió basándose en el software Asterisk. Además, se agregó el componente Pacemaker a todos los módulos, el cual se encarga de dotar de alta disponibilidad a cada uno de los módulos mediante la administración de los componentes de cada módulo mencionados anteriormente.

Finalizada la etapa de rediseño de la central, se realizó una implementación utilizando los componentes de hardware disponibles en el Laboratorio de Redes de la Facultad de Ingeniería para ser sometida a pruebas de disponibilidad y rendimiento, con el fin de comprobar la consecución de los objetivos perseguidos. Respecto de la disponibilidad del sistema, las pruebas permitieron comprobar que el uso de Pacemaker cumple el objetivo de que en todo momento exista al menos un componente de software en cada módulo para poder seguir ofreciendo el servicio.

Respecto a los resultados obtenidos en las pruebas de rendimiento, se pudo demostrar que la principal limitación se encuentra en la infraestructura de interconexión utilizada para las pruebas. Esto se debe al diseño respecto del manejo de los flujos RTP, el cual se concluye debe ser rediseñado para evitar el consumo excesivo. Si se hubiese dispuesto de al menos dos puertos Gigabit Ethernet más, se cree factible el hecho de que el recurso de hardware que hubiese limitado la cantidad de llamadas simultáneas fuese el consumo de procesamiento de los nodos del módulo de administración de servicios.

A futuro es posible realizar implementaciones superadoras para lograr un diseño concluyente para sistemas de producción. Un ejemplo sería el desarrollo de una interfaz gráfica para la configuración de los diferentes componentes de software, aprovechando el hecho de que parte de las opciones de configuración son almacenadas en bases de datos. Otra implementación complementaria, de carácter obligatorio para un sistema en producción; sería el diseño de un esquema de seguridad que garantice la integridad y la confiabilidad del servicio prestado.

Bibliografía

(1) **Yanover, D.** (2007). Análisis del Estudio "*Estudio sobre Telefonía IP en Argentina de Prince & Cooke*"
http://www.mastermagazine.info/articulo/9764.php

(2) **Müller, E. y Franco, L.** (2009). *"Implementación de Servicios de VoIP utilizando Asterisk"*. Tesina de Grado. General Pico - La Pampa : Facultad de Ingeniería - UNLPam.

(3) **Fredes, M., Mora, N. y Mora, H.** (2012). *"Implementación de una Central Telefónica VoIP de Alta Disponibilidad en un Ambiente Distribuido y de Calidad".* Tesina de Grado. General Pico - La Pampa : Facultad de Ingeniería - UNLPam.

(4) **Spencer, M.** (2016). *Asterisk.* Digium Corporation. http://www.asterisk.org/.

(5) **Colouris, G., Dillimore, J., Kindberg, T. y Blair, G.** (2012). *"Distributed Systems: Concepts and Design"*. Quinta Ed., Addison-Wesley. ISBN-13: 978-0-13-214301-1.

(6) **Tanembaum, A. y van Steen, M.** (2006) *"Distributed Systems: Principles and Paradigms"*. Segunda Ed., Pearson. ISBN-13: 978-0132392273.

(7) **Lamport, L.** (1987).
http://research.microsoft.com/en-us/um/people/lamport/pubs/distributed-system.txt.

(8) **Marcus, E. y Stern, H.** (2003). *"Blueprints for High Availability"*. Segunda Ed., Wiley Publishing, ISBN: 0-471-43026-9.

(9) **ReliaSoft Corporation.** (2007) *Weibull.*
http://www.weibull.com/hotwire/issue79/relbasics79.htm.

(10) **Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. y E. Schooler.** (2012). RFC 3261. *"SIP: Session Initiation Protocol"*. http://www.rfc-editor.org/rfc/rfc3261.txt

(11) **Berners-Lee, T., Fielding, R., y L. Masinter.** (2005). RFC 3986. *"Uniform Resource Identifier (URI): Generic Syntax"*. http://www.rfc-editor.org/rfc/rfc3986.txt.

(12) **Resnick, P.** (2008). RFC 5322. *"Internet Message Format"*.
http://www.rfc-editor.org/rfc/rfc5322.txt.

(13) **Handley, M., Jacobson, V., y C. Perkins.** (2006). RFC 4566. *"SDP: Session Description Protocol"*. http://www.rfc-editor.org/rfc/rfc4566.txt.

(14) **Schulzrinne, H., Casner, S., Frederick, R., y V. Jacobson.** (2003). RFC 3550. *"RTP: A Transport Protocol for Real-Time Applications"*.

http://www.rfc-editor.org/rfc/rfc3550.txt.

(15) **Widenius, M.** (2016). *MariaDB.* MariaDB Foundation. https://mariadb.org/.

(16) **Jaakola, S., Yurchenko, A. y Ollakka, T.** (2014). *Galera Cluster*. Codership. http://galeracluster.com/.

(17) **Zaitsev, P.** (2011). *O'reilly.* http://en.oreilly.com/mysql2011/public/schedule/detail/17637.

(18) **Jaakola, S., Yurchenko, A. y Ollakka, T.** (2014). *Github*. Codership. https://github.com/codership/glb.

(19) **Mierla, D. y Modroiu, E.** (2016). *Kamailio.* http://www.kamailio.org/.

(20) **Sobolev, M.** (2016). *Sippy Software.* Sippy Software, Inc. http://www.rtpproxy.org/.

(21) **Cluster Labs.** (2015). *Clusterlabs.* http://clusterlabs.org/.

(22) **Haas, F.** (2010). *The Linux-HA Project.*.

(23) **Dake, S., Caulfield, C. et al.** (2014). *Corosync.* http://corosync.github.io/corosync/.

(24) **Jaakola, S., Yurchenko, A. y Ollakka, T.** (2014). *Galera Cluster Weighted Quorum.* Codership. http://galeracluster.com/documentation-webpages/weightedquorum.html.

(25) **Linksys.** *Andover Consulting Group.* http://www.andovercg.com/datasheets/linksys-cisco-SRW224G4-switch.pdf.

(26) **European Telecommunications Standards Institute (ETSI).** *ETSI.* http://www.etsi.org/technologies-clusters/technologies/mobile/gsm.

(27) **ITU-T.** (2011). *ITU-T.* http://www.itu.int/rec/T-REC-G.711/.

(28) **Gayraud. R, Jacques. Olivier.** (2014). *SIPp.* http://sipp.sourceforge.net/.

(29) **A, Psyllos.** (2014). *3CX.* http://www.3cx.com/blog/docs/bandwidth-utilised-for-voip/.