2424 SISTEMA DE SERVIDORES VOIP DEL PROYECTO GUIFI.NET

Memoria del proyecto de final de carrera correspondiente a los estudios de Ingeniería Superior en Informática presentado por Tomás Velázquez García y dirigido por Gabriel Abadal Berini.

Bellaterra, Septiembre de 2010

El firmante, Gabriel Abadal Berini, profesor del Departamento de Ingeniería Electrónica de la Universidad Autónoma de Barcelona

CERTIFICA:

Que la presente memoria ha sido realizada bajo su dirección por Tomás Velázquez García

Bellaterra, Septiembre de 2010

Firmado: Gabriel Abadal Berini

A los que cada noche me aconsejáis que ya es hora de que me vaya a dormir.

Agradecimientos

A todos aquellos que a lo largo de mi vida me han aportado algo, sabéis quienes sois y no creo que sea justo poner algunos nombres, porque no hay espacio para poneros a todos, por lo que prefiero agradecer por colectivos.

En este caso agradecer especialmente a Gabriel Abadal por aceptar la dirección del proyecto y aportarme su opinión y confianza.

Seguidamente a mi familia por la buena educación que me han dado, apoyo, afecto y gran paciencia que tienen conmigo, además de todos los esfuerzos y sacrificios que han tenido que hacer para que yo tenga una mejor vida que ellos.

Y para finalizar a mis compañeros de facultad de los primeros años y a los de esta última etapa por hacer más llevadero el esfuerzo de la carrera, a todos los que forman APXL y guifi.net por demostrar la fuerza que tiene la sociedad y lo que puede llegar a hacer unida, al equipo BCS y a los colectivos afines a la curiosidad, investigación y desarrollo.

Mi forma de ver la vida ha estado marcada por vosotros, gracias a todos!

Índice general

Índice de figuras

Capítulo 1

Presentación

1.1. Introducción

La telefonía IP es una tecnología que permite realizar llamadas telefónicas sobre redes de paquetes en lugar de utilizar las redes conmutadas de la telefonía tradicional.

En los últimos años se está viendo como la telefonía IP se va imponiendo a la telefonía conmutada. Esto es debido a las ventajas de integrar los servicios en una misma estructura de red con voz y datos proporcionando un mantenimiento más sencillo y a un coste más reducido, ya que las llamadas dentro de la misma red son gratuitas y las externas son más baratas que realizarlas por la red telefónica conmutada.

Por otro lado está el medio utilizado para la transmisión de las llamadas que será la red guifi.net, esta utiliza tecnología inalámbrica debido a que es barata y permite conectar dispositivos de forma fácil al utilizar ondas de radio.

El objetivo de guifi.net es realizar la red de telecomunicaciones de última milla que los ISP no han hecho porque no les es rentable, ya que invertir en el despliegue de xDSL o fibra óptica es difícil de amortizar si hay pocos usuarios y están aislados de los núcleos de población.

En guifi.net se utilizan protocolos abiertos, formatos libres y software libre por las ventajas que tienen a la hora de estudiarlos y mejorarlos en comparación con

los sistemas cerrados, por tanto, en este proyecto únicamente se estudiarán estos.

1.2. Motivación

El desarrollo de este proyecto esta incentivado por querer dar servicios avanzados de telefonía a la red creando un sistema de configuración automático de servidores VoIP a partir de la Web de guifi.net que permita:

- Comunicar con otros usuarios de guifi.net a través de telefonía utilizando la propia red sin necesidad de los servicios de un operador.
- Permitir que zonas donde llega guifi.net pero que no tienen tendido telefónico ni cobertura de telefonía móvil puedan llamar a través de un proveedor VoIP.
- Realizar llamadas a bajo coste a teléfonos fijos (0,01€/min dependiendo del proveedor) sin coste de establecimiento de llamadas y llamadas gratuitas entre cualquier teléfono VoIP del mundo.

1.3. Objetivos del proyecto

El proyecto guifi.net tiene muchas centralitas con Asterisk, pero tiene varios problemas a resolver:

- No están todas conectadas entre ellas.
- No siguen ningún estándar de numeración.
- Altas y modificaciones de los números se hacen de forma manual.
- Uso de distribuciones Linux y servidores sólo dedicados para VoIP.
- Cada centralita depende de la habilidad de su administrador.
- No hay QoS por defecto ni en routers ni en servidores.

- Llamadas no son p2p entre teléfonos.

Por tanto, el objetivo del proyecto es crear un software que configure de forma automática las centralitas de la red guifi.net permitiendo crear un sistema que consiga:

- Comunicar todas las centralitas entre ellas con un sistema de numeración único para la red y que se genere de forma automática tanto la numeración como la configuración de los servidores.
- Permitir realizar llamadas externas contratando los servicios que empresas de VoIP ofrecen.
- Resolver los problemas actuales que hay con la administración manual.

Los servidores de la red utilizan sistemas GNU/Linux con hardware reciclado o de bajo coste, por tanto se creará un paquete software para este SO pensando siempre en optimizar los recursos utilizados.

La tarea de implementar un sistema VoIP es realmente complicada debido a que la documentación es muy extensa, muy dispersa y muy fragmentada en temas muy concretos.

Por lo que con este proyecto se investigarán los protocolos y las soluciones actuales para unificar la información disponible sobre las características más significativas de esta tecnología, creando un documento que sirva de referente a cualquier interesado en el funcionamiento de los sistemas de telefonía IP.

1.4. Estudio de viabilidad

1.4.1. Recursos

Recursos técnicos

Red inalámbrica: Se monta el proyecto sobre la red ya existente y se prepara para funcionar en un futuro con la expansión de la red por el territorio, el coste del material va a cargo de quien quiere tener acceso a la red.

Servidores: Actualmente 8 ordenadores reciclados o de bajo coste pagados por sus propietarios, los recursos son compartidos con más aplicaciones que corren sobre las máquinas.

Sistema Operativo: GNU/Linux por tanto es gratuito.

Centralita: es también software libre, por tanto gratuito.

Recursos humanos

El proyecto se ha dividido en tres tareas que serán las etapas a realizar:

Análisis: se estudiarán los requisitos técnicos, los recursos económicos, el estudio de viabilidad, la investigación y la redacción de la documentación.

Desarrollo: lleva a cabo los resultados determinados por el análisis.

Pruebas: test general para comprobar la funcionalidad y la estabilidad del sistema.

1.4.2. Análisis económico

- Coste de los recursos técnicos.

Recurso	**Coste (€)**
Red Wifi, servidores y software	0

Cuadro 1.1: Coste recursos técnicos

- Coste de los recursos humanos.
    - Coste del personal por hora.

Recurso	**Coste (€/h)**
Analista	25
Desarrollador	18
Pruebas	12

Cuadro 1.2: Coste personal/h

- Coste del personal total.

Recurso	Tiempo (h)	Coste (€/h)
Analista	180	4500
Desarrollador	90	1620
Pruebas	30	360
Total	300	6480

Cuadro 1.3: Coste personal total

- Coste total.

Recurso	Coste (€/h)
Técnicos	0
Humanos	6480
Total	6480

Cuadro 1.4: Coste total

1.4.3. Beneficios

Llamadas más económicas utilizando proveedores VoIP, sobretodo cuanto más lejos se hagan las llamadas. En VoIP no se suele cobrar establecimiento de llamada, ni cuotas de línea.

Según estadísticas las llamadas a fijos con proveedores VoIP resultan un 83 % más baratas de media y las llamadas a móviles un 56 %.

Llamadas gratuitas entre los usuarios de guifi.net ya que estamos en una red libre.

1.4.4. Riesgos

Las redes inalámbricas son muy susceptibles a fallos en comparación con las redes de cable, ya que el medio ambiente y las interferencias pueden provocar

cortes, una posible solución es tener redundancia en la red, si cae un enlace los protocolos de enrutamiento dinámico se encargan de enrutar por nuevos caminos los paquetes.

Las llamadas externas por Internet dependen de la conexión y la fiabilidad de estas, por suerte tenemos muchas salidas a Internet, pero se han de configurar con QoS para permitir priorizar el tráfico VoIP sobre cualquier otro tipo de datos.

Las comunicaciones a través de Internet se hacen normalmente a través de NAT, en guifi.net no hay puertas de enlace predeterminadas, por lo que la centralita tendrá que hacer de proxy del protocolo RTP.

1.4.5. Planificación

El proyecto se divide en las cuatro fases siguientes.

	Nombre	Duración	Inicio	Terminado
1	⊟Servidores VoIP guifi.net	132 days	1/03/10 8:00	31/08/10 17:00
2	Análisis	45 days	1/03/10 8:00	30/04/10 17:00
3	Desarrollo	43 days	1/05/10 8:00	30/06/10 17:00
4	Pruebas	44 days	1/07/10 8:00	31/08/10 17:00
5	Memoria	20 days	15/06/10 8:00	12/07/10 17:00

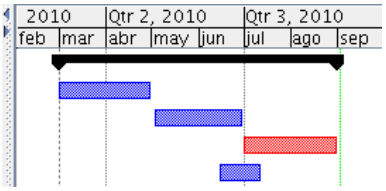

Figura 1.1: Planificación inicial

La duración del proyecto será de 132 días, finalizando el día 31 de Agosto. Se le dedicará una media de 3 horas al día de lunes a viernes.

1.4.6. Conclusiones

Este proyecto no sólo tiene como objetivo la configuración automática de los diferentes servidores VoIP para realizar llamadas internas y el ahorro en las llamadas externas, sino que también es necesario en lugares de España donde no hay ni líneas telefónicas como casas rurales y zonas de pobreza extrema incomunicadas como el Sáhara o otras poblaciones del tercer mundo donde se está empezando a hacer red libre.

El proyecto es totalmente viable ya que como se ha comentado proporciona numerosos beneficios para los usuarios de guifi.net, a parte de que la documentación

del proyecto proporciona un material resumido sobre tecnología VoIP que ayuda a cualquiera que quiera comenzar a entrar en este mundo.

Me gustaría también resaltar el hecho de que la red y los servidores ya están funcionando facilitando mucho todo el proceso de implementación, además de ser una ventaja ya que todo el material hardware no se tiene que comprar.

1.5. Estructura de la memoria

El trabajo se divide en 8 capítulos:

En este **Capítulo 1: Presentación** se han descrito las motivaciones del proyecto, los objetivos a conseguir, los costes y beneficios, y la planificación.

En el **Capítulo 2: Análisis** se hará un estudio del estado del arte de la red inalámbrica guifi.net y la tecnología VoIP, tanto de su arquitectura, protocolos y software que intervienen.

En el **Capítulo 3: Diseño e implementación** se explicará la solución adoptada y los motivos de la elección, y se hará un resumen de la implementación.

En el **Capítulo 4: Pruebas** se harán comprobaciones del sistema a nivel de red, del servidor VoIP y se verán que limitaciones existentes.

En el **Capítulo 5: Conclusiones** se describen los objetivos logrados en el proyecto y las líneas de futuro que podría tener.

En los **Apéndices A, B, C y D** se detallarán las implementaciones y configuraciones de los sistemas utilizados. Respectivamente corresponden a los temas de **Instalación** del sistema, **Optimizaciones** de la red, configuración del servidor VoIP **FreeSWITCH** y configuración de los servicios a través de la **Interfaz Web**.

Capítulo 2

Análisis

El análisis de este proyecto lo podemos dividir en cuatro apartados: el medio de transmisión utilizado y la red guifi.net se verá en la sección de **Wireless**, los protocolos utilizados para las señalización, transmisión y codificación de los datos se verán en la sección de **VoIP**, la forma de dar preferencia al tráfico se verá en la sección de **Sintonización de la red** y para terminar se hará un análisis sobre el **Servicio de directorios** utilizado para centralizar la configuración de los datos en un mismo servicio.

2.1. Wireless

Con el termino Wireless entendemos el concepto de sin cables refiriéndose a la comunicación inalámbrica en la que los extremos de la comunicación utilizan la modulación de ondas electromagnéticas a través del espacio.

Wi-Fi es una marca de la Wi-Fi Alliance organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.

El IEEE802.11 define las capas física y de enlace de datos de las redes inalámbricas tiene gran cantidad de estándares, pero nos centraremos en los más utilizados actualmente.

IEEE	Frecuencia (GHz)	Tasa de transferencia máxima teórica (Mbps)
802.11a	5	54
802.11b	2,4	11
802.11g	2,4	54
802.11n	2,4 y 5	150(1x1), 300(2x2), 450(3x3), 600(4x4)[1]

Cuadro 2.1: IEEE802.11

Entre las muchas ventajas respecto a otro tipo de redes podemos destacar el bajo coste de implantación y la baja complejidad del WiFi, además del hecho de no tener que pagar licencias por el uso de las bandas ya que son frecuencias libres. Por otro lado, que sean frecuencias libres tiene una desventaja debido al uso común ya que podrían haber interferencias de otras redes, actualmente pasa en 2,4Ghz.

2.1.1. Descripción del escenario de la red guifi.net

El proyecto está desarrollado para funcionar sobre la red guifi.net, una red mayoritariamente formada por enlaces wireless usando los estándares anteriormente comentados mayoritariamente en modo infraestructura.

La troncal se estructura de dos formas jerárquicas:

- **modo infraestructura** también llamado red descentralizada, donde hay dos tipos de conexiones:

  WDS P2P de esta forma llamamos a la conexión entre dos puntos de acceso (AP) que tienen cada uno una radio wireless dedicada para realizar el enlace. Se usa enrutamiento BGP y OSPF.

  Usuario este sería el modo tradicional de conexión WiFi donde existe un AP y varios clientes que se conectan a el.

[1] nxn significa que por cada extremo de la conexión hay n radios wireless en cada chip

Figura 2.1: Red descentralizada

- **modo ad-hoc** también llamado red distribuida, donde las dos radios están en modo ad-hoc de igual a igual sin ninguna jerarquía, todos con todos a los que llegue suficiente señal para poder mantener una comunicación. Se usan enrutamiento BMX y OLSR.

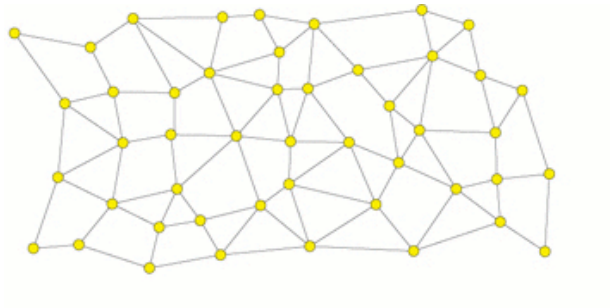

Figura 2.2: Red distribuida

La ventaja del modo ad-hoc con protocolos dinámicos es la conexión automática con nuevos routers que se incorporen y la flexibilidad de readaptarse a la ausencia de otros routers. Por consiguiente la desventaja es el ancho de banda ya que las conexiones son multipunto, necesitan más hardware que un router inalámbrico normal a causa del continuo recalculo de las rutas, los protocolos de enrutamiento son demasiado recientes y poco probados, no se sabe que escalabilidad tendrá en redes de miles de nodos.

La ventaja del modo infraestructura es la utilización de enlaces punto a punto dedicados que proporcionan el mayor ancho de banda posible, la utilización de

protocolos de enrutamiento dinámico estables ya que son los mismos que utiliza Internet desde hace casi 40 años. La desventaja de ser un sistema estático con enlaces p2p es que si hay una desconexión no habrá un cambio de conexión a otro router de forma automática, para conseguir redundancia en la red con este modo se añadirán otros enlaces estáticos a la red con otros puntos.

Por tanto, se decidió adoptar el modelo de red en infraestructura como forma de conexión normal y permitiendo la conexión de nubes en modo ad-hoc para los grupos minoritarios que lo desean.

En la red los usuarios tienen IP fija y se conectan siempre al mismo punto de acceso por lo que no está pensada para hacer roaming ya que necesitaría protocolos que garantizaran una conexión sin cortes al cambiar de AP como 802.11r que no es nuestro caso ya que en la red se usa 802.11abn y como se ha comentado conexiones siempre al mismo AP.

Como el tema principal del proyecto es la VoIP sólo se ha querido hacer un pequeño análisis de lo más importante, sin hacer estudio de otras capas que utiliza la red como son IP, funcionamiento interno de WiFi, ni de protocolos de enrutamiento.

Como datos interesantes podemos destacar que su crecimiento anual es superior a los 2500 nuevos usuarios y que actualmente está formado por más de 10500 nodos interconectados, siendo la red wireless libre, abierta y neutral más grande del mundo. Superando anualmente el Petabyte de transferencia de información interna de la red.

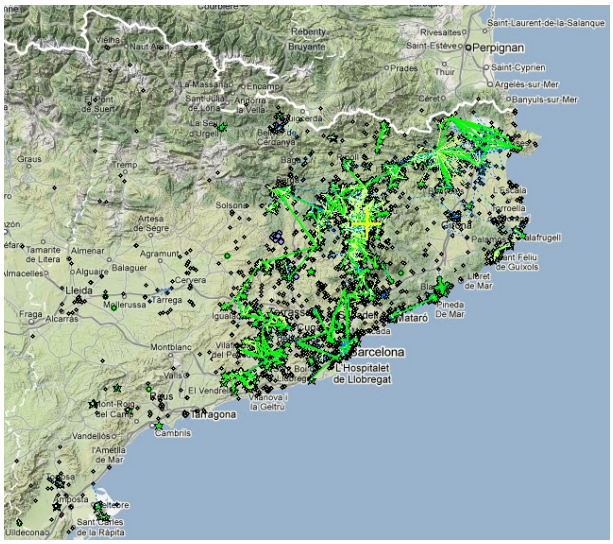

Figura 2.3: Mapa de las conexiones de la red guifi.net

2.2. VoIP

VoIP es la tecnología formada por normas, dispositivos y protocolos que permiten comunicar voz sobre el protocolo IP. Se envía la señal de voz digital formando paquetes en lugar de enviarla digital o analógica a través de circuitos conmutados que se utilizan únicamente para telefonía.

2.2.1. Conceptos básicos

Para entender el funcionamiento de las comunicaciones VoIP veremos las diferencias entre esta y la *RTC*. [1]

- **Redes de voz** conmutación de circuitos:
  - El establecimiento de llamada es necesario para realizar la conexión.
  - Los recursos de red se reservan durante todo el tiempo que dura la conexión.

[1] La Red Telefónica Conmutada es la red de comunicación telefónica tradicional diseñada para la transmisión de voz, aunque pueda también transportar datos.

- El ancho de banda utilizado es fijo y puede o no ser consumido en función del tráfico.
- El coste de las llamadas dependen del tiempo de uso.
- El servicio debe ser universal, independiente del operador que lo proporciona.

- **Redes de datos** conmutación de paquetes:
  - El direccionamiento por paquetes es necesario para asegurar la entrega de datos sin necesidad de establecer la llamada.
  - El consumo de los recursos de red se realiza en función de las necesidades, por lo general no se reservan recursos.
  - Los precios se forman en función de la oferta y la demanda.
  - Los servicios dependen de la demanda, variando en cobertura, velocidad y prestaciones.

Por tanto, implementar una red que integre voz y datos supone estudiar las diferencias entre ambas, comprender los problemas técnicos de los requisitos de los servicios y encontrar un punto de convergencia.

2.2.2. Historia VoIP

En el 1995 surgen los primeros programas de VoIP privativos con el objetivo de realizar llamadas y ahorrar dinero usando tarifas planas de datos.

Poco tiempo después se crearon software privativos para la construcción de gateways que permitían la comunicación entre la red IP de Internet y la RTC.

La primera ventaja que observaron los usuarios es la de poder llamar a grandes distancias pagando la tasa de acceso a Internet, en vez de pagar la cantidad impuesta por la compañía de RTC.

Otra ventaja que existe es la de poder utilizar la infraestructura que se posee para la telefonía habitual y el hecho de que VoIP no envía datos cuando encuentra

un silencio en la conversación, optimizando el ancho de banda utilizado, importante en estos años a causa de la baja velocidad de las conexiones.

Años más tarde con la evolución de la NGN surgen proveedores de VoIP que ofrecen servicios de telefonía permitiendo utilizar RTC a través de su infraestructura sin necesitar una línea analógica en los puntos finales. Esta idea es la que se utilizará para realizar las llamadas externas en este proyecto.

2.2.3. Componentes

Los podemos dividir en dos grupos según su función.

- **Terminales** que son el conjunto de hardware y software que utiliza el usuario para realizar la llamada que pueden ser de los siguientes tipos.
  - **Softphone** es un programa cliente de VoIP que funciona sobre un PC. Es la solución más barata si se dispone de un PC, porque sólo requiere micrófono y auriculares.
  - ***ATA***[2] es un conversor que conecta por una banda con un teléfono tradicional y por el otro extremo a una red Ethernet o WiFi.
  - **Teléfonos IP** es un teléfono que externamente parece uno tradicional, pero incluye conexión Ethernet o WiFi para conectar a la red IP.
- **Servidores** que son el conjunto de hardware y software que gestionan como realizar las conexiones entre los servidores y los terminales. El programa utilizado se suele llamar softswitch y es utilizado para controlar las conexiones en el punto de unión entre el circuito y las redes de paquetes o internamente en la red de paquetes. A la hora de clasificar estos dispositivos se hace una descomposición en Call Agent y en Media Gateway.
  - **Call Agent** se encarga de la facturación, enrutamiento, señalización, servicios de llamadas. Permitiendo controlar diferentes Media Gateways sobre un red IP.

[2] Analog Telephone Adapter

- **Media Gateway** conecta diferentes tipos de flujos de datos digitales creando una ruta de un extremo a otro que permite la comunicación entre ellos en una sesión. Puede tener interfaces para conectarse a Ethernet, RTC, ATM, etc.

2.2.4. Protocolos

Ya que VoIP encapsular la voz en paquetes para transportarlos por las redes vamos a estudiar los protocolos que intervienen.

La pila OSI describe como se encapsulan los datos en las diferentes capas desde la aplicación hasta el medio físico.

Capa	Descripción
Aplicación	Servicios de red a aplicaciones
Presentación	Representación de los datos
Sesión	Comunicación entre los dispositivos de la red
Transporte	Conexión extremo a extremo y fiabilidad de los datos
Red	Direccionamiento lógico (rutas e IP)
Enlace	Direccionamiento físico (MAC y LLC)
Física	Señal y transmisión binaria

Cuadro 2.2: Pila OSI

En nuestro proyecto utilizaremos 801.11 abgn en la capa física, Ethernet en la capa de enlace, IP en la capa de red, UDP en la capa de transporte, RTP en la capa de sesión y SIP en la capa de sesión y aplicación.

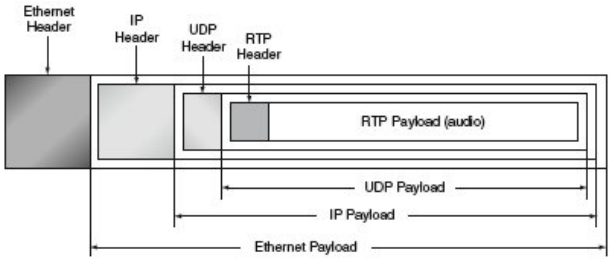

Figura 2.4: Encapsulación de los protocolos

La capa física ya ha sido explicada de forma resumida en la sección de Wireless y la capa Ethernet únicamente es utilizada entre los nodos que están conectados en un mismo rango de IPs, lo que sería la conexión AP-Cliente, entre rangos diferentes con conexiones AP-AP la comunicación es a nivel de IP.

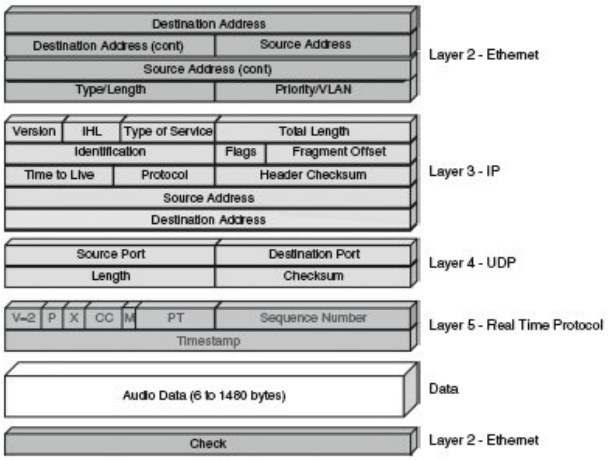

Figura 2.5: Cabeceras de los protocolos

A continuación se explicarán los protocolos desde la capa de red a la capa de aplicación que utiliza VoIP.

2.2.5. IP

Internet Protocol [IP] es un conjunto de procedimientos para la comunicación de datos a través de una red de paquetes.

Ya que es un protocolo best effort los datagramas no son fiables porque la prioridad es mandar la mayor cantidad de paquetes posible, sin preocuparse de la fiabilidad y de la latencia.

No está orientado a la conexión. No tiene funcionalidades para determinar si un paquete alcanza su destino y utiliza checksum únicamente en las cabeceras para proporcionar seguridad.

2.2.6. UDP

Los protocolos de capa de transporte más utilizados en la actualidad son Transmission Control Protocol y User Datagram Protocol [UDP].

Las diferencias entre ellos son las siguientes.

TCP

- Transporte fiable del flujo de datos. Evitando retransmisiones, pérdida de paquetes, orden en el que llegan los paquetes, duplicados de paquetes, etc
- Orientado a conexión.
- TCP asegura la recepción en destino de la información para transmitir.
- Sobrecarga de 20 bytes en cada segmento.

UDP

- Transporte no fiable del flujo de datos.
- No orientado a conexión, por tanto no añade retardos al establecer la conexión.
- No mantiene estado de conexión y no realiza seguimiento de estos parámetros.
- No hay confirmación, por lo que no se sabe si ha llegado o enviado correctamente.

- No tiene control de flujo, por lo que los paquetes pueden adelantarse unos a otros.
- Es más importante la velocidad que la fiabilidad.
- Sobrecarga de 8 bytes en cada segmento.

UDP es generalmente el protocolo utilizado en la transmisión multimedia en tiempo real a través de la red, ya que no hay tiempo para enviar de nuevo paquetes perdidos.

Hay que considerar que TCP y UDP utilizan la misma red y puede suceder que el incremento del tráfico UDP daña el correcto funcionamiento de TCP.

Los dos protocolos son importantes para las aplicaciones dependiendo los requerimientos que tengamos, por lo que encontrar el equilibrio entre ambos es crucial.

Para solucionar la ausencia del número de secuencia de UDP utilizaremos además el protocolo RTP como veremos en secciones posteriores.

2.2.7. Señalización VoIP

La señalización es el conjunto mecanismos para realizar una conexión entre terminales. Para realizarse requiere una serie de transacciones de entre los terminales para establecer los flujos de audio en ambas direcciones.

Los protocolos de señalización más importantes son H.323 y SIP que veremos a continuación.

H.323

El H.323 fue creado en 1996 por el ITU [H323] como protocolo para realizar comunicaciones multimedia sobre IP, así como aplicaciones en las que el funcionamiento con RTC es vital.

Fue el primer protocolos estándar de señalización, por eso fue adoptado ampliamente por ISPs y estos han seguido utilizándolo.

Componentes:

- **Terminal**: dispositivo de usuario final que facilita las comunicaciones con otro terminal, pasarela o MCU. El Terminal puede dar soporte de audio, vídeo y datos.

- **Gateway**: dispositivo que proporciona pasarelas de comunicaciones entre terminales H.323 y otros terminales de otras redes, realizando funciones de traductor entre ambas redes. Por ejemplo el paso de IP-RTC.

- **Gatekeeper**: controlador de acceso (este dispositivo es opcional), proporciona servicio de control de llamada a los puntos finales. En una red H.323 puede existir más de un gatekeeper que interaccionan entre sí. En cada una de las zonas puede haber un o más gatekeeper.

- **Multipoint Control Unit (MCU)**: soporta conferencias entre tres o más terminales y gateway en una conferencia multipunto. Una MCU puede negociar capacidades de los terminales en una conferencia y revisar las capacidades durante la conferencia para garantizar un nivel común de comunicaciones.

Fases de una llamada:

- **Establecimiento de la comunicación** el usuario envía un mensaje de SETUP, el remitente contesta con un mensaje de CallProcceding y Alerting indicando el inicio de establecimiento de la comunicación. Cuando el usuario descuelga el teléfono, se envía un mensaje de connect.

- **Negociación de los parámetros** con el protocolo H.245 (control de conferencia), el intercambio de los mensajes entre los dos terminales para establecer el maestro y el esclavo, las capacidades de los participantes y códecs de audio y vídeo. Luego se abre el canal de comunicación.

- **Comunicación** de los terminales utilizando RTP/RTCP.

- **Finalización de la llamada** con los mensajes CloseLogicalChannel y EndSessionComand.

SIP

El Session Initiation Protocol fue publicado en 1999 por el IETF [SIP] actualmente esa especificación esta obsoleta y ha sido actualizada en posteriores RFC.

Es un protocolo de señalización en la capa de aplicación para establecer y gestionar sesiones permitiendo múltiples participantes.

Se utiliza para comunicar dispositivos multimedia, principalmente conferencia, telefonía, presencia, notificación de eventos y mensajería instantánea.

Las funciones básicas del protocolo son determinar la ubicación de los usuarios, conocer la disponibilidad, aportar movilidad, y establecer, modificar y terminar sesiones entre usuarios.

Su diseño se basa en una estructura de petición y respuesta con una sintaxis simple, similar a HTTP o SMTP, con el objetivo de poder ser legible por las personas.

El protocolo se concentra en el establecimiento, modificación y terminación de las sesiones, complementándose con RTP/RTCP y SDP. El protocolo RTP se usa para transportar los datos de voz y el protocolo SDP se usa para la negociación de las capacidades de los participantes.

Componentes Las funcionalidades de SIP se pueden dividir en dos elementos los agentes de usuario y los servidores.

- **User Agent (UA)** este a su vez, está formado por las siguientes dos partes el User Agent Client y el User Agent Server.

  - **UAC** es una entidad lógica que genera peticiones SIP y recibe respuestas a esas peticiones.
  - **UAS** es una entidad lógica que genera respuestas a las peticiones SIP.

  Ambos se encuentran en todos los agentes de usuario para que puede existir la comunicación entre ellos.

- **Servidores** pueden ser de tres tipos: proxy, registrar y redirect.

- **Proxy Server** decide a que servidores debe retransmitir las solicitudes, alterando los campos de la solicitud. Es una entidad intermedia que actúa como cliente y servidor permitiendo establecer llamadas entre los usuarios. Existen dos tipos los Statefull Proxy y los Stateless Proxy.
  - **Statefull Proxy** mantienen el estado de las transacciones durante el procesamiento de las peticiones. Permite división de una petición en varias, con la finalidad de la localización en paralelo de la llamada y obtener la mejor respuesta para enviarla al usuario que realizó la llamada.
  - **Stateless Proxy** no mantienen el estado de las transacciones durante el procesamiento de las peticiones, únicamente reenvían mensajes.
- **Registrar Server** acepta peticiones de registro de los usuarios y guarda la información de estas peticiones para suministrar un servicio de localización y traducción de direcciones en el dominio que controla.
- **Redirect Server** genera respuestas de redirección a las peticiones que recibe, permitiendo reencaminarlas a otros servidores.

La división de estos servidores es conceptual, normalmente por motivos de escalabilidad y rendimiento. Dependiendo de los objetivos que tenga el servidor puede haber uno o múltiples combinaciones de estos en la misma aplicación.

Mensajes Los mensajes utilizados son en texto y parecidos a los del protocolo HTTP debido a que fueron diseñados para que la telefonía se vuelva un servicio más en Internet.

Los UAC realizan las peticiones y los UAS retornan respuestas a las peticiones de los clientes.

Las comunicaciones se realizan a partir de dos tipos de mensajes [IMF], las solicitudes (métodos) y las respuestas (códigos de estado) que consiste en una línea inicial seguida de un o más campos de cabecera, una línea vacía que indica el final de las cabeceras y el cuerpo del mensaje que es opcional.

Métodos Las peticiones SIP son caracterizadas por la línea inicial del mensaje, llamada Request-Line, que contiene el nombre del método, el identificador del destinatario de la petición (Request-URI) y la versión del protocolo SIP. Existen seis métodos básicos para que los clientes realicen peticiones.

Método	Descripción
REGISTER	Registra al User Agent.
INVITE	Invita a un usuario o servicio a participar en una sesión.
ACK	Confirma el establecimiento de una sesión.
BYE	Indica la finalización de una sesión.
OPTION	Solicita información sobre las capacidades de un servidor.
CANCEL	Cancela una petición pendiente.

Cuadro 2.3: Métodos de petición SIP

Sin embargo, existen otros métodos adicionales que pueden ser utilizados, publicados en otros RFCs como los métodos INFO, SUBSCRIBER, etc.

Respuestas Tras la recepción de una solicitud, el servidor envía una o varias respuestas. Cada respuesta tiene un código que indica el estado de la transacción. Éstas pueden ser de distintas clases tal y como vemos en la siguiente tabla:

Códigos	Clases
100 - 199	Informativa
200 - 299	Éxito
300 - 399	Redirección
400 - 499	Error cliente
500 - 599	Error servidor
600 - 699	Error Global

Cuadro 2.4: Códigos de respuesta SIP

Las respuestas llevan junto con el código de estado una frase legible infor-

mando del significado del código de la respuesta, esto se hace para que el usuario reconozca realmente lo que sucede, ya que el protocolo sólo tiene en cuenta para su funcionamiento el código.

Cabecera Las cabeceras se utilizan para transportar información necesaria a las entidades SIP. A continuación, se detallan los campos:

- **Vía**: indica el transporte utilizado para el envío e identifica la ruta del request, cada proxy añade una línea a este campo.
- **From**: indica la dirección del origen de la petición.
- **To**: indica la dirección del destinatario de la petición.
- **Call-Id**: identificador único para cada llamada que contiene la dirección del host. Debe ser igual para todos los mensajes dentro de una transacción.
- **Cseq**: se inicia con un número aleatorio e identifica de forma secuencial cada petición.
- **Contact**: contiene una o más direcciones que pueden ser usadas para contactar con el usuario.

Direccionamiento Una de las funciones más importantes es la localización de los usuarios ya que el User Agent no conoce la IP del destinatario de la llamada sino su URI. El Uniform Resource Identifiers [URI] tiene un formato muy parecido al del e-mail "sip:user@server:port"donde el puerto es un parámetro opcional.

Autentificación de acceso El Digest Access Authentication [WP-DAA] define su proceso así:

$HA1 = MD5(username : realm : password)$

si qop es 'auth' or no está especificado:

$HA1 = MD5(username : digestURI)$

si qop es 'auth-int':

$HA2 = MD5(method : digestURI : MD5(entityBody))$

si qop es ‘auth’ o ‘auth-int’:

$response = MD5(HA1 : nonce : nonceCount : clientNonce : qop : HA2)$

si qop no esta especificado:

$response = MD5(HA1 : nonce : HA2)$

Otros protocolos

Actualmente hay otros protocolos libres como Jingle creado por Google que son funcionales, pero que aun no están publicados como estándar en RTF por eso no serán analizados en este trabajo.

Recientemente IAX2, Inter Asterisk eXchange, ha sido reconocido como RFC, fue creado por la empresa Digium y licenciado de forma dual con licencia abierta y cerrada. Su función es la de señalización y transporte. Utiliza UDP como capa de transporte y a diferencia de la mayoría de protocolos no utiliza RTP para el tráfico en tiempo real y numerar las secuencia, sino que este proceso lo realiza el mismo. Puede truncar o empaquetar múltiples sesiones dentro de un flujo de datos. Además permite cifrar llamadas utilizando AES.

La empresa Skype ha liberado su códec, pero su sistema de señalización es cerrado, para tener interoperabilidad con software libre ha creado un modulo open source para Asterisk usando protocolo IAX que se conecta con los servidores de Asterisk y permite la interoperabilidad entre los dos.

2.2.8. SDP

Session Description Protocol [SDP] fue publicado en 1998 por el IETF como un protocolo de descripción de parámetros de inicialización de los flujos multimedia. Permite anuncio e invitación de sesión y negociación de parámetros.

Al principio se creó como componente del Session Announcement Protocol (SAP), pero se puede transportar mediante otro protocolos con SIP, RTSP, correo electrónico con aplicaciones MIME o HTTP y como formato de descripción de sesiones multicast.

Los mensajes de SDP están formados por líneas con atributos, estos tienen el formato carácter seguido de '=' y el valor o también existen parámetros opcionales definidos con '=*'. Los valores pueden ser una cadena ASCII, o una secuencia específica de tipos separada por espacios. En la tabla siguiente se especifica el formato usado.

Atributo	**Descripción 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 con información de ancho de banda
z=*	Ajustes de zona horaria
k=*	Clave de cifrado
a=*	Cero o más líneas de atributos de sesión
Atributo	**Descripción tiempo**
t=	Tiempo durante el cual la sesión estará activa
r=*	Cero o más veces de repetición
Atributo	**Descripción medios (si está presente)**
m=	Nombre de medio y dirección de transporte
i=*	Título
c=*	Información de conexión
b=*	Cero o más líneas con información de ancho de banda
k=*	Clave de cifrado
a=*	Cero o más líneas de atributos de sesión

Cuadro 2.5: Atributos SDP

2.2.9. RTP

Es un estándar publicado en 1996 [RTP] que especifica un formato de paquete para la transmisión de audio y vídeo sobre una red IP. Al ser un protocolo de sesión permite ser utilizado en protocolos de señalización.

El protocolo está formado por dos sub-protocolos:

- **Data Tansfer Protocol** que se ocupa de la transferencia de datos multimedia en tiempo real. Información proporcionada por este protocolo incluyen marcas de tiempo (para la sincronización), números de secuencia (para la detección de pérdida de paquetes) y el formato del payload que indica el formato de codificación de los datos.

- **Real Time Control Protocol** (RTCP) se utiliza para especificar la calidad de servicio (QoS) y la sincronización entre los flujos de datos. El ancho de banda del tráfico de RTCP es en comparación con RTP pequeño, alrededor del 5 %.

Una sesión RTP se establece para cada flujo multimedia, cada sesión consta de una IP con un par de puertos para RTP y RTCP. Los puertos que forman una sesión se negocian con otros protocolos como RTSP y SIP. RTP y RTCP suelen utilizar los puertos UDP sin privilegios (1024 a 65535), así como el diseño del protocolo de transporte es independiente.

Características principales de RTP

- Transporte del flujo multimedia en tiempo real.

- No reduce el retardo global, pero puede compensar el jitter.

- Las aplicaciones en tiempo real saben el orden de los paquetes.

- Reordenación de paquetes.

Una de las consideraciones de diseño de RTP era apoyar una amplia gama de formatos multimedia y permitir que los nuevos formatos se añadan sin necesidad

de revisar el estándar. Para cada tipo de aplicación RTP define un perfil y uno o más formatos de carga útil asociados.

Existen varios protocolos basados en RTP que añaden capas de seguridad estos son SRTP y ZRTP que veremos a continuación.

SRTP

Secure Real-time Transport Protocol fue publicado en 2004 [SRTP] como una extensión de RTP para proporcionar confiabilidad, autentificación e integridad, además de protección contra la reproducción de los datos RTP y RTCP en aplicaciones unicast y multicast.

SRTP es ideal para proteger el tráfico VoIP, ya que puede ser usada en conjunto con la compresión de cabecera y no tiene ningún efecto sobre la calidad de servicio. Esto proporciona ventajas para el tráfico de voz con códecs de baja tasa de bits como G.729 y iLBC.

Como algoritmo de cifrado de flujo de datos utiliza AES por defecto, pero puede permite la posibilidad de deshabilitar el cifrado completo si no es requerido.

Aunque SRTP podría utilizar fácilmente nuevos algoritmos de cifrado, no puede ser añadido en ciertas aplicaciones porque no garantiza la interoperabilidad con el protocolo estándar SRTP. La única manera para añadir un algoritmo de cifrado sin dejar la compatibilidad con el estándar SRTP es publicar un nuevo RFC.

ZRTP

ZRTP fue publicado en 2006 como una extensión de RTP [ZRTP] que describe el protocolo de intercambio de claves Diffie-Hellman durante el establecimiento de una llamada en el flujo de datos RTP que ha sido establecido empleando otro protocolo de señalización. Esto genera un secreto compartido que es usado para generar las claves y el salto para una sesión de SRTP.

ZRTP no requiere el intercambio previo de otros secretos compartidos o una Infraestructura de Clave Pública (PKI), a la vez que evita ataques de man in the middle. Además, no delegan en la señalización SIP para la gestión de claves ni

en ningún servidor. Soporta cifrado oportunista detectando automáticamente si el cliente VoIP del otro lado soporta ZRTP.

ZRTP puede usarse con cualquier protocolo de señalización ya que es independiente de la capa de señalización porque realiza toda su negociación de claves dentro del flujo de datos RTP.

Una extensión del protocolo ZRTP podría funcionar en cualquier tipo de redes de telefonía GSM, UMTS, RDSI, RTB, SATCOM, UHF / VHF, ya que es un protocolo de acceso indirecto de banda estrecha y diseñado para realizar todas las negociaciones clave dentro del flujo de bits entre dos extremos.

2.2.10. Códecs

Seguidamente se muestran una comparativa entre distintos códecs libres analizando tasa de bits y su frecuencia de muestreo considerando sólo el flujo de datos de voz, el overhead de las cabeceras del paquete no se ha contado.

Códec	**Tasa de bits (kbps)**	**Frecuencia de muestreo (kHz)**
G.711	64	8
G.726	16, 24, 32, 40	8
G.728	16	8
GSM	13	8
Speex	2.15-24.6 (NB); 4-44.2 (WB)	8, 16, 32
iLBC	13.3	8
SILK	6-40	8, 12, 16, 24
BroadVoice	16,32	8,16
CELT	24-128 (mono); 40-160 (stereo)	32-96
DVI4	32,64	8,16

Cuadro 2.6: Características de los códecs abiertos

El códec g711 tiene dos versiones U-law que es el estándar T1 usado en Estados Unidos y A-law que es el estándar E1 usado en el resto del mundo.

Aunque G.723 y G.729 son muy utilizados por el bajo ancho de banda que consumen no son incluidos ya que utilizan algoritmos patentados. Existen también otros códecs privados que pueden ser utilizados en ciertas condiciones de forma libre como Siren.

2.2.11. Servidores VoIP software

A continuación se hará un estudio de las soluciones VoIP basadas en software libre [VI-OSVS] [WP-CVS], ya no sólo porque el coste de este software sea nulo, sino porque al proporcionar el código fuente nos permite modificarlo y adaptarlo a cualquier necesidad que tengamos.

Con servidores VoIP software nos referimos al concepto de conmutar llamadas en el sentido más amplio de su palabra, desde *softswitch* [3] hasta una *PBX* [4]

Las ventajas de control y gestión de una red multiservicios que presenta el softswitch, hace que la arquitectura NGN se presente claramente como la evolución de la RTC comportándose como una PBX tradicional.

Como se puede observar por ejemplo en [VI-OSVS] existe gran cantidad de servidores VoIP dependiendo de las funcionalidades requeridas. A continuación se explicarán los las centralitas software de código abierto más conocidas y consolidadas.

Bayonne

Bayonne [BAY] es la solución VoIP desarrollada por el proyecto GNU, la primera versión fue publicada en 1998.

Soporta SIP, H.323, VoIP interno y hacia RTC, *IVR* [5] y una interfaz xmlrpc.

No ha tenido nunca desarrolladores con experiencia en el sector de la VoIP, es un software con funcionalidades muy limitadas, catalogándose más en una prueba

[3] Centralita software que proporciona control y procesamiento de llamadas en una red IP.

[4] Centralita software que conecta RTC con otras PBX y que permite llamadas internas, entrantes y salientes con autonomía sobre otra centrales telefónicas.

[5] Interactive Voice Response es la tecnología telefónica que permite detectar voz y tonos del teclado y interactuar con los menús configurados en la centralita.

de concepto que en un programa de uso real que implique fiabilidad, actualmente es mantenido únicamente por una desarrollador.

Asterisk

Asterisk es una PBX software que fue creada en 1999 porque las centralitas comerciales del mercado eran muy caras y su programador necesitaba una centralita de bajo coste ya que había creado una empresa de soporte telefónico para atender dudas sobre Linux, por lo tanto decidió crear su propia centralita para conmutar llamadas de RTC hacia sus teléfonos IP.

En la lista características de Asterisk [AST] se pueden ver todas sus funcionalidades, entre todas ellas se puede resaltar el uso de protocolos Google Talk, H.323, IAX, Jingle, MGCP, SCCP, SIP y de funcionalidades como correo de voz, conferencias, IVR, colas, Trunking, VoIP Gateways. Funciona sobre las plataformas Linux, BSD, Windows (emulado) y MacOSX.

Es la centralita de código abierto más utilizado por ser la primera en ofrecer funcionalidades avanzadas y porque hasta hace pocos años no existían alternativas libres que pudiesen competir.

Hay que tener en cuenta que este sistema fue pensando para manejar varias líneas RTC, del orden de decenas como máximo. Por lo que en entornos de mayor volumen de llamadas tiene problemas de escalabilidad y de estabilidad. Además de que la base no ha sido reprogramada en más de 10 años ya que se han ido añadiendo pequeños parches sobre el código original para añadir nuevas funcionalidades.

Otro de los problema que tiene es que no es un proyecto de código abierto en el que las empresas y programadores añaden de forma abierta mejoras. Sino que este sigue los intereses de la compañía Digium que son la venta de tarjetas para múltiples conexiones a RTC y la venta de una versión de Asterisk privada con ciertas funcionalidades añadidas a la versión abierta. Por estos dos motivos comentados se han creado múltiples *forks* [6] con el objetivo de añadir esas mejoras no permitidas, como el uso de IPv6, soporte de sonido de mayor calidad ya

[6] Desarrollo de software independiente a partir de otro software que se toma como base.

que sólo soporta 8KHz, etc. Aunque todos han fracasado y están prácticamente muertos debido a los pocos colaboradores y a que la base seguía siendo la misma y por tanto no escalaba ni era confiable. Entre este grupo destaca el proyecto CallWeaver que veremos a continuación.

CallWeaver

CallWeaver [CW] es un fork de Asterisk 1.2 que se realizó en 2005. Las principales diferencias [VI-CW] con respecto a Asterisk son que este soporta de forma incorporada *STUN*. [7]

Permite conectividad con RTC, múltiples protocolos de señalización, H.323, IAX2, MGCP, SIP y SCCP y de flujo de datos RTP, SRTP, además de T.38 Fax sobre IP, IVR y conferencias.

Utiliza los códecs SpanDSP, T.38 fax sobre IP, Sqlite en lugar de Berkeley DB, un buffer para el jitter universal y uso de temporizadores POSIX para no depender de la sincronización de Zaptel.

Las razones de crear este software fueron que la comunidad controlara la evolución de este, de forma que permitiera que cualquiera pueda participar, sin tener que renunciar a los derechos de autor y no depender de intereses comerciales antes que la calidad del desarrollo. Además rediseña parte de la estructura interna y se reutilizan las mejores librerías libres disponibles para no tener que reinventar la rueda como hace Asterisk creando las suyas propias. En general se centra más en la fiabilidad, las soluciones genéricas y la compatibilidad multiplataforma que en añadir nuevas funcionalidades.

De forma general incorpora multitud de plataformas soportadas Linux, FreeBSD, NetBSD, OpenBSD, MacOS X/Darwin y Open/Solaris.

[7] Simple Transversal of UDP over NATs permite a clientes NAT encontrar su IP y puerto públicos y el tipo de NAT en el que se encuentra para configurar una comunicación UDP entre dos hosts que se encuentren tras enrutadores NAT.

YATE

YATE [YATE] fue creado en 2004 basándose en ideas de Bayonne, pero no es un fork de otros proyectos.

YATE puede ser usado como aplicación VoIP cliente y servidor, gateway a RTC, gatekeeper, servidor de punto final multipe de H.323, controlador frontera de sesiones, servidor de registro de SIP, cliente y servidor de IAX, IVR y la posibilidad de incluir funcionalidades de pre/post pago.

Su potencial radica en la flexibilidad para poder ser ampliado y de utilizar voz, vídeo, datos y mensajería instantánea de forma unificada. Además es estable, escalable y portable, aunque prioriza el código claro antes que optimizar el rendimiento.

La escalabilidad es buena en algunos módulos ya que utiliza librerías de otros proyectos, pero en otros módulos utiliza librerías modificadas por ellos basadas en las librerías de otros proyectos, por lo que nuevas funcionalidades en la librería original no son añadidas a YATE.

Igual que las anteriores centralitas el grupo de desarrolladores es muy limitado y tiene el peligro de desaparecer si estos dejan el proyecto.

FreeSWITCH

FreeSWITCH [FS] fue publicado en 2005, en un principio formado por tres de los desarrolladores más activos de Asterisk que no formaban parte de Digium. Está diseñado para enrutar y interconectar protocolos de comunicación usando audio, vídeo, texto o cualquier otro formato. Su objetivo son básicamente construir desde cero un sistema más modular, multiplataforma, escalable y fiable que Asterisk.

FreeSWITCH es una librería con un pequeño ejecutable que carga esta librería, arranca el núcleo y realiza las tareas definidas en los módulos. Al ser una librería permite la integración con cualquier aplicación que necesite conmutación con VoIP y se puede hacer en varios lenguajes de programación.

Su diseño modular permite utilizar cualquier librería disponible de forma eficiente, ya que se pretende no reescribir librerías si ya existen.

Permite la construcción de un softphone, un sistema PBX, un softswitch o una interfaz de comunicación con otros sistemas VoIP de código abierto.

Es un B2BUA ya que permite ocultar la topología de la red, realizar transcoding y detectar interrupciones de llamadas. Además tiene IVR, colas, conferencias, buzón de voz, SBC (Session Border Controller), FAX sobre voz y T.38, etc.

Soporta los protocolos de red IPv4 e IPv6, varios protocolos de sincronización como SIP, H.323, Skype, Jingle, GoogleTalk, SCCP, etc, y de tiempo real RTP, SRTP y ZRTP, además de soportar multitud de códecs de voz.

Utiliza códecs de banda ancha y estrecha, por tanto es ideal para mantener la compatibilidad entre los dispositivos nuevos y viejos. Los canales de voz pueden funcionar en 8, 12, 16, 24, 32 o 48 kHz, y puede realizar transcoding entre usuarios de diferentes frecuencias.

Los archivos de configuración utilizan XML para facilitar el análisis.

Tiene capacidad para procesar miles de de llamadas simultaneas.

Funciona en Windows, Max OS X, Linux, *BSD y Solaris.

SIP Router

Para finalizar está el proyecto SIP Router que comenzó en 2008 y es la unión entre los proyectos Kamailio (OpenSER), SIP Express Router (SER) y OpenIMSCore con el objetivo común de colaborar de forma unida para crear un sotfware mejor.

Es un servidor SIP de código abierto capaz de manejar miles de llamada por segundo. Sus principal objetivos son desarrollar un sistema con núcleo flexible, extensible, escalable y estable.

Entre sus características destacan el uso de SIP, RPT, SRTP, XMLRPC control interface, monitorización SNMP. atravesar sistemas con NAT, retransmisión de paquetes RTP en el espacio del núcleo del sistema operativo, enrutamiento de menor coste, balanceo de carga, IPv4 e IPv6.

Puede ser utilizado para construir grandes plataformas de VoIP de alto rendimiento, flexibles y seguras.

Debido a que esta centrado en el uso del protocolo SIP, si se quiere conectar

a otro protocolo o a la RTC hará falta conectarlo con un Asterisk o FreeSWITCH ya que SIP Router no es un B2BUA.

Permite un rendimiento de más de 5000 llamadas/s en un máquina.

Las plataformas soportadas son Linux, SUN/Solaris, *BSD

Para finalizar esta sección en la tabla siguiente podemos ver una comparación entre diferentes servidores VoIP y las características soportadas.

Software	**PBX**	***Media Server***[8]	***B2BUA***[9]	**SIP rápido**	**Escalable**
Bayonne	Si	No	Si	No	No
Asterisk	Si	Si	Si	No	No
CallWeaver	Si	Si	Si	Si	Si
YATE	Si	No	Si	Si	Si
FreeSWITCH	Si	Si	Si	Si	Si
SIP Router	No	No	No	Si	Si

Cuadro 2.7: Características de los servidores VoIP de código abierto

2.3. Sintonización de la red

2.3.1. QoS

Quality of Service son el conjunto de tecnologías que garantizan la transmisión de información en un momento concreto priorizando un determinado tráfico sobre otros.

Es especialmente importante atenuar los efectos de la latencia y la pérdida de sincronía ya que la calidad se ve muy afectada, sobre todo es importante en flujos de datos multimedia en tiempo real y en redes wireless donde es normal la perdida de señal en algún momento.

Los problemas entre la transmisor y la receptor en redes de datos son los siguientes:

- **Paquetes sueltos** dependiendo del estado de la red, ya que los routers pueden fallar en liberar paquetes si llegan cuando los buffers ya están llenos.

- **Retardos** debidos a que los paquetes pueden permanecer en largas colas o tomar una ruta menos directa para prevenir la congestión de la red.
- **Jitter** causado porque los paquetes pueden llegar con diferentes retardos y producir microcortes en el flujo de voz.
- **Entrega de paquetes fuera de orden** a causa de que los paquetes pueden tomar diferentes rutas que proporcionan diferentes retardos. Para solucionarlo se necesita un protocolo que numere los paquetes.
- **Errores** producidos porque los paquetes son mal dirigidos, combinados entre sí o corrompidos cuando se encaminan.

Según especificaciones [G114] se recomienda que en voz la latencia en un solo sentido no debe exceder los 150 ms, ya que si se supera los retrasos son notables, pero tolerables, cuando la latencia supera los 250 ms se vuelve demasiado difícil de llevar a cabo una conversación.

Retransmitir estos paquetes no sería una solución ya que comportaría aumentar más el retraso, por lo que se prefiere omitirlo y continuar con el siguiente aunque haya Jitter. El receptor tiene que detectar estos fallos e intentar disminuir sus efectos utilizando colas diseñadas para limitar la velocidad, algoritmos de planificación, y políticas y colas de prevención de congestión.

A causa de las limitaciones de los sistemas pueden haber circunstancias en las que los requerimientos sobre excedan nuestras prestaciones y se deniegue el servicio o se ralentice todas las llamadas simultaneas, para limitar esto se utiliza CAC.

2.3.2. CAC

Call Admission Control limita el número de llamadas simultáneas en una red VoIP. Asegura que una llamada no puede ser creada si el sistema no la puede soportar. Rechaza las llamadas cuando la CPU no tiene suficiente capacidad de procesamiento, el tráfico ascendente y descendente excede los límites predefinidos, o el número de llamadas atendidas excede un límite especificado.

Se utiliza en la fase previa a la llamada y se aplica al tráfico multimedia en tiempo real.

Protege el tráfico de voz de los efectos negativos de otros tráficos de voz, para asegurar que hay un excedente de ancho de banda.

2.4. Servicio de directorios

Directory Service [WP-DS] es una sistema de información para compartir, localizar, gestionar, administrar y organizar los elementos comunes y los recursos de red, que pueden incluir volúmenes, carpetas, archivos, impresoras, usuarios, grupos, dispositivos, números de teléfono y otros objetos.

Un directorio es un conjunto de objetos con atributos organizados en una manera lógica y jerárquica. A manera de síntesis, LDAP es un protocolo de acceso unificado a un conjunto de información sobre una red.

Las diferencias entre un servicio de directorio y una base de datos relacional son que la información se lee con más frecuencia de lo que se escribe, por tanto las transacciones y el rollback son menos importantes y la segunda diferencia es que los datos pueden ser redundantes si incrementan el rendimiento.

Para estandarizar los datos se han creado unos esquemas que definen las clases de los objetos y los atributos.

Dependiendo del esquema los atributos pueden ser obligatorios o voluntarios, pueden tener múltiples atributos y cada objeto hereda las clases de sus objetos primarios.

Antes de la llegada de la estandarización de los servicios de directorio a través de X.500 ya existían algunos como el DNS y el NIS.

Posteriormente se crea LDAP basado en las especificaciones de X.500.

2.4.1. LDAP

Entre la multitud de servidores Lightweight Directory Access Protocol [LDAP] destaca OpenLDAP que deriva la implementación original de la Universidad de Michigan, pero evolucionado significativamente. Es compatible con Unix y sus

derivados, Windows, z/OS, y una variedad de embebidos y sistemas en tiempo real.

Estructura de directorio:

- Un directorio es un árbol de entradas de directorio.
- Una entrada consta de un conjunto de atributos.
- Un atributo tiene un nombre y uno o más valores. Los atributos son definidos en un esquema.
- Cada entrada tiene un identificador único el Distinguished Name (DN). Este consta de su Relative Distinguished Name (RDN) construido por algunos atributos en la entrada, seguidos del DN de la entrada del padre.

Un ejemplo de entrada en formato LDAP Data Interchange Format (LDIF) podría ser el siguiente.

```
dn: cn=Tomas Velazquez,dc=dominio,dc=com
cn: Tomas Velazquez
givenName: Tomas
sn: Velazquez
telephoneNumber: +94 93 690 8689
telephoneNumber: +94 555 98 1232
mail: tomas.velazquez@dominio.com
manager: cn=Tomas Velazquez,dc=dominio,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

Los servidores también pueden contener referencias a otros servidores, permitiendo alojar partes del árbol de directorios en estos otros servidores.

Algunos servidores también soportan encadenamiento (chaining), que permite que el servidor contacte a otro servidor y retorne el resultado al cliente.

También puede devolver referencias a otros servidores para efectuar nuevas peticiones que el servidor mismo no puede devolver.

Los nombres de entradas están organizados de la misma forma que está la estructura interna de la organización.

Las URL de LDAP sirven para realizar búsquedas en otros servidores y obtener entradas o referencias a servidores. Tienen el siguiente formato:

ldap://host:port/DN?attributes?scope?filter?extensions

De los componentes siguientes únicamente es obligatorio el host.

- **host** es el nombre o la IP del servidor LDAP.
- **port** es el puerto que utiliza el servidor LDAP.
- **DN** es el nombre distinguido a utilizado para la búsqueda.
- **attributes** es una lista separada con comas de atributos a devolver.
- **scope** es el ámbito de la búsqueda.
- **filter** es un filtro de búsqueda.
- **extensions** son nuevos componentes que se puedan añadir a este formato.

Un concepto muy importante de cara a tener un sistema distribuido jerárquicamente es la utilización de referencias hacia otros servidores LDAP que contienen información adicional de las ramas.

Existen diferentes tipos de referencias

- Al producirse un error causado por no encontrar ninguna coincidencia en la búsqueda se retorna una referencia.
- Cuando un cliente intenta actualizar un esclavo se devuelve una referencia.
- Delegación de subárboles retornando una referencia hacia ellos.
- Sigue las referencia en lugar de devolverla y retorna el objeto buscado si lo encuentra.

El último tipo, encadenamiento de referencias (referral chaining) [LDAP-RC] es el que nos permite mayor abstracción al utilizar subárboles y por eso será el utilizado. En la siguiente figura se puede ver un ejemplo del funcionamiento.

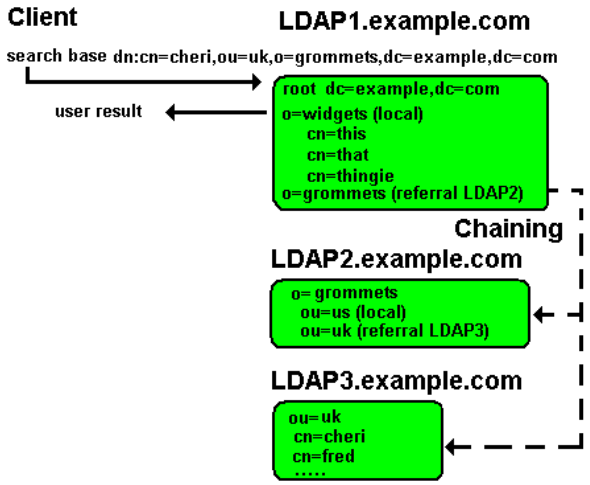

Figura 2.6: Encadenamiento de referencias

Capítulo 3

Diseño e implementación

En este capítulo veremos las soluciones propuestas para la realización del proyecto y un resumen de la implementación.

3.1. Servidores

Los servidores son compartidos con otros muchos servicios a parte del servicio de VoIP por lo que los requerimientos de estos podrían afectar al rendimiento de la centralita, para solucionar esto ya que queremos priorizar el servicio de voz sobre otros servicios tendremos que dar al proceso servidor de VoIP mayor prioridad que a los otros. Como realizar esto se puede ver en el Apéndice C.

3.2. Servicio de directorios

Ya que todos los servidores de guifi.net tienen múltiples servicios en la misma máquina y actualmente cada programa utiliza un fichero con todos los usuarios y contraseñas por cada aplicación.

Se ha construido un sistema de directorios basado en LDAP que proporciona por un lado unificar todos esos datos en un servidor local y que cada servicio acceda a este para realizar la validación.

El otro motivo es que con LDAP se pueden crear jerarquías de niveles al igual

que pasa con los dominios y los subdominios con lo cual conseguimos segmentar los datos en varios servidores ya que tenemos recursos finitos y además añadimos estabilidad ya que no tenemos un servidor de autorización centralizado.

Existen diferentes programas que realizan las funciones de servicio de directorios, pero se ha elegido LDAP por ser actualmente el más evolucionado y permitir característica avanzadas como el encadenamiento de referencias.

3.3. Sistema VoIP

3.3.1. Elección de protocolos

En la capa de transporte preferimos UDP sobre TCP porque no necesita establecer un proceso de inicialización en la comunicación, ni tiene verificación de recepción, ni número de secuencia, todo esto nos permitirá tener menos latencia.

Para solucionar estos problemas utilizaremos en la capa de sesión RTP, tanto SRTP como ZRTP aunque proporcionando la seguridad necesaria son descartados porque únicamente son compatibles con los últimos teléfonos VoIP que tienen un elevado coste.

A la hora de realizar la señalización tenemos varios protocolos.

Jingle [JINGLE] que al no estar estandarizados no creemos que sea una opción a elegir dado su poca madurez.

IAX2 tiene varias limitaciones, la necesidad de utilizar Asterisk para utilizar el protocolo, los trunks tienen límite de llamadas simultaneas y estamos forzados a utilizar códecs de 8Kbps.

Nos quedan H.323 y SIP que soportan casi los mismos servicios.

La complejidad entre los dos se puede ver en la documentación de las especificaciones iniciales donde H.323 tiene 700 páginas y SIP tiene 130 páginas.

A la hora de generar, codificar y depurar H.323 añade complejidad ya que sus mensajes están en binario, en cambio los mensajes de SIP se envían directamente como texto, igual que HTTP, FTP, SNMP, etc.

Una desventaja de SIP es que las nuevas versiones pueden eliminar caracterís-

ticas para reducir complejidad, en cambio en H.323 exige compatibilidad hacia atrás entre las versiones.

En resumen, ya que H.323 es más antiguo, más desplegado, más complejo y descrito con mayor detalle, en cambio SIP es más reciente, mayor crecimiento, más sencillo y más expansible. Se ha optado por utilizar este último, sobretodo por el crecimiento, ya que en el ámbito de la VoIP doméstica es difícil encontrar un dispositivo con H.323, en cambio todos soportan SIP.

3.3.2. Códecs

Los códecs utilizados en las llamadas internas dependerán de los códecs que ambos soporten y se utilizará el códec de mayor compresión que se pueda.

Los códecs utilizados en las llamadas externas dependerán de los códecs soportados por nuestro servidor VoIP y el proveedor, pero lo habitual es que el proveedor utilice G.711.

3.3.3. Llamadas

El registro en el servidor LDAP únicamente se realiza una vez al registrar el terminal en el sistema, en las figuras siguientes se muestra el proceso entero.

Las **llamadas internas** en este proyecto las podemos dividir en dos tipos según si utilizan uno o dos servidor para realizar la sincronización.

Con **un servidor** se realiza la sincronización de las llamadas en las que los usuarios están dados de alta en el mismo servidor, normalmente serán usuarios de la misma localidad geográfica.

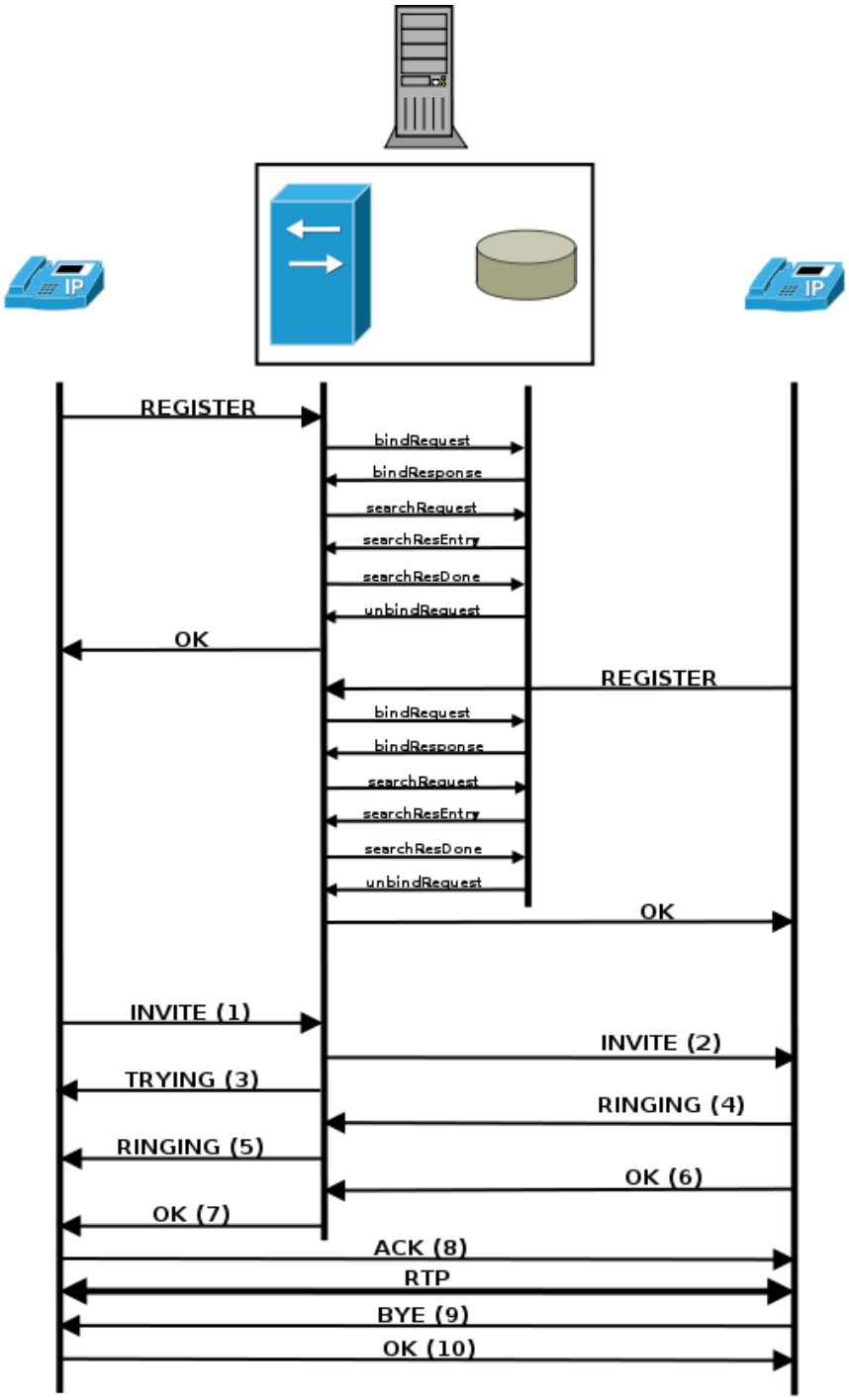

Figura 3.1: Llamada interna con el mismo servidor VoIP

Con **dos servidores** se realiza la sincronización de las llamadas cuando los

usuarios no están dados de alta en el mismo servidor y necesitan entre los dos servidores pasarse la información de sincronización, normalmente serán usuarios que están en pueblos diferentes.

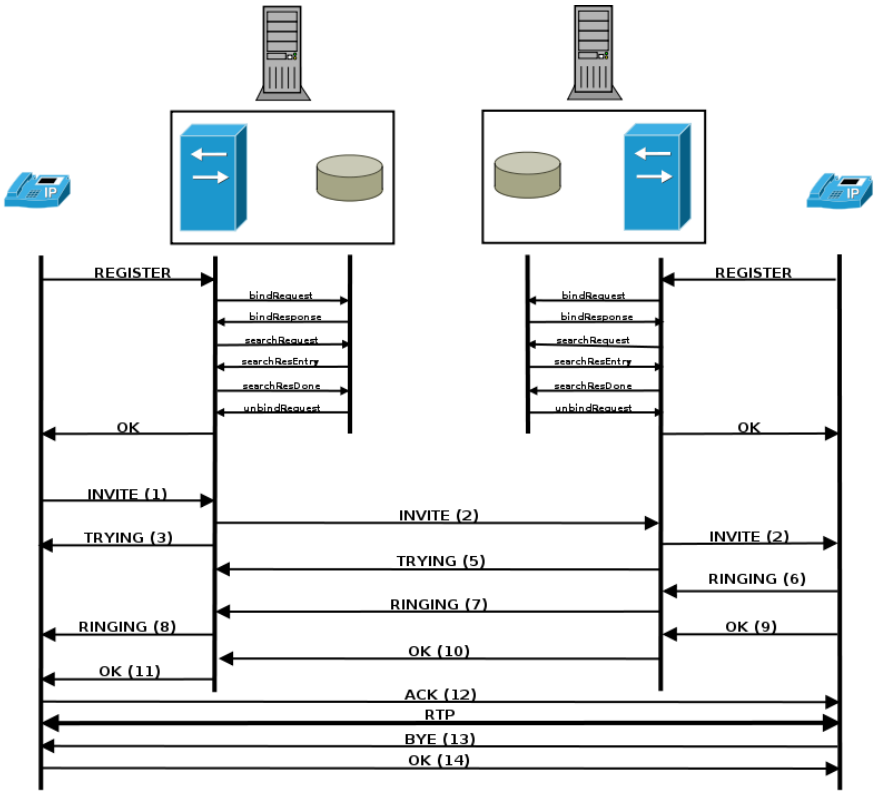

Figura 3.2: Llamada interna entre servidores VoIP diferentes

Conexiones con más de dos servidores no serán necesarias ya que estamos en al misma red IP y todos los servidores pueden acceder contra los otros.

Las **llamadas externas** en este proyecto se realizan igual que una llamada local, pero el destino no será un teléfono de la red, sinó que será el proveedor de VoIP contratado que de forma transparente conectará con el teléfono marcado.

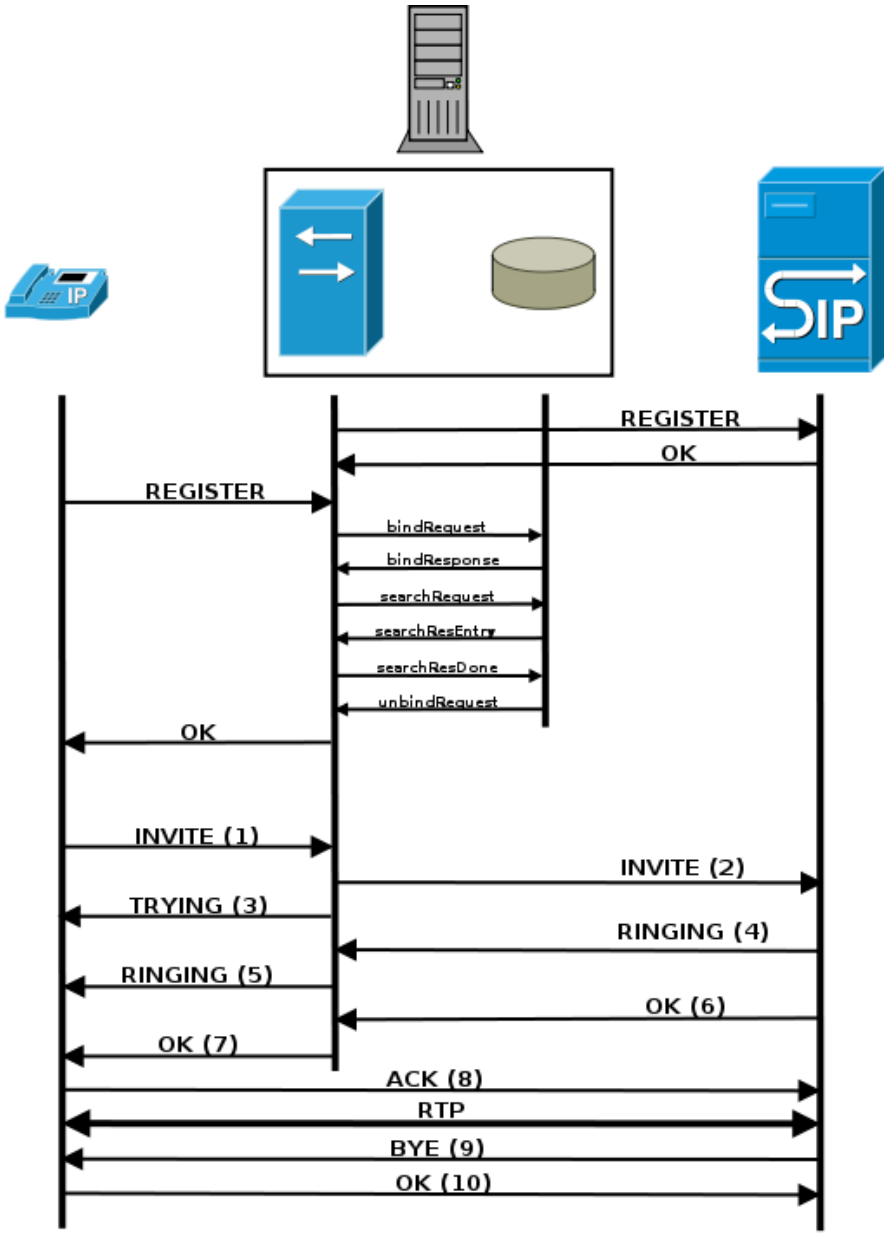

Figura 3.3: Llamada externa

3.4. Web

3.4.1. Servidor de directorios

Anteriormente no existía este servicio por lo que tendremos que hacer la interfaz Web para añadirlo y poderlo asignar a un servidor de los creados en la Web.

Luego habrá que añadir este servidor a la zona geográfica que queremos que se pueda utilizar, por ejemplo un pueblo, una comarca, etc.

No hace falta modificar estructuras de datos porque las tablas ya tienen un campo de tipo de servidor, por lo que únicamente habría que añadirlo y modificar la interfaz.

3.4.2. VoIP

El registro en el sistema para poder realizar llamadas locales se hará desde las preferencias de cada usuario. Los usuarios estarán añadidos al nodo desde el cual se conectarán. En estas preferencias habrá un desplegable que al presionarlo permitirá configurar en que centralita se quiere que el teléfono registre, el número de usuario en esa centralita, la Web asignará el número de usuario menor posible, pero puede ser cambiado sino está ya ocupado, y por último la contraseña.

En las llamadas externas una vez presionado el desplegable de registro de usuario en el sistema, habrá un segundo desplegable con los datos de acceso al proveedor en los que se debe especificar el servidor, el identificador de llamada entrante, el usuario y la contraseña. Estos valores serán cifrados en la base de datos.

Todos estas estructuras de datos irán en una nueva tabla dedicada para VoIP.

Más información de la interfaz Web en el Apéndice D.

3.5. Seguridad

3.5.1. Flujo de datos en las llamadas

En las redes inalámbricas los paquetes viajan en forma de ondas de radio. Nuestra red es abierta y no utiliza cifrado, por tanto alguien con acceso de administrador en cualquiera de los routers por los que pasa la llamada podría interceptarla.

Por compatibilidad con los teléfonos debido a que SRTP y ZRTP son proceloso muy nuevos, se ha decido optar por utilizar RTP que es soportado por todos los teléfonos y dejar la seguridad en un segundo plano.

La solución ideal sería usar los tres protocolos RTP existentes de mayor seguridad a menor ZRTP, SRTP y RTP dependiendo de los protocolos que soporten los terminales. Pero esto no es posible de realizar en FreeSWITCH, por tanto la elección tomada es utilizar sólo RTP.

3.5.2. Autentificación

Los desarrolladores de FreeSWITCH consideran que las contraseñas de la configuración no necesitan estar cifradas ya que la protección del sistema se tiene que hacer desde fuera para que no entren. Esto no resulta un problema en los sistemas normales donde se instala este software ya que estos archivos o están en la maquina local o están en un servidor de la red interna que son seguros.

En el caso de este proyecto tenemos los datos de usuario y contraseña que son enviados por LDAP al hacer la autentificación sin usar ningún cifrado tipo TLS por lo que los datos podrían ser interceptados y conocer el usuario, pero la contraseña es enviada en forma de hash cifrado por lo que no se podría recuperar y es la manera seleccionada para dar seguridad al sistema.

Para los datos del proveedor VoIP FreeSWITCH también necesita tener estos datos sin cifrar esto supone que necesitamos tener confianza con el administrador de este servidor o ser nosotros mismos.

Los datos guardados en la Web están cifrados y al descargarlos también, será

tarea del paquete guifi-voip descifrar los datos y guardar la configuración en plano.

mod_xml_ldap

Existen dos versiones de este modulo la versión 1 "mod_xml_ldap.c"que proporciona campos de LDAP obligatorios a utilizar y que funciona perfectamente y la versión 2 "mod_xml_ldap2.c"que permite utilizar cualquier tipo de campo de LDAP ya que en el archivo de configuración xml permite traducción de campos entre las definidas en LDAP y las que necesita obligatoriamente FreeSWITCH.

El único problema de la versión 2 es que no se ha actualizado su código desde 2008 y la capa superior xml de abstracción ha cambiado, para solucionar este problema se ha realizado un parche que adapta el xml generado al nuevo formato y se han añadido campos nuevos que no estaban contemplados.

NAT

NAT es un sistema de traducción de direcciones de red que se utiliza para asignar una red a una única IP permitiendo traducir las IPs y puertos de un rango de red privada a pública.

Es decir, la máquina interna y la máquina externa a nuestra red no se dan cuenta de estos pasos de traducción que se han producido en las cabeceras de los paquetes IP.

En VoIP hay que tener en cuenta la señalización SIP y el flujo de datos RTP.

En la señalización SIP el inconveniente está en que la IP y el puerto obtenido de la cabecera SIP corresponde con la IP y el puerto que el terminal tiene asignado en la red interna. De esta manera, cuando un dispositivo que se encuentra en la red externa desea responder a esta petición sólo dispone de la IP y puerto de la red interna por lo que no sabe donde debe enviar la respuesta. No existe una ruta para esa respuesta.

En RTP la negociación se realiza mediante el protocolo de descripción de sesión SDP dentro de un intercambio petición y respuesta SIP. Durante este intercambio, cada cliente situado en su propia red especifica la IP y puerto para la

recepción del flujo de información durante la sesión. El problema se debe a que la IP indicada por cada cliente se corresponde con la dirección de la red privada a la que pertenece, la cual no es accesible desde fuera al otro lado del NAT. Es decir, la dificultad aparece cuando el tráfico RTP intenta llegar a la red destino.

Para solventar los problemas expuestos en este proyecto con NAT se ha decidido, ya que disponemos de IPs libres, que es mejor opción utilizar IPs públicas únicamente en todos los dispositivos, porque dependiendo del NAT del router podemos tener problemas, además de estar limitados si queremos utilizar más de un teléfono en nuestra red. FreeSWITCH al utilizar bypass media y permitir llamadas punto a punto nos obliga a utilizar IPs del mismo rango de direccionamiento, en nuestro caso dentro de guifi.net será 10.0.0.0/8.

3.6. Sintonización de la red

3.6.1. Diseño de la capacidad del enlace físico

A priori no conocemos el ancho de banda de los enlaces sabemos que reales con 802.11b tenemos 7Mbps máximo, con 802.11a tenemos 25Mbps máximo y con los nuevos enlaces 802.11n 100Mbps, pero muy pocos tramos de la red tienen este ancho de banda.

De todas formas independientemente de esto utilizaremos colas para priorizar tráfico como veremos a continuación, aunque hay que tener claro que aunque prioricemos tráfico tenemos que tener un mínimo de flujo de datos necesario para llamar.

3.6.2. QoS

La medida obligatoria en una red inalámbrica si se quiere utilizar VoIP es utilizar **colas de priorización de tráfico** para dar prioridad al tráfico de voz sobre los de datos.

Para garantizar la calidad en las llamadas internas es necesario que el mayor número posible de routers de la red tengan colas de prioridad, sino habrá cortes ya

que por defecto la red no está configurada para priorizar tráfico RTP lo que puede causar que hayan ráfagas de paquetes que consuman el máximo posible del ancho de banda y haya cortes en la voz.

Además hará falta aplicarlo también para las llamadas externas en el servidor y en el router que tiene la conexión a Internet.

Más información de la implementación en el Apéndice B.

FreeSWITCH es un B2BUA diseñado para interconecte redes de forma que haya abstracción de la topología de red, para conseguirlo hace de gateway entre las redes y el tráfico por defecto no es punto a punto entre extremos, sino que obligatoriamente se transmite a través del servidor o servidores. Esto produce que el camino entre los dos teléfonos sea más largo que si se realizase punto a punto sin servidores en medio, proporcionando menor latencia.

Las **llamadas punto a punto** utilizando Bypass Media [FS-BM] permiten que el tráfico RTP entre los dos teléfonos se realice punto a punto sin tener que pasar por el servidor, con esto conseguimos reducir la latencia ya que los paquetes en la red siguen el camino más corto entre routers. La configuración se puede ver en el Apéndice C.

3.6.3. CAC

Existen diversos mecanismos para limitar el número de llamadas pero todos se basan en una red estable, normalmente con un gran ancho de banda, que se quiere optimizar para obtener más rendimiento.

El problema que tenemos es que nuestra red no es estable por lo que las condiciones en tiempo real son aleatorias y no se puede hacer planes de prevención contra algo que previamente desconoces.

Por tanto, a nivel de CAC sólo se limitará el número de llamadas simultaneas para que no se produzcan más de las soportadas y evitar el jitter en todas las llamadas.

Esta limitación ya que depende del ancho de banda de cada enlace y de la conexión a Internet no podemos automatizar su configuración porque no sabemos su rendimiento desde la Web y tendrá que ser configurado de forma manual por el

administrador.

En el Apéndice C.4 se describe el procedimiento de configuración de la limitación.

3.7. Numeración

La primera idea fue crear una numeración de longitud variable sin prefijo estilo a la numeración actual de los teléfonos móviles pero de longitud infinita "(\d)". Esta flexibilidad que al principio parece ideal, tiene problemas, ya que requiere un sistema centralizado que conozca todos los números o un sistema distribuido que cada vez que se registre un terminal avise a todas las centralitas.

La primera solución considerando el hardware que disponemos tanto de red como de servidores se ve claramente que es un cuello de botella y que el sistema no escala ni sería fiable. Necesitaríamos buenos servidores para manejar todas las llamadas, una excelente red troncal en el servidor , además de que si la red falla en algún tramo, esa subred no podría realizar ninguna llamada.

La segunda solución fue probada utilizando la API de FreeSWITCH llamada Event Socket Layer que permite realizar acciones en lenguaje de script al recibir eventos. Tiene dos problemas, la fiabilidad del sistema depende de otra capa a nivel de script que gestionaría todos los registros y la segunda más importante es que por cada registro y desregistro tendríamos que avisar a todas las centralitas, por lo que produciríamos muchísimo tráfico.

La idea final propuesta es utilizar numeración variable, pero especificando también de forma variable el número de centralita "(\d)*(\d)"donde el primer número sería la centralita y el segundo el usuario. Ejemplo: "2*3"

Si una llamada es local en la misma centralita se puede utilizar la extensión anterior o únicamente el usuario "(\d)".

Para las llamadas externas se utilizará "*(\d)"donde el número de teléfono será un número fijo o móvil de la RTC. Ejemplo: "*936908689"

3.8. Implementación

Para materializar el proyecto se han programado 3 parches y 2 paquetes.

- **freeswitch-1.0.6-mod_xml_ldapv2.patch**: Parche que contienes las modificaciones necesarias del modulo de FreeSWITCH mod_xml_ldap en su versión 2 actualizando la exportación al nuevo formato xml de FreeSWITCH.
- **drupal-guifi6x-ds.patch**: Parche con las modificaciones para añadir al modulo de drupal de guifi.net funcionalidades que proporcionan poder configurar y almacenar el servicio de directorios.
- **drupal-guifi6x-voip.patch**: Parche con las modificaciones del drupal de la página Web de guifi.net que añaden las funcionalidades para guardar los datos de las centralitas y la interfaz Web para configurar las propiedades de los usuarios.
- **guifi-ds**: Es el paquete que descarga a partir de la Web de guifi.net cada 24 horas la configuración del fichero slap.conf de LDAP y la de los ficheros ldif que contienen los objetos de LDAP de manera automática.
- **guifi-voip**: Es el paquete que descarga la configuración VoIP de las llamadas externas cada 24 horas ya que los datos, no pueden ser añadidos en mod_xml_ldap porque son datos de registro de usuarios y la configuración de proveedores se configura en otros archivos. Hay que considerar que FreeSWITCH necesita los datos sin cifrar tanto usuario como contraseña del proveedor, por tanto, la configuración de la Web esta cifrada, pero una vez descargada se descifra y se añade en claro a la configuración.

Capítulo 4

Pruebas

En este capítulo se verificará el correcto funcionamiento del sistema analizando los flujo de sincronización y analizando la latencia de la red.

4.1. Topología de la red de pruebas

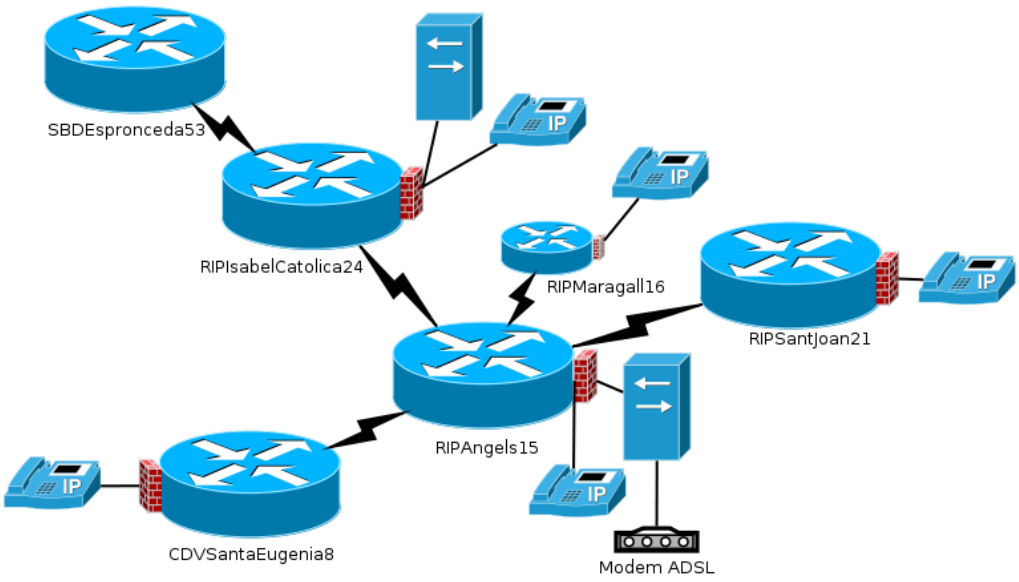

Figura 4.1: Topología de la red de pruebas

4.2. Funcionalidad

4.2.1. Instalación

La primera prueba que debemos realizar es comprobar la correcta y fácil instalación de la aplicación. Añadiendo el repositorio oficial de paquetes de guifi.net y ejecutando el comando de instalación, la aplicación descarga la información necesaria de la Web y configura tanto el servidor LDAP (sino está instalado) como el servidor FreeSWITCH. Más información del proceso en el Apéndice A.

4.2.2. LDAP

Antes de realizar llamadas podemos comprobar si el servidor LDAP funciona buscando un usuario, por ejemplo la extensión '1*1'.

```
$ ldapsearch -x -h 10.139.50.2 -b dc=ripollet,guifi,dc=net -LLL SIPIdentityUserName=1*1
dn: uid=velazquez.tomas,ou=People,dc=ripollet,dc=guifi,dc=net
uid: velazquez.tomas
cn: ,
objectClass: account
objectClass: SIPIdentity
objectClass: posixAccount
objectClass: top
userPassword:: MTAwMg==
uidNumber: 99
gidNumber: 99
homeDirectory: /home/nobody
SIPIdentityPassword: dda3f702b9aaed43fe3c4e4beb96e0b3
SIPIdentityUserName: 1*1
```

Podemos ver que retorna los datos, por tanto el usuario existe y entre los datos devueltos está el SIPIdentityPassword que es el hash de la contraseña de autentificación, así que tenemos todos los datos para validar este usuario.

4.2.3. Registro de usuarios

El proceso de registro de usuarios necesita como se ha visto anteriormente los datos del servidor LDAP.

En la siguiente figura obtenida a partir de Wireshark se puede observar el proceso de petición al servidor LDAP y la autorización del usuario

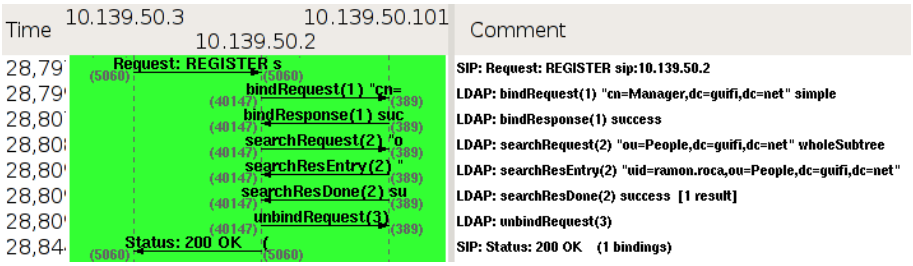

Figura 4.2: Registro de usuario

Tanto para crear una llamada interna como una externa se necesita registrar en el sistema, en los ejemplos siguientes este proceso de registro ya está hecho y únicamente se analiza la señalización SIP.

4.2.4. Llamada interna

La siguiente prueba consiste en analizar el tráfico de una llamada interna que se ha hecho desde el nodo *RIPIsabelcatolica24* [1] al nodo *RIPAngels15* [2] utilizando Wireshark.

[1] http://guifi.net/node/18938
[2] http://guifi.net/node/14206

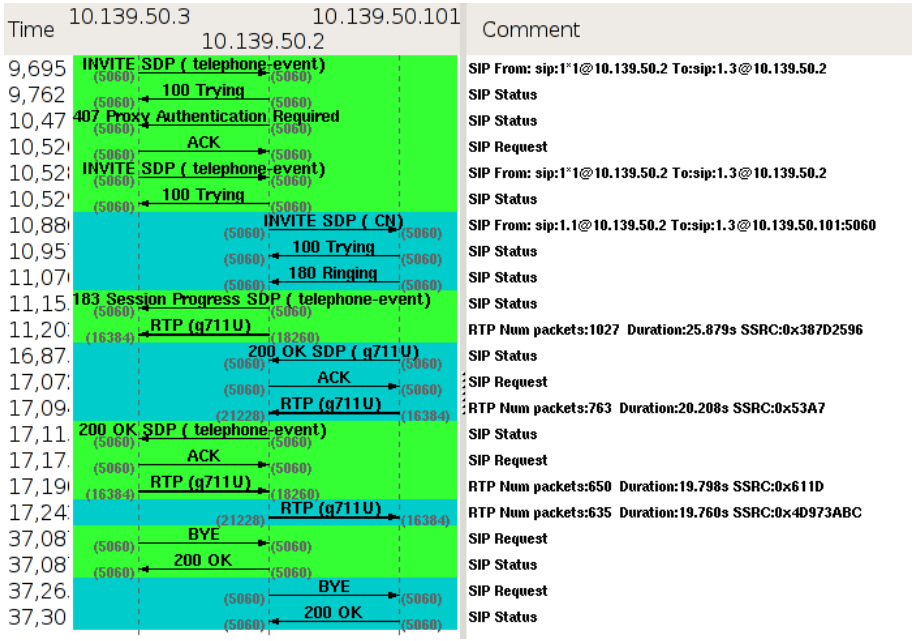

Figura 4.3: Llamada interna

Podemos ver en la figura que el usuario 1*1 quiere llamar a 1*3 y se lo comunica al servidor. Se activa el tono de espera de llamada y se hace la petición al teléfono destino. Este emite los pitidos de una llamada. El usuario descuelga y se negocia el códec a utilizar que sea compatible con los dos terminales. Se activa el flujo de datos de voz en ambas direcciones. La persona que realiza la llamada cuelga y la persona que la ha recibido también cuelga.

4.2.5. Llamada externa

Esta prueba es como la anterior pero en vez de ser una llamada interna se realiza de forma externa utilizando un proveedor de VoIP.

La siguiente figura generada por Wireshark muestra el tráfico generado.

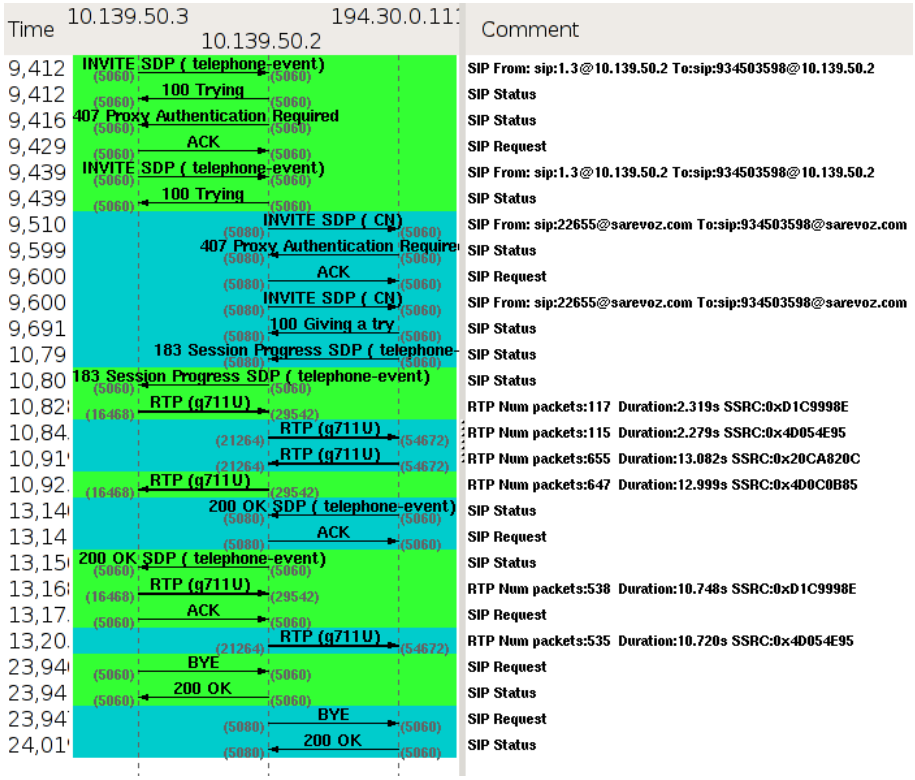

Figura 4.4: Llamada externa

Podemos ver que es el mismo proceso de sincronización que realiza las llamadas internas, la única diferencia es que el destino no es un teléfono, sino un servidor VoIP de una empresa. Hay que considerar que la llamada se realiza previo registro de la cuenta de usuario en la centralita del proveedor.

4.3. Rendimiento

Para buscar el rendimiento máximo tenemos que buscar el elemento más lento del sistema que será el limitador de nuestros resultados.

En este caso el límite de las llamadas internas es causado por el enlace que tenga menos ancho de banda.

En el caso de llamadas externas el límite viene dado por la subida del ADSL que normalmente está entorno a 1M.

Hay que considerar que el ancho de banda en un ADSL no es constante ya que interviene el ruido de la señal de cable y otros factores que limitan la velocidad.

4.3.1. Tráfico utilizando diferentes códecs

Sabemos la tasa de bits de los códecs, pero para realizar pruebas no es un dato significativo ya que hay paquetes que utilizan más bytes en su cabecera que en los datos multimedia que contienen. Por tanto, sumaremos la cabecera por cada paquete y luego calcularemos la tasa de bits total que utilizan los paquetes por segundo en cada códec.

$Cabeceras = Ethernet + IP + UDP + RTP = 14 + 20 + 8 + 13 = 54bytes$

$PaqueteRTP = 54 + bitratecodec$

A partir de analizar el tráfico con Wireshark de las diferentes llamadas utilizando diferentes códecs se puede observar el tamaño de los paquetes y calcular el ancho de banda necesario para realizar una llamada utilizando estas fórmulas:

$Tasadebitsporsegundo = paqueteRTP * 1000/frecuenciademuestreo * 8$

Ejemplo G.711: $214 * 1000/20 * 8 = 85600 = 85,6kbps$

En la tabla siguiente se puede ver los resultados con diferentes códecs.

Códec	**Tamaño paquete (bytes)**	**Tasa bits total (kbps)**	**Datos útiles (%)**
G.711	214	85,6	74.77
G.726	134	53,6	59.7
G.728	114	30,4	52.63
GSM	87	34,8	37.356
Speex	100	40	80
iLBC	92	36.8	36.14

Cuadro 4.1: Tamaño y tasa de bits real de los códecs

Al comparar esta tabla con la de la sección de análisis en la que únicamente se tenía en cuenta el consumo del códec se puede observar que las cabeceras de todos los protocolos y el número de paquetes enviados por segundo influyen mucho en el ancho de banda real utilizado.

Existe el Compressed Real-time Transport Protocol (CRTP) [CRTP] como una posible solución para comprimir las cabeceras y así utilizar mejor el ancho de banda.

Se han encontrado herramientas para verificar el número de señalizaciones que soporta un servidor VoIP como SIPp y sipsak, pero este factor no es de utilidad en nuestra red porque el límite de conexiones estará establecido por el ancho de banda simultaneo consumido por el flujo de datos.

Las pruebas siguientes han sido realizadas generando nuevas conexiones UDP limitadas a 86kbps cada una y añadiendo nuevas hasta que la latencia aumentaba hasta 150 ms. Estas pruebas se han realizado de forma manual porque no se ha encontrado ningún software capaz de generar una llamada completa, desde sincronización al envío del flujo de datos. Considerando los códecs de la tabla anterior y conociendo el códec más soportado por los terminales se ha optado por utilizar G.711 en las pruebas.

Actualmente existen dos tipos de ancho de banda reales en la red:

- Nodo troncal normalmente 20mbps en half duplex son 10mbps en full duplex (simétricos). A partir de las pruebas obtenemos 97 llamadas simultaneas sin cortes de media.

- Nodo cliente normalmente 1Mbps en half duplex son 0,5Mbps en full duplex (simétricos). A partir de las pruebas realizadas obtenemos una media 14 simultaneas sin cortes.

Estos resultados sorprendentes en el nodo cliente son debidos a que en ese momento el ancho del enlace era alrededor de los 2Mbps simétricos reales debido a que había poca gente utilizando la red.

4.4. QoS

4.4.1. Prioridad del tráfico en una llamada

Para realizar esta prueba descargaremos un fichero por FTP utilizando el 100 % del ancho de banda y luego haremos una llamada.

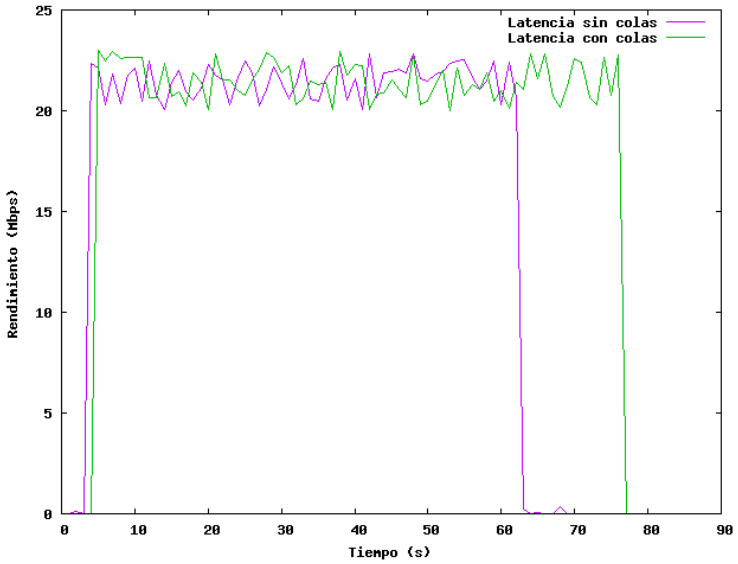

Figura 4.5: Rendimiento con y sin colas de prioridad

Vemos como al máximo de velocidad depende de la señal del enlace por eso vemos que fluctúa siempre superando los 20M.

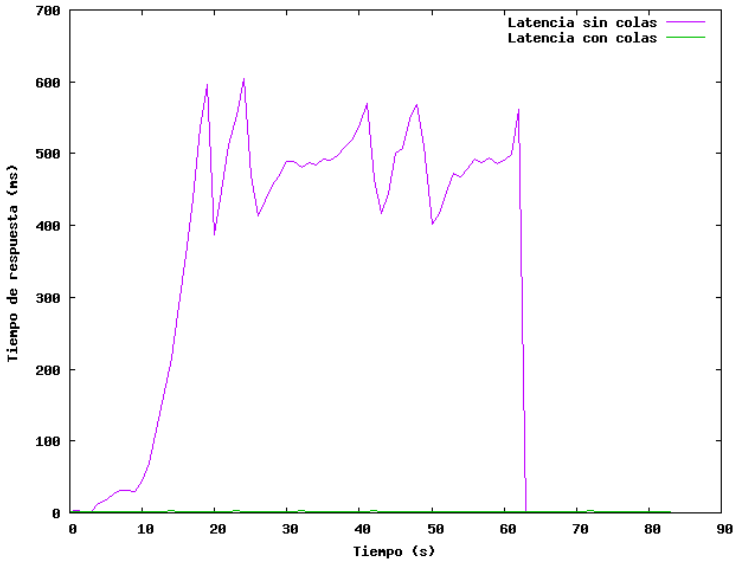

Figura 4.6: Latencia con y sin colas de prioridad

Utilizando al máximo un enlace nos provoca que sea impracticable realizar

una llamada, ya que al pasar de 150 ms de latencia tenemos jitter en la llamada y al pasar de 250 ms no podemos entender nada de la llamada, por tanto la utilización de colas de prioridad es obligatorio para garantizar la funcionalidad del sistema.

4.5. Latencias

4.5.1. Distancia normal

La siguiente prueba es un ejemplo de la latencia producida por 4 saltos que sería la distancia normal entre un nodo cliente y un supernodo donde estaría la centralita y la conexión a Internet, a parte en un acceso a un proveedor de VoIP se tendría que sumar el tiempo de latencia entre el ADSL y el proveedor que este dependería del ADSL, pero suele estar alrededor de los 50 ms.

Desde el nodo *RIPAngels15* [3] de Ripollet al nodo *SBDAlfonsXIII* [4]

```
$ traceroute -q 1 10.138.105.161
traceroute to 10.138.105.161 (10.138.105.161), 30 hops max, 40 byte packets
 1  192.168.1.2 (192.168.1.2)  0.697 ms
 2  RIPAngels15ST--9854.ip.guifi.net (10.139.50.97)  8.594 ms
 3  RIPIsabelcatolica24ST--13651.ip.guifi.net (172.25.33.190)  8.719 ms
 4  SBDpsgEsproncedaST1--15400.ip.guifi.net (172.25.34.189)  8.891 ms
 5  SBDAlfonsXIII-Forniolcat--16047.ip.guifi.net (10.138.105.161)  9.223 ms
```

[3] http://guifi.net/node/14206
[4] http://guifi.net/node/12981

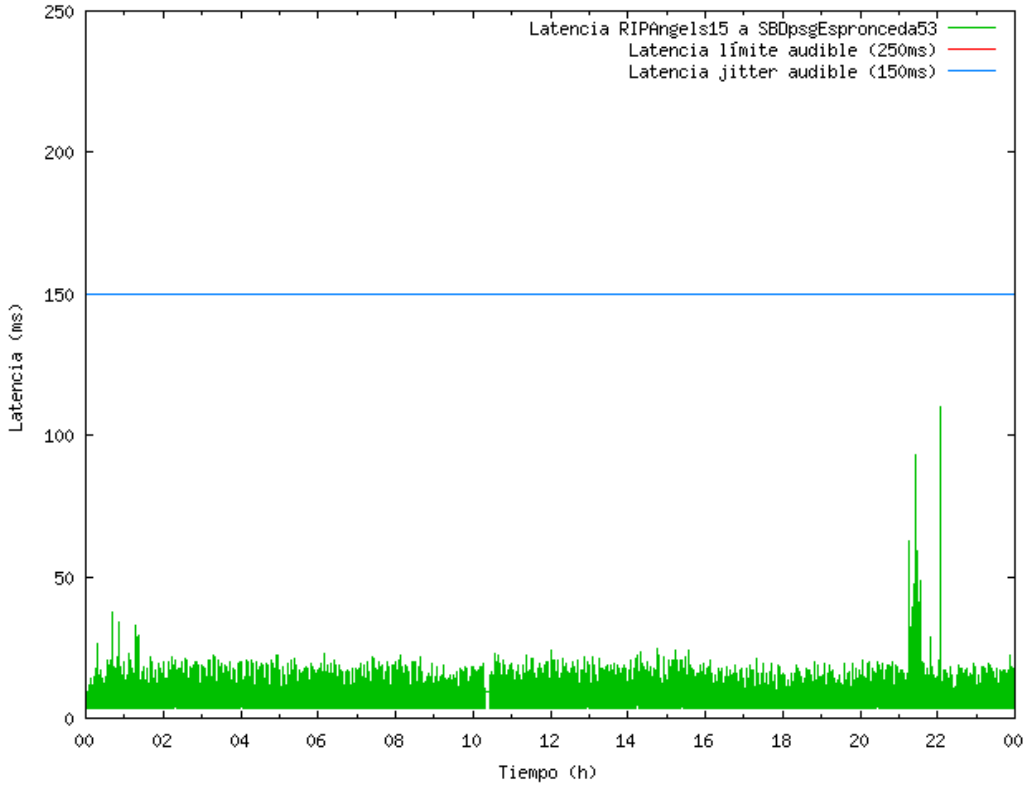

Figura 4.7: Latencia en distancia normal

Podemos observar que entre las 10:18 y las 10:25 ha ocurrido un corte en la red y no tenemos respuesta a las peticiones mandadas. También se puede observar que entre las 21 y las 22 hay un aumento de tráfico puntual que produce por consecuencia un aumento de la latencia ya que en esta prueba no hay colas de prioridad en todos los tramos.

La media de la latencia en 24 horas es de 6,7 ms.

4.5.2. Distancia larga

Desde el nodo *RIPAngels15* [5] de Ripollet al nodo *Roses-Badia* [6] de Roses donde se puede ver que entre CdmMPonsDiposits-ST1 y MARCanPareraST1 hay mucha latencia a causa de que el enlace tiene mala señal y circula mucho tráfico.

[5] http://guifi.net/node/14206
[6] http://guifi.net/node/23467

```
$ traceroute -q 1 10.139.158.1
traceroute to 10.139.158.1 (10.139.158.1), 30 hops max, 40 byte packets
 1  192.168.1.2 (192.168.1.2)  0.700 ms
 2  RIPAngels15ST--9854.ip.guifi.net (10.139.50.97)  8.624 ms
 3  RIPIsabelcatolica24ST--13651.ip.guifi.net (172.25.33.190)  8.793 ms
 4  SBDpsgEsproncedaST1--15400.ip.guifi.net (172.25.34.189)  8.971 ms
 5  SBDAlfonsXIII-Forniolcat--16047.ip.guifi.net (172.25.34.10)  9.152 ms
 6  SBDAlfonsXIII--12281.ip.guifi.net (172.25.32.130)  9.693 ms
 7  SBDCampanarSSRouter1--13838.ip.guifi.net (172.25.32.161)  10.064 ms
 8  SBDFira--9432.ip.guifi.net (172.25.32.106)  12.023 ms
 9  CdmMPonsDipositsRd2--15172.ip.guifi.net (172.25.34.170)  12.832 ms
10  CdmMPonsDiposits-ST1--5049.ip.guifi.net (172.25.34.166)  14.329 ms
11  MARCanPareraST1--4232.ip.guifi.net (172.25.5.169)  15.616 ms
12  CNTTVCentellesST2--4231.ip.guifi.net (172.25.5.54)  16.499 ms
13  CNTTVCentellesST1--4013.ip.guifi.net (172.25.5.90)  18.552 ms
14  GurbPavelloST1--1916.ip.guifi.net (172.25.0.5)  20.597 ms
15  TorelloDipSudST2--2820.ip.guifi.net (172.25.3.218)  27.772 ms
16  TorelloDipSudBellmunt--4356.ip.guifi.net (172.25.0.106)  29.081 ms
17  BellmuntSud--4327.ip.guifi.net (172.25.0.17)  19.618 ms
18  VidraPuigCubellRd1--19098.ip.guifi.net (172.25.224.42)  17.131 ms
19  knoppixRadio3--9201.ip.guifi.net (172.25.227.26)  22.416 ms
20  MontolivetRadio1--9200.ip.guifi.net (172.25.32.54)  91.985 ms
21  AlbMMontSTSud--5882.ip.guifi.net (172.25.9.54)  95.071 ms
22  AlbMMontSTNord--10360.ip.guifi.net (172.25.2.117)  96.199 ms
23  LldPjl25Rd1--19709.ip.guifi.net (172.25.35.105)  97.409 ms
24  NVTCampanarRd1--20205.ip.guifi.net (172.25.32.169)  80.422 ms
25  GrrgllDipRd2--22380.ip.guifi.net (172.25.39.22)  80.724 ms
26  CbnsCanalRd1--21753.ip.guifi.net (172.25.35.253)  81.637 ms
27  vajolNegrinRd2--16783.ip.guifi.net (172.25.38.186)  82.030 ms
28  Roses-Badia--18382.ip.guifi.net (10.139.158.1)  82.854 ms
```

Como se puede ver en el traceroute se han hecho en horario de poco tráfico para obtener buenos resultados.

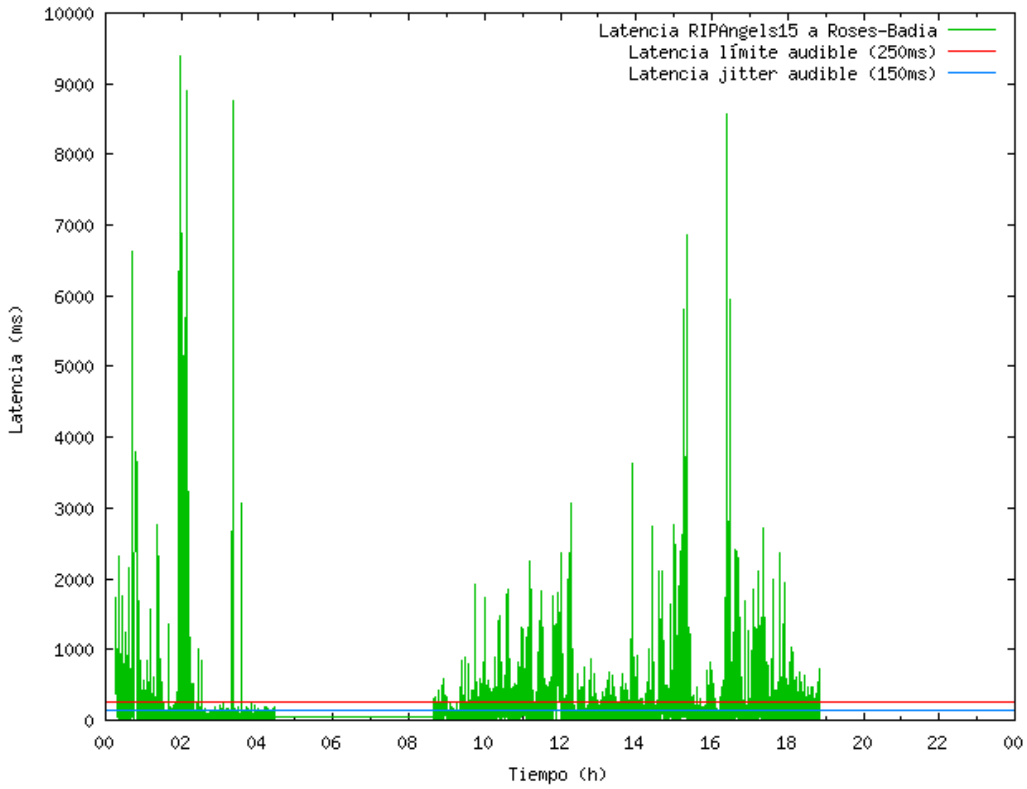

Figura 4.8: Latencia en distancia muy larga

Podemos observar de la monitorización de la latencia que no es viable hacer una llamada al otro extremo de la red generalmente, aunque en horarios de poco tráfico como en las noches de 4:30 a 8:30 si que se podría.

Para solucionar esto los nodos intermedios tendrían que priorizar el tráfico VoIP sobre los otros tipos. Hay que resaltar que entre las 0:00 y las 0:15 no había conexión, ni de 19:00 a 0:00 ya que algún tramo de la red estaría cortado, así que aunque hubiese colas de prioridad en todos los routers en ese horario no se podría realizar llamadas.

La media de la latencia en 24 horas es de 136.5 ms en el periodo que existía conexión.

4.6. Seguridad

Además de poder analizar la señalización con Wireshark, podemos analizar el protocolo RTP y la codificación de la voz, consiguiendo reproducirla ya que RTP no utiliza cifrado.

A continuación se puede ver un ejemplo de voz capturada, para conseguir esto se necesitaría poder capturar tráfico en alguno de los routers intermedios entre los extremos de la llamada.

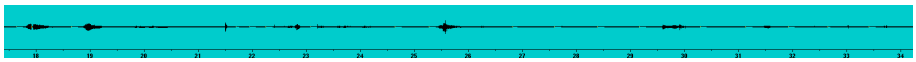

Figura 4.9: Voz del llamante

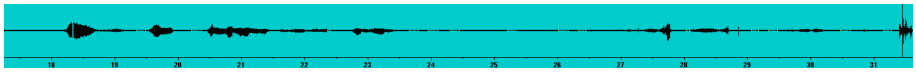

Figura 4.10: Voz del llamado

Capítulo 5

Conclusiones

Para finalizar la memoria se hará un resumen del trabajo realizado, una valoración personal de las sensaciones obtenidas y las futuras líneas de evolución del proyecto hacia otros ámbitos.

5.1. Resumen

La primera fase realizada consistió en un análisis tanto de la red como de la VoIP, aplicaciones y protocolos existentes, siempre utilizando software libre y protocolos abiertos. Más tarde, a partir de los requerimientos impuestos de prestaciones y de las limitaciones de la red se buscó la solución más adecuada para dar una solución completa.

Como resultado se ha creado unos paquetes software para GNU/Linux y una interfaz Web de administración de usuarios que permiten configurar y comunicar de forma automática todos los servidores VoIP de la red guifi.net utilizando un sistema de numeración único. Además permite realizar llamadas externas a través de proveedores VoIP utilizando líneas con acceso a Internet.

En las pruebas realizadas se ha podido ver que es fundamental para el tráfico VoIP, sobretodo en redes inalámbricas donde el entorno es muy hostil, utilizar colas de prioridad en los routers para garantizar calidad en las llamadas.

Al no tener ningún tipo de obligación ni de software ni de protocolos a utilizar

en el proyecto, se ha tenido que buscar todas las alternativas y decidir cual era la más conveniente para el proyecto, lo que ha hecho que la fase de análisis fuera considerablemente compleja.

En la parte del desarrollo se ha tenido que ver como implementar el sistema integrándolo con el complejo modulo drupal de la Web de guifi.net. Por la parte de FreeSWITCH se ha tenido que modificar el modulo de LDAP para actualizarlo.

En conclusión, se han cumplido las expectativas propuestas y todos los objetivos especificados en el informe previo, utilizando mucho software y protocolos realizados por terceros de muy alta calidad y se ha creado un software para interconectar y configurar este software de terceros, automatizándolo a partir de la Web de guifi.net y optimizando todos los recursos disponibles lo máximo que se ha podido.

5.2. Planificación final

Si comparamos la planificación inicial con la final vemos que el análisis, desarrollo y las pruebas han seguido lo planificado, pero en la memoria se han utilizado el mismo número de horas planificadas, pero se ha trabajado menos por día de lo que se tenía previsto.

	Nombre	Duración	Inicio	Terminado
1	⊟**Servidores VoIP guifi.net**	132 days	**1/03/10 8:00**	**31/08/10 17:00**
2	Análisis	45 days	1/03/10 8:00	30/04/10 17:00
3	Desarrollo	43 days	1/05/10 8:00	30/06/10 17:00
4	Pruebas	44 days	1/07/10 8:00	31/08/10 17:00
5	Memoria	45 days	15/06/10 8:00	16/08/10 17:00

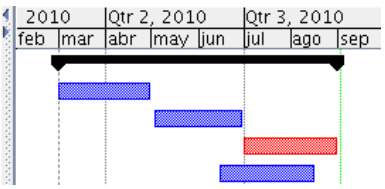

Figura 5.1: Planificación final

5.3. Valoración personal

En la realización del proyecto he aprendido mucho ya que partía únicamente con los conceptos del funcionamiento de la red guifi.net, pero no sabía nada de VoIP, ha sido todo un reto para mi.

Es un orgullo hacer posible que zonas donde llega guifi.net, pero que no tienen tendido telefónico ni cobertura de telefonía móvil, puedan llamar a través de un proveedor VoIP utilizando mi proyecto como medio de interconexión.

Además creo que es muy bueno para fomentar la competencia dar la opción de realizar llamadas a muy bajo coste a teléfonos fijos y sin que haya coste de establecimiento de llamadas, sin la necesidad de que el usuario final disponga de una línea telefónica y utilizando un proveedor VoIP.

5.4. Futuras líneas

A continuación se presentan posibles extensiones de este proyecto con otras tecnologías y protocolos.

La conexión de nuestro sistema con la red de Radio over IP que tiene un grupo de radioaficionados Españoles que conectan sus repetidores por Internet utilizando Asterisk, con lo que consiguen hablar con otros radioaficionados a muchos kilómetros utilizando Internet como conexión de las antenas de estos repetidores.

Es posible conectar nuestro sistema a otras comunidades VoIP del mundo aprovechando las tarifas planas de Internet y el hecho de que las llamadas locales en todos los países son gratuitas.

Dentro de unos años con la aparición normalizada de ZRTP en los teléfonos y la bajada de precios de estos se podría forzar todas las llamadas a utilizar este protocolo para tener seguridad en las llamadas.

Con la misma idea anterior normalizar los códecs de alta compresión que ofrecen los teléfonos y poder realizar llamadas con menor tráfico.

Añadir colas de prioridad por defecto en el 'unsolclic' que es la aplicación que genera a partir de la Web de guifi.net la configuración de los routers, para asegurarnos que toda la red dispone de cierta calidad para el servicio de voz. Aunque esto puede ser difícil de que ocurra, al menos por defecto, porque el objetivo actual de 'unsolclic' es proporcionar únicamente la configuración básica para que funcione la interconexión de los nodos.

Sería un sueño conectar nuestro sistema al punto neutro de Cataluña (CAT-

NIX) donde ya tenemos actualmente routers y aprovechar este gran ancho de banda para realizar las llamadas hacia proveedores VoIP.

También se podría añadir tramos de fibra en la red de guifi.net para bajar la latencia entre enlaces que será necesario para realizar llamadas de muy larga distancia en un futuro.

Bibliografía

[IP] Internet Protocol. DARPA Internet Program Protocol Specification.
<http://www.ietf.org/rfc/rfc0791.txt>

[UDP] User Datagram Protocol.
<http://www.ietf.org/rfc/rfc0768.txt>

[H323] Usage of H.323 on the Internet.
<http://tools.ietf.org/id/draft-rfced-info-lantz-00.txt>

[SIP] SIP: Session Initiation Protocol.
<http://www.ietf.org/rfc/rfc2543.txt>

[IMF] Internet Message Format.
<http://tools.ietf.org/rfc/rfc2822.txt>

[URI] Uniform Resource Identifiers (URI): Generic Syntax.
<http://www.ietf.org/rfc/rfc2396.txt>

[WP-DAA] Wikipedia. Digest access authentication.
<http://en.wikipedia.org/wiki/Digest_access_authentication>

[SDP] SDP: Session Description Protocol.
<http://www.ietf.org/rfc/rfc2327.txt>

[RTP] RTP: A Transport Protocol for Real-Time Applications.
<http://tools.ietf.org/html/rfc3550>

[SRTP] The Secure Real-time Transport Protocol (SRTP).
<http://www.rfc-editor.org/rfc/rfc3711.txt>

[ZRTP] ZRTP: Media Path Key Agreement for Unicast Secure RTP.
<http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-22>

[JINGLE] Jingle. Official website.
<http://xmpp.org/tech/jingle.shtml>

[IAX2] IAX: Inter-Asterisk eXchange Version 2.
<http://tools.ietf.org/search/rfc5456>

[VI-OSVS] Voip-Info. Open Source VOIP Software.
<http://www.voip-info.org/wiki/view/Open+Source+VOIP+Software>

[WP-CVS] Wikipedia. Comparison of VoIP software.
<http://en.wikipedia.org/wiki/Comparison_of_VoIP_software>

[BAY] GNU Bayonne. Official website.
<http://www.gnutelephony.org/index.php/GNU_Bayonne>

[AST] Asterisk. Asterisk Feature List.
<http://www.asterisk.org/features>

[CW] Callweaver. Official website.
<http://www.callweaver.org>

[VI-CW] Voip-Info. CallWeaver.
<http://www.voip-info.org/wiki/view/CallWeaver>

[YATE] Yet Another Telephony Engine. Official website.
<http://yate.null.ro>

[FS] FreeSWITCH. Specsheet.
<http://wiki.freeswitch.org/wiki/Specsheet>

[SR-H] sip-router. History.
<http://sip-router.org/history>

[FS-BM] FreeSWITCH. Bypass Media.
<http://wiki.freeswitch.org/wiki/Bypass_Media>

[WP-QOS] Wikipedia. Calidad de servicio.
<http://es.wikipedia.org/wiki/Calidad_de_servicio>

[WP-DS] Wikipedia. Directory service.
<http://en.wikipedia.org/wiki/Directory_service>

[LDAP] Lightweight Directory Access Protocol.
<http://www.ietf.org/rfc/rfc1777.txt>

[LDAP-RC] Chapter 7.4 Referrals. 7.4.4 Referral Chaining.
<http://www.zytrax.com/books/ldap/ch7/referrals.html#chaining>

[CRTP] Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering.
<http://www.rfc-editor.org/rfc/rfc3545.txt>

[G114] International Telecommunication Union. ITU-T Recommendation G.114
<http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.114-200305-I!!PDF-E>

Apéndice A

Instalación

El objetivo de este apéndice es resumir el proceso de instalación de la centralita en el servidor local.

Lo primero es crear el servidor de directorios si no está ya dado de alta y luego lo mismo con el servidor de VoIP. Una vez creado cada uno tendrá un identificador en la Web que será necesario para la configuración. Por ejemplo, al crear el servicio se crearía la URL http://guifi.net/node/33184 y el identificador sería 33184.

Ahora desde una terminal ejecutamos los siguientes comandos para añadir el repositorio, actualizar la lista de paquetes local e instalar el paquete guifi-voip.

```
# echo deb http://repo.vic.guifi.net/debian/  ./ > /etc/apt/sources.list.d/guifi.list
# aptitude update
# aptitude install guifi-voip
```

El paquete guifi-voip tiene como dependencia para funcionar los paquetes FreeSWITCH que es el software de VoIP y guifi-ds que es el conjunto de script para configurar el software LDAP y este será dependencia del paquete guifi-ds.

Durante la instalación se pedirán los siguientes datos para que los scripts puedan obtener los datos de configuración de los servicios.

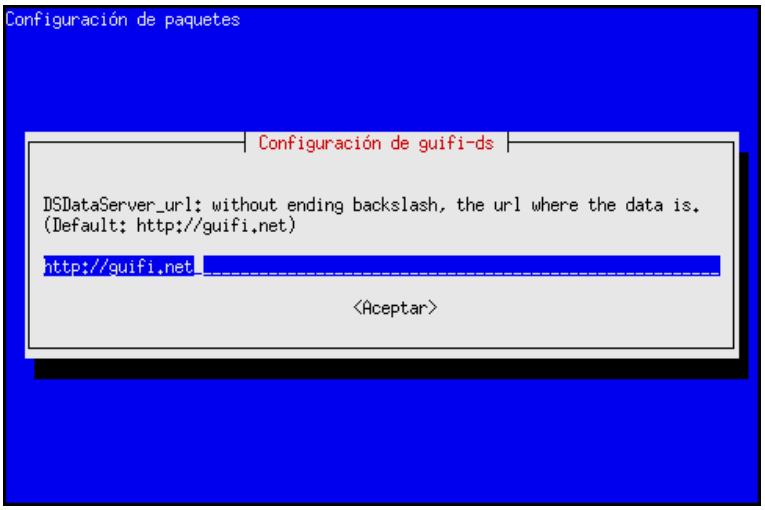

Figura A.1: Configuración guifi-ds, URL del servidor que proporcionará la configuración de LDAP

Figura A.2: Configuración guifi-ds, identificador del servicio

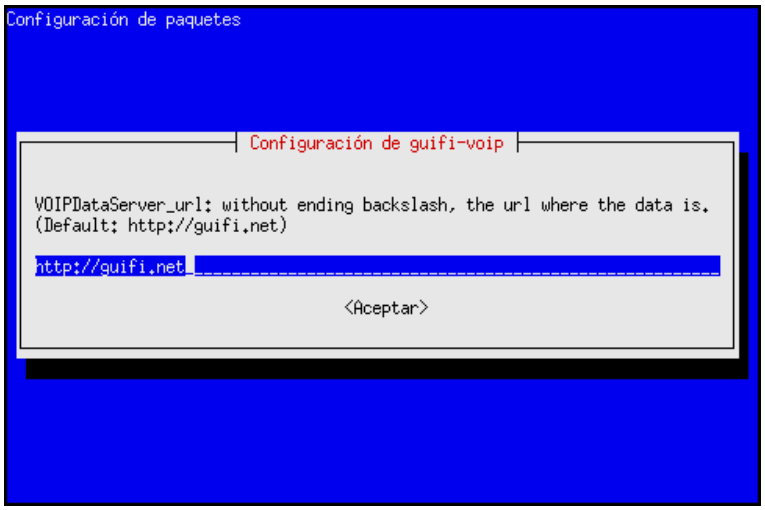

Figura A.3: Configuración guifi-voip, URL del servidor que proporcionará la configuración de FreeSWITCH

Figura A.4: Configuración guifi-voip, identificador del servicio

Una vez descargadas las configuraciones se inician los servicios y termina el proceso de instalación.

Apéndice B

Optimizaciones

En este apéndice veremos algunos métodos de priorización de tráfico y priorización a nivel de proceso del servidor. Ya que estamos en un entorno muy limitado y queremos obtener el mayor rendimiento posible de nuestro sistema en todos los aspectos.

QOS

A causa de las altas latencia y de la no priorización del tráfico por defecto es obligatorio utilizar calidad de servicio en todos los routers posibles para garantizar, dentro de lo posible, la no aparición de jitter y conseguir el máximo de llamadas que soporte el enlace.

En guifi.net generalmente se utilizan dos tipos de routers según su sistema operativo: RouterOS (basado en Linux) y dd-wrt o OpenWRT que son distribuciones de Linux. A continuación se muestra el proceso de creación de colas de priorización en los dos sistemas.

RouterOS

RouterOS soporta ocho prioridades por cada cola que se le añada en el rango de 1 a 8, siendo 8 la prioridad más baja mientras que 1 se suele reservar para

protocolos de enrutamiento.

El siguiente script crea una cola base con tres subcolas con diferentes niveles de prioridad: enrutamiento, tiempo real y el resto de tráfico. Una vez creadas definimos las reglas para marcar estos paquetes: en routing el tráfico BGP y OSPF, en realtime el tráfico ICMP, DNS, SIP y RTP, y en bulk todo el tráfico no marcado previamente.

La interfaz wLan/Lan es un bridge entre la conexión ethernet y la wireless por lo tanto afecta a todo el tráfico que circula por las interfaces de red del sistema.

```
# Make tree queues with different priorities
/ queue tree
add name="guifi-qdisc" parent=wLan/Lan \
    queue=default priority=8 limit-at=0 max-limit=0 \
    burst-limit=0 burst-threshold=0 burst-time=0s disabled=no
add name="guifi-routing" parent=guifi-qdisc packet-mark=wLan/Lan-routing \
    queue=default priority=1 limit-at=0 max-limit=0 \
    burst-limit=0 burst-threshold=0 burst-time=0s disabled=no
add name="guifi-realtime" parent=guifi-qdisc packet-mark=wLan/Lan-realtime \
    queue=default priority=2 limit-at=0 max-limit=0 \
    burst-limit=0 burst-threshold=0 burst-time=0s disabled=no
add name="guifi-bulk" parent=guifi-qdisc packet-mark=wLan/Lan-bulk \
    queue=default priority=3 limit-at=0 max-limit=0 \
    burst-limit=0 burst-threshold=0 burst-time=0s disabled=no

# Set packet marks
/ ip firewall mangle
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-routing \
    passthrough=no protocol=tcp dst-port=179 \
    comment="wLan/Lan - bgp" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-routing \
    passthrough=no protocol=ospf \
    comment="wLan/Lan - ospf" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-realtime \
    passthrough=no protocol=icmp \
    comment="wLan/Lan - ping" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-realtime \
    passthrough=no dst-port=53 protocol=udp \
    comment="wLan/Lan - DNS" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-realtime \
    passthrough=no protocol=tcp dst-port=5060 comment="wLan/Lan - TCP SIP" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-realtime \
    passthrough=no protocol=udp dst-port=5060 comment="wLan/Lan - UDP SIP" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-realtime \
    passthrough=no protocol=udp dst-port=16384-18000 comment="wLan/Lan - RTP" disabled=no
add chain=prerouting action=mark-packet new-packet-mark=wLan/Lan-bulk \
    passthrough=no comment="wLan/Lan - Other" \
    disabled=no
```

Linux

RouterOS utiliza kernel Linux, por tanto estamos en el mismo caso anterior con 8 colas, pero no tenemos una interfaz tan amigable de comandos. Crearemos

las colas y marcaremos el mismo tráfico que en RouterOS como podemos ver seguidamente.

```
tc qdisc del dev eth0 root
tc qdisc add dev eth0 root handle 1: prio bands 8

tc qdisc add dev eth0 parent 1:1 handle 10: sfq
tc qdisc add dev eth0 parent 1:2 handle 20: sfq
tc qdisc add dev eth0 parent 1:3 handle 30: sfq

ip route add 192.168.2.32/27  src 192.168.2.1 realm 2 dev eth2
ip route add 192.168.2.64/27  src 192.168.2.1 realm 3 dev eth2
ip route add 192.168.2.96/27  src 192.168.2.1 realm 4 dev eth2

tc filter add dev eth0 parent 1: protocol ip prio 1 route from 2 flowid 1:2
tc filter add dev eth0 parent 1: protocol ip prio 1 route from 3 flowid 1:3
tc filter add dev eth0 parent 1: protocol ip prio 1 route from 4 flowid 1:4

iptables -t mangle -A PREROUTING -p tcp -dport 179 -j MARK --set-mark 2 # bgp
iptables -t mangle -A PREROUTING -p ospf -j MARK --set-mark 2
iptables -t mangle -A PREROUTING -m state --state NEW,RELATED -p tcp \
         --tcp-flags ALL SYN -j MARK --set-mark 2
iptables -t mangle -A PREROUTING -m state --state NEW,RELATED -p tcp \
         --tcp-flags ALL ACK -j MARK --set-mark 2
iptables -t mangle -A PREROUTING -p icmp -j MARK --set-mark 2
iptables -t mangle -A PREROUTING -p tcp -dport 53 -j MARK --set-mark 2

iptables -t mangle -A PREROUTING -p tcp -dport 5060 -j MARK --set-mark 3
iptables -t mangle -A PREROUTING -p udp -dport 5060 -j MARK --set-mark 3
iptables -t mangle -A PREROUTING -p udp -dport 16384:18000 -j MARK --set-mark 3

iptables -t mangle -A PREROUTING -i eth2  -j MARK --set-mark 4 # guifi
```

Una forma más avanzada si el router tiene el paquete l7-protocolo es añadir prioridad por protocolo SIP y RTP. Pero ya que no es un paquete que vaya incluido por defecto en nuestros routers y ya que el aumento del consumo de CPU es significativo se prefiere no utilizar esta opción.

Apéndice C

FreeSWITCH

El objetivo de este apéndice es resumir la configuración de FreeSWITCH que ha sido automatizada y de algunos aspectos de optimización que requieren ser configurados por el administrador de forma manual como CAC ya que desde la aplicación no conocemos el ancho de banda real de los enlaces.

ACL

Por defecto prohibimos todos los accesos y añadimos el rango de IPs que utiliza guifi.net como permitido en 'conf/autoload_configs/acl.conf.xml'.

```
<configuration name="acl.conf" description="Network Lists">
  <network-lists>
    <list name="lan" default="deny">
      <node type="allow" cidr="10.0.0/8"/>
    </list>
  </network-lists>
</configuration>
```

LDAP

Para permitir al mod_xml_ldap de FreeSWITCH tener acceso al servidor LDAP se configurará de forma automática los datos necesarios: host del servidor de di-

rectorios, la contraseña de acceso, la base inicial de la jerarquía, el usuario que buscamos y la conversión entre los atributos usuario y contraseña de LDAP con las variables de FreeSWITCH. Todo esto se configurará automático en 'conf/autoload_configs/xml_ldap.conf.xml'

```
<configuration name="xml\_ldap.conf" description="XML LDAP">
  <bindings>
    <binding name="directory">
      <param name="filter" value="(SIPIdentityUserName=\%s)" bindings="directory"/>
        <param name="basedn" value="ou=People,dc=guifi,dc=net"/>
        <param name="url" value="ldap://10.139.50.2"; />
        <param name="binddn" value="cn=Manager,dc=guifi,dc=net"/>
        <param name="bindpass" value="guifi"/>
        <trans>
          <tran name="id" mapfrom="SIPIdentityUserName"/>
          <tran name="a1-hash" mapfrom="SIPIdentityPassword"/>
        </trans>
    </binding>
    <binding name="configuration">
      <param name="filter" value="(SIPIdentityUserName=\%s)" bindings="configuration"/>
      <param name="basedn" value="ou=People,dc=guifi,dc=net"/>
      <param name="url" value="ldap://10.139.50.2"; />
      <param name="binddn" value="cn=Manager,dc=guifi,dc=net"/>
      <param name="bindpass" value="guifi"/>
    </binding>
  </bindings>
</configuration>
```

Dialplan

El diaplan define la forma de acceder a una extensión determinada.

Llamadas internas

El dialplan interno de guifi.net de las llamadas salientes se basa en buscar en la extensión que queremos llamar a que centralita corresponde. El siguiente código es un ejemplo de 'conf/dialplan/default/guifi.xml'.

```
<include>
  <extension name="guifi">
```

```
    <condition field="destination\_number" expression="^(\d+)$">
      <X-PRE-PROCESS cmd="include" data="guifi/precall.xml"/>
      <action application="bridge" data="user/1\*$1@127.0.0.1"/>
      <X-PRE-PROCESS cmd="include" data="guifi/postcall.xml"/>
    </condition>

    <condition field="destination\_number" expression="^(1\*\d+)$">
      <X-PRE-PROCESS cmd="include" data="guifi/precall.xml"/>
      <action application="bridge" data="user/$1@127.0.0.1"/>
      <X-PRE-PROCESS cmd="include" data="guifi/postcall.xml"/>
    </condition>

    <condition field="destination\_number" expression="\^(2\*\d+)\$">
      <X-PRE-PROCESS cmd="include" data="guifi/precall.xml"/>
      <action application="bridge" data="user/$1@10.139.54.2"/>
      <X-PRE-PROCESS cmd="include" data="guifi/postcall.xml"/>
    </condition>

  </extension>
</include>
```

Todas las llamadas tienen un proceso previo y posterior genérico que para optimizar el código se ha creado en archivos separados.

Prellamada establece el tiempo de espera antes de pasar al buzón de voz, el nombre del archivo wav con el audio, el tipo de tono, el identificador de llamada, el fichero de configuración sería 'conf/dialplan/default/guifi/precall.xml'.

```
<include>
  <action application="bind_meta_app" data="1 b s execute_extension::dx XML features"/>
  <action application="bind_meta_app" data="2 b s record_session::$${recordings_dir}/ \
${caller_id_number}.${strftime(\%Y-\%m-\%d-\%H-\%M-\%S)}.wav"/>
  <action application="bind_meta_app" data="3 b s execute_extension::cf XML features"/>
  <action application="set" data="ringback=${us-ring}"/>
  <action application="set" data="transfer_ringback=$${hold_music}"/>
  <action application="set" data="call_timeout=30"/>

  <action application="set" data="hangup_after_bridge=true"/>
  <action application="set" data="continue_on_fail=true"/>
  <action application="hash" data="insert/${domain_name}-call_return/$1/${caller_id_number}"/>
  <action application="hash" data="insert/${domain_name}-last_dial_ext/$1/${uuid}"/>
  <action application="set" data="called_party_callgroup=${user_data($1@${domain_name} \
var callgroup)}"/>
  <action application="hash" data="insert/${domain_name}-last_dial/
\${called_party_callgroup}/${uuid}"/>
```

```
</include>
```

Postllamada establece la respuesta a la llamada y activa el buzón de voz si se consumió el tiempo de espera, el fichero de configuración sería 'conf/dialplan/default/guifi/postcall.xml'.

```
<include>
  <action application="answer"/>
  <action application="sleep" data="1000"/>
  <action application="bridge" data="loopback/app=voicemail:default ${domain_name} $1"/>
</include>
```

El dialplan de las llamadas se basa en transferir la llamada del dialplan público al default que es donde tenemos configurada las extensiones. El fichero de configuración público es 'conf/dialplan/public.xml'.

```
<extension name="public\_extensions">
  <condition field="destination\_number" expression="^(\d+\*\d+|\d+)$">
    <action application="transfer" data="$1 XML default"/>
  </condition>
</extension>
```

Llamadas externas

El dialplan de las llamadas salientes a proveedor se basa en comprobar que el usuario es el que tiene acceso a usar ese proveedor, luego se realiza con el mod_sofia que es el modulo SIP de FreeSWITCH la llamada al proveedor que este se encargará de enrutarla correctamente al destino, se define en

'conf/dialplan/default/guifi_gateway_outgoing_1*1.xml'.

```
<extension name="outgoing_1\*1">
  <condition field="ani" expression="^(1\*1)$" />
  <condition field="destination_number" expression="^\*(\d+)$">
    <action application="bridge" data="sofia/gateway/1\*1/$1"/>
  </condition>
</extension>
```

Y para las llamadas entrantes del proveedor nos llegan con un identificador que normalmente es el usuario o el teléfono público que el proveedor nos ha dado, lo que hace este dialplan es transferir la llamada a nuestro plan default donde tenemos configuradas las extensiones, se define en

'conf/dialplan/public/guifi_gateway_incoming_1*1.xml'.

```
<include>
  <extension name="incoming_1\*1">
    <condition field="destination\_number" expression="^(22801)$">
      <action application="set" data="domain_name=$${domain}"/>
      <action application="transfer" data="1\*1 XML default"/>
    </condition>
  </extension>
</include>
```

Antes de poder llamar hay un proceso de registro en el servidor del proveedor, estos son los datos de acceso necesarios que se especifican en

'conf/sip_profiles/external/11.xml'.

```
<include>
  <gateway name="1\*1">
    <param name="username" value="22801"/>
    <param name="password" value="contraseña"/>
    <param name="realm" value="sarevoz.com"/>
    <param name="proxy" value="sarevoz.com"/>
    <param name="expire-seconds" value="600"/>
    <param name="register" value="true"/>
    <param name="register-transport" value="udp"/>
    <param name="retry-seconds" value="30"/>
    <param name="ping" value="25"/>
  </gateway>
</include>
```

QoS

Utilizaremos en las llamadas internas Bypass Media para mejorar la latencia, para configurarlo en el dialplan de guifi añadiremos la opción de conexión directa entre terminales.

<action application="set"data="bypass_media=true/>

CAC

FreeSWITCH dispone de restricciones de control de la llamadas simultaneas, que lo llaman Set Call Admission Control, a continuación se ve la sintaxis del fichero

'conf/autoload_configs/switch.conf.xml'

```
<param name="max-sessions" value="10"/>
<param name="sessions-per-second" value="10"/>
```

Donde max-sessions controla el máximo de llamadas simultaneas y sessions-per-seconds lo mismo, pero por segundo. Estas dos limitaciones se aplican a nivel general del servidor por lo que limitan tanto el número de llamadas internas como externas.

Otra opción menos eficaz podría ser utilizar el módulo de FreeSWITCH mod_limit y limitar el número de llamadas simultaneas por cada extensión. El problema que tiene es que no sería a nivel general de la aplicación por lo únicamente sería eficaz para limitar a un usuario, pero sin garantizar la accesibilidad global del sistema.

```
<extension name="provider">
  <condition field="destination_number" expression="gateway">
    <action application="limit" data="db outgoing provider 2" />
    <action application="bridge" data="sofia/gateway/provider/${destnum}" />
  </condition>
</extension>
```

La solución perfecta sería poder limitar por interfaz de red ya que en guifi.net tenemos más ancho de banda que hacia internet, pero esta opción no es soportada por FreeSWITCH.

Procesos

Ya que nuestro servidor tiene recursos limitados y comparte los ciclos de CPU con más procesos es recomendable darle la prioridad adecuada con respecto a

los demás. Las prioridades en Linux van desde la más favorable -20 a la menos favorable que es 19 y por defecto, esta crea procesos de usuario con prioridad 20. Por ejemplo, usaremos prioridad 9.

```
# nice -n 9 freeswitch
```

Ya que en Linux por defecto el tamaño de la pila de los procesos no esta especificada, FreeSWITCH recomienda que si se quiere utilizar ulimit un tamaño máximo de 240.

```
# ulimit -s 240
```

Otro método es dejando que FreeSWITCH auto ajuste el tamaño de la pila para un rendimiento óptimo con la opción -waste que es la solución implementada en el proyecto.

```
# nice -n 9 freeswitch -waste
```

Apéndice D

Interfaz Web

El objetivo de este apéndice es resumir el proceso de configuración de la aplicación Web para el servicio de directorios y para el servidor de VoIP.

Servicio de directorios

Accedemos a http://guifi.net/node/add/guifi-service para crear el servidor de directorios y completamos los datos del formulario.

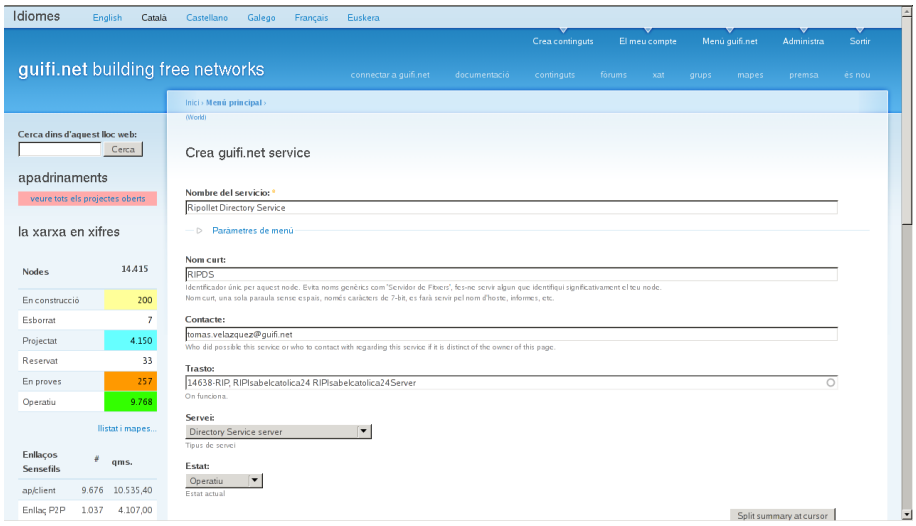

Figura D.1: Creación del servidor de directorios en la Web

Una vez presionado el botón de aceptar se carga la información del servicio que hemos creado que tiene el identificador 30425 y que nos hará falta a la hora de configurar el paquete guifi-ds.

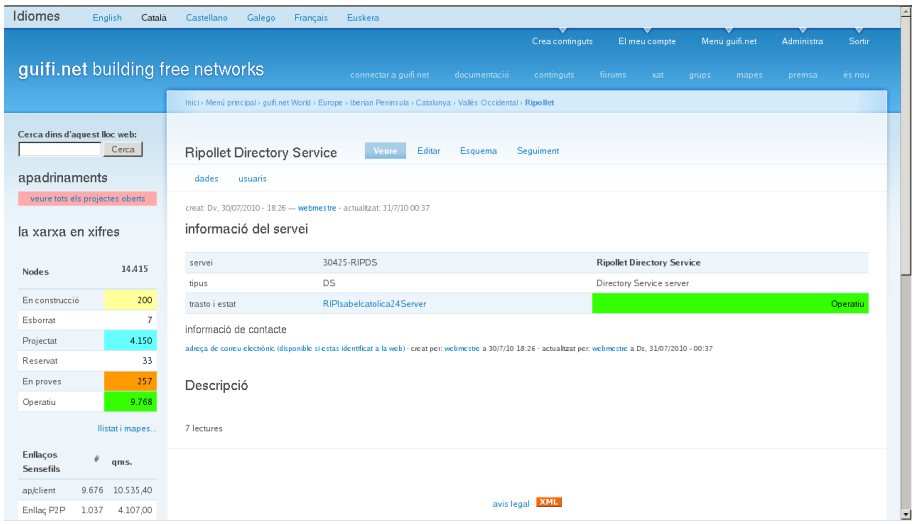

Figura D.2: Servidor de directorios una vez creado

El siguiente paso es asignar el servicio a la zona. Editamos en este caso esta zona 'http://guifi.net/node/9622/edit' y asociamos la zona con el servicio creado.

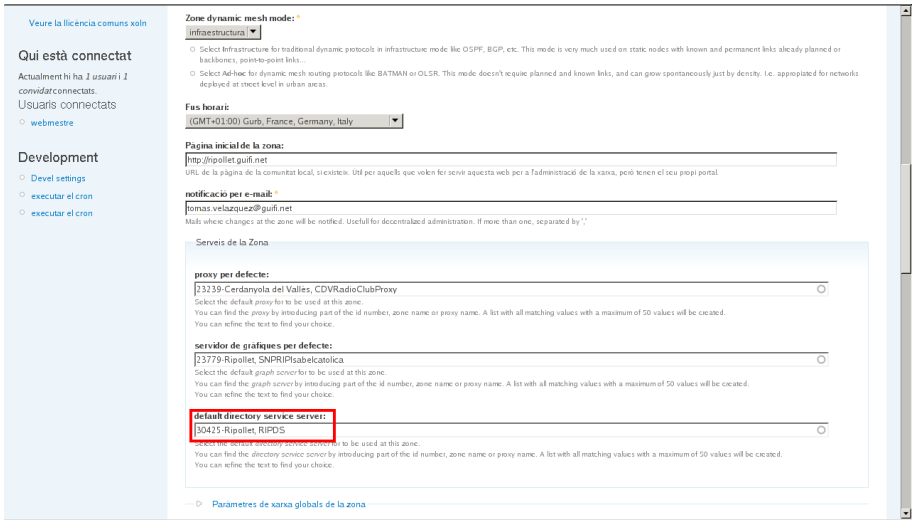

Figura D.3: Asociando el servidor de directorios a la zona

FreeSWITCH

Accedemos de nuevo a http://guifi.net/node/add/guifi-service para crear el servidor VoIP y completar los datos necesarios del formulario.

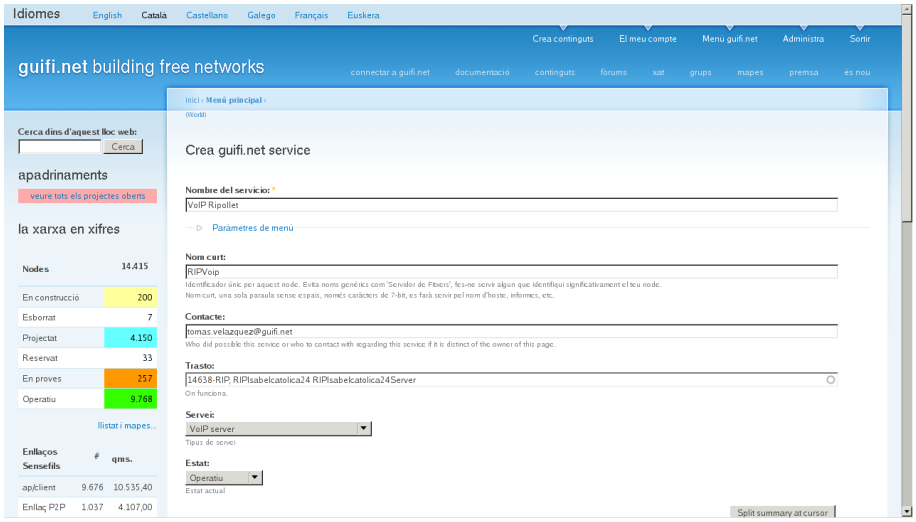

Figura D.4: Creación del servidor VoIP en la Web

Después de aceptar se carga la información del servicio creado y vemos que tiene el identificador 30426 que nos hará falta a la hora de configurar el paquete guifi-voip.

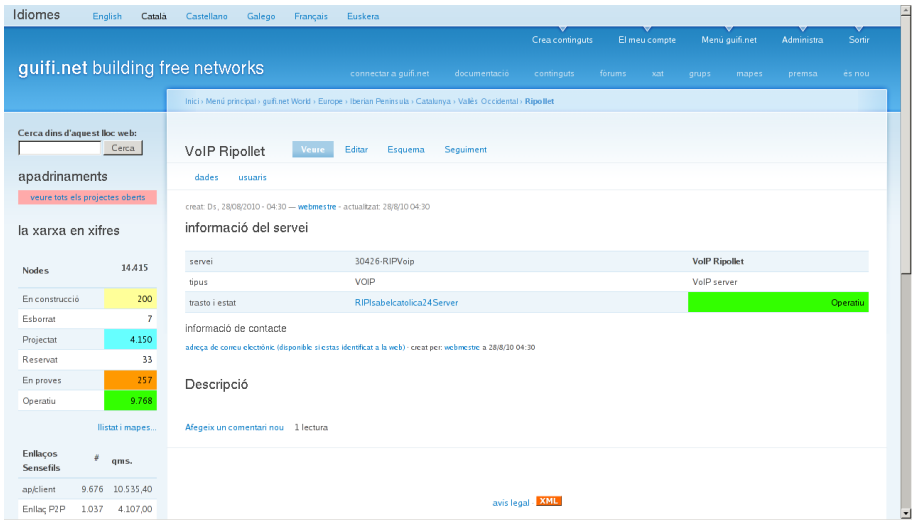

Figura D.5: Servidor VoIP una vez creado

El último paso es configurar las extensiones en los usuarios de un nodo. Podemos crear un nuevo usuario o editar uno ya creado añadiendo las opciones de VoIP. En este caso edito un usuario ya creado http://guifi.net/guifi/user/6281/edit.

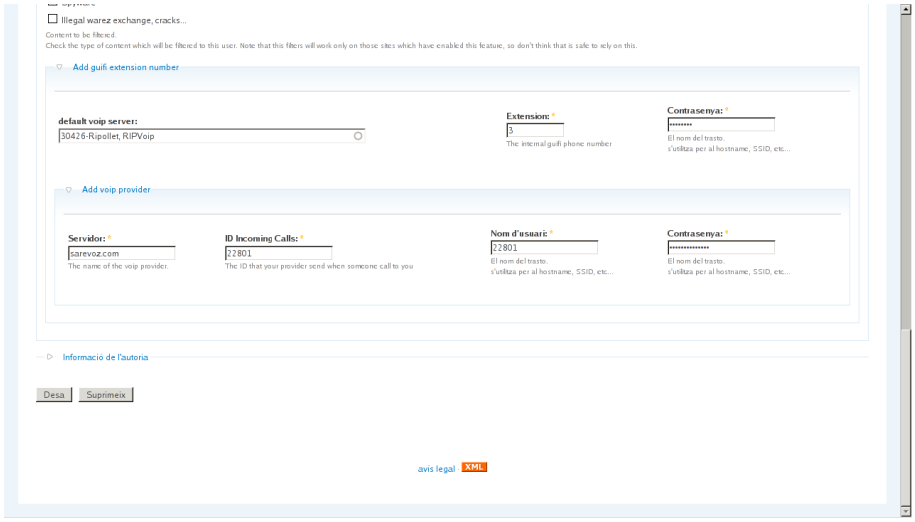

Figura D.6: Configuración VoIP de un usuario

Aceptamos y ya tenemos el usuario creado. El sistema cada 24 horas actualiza los datos del servidor LDAP y del servidor de VoIP.

Firmado: Tomás Velázquez García

Bellaterra, Septiembre de 2010

Resumen

Las redes inalámbricas y la VoIP están revolucionando la manera de comunicarnos debido a la fácil implementación y al bajo coste. En este proyecto se desarrolla un sistema de telefonía interna y externa para la red guifi.net que permite comunicar los servidores y los teléfonos de la red, y realizar llamadas a la red telefónica a través de un proveedor de VoIP. También se comprueban las limitaciones de la red inalámbrica y se muestran soluciones para optimizar los recursos disponibles para obtener mayor rendimiento.

Resum

Les Xarxes sense fils i la VoIP estan revolucionant la manera de comunicar-nos degut a la fàcil implementació i al baix cost. En aquest projecte es desenvolupa un sistema de telefonia interna i externa per a la xarxa guifi.net que permet comunicar els servidors i els telèfons de la xarxa, i realitzar trucades a la xarxa telefònica a través d'un proveïdor de VoIP. També es comproven les limitacions de la xarxa sense fils i es mostren solucions per optimitzar els recursos disponibles per obtenir major rendiment.

Abstract

Wireless networks and VoIP are revolutionizing the way we communicate for their easy implementation and low cost. This project develops a system of internal and external telephony for the network guifi.net, allowing servers and phones to communicate on the network, and make calls to the telephone network through a VoIP provider. Also check the limitations of the wireless network and show solutions to optimize available resources for more performance.