Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Computación

Sistema de comunicación con niveles de servicio basado en SIP para ambientes heterogéneos

Tesis que presenta

Ing. Rodrigo Vega García

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. José G. Rodríguez García

México, D.F. **Diciembre 2008**

Resumen

Las telecomunicaciones han logrado obtener velocidades grandes en el ancho de banda disponible para el medio inalámbrico. Sin embargo, están aún por debajo del ancho de banda disponible en un medio alámbrico. Hoy en día existen diferentes tipos de redes donde una de las principales características es el tipo de datos que transportan: así, en una Red pública de telefonía conmutada (PSTN) tiene como objetivo transportar audio, mientras que en una red de tipo Ethernet sólo transporta datos sin importar los requerimientos de los flujos enviados. Un flujo de audio tiene requerimientos de tiempo real, siendo muy sensible a retrasos y a la pérdida de paquetes. Los servicios basados en el Protocolo de Inicio de Sesiones (SIP) proveen una intercomunicación entre diferentes plataformas y uno de estos servicios es el de Voz sobre IP (VoIP).

En esta tesis se presenta una plataforma de VoIP que integra diferentes aspectos: (i) una interoperabilidad entre diferentes sistemas operativos al tener presente una heterogeneidad de dispositivos; (ii) una heterogeneidad de la red (alámbrica e inalámbrica); (iii) una convergencia del dominio PSTN y el de una red tipo Ethernet/Wi-Fi al utilizar una puerta de enlace mediante el Intercambio Privado de Ramificación (PBX) Asterisk y hardware asociado (tarjeta TDM); (iv) la plataforma tiene mecanismos de Calidad de Servicio (QoS), al utilizar un conmutador con capacidades de QoS DiffServ y un Punto de Acceso (PA) mejorado QoS (QPA) que utiliza el estándar 802.11e.

La provisión de QoS para el medio inalámbrico Wi-Fi bajo el estándar 802.11e requiere que tanto el PA, el dispositivo (tarjeta inalámbrica) y la aplicación tengan la capacidad de dar soporte de QoS. La plataforma propuesta cumple con dichos requisitos y principalmente integra en la aplicación VoIP el mecanismo de QoS necesario.

Para la evaluación de la plataforma, se realizaron pruebas experimentales que muestran el uso de dos Controles de Acceso al Medio (CAMs) en la conexión inalámbrica, para el transporte de un flujo con requerimientos de tiempo real como lo es un flujo de VoIP en un medio inalámbrico. Uno de estos controles provee QoS (802.11e) y el otro que no provee ningún mecanismo (802.11), demostrando que es indispensable agregar aspectos de QoS en la plataforma. La plataforma presentada también demuestra la interoperabilidad de diversos sistemas donde la piedra angular es SIP.

Abstract

Wireless networks are getting higher speeds, however available speeds in wired networks are highest. Nowadays there are several kinds of networks for different kinds of data flows. In this way, we have the Public switched telephone network (PSTN) for audio communication and Ethernet networks for data flows without traffic requirements. In PSTN networks audio traffic requests real-time requirements where packet lost and delays can not be tolerated. The SIP protocol gives services making an interface between several platforms, one of these services allows to implement Voice over IP (VoIP).

In this work we present a platform for VoIP including different capacities: i) interoperability between several operating systems given the heterogeneity of devices; ii) heterogeneity of networks through the use of wired and wireless networks; iii) convergence of networks sending traffic between PSTN and Ethernet/Wi-Fi netwoks through, a Private Branch eXchange PBX (Asterisk), and a Time-division multiplexing (TDM) card; iv) the platform we are proposing includes QoS capabilities by using a QoS-DiffServ capable switch and an improved access point using the 803.11e standard.

Supplying QoS in a Wi-Fi environment through the 802.11e standard needs that the Access Point, the devices, and the applications have capacities to support QoS. The proposed platform satisfies these requirements giving QoS capabilities to VoIP traffic.

We have evaluated our platform through some experimental tests showing the use of the Media Access Control (MAC) in a Wi-Fi environment with the 802.11 and the 802.11e standards to show the need of adding QoS capabilities to our VoIP Platform, at same time interoperability is shown through the use of devices with different operating systems and hardware.

Agradecimientos

Agradezco a mis padres por todo el apoyo incondicional que he tenido todos estos años de mi vida, por dejarme ser como soy y respetar mis ideas. A mi papá Luis Vega M. por ser un modelo a seguir y a mi mamá Delia García B. no sólo por ser una madre excepcional sino por haber encontrado en ella una gran amiga.

A mis hermanos L. Alberto, L. Alfonso y Alma D. por su gran apoyo, comprensión y sobre todo el haberme soportado en los tiempos difíciles.

A Alejandra por todo el amor y cariño que me has dado. Este trabajo también es tuyo por haber vivido estos dos emocionantes años de posgrado junto conmigo.

A mi gran amigo de la infancia el *chef* Iván. A mis tíos, amigos y compañeros de farra Ciro y Manolo. A E. Huacuz, Anel, Cynthia, Sofia y a todos mis amigos que no son menos importantes: los de la ESCOM, los de tepoz y los compañeros de posgrado.

A mi asesor el Dr. José G. Rodríguez García por todo el apoyo brindado en el desarrollo de la tesis, por la orientación, por sus conocimientos, por sus comentarios y cuantiosas observaciones a lo largo de este último año. También agradezco la confianza depositada en mi en diversos aspectos.

A los doctores Dominique Decouchant y Jorge Buenabad Chávez por las revisiones y aportaciones al trabajo de tesis.

A Sofia Reza por todo el apoyo brindado desde los primeros hasta los últimos días del posgrado.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo financiero durante los estudios y durante la investigación de la tesis de posgrado.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Hoy en día las redes de comunicación móviles de tercera generación 3G se vuelven más populares, gracias a la red de Acceso Múltiple de Banda Ancha por División de Código (WCDM) que fue inicializada en Japón en Octubre del 2001. Los servicios asociados con red de comunicación móvil de tercera generación (3G) proveen la habilidad de enviar voz y datos, además de otros servicios como correo electrónico, mensajerita instantánea [23]. Aunado a esto los dispositivos móviles soportan cada vez más funcionalidades [12], por ejemplo: cámara integrada, TV/Video, correo electrónico, etcétera.

El desarrollo de tecnologías como Wimax y Wibro [23] (consideradas como de generación 3.5 al ofrecer un mayor ancho de banda y alcance) y el incremento en la velocidad de transmisión de datos en los sistemas de comunicación móviles, permiten el desarrollo de aplicaciones para una gran variedad de servicios.

Ante el rápido desarrollo de las tecnologías inalámbricas, algunas organizaciones como la Unión Internacional de Telecomunicaciones (ITU), el Instituto de Ingenieros Eléctrico y Electrónicos (IEEE), el Instituto Europeo de Normas de Telecomunicaciones (ETSI) entre otras, deciden tomar un papel más importante en la definición de especificaciones para la tecnología móvil.

Uno de los resultados del trabajo conjunto de estas organizaciones fue la definición de Subsistema Multimedia IP (IMS) cuya arquitectura fue definida por el Proyecto de Colaboración de Tercera Generación (3GPP); este sistema propone una infraestructura de red móvil, que permite la convergencia de datos y voz, sobre una tecnología basada en el Protocolo de Internet (IP) [1]. El protocolo principal de IMS es el Protocolo de Inicio de Sesiones (SIP) que ofrece, extensiones de acuerdo al tipo de servicio que se quiere brindar [24]. La *convergencia* en IP es la fusión de dominios móviles en Internet, utilizando redes basadas en paquetes como infraestructura común.

El protocolo SIP fue definido por la Fuerza de Trabajo de Ingenieros de Internet (IETF) como un protocolo para crear y modificar sesiones, donde se permite a los participantes de acordar el conjunto de tipos de media compatibles. SIP provee sólo primitivas y no servicios, por lo cual puede ser visto como un componente que debe

ser utilizado en conjunto con otros protocolos, a fin de poder construir una plataforma multimedia [28].

Ante el éxito de Internet numerosos proyectos han surgido para hacer uso de ella con fines específicos, como es el caso de Voz o Televisión por IP. Asterisk es una plataforma para la telefonía sobre IP, además de ser un software gratuito, que funciona sobre todo en plataformas Linux. Puede considerarse a Asterisk como un puente entre la telefonía tradicional (PSTN) y la telefonía sobre IP (VoIP), ya que maneja un canal de comunicación basado en SIP para la señalización de VoIP [22].

Para poder brindar un servicio adecuado (en cuanto a calidad de audio y coherencia de mensajes) es necesario utilizar mecanismos que brinden diferentes niveles de servicio para los diferentes flujos de tráfico que se producen, es decir niveles de Calidad de Servicio (QoS) [17]. La necesidad de la QoS ha surgido por la evolución de las redes de telecomunicaciones, dado que mientras más diversidad de tipos de tráfico se tengan (cada uno con requerimientos específicos) más necesario es contar con un control adecuados en el manejo de los recursos. El manejo adecuado de los recursos en las redes móviles heterogéneas es una nueva línea de investigación en las áreas de redes, sistemas distribuidos y protocolos.

El objetivo principal en esta tesis incluye la instalación de una red y el desarrollo de un sistema de comunicación basado en VoIP, que permita la comunicación entre dispositivos con diferentes características (como arquitecturas, capacidades de almacenamiento, tamaño de pantalla, etc.). Además este sistema debe permitir la interacción de usuarios con diferentes privilegios y soportar aplicaciones con requerimientos diferentes. Todo esto debe garantizar una calidad de comunicación aceptable.

Para lograr el objetivo principal se construyó una plataforma de Voz sobre IP (VoIP). Esta plataforma tiene la habilidad de conectar el dominio Red pública de telefonía conmutada (PSTN) con el dominio de IP, con el objetivo de experimentar el concepto de convergencia en IP. Un servidor Asterisk es pieza fundamental para lograr dicho objetivo.

La plataforma incluye características como la interacción de sistemas operativos diversos, como lo son los sistemas embebidos Symbian, Windows Mobile y Familiar Linux, tanto como los sistemas no embebidos Linux y Windows. La red construida es capaz de comunicar dispositivos tanto alámbricos como inalámbricos que cuenta principalmente con mecanismos de QoS.

La provisión de QoS en el medio alámbrico se centra al hacer uso del mecanismo de Servicios Diferenciados (DiffServ) y para el medio inalámbrico Wi-Fi se utiliza la tecnología relativamente nueva Multimedia Inalámbrica (WMM). Con los mecanismos utilizados de QoS es posible distinguir un número limitado de flujos producidos por las aplicaciones que usan la plataforma. Esta es la manera de como se realiza la distinción de privilegios entre los usuarios y que en realidad son grupos de flujos.

En esta plataforma las aplicaciones son quienes dictan el nivel de servicio que requieren los flujos IP, donde la distinción básica presente es la de diferenciar tres tipos de flujos: el flujo de voz (los paquetes RTP), la señalización para el manejo

de las llamadas VoIP (SIP) y algún otro tipo de flujo presente. La última pieza fundamental que complementa la plataforma, es el desarrollo de aplicaciones para los diversos sistemas operativos que ella soporta, encontrándose la solución al utilizar la biblioteca PJ que resulta muy conveniente para el desarrollo de aplicaciones basadas en SIP y a un nivel alto siendo posible utilizar una Interfaz de Programación de Aplicaciones (API) para aplicaciones VoIP.

La metodología que fue utilizada para lograr el objetivo principal del trabajo se muestra a continuación:

1. Se revisó la documentación y el estado del arte, la literatura y las propuestas existentes de tecnologías que se usan en la convergencia. Se estudió concretamente SIP, el estándar IEEE 802.11 WLAN, mecanismos de QoS DiffServ y WMM y los protocolos de comunicación que se utilizan al haber establecido una sesión, por ejemplo: RTP, SDP, RTCP, etc.

2. Se realizó una instalación y configuración de la red y de dispositivos (*smartphone*, Asterisk, IP-phone, PDA, etc.). Se cuenta con un laboratorio de comunicación entre dispositivos con arquitecturas diferentes. Se estudiaron manuales de usuario y de desarrollo.

3. Se realizaron pruebas de comunicación entre los dispositivos. En esta etapa sólo se buscó que la comunicación se establezca sin problemas, localización de dispositivos.

4. Se probaron aplicaciones de la biblioteca PJ para dispositivos que cuentan con diferentes plataformas (sistema operativo y procesador) con la finalidad de demostrar la interoperabilidad entre ellos.

5. Se desarrolló una aplicación básica de VoIP, que logra hacer y recibir llamadas VoIP.

6. Se estableció una diferenciación de tipos de flujos de datos para priorizar a unos sobre otros. El objetivo principal fue el proveer al flujo de voz con la mejor prioridad.

7. Se realizaron pruebas de desempeño para verificar la calidad de las sesiones de voz en la red compartida. El eslabón más débil es el medio inalámbrico y en ella se centraron las pruebas.

8. Se reporta de manera detallada el planteamiento de la prueba anterior y los resultados obtenidos.

El resultado del trabajo desarrollado contiene toda una infraestructura de comunicación VoIP, donde diversas plataformas logran establecer comunicación, es posible

desarrollar aplicaciones basadas en SIP y se cuenta con una propiedad no menos importante al establecer la prioridad que sus tráficos requieren a través de la red. Esto último requirió la incorporación del mecanismo de diferenciación de servicios en la biblioteca PJ.

Debido a la gran cantidad de acrónimos existentes en este campo de investigación, durante la lectura del presente reporte se ruega al lector que acuda a la lista pertienente que se encuentra al final de este trabajo.

El presente documento de tesis se encuentra organizado en cinco capítulos de la siguiente manera. El capitulo 2 estudia los protocolos más sobresalientes en una comunicación VoIP, además de conceptos teóricos básicos de tipos de codificadores, *jitter buffer*, se da también una visión general de la biblioteca PJ y Asterisk.

El capítulo 3 contiene un compendio de conceptos fundamentales de QoS para redes IP, mecanismos de QoS para el medio alámbrico con DiffServ e información basta del soporte WMM para la provisión de QoS para el medio inalámbrico en redes Wi-Fi.

En el capítulo 4 se presenta el desarrollo y características de la plataforma de VoIP, se muestran configuraciones pertinenes acerca de la plataforma (red, Intercambio Privado de Ramificación (PBX), TDM, etc.) y el como se integra QoS en la plataforma con el objetivo principal de contar con el soporte WMM. Se presenta el desarrollo de una aplicación simple para llamadas VoIP.

El capítulo 5 plantea una serie de pruebas del desempeño del flujo VoIP, bajo diferentes condiciones en el medio inalámbrico, además de desarrollar una descripción de como se obtuvieron los parámetros objetivos presentes en un flujo VoIP para ser analizados.

En el capítulo 6 se presentan los resultados obtenidos y un análisis a detalle de las diferentes pruebas descritas en el capítulo 5. Los resultados presentados son parámetros objetivos utilizados para la medición de una QoS cuantitativa, específicamente para medir el desempeño que tiene el canal para enviar flujos con requerimientos como lo es un flujo VoIP.

Por último, en el capítulo 7 se concluye la investigación que abarca la tesis y se presenta el trabajo a futuro.

Capítulo 2

Protocolos para una comunicación de voz sobre IP

En este capítulo se presentan varios de los protocolos relacionados en una comunicación de Voz sobre IP (VoIP). Los protocolos aquí presentados sólo son algunos de los utilizados para lograr y realizar una comunicación VoIP. El campo de investigación y desarrollo de una plataforma para voz sobre el Protocolo de Internet (IP) es vasto, y en el caso de esta investigación no incluye todos sus ámbitos.

En una red local no son necesarios ciertos mecanismos para lograr una conexión entre dos computadoras. Un servidor STUN, es un ejemplo de este tipo de mecanismos: Transversal Simple de UDP a través de NATs (STUN) es un protocolo ligero que permite a las aplicaciones descubrir la presencia, tipos de NATs, cortafuegos entre ellas y la Internet pública [29]. Otros mecanismos en desarrollo utilizados son los borradores IETF: Transversal que Utiliza Repetidores en torno a NAT (TURN) [27] y el Establecimiento de Conectividad Interactiva (ICE) [26].

TURN permite a un *host* dentro de una NAT (llamado cliente TURN) requerir a otro *host* (servidor TURN) actuar como repetidor de paquetes. ICE utiliza protocolos, como STUN y TURN. ICE hace uso de STUN de una manera cooperativa punto a punto, permitiendo a los participantes descubrir, crear y verfificar conectividad mutua.

Los protocolos presentados en este capítulo son extractos de Solicitud de Comentarios (RFC) provistos por la Fuerza de Trabajo de Ingenieros de Internet (IETF) con la intención de denotar el funcionamiento básico, para lograr un entendimiento de subsecuentes secciones.

2.1 Protocolo de Inicio de Sesiones (SIP)

El Protocolo de Inicio de Sesiones (SIP) ha ganado su lugar como el protocolo necesario para VoIP. Se espera que todos los productos nuevos de usuario y empresariales cuenten con soporte SIP, además de que SIP provee más allá que sólo la

capacidad VoIP, incluyen la habilidad de transmitir video, música y cualquier tipo de flujo multimedia de tiempo real [22]. SIP se especifica en el RFC 3261 [28] y la información a continuación presentada es una síntesis de dicho RFC.

Existen varias aplicaciones de Internet que requieren la creación y manejo de una sesión. Una sesión es considerada como un intercambio de datos entre una asociación de participantes. La implementación de dichas aplicaciones resultan complicadas debido a la practicas de los participantes: los usuarios podrían moverse entre varios lugares, podrían estar referenciados por múltiples nombres y podrían estar comunicados con diferentes tipos de media (algunas veces simultáneamente). Numerosos protocolos han sido creados para transportar varias tipos de datos de sesiones multimedia en tiempo real tales como voz, video o mensajes de texto. SIP trabaja conjuntamente con esos protocolos, al perimitir que terminales llamadas Agentes de Usuario (AU) puedan descubrirse entre sí y establecer el acuerdo en la caracterización de una sesión que a ellos les gustaría compartir. Para la eventual localización de participantes en una sesión y para otras funciones, SIP permite la creación de una infraestructura de computadoras anfitrionas en red (llamados servidores *proxy*) a las cuales las AU pueden enviar registros, invitaciones a sesiones y otras peticiones. SIP es un herramienta ágil de propósito general para crear, modificar y terminar sesiones, que funciona de manera independiente de las capas inferiores de protocolos de transporte y sin dependencia del tipo de sesión que esté siendo establecida.

2.1.1 Funcionalidad general de SIP

SIP es un protocolo de control en la capa de aplicación que puede establecer, modificar y terminar sesiones multimedia (conferencias) tales como llamadas telefónicas por Internet. SIP puede invitar participantes a sesiones existentes, como en conferencias de multidifusión. La media puede ser agregada (y removida) de una sesión existente. SIP soporta de manera transparente el mapeo de nombres y servicios de redirección, dando soporte a una movilidad personal (los usuarios pueden mantener un sencillo identificador externamente visible a pesar de la localización en la red). SIP da soporte a cinco facetas para establecer y terminar comunicaciones multimedia:

localización de usuario: determinación del sistema final (la IP del dispositivo del usuario) para ser usado en la comunicación.

disponibilidad de usuario: determinación de la disponibilidad de la parte a quien se llama para enlazar una comunicación.

capacidades de usuario: determinación de la media y los parámetros de media a ser utilizados.

preparación de sesión: "*ringing*", establecimiento de los parámetros de la sesión en ambas partes, a quien se llama y quien llama.

manejo de sesión: incluye la transferencia y la terminación de sesiones, la modificación de parámetros de sesión e invocar servicios.

SIP no es un sistema de comunicaciones verticalmente integrado, SIP es un componente que puede ser utilizado con otros protocolos de la IETF para construir una arquitectura multimedia completa. Típicamente, este tipo de arquitecturas incluirán otros protocolos como el Protocolo de Tiempo Real (RTP) RFC 3550 [31] para transportar datos de tiempo real y proveer retroalimentación de Calidad de Servicio (QoS); el Protocolo de Flujo de Tiempo Real (RTSP) RFC 2326 [32] para controlar la entrega del flujo de media; el Protocolo de Control de Puertas de enlace de Media (MEGACO) RFC 3015 [10] para controlar puertas de enlace en la Red pública de telefonía conmutada (PSTN) y el Protocolo de Descripción de Sesión (SDP) RFC 2327 [13] para describir sesiones multimedia. Por consiguiente, SIP deberá ser usado en conjunto con otros protocolos a fin de proveer servicios completos a los usuarios. Sin embargo, la operación y funcionalidad básica de SIP no depende de cualquiera de aquellos protocolos.

SIP no provee servicios. SIP provee únicamente primitivas que puedan ser utilizadas para implementar diferentes servicios. Por ejemplo, SIP puede localizar a un usuario y entregar un objeto no transparente en su actual localización. Si dicha primitiva es utilizada para hacer entrega de una descripción de sesión escrita en SDP, en principio las terminales pueden acordar los parámetros de una sesión. Si la misma primitiva es utilizada para hacer entrega de una foto de quien llama, tanto como la descripción de la sesión, un servicio de "identificador de llamadas" puede ser fácilmente implementado. Como lo muestra el ejemplo anterior, una primitiva simple es típicamente utilizada para proveer múltiples servicios diferentes.

SIP no ofrece servicios de control de conferencia como lo es un *control de piso* o una votación y tampoco determina como una conferencia tiene que ser manejada. SIP puede ser utilizada para iniciar una sesión que use algún otro protocolo de control de conferencia. Dados que los mensajes y las sesiones establecidas SIP puedan atravesar por completo a través de diferentes redes, SIP no hace (y realmente no puede), la provisión de cualquier tipo de capacidad de reservación de recursos de red.

La naturaleza de los servicios provistos hacen de la seguridad un aspecto importante. Para tal fin, SIP provee una serie de servicios de seguridad, las cuales incluyen una prevención de *denegación de servicio*, autenticación (usuario a usuario y *proxy*[1] a usuario), protección de integridad, cifrado y servicios de privacidad.

SIP funciona tanto para IP versión 4 e IP versión 6.

2.1.2 Esquema de operación

El ejemplo que vamos a presentar tiene por objetivo de ilustrar la funcionalidad básica de SIP: localización de una terminal, señalamiento de la intención de comuni-

[1]Un *proxy* hace referencia a un programa o dispositivo que realiza una acción en representación de otro.

cación, negociación de parámetros de sesión para establecer la sesión y desmontar la sesión una vez establecida.

La figura 2.1 muestra un ejemplo típico de un intercambio de mensajes SIP entre dos usuarios, "Alicia" y "Bob". Cada mensaje es etiquetado con la letra "F" seguido de un número para ser referido en el texto. En el ejemplo, Alicia utiliza una aplicación SIP en su Computadora Personal (PC) (referido como un *softphone*) para llamar al teléfono SIP de Bob a través de Internet. También se muestran dos servidores *proxy* que actúan en beneficio de Alicia y Bob para facilitar el establecimiento de la sesión. Este arreglo es comúnmente referido como un "trapecio SIP", obsérvese la figura geométrica en la parte superior de la figura 2.1.

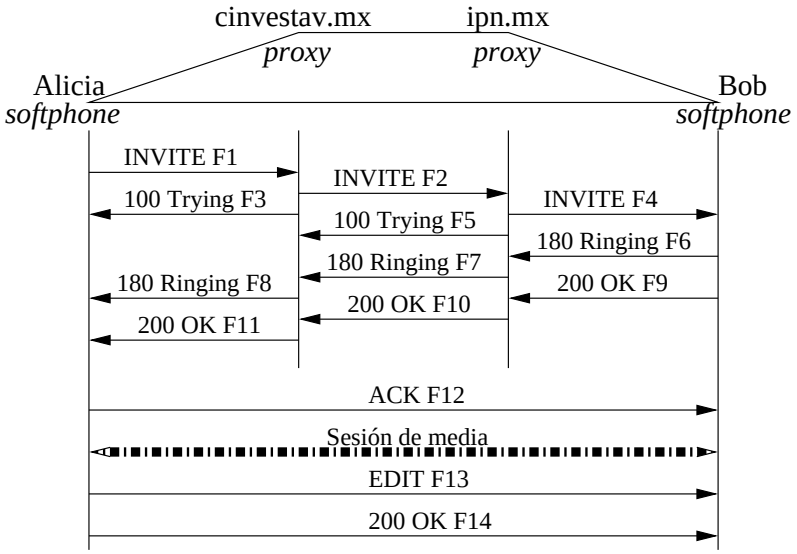

Figura 2.1: Ejemplo de una conformación de una sesión SIP

Alicia "llama" a Bob utilizando su identidad SIP, un tipo de Identificador Uniforme de Recurso (URI) llamado SIP URI. El identificador tiene una forma similar a una dirección de correo electrónico, típicamente contiene un nombre de usuario y un nombre de un servidor. En este caso, resulta ser *sip:bob@ipn.mx*, donde ipn.mx es el dominio del proveedor de servicios SIP de Bob. Alicia tiene un SIP URI de *sip:alicia@cinvestav.mx*. SIP también provee un URI seguro, llamado SIPS URI. Un ejemplo seria *sips:bob@ipn.mx*. Una llamada hecha a un SIPS URI garantiza que un trasporte cifrado seguro (llamado TLS[2]) es utilizado para llevar todos los mensajes SIP. Desde este punto, la petición es enviada de manera segura hacia quien se llama, pero con mecanismos de seguridad que dependen de las políticas del dominio de a quien se le llama.

[2]En inglés *Transport Layer Security*.

SIP esta basado en un modelo de transacción petición/respuesta como en HTTP. Cada transacción consiste de una petición que invoca un método en particular o una función en un servidor, de donde al menos se obtiene una respuesta. En el ejemplo, la transacción inicia en el *softphone* de Alicia que envía una petición *INVITE* direccionada al SIP URI de Bob. *INVITE* es un ejemplo de un método SIP que especifica la acción que el solicitante (Alicia) quiere que el servidor (Bob) realice. La petición *INVITE* contiene un número de campos de encabezado. Los campos de encabezado son llamados atributos, que proveen información adicional acerca del mensaje. Los presentes en un *INVITE* incluyen un identificador único para la llamada, la dirección de destino, la dirección de Alicia y la información acerca del tipo de sesión que Alicia desea establecer con Bob. El mensaje F1 de la figura 2.1 podría tener la siguiente estructura:

```
INVITE sip:bob@ipn.mx SIP/2.0
Via: SIP/2.0/UDP pc33.cinvestav.mx;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@ipn.mx>
From: Alicia <sip:alicia@cinvestav.mx>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.cinvestav.mx
CSeq: 314159 INVITE
Contact: <sip:alicia@pc33.cinvestav.mx>
Content-Type: application/sdp
Content-Length: 142
(El SDP de Alicia no es mostrado)
```

La primer línea del mensaje de texto codificado contiene el nombre del método *INVITE*, las líneas que siguen son una lista de campos. El ejemplo de mensaje anterior contiene un conjunto de campos mínimo requerido. Los campos de encabezado son brevemente descritos a continuación:

Via contiene la dirección (`pc33.cinvestav.mx`) en la cual Alicia espera recibir respuestas de la petición. También contiene un parámetro que identifica a la transacción.

To contiene un nombre desplegable (Bob) y un SIP o SIPS URI (`sip:bob@ipn.mx`) hacia el cual la petición fue originalmente dirigido.

From también contiene un nombre desplegable (Alicia) y un SIP o SIPS URI (`sip:alicia@cinvestav.mx`) que indica el origen de la petición. Este campo de encabezado también tiene un parámetro etiqueta que contiene una cadena aleatoria (`1928301774`), que fue agregada al URI por el *softphone*. Es utilizado para propósitos de identificación.

Call-ID contiene un identificador único global para la llamada en cuestión, generada por la combinación de la cadena aleatoria y el nombre del servidor o dirección IP del *softphone*. La combinación de la etiqueta `To`, `From` y `Call-ID` define completamente una relación SIP punto-a-punto entre Alicia y Bob y es referida como un diálogo.

secuencia de comando (CSeq) contiene un entero y un nombre de método. El número `CSeq` es incrementado por cada petición nueva dentro de un diálogo y es una secuencia tradicional de números.

Contact contiene un SIP o SIPS URI que representa una ruta directa para contactar a Alicia, usualmente compuesta de un nombre de usuario y un Nombre de Dominio Totalmente Calificado (FQDN). Mientras un FQDN es preferido, varios sistemas no tienen nombres de dominio registrados, de tal modo qué direcciones IP son permitidas. Mientras el campo de encabezado `Via` le dice a los demás elementos a donde enviar las respuestas, el campo de encabezado `Contact` le dice a los otros elementos a donde enviar peticiones futuras.

Max-Forwards sirve para limitar el número de saltos que una petición puede hacer durante el camino a su destino. Consiste en un entero que es decrementado en uno por cada salto.

Content-Type contiene una descripción del cuerpo del mensaje (no mostrado).

Content-Length contiene una cuenta del cuerpo del mensaje en un octeto (byte).

La lista completa de los campos del encabezado SIP se encuentra en la sección 20 del RFC de SIP [28].

Los detalles de la sesión, tales como el tipo de media, codificador o la tasa de muestreo, no son descritos utilizando SIP. El cuerpo de un mensaje SIP contiene una descripción de la sesión, codificada en algún otro formato de protocolo. Un tipo de dicho formato es el Protocolo de Descripción de Sesión (SDP) [13]. El mensaje SDP, es transportado de una forma similar al documento adjunto en un mensaje de correo electrónico, o una página web al ser transportada en un mensaje HTTP.

Dado que el *softphone* no sabe la localización de Bob o el servidor SIP en el dominio `ipn.mx`, el *softphone* envía un *INVITE* al servidor SIP que sirve al dominio de Alicia (`cinvestav.mx`). La dirección del servidor SIP `cinvestav.mx` pudo haber estado configurado en el *softphone* de Alicia, o pudo haber sido descubierto mediante el Protocolo de la Configuración de *Host* Dinámico (DHCP)[3].

El servidor SIP `cinvestav.mx` es un tipo de servidor SIP conocido como servidor *proxy*. Un servidor *proxy* recibe peticiones SIP y las reenvía en beneficio del solicitante. En el ejemplo, el servidor *proxy* recibe la petición *INVITE* y envía una respuesta 100 (*Trying*) hacia el *softphone* de Alicia. La respuesta *Trying* 100 indica que el *INVITE*

[3]En inglés *Dynamic Host Configuration Protocol*.

ha sido recibido y que el *proxy* se encuentra en funcionamiento para encaminar el *INVITE* hacia su destino.

Las respuestas en SIP utilizan un código de tres dígitos seguidos de una frase descriptiva. Esta respuesta contiene los mismos parámetros `To`, `From`, `Call-ID`, `CSeq` y vertientes en `Via` encontrados en el *INVITE*, lo cual permite al *softphone* de Alicia correlacionar la respuesta con el *INVITE* enviado. El servidor *proxy* `cinvestav.mx` localiza al servidor *proxy* en `ipn.mx`, posiblemente al ejecutar un tipo particular de búsqueda DNS. Como resultado, obtiene la dirección IP del servidor *proxy* `ipn.mx` y reenvía, o *proxies*, la petición *INVITE* a dicho servidor.

Antes de reenviar la petición, el servidor *proxy* `cinvestav.mx` agrega un valor adicional en el campo de encabezado `Via` que contiene su propia dirección. El servidor *proxy* `ipn.mx` recibe el *INVITE* y responde con una respuesta 100 *Trying* hacia el servidor *proxy* `cinvestav.mx`, para indicar que ha recibido el *INVITE* y que se encuentra procesado la petición. El servidor *proxy* consulta una base de datos, generalmente llamado un servicio de localización, que contiene la dirección IP actual de Bob. El servidor *proxy* `ipn.mx` agrega otro valor en el campo `Via` del encabezado con su propia dirección en el *INVITE* y lo *proxea* hacia el teléfono SIP de Bob.

El teléfono SIP de Bob recibe el *INVITE* y alerta a Bob de la llamada entrante de Alicia de tal manera que Bob pueda decidir en contestar la llamada, que resulta en el timbrar del teléfono de Bob. El teléfono SIP de Bob indica esto con una respuesta *Ringing* 180, la cual es encaminada de regreso a través de los dos *proxies* en dirección contraria. Cada *proxy* utiliza el campo `Via` del encabezado para determinar a donde enviar la respuesta y remueve su propia dirección que se encuentra al último.

Como resultado, a pesar del DNS y del servicio de localización requerido para encaminar el *INVITE* inicial, la respuesta *Ringing* 180 puede ser retornada hacia quien llama sin búsquedas o sin utilizar los *proxies*. Esto tiene una propiedad posiblemente deseable de que cada *proxy* que vea el mensaje *INVITE* podrá también ver todas las respuestas al *INVITE*.

Cuando el *softphone* de Alicia recibe la respuesta *Ringing* 180, transfiere esta información a Alicia, quizás utilizando un audio de tono de timbre o al desplegar un mensaje en la pantalla de Alicia.

En el ejemplo, Bob decide contestar la llamada. Cuando descuelga el auricular, su teléfono SIP envía una respuesta *OK* 200 para indicar que la llamada ha sido contestada. La respuesta *OK* 200 contiene un cuerpo de mensaje con una descripción de media SDP del tipo de sesión que Bob está dispuesto a establecer con Alicia.

Como resultado, existe un intercambio de dos facetas de mensajes SDP: Alicia envía una a Bob y Bob envía una en respuesta a Alicia. Este intercambio de dos facetas provee una capacidad básica de negociación y se encuentra basada en un modelo simple de ofrecer/contestar de intercambio SDP. Si Bob no deseó contestar la llamada o se encontraba ocupado en otra llamada, una respuesta de error habría sido enviada en lugar del *OK* 200, lo cual hubiera resultado en no haber establecido una sesión de media. La lista completa de los códigos de respuestas SIP se encuentran en

la sección 20 del RFC de SIP.

La respuesta *OK* 200 que Bob envía (mensaje F9 de la figura 2.1) podría ser de la siguiente manera:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.ipn.mx
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.ipn.mx
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.cinvestav.mx
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@ipn.com>;tag=a6c85cf
From: Alicia <sip:alicia@cinvestav.mx>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.cinvestav.mx
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
(No se muestra el SDP de Bob)
```

La primer línea de la respuesta contiene un código de respuesta 200 y una frase *OK*. Las líneas restantes contienen campos del encabezado. Los campos del encabezado `To`, `From`, `Call-ID` y `CSeq` son copiados de la petición *INVITE*. El teléfono SIP de bob ha agregado un parámetro etiqueta al campo `To`. Esta etiqueta será incorporada por ambas terminales en el diálogo y será incluida en todas las peticiones y respuestas futuras en la llamada. El campo `Contact` contiene una URI en la cual Bob puede ser directamente encontrado en su teléfono SIP. Los campos `Content-Type` y `Content-Length` refieren al cuerpo del mensaje (no mostrado) que contienen información de la media SDP de Bob.

Además de Servidor de Nombres Dinámico (DNS) y el servicio de búsqueda de localización mostrados en este ejemplo, los servidores *proxy* puede hacer *decisiones de encaminamiento* flexibles para decidir a donde enviar una petición. Por ejemplo, si el teléfono SIP de Bob regresa una respuesta *Busy Here* 486, el servidor *proxy* `ipn.mx puede` *proxear* el *INVITE* al servidor de correo de voz de Bob. Un servidor *proxy* puede incluso enviar un *INVITE* a varias locaciones al mismo tiempo. Este tipo de búsqueda paralela es conocida como *forking*.

En tal caso, la respuesta *OK* 200 es encaminada de regreso a través de dos *proxies* y es recibida por el *softphone* de Alicia, la cual entonces detiene el tono de timbre e indica que la llamada ha sido contestada. Finalmente, el *softphone* de Alicia envía un mensaje de reconocimiento, *ACK*, dirigido al teléfono SIP de Bob para confirmar la recepción de la respuesta final (*OK* 200). En el ejemplo, el *ACK* es enviado directamente del *softphone* de Alicia hacia el teléfono SIP de Bob, eludiendo ambos *proxies*. Esto ocurre debido a que las terminales han conocido la dirección de la otra a partir

de los campos `Contact` del encabezado que generó el intercambio *INVITE*/*OK* 200, el cual no era conocido cuando el *INVITE* inicial fue enviado. Las búsquedas ejecutadas por los dos *proxies* ya no son requeridas, de tal modo que los *proxies* quedan fuera del flujo de llamada. Esto completa el acuerdo triple *INVITE*/200/*ACK* utilizado para establecer sesiones SIP.

La sesión media de Alicia y Bob es ahora iniciada y envían paquetes de media utilizando el formato que ellos acordaron en el intercambio de SDP. En general, los paquetes media principio-a-fin toman un camino diferente a los mensajes de señalización SIP.

Durante la sesión, tanto Alicia como Bob podrían decidir el cambiar las características de la sesión media. Esto es logrado al enviar un re-*INVITE* que contiene una descripción nueva de media. Dicho re-*INVITE* referencia al diálogo existente de tal modo que la otra parte sabe que es para modificar la sesión existente en lugar de establecer una sesión nueva. La otra parte envía un *OK* 200 para aceptar el cambio. El solicitante responde al *OK* 200 con un *ACK*. Si la otra parte no acepta el cambio, envía una respuesta de error tal como un *Not Acceptable Here* 488, el cual también recibe un *ACK*. Sin embargo, el fracaso de un re-*INVITE* no causa que la llamada existente sea suspendida, la sesión continua utilizando las características previamente negociadas.

Al final de la llamada, Bob desconecta (cuelga) primero y genera un mensaje *BYE*. El mensaje *BYE* es encaminado directamente al *softphone* de Alicia, nuevamente eludiendo a los *proxies*. Alicia confirma el mensaje *BYE* con una respuesta *OK* 200, el cual termina la sesión y la transacción *BYE*. No es enviado un *ACK*.

En algunos casos, puede ser útil que los *proxies* se encuentren en el camino de la señalización SIP y así puedan ver todos los mensajes que generan las terminales durante la duración de la sesión. Por ejemplo, si el servidor *proxy* `ipn.mx` deseara permanecer en la ruta de los mensajes SIP más allá del *INVITE* inicial, podría agregar al *INVITE* un campo de ruta requerido en el encabezado llamado *Record-Route* que contiene una URI de la dirección IP o nombre del *proxy*. Esta información podría ser recibida por el teléfono SIP de Bob y el *softphone* de Alicia y ser almacenada durante la duración del diálogo. El servidor *proxy* `ipn.mx` podría entonces recibir y *proxear* el *ACK*, *BYE* y el *OK* 200 de *BYE*. Cada *proxy* puede decidir independientemente el recibir mensajes subsecuentes y aquellos mensajes pasarán a través de todos los *proxies* que eligieron recibirlos.

Los registros son otras operaciones comunes en SIP. Un registro es una manera en la que el servidor `ipn.mx` puede aprender la localización actual de Bob. Sobre la inicialización y en intervalos periódicos, el teléfono SIP de Bob envía mensajes *REGISTER* a un servidor en el dominio `ipn.mx` conocido como un "registrador SIP". Los mensajes *REGISTER* asocian el SIP o SIPS URI con la máquina en la cual se encuentra actualmente ingresado. El registrador escribe esta asociación en una base de datos, llamada servicio de localización, que puede ser utilizada por el *proxy* en el dominio `ipn.mx`. Continuamente, un servidor registrador para un dominio es

co-localizado con el *proxy* para dicho dominio. Es importante conceptualizar que la distinción entre tipos de servidores SIP es lógica, más no física.

Bob no se encuentra limitado a registrarse solamente desde un dispositivo. Por ejemplo, tanto el teléfono SIP en casa como el de la oficina pueden enviar registros. Dicha información es almacenada conjuntamente en el servicio de localización y permite a un *proxy* ejecutar varios tipos de búsquedas para localizar a Bob. Similarmente, más de un usuario puede estar registrado en un dispositivo simple al mismo tiempo.

El servicio de localización es solamente un concepto abstracto. Contiene generalmente información que permite a un *proxy* el agregar una URI y recibir un conjunto de cero o más URIs que dictaminan al *proxy* a donde enviar la petición. Los registros son una manera de crear esta información, pero no la única. Funciones arbitrarias de mapeo puede estar configuradas al albedrío del administrador.

Finalmente, es importante hacer notar que en SIP el registro es utilizado para encaminar peticiones entrantes SIP y no tiene un rol de autorizar peticiones salientes. La autorización y autenticación en SIP son manejadas tanto en una base de petición-por-petición con un mecanismo reto/respuesta, o al utilizar un esquema de capa inferior.

2.1.3 Estructura del protocolo

SIP se encuentra estructurado como un protocolo en capas, lo que significa que su comportamiento está descrito en términos de un conjunto de etapas de procesamiento independientes que tienen un acoplamiento débil entre cada etapa.

No todos los elementos especificados en el protocolo contienen todas las capas. Aún más, los elementos especificados por SIP son elementos lógicos, más no físicos. Una realización física puede elegir actuar como diferentes elementos lógicos, quizás en una base de transacción-por-transacción.

La capa menor de SIP es su sintaxis y codificación. Su codificación es especificada al utilizar una gramática aumentada *Backus-Naur Form*, en el RFC de SIP [28] se encuentra en la sección 25.

La segunda capa es la capa de transporte. Define como un cliente envía peticiones y recibe respuestas y como un servidor recibe peticiones y como envía respuestas a través de la red. Todos los elementos SIP contienen una capa de transporte.

La tercer capa es la capa de transacción. Las transacciones son el componente fundamental de SIP. Una transacción es una petición enviada por un cliente de (utilizando la capa de trasporte) hacia un servidor de transacciones, conjuntamente con todas las respuestas que la petición envió y que regresa del servidor de transacciones hacia el cliente.

La capa de transacción maneja las retransmisiones de la capa de aplicación, cotejando las respuestas de las peticiones y manejando el vencimiento en la capa de aplicación. Cualquier tarea que un Agente de Usuario Cliente (AUC) acomete, toma lugar al utilizar una serie de transacciones. Los AU tienen una capa de transacción,

tanto como *proxies* que lleven el estado completo. La capa de transacción tiene un componente cliente (referido como un cliente de transacción) y un componente servidor (referido como un servidor de transacción), cada uno de ellos se encuentra representado por una máquina finita de estados que es construida para procesar una petición en particular.

La capa superior de la capa de transacción es llamada *transacción usuario*. Cada entidad de SIP, excepto el *proxy* sin estados, es una transacción usuario. Cuando una transacción usuario desea enviar una petición, crea una instancia de transacción cliente y le pasa la petición junto con la dirección IP destino, puerto y transporte para el cual envía la petición. Una transacción usuario que crea una transacción cliente puede incluso cancerlarla. Cuando un cliente cancela una transacción, hace petición al servidor de detener el procesamiento futuro y revertir al estado que existía anteriormente a la transacción. Esto es realizado con una petición *CANCEL*, el cual constituye su propia transacción, pero referencía a la transacción a ser cancelada.

Los elementos SIP, los cuales son, AUCs y Agentes de Usuario Servidor (AUS), *proxies* con y sin estado y registradores, contienen un núcleo que los distingue entre sí. Los núcleos, excepto para el *proxy* sin estados, son transacciones usuario. Mientras el comportamiento del núcleo de AUC y AUS depende del método, existen algunas reglas comunes para todos los métodos. Para un AUC, dichas reglas gobiernan la construcción de una petición; para Agente de Usuario Servidor (AUS), ellos gobiernan el procesamiento de una petición y generan una respuesta. Mientras los registros juegan un importante papel en SIP, a un AUS, que maneja un *REGISTER*, le es dado un nombre especial de registrador.

Otras peticiones son enviadas dentro de un diálogo. Un diálogo es una relación SIP punto-a-punto entre dos AU que persisten por algún tiempo. El diálogo facilita la secuencia de mensajes y un encaminamiento propicio de peticiones entre los AU. El método *INVITE* es la única manera definida en esta especificación para establecer un diálogo. Cuando un AUC envía una petición que se encuentra dentro del contexto de un diálogo, las reglas comunes de AUS son seguidas pero también las reglas para peticiones de un semi-diálogo.

El método más importante en SIP es el método *INVITE*, el cual es utilizado para establecer una sesión entre participantes. Una sesión es una colección de participantes y flujos de media entre ellos, para propósitos de comunicación.

2.1.4 Peticiones

Las peticiones SIP se distinguen al tener una petición como componente de la primer línea. Una línea de petición contiene un nombre de método, una petición URI y la versión del protocolo separado por un carácter de espacio simple.

método en esta especificación se definen seis métodos: *REGISTER* para registrar información de sesión, *BYE* para terminar sesiones y *OPTIONS* para consultar

servidores acerca de sus capacidades. Extensiones SIP documentadas en RFCs, podrían definir métodos adicionales.

petición URI es un SIP o SIPS URI (o un URI genérico), indica el usuario o servicio al cual esta petición es dirigida.

versión SIP tanto para los mensajes de petición o respuesta, incluyen la versión de SIP en uso.

2.1.5 Respuestas

Las respuestas SIP se distinguen de las peticiones, al tener una línea de estado como su primer línea. Una línea de estado consiste en la versión del protocolo seguida por un *código de estado* numérico y su frase textual asociada, con cada elemento separado por un carácter de espacio simple.

El código de estado es un entero de tres dígitos, que indica el resultado de un intento para entender y satisfacer la petición. La frase textual asociada tiene como fin brindar una descripción textual corta del código de estado. El código de estado es utilizado por el autómata, mientras la frase es intencionada para el usuario humano.

El primer dígito del código de estado define la clase de respuesta. Los dos siguientes dígitos del código de estado no tienen algún tipo de categorización. Por tal motivo, cualquier respuesta con un código de estado entre 100 y 199 es referido como una "respuesta 1xx", cualquier respuesta con un código de estado entre 200 y 299 como una "respuesta 2xx" y así sucesivamente. SIP/2.4 permite seis valores para el primer dígito:

1xx provisional - petición recibida, el proceso de la petición se encuentra en proceso,

2xx satisfactorio - la acción fue exitosamente recibida, entendida y aceptada,

3xx redirección - acciones posteriores necesitan ser tomadas a fin de completar la petición,

4xx error de cliente - la petición contiene una mala sintaxis o no puede ser satisfecha en el servidor,

5xx error de servidor - el servidor falló en satisfacer una petición aparentemente válida,

6xx falla - la petición no puede ser realizada en el servidor.

En la sección 21 del RFC de SIP se definen estas clases y se describen los códigos individuales.

2.2 Protocolo de Tiempo Real

El Protocolo de Tiempo Real (RTP) RFC 3550 [31] proveé funciones de transporte de red principio a fin, adecuadas para que las aplicaciones puedan transmitir datos de tiempo real, tales como audio, video o datos de simulación, a través de servicios de red como multidifusión o de envío a un único destinatario. El transporte de datos es ampliado con un protocolo de control, el Protocolo de Control de Tiempo Real (RTCP), que permite el monitoreo de los datos entregados de una manera escalable en redes de multidifusión, además de proveer un control mínimo y una funcionalidad de identificación.

Los servicios provistos por RTP incluyen una identificación del tipo de carga[4], una numeración secuencial, marcas de tiempo[5] y un monitoreo de entrega. Se utiliza RTP sobre el Protocolo de Datagrama de Usuario (UDP) para hacer uso del multiplexaje y servicios de verificación[6].

RTP no se encarga de la reservación de recursos y no garantiza Calidad de Servicio para servicios de tiempo real. RTP no garantiza la entrega o previene una entrega fuera de orden, tampoco asume que la red es confiable y que entregue los paquetes en secuencia. Los números de secuencia incluidos en RTP permiten al receptor reconstruir la secuencia de paquetes del emisor.

Una especificación completa de RTP para una aplicación en particular requerirá de uno o más documentos adicionales, por ejemplo:

- Un documento de especificación de perfil, el cual define un conjunto de códigos de tipos de carga y su mapeo a formatos de tipos de carga. Normalmente una aplicación operará bajo un solo perfil. Un perfil para datos de audio y video puede ser encontrado en el RFC 3551 [30].

- Documentos de especificación del formato de la carga, las cuales definen como una carga en particular, tal como una codificación de audio o video, debe ser transportada con RTP.

2.2.1 Conferencia de audio en multidifusión simple

RTP utiliza dos puertos de conexión: un puerto es utilizado para los datos de audio, el otro es utilizado para los paquetes de control (paquetes RTCP). La dirección IP e información de puertos es distribuida a los participantes intencionados. Los detalles exactos del mecanismo de distribución se encuentran fuera del ámbito de RTP.

La aplicación de conferencia de audio utilizada por cada participante en la conferencia, envía los datos de audio en pequeños trozos de, por ejemplo, una duración de 20 milisegundos (ms). Cada trozo de datos de audio es precedido por un encabezado

[4]En inglés *payload type*.
[5]En inglés *timestamp*.
[6]En inglés *check sum*.

RTP; el encabezado RTP y los datos se encuentran contenidos en turno en un paquete UDP. El encabezado RTP indica que tipo de codificación de audio (tal como PCM, ADPCM o LPC) es contenido en cada paquete de tal modo que los transmisores puedan cambiar la codificación durante una conferencia.

La Internet, como otras redes de paquetes, ocasionalmente pierden, reordenan paquetes y los retrasan por una cantidad variable de tiempo. Para hacer frente a estos deterioros, el encabezado RTP contiene información de tiempo y una secuencia numérica que permite al receptor reconstruir el cronometraje producido por la fuente, los trozos de audio son continuamente reproducidos en la bocina cada 20 ms. Esta reconstrucción del tiempo es realizada de forma separada para cada fuente de paquetes RTP en la conferencia. La secuencia numérica puede incluso ser utilizada por el receptor para estimar cuantos paquetes han sido perdidos.

2.2.2 Definiciones RTP

Carga RTP: son los datos transportados por RTP en un paquete, por ejemplo las muestras de audio o datos de video comprimido. El formato de carga y la interpretación se encuentran fuera del ámbito de RTP.

Paquete RTP: es un paquete de datos que consiste de un encabezado fijo RTP, una lista posiblemente vacia de fuentes colaboradoras y la carga de datos.

Paquete RTCP: es un paquete de control que consiste de un ecabezado fijo que en parte es similar al paquete RTP, seguido por elementos estructurados que varian dependiendo del tipo de paquete RTCP.

Puerto: es la abstracción que los protocolos de transporte utilizan para distinguir entre destinatarios múltiples dentro de una computadora dada. Protocolos TCP/IP identifican los puertos utilizando enteros positivos grandes.

Dirección transporte: es la combinación de una dirección de red y puerto que identifica un nivel de transporte final, por ejemplo una dirección IP y un puerto UDP.

Tipo de media RTP: es la colección de tipos de carga que pueden ser transportados dentro de una simple sesión RTP.

Sesión RTP: es una asociación entre un conjunto de participantes comunicándose por RTP. Un participante podría estar involucrado en múltiples sesiones RTP con sus propios paquetes RTCP, a menos que el decodificador multiplexe múltiples medias en un único flujo de datos.

Sincronización fuente (SSRC): es la fuente de un flujo de paquetes RTP, denotado por un identificador númerico SSRC incluido en el encabezado RTP y no

es dependiente de la dirección de red. El identificador SSRC es un valor elegido de forma aleatoria, siendo globalmente único dentro de una sesión RTP en particular.

Contribuidor fuente (CSRC): es un flujo fuente de paquetes RTP que ha contribuido a la combinación del flujo producido por un mezclador RTP.

Mezclador: es un sistema intermediario que recibe paquetes RTP de una o más fuentes, si es necesario cambia el formato de datos, combina los paquetes de alguna manera y por último reenvía un nuevo paquete RTP.

medios no RTP: son protocolos y mecanismos que pueden ser utilizados en conjunto con RTP para proveer un servicio que pueda ser consumible. Ejemplo de tales protocolos son el Protocolo de Inicio de Sesiones (SIP) RFC 3261 [28] y aplicaciones que utilizen el Protocolo de Descripción de Sesión (SDP) RFC 2327 [13].

2.2.3 Protocolo de transferencia de datos RTP

Campos de encabezado fijos RTP

El encabezado RTP se muestra en la figura 2.2.

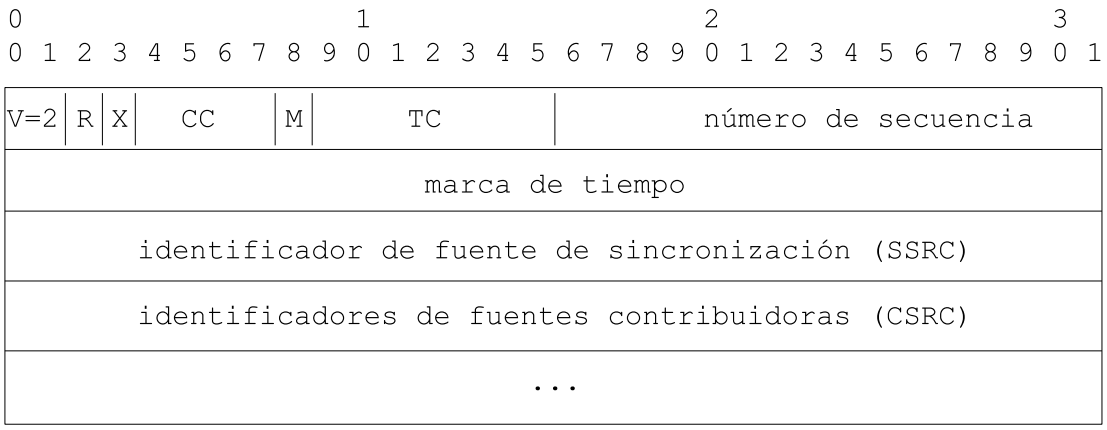

Figura 2.2: Encabezado del paquete RTP

Los primeros doce octetos se encuentran presentes en cada paquete RTP, mientras que la lista de identificadores CSRC está presente sólo cuando es insertada por un mezclador. Los campos tienen el siguiente significado:

versión (V): 2 bits. Este campo identifica la versión de RTP.

relleno (R): 1 bit. Si el campo de relleno es establecido, el paquete contiene uno o más octetos adicionales al final y no forman parte de la carga. El octeto final del relleno contiene un conteo de cuantos octetos de relleno deberán ser ignorados, incluyendo este octeto.

extensión (X): 1 bit. Si el campo de extensión es establecido, el encabezado fijo deberá estar seguido por exactamente un encabezado de extensión.

conteo CSRC (CC): 4 bits. El conteo CSRC contiene el número de idetificadores CSRC que sigue el encabezado fijo.

marcador (M): 1 bit. La interpretación del marcador es definido por un perfil.

tipo de carga (TC): 7 bits. Este campo identifica el formato de la carga RTP y su interpretación es determinada por la aplicación. Un conjunto de mapeos por omisión para audio y video se especifican en el RFC 3551 [30].

número de secuencia: 16 bits. El número de secuencia se incrementa en uno por cada paquete de dato RTP enviado, puede ser utilizado por el receptor para detectar la pérdida de paquetes y para restablecer la secuencia de paquetes. El valor inicial del número de secuencia podría ser aleatorio, para hacer más difícil un ataque de cifrado de tipo texto en claro conocido.

marca de tiempo: 32 bits. La marca de tiempo refleja el instante muestreado del primer octeto en el paquete de datos RTP. El instante de muestreo deberá estar derivado de un reloj que se incremente de manera monótona y de forma lineal en tiempo para permitir cálculos de sincronización y de *jitter*[7]. La frecuencia del reloj es dependiente del formato de los datos transportados como carga y se especifica de manera estática en el perfil o en la especificación del formato de carga que define el formato, de otra manera se podría especificar de manera dinámica para formatos de carga definidos a través de medios no RTP. Si los paquetes RTP son generados periódicamente, el instante de muestreo nominal es determinado por el reloj de muestreo y será el utilizado, no es una lectura del reloj del sistema. Como ejemplo, para un audio de tasa fija, la marca de tiempo del reloj segúramente se incrementará en uno por cada periodo de muestreo. Si una aplicación de audio lee bloques que cubren 160 periodos de muestreo del dispositivo de entrada, la marca de tiempo podría ser incrementada en 160 por cada bloque, sin importar cuando el bloque sea transmitido en un paquete o sea descartado como un silencio.

SSRC: 32 bits. El campo SSRC identifica la fuente de sincronización. Este identificador debería ser elegido de manera aleatoria, con la intención de no darse el caso que dos fuentes de sincronización dentro de una misma sesión RTP tengan el mismo identificador SSRC.

[7] El *jitter* es la variación del retraso.

CSRC: 32 bits. La lista CSRC identifica las fuentes contribuidoras para la carga contenida en el paquete. El número de identificadores está dado por el campo CC. Los identificadores CSRC son insertados por mezcladores.

2.2.4 Protocolo de control RTP

El Protocolo de Control de Tiempo Real (RTCP) está basado en la transminsión periódica de paquetes de control a todos los participantes en la sesión, utilizando el mismo mecanismo de distribución como el de paquete de datos. RTCP lleva a cabo cuatro funciones:

1. La función primaria es de proveer retroalimentación de la calidad de distribución de los datos. Esto es una parte integral del papel de RTP como un protocolo de transporte y es relacionado al flujo y a las funciones de control de congestión de otros protocolos de transporte. La retroalimentación puede ser directamente utilizada para el control de codificadores adaptativos, pero experimentos con multidifusión IP han mostrado que es crítico obtener retroalimentación de los receptores para diagnosticar fallas en la distribución. Esta retroalimentación es realizada por los reportes RTCP del emisor y receptor.

2. RTCP conlleva un identificador de nivel de transporte persistente por una fuente RTP llamada CNAME o de nombre canónico. Dado que el identificador SSRC puede cambiar si un conflicto es encontrado o un programa es reiniciado, los receptores requieren el Nombre canónico (CNAME) para mantenerse al tanto de cada participante.

3. Las primeras dos funciones requieren que todos los participantes envíen paquetes RTCP, por lo tanto la tasa deberá ser controlada para que RTP sea escalado a un gran número de participantes. Al obtener que cada participante envíe sus paquetes de control hacia los demás, cada uno puede observar de manera independiente el número de participantes. Este número es utilizado para calcular la tasa en que los paquetes son enviados.

4. Una función opcional es el convenir una sesión de control de información mínima, por ejemplo, la identificación de participantes para ser desplegados en la interfaz de usuario.

Las tres primeras funciones podrían ser utilizadas en todos los entornos, pero particualmente en el entorno de multidifusión IP.

Si la recepción de reportes de cada participante fuera enviada con una tasa constante, el tráfico de control crecerá de forma lineal dado el número de participantes. Por tanto, la tasa de envío de paquetes RTP deberá estar reducida a calcular de manera dinámica el intervalo entre la transmision de paquetes RTCP. El valor recomendado para un intervalo fíjo mínimo es de 5 segundos.

La descripción de los campos de un paquete RTCP no serán descritos, ya que la inclusión de RTCP en este texto es solamente de manera introductoria, con el fin de definir algunos de los parámetros cuantitativos que son de utilidad para la medición objetiva de la calidad en una comunicación VoIP. A continuación se explican algunos parámetros que calcula RTCP.

número acumulado de paquetes perdidos: definido como el número de paquetes esperados menos el número de paquetes recibidos, dónde el número de paquetes recibidos incluye cualquiera que sea retrasado o duplicado. De manera que, paquetes que llegan tarde no son contados como paquetes perdidos, y el número de paquetes perdidos puede ser negativo si hay duplicados. El número de paquetes esperados se define como el número de secuencia mayor recibido de los paquetes de datos RTP, menos el número de secuencia inicial recibido.

***jitter* de entre-llegada[8]:** es un estimado de la varianza estadística de tiempo de entre-llegada del paquete de datos RTP, medido en unidades de marcas de tiempo y expresado como un entero sin signo. El *jitter* de entre-llegada J está definido como la desviación media (de valor absoluto suavizado) de la **diferencia** D del espaciado de paquetes en el receptor comparado al del emisor para un par de paquetes. Como es mostrado en la ecuación 2.1, esto es equivalente a la diferencia en el "tiempo transitorio relativo" por los dos paquetes; el tiempo transitorio relativo es la diferencia entre una marca de tiempo de un paquete RTP y el reloj del receptor en el tiempo de llegada, medidos en las mismas unidades.

Si S_i es el marcado de tiempo RTP del paquete i, y R_i es el tiempo de llegada en unidades de marcado de tiempo RTP para el paquete i; entonces para dos paquetes i y j, D deberá ser expresado como:

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \tag{2.1}$$

El *jitter* de entre-llegada debería ser calculado continuamente, en la misma proporción en que cada paquete de datos i sea recibido de una fuente, utilizando la diferencia D entre el paquete (i) y el paquete previo $(i-1)$ en orden de llegada (no necesariamente en secuencia), conforme a la fórmula:

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16 \tag{2.2}$$

[8]En inglés *interarrival jitter*.

Cuando un reporte de recepción es emitido, el valor actual de J es tomado. El cálculo del *jitter* debe estar conforme a la fórmula 2.2, a fin de permitir monitores de perfiles independientes, para hacer interpretaciones válidas de reportes provenientes de implementaciones diferentes. Este algoritmo es un estimador de primer orden óptimo y el parámetro de ganancia 1/16 da una buena tasa de reducción de ruido mientras se mantiene una tasa razonable de convergencia.

2.3 Codificadores de voz

Los codificadores son generalmente entendidos como varios modelos matemáticos utilizados para codificar (y comprimir) digitalmente información de audio análoga. Muchos de esos modelos toman en cuenta la habilidad del cerebro humano para dar una impresión de información incompleta [22].

Originalmente, el término codificador referido como COdificador/DECodificador[9]: un dispositivo que hace conversiones análogas y digitales. Ahora, el término parece estar más relacionado a COmpresión/DEScompresión.

Los codificadores de voz son diseñados para estándares de la Unión Internacional de Telecomunicaciones (ITU), los cuales especifican cómo los segmentos de una señal de voz análoga, serán codificados en flujos digitales de datos. El diseño de un codificador en particular, utilizado para digitalizar señales de voz que serán transportadas en una red de paquetes conmutados, determina tanto el número mínimo de bytes que pueden ser razonablemente incluidos en un paquete de voz como la cantidad de bits que deberán ser producidos en un paquete para transmitir una señal digitalizada de voz. Las diferencias en las técnicas para codificar, crean diferencias substanciales tanto en la duración mínima del segmento de voz, que esta siendo muestreado y la cantidad de datos transmitidos para la sustentación de la regeneración de las señales analógicas. Las características de un codificador como el tiempo que dura un segmento de voz, el tamaño del segmento de voz codificado y la tasa de bits necesario por el codificador, afectan directamente dos características de las señales de voz escuchadas por los usuarios a través de una conexión de voz digitalizada: retrasos y fidelidad de señal [14].

Retrasos: para modelar los segmentos de voz con la duración mostrada en la tabla 2.1, el codificador deberá tener el segmento completo y posiblemente de otros disponibles para procesarlos. Por ejemplo, el codificador G.723.1 deberá recibir y almacenar 37.5 ms de muestras de voz digital antes de poder efectuar la codificación con el número de bits mostrados. Por tanto la utilización del codec G.723.1 incrementa características de conexión, como eco por el retraso en el camino y retrasos conversionales de recorrido completo de almenos 67.5 ms (37.5 ms de tiempo de codificación y 30 ms de tiempo de decodificación) sobre la transmisión contínua de señales codificadas con el codificador G.711 [14].

[9]En inglés el término codificador es *codec*.

Codificador	Técnica de codificación	Duración del segmento de voz (ms)	Tamaño del segmento codificado (bits)	Tasa de transferencia (bits/s)
G.711	PCM	0.125	8	64,000
G.723.1	MP-MLQ	30	189	6,300
G.723.1	ACELP	30	158	5,300
G.729	CELP	10	80	8,00

Tabla 2.1: Comparación de características de codificadores de audio

Fidelidad de señal: aún cuando la transmisión digital sea perfecta, habrá invariablemente diferencias entre la señal de onda análoga inyectada y la extraída en el destino final. Para todos los codificadores estándar, las diferencias entre las señales de onda inyectadas y extraídas, dada una transmisión digital libre de errores, no se espera que sean suficientemente grandes para materialmente afectar la calidad de transmisión de voz, respecto a una inteligibilidad o un reconocimiento del hablante. Las distorciones en una señal de onda, producidas en una transferencia digital con un codificador en particular pueden, ser suficientemente grandes para producir una degradación notable. En todos los codificadores los errores ocurridos en la transmisión de bits pueden deformar la aproximación digital de la señal de onda inyectada, en términos que pueden ser percibidos por usuarios como distorsión de habla.

Otras características opcionales, que dependen de la forma en que el codificador fue implementado, pueden afectar directamente la percepción del usuario de la incidencia o austeridad del reconocimiento de la degradación. Éstas incluyen supresión de silencios, ruido confort y cancelación de pérdida de paquetes.

Supresión de silencios: con propósitos de minimizar la transferencia de datos innecesarios para transmitir señales de voz digitalizadas, los codificadores pueden ser configurados para monitorear la señal análoga inyectada, digitalizar y transmitir sólo lo que parece ser voz. Esta característica opcional en un codificador es referida como una Actividad de Compresión de Voz (VAC) o como una Detección de Actividad de Voz (VAD) y supresión de silencios. Esta supresión reduce la percepción de incidencia y austeridad de ruido. Por otro lado, la supresión de silencios tiene dos efectos negativos. La primera es que en señales de diálogos de bajo volumen, la VAD será lenta en detectar suaves inicios y términos de palabras o sílabas, produciendo algo que se conoce como recorte VAC, bajo los cuales los usuarios notan que sonidos esperados hacen falta en el diálogo recibido. El segundo efecto consiste en que la supresión de silencios produce una completa ausencia de señal en el final distante.

Ruido confort: una de las características de todas las conexiónes digitales de voz, es que no hay ruido absoluto en la línea cuando no se encuentra alguien hablando. Los usuarios comunmente experimentan niveles de bajo ruido cuando alguna de las partes de la conexión es análoga, esta *profunda invalidez* hace parecer que la línea ha muerto. Para evadir este problema, el decodificador puede ser programado para insertar una señal de ruido pseudo-aleatória de bajo nivel, cuando no exista señal recibida. Este ruido es conocido como ruido confort. La percepción de la calidad de la conexión estará, por tanto, afectada por la elección de insertar el ruido confort y de los procedimientos para lograrlo.

Cancelación de pérdida de paquetes: para algunos codificadores, que son utilizados en servicios de paquetes de voz conmutados, existe la posibilidad de que algunas tramas de señales de voz digitales no lleguen en el tiempo que son necesarias para regenerar el siguiente segmento de voz. Para crear una elasticidad en los efectos que se presentan, dada una falta de muestras de las señales de onda reconstruidas en el destino final, el codificador puede ser programado para llenar los huecos de los datos muestreados. Componentes comunes de este tipo de programación incluyen la repetición de la última trama recibida o la generación de un segmento de voz artificial consonante con tramas previas de los datos muestreados. Tales compensaciones de paquetes, pueden substancialmente reducir los efectos destructivos que causa la pérdida de paquetes en la percepción del usuario.

En la tabla 2.2 se listan algunos codificadores con su respectiva tasa de datos y si es necesario obtener una licencia para su uso.

Codificador	Tasa de datos (kbps)	¿Licencia requerida?
G.711	64 kbps	No
G.726	16, 24, 32, 40 kbps	No
G.729A	8 kbps	Si
GSM	13kbps	No
iLBC	13.3 kbps (frames de 30 ms) o 15.2 kbps (frames de 20 ms)	No
Speex	Variable (entre 2.15 y 22.4 kpbs)	No

Tabla 2.2: Codificadores de audio

El codificador G.711

G.711 es el codificador fundamental en la PSTN. De hecho, si alguien refiere a la Modulación por Codificación de Pulsos (PCM) respecto a la red telefónica, se está permitido pensar en el codificador G.711. Dos métodos de compresión/expanción[10] son

[10]En inglés el término compresión/expanción es *companding*.

utilizados en este codificador, μlaw en Norte América y *alaw* en el resto del mundo. Cualquiera de los dos codificadores entregan una palabra de 8-bits transmitida $8,000$ veces por segundo. Es por tanto que se requiere transmitir $64,000$ bits por segundo. Este codificador es básico en los codificadores listados en la tabla 2.2. G.711 e impone una carga mínima (de casi cero) en el CPU [22].

En el RFC 3551 [30] se presentan a los codificadores PCMU y PCMA como codificadores de 8 bits por muestra, después de aplicar un escalado logarítmico. PCMU usa un escalado μlaw, PCMA usa un escalado *alaw*. Los modos de transferencia de audio de 56 kbps y 48 kbps no son aplicables en RTP, dado que PCMA y PCMU deberán estar siempre transmitidos en muestras de 8 bits. El tipo de carga para PCMU es 0 y para PCMA es 8, ambos codificadores son de audio, tienen un reloj de muestreo de $8,000$ Hz y soportan sólo un canal de audio. Cada paquete lleva 20 ms de audio y dado el reloj de muestreo se obitenen 160 muestras en este intervalo.

El encabezado de un paquete IP comprende un total de 160 bits (20 bytes). El Protocolo de Datagrama de Usuario (UDP) y el Protocolo de Tiempo Real (RTP) crean la necesidad de 8 y 12 bytes adicionales respectivamente de información en el encabezado, demandando un número total de 320 bits necesarios para todos los **encabezados del paquete de audio** [14].

2.4 Jitter buffer

En un servicio de voz, las muestras digitales deben ser presentadas al codificador, de manera tal que la siguiente muestra de un flujo se encuentre presente para ser procesada, al momento en que el decodificador ha terminado con el predecesor inmediato. Dicho requerimiento condiciona severamente la cantidad de *jitter* que puede ser tolerado en un servicio de paquetes conmutados, sin tener que desfasar las muestras. Cuando el *jitter* resulta en un tiempo de entre-llegada de paquetes, que cargan muestras consecutivas mayor que el tiempo requerido para reconstruir la señal de onda de una muestra, el decodificador no tiene mas opción que continuar funcionando sin la información de la siguiente muestra [14].

El efecto de *jitter*, en la tasa de paquetes descartados, implica que la incidencia de paquetes desechados medidos, para un servicio de voz de paquetes conmutados, será mucho mayor que el medido en el transporte de paquetes conmutados. Los efectos de *jitter* se manifestarán como un incremento en las tasas de tramas desechadas.

La generación de una señal análoga de voz de manera contínua en el destino final, que resulte menos susceptible a variaciones en el tiempo de arribo de los paquetes a través de la red, los codificadores utilizados en servicios de voz de paquetes conmutados proveen un número de segmentos digitalizados de voz en una *lista de espera* antes de comenzar la codificación. Esto tiene el efecto de poder incrementar la magnitud del tiempo de entre-llegada entre muestras que pueden ser toleradas, sin dejar espacios vacios de muestras de voz mientras el decodificador despeja la lista de espera.

El *buffer* que guarda los segmentos encolados es llamado *jitter buffer*. El empleo efectivo de un *jitter buffer* define la relación entre el *jitter* en un flujo digital de voz y las tasas de paquetes desechados; proporciona un balance entre probabilidades de tramas desechadas e incrementos en retrasos de transmisión, quedando definidos por el tamaño del *jitter buffer*. La magnitud de la diferencia en el retraso que puede ser tolerado, viene a ser un descriptor escencial de la calidad intrínseca que relega el *jitter* en el caso de un servicio de voz de paquetes conmutados.

2.5 Biblioteca PJ

En [52] se brindan bibliotecas de comunicación multimedia de código abierto, comprensibles, de alto rendimiento, portables, escritas en lenguaje C para construir aplicaciones VoIP embebidos/no-embebidos [52]. La versión 0.1 de PJSIP salió en el año 2003 y se prevé que para finales del 2008 se cuente con la versión 1.0 de la biblioteca PJ.

PJSIP es una pila SIP que soporta muchas extensiones/características SIP y cuenta con los siguientes beneficios:

Extremadamente Portable - Escribir aplicaciones sólo una vez y se ejecutará en varias plataformas (Windows, Windows Mobile, Linux, MacOS X, RTEMS, Symbian OS, etc.)

Tamaño pequeño - Menos de 150 KB con características SIP completas, ideal para dispositivos embebidos donde el espacio es costoso y también para aplicaciones generales.

Alto rendimiento - menor consumo en la Unidad de Procesamiento Central (UPC) significa más transacciones/llamadas SIP pueden ser manejadas por segundo.

Documentación - Se cuenta con cientos de páginas de documentación y foros de discusión.

Varias extensiones/características tales como múltiples usos en un diálogo, *framework* de subscripción de eventos, presencia, mensajería instantánea, transferencia de llamadas, etc. han sido implementadas en la biblioteca.

PJMEDIA es otra biblioteca complementaria para construir aplicaciones de AU SIP tales como *softphones*, que comparte casi las mismas características mencionadas anteriormente. La biblioteca dispone de codificadores de audio como PCM escalado *alaw* (PCMA), PCM escalado μlaw (PCMU), speex, iLBC entre otros.

PJNATH es una nueva biblioteca, disponible en el troncal *svn* del proyecto, que ayuda a las aplicaciones a que se encuentren en una red NAT. Implementa también las últimas especificaciones de STUN, TURN y ICE.

PJLIB-UTIL es una biblioteca auxiliar que provee soporte a PJMEDIA y PJSIP. Algunos de los componentes/funciones en esta biblioteca son: analizador XML de

tamaño pequeño, biblioteca STUN cliente, asíncrono/copias de resolución de DNS, funciones de digesto/cifrado, etc.

PJLIB es la única biblioteca básica para PJLIB-UTIL, PJMEDIA y PJSIP. PJLIB provee una abstracción completa no sólo de las características del sistema operativo, sino también está diseñado para abstraer LIBC y provee algunas estructuras de datos útiles.

Los lenguajes soportados son C, C++, Python y ActiveX.

PJSUA API es una Interfaz de Programación de Aplicaciones (API) de alto nivel disponible para C/C++ y Python para construir aplicaciones de AU de multimedia SIP. Incluye la señalización y funcionalidad de la media para facilitar el uso de la API de llamadas. Provee manejo de cuenta, manejo de grupos, presencia, mensajería instantánea e incluye características de multimedia tales como conferencia, flujo de archivos, grabador de voz, etc.

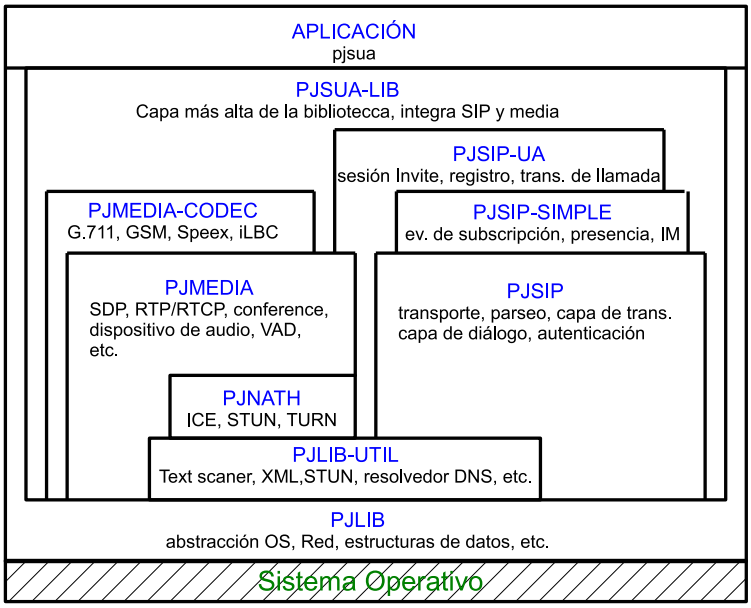

Figura 2.3: Arquitectura de la biblioteca PJ

La figura 2.3 muestra la arquitectura de la biblioteca PJ. En ella se puede observar la dependencia entre las diferentes bibliotecas, en donde PJLIB es la que logra la abstracción del sistema operativo y la API PJSUA integra la funcionalidad de todas las bibliotecas.

2.6 Asterisk

Asterisk es un software libre que brinda una plataforma de convergencia telefónica, la cual está diseñada para ejecutarse en Linux. Asterisk provee un Intercambio Privado de Ramificación (PBX) que puede ser visto como un servicio ofrecido por una empresa de telecomunicaciones, en la que una cantidad de n de líneas o números son agrupados en un único número que se publica y al cual se puede llamar. La empresa proveedora se encarga de distribuir las llamadas entrantes por las líneas disponibles contratadas por el cliente (empresa privada, ONG, ORG, Gobierno, escuelas, etc).

Aplicaciones como correo de voz, conferencias, llamadas en espera y agentes, música de espera y aparcamiento de llamadas son ventajas estandarizadas incluidas en el software. Además, Asterisk puede integrarse con otras tecnologías comerciales [22].

Asterisk provee las siguientes cualidades:

- Es una plataforma en la que converge la telefonía y es software libre, diseñado principalmente para utilizarse sobre Linux.
- Es un puente entre telefonía tradicional PSTN y de VoIP.
- Es un canal de comunicación basado en SIP para la señalización VoIP.
- Es un proyecto en desarrollo y cuenta con actualizaciones constantes.

Aún cuando Asterisk es un software gratuito y completamente disponible en Internet, no es fácil administrarlo, configurarlo y darle mantenimiento [18].

Un módulo principal de Asterisk es el proyecto *Zapata*, que consiste en una tarjeta PCI que intercomunica un circuito de telefonía con una computadora personal (llamadas tarjetas TDM), dejando el Procesamiento Digital de Señales ser manejada en la UPC por software.

Zaptel es el software que se comunicará con tarjetas TDM instaladas en la PC. Estas tarjetas dependiendo de sus módulos pueden conectarse a teléfonos de la PSTN y a la misma línea telefónica análoga [49].

La interfaz Zaptel es un módulo del Kernel que presenta una capa de abstracción entre los manejadores del hardware y del módulo *Zapata* en Asterisk. Este es el concepto que permite a los manejadores del dispositivo ser modificados sin realizar cambios al programa fuente de Asterisk. El manejador del dispositivo es utilizado para comunicarse con el hardware directamente e intercambiar información entre Zaptel y el hardware [22].

Las tarjetas de la serie TDM utilizan el puerto PCI y soporta interfaces Estación de Intercambio Foráneo (FXS) y Oficina de Intercambio Foráneo (FXO) que conecta teléfonos análogos y líneas análogas respectivamente a través de una PC. El productor de esta tarjeta es la empresa *Digium* y ofrece el manual de instalación en su sitio Web [45].

El plan de marcado es la parte esencial del sistema Asterisk, dado que define como manejar las llamadas entrantes y salientes. El plan de marcado consiste en una lista de instrucciones o pasos que Asterisk deberá seguir. El plan de marcado se encuentra dividido en secciones llamados "contextos". Los contextos son grupos nombrados que tienen extensiones, los cuales sirven para diversos propósitos. Los contextos mantienen diferentes partes del plan de marcado para interactuar unos con otros. Una extensión que se encuentre definida en un contexto está completamente aislada de las extensiones de cualquier otro contexto, amenos que se especifique una interacción permitida, obśervese el ejemplo siguiente:

```
;;Éste es un "plan de marcado"

[entrantes] ;; Definición de un contexto con nombre "entrantes"

;; Dentro de un contexto existen extensiones
exten => 100,1,Dial(SIP/john)

...

;; Definir otros contextos
[otrocontexto]
```

En el mundo de las telecomunicaciones, la palabra extensión usualmente refiere a un identificador numérico dado a una línea que hace sonar a un teléfono en particular. En Asterisk una extensión es mucho más poderosa, tanto que define una serie de pasos únicos (cada paso conteniendo una aplicación) que Asterisk llevará a cabo para establecer la llamada. Dentro de cada contexto, se pueden definir tantas extensiones como se requiera. Cuando una extensión en particular es disparada (por una llamada entrante o por dígitos marcados en un canal), Asterisk seguirá los pasos definidos por la extensión. Son entonces las extensiones las que especifican que pasa con las llamadas en tanto que ellas marcan el camino a seguir sobre el plan de marcado. Aunque las extensiones pueden ser utilizadas para especificar extensiones de teléfonos de una forma tradicional (por ejemplo, extensión 153 causará que suene el teléfono SIP de un usuario), en un plan de marcado de Asterisk, se pueden utilizar con una funcionalidad mayor. Un ejemplo es el siguiente: el usuario que marque la extensión 123 como primer paso le será notificado que está entrando la llamada (con la aplicación *Ring()*), el paso siguiente es contestar la llamada (se lleva a cabo la sesión de media), en el paso 3 en la sesión de media el audio que es enviado es regresado (por la aplicación *Echo()*), cuando el usuario decide colgar el plan de marcado en su fase 4 también cuelga, obsérvese la siguiente extensión:

```
;; Una extension más elaborada
exten => 123,1, Ring()   ;; Ring() es una aplicación
```

```
exten => 123,2, Answer()
exten => 123,3, Echo()
exten => 123,4, Hangup()
```

Cada extensión puede tener múltiples pasos llamados prioridades (una prioridad dicta una jerarquía en los pasos). Cada prioridad está numerada secuencialmente, iniciando desde 1 (de mayor prioridad) y ejecuta una aplicación en específico. La prioridad n significa tomar la prioridad anterior y sumar 1. Mediante la prioridad n resultan fáciles los cambios al plan de marcado, tanto que no se tienen que re-enumerar todos los pasos, por ejemplo, de la extensión anterior puede escribirse de la siguiente manera:

```
exten => 123,1, Ring()   ;; Ring() es una aplicación
exten => 123,n, Answer()
exten => 123,n, Echo()
exten => 123,n, Hangup()
```

Las aplicaciones son la funcionalidad núcleo del plan de marcado. Generalmente las aplicaciones operan sobre el canal, mientras que las funciones, meramente retornan valores que pueden ser utilizados por las aplicaciones.

Asterisk cuenta con una Interfaz de Puerta de Enlace (AGI), que provee una interfaz estándar, por la cual programas externos pueden controlar el plan de marcado de Asterisk. Usualmente los *scripts* AGI son utilizados empleando una lógica avanzada, para comunicarse con una base de datos relacional y acceder otros recursos externos. Esta AGI da una funcionalidad que va más allá que de un sistema Respuesta de Voz Interactiva (IVR) [22].

IVR es un sistema telefónico capaz de recibir una llamada e interactuar con el usuario a través de grabaciones de voz. Es un sistema de respuesta interactiva, orientado a entregar y/o capturar información automatizada a través del teléfono, permitiendo el acceso a los servicios de información y operaciones autorizadas [41].

Capítulo 3

Calidad de Servicio

La Calidad de Servicio (QoS) resulta muy importante cuando se tienen flujos de tráfico que deben cumplir con ciertos requerimientos. En el capítulo se estudia la QoS en entornos alámbricos e inalámbricos en una red IP. El modelo elegido para brindar QoS es mediante la diferenciación de servicios.

Se presenta la certificación Multimedia Inalámbrica (WMM) que implementa parcialmente el estándar IEEE 802.11e. El estándar IEEE 802.11e brinda mecanismos de QoS no incluidos en el estándar 802.11.

El nivel de servicio, por omisión, de IP es conocido como "de mejor esfuerzo" significando que la red hará su mejor esfuerzo para asegurar los requerimientos de ancho de banda, pero sin alguna garantía. El resultado es que servicios móviles IP de tiempo real funcionan de una manera pobre o incluso no funcionan, dependiendo principalmente del ancho de banda disponible y de la congestión en la red.

Los mecanismos QoS fueron desarrollados con el fin de superar esos problemas y proveer algún tipo de nivel de garantía de transmisión en lugar del "mejor esfuerzo". QoS asegura que elementos críticos de transmisión IP, tales como la tasa de transmisión, retardo y la tasa de errores puedan ser medidos, mejorados y garantizados de antemano [1].

3.1 Conceptos fundamentales de QoS

Desde el punto de vista de la red, QoS es la habilidad de un elemento de la red (por ejemplo una aplicación, equipo de cómputo o un encaminador) de poder brindar un nivel de garantía a los requerimientos de servicio y tráfico, para que éstos puedan ser satisfechos. QoS administra el ancho de banda de acuerdo a las demandas de la aplicación y a la configuración de la red [21].

La IETF considera a QoS como la habilidad de segmentar el tráfico o diferenciarlo entre tipos de tráfico, para que la red atienda a cierto tipo de flujos de tráfico de manera diferente entre los demás.

El término QoS es utilizado con diferentes significados, desde la percepción de los usuarios por el servicio, hasta el conjunto de parámetros de la conexión necesarios para lograr un servicio de calidad particular. Se identifican tres tipos de QoS: intrínsecos, percibido y evaluado. La QoS intrínseca es directamente proveída por la red misma y puede ser descrita en términos de parámetros objetivos tales como pérdida y retraso de paquetes. La QoS percibida (P-QoS) es la calidad percibida por los usuarios; depende directamente del desempeño de la red pero es medida por una opinión promedio de los usuarios. Los métodos de Puntaje de Opinión Media (MOS) son muy utilizados para realizar la medición de la calidad: los usuarios asignan una evaluación MOS a la aplicación que ellos están calificando.

3.1.1 Aplicaciones QoS

Todas las aplicaciones que requieren un nivel específico de certidumbre en la red necesitan QoS. Esto no da una idea acerca de la cantidad de aplicaciones que requieren QoS. Algunas de ellas se listan a continuación: servicios básicos de transferencia de información, accesos garantizados a bases de datos para recibir información, telemedicina, control a distancia, operaciones financieras y bancarias, adquisiciones y entregas, telefonía, videoconferencias y aplicasiones de emergencias y seguridad.

Encontrándose diferentes características, las aplicaciones antes mencionadas ameritan un grado específico de servicio, definido en la capa de aplicación. La ITU-T sugiere una definición de clases QoS (para el mundo IP) que se resume en la tabla 3.1 [21].

Clase QoS	Características
0	Tiempo real, jitter sensible y altamente interactivo
1	Tiempo real, jitter sensible e interactivo
2	Datos de transacción y altamente interactivo
3	Datos de transacción e interactivo
4	Sólo de baja pérdida (transacciones cortas, datos en general y flujo de video)
5	Aplicaciones tradicionales de redes IP

Tabla 3.1: Clases QoS ITU-T

El proyecto ETSI TIPHON propone una definición alternativa de clases QoS, reportado en la tabla 3.2 [21].

Clase QoS	Componentes	Características QoS
Conversaciones de tiempo real (telefonía , teleconferencia, videofonia y videoconferencia)	Discurso, audio, video, multimedia	Retraso y variación de retraso sensible, tolerancia limitada a pérdida y errores, tasa de bit constante y variable
Flujo de tiempo real (*broadcast* de audio y video, supervisión, gráficos	Audio, video multimedia	Tolerante al retraso, sensible a la variación del retraso, tolerancia limitada a pérdidas y errores, tasa de bit variable
Interacción cercana a tiempo real (navegación web)	Datos	Sensible al retraso, tolerante a la variación del retraso, sensible al error, tasa de bit variable
No de tiempo real *background* (correo electrónico y transferencia de archivos)	Datos	No sensible al retraso o a la variación del retraso, sensible al error, de mejor esfuerzo

Tabla 3.2: Clases QoS TIPHON

3.1.2 Métricas QoS

Concerniente al entorno IP, las métricas objetivas de QoS más utilizadas son las siguientes [21]:

- tasa de paquetes perdidos IP
- retraso en la transferencia de paquetes IP
- variación de retraso de paquetes IP
- tasa de errores de paquetes IP

El sesgo es otra métrica comúnmente considerada, la cual es el valor promedio de la diferencia de retrasos medido en paquetes que pertenecen a diferente media, por ejemplo, voz y video dentro de un servicio de conferencia. En tal caso, si un sesgo es prolongado, no existe una sincronización entre voz y video provocando un efecto de un mal doblaje.

3.1.3 QoS en el medio inalámbrico

Mecanismos y protocolos de QoS utilizados en ambientes alámbricos, usualmente no pueden ser directamente aplicados en ambientes inalámbricos o híbridos alámbri-

cos/inalámbricos, debido a las siguientes características de las comunicaciones alámbricas [9]:

- Insuficiencia del canal del ancho de banda inalámbrico y la interferencia del vínculo multisalto hacen más difícil el soporte de servicios de alta calidad.
- Inestabilidad de conexión debido a interferencias inalámbricas.
- Prolongados retrasos y prolongadas variaciones de retrasos hacen más difícil dar soporte a servicios de tiempo real.
- Diversidad y complejidad de tecnologías de acceso alámbrico encaminan soluciones múltiples en diferentes dominios con diferentes perspectivas, dejando una interoperabilidad difícil de lograr.

3.1.4 Los protocolos QoS

Algunas aplicaciones son más estrictas en sus requerimientos de QoS que otras y por esta razón se tienen dos tipos básicos de QoS disponibles [33]:

- Por reservación de recursos: recursos de red son asignados de acuerdo a las peticiones de la QoS de una aplicación y se encuentra sujeta a las políticas de manejo del ancho de banda.
- Por prioridad: el tráfico de la red es clasificado y los recursos de la red son asignados de acuerdo a criterios de políticas del manejo del ancho de banda. Para permitir la QoS, los elementos de la red brindan un trato preferencial a clasificaciones identificadas al tener una demanda mayor de requerimientos.

Estos tipos de la QoS pueden ser empleados a flujos individuales de aplicaciones o hacia flujos agregados, por lo tanto hay otras dos maneras de caracterizar tipos de la QoS [33]:

- Por flujo: un flujo es definido como un individual y uni-direccional, flujo de datos entre dos aplicaciones (emisor y receptor), unívocamente identificado por una quíntupla (protocolo de transporte, dirección fuente, número de puerto fuente, dirección destino y número de puerto destino).
- Por agregado: un agregado es simplemente dos o mas flujos. Típicamente los flujos tendrán algo en común (por ejemplo, cualquiera de uno o más de los parámetros de la quíntupla, una etiqueta o un número de prioridad o tal vez alguna información de autenticación).

Las aplicaciones, la topología y políticas de la red dictan que tipo de QoS es más apropiada para flujos individuales o agregados. Para poner en orden las necesidades de aquellos diferentes tipos de QoS, existen diferentes protocolos y algoritmos QoS:

- Protocolo por Reservación (RSVP) [34]: provee la señalización para permitir la reservación de recursos de la red (también conocida como Servicios Integrados (IntServ) [8]). A pesar de ser típicamente utilizado en flujos individuales, RSVP es también utilizado para reservar recursos en flujos agregados.

- Servicios Diferenciados (DiffServ) [7]: provee una manera simple y general para categorizar y priorizar tráfico de la red (flujo) agregado.

- Protocolo Múltiple de Conmutación por Etiquetas (MPLS) [25]: provee el manejo del ancho de banda para flujos agregados por medio del control de ruteo de la red de acuerdo a etiquetas (encapsulando) en los encabezados de los paquetes.

RSVP ofrece el mayor nivel de QoS disponible. Permite a una aplicación solicitar QoS con un nivel alto de granularidad y con las mejores garantías de entrega de servicio posible. La complejidad y sobrecarga son el precio intrínseco de brindar estas características, siendo un gran golpe para muchas aplicaciones y para algunas porciones de la red. Métodos más simples, menos ajustados son necesarios y es lo que DiffServ provee.

3.2 Servicios Diferenciados (DiffServ)

El RFC 2475 DiffServ [7] define una arquitectura para implementar un servicio escalable de diferenciación en Internet. Un "Servicio" define algunas características significativas para la transmisión de paquetes en una dirección a través de un conjunto de uno o más caminos dentro de una red. Tales características pueden ser especificadas en términos de rendimiento cuantitativos o estadísticos, retrasos, *jitter* y pérdida; o de otra manera pueden ser especificadas en términos de alguna prioridad relativa de acceso a los recursos de la red. Una diferenciación de servicios es necesaria para incorporar requerimientos de aplicaciones heterogéneas y de expectativas de los usuarios.

Esta arquitectura está compuesta de un número de elementos funcionales, implementados en los nodos de la red, incluyendo un conjunto pequeño de comportamientos de reenvío por salto, funciones de clasificación de paquetes y funciones de condicionamiento de tráfico que incluye medidores, marcadores, perfiladores y contralores. La arquitectura consigue la escalabilidad al implementar una clasificación compleja y funciones de acondicionamiento solamente en nodos en la demarcación de la red, y al aplicar Comportamientos por Salto (PHB) a agregados de tráfico, los cuales han sido marcados apropiadamente utilizando el campo DiffServ (DS) en los encabezados IPv4 o IPv6. Los PHBs son definidos para permitir una manera granular razonable de destinar recursos de *buffer* y ancho de banda en cada nodo, entre flujos de tráfico compartiendo la red. Es imporante tener claro que:

- El servicio es brindado a un agregado de tráfico,

- las funciones de condicionamiento y PHBs son utilizados para brindar niveles de servicios,
- el valor del campo DS es utilizado para marcar paquetes para seleccionar un PHB, y
- los mecanismos particulares en la implementación del nodo constituyen un PHB

El abastecimiento de servicios y las políticas de condicionamiento de tráfico están suficientemente desacopladas de los comportamientos de reenvío al interior de la red, para permitir la implementación de una variedad de comportamientos de servicio, con cabida a una futura expansión.

Esta arquitectura sólo provee diferenciación de servicios en el flujo de tráfico en una sola dirección y es por tanto asimétrica.

DiffServ [7] especifica la utilización del byte Tipo de Servicio (ToS)/DS que se encuentra en el encabezado de IPv4 (capa 3 del modelo OSI). Los seis bits de mayor precedencia del byte DS son los utilizados para definir los Código de Punto DiffServ (DSCP), los dos bits restantes no son utilizados (NU) por DiffServ, obsérvese la figura 3.1. El campo DSCP especifica el comportamiento de envío que el paquete tiene que recibir dentro del dominio DiffServ de cada operador. El comportamiento es llamado Comportamiento por Salto (PHB) y es definido de manera local; no es una especificación principio-a-final (como RSVP) pero está estrictamente relacionada con un dominio específico. El mismo DSCP podría tener dos significados diferentes en dos dominios diferentes, negociaciones entre todos los dominios adyacentes son requeridas para asegurar el correcto comportamiento de envío principio-a-final.

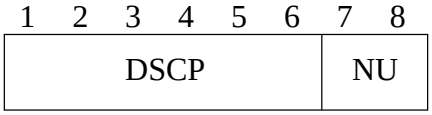

Figura 3.1: Significado del byte ToS en DiffServ

El objetivo principal de DiffServ es diferenciar tipos de flujos y dar prioridad a unos sobre otros. El enfoque DiffServ no distingue de forma específica a cada uno de los flujos de usuario en la red. El tráfico es clasificado y agregado a diferentes clases de servicio, cada una de ellas individualizada por el valor en el campo DSCP [21]. En la práctica, el DSCP se encuentra asignado (especificando una clase de servicio), el tráfico se encuentra entonces marcado. El campo DSCP es asignado por un encaminador llamado "encaminador de borde", el cual es la puerta de entrada a la red DiffServ. En teoría, el campo DSCP podría encontrarse asignado directamente por la fuente de tráfico, pero en redes DiffServ, la operación se encuentra desempeñada por encaminadores de borde.

Clases de servicio

Babiarz et al. [6] describe las clases de servicio configuradas con DiffServ y recomienda como deben ser utilizadas.

Algunos códigos de punto utilizados son el valor 101110_2 recomendado para el PHB Envío Acelerado (EF) [11], utilizado para servicios de baja perdida, bajo retraso y baja variación de retraso. Los valores para Envío Asegurado (AF) [15] está formado por 4 clases independientes de envío AF y por cada clase define 3 precedencias de descarte de paquetes. Otro código es el de BE siendo el valor 000000_2 para dar la prioridad de mejor esfuerzo, la que es utilizada en la red cuando no existen mecanismos para utilizar DiffServ.

Babiarz et al. [6] define un Selector de Clase (SC) que provee soporte para definiciones históricas de códigos de punto y requerimientos PHB. El campo DS de un SC provee una compatibilidad limitada con esquemas pre DiffServ. De tal manera que el DSCP es dividido en dos partes: los tres primero bits son el Selector de Clase y los otros tres bits son la precedencia.

Al seguir la clasificación propuesta por Babiarz et al. , algunos valores DSCP pueden ser dedicados a la administración y control del tráfico (incluyendo la señalización). En dicho documento se provee una tabla de una posible asignación DSCP dentro de una red DiffServ y en [21] sección 3.4.4 se complementa dicha tabla agregando como cuarta columna (obsérvese la tabla 3.3) valores decimales del SC, la precedencia y el número DSCP (6 bits).

Diferentes servicios son construidos, al fijar los bits en el encabezado IP en la demarcación de la red (definido por el límite del sistema autónomo, los límites administrativos internos o por los *hosts*); al utilizar dichos bits para determinar cómo son reenviados los paquetes por los nodos dentro de la red y al condicionar los paquetes marcados en los límites de la red en concordancia con los requerimientos o reglas de cada servicio [20].

Los componentes de una red DiffServ incluye: (i) los clasificadores de paquetes para seleccionar paquetes en un flujo de tráfico, basándose en la información del paquete, (ii) los perfiles de tráfico, que especifica las propiedades temporales de un flujo de tráfico por un clasificador y (iii) los condicionadores de tráfico, que pueden contener (a) un medidor para cuantificar propiedades temporales, (b) un marcador para fijar el campo DS, (c) un modelador para conllevar los paquetes en conformidad con el flujo, y (d) dosificadores para desechar algunos o varios de los paquetes en un flujo, a fin de obtener el flujo en conformidad con un perfil de tráfico.

3.3 QoS en WLAN

La información presentada sobre este tópico es una extracción de información de las referencias [9], [16] y [5].

En una red cableada Ethernet se encuentra un ancho de banda grande y bajas

Servicio	Clase de tráfico	DSCP (base 2)	SC-P-DSCP (base 10)	Aplicaciones ejemplo
Telefonía	EF	101110	5-6-46	Portador de telefonía IP
	AF41	100010	4-2-34	
Conferencia multimedia	AF42	100100	4-4-36	Video conferencia
	AF43	100110	4-6-38	
	AF31	011010	3-2-26	
Flujo multimedia	AF32	011100	3-6-28	Transmitir video y audio
	AF33	011110	3-6-30	
Datos de transacciones de baja latencia	AF21	010010	2-2-18	Transacciones basadas en web cliente/servidor
	AF22	010100	2-4-20	
	AF23	010110	2-6-22	
	AF11	001010	1-2-10	
Datos de alto rendimiento	AF12	001100	1-4-12	
	AF13	001110	1-6-14	
Datos por omisión	BE	000000	0-0-0	Sin especificar
Datos de baja prioridad	CS1	001000	1-0-8	Mejor esfuerzo
Eventos de *broadcast* de video	CS3	011000	3-0-24	*Broadcast* de TV
Interacción de tiempo real	CS4	100000	4-0-32	Aplicaciones interactivas y videojuegos

Tabla 3.3: Clases QoS ITU-T

tasas de paquetes erróneos. Por el contrario, el medio inalámbrico tiene relativamente un ancho de banda limitado y mayores tasas de paquetes erróneos con largos retrasos de los paquetes. Esto podría potencialmente limitar el uso de Redes de Área Local Inalámbrica (WLANs) para la entrega de tráfico de aplicaciones de tiempo real, tales como telefonía VoIP y aplicaciones multimedia. Los usuarios finales no sólo esperan la movilidad proveída por las WLANs si no también soporte la QoS. El estándar IEEE 802.11e [4] incorpora la QoS al estándar IEEE 802.11 [2].

El estándar original 802.11 no fue diseñado para proveer priorización y diferenciación basada en el tipo de tráfico, de tal modo que suministra un menor desempeño para voz y video, sobre aplicaciones en WLAN. Las aplicaciones de voz requieren buenas conexiones y pocas llamadas fallidas. Sin embargo, las aplicaciones de audio/video requieren un ancho de banda suficiente para mantener flujos de audio/video de alta calidad. Por otra parte, las aplicaciones de correo electrónico y de intercambio de archivos requieren una entrega libre errores en los archivos. Para cumplir con estos requerimientos, el estándar 802.11e agrega varias características de QoS y mejoras al estándar 802.11:

- Reduce la latencia, al priorizar paquetes inalámbricos basados en su tipo de tráfico.

- Habilita al Punto de Acceso (PA), el planificar recursos basándose en los requerimientos de la tasa de transmisión y de retraso del cliente.

- Mejora la eficiencia del ancho de banda inalámbrico y la administración de paquetes.

3.3.1 Control de Acceso al Medio (CAM) 802.11

El CAM 802.11 original no provee servicios diferenciados basados en el tipo de tráfico. Sin embargo, debido a que las redes inalámbricas proveen servicios multimedia que incluyen voz y video, los altos costos operativos en los paquetes con un ancho de banda limitado en una WLAN, pueden llegar a ser un impedimento para la entrega de paquetes sensibles al retraso. En el estándar original 802.11, hasta una tercera parte en la tasa de datos es consumida por la fragmentación de paquetes, espaciados de entre-trama y reconocimientos [16]. Además, bajo condiciones de carga de tráfico, colisiones y *backoffs*[1] pueden deteriorar severamente la calidad de aplicaciones de voz y video.

El estándar 802.11 especifica dos mecanismos de acceso al canal: una Función de Coordinación Distribuida (DCF) y una Función de Coordinación de Punto (PCF). El DCF permite compartir el medio inalámbrico entre las estaciones (ESTs) y el PA utilizando el Acceso Múltiple por Detección de Portadora con Evasión de Colisiones (CSMA/CA). DCF provee un servicio de mejor esfuerzo, es decir, no provee alguna priorización en el acceso al medio ni da soporte a requerimientos de retraso y de ancho de banda de diferentes aplicaciones. El modo DCF opera como sigue: cada estación verifica cuando el medio se encuentra inactivo, antes de poder transmitir. Si el medio es detectado como inactivo por un intervalo de tiempo equivalente al Espacio de Entre-Trama Distribuido (DIFS), la estación comienza a transmitir. En el caso donde el medio se encuentra ocupado, la estación aplaza la transmisión hasta que el medio se encuentre inactivo por un tiempo DIFS. La estación selecciona un intervalo aleatorio de *backoff* (utilizando un algoritmo de *backoff*) y decrementa un contador de *backoff* mientras el medio se encuentra inactivo. El mecanismo de *backoff* es utilizado para prevenir a dos o más estaciones de transmitir simultáneamente. Una vez que el intervalo *backoff* ha expirado, la estación comienza su transmisión. El rango en el cual el intervalo *backoff* es elegido es llamado Ventana de Contienda (CW) y depende del número de intentos previos de retransmisión. Una vez que la Unidad de Datos de Servicio CAM (MSDU) ha sido transmitida, la estación espera durante un Espacio Corto de Entre-Trama (SIFS) por el Reconocimiento (ACK) del receptor.

[1]Un *backoff* resulta cuando una terminal que ha experimentado una colisión en una red, espera por una cantidad de tiempo antes de intentar retransmitir.

PCF es un mecanismo opcional de acceso al canal en el estándar 802.11, que no es comúnmente implementado debido a la falta de demanda en el mercado. PCF provee un acceso al medio libre de contienda. Fue diseñado para dar soporte a aplicaciones sensibles al tiempo. El Punto Coordinador (PCo) residente en el PA provee un acceso al medio inalámbrico libre de contención. Un método de votación es utilizado para conceder accesos, con el PCo, actuando como votador maestro. Esto elimina colisiones, el tiempo gastado en el *backoff* y la contención descrita anteriormente en DCF.

El acceso libre de contención hacia el medio no es provisto todo el tiempo. Cuando se utiliza PCF, el tiempo en el medio está dividido entre un Periodo Libre de Contención (PLC) y un Periodo de Contención (PCon) dado por el PCo. Durante PLC y PCon, PCF y DCF son utilizados por el acceso al medio respectivamente.

Ni DCF ni tampoco PCF cuentan con una funcionalidad suficiente para proveer la QoS demandada por aplicaciones multimedia. DCF trata a todo el tráfico de manera igual, con todas las estaciones compitiendo por el medio con la misma prioridad.

3.3.2 Estándar IEEE 802.11e

La fuerza de trabajo E IEEE 802.11 (IEEE 802.11e) [4] definió mejoras al original CAM 802.11 para proveer QoS. El estándar 802.11e introduce la Función de Coordinación Híbrida (HCF), la cual combina funciones de DCF y PCF con mejoras de mecanismos específicos QoS y tipos de tramas. HCF tiene dos modos de operación: el Acceso Coordinado Distribuido Mejorado (EDCA) y el Acceso al Canal Controlado HCF (HCCA). EDCA y HCCA son mecanismos basados en contención y basados en votaciones, para el acceso al canal respectivamente y operan concurrentemente. Durante el periodo de contención, EDCA es utilizado para el acceso al canal; mientras que durante el PLC, el HCCA es en su mayor parte utilizado. Una estación (EST) que da soporte QoS es referida como una estación mejorada QoS (QEST); mientras que un PA que da soporte QoS es referido como un PA mejorado QoS (QPA).

HCF asigna a las QESTs el derecho a transmitir dentro de una Oportunidad de Transmisión (TXOP). Una TXOP define el tiempo de inicio y la duración máxima durante la cual una QEST puede transmitir una serie de tramas.

3.3.3 Acceso Coordinado Distribuido Mejorado (EDCA)

La contención de acceso EDCA es una extensión de DCF y provee un acceso priorizado al medio inalámbrico. El mecanismo de acceso al canal EDCA define cuatro Categorías de Acceso (CA), basadas en el estándar IEEE 802.1d [3] para proveer prioridades.

Cada Categoría de Acceso (CA) tiene su propia lista de espera para transmitir. Los siguientes cuatro parámetros clave de Especificaciones de Tráfico (TSPECs) son utilizados para la diferenciación:

- Ventana de Contienda mínima (CWmin). Una CA de mayor prioridad le está asignada una CWmin corta.

- Ventana de Contienda máxima (CWmax).

- La TXOP limita/especifica la duración máxima que una QEST puede transmitir y está especificada por la CA. La limitación de TXOP puede ser utilizada para asegurar que tráfico de alto ancho de banda obtenga un mayor acceso al medio. La limitación de TXOP también hace que el protocolo de acceso al canal sea significativamente más eficiente.

- Espacio Entre-Trama Arbitrario (AIFS). Especifica el intervalo de tiempo entre que el medio inalámbrico estará inactivo y el inicio de la negociación de acceso al canal. Cada CA es asignada a diferentes AIFS[CA] basado en la CA para fomentar el suministro de diferenciación de QoS.

Cada CA compite independientemente por TXOPs, basado en los parámetros antes mencionados dentro de la QEST. Una vez que la CA ha censado que el medio ha estado inactivo por AIFS[CA], inicia su tiempo de *backoff* (de manera similar que DCF). Si existiése una colisión entre las CAs dentro de una QEST, las tramas de datos de la CA de mayor prioridad recibe una TXOP. Las tramas de datos de las sobrantes CA se comportarán como si hubiese una colisión externa.

3.3.4 Acceso al Canal Controlado HCF (HCCA)

HCCA utiliza un Coordinador Híbrido (CH) para manejar de forma centralizada el acceso al medio inalámbrico, para proveer una QoS parametrizadq. La QoS parametrizada se refiere a la capacidad de proveer flujos de tráfico de aplicaciones con parámetros específicos QoS (tales como la tasa de datos y latencia entre otros). Tanto el PCF, como el HCCA utilizan un mecanismo basado en votación para acceder al medio, por lo tanto reduce la contención en el medio inalámbrico. Las diferencias clave entre HCCA y PCF son que HCCA puede elegir a las estaciones durante un PCon y que soporta una planificación de paquetes basado en los requerimientos específicos de flujo de tráfico en la QEST. Los requerimientos de flujo de tráfico en las QEST están especificados utilizando Especificaciones de Tráfico (TSPECs) presentados en la siguiente sección.

El modo PCF del legado IEEE 802.11 tiene problemas con retrasos en el *beacon* que impide su implementación y este problema persiste en el modo HCF del IEEE 802.11e.

El CH tiene la mayor prioridad sobre todas las QEST, para ganar el acceso al medio, tanto que tiene el tiempo de espera menor, comparado a los tiempos de *backoff* de las QESTs. El CH provee una trama de intercambio libre de contención con retrasos cortos, por lo tanto brinda una latencia estrecha controlada.

3.3.5 Especificaciones de Tráfico (TSPECs)

Dado que las WLANs tienen un ancho de banda limitado y proveen accesos al medio basados en contención, son susceptibles a una congestión de tráfico, la cual puede encaminar a una degradación severa en el desempeño de la red. Tanto como la red se sobrecargue, los tamaños de la CW puede incrementarse significativamente, conduciendo a tiempos prolongados de *backoff*. Esto demanda algún mecanismo de control de admisión para ser construido dentro del estándar para regulación del tráfico. El estándar IEEE 802.11e especifica el uso de TSPECs en la negociación del control de admisión para EDCA y HCCA.

Las TSPECs son utilizadas por las QEST para especificar requerimientos de flujo de tráfico, tales como tasa de datos, retraso, tamaño de paquete e intervalo de servicio. El QPA podría aceptar o rechazar una petición nueva de TSPEC, basada en las condiciones de la red. Si una TSPEC es rechazada por el QPA, a una CA de alta priorida dentro de la QEST demandante no se le permite utilizar parámetros de acceso de alta prioridad.

3.3.6 Mejoras en el Control de Acceso al Medio (CAM) 802.11e

Aparte de proveer funcionalidad EDCA y HCCA, el estándar 802.11e provee varias mejoras en el CAM. Alguna de las mejoras se mencionan a continuación.

Ráfaga libre de contención. Permite a un QEST/QPA enviar múltiples tramas en línea sin tener que contender por el medio una y otra vez. La QEST o el QPA continúa a transmitir después del retraso SIFS si existiera tiempo remanente en una TXOP concedida.

Nuevas reglas de reconocimiento. En el estándar 802.11, todas las tramas de datos *unicast* requieren una trama de control inmediata ACK. HCF agrega las siguientes opciones:

- No ACK: incrementa la eficiencia al no enviar ACK para ciertas aplicaciones. Esta característica es útil para aplicaciones con tolerancias muy bajas de latencia, pero que pueden ser tolerantes a una cantidad significativa de paquetes perdidos.
- Bloqueo ACK: incrementa la eficiencia al agregar ACKs de múltiples tramas en una sola respuesta.

Protocolo de enlace directo. La especificación original 802.11 permite que el tráfico en una red, basada en PA, fluya solamente entre las ESTs y el PA. El protocolo de enlace directo en el estándar 802.11e agrega la capacidad para que las ESTs puedan enviar tráfico directamente entre ellas sin atravesar el PA. Aunque esta capacidad puede incrementar potencialmente el ancho de banda, se limita a que las ESTs se encuentren en un alcance entre ellas.

Sobreposición. La sobreposición reduce ACKs y votaciones al enviar datos sobrepuestos en votaciones y ACKs.

Entrega Automática de Ahorro de Enegía (APSD). Habilita una extensión en el tiempo de la batería al permitir a los dispositivos el apagar sus radios en la mayoría del tiempo. Esto es una mejora al mecanismo de ahorro de energía en 802.11. APSD permite a una estación el programar una planificación para la entrega de tramas, basado en un patrón repetitivo de un específico número de intervalos *beacon*.

3.3.7 Multimedia Inalámbrica (WMM)

La Alianza Wi-Fi inició una certificación de interoperabilidad para Multimedia Inalámbrica (WMM) o Multimedia Wi-Fi, como un perfil de la extensión QoS IEEE 802.11e para redes 802.11. WMM prioriza las demandas de tráfico de diferentes aplicaciones. WMM define cuatro Categorías de Acceso (voz, video, mejor esfuerzo y *background*), que son utilizadas para priorizar tráfico de tal manera que las aplicaciones puedan acceder a los recursos necesarios de la red.

El programa de certificación se encuentra disponible en todos los nuevos dispositivos Wi-Fi-habilitados. Existen dispositivos Wi-Fi que pueden recibir una actualización de software.

Para tomar ventaja de la funcionalidad WMM en una red Wi-Fi, tres requerimientos tienen que cumplirse: (1) el PA tiene la certificación Wi-Fi para WMM y tiene habilitado WMM; (2) el cliente (dispositivo) dónde se encuentra ejecutándose la aplicación deberá ser certificado Wi-Fi para WMM; y (3) la aplicación fuente soporta WMM.

WMM provee acceso a media priorizado y está basado en el método EDCA. Algunos sistemas operativos y aplicaciones proveen soporte WMM/EDCA pero carecen de soporte WMM-Planificado. El Acceso WMM-Planificado (WSM), también conocido como Multimedia Planificada Wi-Fi, es un subconjunto del borrador del estándar 802.11e. La principal característica del Acceso WMM-Planificado es HCCA.

La mayoría de los dispositivos Wi-Fi en el mercado no dan soporte QoS. Esto podría no cambiar, tanto que muchos dispositivos y aplicaciones no requieren aptitudes QoS. WMM permite que clientes Wi-Fi con y sin habilidades WMM puedan coexistir en la misma red.

En una red Wi-Fi, la funcionalidad WMM requiere que tanto el PA y los clientes que ejecutan aplicaciones que requieren QoS se encuentren certificadas para WMM y tengan habilitado WMM. Al mismo tiempo, es importante dar cuenta que dispositivos WMM habilitados puedan tomar ventaja de su funcionalidad QoS, sólo cuando se utilizan aplicaciones que soportan WMM y pueden asignar el nivel de prioridad apropiada para los flujos de tráfico que ellas generan. Aplicaciones o sistemas operativos que no dan soporte WMM no pueden acceder a las capacidades de red QoS.

El tráfico que que generen es tratado como de mejor esfuerzo y recibe una prioridad inferior a la de voz y video.

WMM está basado en la arquitectura IETF DiffServ [7], la cual se encuentra apropiada para brindar QoS en tecnologías de media compartida como Wi-Fi, al habilitar una priorización de tráfico efectiva sin la imposición de un laborioso gasto operacional. Paquetes individuales son marcados ya sea en el encabezado IETF DSCP o en las etiquetas IEEE 802.1d.

WMM define cuatro CAs **derivadas** del estándar 802.1d [3], que corresponden a niveles de prioridad, véase tabla 3.4. Mientras que cuatro CAs fueron diseñadas con tipos de tráfico específicos, WMM deja al administrador de la red la libertad de elegir la política más apropiada sobre la red y el decidir las prioridades de las CAs. Por ejemplo, un administrador de la red podría preferir dar más prioridad a un flujo de video sobre un flujo de voz. Una política construida a la medida para las CAs puede ser determinado a través de una interfaz en la cual los niveles de prioridad de las CAs pueden ser modificados. WMM especifica un protocolo utilizado por el PA para comunicar la política a las QEST y de las peticiones de transmisión de las QEST.

Categoría de Acceso	Descripción	802.1d etiq.
WMM prioridad voz	De mayor prioridad Permite múltiples llamadas concurrentes VoIP, con baja latencia y calidad de voz	7, 6
WMM prioridad video	Prioriza el tŕafico de video por encima de otro tráfico de datos	5, 4
WMM prioridad de mejor esfuerzo	Tráfico de dispositivos obsoletos, tráfico de aplicaciones o dispositivos que carecen de capacidad QoS Tráfico menos sensitivo a una latencia, pero afectada por retrasos largos, como al navegar en internet	0, 3
WMM prioridad de *background*	Tráfico de baja prioridad (descarga de archivos, impresiones) que no tienen una latencia estricta o requerimientos de rendimiento	2, 1

Tabla 3.4: Categorías de Acceso WMM

Esquema de operación WMM

WMM es una mejora de la subcapa del CAM para agregar funcionalidad de QoS a redes Wi-Fi. WMM es una extensión del mecanismo original DCF basado

en CSMA/CA que brinda a todos los dispositivos la misma prioridad y que se encuentra basada en un algoritmo de mejor esfuerzo escuchar-antes de-hablar. Cada cliente espera durante un tiempo aleatorio de *backoff* y luego transmite sólo si no hay otro dispositivo transmitiendo al mismo tiempo. Este método de evasión de colisiones le da a todos los dispositivos la oportunidad de transmitir, pero, bajo condiciones de demandas de alto tráfico, las redes se sobrecargan y el desempeño de todos los dispositivos es de la misma manera afectadas.

Las etiquetas 802.1d son mapeadas con los tres bits más significativos del campo DSCP [47], véase tabla 3.5.

DSCP	802.1d	CA WMM
111XXX	7	AC_VO
110XXX	6	AC_VO
101XXX	5	AC_VI
100XXX	4	AC_VI
011XXX	3	AC_BE
010XXX	2	AC_BK
001XXX	1	AC_BK
000XXX	0	AC_BE

Tabla 3.5: Mapeo DSCP a 802.1d y Categoría de Acceso (CA) WMM

Un PA WMM coexiste con dispositivos legados (o dispositivos que no están habilitados con WMM): paquetes no asignados a una CA específica son categorizados por omisión con una prioridad de mejor esfuerzo.

La priorización funciona como se muestra en la figura 3.2. Las aplicaciones asignan cada paquete de datos a una CA dada. Los paquetes son entonces agregados a una de las cuatro listas de espera de transmisión independientes en el cliente. El cliente tiene un mecanismo interno de resolución de colisiones, para atender las colisiones entre diferentes listas, el cual selecciona las tramas con mayor prioridad para transmitirlas. El mismo mecanismo trata con las colisiones externas, para determinar a que cliente debería ser permitido la TXOP.

El algoritmo de resolución de colisiones es responsable para la priorización de tráfico, es probabilístico y depende de dos parámetros de tiempo, que varían para cada CA:

- El espacio mínimo de entre-trama o Número AIFS (AIFSN).
- La Ventana de Contienda (CW), algunas veces referida como espera aleatoria de *backoff*.

Ambos valores son pequeños para el tráfico de alta prioridad. Para cada CA, un valor de *backoff* se calcula como la suma del AIFSN y un valor aleatorio de cero a CW. El

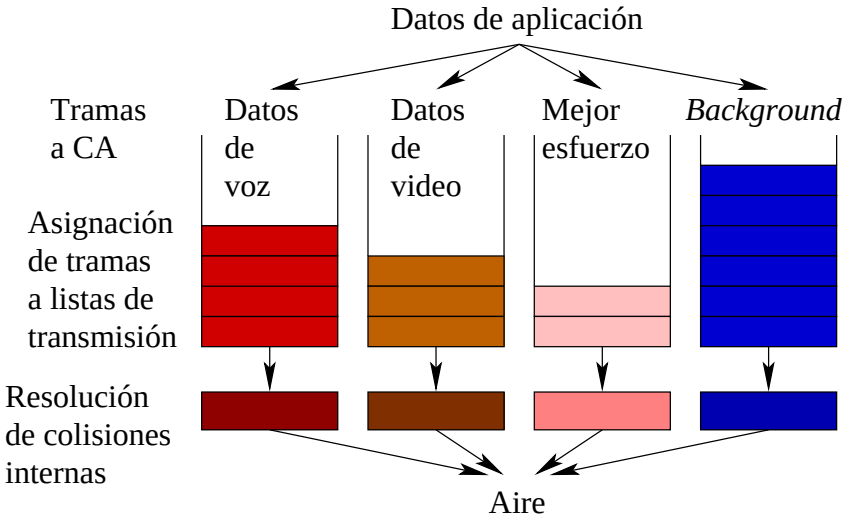

Figura 3.2: Listas de transmisión dentro de un cliente WMM

valor de la CW varía respecto al tiempo. Inicialmente la CW es definida con un valor que depende de la CA.

Después de cada colisión, la CW es aumentada al doble hasta alcanzar un valor máximo (también dependiente de la CA). Después de una transmisión satisfactoria, la CW es reanudada a su valor inicial dependiente de la CA. La CA con el menor valor *backoff* obtiene la TXOP. Tanto como tramas con la mayor CA tiendan a tener los menores valores *backoff*, son más propensas a obtener una TXOP.

Una vez que un cliente gana una TXOP, le es permitido transmitir por un tiempo dado que depende de la CA y la tasa de la capa física. Por ejemplo, el límite TXOP varia de 0.2 ms (prioridad de *background*) a 3 ms (prioridad de video) en una red 802.11a/g y de 1.2 ms a 6 ms en una red 802.11b. Esta capacidad de ráfaga mejora la eficiencia para tasas de tráfico de datos elevadas, tales como flujos de audio y video.

El borrador 802.11e incluye características adicionales que pueden ser agregadas a WMM como módulos opcionales, las cuales incluyen Acceso Planificado, disposición directa de enlace, bloqueo ACK y ahorro de energía.

Acceso Planificado

El Acceso Planificado permite a las aplicaciones la reservación de recursos de red basado en sus características de tráfico a través de peticiones enviadas por el cliente al PA. El Acceso Planificado soporta parametrización, accesos planeados y corresponde al HCCA en 802.11e. Dado que el impacto producido por un retraso de *backoff* es menor, los Accesos Planificados pueden reducir en promedio la latencia en la red al utilizar un mecanismo de control planificado centralizado.

La planificación de elecciones para el Acceso Planificado está diseñado para con-

centrarse en requerimientos de aplicaciones de rendimiento y latencia al asignar los tiempos de cuando las aplicaciones pueden transmitir. El cliente envía una petición de reservación al PA, el cual en turno asigna TXOPs a los flujos de tráfico apropiados, tales como tasas de datos, tasa de la capa física, tamaños de paquetes e intervalo de servicio.

Al contrario de la referencia WMM, el Acceso Planificado requiere que los clientes conozcan de antemano cuales recursos necesitan y que el PA realice suposiciones (por ejemplo, tamaño de paquete mínimo vs. máximo, tasa física mínima vs máxima, el tiempo de inicio de servicio y excedentes reservadas de ancho de banda de reintentos) para planificar de manera efectiva al tráfico concurrente.

Capítulo 4

Plataforma y herramientas para VoIP con QoS

En este capítulo, se describe la plataforma que se integró para montar un sistema de comunicación VoIP, que tiene como principal canal de comunicación el protocolo SIP. La red utilizada es una red heterogénea IP de tipo WLAN, para el medio inalámbrico, y de tipo Ethernet para el medio alámbrico. La QoS que se provee está directa y estrechamente ligada con los mecanismos que proveen los dispositivos de la Red WLAN/Ethernet. La manera en que se brindará QoS será a través de DiffServ en la plataforma.

El desarrollo de aplicaciones para diversas plataformas significa obtener el conjunto de herramientas apropiadas y crear aplicaciones ejecutables para dichas plataformas. Para resolver el aspecto de la heterogeneidad de dispositivos, que en principio cuentan con diferentes sistemas operativos, resulta útil contar con las bibliotecas PJ para desarrollar aplicaciones basadas en SIP.

La convergencia multimedia que se experimenta en esta plataforma, resulta al utilizar una tarjeta TDM para conectar el PBX con líneas analógicas y teléfonos analógicos comunes en la PSTN, además del canal SIP que conecta teléfonos SIP como los *softphone*. De esta manera se están interconectando dos redes distintas, por un lado se encuentra la red IP y por el otro la PSTN.

4.1 Infraestructura

A continuación se muestra una lista de las características más notables de los dispositivos en la plataforma.

- PC Intel(R) Pentium(R) D CPU 2.80GHz, 2GB, S.O. Ubuntu Linux.
- PC Intel(R) Pentium(R) 4 CPU 3.00GHz, S.O. Fedora Linux.
- TDM422P con 2 puertos FXO y 2 puertos FXS.

- Conmutador SRW2008x Linksys.
  - Gigabit 10/100/1000 de 8 puertos.
- Punto de Acceso (PA) inalámbrico WRT54G Linksys.
  - WLAN 802.11bg.
  - Firmware WRT54G V8 estándar.
- Smartphone Nokia E65
  - S60 3ra Edición, basado en el S.O. Symbian.
  - WLAN 802.11bg, seguridad: WPA2-Enterprise, WPA2-Personal, WPA-Enterprise, WPA-Personal.
  - WLAN QoS: WMM.
  - Memoria de usuario de 70 MB, puerto MicroSD (2GB).
  - Velocidad del procesador 220 MHz.
- PDA, HP iPAQ HW6945.
  - Procesador Intel PXA270 a 416MHz.
  - S.O. Windows Mobile 5.0.
  - Memoria de usuario 45MB, 64MB SDRAM para aplicaciones en ejecución.
  - Wi-Fi 802.11b, Bluetooth 1.2 y puerto Infrarojo.
- IP-phone, SPA901.
  - IETF SIP (RFC3261).
  - Puerto Ethernet 10/100.
- HP iPAQ 200 *Enterprise Handheld*.
  - WLAN 802.11bg, WEP a través de WEP2-Enterprise y Wi-Fi Multimedia
  - S.O. Microsoft Windows Mobile 6 Classic.
  - Procesador Marvell PXA310, 624MHz.
  - Memoria SDRAM de 128 MB y 256 MB flash ROM.

Como puede observase en la infraestructura, se cuenta con dispositivos móviles que cuentan con diferentes sistemas operativos (Windows Mobile y Symbian) con diferentes capacidades de procesamiento y almacenamiento. Aunque no se enlistan computadoras personales portátiles también se incluyen sistemas operativos como Linux y Windows. Uno de los beneficios de la biblioteca PJSIP es la compatibilidad con

todos estos sistemas operativos y el tamaño del programa ejecutable y su complejidad de cómputo es apto para dichos sistemas embebidos.

La PC con sistema operativo Ubuntu Linux es el servidor Asterisk, mientras que la segunda PC con sistema operativo Fedora Linux no es utilizada de manera formal en la arquitectura y podría servir para complementar con otros servicios a la plataforma.

Un ejemplo de la interconexión esperada se muestra en la figura 4.1.

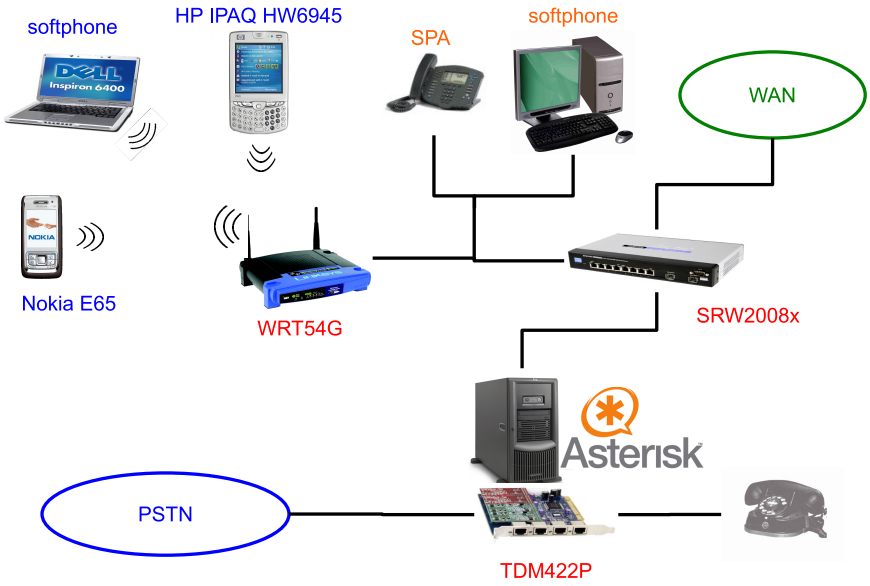

Figura 4.1: Interconexión de los dispositivos

4.2 Interconexión de la Red

En esta sección se presenta la red local en la cual todos los dispositivos de la infraestructura puedan comunicarse. Esta red cuenta con acceso a Internet. En el departamento de Computación del Cinvestav, las IPs de Internet tienen una red 148.247.0.0 y la puerta de enlace es la IP 148.247.102.254. En la arquitectura contamos con 3 IPs de este tipo.

Las dos computadoras presentes en la arquitectura tienen una IP de Internet y la tercer IP es utilizada por el PA.

La red está configurada de la siguiente manera: el PA por su puerto de Internet fue conectado a la red del Cinvestav con su puerto WAN y se le asignó una IP fija 148.247.102.20. El AP tiene habilitado el servidor DHCP y se encuentra debidamente

configurado, de tal manera que los dispositivos conectados por Wi-Fi cuentan con acceso a Internet.

La red local en el PA es la red 192.168.201.0 y el DHCP proveerá IPs de esta red a los clientes inalámbricos que realicen una petición DHCP. El PA tiene la IP local 192.168.201.1.

El conmutador SRW2008x tiene conectados por Ethernet las dos PC, el servidor Asterisk y una PC de pruebas. Las IPs locales de estas computadoras son la 192.168.201.22 y la 192.168.201.23 respectivamente. Dichas IPs se encuentran configuradas de manera estática en las computadoras y no realizan una petición DHCP pero forman parte de la red local. Bajo esta red las computadoras no se encuentran configuradas con una puerta de enlace.

El PA y el conmutador se encuentran interconectados mediante un puerto de red local del PA hacia alguno del conmutador. El conmutador es capaz de realizar una petición de DHCP, pero se optó por una configuración estática. La IP de área local que tiene el conmutador es la 192.168.201.254.

Para que las dos PCs tengan red de Internet, se instaló una interfaz de red adicional en cada una de estas máquinas y se configuró apropiadamente. La IP para la interfaz del servidor Asterisk es 148.247.102.22 y la IP para la PC de pruebas es 148.247.102.23, ambas IPs son del dominio de la red del Cinvestav y cuentan con acceso a Internet. La puerta de enlace es la IP 148.247.102.254. Estas interfaces son conectadas a un conmutador de la red del Cinvestav. Mediante este arreglo, las dos computadoras cuentan con una IP de la red local y una IP con acceso a Internet.

Los dispositivos conectados por Ethernet o Wi-Fi en el PA/conmutador pueden realizarse *pings* sin problemas. Esta simple prueba de *pings* nos dice si hay comunicación entre los dispositivos.

En la figura 4.2 se muestra la interconexión de la red, en la figura se aprecia que el IP-*phone* fue conectado a uno de los puertos Ethernet del PA, la configuración del IP-*phone* se mostrará más adelante.

4.3 Instalación, configuración y administración de un Intercambio Privado de Ramificación (PBX)

Una vez que se tiene configurada la red local, en donde dispositivos ya sean alámbricos o inalámbricos pueden conectarse, se procede a la instalación y configuración del Intercambio Privado de Ramificación (PBX). El PBX electo es Asterisk debido a las características que éste presenta, el cual suministra una gran potencialidad en una plataforma VoIP.

Para la plataforma VoIP, Asterisk funcionará no tan solo como PBX, sino también como servidor *proxy* y *registrador*.

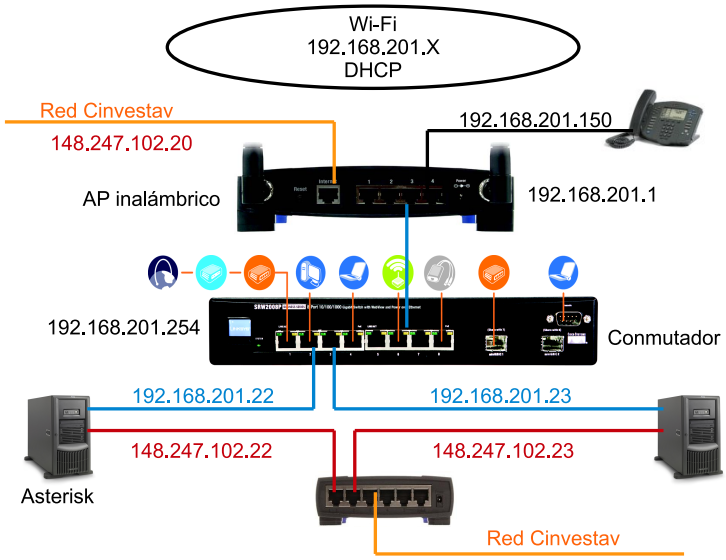

Figura 4.2: Interconexión de red

El servidor Asterisk tiene instalado un sistema operativo *Ubuntu Linux*. Esta distribución cuenta con una utilidad para manejar paquetería de software llamado `apt-get`, la descarga del software a instalar es obtenido de repositorios que se encuentran en la Internet. Mediante dicha herramienta puede ser instalado Asterisk. Las dependencias del software serán también instaladas, pero el inconveniente principal de este método es que los repositorios podrían no contar con las últimas versiones del software.

En [22] y [44] se encuentran las guías acerca de la instalación del software, además de una basta información respecto a la configuración y administración de Asterisk. En la primer referencia hay un ejemplo de instalación del software y sus dependencias mediante el programa `yum`. El programa `yum` actúa de manera similar que el programa `apt-get` y se encuentra instalado en la distribución Fedora Linux, dado que es su manejador de paquetes por omisión.

El código fuente que utiliza Asterisk puede ser descargado de sitio Web [44] en su sección de descargas, y es así como se obtiene el software más actualizado. El software y las versiones que fueron instaladas son las siguientes:

- asterisk-1.4.14
- zaptel-1.4.6
- libpri-1.4.2

El proceso de instalación de cada paquete se encuentra en el archivo `README` al descomprimir los archivos fuente. Se requiere resolver las dependencias que estos paquetes tengan con otro software (gcc, libc6, etc.) y el orden de instalación es: zaptel, libpri y asterisk.

Una vez instalado la paquetería necesaria, se procede a la configuración básica del PBX y a continuación se muestran los pasos a seguir para lograr una comunicación VoIP.

Uno de los canales de comunicación de Asterisk es SIP. Para poder agregar un usuario (en realidad un dispositivo) comunicado por un canal SIP, se agregan los campos siguientes en el archivo `/etc/asterisk/sip.conf`:

```
[john]
type=friend
secret=welcome
nat=yes
context=internal
host=dynamic
```

El nombre de usuario es `john`, de tipo `friend` que significa que puede hacer y recibir llamadas, el *password* del usuario es `welcome`, con `nat=yes` significa que el teléfono/usuario se encuentra bajo una red Traducción de Dirección de Red (NAT). El contexto que se utilizará con esta cuenta será la llamada `internal` y por último el anfitrión puede tener una IP diferente entre sesiones al ser `dynamic`, de otro modo se puede especificar la IP fija.

Muchos más parámetros pueden ser configurados, como los tipos de codificadores de audio que podrían ser utilizados o permitidos por el dispositivo. Resulta importante conocer qué tipo de codificadores de audio soporta el dispositivo, agregar un codificador no soportado resultaría en una re-negociación SDP o incluso produciría un error para enlazar la llamada. Aún cuando dos dispositivos no tengan algún codificador en común y quieran comunicarse, Asterisk tiene la funcionalidad automática de re-muestrear el codificador de uno al codificador del otro y así enlazar la comunicación.

La información que necesita el *softphone* para poder ser registrado en el servidor Asterisk en general es:

- Nombre de usuario.
- IP del servidor Asterisk.
- Password.
- Puerto.

El puerto por conocer se define en el archivo `/etc/asterisk/sip.conf`, conforme al RFC de SIP [28] es el número 5060.

De esta manera se configuró un *softphone* que puede iniciar llamadas a través del canal SIP. Como ejemplo, se muestra en la figura 4.3 la ventana de configuración del *softphone* **kphone**. Otros *softphones* han sido probados y configurados (Xlite Windows/Linux [37] y Nokia E65 Symbian [36]). El teléfono celular Nokia E65 cuenta con un *softphone* instalado.

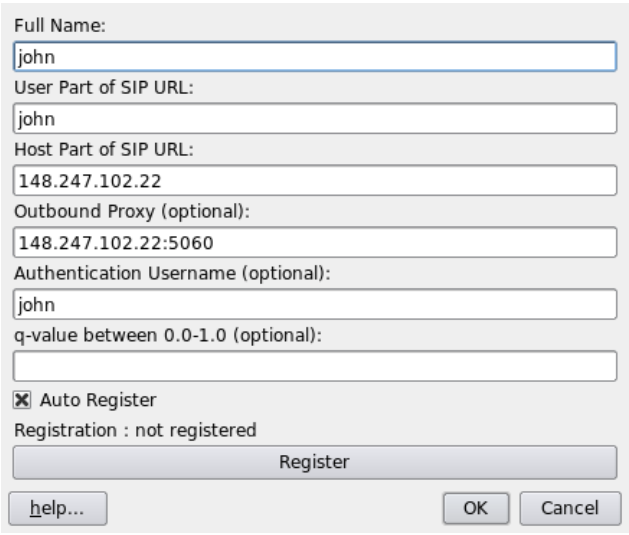

Figura 4.3: Configuración del *softphone* kphone

Obsérvese que en la configuración de la figura 4.3 se utiliza la IP de Internet del servidor Asterisk, aunque sin ningún inconveniente pudo haberse utilizado la IP local, siempre y cuando los dispositivos que se conectan al servidor se encuentren bajo la misma red local.

El parámetro `context` que ha sido evaluado como `internal`, significa que los números telefónicos/nombres marcados por el usuario que pertenece a ése contexto, será atendido por un plan de marcado llamado **internal**. Este plan de marcado se especifica en el archivo `/etc/asterisk/extensions.conf`. Como ejemplo, el siguiente plan de marcado devuelve al usuario todo lo que éste emite por su micrófono, cuando se ha marcado el número *611*, es decir, un eco:

```
 [internal]
exten => 611,1,Ring( )
exten => 611,2,Answer( )
exten => 611,3,Echo( )
```

Aunque un plan de marcado un poco más funcional que involucre a otros usuarios sería:

```
 [internal]
exten => 100,1,Dial(SIP/john)
exten => 666,1,Dial(SIP/ringo)
exten => 213,1,Dial(SIP/bowie)
exten => 300,1,Dial(SIP/spa)


exten => 611,1,Ring( )
exten => 611,2,Answer( )
exten => 611,3,Echo( )
```

Estos ejemplos son muy básicos y el alcance que provee Asterisk para realizar un plan de marcado incluye el poder realizar un Respuesta de Voz Interactiva (IVR), siendo incluso más funcional que un PBX.

Si por ejemplo, el *softphone* Xlite instalado en Windows se le asociase el usuario *ringo*, al *softphone* kphone instalado en Linux el usuario *john* y por último al *softphone* del dispositivo Nokia E65 el usuario *bowie*, ellos podrían comunicarse si pertenecen al contexto `internal` y se marcaran entre si por medio de sus extensiones definidas.

Esta fue la primer prueba del servidor Asterisk con interconexión de tres *softphones* ejecutados en diferentes sistemas operativos. Tómese en cuenta que no se mostró como se realizó la configuración de red de los dispositivos, ya que pudo haber haber sido configurado mediante una configuración de IP estática o por medio del servicio DHCP de la red local.

4.4 Configuración del IP-*phone*

Un IP-*phone* es un teléfono de apariencia común, como los encontrados en la PSTN, pero que se conecta a través de IP y el modelo que usamos utiliza el protocolo SIP para recibir y hacer llamadas VoIP.

Para comenzar la configuración, se agrega un nuevo usuario en Asterisk, específicamente en el archivo `/etc/asterisk/sip.conf`, como anteriormente se ha hecho.

Se realiza la conexión física Ethernet al dispositivo SPA-901 y al descolgar por primera vez el IP-Phone se accede al IVR de configuración básica. La configuración actual consiste en realizar la petición de una IP para el IP-phone SPA por medio del protocolo DHCP. El SPA está conectado al PA por medio de los puertos Ethernet ofrecidos por éste. El PA atenderá la solicitud DHCP asignándole una IP de clase C (red local).

A través de un servidor `HTTP` que tiene el SPA se realiza la configuración del dispositivo. La configuración es muy parecida a la realizada con el *softphone* kphone. Una vez completado el procedimiento, Asterisk registra el dispositivo.

Finalmente se agrega una extensión por el cual el IP-*phone* será llamado, esto se realiza en el archivo `/etc/asterisk/extensions.conf`.

4.5 Configuración de la tarjeta TDM422P

La tarjeta TDM nos permite enlazar una comunicación con la red PSTN. Con la disposición completa de parte del conmutador del Cinvestav, fue proporcionada una línea análoga que conecta a la PSTN del Cinvestav. De tal manera que el servidor Asterisk se encuentra conectado al PBX del Cinvestav. El otro tipo de líneas telefónicas que provee el PBX del Cinvestav son líneas digitales, por lo que se requeriría otro tipo de tarjeta que conecte a líneas digitales, alguna de la de serie TE400 de Digium.

Para iniciar la configuración, se identifican los módulos FXS y FXO, para la tarjeta TDM422P se tiene que los primeros dos puertos son FXS (de color verde) y los 2 siguientes son FXO (de color rojo). Una vez identificados los puertos se instala físicamente la tarjeta en el servidor Asterisk en un puerto PCI.

Al haber instalado el programa Zaptel, Linux reconoce la tarjeta instalada (el comando `dmesg` da prueba de ello) y se puede ver la siguiente línea al ejecutar el comando `lspci`:

```
04:02.0 Communication controller: Tiger Jet Network Inc. Tiger3XX
        Modem/ISDN interface
```

Una vez que el módulo es reconocido, se edita el archivo de configuración `/etc/zaptel.conf`, la configuración básica es la siguiente:

```
fxoks=1,2
fxsks=3,4
loadzone=mx
defaultzone=mx
```

Nótese que se especifica el uso de una señalización de tipo FXO para los módulos FXS de la tarjeta (`fxoks=1,2`, los primeros dos puertos son FXS), y de manera similar para los módulos FXO. Esto es porque el módulo FXS se comunica con un dispositivo/terminal FXO, por lo tanto tendrá que utilizar una señalización FXO.

Para actualizar los cambios, se requiere ejecutar el comando `ztcfg -vv`, el argumento es para desplegar más información de la ejecución del programa en la terminal. Si hubiera alguna falla se revisa la configuración, se realizan los cambios y se procede de esta manera hasta llegar a una configuración correcta. La tarjeta TDM se encuentra configurada y reconocida por el sistema operativo.

Enseguida se procede a configurar Asterisk, para hacer uso de los canales ofrecidos por la tarjeta TDM. En el archivo `/etc/asterisk/zapata.conf` se agregan las opciones generales:

```
usecallerid=yes
hidecallerid=no
callwaiting=yes
threewaycalling=yes
```

```
transfer=yes
echocancel=yes
echocancelwhenbridged=yes
rxgain=0.0
txgain=0.0

;;Módulos FXS
Group=1
context=internal
signalling=fxo_ls
channel => 1-2

;;Módulos FXO
Group=2
context=incoming
signalling=fxs_ls
channel=> 3-4
```

De esta manera se especifica que los números marcados o llamadas realizadas en los canales 1-2, que son teléfonos análogos comunes de la PSTN conectados al servidor Asterisk por medio de los puertos FXS, sean atendidos por el contexto `internal` que se especifica en el archivo `/etc/asterisk/extensions.conf`.

Haciendo uso de la extensión de eco del contexto `internal`, que resulta al marcar 611, se obtiene el eco en el teléfono análogo al marcar dicha extensión. Se agregan dos extensiones al archivo `/etc/asterisk/extensions.conf`, en el plan de marcado `internal`, para que los teléfonos análogos conectados a la TDM puedan ser accedidos:

```
 [internal]
exten => 100,1,Dial(SIP/john)
exten => 666,1,Dial(SIP/ringo)
exten => 213,1,Dial(SIP/bowie)

exten => 611,1,Ring( )
exten => 611,2,Answer( )
exten => 611,3,Echo( )

exten => 1000,1,Dial(zap/1,20,rt)   ;extensión del teléfono análogo 1
exten => 2000,1,Dial(zap/2,20,rt)   ;extensión del teléfono análogo 2

;extensión para hacer llamadas sobre la PSTN del Cinvestav
exten => _9.,1,Dial(zap/g2/www${EXTEN:1})
exten => _9.,2,Congestion
```

De tal manera que un usuario perteneciente al plan de marcado `internal` al marcar 1000 estaría llamando al primer teléfono análogo conectado a la TDM.

Para salir del PBX de Asterisk, a través de las líneas telefónicas PSTN conectadas a los puertos FXO, se marca 9 y a continuación todo número marcado es enviado a la PSTN, obsérvese la extensión `_9`. Por ejemplo, si un teléfono análogo conectado a algún puerto FXS o incluso un *softphone*/IP-*phone* conectado por el canal *SIP*, pertenecientes al contexto `internal` marca la extensión 96556, indica al servidor Asterisk marcar la extensión 6556 sobre algún puerto FXO disponible. Si dicha extensión existe en el plan de marcado del PBX de la PSTN, se enlazaría la comunicación.

Cabe mencionar, que el ejemplo de enlace antes mencionado podría obligar a Asterisk realizar una conversión entre codificadores, por ejemplo, si el *softphone* utiliza el codificador *Speex* y la PSTN siempre utiliza el codificador G.711, deberá realizar una conversión de un codificador a otro en una dirección o viceversa. La conversión entre codificadores resultaría en una carga al procesador del servidor Asterisk.

Regresando a la última parte del archivo `/etc/asterisk/zapata.conf`, previamente configurado, se especifica que las llamadas entrantes por los módulos FXO conectados a líneas telefónicas análogas de la PSTN, serán atendidas según su plan de marcado llamado `incoming`, este plan de marcado también deberá estar presente en el archivo `/etc/asterisk/extensions.conf`.

El plan de marcado para el contexto `incoming` se presenta a continuación:

```
[incoming]
exten => s,1,Ring()
exten => s,n,Answer()
exten => s,n,Echo()
```

El contexto significa que al llegar una llamada por la línea telefónica análoga, conectada a un puerto FXO, será contestada y todo aquello que llega como sonido es regresado a quien llama, es decir nuevamente un eco. Dicho de una manera más práctica, un teléfono perteneciente a la red telefónica del Cinvestav, digamos la extensión 6556, llama a la línea telefónica que fue conectada a alguno de los puertos FXO, la llamada será contestada automáticamente por el servidor Asterisk y se producirá el esperado eco.

El contexto `incoming` aquí definido no proporciona algún beneficio y sólo funciona como muestra del funcionamiento. Se podría realizar un plan de marcado más complejo, que pudiera contestar las llamadas entrantes en los módulos FXO y mediante la implementación de un IVR con las aplicaciones, funciones y AGIs que provee Asterisk, poder entre muchas otras cosas, enlazar a las extensiones del contexto `internal`.

4.6 Arquitectura de intercomunicación

La arquitectura de intercomunicación para la plataforma VoIP se muestra en la figura 4.4. En ella se puede observar una convergencia IP. El unir el dominio de IP

junto con el dominio de la red telefónica PSTN es un claro ejemplo. Las dos redes se diseñaron con diferentes propósitos, una de ellas para proveer servicios de telefonía y la otra fue diseñada para proveer el transporte de datos sin importar su tipo.

Entre los cuatro extremos visibles en la figura, se pueden enlazar llamadas telefónicas, por ejemplo, de un teléfono de la PSTN a cualquier *softphone*/IP-*phone* conectado ya sea alámbricamente o inalámbricamente y viceversa. El servidor Asterisk desempeña el trabajo de una puerta de enlace entre los diferentes dominios.

La figura también presenta de forma parcial el trapecio SIP que se definió en la sección 2.1.2. Asterisk es un servidor *proxy* al enlazar a los usuarios e incluso los flujos de audio son dirigidos al servidor Asterisk.

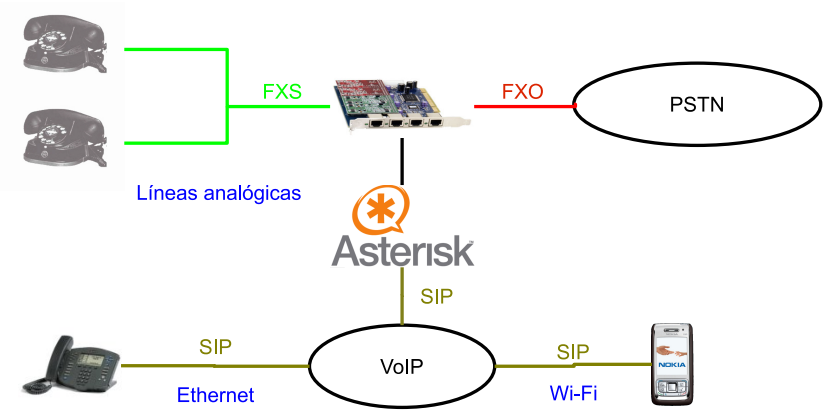

Figura 4.4: Interconexión de los dispositivos

Aún teniendo la parte medular de la plataforma, no podría estar completa sin contar con aplicaciones basadas en SIP que interactúen en la plataforma. A pesar de que ya se ha mostrado el funcionamiento de la plataforma con *softphones* ya desarrollados para sistemas operativos específicos, se requiere de un contacto directo en el desarrollo de un *softphone*.

4.7 Biblioteca PJ

Las bibliotecas PJ ofrece interoperabilidad entre varias arquitecturas al realizar una abstracción del sistema operativo, resultando muy conveniente para el propósito

de este tema de investigación. El primer paso para poder explotar esta biblioteca es saber y conocer como portar el código a los sistemas embebidos y no embebidos. En las siguientes secciones se reporta como compilar la biblioteca para los diferentes sistemas operativos presentes en la plataforma.

En un inicio se decidió utilizar **Python** como lenguaje de programación para el desarrollo de aplicaciones basadas en SIP, dado que la biblioteca PJ lo soporta. El lenguaje de programación Python se basa en *scripts*, que resultan portables entre diferentes sistemas operativos, con la condición de que las bibliotecas importadas se encuentren disponibles para dichos sistemas. El beneficio mayor es contar con un mismo *script* que se interpreta en varios sistemas operativos.

Se probó el funcionamiento de la biblioteca PJ con el módulo de Python en los sistemas operativos Linux y Windows, el resultado fue satisfactorio. Pero para sistemas embebidos aún contando con Python PyS60 y PythonCE (para Symbian y Windows Mobile respectivamente), la API no se encuentra portable, requiere de un desarrollo que se encuentra fuera del alcance de este proyecto de investigación.

El módulo de Python resulta muy conveniente para realizar programas demostrativos, con un desarrollo relativamente rápido, a diferencia de un desarrollo en un lenguaje de programación como lo es C o C++. Pero para los sistemas embebidos como Symbian o Windows Mobile, se requiere de una compilación individual de las bibliotecas PJ para dichas arquitecturas y comprobar su funcionamiento en una aplicación. Afortunadamente, la biblioteca cuenta con varios ejemplos de aplicaciones para probar la funcionalidad de la biblioteca en varias arquitecturas. Para los sistemas operativos como Symbian o Windows Mobile se encuentra sólo un par de aplicaciones de ejemplo.

Las bibliotecas PJ se encuentran distribuidas en un solo árbol de código fuente llamado PJPROJECT [50]. Tales bibliotecas se pueden obtener al descargarlas en su edición *tarball* (un archivo comprimido) o al obtenerlo del troncal *Subversion* (un repositorio de código fuente que se encuentra en constantes actualizaciones).

Descargar el código desde el repositorio da la ventaja de tener el código fuente más actualizado, el cual cuenta con menos errores que la versión *tarball*, al encontrarse en constante desarrollo. Además, se puede actualizar el código descargado desde el troncal al ejecutar `svn update` en cualquier momento.

A lo largo del desarrollo del trabajo de investigación, se ha mantenido en constante actualización el código fuente de la biblioteca. Se comenzó a utilizar cuando la biblioteca se encontraba en su versión 0.8 y al momento de la escritura de esta sección se encuentra en vísperas de la versión 1.0.

Aún con la gran cantidad de información acerca de las bibliotecas, se requirió realizar el registro en la *listas de correos* del proyecto. En las listas de correos se pueden hacer preguntas directas con base al funcionamiento, reportar errores e incluso dar sugerencias respecto a la biblioteca. Durante casi un año de formar parte de la listas de correos se han planteado dudas propias así como el reporte de posibles *bugs*. Se podría estimar que por semana se tienen decenas de correos.

4.7.1 Compilar PJ para Linux

Se obtiene el código fuente desde el troncal *Subversion* y en una terminal ejecutar:

```
svn co http://svn.pjsip.org/repos/pjproject/trunk pjproject
```

El código se encuentra en el directorio `pjproject`. Una vez obtenido el código, se cumplen con las dependencias, requiriéndose las siguientes herramientas:

- GNU make
- GNU binutils
- GNU gcc
- GNU autoconf

Se realizan las siguientes acciones para compilar los ejemplos y las bibliotecas:

```
$ cd pjproject
$ ./configure --disable-oss
...
```

Se da la opción de construir a la medida ciertas características, para ver tales opciones se ejecuta `./configure --help`. Una de las opciones que se ha utilizado es `--disable-oss`, para no incluir el sistema de sonido OSS y por omisión utilizar el sistema de sonido ALSA. Esto es debido a conflictos al utilizar una comunicación *fullduplex* en el sonido (micrófono y bocinas). Para poder compilar con el sistema ALSA es necesario tener la biblioteca de desarrollo `libasound2-dev` instalada en la distribución.

Una vez completada la configuración, se procede lo siguiente:

```
$ cd pjproject
$ make dep
$ make
```

Ahora se cuenta con todas las bibliotecas y ejemplos compilados. Se realizó una prueba de ejecución del programa `pjsua-i686-pc-linux-gnu`, se configuró una cuenta que se conecta con Asterisk y se pudo realizar una llamada.

Para crear el módulo *py_pjsua* para Python se realizó lo siguiente [51]:

- Ir al directorio `pjsip-apps/src/py_pjsua`.
- Ejecutar `python ./setup.py build`.
- El módulo Python se guardó en el directorio `build` que se encuentra en el directorio actual.

- Alternativamente ejecutar `python ./setup.py install` para instalar el módulo en el directorio `site_packages` de Python.

De igual forma se configuró una cuenta que se conecta con Asterisk y se pudo realizar una llamada desde el script de Python ejemplo `pjsua_app.py`.

Los programas aquí compilados deben ser ejecutados en consola y no hay una Interfaz Gráfica de Usuario (GUI). Son programas muy básicos e incluso la notificación de una llamada entrante es por medio de la consola en modo texto, es decir, no hay un tono de llamada entrante.

4.7.2 Compilar PJ para Windows

Se deberá descargar el código fuente del troncal *Subversion* del proyecto PJ con el programa `svn` específico para Windows. Esta tarea es realizada de la misma manera que en Linux ejecutando el comando `svn` en el programa `cmd` de Windows.

El proyecto PJ brinda archivos de proyecto para la herramienta Visual Studio. Se utilizó la herramienta *Visual Studio* 2005 *Express* y se logró la compilación del proyecto e incluso se compiló el módulo de Python para Windows. Como se explicará en la siguiente sección, resulta más conveniente utilizar *Visual Studio* 2005 *Pro*.

Para obtener las bibliotecas y programas ejemplo para Windows se instala lo siguiente:

- *Platform* Kit de Desarrollo de Software (SDK) para Windows XP con Service Pack 2
- DirectX SDK 2007
- Python
- *Visual Studio* 2005 *Pro*

Se agregan los `paths` necesarios de las bibliotecas *Platform* SDK, DirectX SDK y Python en el *Visual Studio* 2005 *Pro*.

Para compilar las bibliotecas/aplicaciones se realiza:

- Abrir desde VS2005 el archivo `pjproject-vs8.sln`, encontrado en el directorio raíz del proyecto.
- Elegir `pjsua` como proyecto activo.
- Seleccionar *Debug* o *Release build*.
- Construir el proyecto. Esto creará la aplicación `pjsua` y todas las bibliotecas utilizadas por `pjsua`.
- Después de una construcción exitosa, la aplicación `pjsua` se encuentra en el directorio `pjsip-apps/bin` y las bibliotecas en el directorio `lib` de cada proyecto.

Para compilar el resto de los ejemplos:

- Seleccionar el proyecto `samples` como proyecto activo (en el mismo proyecto que el anterior).
- Seleccionar *Debug* o *Release build*.
- Construir el proyecto. Esto compilará todas las aplicaciones ejemplo y las bibliotecas necesarias.
- Después de una compilación exitosa, las aplicaciones ejemplo se encuentran en el directorio `pjsip-apps/bin/samples` y las bibliotecas en el directorio `lib` de cada proyecto.

Para compilar el módulo py_pjsua de Python para Windows:

- Abrir desde VS2005 el proyecto `pjsip-apps.dsw` que se encuentra en el directorio `pjsip-apps\build`.
- Seleccionar el proyecto `py_pjsua`.
- Compilar el proyecto.
- El módulo será guardado en el directorio `pjsip-apps\lib`.

Es requisito indispensable tener instalado Python en Windows.

Al igual que en Linux, se probaron los ejecutables y se cuenta con la misma funcionalidad que los programas en Linux.

4.7.3 Compilar PJ_SIP para Familiar Linux

Hay dos maneras de compilar software para este tipo de sistemas operativos que son embebidos, uno de ellos es compilar la aplicación directamente en el dispositivo, requiriéndose tener instalado el compilador y las bibliotecas a utilizar. Esto no siempre es fácil de lograr, una de las principales limitaciones es el espacio disponible en el dispositivo. El dispositivo en cuestión es una iPAQ 5400 que cuenta con 64 MB en RAM y 64 MB para almacenamiento, realmente no es candidato ya que un ambiente de desarrollo requiere quizá de GBs de memoria disponible. EL Asistente Personal Digital (PDA) tiene un sistema operativo Familiar Linux con GUI GPE (basado en gnome).

La otra manera de lograr la compilación es a través de una compilación "cruzada" que se realiza con el llamado *toolchain* [63]. El encontrar el *toolchain* apropiado no resulta ser fácil, más aún cuando no hay experiencia en tal cuestión.

Se presentan en [55] [60] [56] [35] [58] algunas de las formas para realizar una compilación cruzada, alguna de ellas fue probada pero el compilado no era compatible con la plataforma. Al compilar el ejemplo básico "*hola mundo*" no se ejecutaba en la

consola del PDA. El tutorial que resulta ser el apropiado es encontrado en [43], que realiza la compilación cruzada sobre un Linux como sistema anfitrión para obtener compilados para el sistema embebido Familiar Linux.

Los pre-requisitos en cuanto al software es tener instalado `Git` y `Zenitty` principalmente, y se obtiene una copia del árbol de desarrollo de *Familiar Linux* al ejecutar el siguiente comando:

```
$ git clone http://familiar.handhelds.org/git/familiar-build.git
```

Una vez descargado el árbol de desarrollo se ejecuta un *script*, que a manera de diálogos GUI, configura el árbol de desarrollo:

```
$ cd /path/to/familiar-build
$ sh setup/build-env.sh
```

Acto seguido se ejecutan las siguientes instrucciones en consola:

```
$ cd /path/to/newly/created/build/directory
$ source conf/env.sh
$ bitbake nano
```

El primer comando sitúa la consola sobre el *path* adecuado. El segundo comando inicializa algunas variables de entorno y el último desarrollará un paquete instalable en *Familiar Linux*. En este ejemplo, al ejecutar el tercer comando y al ser ejecutado por primera vez, se realiza la descarga del código fuente necesario para crear el *toolchain*. Posteriormente se descarga el código fuente y sus dependencias de la aplicación `nano` (la aplicación `nano` es como un editor `vi` de menor tamaño). Se compila el código fuente de la aplicación `nano` con el *toolchain* y una vez que termina la compilación se crea un paquete instalable en el PDA mediante el programa `ipkg`.

Aún no hemos cumplido nuestro objetivo, para compilar el proyecto PJ basta situarse en el código fuente del mismo, utilizar el *toolchain* antes generado y compilar la biblioteca PJ.

El *toolchain* consta básicamente de un compilador gcc o g++ y de las bibliotecas básicas del sistema operativo *Familiar Linux*. Una vez que se tiene identificado el *path* tanto del compilador como de las bibliotecas, queda por ejecutar la siguiente instrucción para configurar la compilación cruzada de la biblioteca PJ:

```
./configure --host=arm-linux --build=i686-linux
--with-lib-path=/home/cross/familiar-build/build-h3900-gpe/tmp/cross/
arm-linux/lib LDFLAGS=-Wl,-elf2flt CFLAGS=-O2
```

Un requisito indispensable, antes de ejecutar el comando anterior, es asegurarse que el compilador del *toolchain* se encuentre en la variable `PATH`, si no fuese el caso la siguiente línea serviría para agregar el *path* en la variable:

```
$ export \
PATH=~/familiar-build/build-h3900-gpe/tmp/cross/arm-linux/bin:$PATH
```

Nótese que fue necesario utilizar algunas de las opciones del comando `configure`. Esto es uno de los puntos clave para la compilación cruzada ya que declara de cierta manera el *toolchain*. Posteriormente ejecutar "`make dep`" y "`make`" compilará todo el proyecto, obteniendo como producto final la compilación de las bibliotecas y de los ejecutables para la plataforma Familiar Linux.

Al realizar pruebas de ejecución de los programas ejemplo en el dispositivo, se tiene un problema identificado con el audio. El programa `pjsua-arm-unknown-linux-gnu` es configurado y ejecutado en la PDA, el programa se registra en Asterisk y cuando se realiza la llamada, la aplicación hace uso del dispositivo de audio y es cuando el programa es suspendido.

El problema resulta por la manera en que la biblioteca PJ hace uso del dispositivo de audio del PDA. Una llamada telefónica involucra tanto la captura de audio por un micrófono como el despliegue de audio en el auricular. Aún cuando se puede realizar la captura de audio y tocar audio por dos programas diferentes ejecutados al mismo tiempo en el PDA, no se puede abrir el dispositivo de audio para lectura y escritura simultáneamente por un programa, o al menos así no fue permitido.

El software del sistema de audio utilizado en el PDA es OSS. La versión del kernel en la PDA es la 2.4, que resulta ser bastante atrasado a los kernels encontrados para PC (por ejemplo, la versión actual de *Ubuntu Linux* resulta ser la 2.6.24). Para la versión del kernel Linux 2.6 se utiliza el sistema de audio ALSA, aunque por omisión el sistema de audio en las versiones recientes de *Ubuntu* es *Pulseaudio*.

Aún no se tiene una solución para enfrentar el problema de audio. Sin embargo, realizando un análisis de la necesidad de hacer funcionar el audio en PJ en el PDA, se opta por dejar en este punto la investigación respecto a Linux Familiar, con la confianza en que el problema se supere a la llegada de nuevas tecnologías o mejoras en el software.

El aporte mayor de esta sección es haber compilado la biblioteca PJ para un sistema embebido que es totalmente de software libre. El trabajo aquí presentado puede reutilizarse para futuras investigaciones.

4.7.4 Compilar PJ_SIP para Windows Mobile

En un principio se trabajó con la herramienta *embedded Visual Studio* 4.0 con el SDK Windows Mobile. Desafortunadamente *embedded Visual Studio* no es compatible con el SDK de Windows Mobile 5.0.

Por otro lado, la herramienta *Visual Studio* 2005 *Express* no es candidato para el SDK Windows Mobile 5.0, así que se requirió instalar *Visual Studio* 2005 *Pro*.

La biblioteca PJ es compatible con Windows Mobile. Una aplicación de AU SIP (con media) simple (*softphone* básico), llamada *WinCE* es proveída sólo como una prueba del concepto de que la biblioteca es portable.

La PDA HP iPAQ hw6945 cuenta con el sistema operativo Windows Mobile 5.0, por tal razón es necesario probar la funcionalidad de la biblioteca PJ en este dispositivo.

El software requerido en una computadora con Windows que permita la compilación de programas para Windows Mobile 5.0 es el siguiente:

- Visual Studio 2005 Pro.
- Active Sync 4.5 (requerido por el Windows Mobile SDK).
- Windows Mobile 5.0 Pocket PC SDK

Para compilar el programa ejemplo WinCE se realiza lo siguiente:

- Abrir el archivo `pjsip-apps/build/wince-evc4/wince_demos.vcw`, el archivo será convertido apropiadamente al utilizar VS2005.
- Seleccionar el proyecto `pjsua_wince` como proyecto activo.
- Seleccionar el SDK apropiado, en este caso `Windows Mobile 5.0 Pocket PC SDK`
- Seleccionar la configuración apropiada, *debug* o *release*.
- Seleccionar la configuración apropiada, *device* o *emulator*.
- Construir el proyecto. Esto creará la aplicación ejemplo WinCE y las bibliotecas que requiere la aplicación.

Al completarse el procedimiento anterior, el directorio `Windows Mobile 5.0 Pocket PC SDK (ARMV4I)` es creado bajo la ruta `pjsip-apps\src\pjsua_wince`. Dependiendo de haber seleccionado una versión *Release* o *Debug*, el ejecutable se encontrará en un directorio con el mismo nombre bajo la ruta anteriormente mencionada.

Para instalar la aplicación creada en la PDA, se copia el ejecutable en algún directorio en la PDA. Los directorios son visibles al conectar la PDA a una PC por medio del programa ActiveSync.

La configuración de la cuenta de usuario para poder conectarse a Asterisk es configurada a nivel código fuente de la aplicación WinCE; los parámetros como usuario, password, etc. son configurados a nivel macros del código fuente. Estas macros se encuentran en el archivo `pjsua_wince.cpp`, quedando de la siguiente manera:

```
#define SIP_PORT        5060             // puerto SIP
// URI a llamar por omisión
#define SIP_DST_URI "sip:echo@192.168.201.140:5061"
#define HAS_SIP_ACCOUNT 1                // habilita el registro
#define SIP_DOMAIN      "192.168.201.22" // IP del servidor Asterisk
```

```
#define SIP_REALM       "*"
#define SIP_USER        "wincewm"        // Usuario
#define SIP_PASSWD      "winc"           // Password
#define SIP_PROXY       "sip:192.168.201.22;lr" // El proxy
#define NAMESERVER      NULL
// no se requiere del soporte STUN
#   define STUN_DOMAIN  NULL
#   define STUN_SERVER  NULL
// no se requiere del soporte ICE
#define USE_ICE         0
```

Durante el periodo de la investigación se adquirieron un par de dispositivos móviles que cuentan con el sistema operativo Windows Mobile 6.0. Para obtener el ejecutable del ejemplo WinCE para esta plataforma, se desinstaló el SDK Windows Mobile 5.0 y se instaló el SDK de Windows Mobile 6.0. El nuevo SDK requiere de la previa instalación del paquete `Microsoft .NET Compact Framework v2 SP2`.

El ejecutable obtenido para Windows Mobile 6.0 es también compatible con el sistema operativo Windows Mobile 5.0.

Se realizó la prueba de eco y también se marcó a otro usuario en los dispositivos con Windows Mobile, la aplicación ejemplo funcionó adecuadamente. La aplicación cuenta con una interfaz GUI simple pero funcional.

4.7.5 Compilar PJ_SIP para Symbian

En trac.pjsip.org [57] se puede encontrar un procedimiento paso a paso, para poder lograr compilar una aplicación PJ e instalarla en el teléfono celular Nokia E65, que cuenta con sistema operativo Symbian S60 tercera edición.

Previamente PJSIP no soportaba la herramienta de desarrollo **Carbide C++** para Symbian, dado que Carbide no importa archivos MMP de manera apropiada. Con el lanzamiento de Carbide C++ 1.2 es posible construir proyectos PJ.

El software y hardware necesario es el siguiente:

- Windows XP con Service Pack 2
- Symbian S60 3rd Edition Maintenance Release (MR)
- Carbide C++ 1.2, Developer Edition
- Nokia PC Suite versión 6.85
- Dispositivo Nokia S60 3ra edición, Nokia E65
- Cable de datos Nokia
- ActivePerl 5

Una vez instalado y habiendo seguido el tutorial [57] se ha compilado un ejemplo en modo *debug*. El ejecutable no presenta GUI y el funcionamiento es a nivel consola con un manejo muy pobre de la aplicación. Al igual que en Windows Mobile el usuario es configurado a nivel macros en el código fuente. Al inicio de pruebas del ejecutable instalado en el dispositivo, no se obtenía sonido ni de entrada ni de salida, aún cuando el programa se registraba en Asterisk. Después de varios días de la búsqueda del error, se realizó una actualización del código fuente (ejecutar `svn update`) del proyecto PJ. Tras dicha actualización el ejecutable funcionó adecuadamente.

Para poder trabajar en modo *debug* se requiere pagar una licencia. Se cuentan con 21 días de licencia libre.

4.7.6 Virtualización de desarrollo

Al utilizar varias plataformas a la vez, resulta conveniente trabajar sobre una misma computadora. Esto resulta posible al utilizar herramientas que virtualizen un sistema operativo. Se decidió utilizar el sistema operativo Linux, virtualizando dos sistemas operativos Windows mediante la herramienta `VmWare` [64].

Se utiliza un sistema operativo Windows virtualizado para el desarrollo de aplicaciones Windows Mobile y otro sistema operativo Windows virtualizado, para el desarrollo de aplicaciones Symbian. Como ya se ha visto, desarrollar aplicaciones para Linux y *Familiar Linux* es realizado bajo el sistema operativo nativo (Linux) de la computadora de desarrollo.

Se ha probado este esquema y los dispositivos conectados al sistema operativo Linux pueden traspasarse a las máquinas virtuales, es decir, los dispositivos son reconocidos por la máquina virtual y los programas como `ActiveSync` y `Nokia PC suite` funcionan adecuadamente en el ambiente virtual. También los emuladores de Symbian y de Windows Mobile se hicieron funcionar sobre las máquinas virtuales.

La configuración y los pasos a seguir para tener tal virtualización de desarrollo no se incluirán en este trabajo. La inclusión de la sección es para demostrar y anunciar que es posible este tipo de desarrollos en ambientes virtuales.

4.7.7 Desarrollo de una aplicación VoIP

El desarrollo hasta ahora descrito ha tomado gran parte del tiempo para el término de la tesis. Razón suficiente para no iniciar una elaboración de un *softphone* completo que incluya además de una Interfaz Gráfica de Usuario (GUI) amigable, otras características como de presencia y mensajería.

Por otro lado, se desarrolló un *softphone* con el requisito de contar con una GUI, que tenga la habilidad de configurar un usuario SIP y que pueda hacer y recibir llamadas VoIP.

La aplicación se desarrolla a partir del *script* no GUI del proyecto PJ para Python. Dicho *script* utiliza el módulo `py_pjsua`, que contiene la API PJSUA. El *script*

ejemplo que viene en el proyecto demuestra funciones básicas pero no implementa un *softphone* completo.

En *Python* existe la biblioteca **PyGTK** que es utilizada para la elaboración de la GUI. PyGTK es también portable como *Python* y para el entendimiento de la biblioteca se revisó el tutorial [42].

El desarrollo se hizo bajo el sistema operativo Linux y mediante una propia instalación de Python y PyGTK en Windows, el *script* es también ejecutable en Windows. El nombre del *script softphone* es `mysp.py`.

El GUI se agregó como una clase en el *script* básico, de manera que una vez que se tiene la funcionalidad básica del *softphone* en tiempo de ejecución, se instancia un objeto de la clase GUI. Los eventos generados en el *softphone* serán transmitidos al objeto GUI y éste reflejará los eventos ocurridos, por ejemplo, el estado inicial del GUI mostrará un usuario no conectado, cuando el *softphone* se registra entonces se mostrará el estado conectado. Cuando un evento ocurre en el GUI, como cuando el usuario quiere realizar una llamada, el evento es transmitido al *softphone* por medio de alguna función.

En la figura 4.5 se exhiben las tres ventanas del *script* `mysp.py`. La ventana principal es la llamada *mysoftphone*. La ventana con nombre *Configuración SIP* es presentada ya sea al pulsar el botón que muestra la URI del usuario, mediante el menú `Archivo->Identificación` o al teclear la secuencia `Ctl+I`. En esta ventana se realiza la configuración del usuario SIP al estilo de la aplicación `kphone`.

La tercer ventana con nombre *Llamada saliente*, se muestra cuando se presiona el botón *CALL*, mediante el menú `Archivo->Nueva llamada` o cuando se teclea la secuencia `Ctl+N`. Para iniciar la llamada se escribe el contacto en el campo *Remoto* y se presionada el botón *Marcar*. Una forma más directa es escribir un contacto en el campo de entrada de texto de la ventana principal y presionar el botón *Call*. La SIP o SIPS URI del contacto a llamar, puede ser escrito de las siguientes maneras:

- 611
- sip:611
- 611@localhost
- sip:611@localhost

Una función se encargará de corregir o completar la SIP o SIP URI. Antes de los ":" son válidas las cadenas `sips` o `sip` y cualquier otra cadena será substituida por la cadena "sip". Después de los dos puntos y antes del "@" es válida cualquier cadena de caracteres. Si no existe un dominio especificado, se utiliza el mismo que se tiene configurado como registrador.

Cuando se tiene una llamada entrante en la aplicación, se presenta una ventana parecida a la de *Llamada saliente* pero con nombre *Llamada entrante*. Se muestra la SIP URI de quien llama y dos botones activos con etiquetas *Contestar* y *Rechazar*.

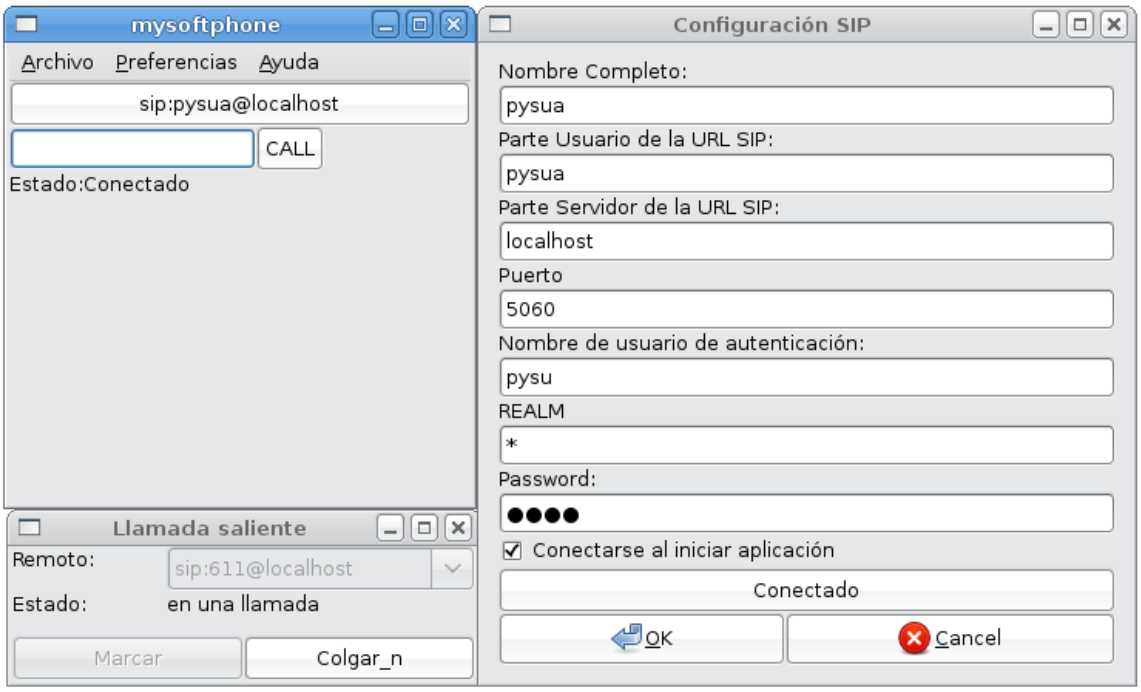

Figura 4.5: Ventanas del *softphone* en PyGTK

El *softphone* `mysp.py` agrega 2 tonos, uno de ellos utilizado como tono de marcado y el segundo como tono de llamada entrante.

La figura 4.5 refleja el estado del *softphone*, tras haber realizado el siguiente procedimiento: primero la ventana principal muestra un estado de *Conectado* porque se ha registrado con Asterisk. La ventana de *Llamada saliente* muestra un estado de *en una llamada* porque se ha marcado la extensión 611. Obsérvese que el botón *Marcar* se encuentra deshabilitado mientras que el botón *Colgar_n* está activo, si éste fuera presionado se estaría terminando la llamada.

A continuación se presentan los eventos en Asterisk, debido a la actividad que el *softphone* realizó en el procedimiento anterior.

```
Registered SIP 'pysua' at 127.0.0.1 port 5061 expires 300
   -- Saved useragent "PJSUA/Python 0.1" for peer pysua
   -- Executing [611@internal:1] Ringing("SIP/pysua-081e9428", "")
   -- Executing [611@internal:2] Answer("SIP/pysua-081e9428", " ")
   -- Executing [611@internal:3] Echo("SIP/pysua-081e9428", " ")
```

Los respuestas o notificaciones que ocurren en las transacciones SIP son reflejadas en el *callback* `on_call_state` que provee la API PJSUA. Se complementó este *callback* con varias definiciones de respuestas (véase sección 2.1.5) para obtener un *softphone* más funcional que el proveído por el ejemplo de la biblioteca PJ.

4.8 QoS en la plataforma

En seguida se listan los mecanismos de QoS con los que cuentan los dispositivos de la arquitectura:

- WRT54G: Punto de Acceso (PA) que tiene habilitado WMM
- SRW2008: DiffServ (Capa 3) y Clase de Servicio (capa 2)
- Servidor Asterisk: DiffServ (Capa 3)
- Nokia E65: soporte WMM
- iPAQ serie 200: soporte WMM

Al contar el WRT54G con la certificación WMM en la conexión inalámbrica, los dispositivos que cuenten con certificación WMM harán uso de este recurso de provisión de QoS. El conmutador SRW2008 cuenta con mecanismos para la provisión de QoS. Sólo resta que los paquetes que sean transmitidos se encuentren marcados apropiadamente en el campo DSCP según el tipo de flujo. De esta manera se aprovechará la priorización de paquetes y se tendrá un mejor tratamiento que el de *mejor esfuerzo* para los flujos de tiempo real como lo es RTP.

En la sección 3.3.7 se dice que para hacer uso de la funcionalidad WMM se requiere cumplir con:

1. El Punto de Acceso tiene la certificación WMM y tiene habilitado WMM.
2. El dispositivo dónde se ejecuta la aplicación deberá ser certificado WMM.
3. La aplicación fuente soporta WMM.

En las siguientes secciones se trata sobre la configuración de algunos de los dispositivos y de cómo se logra que una aplicación soporte WMM para proveer QoS.

4.8.1 Conmutador SRW2008

El conmutador SRW2008 cuenta con dos mecanismos principales de QoS, uno de ellos funciona sobre la capa 2 del modelo OSI. Dado que esta capacidad está presente principalmente en redes VLANs, no será utilizada. El otro mecanismo funciona sobre la capa 3 del modelo OSI, al observar el código de punto que se encuentra en el campo DSCP del paquete, se determina a cuál de las 4 colas se despachará dicho paquete. El conmutador cuenta con 4 colas de servicio en hardware, planificadas ya sea por *Prioridad Estricta* o por *Peso Round Robin*. Las colas se encuentran numeradas del 1 al 4, donde la cola número 1 es la de menor prioridad mientras que la cola número 4 es la de mayor prioridad [19].

El conmutador está configurado con la planificación de *Prioridad Estricta* en las colas, así los paquetes pertenecientes a la cola 4 serán despachados antes que los paquetes de otra cola. El mapeo de los 63 valores posibles de DSCP de los paquetes entrantes en el conmutador hacia las 4 colas se muestra en la tabla 4.1:

Intervalo DSCP	Cola
0-15	1
16-31	2
32-47	3
48-63	4

Tabla 4.1: Mapeo DSCP a las 4 colas del conmutador

4.8.2 Actualización del firmware para el WRT54G

El dispositivo Linksys WRT54G es una puerta de enlace Wi-Fi y es capaz de compartir la conexión a Internet entre varias computadoras, vía puertos 802.3 Ethernet y 802.11b/g Wireless. El WRT54G es notable al ser el primer dispositivo comercial que ha liberado el código fuente del *firmware* para satisfacer las obligaciones de la licencia GNU GPL. Esto permite a los programadores modificar el *firmware* para cambiar o agregar funcionalidad al dispositivo [62].

Uno de estos proyectos es el llamado DD-WRT, siendo un *firmware* basado en Linux para varios encaminadores inalámbricos, el más notable es el WRT54G de Linksys. La licencia utilizada se encuentra bajo GNU GPL versión 2.

Varias de las funcionalidades con las que cuenta DD-WRT no son incluidas en el *firmware* original del encaminador (el proveído por su fabricante). Estas características incluyen soporte para redes *Kai*, servicios basados en *demonios*, IPv6, Sistema de Distribución Inalámbrica, RADIUS, QoS avanzada, capacidad de incrementar la frecuencia del reloj y soporte de software para modificación del hardware de la tarjeta SD [61].

Dada una mayor funcionalidad encontrada en el *firmware* DD-WRT, se instaló dicho *firmware* en el encaminador WRT54G. El *firmware* original del dispositivo tiene la opción de habilitar o deshabilitar el soporte WMM, pero no brinda un método para cambiar los valores de los parámetros que definen las Categorias de Acceso (CAs). El *firmware* DD-WRT provee dichos parámetros y pueden ser modificados, con el propósito de cambiar la prioridad de las CAs. Esta funcionalidad se utiliza en las pruebas del medio inalámbrico.

El software y guías para instalar se encuentran en la página del proyecto [39], para conocer y saber si el *firmware* es instalable en el dispositivo en cuestión véase [40]. Una vez reconocido el dispositivo y el *firmware* apropiado se realiza un proceso llamado reprogramación (*flashing* en inglés).

Se deberá tener mucho cuidado a la hora de elegir el *firmware* y seguir el procedimiento. Fallar en tales aspectos podría significar la avería permanente del dispositivo.

4.8.3 Servidor Asterisk

El servidor Asterisk cuenta con QoS, al etiquetar los paquetes salientes. En el archivo de configuración `sip.conf` se definen de manera global los códigos de punto que deberán marcarse en los paquetes. Se definen 3 tipos de flujos en Asterisk, SIP, audio y video. Para cada uno de estos flujos se define el DSCP por omisión de CS3, EF y AF41 respectivamente.

En nuestra plataforma se elige utilizar el DSCP $CS7 = 111000_2$ ($ToS = 0xE0$) para el flujo de audio, los paquetes RTP y RTCP son marcados con este número y el DSCP $CS5 = 101000_2$ ($ToS = 0xA0$) para los paquetes SIP.

Los paquetes marcados con la Clase de Servicio 7 irán a la CA de voz en el PA y en el conmutador irán a la cola 4. En ambos dispositivos estos paquetes tendrán la mayor prioridad para su transporte.

Los paquetes SIP irán a la CA de video en el PA y en el conmutador irán a la cola 3. El nivel de prioridad dado a SIP es mayor que el flujo sin clasificar pero de menor prioridad que el flujo de audio. Resulta importante brindar este nivel de prioridad a los paquetes SIP ya que este protocolo se encarga del inicio, manejo y término de las sesiones. Cuando existe una carga excesiva en el ancho de banda, la calidad del audio de las llamadas VoIP es severamente afectada, debido a sus requerimientos de entrega de paquetes. Bajo estas condiciones, los AU podrían realizar una re-negociación en el codificador a utilizar, acordando por medio de SIP y SDP un codificador que utilice un menor ancho de banda. Si estos mecanismos no se logran o se retrasan, se reflejaría directamente en una mala calificación de QoS percibida.

Resulta entonces que Asterisk es una aplicación que da soporte WMM al categorizar los paquetes. Pero sólo tiene la posibilidad de marcar los paquetes salientes, más no los entrantes. Para que los paquetes entrantes en el servidor Asterisk se encuentren marcados, se modifica ligeramente la biblioteca PJ para proveer QoS basado en DiffServ en las aplicaciones.

4.8.4 Marcado de paquetes por la aplicación

Esta sección trata sobre como proporcionar a la biblioteca PJ los mecanismos basados en DiffServ para la provisión de QoS. Esta provisión consiste en marcar los paquetes SIP, RTP y RTCP con un número DSCP adecuado. Mediante el marcado, se cumple con la tercer condición para obtener la funcionalidad de WMM.

Plataformas Linux y Windows

Teóricamente, se puede realizar el marcado del campo DSCP con la función `setsockopt` tanto en Windows XP como en Linux, al utilizar sus bibliotecas per-

tinentes (en Linux es `sys/socket.h`).

Bajo la pregunta de porqué la biblioteca no realiza un marcado DSCP o más especificamente del campo Tipo de Servicio (ToS), en la lista de correos del proyecto, la respuesta del desarrollador de la biblioteca Benny Prijono [53] fue la siguiente:

"*Lo he considerado, pero el problema de situar el ToS con setsockopt no parece funcionar en Windows (de acuerdo con MSDN dicho mecanismo ha sido desaprobado), así que no resulta una manera directa para dar soporte a esto. Sin mencionar otras plataformas como lo es Symbian. Debido a esto, creo que no podemos hacer ToS a nivel pila por el momento.*"

En Linux, el asignar un código de punto como *CS7* requiere que la aplicación se ejecute con permisos de super usuario. En Windows XP se requiere incluir una entrada al registro de tipo DWORD con nombre y valor `DisableUserTOSSetting=0` sobre la siguiente llave de registro [38]:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

Mediante la búsqueda de como situar la función `setsockopt` apropiadamente para marcar los paquetes SIP, RTP y RTCP cuando se utiliza la API PJSUA, se encontró que después de crear los *sockets* que transportan los paquetes antes mencionados, resulta conveniente indicar a los *sockets* marcar el campo ToS.

El manual de Linux de la función `setsockopt` define una sinopsis como sigue:

`int setsockopt( int` *socket*`, int` *nivel*`, int` *nombre_opción*`, const void *` *valor_opción*`, socklen_t` *long_opción*`);`

La función `setsockopt` fija la opción especificada por el argumento *nombre_opción*, en el nivel de protocolo especificado por el argumento *nivel*, al valor apuntado por el argumento *valor_opción*, para el `socket` asociado con el descriptor de archivo especificado por el argumento *socket*.

Se utiliza la función `pj_sock_setsockopt` que abstrae a la función real `setsockopt` dependiente de la plataforma. El siguiente código muestra como se define el valor ToS en un *socket* para la biblioteca PJ.

```
tos = 0xE0;   //Valor ToS
// Se fija el valor ToS en el socket
status = pj_sock_setsockopt(sock[0], pj_SOL_IP(), pj_IP_TOS(),
        &tos, sizeof(tos));
// Se verifica el valor retornado por la función pj_sock_setsockopt
if (status != PJ_SUCCESS) {
    char errmsg[PJ_ERR_MSG_SIZE];
    pj_strerror(status, errmsg, sizeof(errmsg));
    PJ_LOG(4,(THIS_FILE, "Error setting RTP sock TOS=%X: %s [%d]",
    tos, errmsg, status));
}else{
```

```
    PJ_LOG(4,(THIS_FILE, "RTP sock TOS=%X DSCP=%X", tos, tos>>2));
}
```

Los *sockets* de RTP y RTCP se crean en el archivo `pjsua_media.c` con *path* relativo al directorio raíz del proyecto PJ `pjsip/src/pjsua-lib` en la función `create_rtp_rtcp_sock`. Para el *socket* de SIP se agrega el código en el archivo `sip_transport_udp.c` con ruta relativa `pjsip/src/pjsip` en la función `udp_set_socket`. Tómese en cuenta que el valor de la variable `tos` cambiará a $0xA0$ para el *socket* de SIP.

El proyecto es nuevamente compilado para aplicar los cambios. La biblioteca PJSUA se encuentra ahora con QoS DiffServ.

Las pruebas en Linux muestran los paquetes salientes de SIP, RTP y RTCP están marcados en su campo ToS con los valores correspondientes antes mencionados. Se capturaron los paquetes con el programa *wireshark* (un analizador de protocolos de red) y se verificaron los valores del campo ToS.

Con las modificaciones al código fuente, se compilaron nuevamente los programas ejemplo en Windows y tras ejecutar el programa `pjsua_vc8d.exe` se observa que aparentemente se fijaron los valores de ToS en los *sockets* (no hubo error). Desafortunadamente el proyecto PJ utiliza la biblioteca *winsock2.h* para el soporte de *sockets* en Windows y la solución propuesta para habilitar el marcado de paquetes en el campo ToS (la que agrega una entrada en el registro) funciona cuando se utiliza la biblioteca `winsock.h`. Cuando se verificó el marcado con el programa `wireshark` el ToS resultó ser 0.

Windows Mobile

En el SDK de Windows Mobile 5.0 no se cuenta con la habilidad de marcar paquetes en el campo ToS, pero en el SDK de Windows Mobile 6.0 si es posible. WMM es una tecnología reciente y obligó a que Windows Mobile 6.0 incorporara una manera de poder marcar los paquetes con la función `setsockopt`.

Las modificaciones que se realizaron en la biblioteca PJ para que la aplicación ejemplo WinCE fijara el marcado fueron las siguientes.

En el archivo *sock_bsd.c* bajo la ruta *pjlib\src\pj* se definen las siguientes constantes:

```
#ifdef IPPROTO_IP
const pj_uint16_t PJ_IPPROTO_IP = IPPROTO_IP;
#else
const pj_uint16_t PJ_IPPROTO_IP = 0xFFFF;
#endif

#ifdef IP_DSCP_TRAFFIC_TYPE
const pj_uint16_t PJ_IP_DSCP_TRAFFIC_TYPE = IP_DSCP_TRAFFIC_TYPE;
```

```
#else
const pj_uint16_t PJ_IP_DSCP_TRAFFIC_TYPE = 100 ; //valor en Winsock.h
#endif
```

De esta manera las constantes `PJ_IPPROTO_IP` y `PJ_IP_DSCP_TRAFFIC_TYPE` podrán utilizarse en el archivo que se mencionará en el siguiente párrafo. La biblioteca que utiliza la aplicación para los *sockets* es `winsock2.h` y ésta no define la macro `IP_DSCP_TRAFFIC_TYPE` pero si la macro `IPPROTO_IP`. El valor de `PJ_IP_DSCP_TRAFFIC_TYPE=100` se obtuvo al observar que en la biblioteca `winsock.h` se da ese valor a la macro.

Se agrega el siguiente código en el archivo `sock.h` que se encuentra en la ruta `pjlib\include\pj`.

```
extern const pj_uint16_t PJ_IPPROTO_IP;
extern const pj_uint16_t PJ_IP_DSCP_TRAFFIC_TYPE;

#if defined(PJ_DLL)
#else

#   define pj_IPPROTO_IP() PJ_IPPROTO_IP
#   define pj_IP_DSCP_TRAFFIC_TYPE() PJ_IP_DSCP_TRAFFIC_TYPE

#endif

 /* tipos de tráfico para servicios diferenciados */
typedef enum PJ_DSCP_TRAFFIC_TYPE
{
    PJ_DSCPTypeNotSet        = 0,
    PJ_DSCPBestEffort        = 1,
    PJ_DSCPBackground        = 2,
    PJ_DSCPExcellentEffort   = 3,
    PJ_DSCPVideo             = 4,
    PJ_DSCPAudio             = 5,
    PJ_DSCPControl           = 6,
    PJ_NumDSCPTrafficTypes   = 6
} PJ_DSCP_TRAFFIC_TYPE;
```

Los tipos de tráfico enumerados en la definición `PJ_DSCP_TRAFFIC_TYPE` se obtuvieron del encabezado *winsock.h*. Con este código podemos utilizar las macros definidas para ser utilizadas en la función `pj_sock_setsockopt` como se muestra a continuación.

```
int tos = PJ_DSCPAudio;
status = pj_sock_setsockopt(sock, pj_IPPROTO_IP(),
    pj_IP_DSCP_TRAFFIC_TYPE(), &tos, sizeof(tos));
```

```
if (status != PJ_SUCCESS) {
    char errmsg[PJ_ERR_MSG_SIZE];
    pj_strerror(status, errmsg, sizeof(errmsg));
    PJ_LOG(4,(THIS_FILE, "Error al fijar RTP TOS: %s [%d]", errmsg,
                status));
}else{
    PJ_LOG(4,(THIS_FILE, "Sin error al fijar RTP ToS"));
}
```

Al igual que en la sección anterior, para fijar el ToS en el `socket` de RTP y de RTCP se realiza en la función `create_rtp_rtcp_sock` del archivo `pjsua_ media.c`. Para el *socket* de SIP se agrega el código en el archivo `sip_transport_udp.c` en la función `udp_set_socket`. El *socket* de SIP se marcó con un tipo de tráfico `PJ_DSCPVideo`, los *sockets* de RTP y RTCP con el tipo de tráfico `PJ_DSCPAudio`, dichos valores mapean a valores de ToS $0xA0$ y $0xE0$ respectivamente. Se verificó el marcado de paquetes con `wireshark` [54] y fue así como se conoció el mapeo.

Cabe mencionar que este método, propuesto para marcar paquetes en Windows Mobile 6.0 se realizó a prueba y error.

Symbian

Para esta plataforma no se tiene implementada la función que abstrae a la función `setsockopt` en el proyecto PJ. La tarjeta inalámbrica del dispositivo Nokia E65 tiene la certificación WMM y la manera en se puede fijar el campo ToS es mediante la API *Symbian Socket*. En la referencia [48] se encuentra la siguiente información.

Para lograr fijar la CA de voz (AC_VO), los bits QoS DSCP del encabezado IP de paquetes deberán ser modificados. Esto puede ser logrado a nivel *socket* (RSocket) como sigue:

`iSocket.SetOpt(` *KSoIpTOS*, *KProtocolInetIp*, $0xE0$ `);`

El lograr integrar QoS en la aplicación de Windows Mobile 6.0 requirió de un largo proceso, lo que hace suponer que el brindar a la aplicación de Symbian con QoS un proceso igual o de mayores dimensiones. La provisión de QoS para Symbian quedará hasta este punto.

4.8.5 Hardware no incluido en la arquitectura

La tarjeta de red inalámbrica Intel Pro/Wireless 3945ABG en Windows XP cuenta con una actualización del software para ser un dispositivo certificado WMM.

En Linux, dicha tarjeta cuenta con los módulos *mac80211* (anteriormente llamado *d80211*) e *iwl3945* del proyecto **iwlwifi** [46]. El módulo soporta WMM basado en el valor IP ToS/DS [59]. Para instalar dichos módulos se requiere aplicar *patches* y

recompilar el Kernel. Una alternativa es instalar Ubuntu Linux 8.04 Hardy Heron con Kernel 2.6.24, versión liberada como estable el 24 de abril del 2008 y utiliza los módulos antes mencionados en su distribución. El módulo *iwl3945* es utilizado en las pruebas del soporte WMM en los siguientes capítulos.

Otra tarjeta inalámbrica que provee WMM tanto en Linux como en Windows XP es la Intel Wireless WiFi Link 4965AGN. Tanto ésta como la tarjeta de red inalámbrica Intel Pro/Wireless 3945ABG están asociadas a los procesadores Intel y se encuentran comúnmente en computadoras personales.

Capítulo 5

Planteamiento de las pruebas

El objetivo principal de las pruebas es apreciar el comportamiento de la conexión inalámbrica al transportar audio de voz. El control de acceso al medio inalámbrico juega un papel importante, al definir con cuánta frecuencia y por cuánto tiempo un dispositivo tiene la capacidad de transmitir información. Se realizan pruebas utilizando el Control de Acceso al Medio (CAM) dado en el estándar IEEE802.11 y el encontrado en el extracto WMM del estándar IEEE802.11e.

Además del flujo VoIP, se incorpora un flujo extra a manera de competir por el medio inalámbrico (aunque podrían ser vários tipos de flujos, el estudio de cada uno de éstos haría más complejo el análisis del comportamiento de los flujos). Esta competencia es normal en este tipo de redes ya sea de tipo alámbrica o inalámbrica, es habitual que viajen varios tipos de flujos a la vez. Para brindar una comunicación de voz de calidad, se requiere que dichos paquetes de voz lleguen a su destino íntegros y sin contra-tiempos, debido a sus requerimientos de tiempo real.

No se presentará un planteamiento ni análisis de Calidad de Servicio (QoS) DiffServ en el medio alámbrico. Dada la infraestructura, no se cuenta con tarjetas *Ethernet* gigabit en las computadoras, razón por la cual, las pruebas realizadas no saturaron al conmutador *Ethernet* gigabit. En dado caso, se estaría saturando la interfaz de red de las computadoras pero no así el conmutador. En caso de poder saturar el conmutador, se podría llegar a mostrar que el conmutador prioriza al tráfico marcado como de mayor prioridad sobre las restantes prioridades, y así el tráfico de tiempo-real cumpla con sus requerimientos.

5.1 Infraestructura de pruebas

A continuación se listan las características importantes de los equipos utilizados en las pruebas. Cabe mencionar que no necesariamente se utilizan todas las computadoras en todas las pruebas, esto se encuentra determinado según el tipo y el fin de la prueba:

Asterisk Procesador Intel Pentium D 2.80 GHz, tarjeta de red Ethernet de 100Mbps, 2GB de Ram y SO Ubuntu Linux Gutsy 7.10.

Shirl Procesador Core 2 Duo 2.0 Ghz, tarjeta de red inalámbrica Intel PRO/Wireless 3945ABG, 2GB y SO Ubuntu Linux Hardy 8.04.

Tama Procesador Core 2 Duo 1.8 Ghz, tarjeta de red inalámbrica Intel PRO/Wireless 3945ABG, 1GB y SO Ubuntu Linux Feisty 7.04.

C007 Procesador Core 2 Duo 1.6 Ghz, tarjeta de red inalámbrica Intel PRO/Wireless 3945ABG, 1GB y SO Ubuntu Linux Gutsy 7.10.

El Punto de Acceso (PA) cuenta con el firmware *DD-wrt*, con la propiedad de habilitar o deshabilitar el soporte WMM.

5.2 Función de los recursos

Las funciónes que desempeñan las computadoras mencionadas anteriormente en las pruebas es la siguiente:

Asterisk esta computadora es el servidor Asterisk y se encuentra directamente conectado por su interfaz de red *Ethernet* al Punto de Acceso (PA).

Shirl esta computadora obtiene una conexión de red inalámbrica al Punto de Acceso (al igual que Tama y C007). La función principal es iniciar y terminar llamadas VoIP hacia el servidor Asterisk. Esto es realizado por medio de un *softphone* capaz de manejar múltiples llamadas VoIP y marcar el campo DSCP en los paquetes salientes SIP y RTP.

Tama su función es generar un flujo de transferencia de archivos.

C007 su función es recibir el flujo de transferencia de archivos, generado por la computadora Tama.

5.3 Escenarios

Se plantean 5 escenarios para las pruebas.

no-WMM bajo este escenario no existe el soporte WMM. En la computadora `Shirl` que emite y recibe flujos VoIP utiliza el módulo *iwl3945*. El *softphone* no realiza el marcado DSCP, siendo el valor $DSCP = 0$ para los paquetes RTP. El Punto de Acceso (PA) tiene deshabilitado el soporte WMM. En la computadora donde se ejecuta Asterisk, no se realiza marcado DSCP para los paquetes salientes ($DSCP = 0$).

no-WMM con flujo al igual que el escenario no-WMM pero con un flujo de transferencia de archivos. En las computadoras se utiliza el módulo de Linux *ipw3945* (sin mecanismos para brindar WMM) y sólo en la computadora `Shirl`, que emite y recibe flujos VoIP utiliza el módulo *iwl3945*.

WMM en este escenario existe el soporte WMM en la computadora que emite y recibe flujos VoIP, utilizando el módulo *iwl3945*. El Punto de Acceso (PA) tiene habilitado el soporte WMM. El *softphone* realiza el marcado de paquetes SIP con valor $DSCP = 0x80 = 001000_2$ o $CS1$ y para los paquetes RTP el valor $DSCP = 0x38 = 111000_2$ o CS7. Tales valores son también utilizados en el marcado de paquetes salientes SIP y RTP en el servidor Asterisk.

WMM* con flujo igual que el escenario WMM pero con un flujo de transferencia de archivos. Se utiliza el módulo *ipw3945* en las computadoras que participan en el flujo de transferencia de archivos, de tal manera que su flujo será tratado como de mejor esfuerzo, al utilizar el Control de Acceso al Medio del estándar IEEE802.11 [2] al no contar con mecanismos WMM.

WMM con flujo en este escenario existe el soporte WMM en su totalidad y existe un flujo de transferencia de archivos. A diferencia del escenario WMM* con flujo, todas las computadoras utilizan el módulo *iwl3945* en su interfaz inalámbrica, que provee el soporte WMM.

El escenario "***lo***" es la **simulación** de las pruebas (el nombre está relacionado con el nombre de la interfaz de red *local loopback* en Linux). Esta simulación consiste en tener a un servidor Asterisk y un *softphone* de pruebas ejecutándose en la misma computadora. La interfaz de red virtual *loopback* es por ende la utilizada. En este escenario sólo existen los flujos VoIP y los parámetros recaudados serán las cotas de los datos obtenidos en los escenarios mencionados anteriormente.

En la figura 5.1 se muestran las computadoras que participan en las pruebas, donde existe flujo VoIP además de un flujo de transferencia de archivos (TA).

5.4 Configuración WMM

El PA cuenta con la posibilidad de modificar los parámetros WMM. Esto es posible al acceder a la página Web de configuración del PA. En la pestaña *Wireless->Advanced*, en la parte inferior, se encuentra la opción de habilitar o deshabilitar el soporte WMM. La opción *No-Acknowledgement* se dejó deshabilitada en las pruebas y así obtener un número menor de paquetes perdidos.

Como configuración adicional en el PA, se proveen las tablas de los parámetros WMM de conexión del Acceso Coordinado Distribuido Mejorado (EDCA) con sus valores por omisión. La tabla 5.1 contiene los parámetros EDCA, que gobierna el acceso al medio entre PA a la estación cliente. La tabla 5.2 contiene los parámetros EDCA,

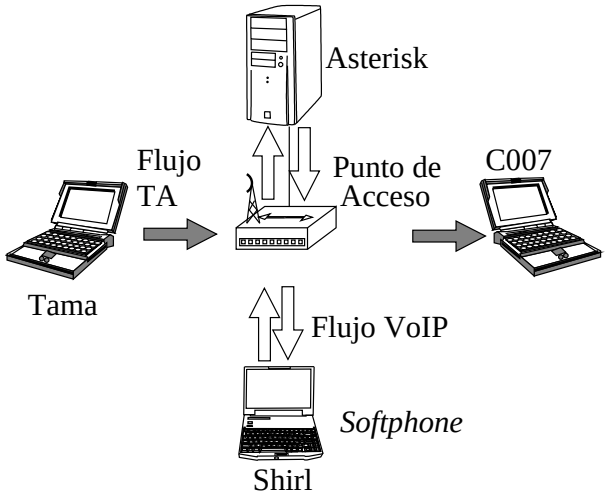

Figura 5.1: Infraestructura utilizada en las pruebas

que gobierna el acceso al medio entre un cliente al PA. Un cliente es un dispositivo conectado al PA, en este caso específicamente para una conexión inalámbrica.

Categoría de Acceso	CWmin	CWmax	AIFSN	TXOP(b)	TXOP(a/g)
Background	15	1023	7	0	0
Mejor esfuerzo	15	63	3	0	0
Video	7	15	1	6016	3008
Voz	3	7	1	3264	15004

Tabla 5.1: Parámetros EDCA del PA (PA a cliente) por omisión

En ambas tablas de configuración del PA se modificaron las Categorias de Acceso (CAs) de Voz y de Mejor esfuerzo, con el fin de dar una mejor prioridad a la CA de **Voz** y de priorizar al flujo de transferencia de archivos como CA de ***Background***.

Para lograr lo antes mencionado, a la CA de Voz se le brinda una oportunidad de transferencia del doble al valor por defecto (mediante el parámetro TXOP), tanto para la configuración del PA al cliente y viceversa. Para canalizar el flujo de transferencia de archivos en la Categoría de Acceso de *Background*, no se marcan los paquetes de este tipo con DSCPs como 001000_2 (CS1) o $C1 = 010000_2$ (CS2), cuestión que se encuentra fuera del ámbito de la tesis. Una manera directa es copiar los parámetros por omisión de la CA de *Background* en la CA de Mejor esfuerzo.

En la tabla 5.3 se denotan los parámetros utilizados en las pruebas para las CA de Voz y de Mejor esfuerzo.

Categoría de Acceso	CWmin	CWmax	AIFSN	TXOP(b)	TXOP(a/g)
Background	15	1023	7	0	0
Mejor esfuerzo	15	1023	3	0	0
Video	7	15	1	6016	3008
Voz	3	7	2	3264	15004

Tabla 5.2: Parámetros EDCA del cliente (cliente a PA) por omisión

Categoría de Acceso	CWmin	CWmax	AIFSN	TXOP(b)	TXOP(a/g)
Parámetros EDCA del PA					
Mejor esfuerzo	15	1023	7	0	0
Voz	3	7	1	3264	15004
Parámetros EDCA del cliente					
Mejor esfuerzo	15	1023	7	0	0
Voz	3	7	2	3264	15004

Tabla 5.3: Parámetros EDCA del PA y del cliente finales

5.5 Asterisk y el *softphone* de pruebas

Utilizamos el canal de comunicación SIP de Asterisk para las pruebas, para ello es necesario declarar la configuración del servidor Asterisk.

En el archivo `sip.conf` se configura el usuario, password, el codificador de audio que utiliza y otros parámetros del *softphone* de pruebas:

```
[pysua]           ; usuario
type=friend       ; realiza y recibe llamadas
secret=pysu       ; palabra secreta
nat=no            ; se encuentra en la misma red local
host=dynamic      ; puede tener una ip dinámica
context=internal  ; contexto al que pertenece
disallow=all      ; deshabilita todos los codificadores de audio
allow=ulaw        ; se habilita el codificador de audio ulaw
```

En este archivo también se puede habilitar el marcado del campo Tipo de Servicio (ToS) de paquetes salientes SIP y RTP de Asterisk:

```
tos_sip=cs1       ; Clase de servicio 1 tos=0x20
tos_audio=cs7     ; Clase de servicio 7 tos=0xE0
```

En el archivo `extensions.conf` se agrega en el contexto `internal` la siguiente extensión:

```
exten => noecho,1,Ringing()
exten => noecho,2,Answer( )
exten => noecho,3,Milliwatt( )
```

Esta extensión, con la primer función avisará al *softphone* que su llamada está en proceso y con la siguiente se confirma la llamadada (se *descuelga*). La tercer función `Milliwatt( )` genera un tono constante de $1,000$ Hz y es tomado como audio dirigido al *softphone*.

La configuración del usuario *pysua* anteriormente definida, es la que será utilizada por el *softphone* de pruebas. La información necesaria en el *softphone* es el nombre de usuario, el *password* y la dirección IP del servidor Asterisk.

El *softphone* de pruebas utiliza el módulo **py_pjsua** de la biblioteca **pjsua** para *Python*. Puede realizar múltiples llamadas VoIP simultáneas, registrarse con un *proxy* y es compatible con Asterisk. El *softphone* es el encargado de iniciar y terminar las llamadas.

Se obtuvieron dos compilados diferentes del módulo *py_pjsua*, pero que son del código fuente de la misma versión de la biblioteca (svn revisión 1931, que forma parte de la versión 0.8). Un compilado no realiza marcado de los paquetes SIP y RTP, el otro compilado realiza el marcado apropiado para SIP y RTP. La elección del módulo es intrínseca al modelo de prueba.

El *softphone* es capaz de utilizar el codificador de audio PCMU entre de otros. El usuario *pysua* en el servidor Asterisk tiene configurado el codificador PCMU como único codificador de audio, por lo cual en la negociación de la llamada se establecerá el codificador PCMU como codificador de llamada.

En el *softphone* se deshabilita la opción de cancelación de eco, la opción de Detección de Actividad de Voz (VAD) y la calidad de audio tiene un parámetro de 10 (la mejor). La cancelación de eco consiste en suprimir el eco en la llamada. VAD tiene como función el detectar cuando no existe audio de voz presente y por lo tanto, el audio sin voz no es transmitido. La calidad de media con número entre 5 y 10, utiliza un filtro amplio en el cambio de la frecuencia de muestreo, significando una mejor calidad de audio.

El *softphone* no hace uso de la tarjeta de sonido de la computadora donde se ejecuta. El principal argumento es que el propio hardware puede introducir un pequeño retraso en la lectura del audio y al realizar múltiples llamadas quizá se introduzcan retrasos que no sean intrínsecos de la red. En el *softphone* se registra un tono de audio de 440 Hz, este es el audio que será transportado del *softphone* al servidor Asterisk. En un principio se creaba un tono por cada llamada, utilizándose mucho el recurso procesador. Se optó por crear sólo un tono que es agregado como audio en cada llamada.

Cada llamada realizada por el *softphone* tiene una duración de 40 segundos y existe un pequeño **lapso de tiempo** que separa el inicio de las llamadas. Este lapso de tiempo también es utilizado para la terminación entre llamadas. La primer llamada

que inicia es la primer llamada que termina, las siguientes llamadas siguen el mismo patrón a manera de que todas ellas tengan el mismo tiempo de duración.

El lapso de tiempo calculado entre llamada para cada prueba se obtiene de la siguiente ecuación:

$$lapso\_de\_tiempo = 500 + 20/n$$

Donde *lapso_de_tiempo* está dado en milisegundos y n es el número de llamadas simultáneas para la prueba. El lapso de tiempo por omisión, es de al menos 500 milisegundos (ms) más una pequeña porción de tiempo calculada como $20/n$. La pequeña porción de tiempo es la que en realidad nos permite equiespaciar las llamadas, el valor 20 se encuentra directamente relacionado con la carga de audio del codificador PCMU o μlaw, que lleva cada paquete RTP.

La razón de iniciar de este modo las llamadas, es que nos permite no saturar tanto el *softphone* como el servidor Asterisk, dado que ambos realizan una transacción SIP para efectuar la llamada, y sobre todo homogeneiza la utilización del ancho de banda de la red. Todas las llamadas se encuentran equiespaciadas debido al lapso de tiempo entre ellas, obteniendo un flujo total que teóricamente no se recarga hacia un intervalo de tiempo específico.

En la figura 5.2 se muestra la razón de la importancia de equiespaciar el inicio de las llamadas: la figura representa el primer paquete RTP enviado al inicio de cada llamada, el primer paquete enviado por llamada es representado por una flecha. Las curvas representan la tasa de bit[1] cuando hay un equiespaciado y cuando no lo hay. Por cada intervalo de 4 milisegundos se obiene la tasa de bit (número de bits que llegaron en 4 segundos y cada paquete tiene 1600 bits) y el valor obtenido es graficado con splines cúbicos en la mitad del intervalo. Por ejemplo, el primer valor puntual en la gráfica no equiespaciada es la coordenada $(2, 1200)$, para el primer intervalo $[0, 4)$ de 4 segundos.

El codificador utilizado en las pruebas es el μlaw o PCMU, principalmente porque no es un codificador de tasa de bit variable y la cantidad de carga de datos es constante (160 muestras) transportando 20 ms de audio, véase la sección 2.3. El codificador PCMU es de los codificadores que tienen una tasa de bit mayor de los soportados en Asterisk. Asterisk cuenta con codificadores que son llamados de bajo ancho de banda, ya que los conocidos como de alto ancho de banda pueden pasar el megabit de tasa de bit.

5.6 Notas importantes del planteamiento de las pruebas

Las pruebas de escenarios que no incluyen el flujo de transferencia de archivos, tienen el objetivo de observar el comportamiento al disponer totalmente del ancho

[1]En inglés el término tasa de bit es *bitrate*.

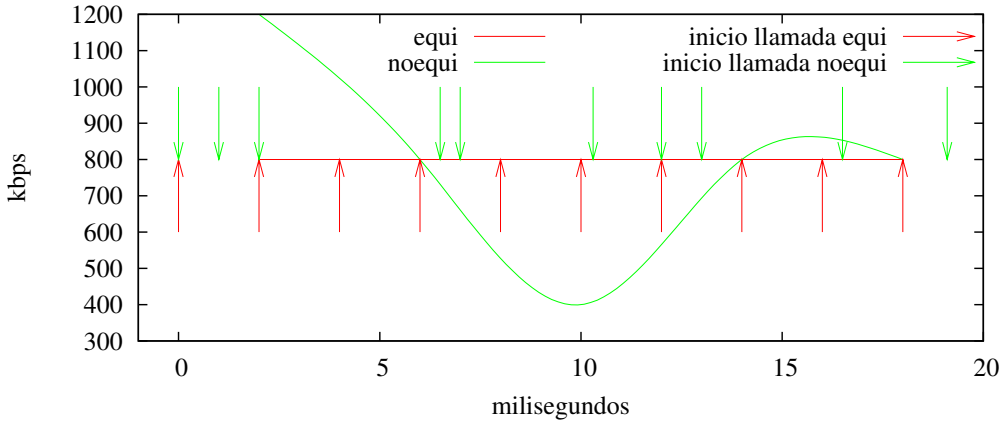

Figura 5.2: Equiespaciado en las llamadas

de banda. En los escenarios donde existe una competencia por el acceso al medio inalámbrico, se realizaron pruebas con flujos VoIP además de un flujo adicional de transferencia de datos. Esto nos permite recabar información comparable no sólo entre diferentes escenarios con igual número de llamadas, si no también cuando no se dispone de todo el ancho de banda para las llamadas VoIP y existe otro flujo bajo un mismo escenario.

La transferencia de archivos se realiza de la computadora `Tama` hacia la computadora `C007` mediante el programa *scp*. Se transfiere un archivo de tamaño de 1 gigabyte y por cuestiones de espacio, se guarda el archivo recibido en */dev/null*. En general, no se consigue la transmisión total de la información, debido a que cada prueba tiene una duración de menos de un minuto.

Se realizaron pruebas para cada escenario con 5, 10, 15 y 20 llamadas simultáneas.

Las computadoras participantes en las pruebas, guardan el flujo entrante y saliente de su interfaz de red (*Ethernet* o inalámbrica según su conexión) mediante el programa `tcpdump`. A continuación se presenta el comando ejecutado en la computadora `Shirl`, que captura los paquetes entrantes y salientes en la interfaz inalámbrica `eth1`, almacenando los paquetes en el archivo `cap.Shirl.01`:

```
$tcpdump -i eth1 -w cap.Shirl.01
```

Una de las limitaciones del comando es que no necesariamente guardará todo el contenido del paquete capturado. Para el caso de las pruebas, se captura la información necesaria para realizar la medición de parámetros como tiempo de llegada del mensaje, tamaño total del paquete etc. Hay información que no se llega a capturar, como la carga de audio del paquete RTP, esto debido a que podría darse el caso de perder paquetes entrantes o salientes al estar almacenando otros paquetes.

El valor utilizado cuando existe marcado en el paquete RTP es $ToS = 0xE0$, siendo entonces el $DSCP = 111000_2$ que mapea directamente a la Clase de Servicio

7 (CS7), WMM mapea dicho valor a la Categoría de Acceso de Voz. Cuando existe marcado RTP también existe marcado SIP, los paquetes SIP fueron marcados con un $ToS = 20$ o CS1. De esta manera el flujo RTP tendrá la mejor prioridad sobre el flujo de transferencia de archivos y cualquier otro flujo, incluso del flujo SIP.

El motivo por el cual el flujo SIP es enviado a una Clase de Servicio (CS) de *background* es porque se estudia el comportamiento del flujo RTP y no el comportamiento del flujo SIP en condiciones de carrera. Los parámetros VoIP no se ven afectados (se incrementen o decrementen) al configurar la señalización como de baja categoría. Una vez iniciada la sesión VoIP no se requieren acciones de re-INVITE para ejecutar una renegociación de media, puesto que el codificador PCMU no es de tasa de bit variable y en la configuración se limita a sólo el uso de éste. Un posible inconveniente es el retraso en los paquetes SIP.

El *softphone* siempre es ejecutado como súper usuario.

Con todo este esquema, las pruebas VoIP proporcionarán información muy independiente de hardware o software que realicen mecanismos que no son objetivo de estudio en esta tesis y que además pueden significar algún tipo de deterioro en la comunicación.

5.7 Datos obtenidos de las pruebas

Esta sección se trata sobre la extracción de los parametros que se analizarán en subsecuentes secciones. Se obtuvieron los paquetes entrantes y salientes de la interfaz de red de las computadoras, presentes en las pruebas mediante el comando `tcpdump`. Esta información obtenida se encuentra en bruto y en cantidad considerable.

Se realizaron múltiples *scripts* en **Python** y en **Bash**, cuya finalidad consistió en obtener información resumida de parámetros y en obtener datos parcialmente promediados para ser observados gráficamente. El programa *WireShark* puede realizar el cálculo de parámetros VoIP y gráficas de *jitter* y *diferencia*, pero debido al número de llamadas y al número de pruebas, el simple hecho de utilizar la interfaz de usuario para tal actividad resultaría extenuante.

El análisis se realizará en paquetes que fueron recibidos por el destinatario. Se puntualiza que los paquetes que salen de una interfaz de red alámbrica o inalámbrica no serán tomados en cuenta en el análisis, ya que dicho paquete puede perderse en su trayecto o retrasarse, el tiempo de salida del paquete no proporciona esta información.

5.7.1 Parámetros VoIP

Las computadoras que generaron información VoIP son `Asterisk` y `Shirl`. La identificación de cada llamada se hace única al contar con la tupla computadora/puerto.

El softphone de pruebas siempre inicia el flujo RTP de la primer llamada en el puerto 4000, la siguiente por el puerto 4002 y así sucesivamente en número par hasta

completar el número de llamadas requerido. El puerto RTCP para la primer llamada es el puerto 4001, pero este flujo no será analizado ni el encabezado de estos paquetes. El encabezado del paquete RTP es la fuente de información para obtener los parámetros VoIP. Para cada llamada se tiene un puerto individual de RTP y otro para RTCP, ésto se debe a que son llamadas individuales y no forman una conferencia.

El primer filtro de los archivos de la captura de los paquetes, es hecho al ejecutar el comando:

```
tcpdump  $politic -r cap.${host}.${testn} -v -T rtp -tt  > \
 ${testn}/dataD/${filebase}.dump
```

En la expresión anterior, la variable `politic` contiene una política de filtro. Para los paquetes RTP, que tienen el puerto 4000 como destino y fuente, es la política `udp dst port` 4000 `|| udp src port` 4000. Se pueden incluir otros puertos (como el 4002) al separarlos con `||`.

El argumento `-r` indica cuál es el archivo a leer, en este caso nuestra captura previa. El argumento `-v` nos da información extra como el tiempo de vida, identificación y el tamaño total del paquete. El argumento `-T` seguido de `rtp` fuerza a los paquetes seleccionados por la política, ser interpretados como paquetes RTP, dando la información del encabezado del paquete RTP, véase sección 2.2.3. El último argumento `-tt` imprime al inicio de cada paquete la marca de tiempo de llegada registrado, sin formato en segundos. Finalmente, la información de cada paquete RTP es impreso por línea y es almacenado en un archivo con extensión `.dump`. La información se encuentra en ASCII.

Las variables como `politic` son inicializadas por un *script* Bash, de acuerdo al número de llamadas, a la computadora y al escenario que trate la prueba.

Un ejemplo de una línea de éste tipo de información obtenida es la siguiente:

```
1214970811.138497 IP (tos 0xe0, ttl 64, id 0, offset 0, flags [DF],
 proto UDP (17), length 200) Asterisk.14312 > 192.168.201.102.4000:
 udp/rtp 160 c0  22053 640 591298146
```

Con la marca de tiempo y el tamaño del paquete, mediante un *script* en Python, se obtiene la tasa de transmisión de todas las llamadas de cada prueba, aunque se podría obtener por cada llamada. La tasa de transmisión se obtiene cada segundo transcurrido, obteniendo una curva discreta que puede ser graficada de manera continua al aplicar algún tipo de *spline*.

Otro *script* en Python, filtra cada flujo RTP que se identifica por el número de puerto. Por cada flujo imprime en un archivo el tiempo de llegada relativo al primer paquete, el tiempo de llegada entre el anterior y el actual paquete (delta), la diferencia y el *jitter*, para estos dos últimos parámetros véase la definición de ***jitter* de entrellegada** en la página 22. La *diferencia* y el *jitter* es calculado al aplicar las siguientes ecuaciones:

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \tag{5.1}$$

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16 \tag{5.2}$$

Existe un cuarto dato que sólo es impreso cuando en el tiempo relativo al primer paquete ha transcurrido un segundo, el dato es la tasa de bit del flujo RTP. Se tiene una de estas entradas por cada paquete RTP y en general por la cantidad de información todavía presente, estos datos son candidatos a ser graficados. A continuación podemos ver un ejemplo de un archivo de este tipo (las 10 primeras líneas):

```
#  2002 2002 0 (esperados recibidos perdidos)
# tiempo-rel delta(ms) D(ms) jitter(ms) tasa
0.0 0.0 0.0 0.0
0.0192041397095 19.2041397095 0.795860290527 0.049741268158
0.0391609668732 19.9568271637 0.0431728363037 0.0493307411671
0.0596520900726 20.4911231995 0.491123199463 0.0769427698106
0.0794529914856 19.800901413 0.199098587036 0.0845775083872
0.100085020065 20.6320285797 0.632028579712 0.118793200345
0.119307994843 19.2229747772 0.777025222778 0.159932701747
0.139415979385 20.1079845428 0.107984542847 0.156685941816
```

la primer línea nos informa de cuantos paquetes son esperados, cuantos recibidos y cuantos perdidos. Véase la definición **número acumulado de paquetes perdidos** de la página 22.

Los datos hasta ahora son difíciles de comparar debido a la gran cantidad de ellos. Sobre los datos anteriormente mencionados, por cada flujo de voz, se obtienen los siguientes datos a manera de resumir todos los datos del flujo:

paquetes recibidos - es el número de paquetes recibidos en la interfaz de red.

paquetes perdidos - los paquetes perdidos se calculan al realizar la resta de paquetes esperados menos paquetes recibidos.

delta máximo - es el valor máximo del parámetro delta del flujo.

***jitter* máximo** - es el valor máximo que obtiene el *jitter*.

promedio del *jitter* - es el promedio del *jitter* de entre llegada, el *jitter* de entre llegada es calculado cada vez que se tiene un nuevo paquete RTP.

duración - es la diferencia de las marcas de tiempo del último y primer paquete RTP recibido en el flujo.

Cada una de las pruebas realizadas tiene llamadas VoIP simultáneas, como última forma de reducción de información, se promedian los datos obtenidos de las llamadas simultáneas.

Tómese en cuenta que una llamada simple VoIP contiene 2 flujos de voz. Dados 2 fuentes se tienen 2 flujos de voz.

5.7.2 Parámetros del flujo de transferencia de archivos

El análisis exhaustivo, sobre este tipo de flujo, requiere de un estudio completo, actividad que queda fuera del ámbito de la tesis. El protocolo de red que utiliza el programa `scp` para la transmisión de archivos es TCP.

TCP provee un transporte de datos confiable de extremo a extremo sobre una red no fiable. TCP puede adaptarse dinámicamente a las propiedades de la red y manejar fallas de muchas clases.

Aún cuando se tiene un medio fiable de transmisión de los datos, al utilizar un protocolo como TCP, es también conveniente la verificación de los datos transmitidos. Es por eso que se encuentran 2 flujos en la transferencia de archivos. El primer flujo se encarga de transportar la información del archivo y el segundo flujo es el que notifica si han llegado de manera íntegra n número de paquetes (reconocimientos). El flujo que transporta los datos del archivo utiliza una tasa de transferencia de bits mucho mayor que el segundo flujo.

El parámetro que se analizará es sólo la tasa de bit que tiene el flujo que transporta los datos del archivo que se envía.

Capítulo 6

Análisis de datos

En este capítulo se mostrará el flujo VoIP que recibió tanto la computadora `Shirl` como la computadora `Asterisk` bajo los escenarios planteados en el capítulo anterior. Además se analizan los resultados obtenidos de los parámetros que caracterizan a un flujo VoIP.

El primer parámetro a analizar es la tasa de bit total que generan los flujos VoIP. Las gráficas resultantes, tienen un inicio de ascenso (inicio de llamadas) un periodo de tiempo de llamadas concurrentes (con una tasa de bit constante) y finalmente un descenso (término de llamadas), correspondiendo al planteamiento de las pruebas. Cada llamada tiene una duración de 40 segundos, el ascenso inicia en el segundo 0 y el descenso inicia en el segundo 40. El tiempo que transcurre en el ascenso y en el descenso es calculado dependiendo al número de llamadas simultáneas, debido al equiespaciado que se plantea en la sección 5.5. Este comportamiento es el mismo en cada uno de los escenarios y por lo tanto comparables. La diferencia más notoria es la utilización de distintos Controles de Acceso al Medio (CAM) para el medio inalámbrico.

Se exhibe el flujo total de las llamadas VoIP de los tres escenarios con flujo (ver sección 5.3), además de los flujos de transferencia de archivos, a manera de observar el comportamiento del acceso al medio inalámbrico. También se presenta un análisis de los parámetros VoIP, como *jitter* y diferencia de llamada número 10. Finalmente se muestran en una tabla los valores promedio o máximos de varios parámetros VoIP.

Mediante el planteamiento de las pruebas y de los datos extraidos, se muestran resultados y se hacen análisis experimentales cuantitativos. Estos son realizados al obtener mediciones intrínsecas al medio por el que viaja el flujo de voz. En un estudio cualitativo se obtiene una medición de calidad percibida que puede ser evaluada al aplicar Puntaje de Opinión Media (MOS).

6.1 Tasa de bit del flujo VoIP en el escenario WMM y no-WMM

En la figura 6.1 se muestra la tasa de bit del flujo VoIP total entrante en la computadora `Shirl` para 5, 10, 15 y 20 llamadas simultaneas, en los escenarios no-WMM y WMM. Nótese que en estas pruebas no existe otro flujo. Los resultados de las pruebas del escenario *lo* (ver sección 5.3) sirven de referencia y cota para las otras pruebas. **Los escenarios WMM y no-WMM muestran que el acceso al medio cumple con el ancho de banda requerido por el flujo VoIP, para el número de llamadas antes mencionadas**.

Para 5 llamadas simultáneas, el flujo entrante en la computadora `Shirl` es de 5 flujos de voz provenientes de la computadora `Asterisk`, teniendo un flujo entrante teórico de 80 kbps $*$ 5 = 400 kbps. Del mismo modo, para 10 flujos de voz se requiere una tasa de bit de 800 kbps, para 15 flujos es de 1200 kbps y para 20 flujos es de 1600 kbps.

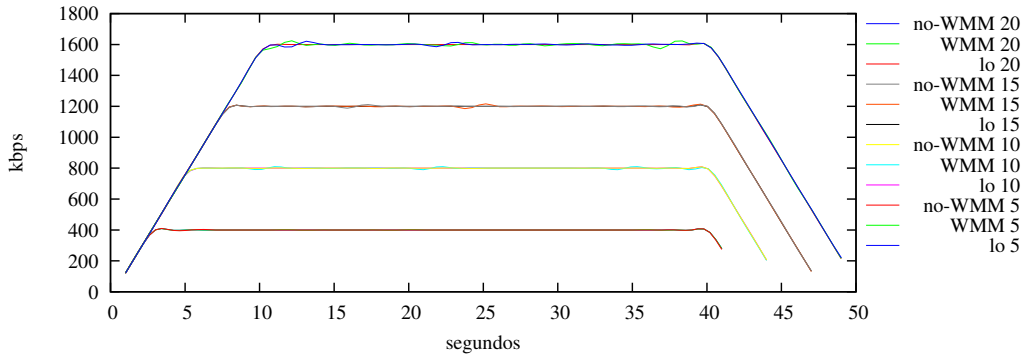

Figura 6.1: Flujo total VoIP entrante en `Shirl` de 5, 10, 15 y 20 llamadas simultaneas en los escenarios *lo*, WMM y no-WMM

Los flujos de voz entrantes en la computadora `Asterisk` provenientes de `Shirl` son representados por curvas muy similares a las de la figura 6.1.

6.2 Tasa de bit del flujo VoIP en el escenario no-WMM con flujo

En la figura 6.2 se muestra el flujo VoIP entrante en `Shirl` bajo el escenario no-WMM con flujo. Aquí se muestra que el acceso al medio para el flujo VoIP se ve afectado para 10, 15 y 20 llamadas simultáneas. La tasa de bit para dicho número de llamadas permanece alrededor de 600 kbps, significando que el flujo de transferencia

de archivos afectó en gran medida al flujo VoIP. Para 5 llamadas VoIP se aprecia que si se cumple con el ancho de banda requerido. El flujo de 15 y 20 llamadas muestran un pico en sus gráficas que sobrepasan el ancho de banda requerido, esto podría deberse a la presencia de un rezago de múltiples paquetes RTP para ser envíados y que tuvieron una pequeña oportunidad de enviarse.

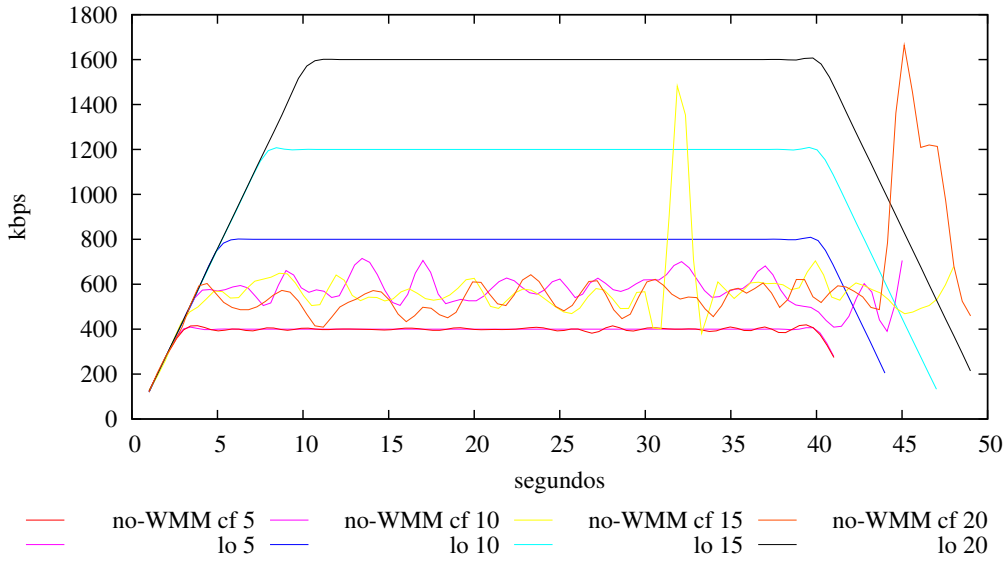

Figura 6.2: Flujo VoIP entrante en `Shirl` de 5, 10, 15 y 20 llamadas en el escenario no-WMM con flujo

En la figura 6.3 se contempla el flujo VoIP entrante en la computadora `Asterisk` bajo el escenario no-WMM con flujo. Para un número de 5 y 10 llamadas simultáneas se cumple con el ancho de banda necesario. En la prueba de 10 llamadas no se nota un descenso en la gráfica, parecido a la prueba *lo* con el mismo número de llamadas. Para un número de 15 llamadas parece cumplirse con el ancho de banda requerido, aunque con un acceso al medio que no permite una curva parecida a la prueba *lo*. Además, se puede apreciar que este flujo en su fase de descenso permanece por más de 5 segundos, con una tasa de bit que pareciera encontrarse en la fase de llamadas concurrentes.

En la prueba de 20 llamadas simultáneas, no se cumple con el ancho de banda requerido, estancado en una tasa de bit alrededor de 1200 kbps, con una gráfica similar a la prueba con 15 llamadas. Se aprecia claramente que el flujo en la fase de descenso tiene un súbito incremento de la tasa de bit, con un pico que sobrepasa el ancho de banda necesario, a diferencia de la prueba *lo* de 20 llamadas.

Los flujos VoIP en la figura 6.2 quedan estancados en un ancho de banda alrededor de los 600 kbps y en la figura 6.3 alrededor de los 1200 kbps. Esta diferencia notable

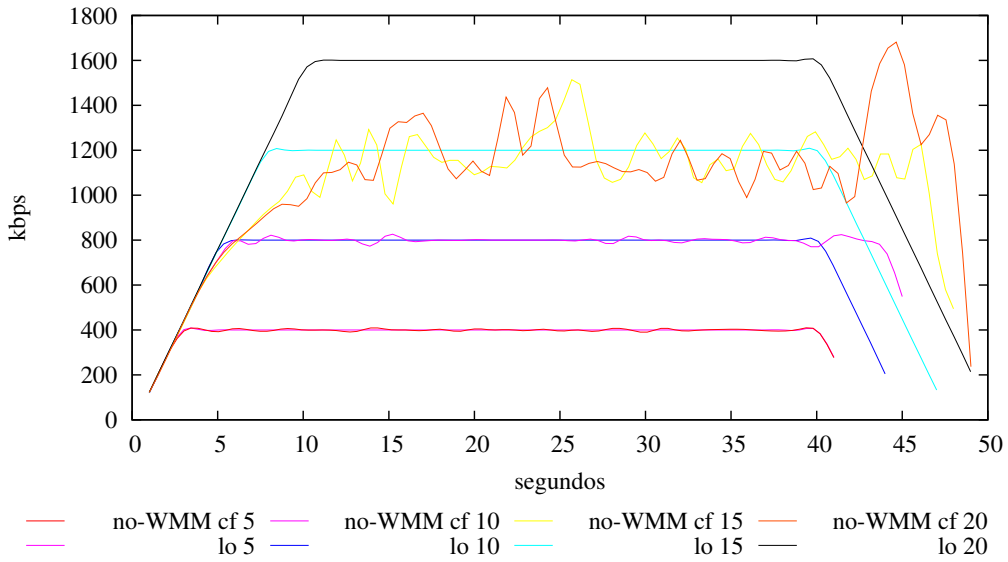

Figura 6.3: Flujo VoIP entrante en `Asterisk` de 5, 10, 15 y 20 llamadas en el escenario no-WMM con flujo

de anchos de banda bajo el mismo escenario y número de llamadas, es resultado de la competencia por el acceso al medio.

En la figura 6.2 se muestra el flujo que generó la computadora `Asterisk` hacia la computadora `Shirl`. La computadora `Asterisk` toma una parte del acceso al medio que tiene el PA para transmitir.

En la figura 6.3, se muestran los flujos VoIP recibidos en la computadora `Asterisk`, que son enviados por la computadora `Shirl`. El acceso al medio de `Shirl` es dedicado al flujo VoIP y su oportunidad de envío de paquetes VoIP es mejor que la del PA, ya que éste último también tiene que enviar los datos de la transferencia de archivos hacia la computadora `C007` y el flujo VoIP que envía la computadora `Asterisk`.

6.3 Tasa de bit del flujo VoIP en el escenario WMM* con flujo

En el escenario WMM* con flujo, las computadoras que envían flujo VoIP cuentan con el soporte WMM, mientras que las computadoras involucradas con el flujo de transferencia de archivos no cuentan con el soporte WMM.

La tasa de bit de los flujos VoIP entrantes en la computadora `Shirl`, se muestran en la figura 6.4 y los flujos VoIP entrantes en la computadora `Asterisk`, en la figura 6.5, para el escenario WMM* con flujo. Las figuras muestran que se cumple con el ancho de banda requerido por los flujos VoIP de 5, 10, 15 y 20 llamadas simultáneas,

aún con la presencia de un flujo de transferencia de archivos. Nuevamente el flujo entrante en la computadora `Asterisk` se apega más a la prueba *lo* en comparación al flujo entrante en la computadora `Shirl`.

En la gráfica de 20 llamadas de la figura 6.4, se observa una oscilación notable antes de iniciar el descenso, con un mínimo de 1300 kbps y un máximo de 1820 kbps, −300 y +220 kbps del ancho de banda requerido. En la figura 6.5 se presenta esta oscilación en el mismo lugar, con un valle de −133 kbps y una cresta de +132 kbps en referencia a 1600 kbps. Las oscilaciones de la figura 6.4 y de la figura 6.5, atestiguan que el acceso al medio de la computadora `Shirl` es mejor, sobre el acceso al medio del PA para el flujo VoIP.

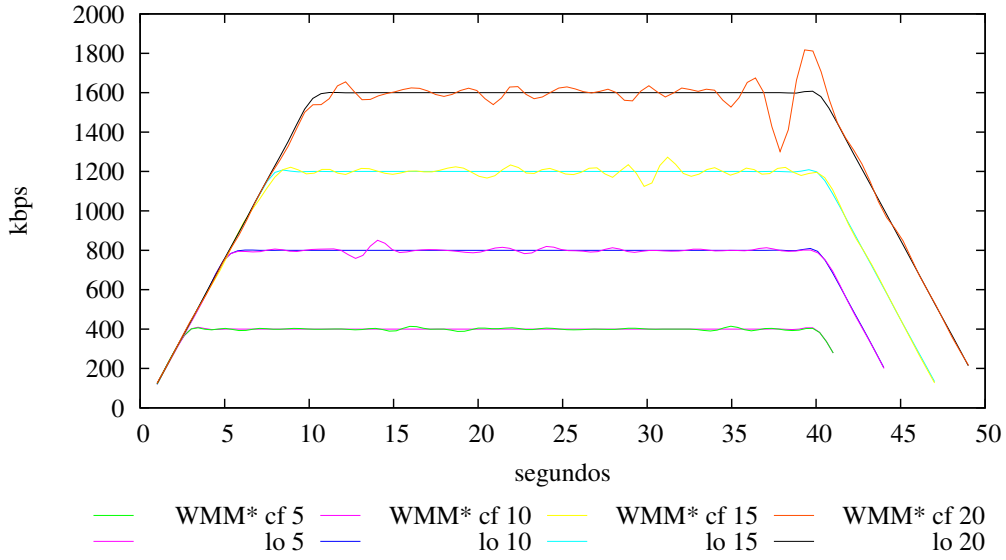

Figura 6.4: Flujo VoIP entrante en `Shirl` de 5, 10, 15 y 20 llamadas en el escenario WMM* con flujo

6.4 Tasa de bit del flujo VoIP en el escenario WMM con flujo

En el escenario WMM con flujo, tanto las computadoras que generan el flujo VoIP como las computadoras que generan el flujo de transferencia de archivos cuentan con el soporte WMM.

El escenario WMM con flujo para 5, 10, 15 y 20 llamadas simultáneas presenta un flujo VoIP de entrada en la computadora `Shirl` que se muestra en la figura 6.6. La figura 6.7 presenta el flujo VoIP entrante en la computadora `Asterisk` en el mismo escenario.

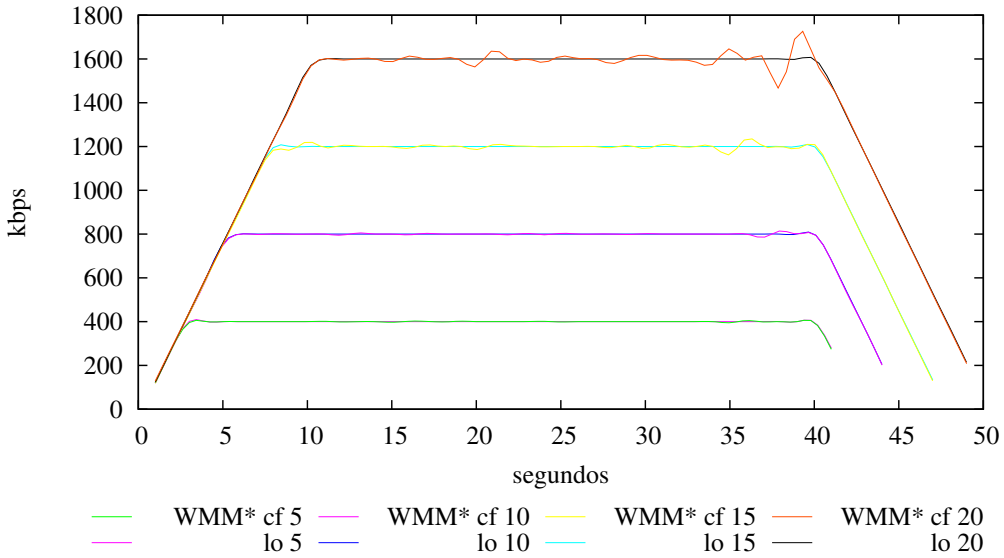

Figura 6.5: Flujo VoIP entrante en `Asterisk` de 5, 10, 15 y 20 llamadas en el escenario WMM* con flujo

El flujo VoIP entrante en la computadora `Asterisk` parece tener una forma más cercana al flujo VoIP de la prueba *lo*, en comparación al flujo entrante en la computadora `Shirl`. Al inicio de las gráficas de la figura 6.6, les toma más de 5 segúndos alcanzar la tasa de bit teórica para cada número de llamadas (excepto para la prueba de 5 llamadas), notandose un pico en las gráficas de 10 y 15 llamadas. Estos picos reflejan que ha ocurrido un tipo de modificación en el acceso al medio, brindando al flujo VoIP un mayor tiempo de transferencia. En esta misma figura, el flujo VoIP tiene una tasa de bit que varía muy poco en el resto de la prueba, pero que no es mejor en comparación a las tasas de bit resultantes en las pruebas del escenario WMM* con flujo.

En la figura 6.7 se observa que las rectas en la fase de inicio de llamadas, tiene una pendiente menor a la referencia dada por la prueba *lo*. Dicho fenómeno también ocurre en la figura 6.6 y es más notable en la gráfica de 20 llamadas. Este comportamiento no se había observado en los escenarios anteriores. Podría pensarse que mejorar el acceso al medio de un flujo de mayor prioridad, no resulta instantáneo al encontrarse en competencia con otros flujos, pero por otro lado, el flujo SIP se encuentra en una Categoría de Acceso (CA) de *Background* siendo ésta la más probable razón de dicho fenómeno. Al encontrarse los paquetes SIP en una CA de *Background*, es muy probable que se presente un retraso grande en la entrega de los paquetes SIP. Obsérvese la fase de inicio de llamadas en las figuras 6.5 y 6.7.

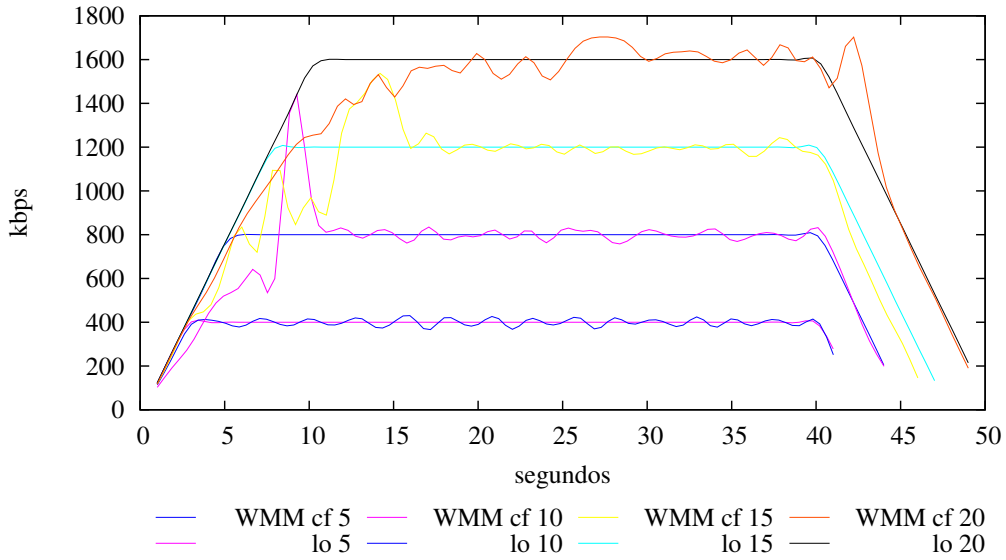

Figura 6.6: Flujo VoIP entrante en `Shirl` de 5, 10, 15 y 20 llamadas en el escenario WMM con flujo

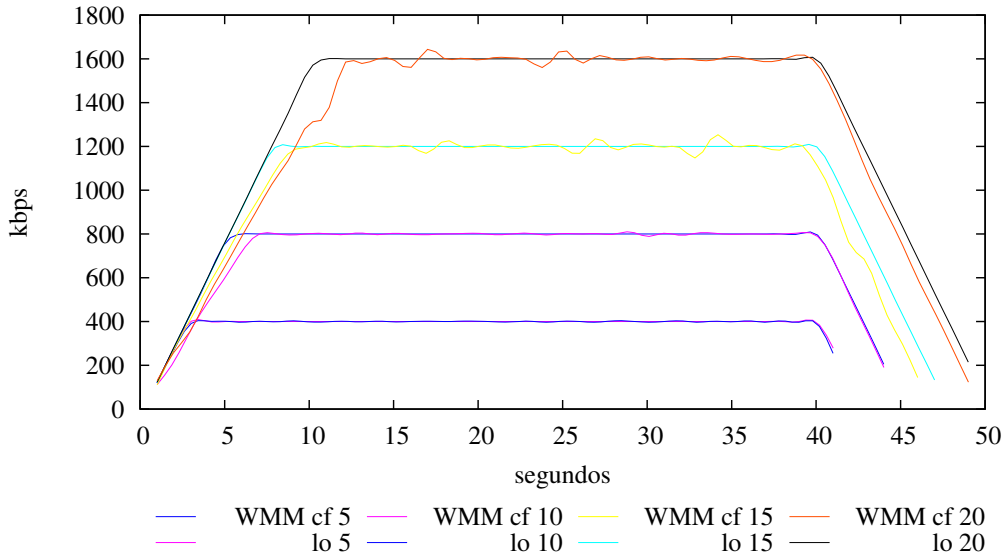

Figura 6.7: Flujo VoIP entrante en `Asterisk` de 5, 10, 15 y 20 llamadas en el escenario WMM con flujo

6.5 Análisis de los escenarios con flujo en la prueba con 20 llamadas

Los resultados anteriores muestran que bajo los escenarios planteados, la prueba que resultó menos favorable en todas ellas fue la prueba con 20 llamadas VoIP. Esto era esperado, dado que se consume un mayor ancho de banda. Con 20 llamadas VoIP simultáneas se utiliza 3200 kbps de ancho de banda (tómese en cuenta que son 40 flujos de voz).

En la figura 6.8-a se muestran las gráficas del flujo de transferencia de archivos recibido en la computadora `C007`, en la figura 6.8-b se muestran las gráficas del flujo VoIP entrante en la computadora `Shirl` y en la figura 6.8-c el flujo VoIP entrante en la computadora `Asterisk` . Estos son los flujos resultantes en los escenarios no-WMM con flujo, WMM* con flujo y WMM con flujo con 20 llamadas VoIP simultáneas respectivamente.

Por ejemplo, la gráfica flujo 1 de la figura 6.8-a, es la tasa de bit del flujo de transferencia de archivos, la gráfica WMM cf 1 en la figura 6.8-b es la tasa de bit de 20 flujos VoIP entrantes en `Shirl` y finalmente la gráfica WMM cf 1 en la figura 6.8-c es la tasa de bit de 20 flujos entrantes en `Asterisk`, todos estos flujos bajo el escenario WMM con flujo en la prueba de 20 llamadas simultáneas. De manera similar, las gráficas flujo 2 en la figura 6.8-a, WMM* cf 2 en la figura 6.8-b y WMM* cf 2 en la figura 6.8-c pertenecen a la prueba de 20 llamadas simultáneas bajo el escenario WMM* con flujo. Por último, las gráficas flujo 3 en la figura 6.8-a, no-WMM cf 3 en la figura 6.8-b y no-WMM cf 2 en la figura 6.8-c pertenecen a la prueba de 20 llamadas simultáneas bajo el escenario no-WMM con flujo.

La figura 6.8-b exhibe que WMM* cf 2 es una curva más cercana a la prueba *lo*, seguida de la curva WMM cf 1 y muy por de bajo la curva no-WMM cf 3. En la figura 6.8-c hay un comportamiento similar pero se mejora la tasa de bit en cada una de las curvas. La curva no-WMM cf 3 en la figura 6.8-c es mejor que la curva no-WMM cf 3 en la figura 6.8-b, siendo la diferencia más notoria entre la figura 6.8-b y la figura 6.8-c.

Los flujos de transferencia de archivos junto con los flujos VoIP compitieron por el acceso al medio. La idea principal de la figura 6.8 es indicar que el flujo de transferencia de archivos, compitió con el flujo VoIP bajo los tres escenarios con flujo (el flujo 1 compitió con el flujo VoIP bajo el escenario WMM con flujo).

El flujo 3 inicia con una tasa de bit de poco menos de 10 Mbps y cuando están presentes las 20 llamadas VoIP, permanece alrededor de los 8 Mbps. Al término de las llamadas VoIP, el flujo 3 recupera el ancho de banda que tenía al inicio. Observando el flujo VoIP del escenario no-WMM con flujo en la figura 6.8-b y en la figura 6.8-c, se aprecia que tienen un ancho de banda de aproximadamente 600 kbps y 1200 kbps respectivamente, utilizando un ancho de banda total de 1.8 Mbps. El flujo 3 se decrementó 2 Mbps para dar paso al flujo VoIP, este decremento no es suficiente para

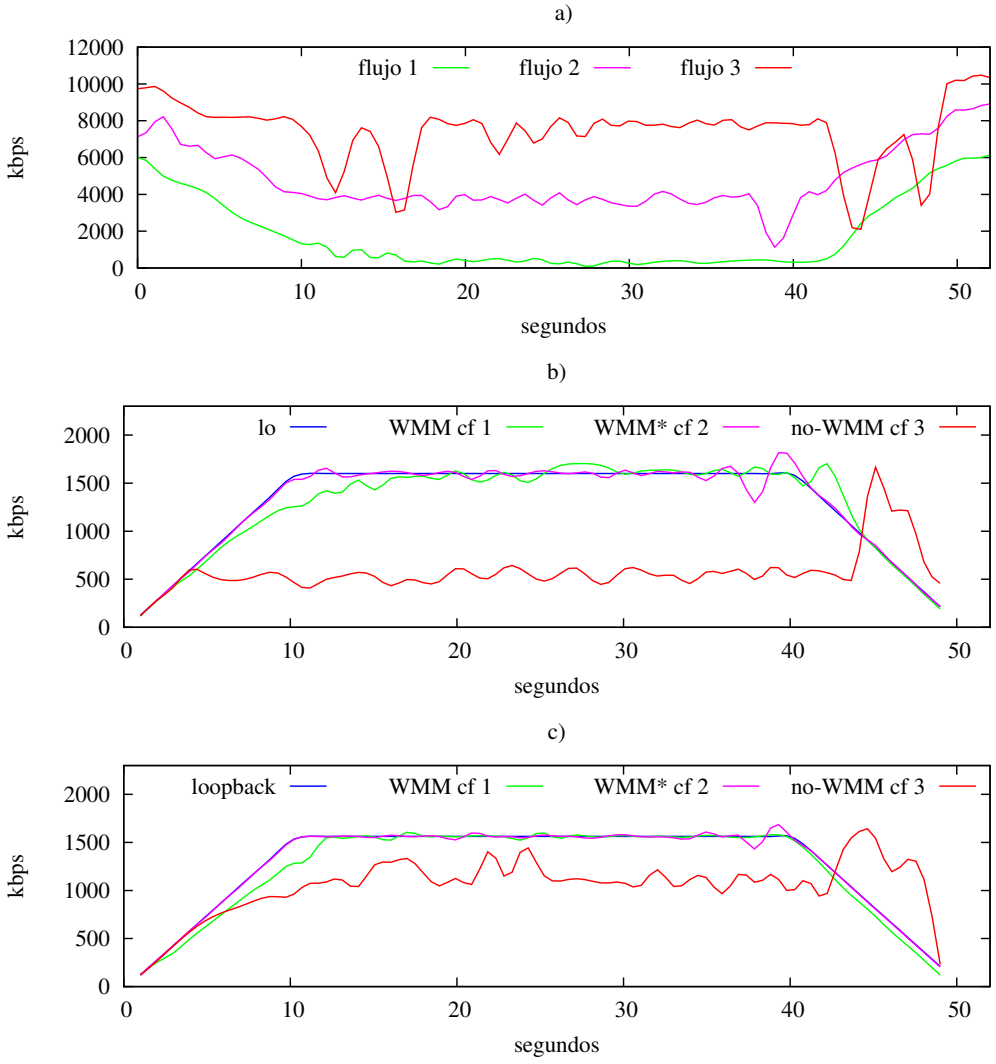

Figura 6.8: En a) se muestra la tasa de bit del flujo de transferencia de archivos recibido en `C007` que resultaron en las pruebas de 20 flujos VoIP simultáneos, bajo los tres escenarios con flujo. En b) se muestra la tasa de bit de los 20 flujos VoIP entrantes en `Shirl` para los tres escenarios con flujo (cf); En c) al igual que en a) pero del flujo entrante en `Asterisk`

cumplir con los 3200 kbps teóricos que necesitan los flujos VoIP.

El flujo 2 inicia con una tasa de bit poco menos de 8 Mbps y decae a una tasa de bit por encima de los 4 Mbps. La diferencia es de casi 4 Mbps y en este caso el escenario WMM* con flujo cumple con la tasa de bit teórico requerido para el flujo VoIP. El flujo 1 inicia con una tasa de bit de 6 Mbps y decae por encima de los 200 kbps, dejando prácticamente el ancho de banda disponible para los flujos VoIP. El flujo VoIP en el escenario WMM tiene una forma parecida al de la prueba *lo*, pero que resulta con más variaciones que la prueba WMM* con flujo.

Los resultados que podemos observar, van de acuerdo al planteamiento de las pruebas. El escenario no-WMM no cuenta con ningún mecanismo de QoS al utilizar el acceso al medio del estándar IEEE 802.11 [2] y se muestra que el flujo VoIP resulta seriamente afectado.

El escenario WMM* cuenta con QoS WMM [4] en las computadoras que producen el flujo VoIP, mientras que las computadoras del flujo de transferencia de datos utilizan el Control de Acceso al Medio (CAM) del estándar 802.11. Teóricamente los flujos que utilicen el CAM 802.11 en redes que cuenten con WMM, serán tratados con una prioridad de CA de Mejor esfuerzo. Se puede apreciar que el flujo VoIP no es perturbado por el flujo 2.

El escenario WMM cuenta con QoS WMM tanto en las computadoras de flujo VoIP, como en las de flujo de transferencia de archivos. El flujo 1 es enviado a la CA de *background* y es por tanto que su degradación se encuentra muy afectada, dado que existe el flujo VoIP que tiene la mejor prioridad (CA de Voz). En estas pruebas, el flujo VoIP se ve ligeramente afectado por el flujo de transferencia de archivos, ambos flujos fueron asignados a diferentes colas de transmisión, el CAM puede ser el causante de dicha interferencia en el flujo VoIP. El mecanismo de resolución de colisión interna decide entre las diferentes colas, a la trama que tenga la mayor prioridad para transmitir. En esta prueba todos los dispositivos inalámbricos utilizan el acceso al medio implantado por WMM, significando que los flujos presentes compiten bajo el mismo algoritmo de acceso al medio, diferenciados por su categorización.

Gráficas de diferencia y *jitter*

La figura 6.9 nos muestra gráficas del *jitter* y de la *diferencia* característica de la llamada número 10, en la prueba de 20 simultáneas bajo los escenarios *lo*, no-WMM con flujo, WMM* con flujo y WMM con flujo. Los dos parámetros fueron calculados a partir de los paquetes entrantes en la computadora `Shirl`.

Esta figura nos muestra cómo es el retraso que tiene el flujo VoIP al viajar por la Red. Como puede observarse, en la figura 6.9-a los valores de *jitter* y de diferencia se encuentran por debajo de 1 ms, con un pico no mayor a los 18 ms. En la figura 6.9-a se utiliza el escenario *lo* (sin flujo) y es por tal motivo que resultan tiempos muy pequeños.

En la figura 6.9-b al encontrarse el flujo bajo el escenario no-WMM, el *jitter* se

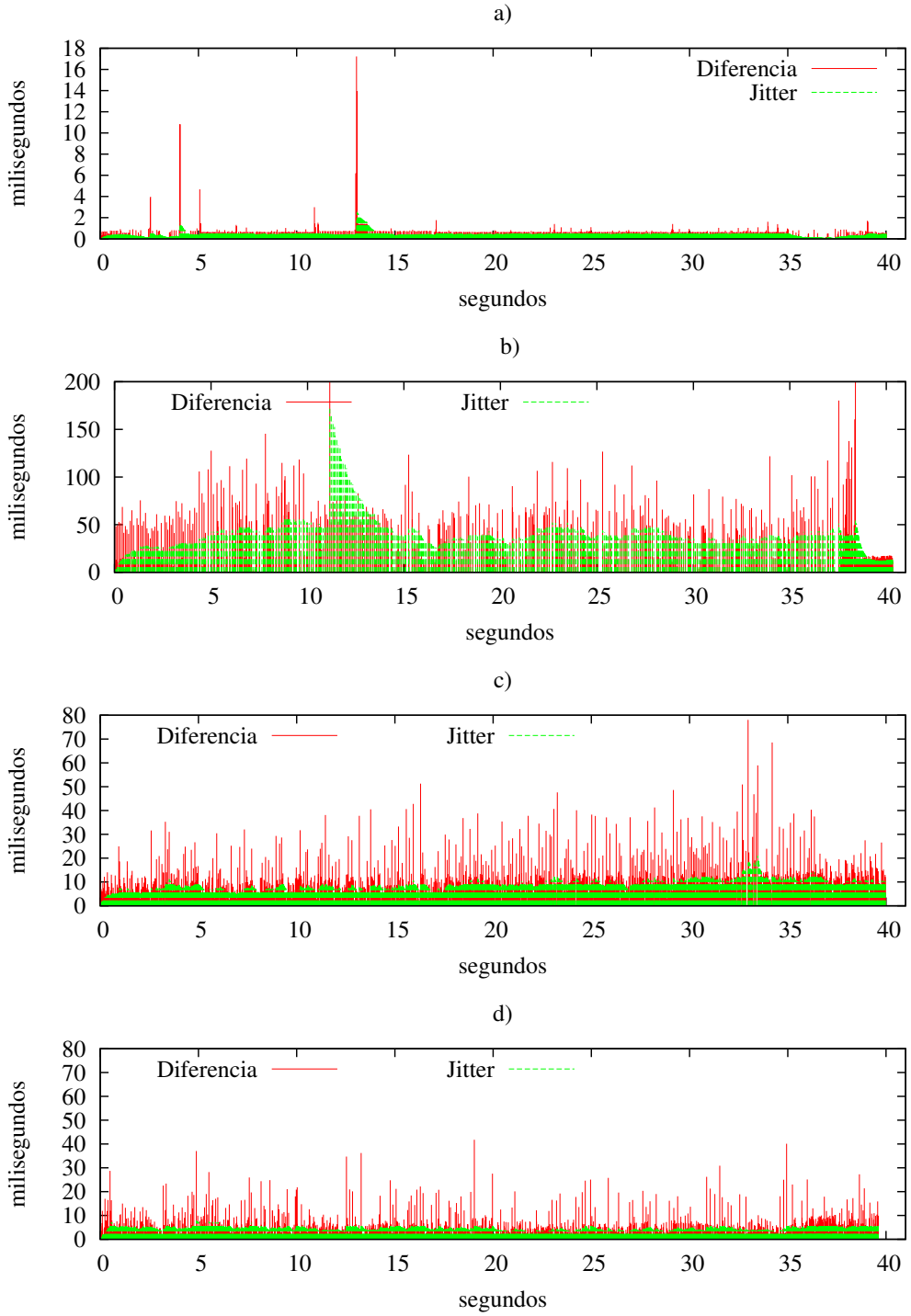

Figura 6.9: Diferencia y *jitter* de la llamada 10 de 20 simultáneas, en los escenarios a) *lo*, b) no-WMM con flujo, c) WMM* con flujo y d) WMM con flujo en `Shirl`

encuentra por debajo de los 50 milisegundo (ms) llegando a un valor pico de menos de 170 ms. En un intervalo de 50 ms existen 2.5 paquetes RTP que transportan audio, al utilizar el codificador PCMU, esto nos da una idea de una entrega de audio con un retraso considerable.

La gráfica 6.9-c muestra un *jitter* que está alrededor de los 10 ms, con valores que permanecen por encima de dicho valor en los últimos segundos. Esto refleja una mayor saturación al final de la prueba. Debe tomarse en cuenta que en los últimos segundos las llamadas creadas antes de la décima llamada están siendo terminadas. Obsérvese la figura 6.9-b en su último segundo, se nota una disminución drástica en el *jitter* y en la figura 6.9-c no se observa tal comportamiento.

En la figura 6.9-d el *jitter* siempre se encuentra por debajo de los 10 ms y permanece sin muchas variaciones dentro de los primeros 20 segundos. Dentro de los segundos del 20 al 35, hay unas pequeñas variaciones de *jitter* que oscilan por debajo al valor que permaneció casi constante al inicio de la llamada. A partir del segundo 35, el valor permanece casi constante y no se observa una disminución en el *jitter* e incluso la llamada termina antes del segundo 40.

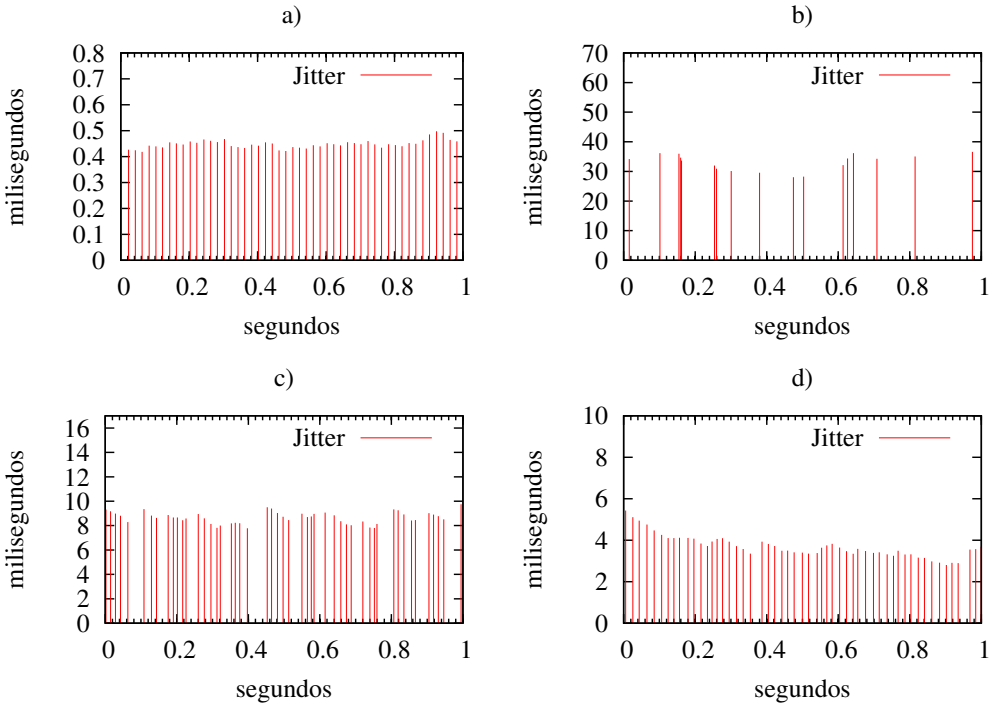

Figura 6.10: *Jitter* de la llamada 10 de 20 simultáneas del segundo 20 al 21, en los escenarios a) *lo*, b) no-WMM con flujo, c) WMM* con flujo y d) WMM con flujo en `Shirl`

La gráfica de *jitter* del escenario WMM con flujo en la figura 6.9-d resulta la más parecida al escenario *lo*, seguida del escenario WMM* y al final por el escenario

no-WMM. La gráfica 6.9-b muestra espacios en blanco (como en el segundo 15 aproximadamente) que son reflejados directamente en los valores que toma el *jitter*. Estos espacios son producto de un retraso en la red o incluso por la pérdida de paquetes VoIP.

En la figura 6.10 puede observarse una semejanza con la figura 6.9, que grafica el *jitter* del segundo 20 al 21. En un segundo hay 50 paquetes RTP cuando se utiliza el codificador PCMU. En la gráfica 6.10-a hay 50 impulsos de *jitter* distribuidos muy uniformemente. En la figura 6.10-b hay 17 impulsos de *jitter* que generan 340 ms de audio. En la figura 6.10-c hay 50 impulsos de *jitter* pero con un espaciado no muy uniforme. En la figura 6.10-d hay 53 impulsos de *jitter*, el excedente de 3 paquetes puede deberse a diversos factores, como es la repetición de paquetes o el rezago de paquetes en segundos anteriores.

La figura 6.11 muestra las gráficas de diferencia y jitter del flujo entrante en `Asterisk` de la llamada número 10 de las 20 simultáneas, para los escenarios *lo*, no-WMM, WMM* y WMM. En estos resultados los valores de *jitter* y diferencia son menores a los presentados anteriormente. El valor más significativo son los del escenario no-WMM que se muestra en la figura 6.11-b, con un valor de *jitter* por debajo de los 25 ms. En la figura 6.11-c y en la figura 6.11-d también se muestra una disminución en el valor del *jitter*.

6.6 Análisis de los parámetros VoIP de los escenarios con 15 y 20 llamadas

En la sección 5.7.1 se definen una serie de parámetros VoIP que fueron extraidos de las pruebas realizadas. En esta sección se presentan dichos parámetros que resultaron de los diferentes escenarios, específicamente para aquellos con 15 y 20 llamadas simultáneas. Los datos que se presentan son el valor promedio de los parámetros resultantes en cada una de las llamadas simultáneas.

En la tabla 6.1 se muestran los parámetros promedio de los escenarios sin flujo con 15 y 20 llamadas simultáneas entrantes en la computadora `Shirl`. Se puede apreciar que el número de paquetes recibidos en todos los resultados, cumplen con el tiempo de audio que registró la duración real promedio de las llamadas. La prueba WMM con 15 llamadas registra un promedio de 0.7 de paquetes perdidos y las restantes registran ningún paquete perdido. El valor delta en la prueba de 15 llamadas tiene una diferencia de +9.88 ms en el escenario no-WMM, respecto al escenario *lo* y una diferencia de +20.82 ms en el escenario WMM respecto al escenario *lo*. El *jitter* máximo es de más del doble en el escenario no-WMM y casi el triple en el escenario WMM, respecto al escenario *lo*, siendo éste de 1.56 ms (un valor pequeño). El *jitter* promedio registra un valor menor a 2 ms para 15 y 20 llamadas en los escenarios mostrados en la tabla. El valor delta en 20 llamadas simultáneas se incrementa un poco para los escenarios *lo* y no-WMM mientras que en el escenario WMM se decrementó, esto respecto al

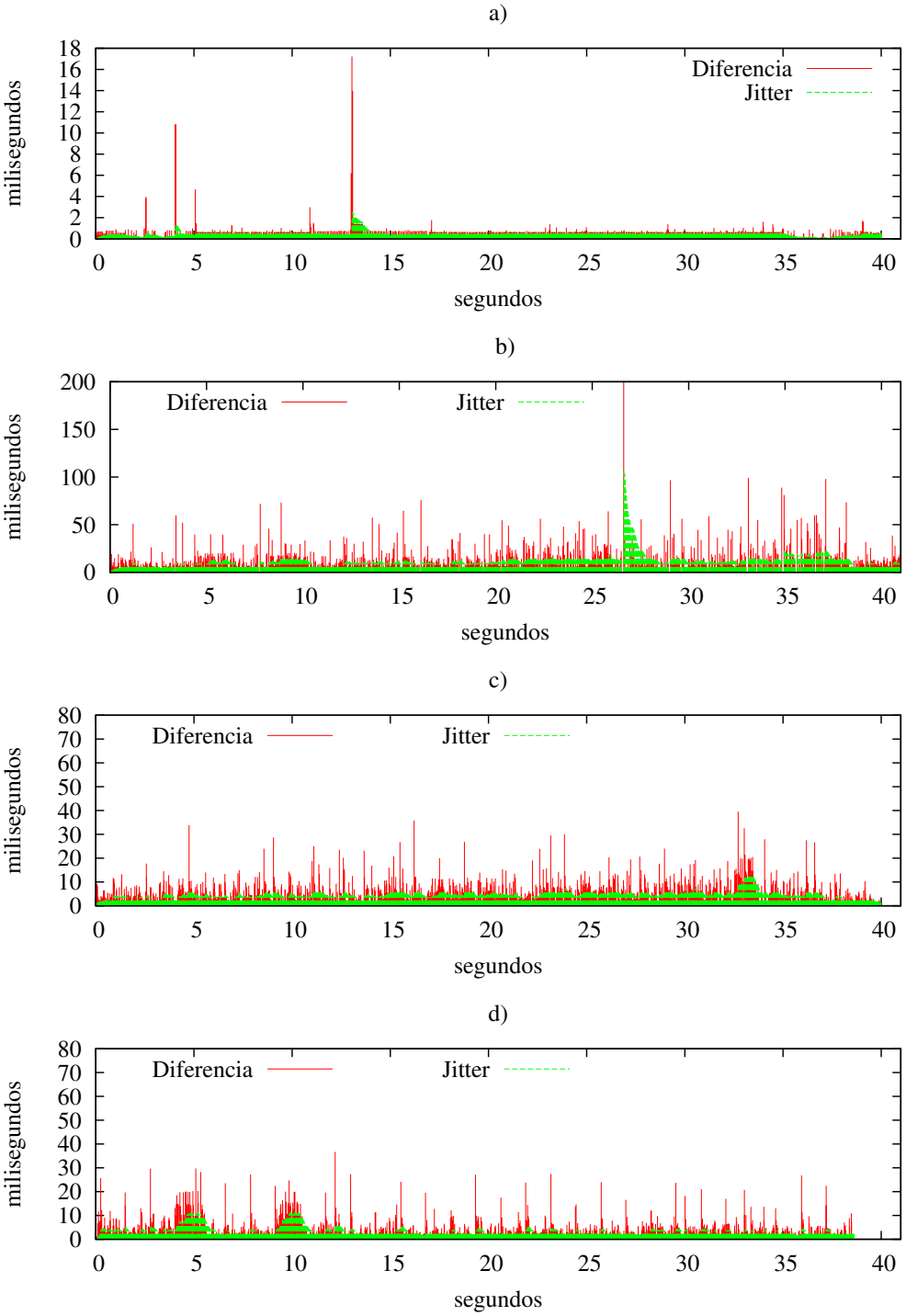

Figura 6.11: Diferencia y *jitter* de la llamada 10 de 20 simultáneas, en los escenarios a) *lo*, b) no-WMM con flujo, c) WMM* con flujo y d) WMM con flujo en `Asterisk`

Escenario	Paquetes recibidos	Paquetes perdidos	Delta max. (ms)	*Jitter* max. (ms)	*Jitter* prom. (ms)	Duración (s)
15 llamadas						
lo	2001.6	0.0	29.02	1.56	0.27	40.01
no-WMM	2001.73	0.0	38.90	3.50	1.30	40.00
WMM	2001.8	0.07	49.84	4.63	1.63	40.01
20 llamadas						
lo	2002.4	0.0	37.24	2.59	0.47	40.03
no-WMM	2002.95	0.0	44.23	10.19	1.69	40.02
WMM	2002.9	0.0	47.54	4.71	1.78	40.03

Tabla 6.1: Parámetros promedio en los escenarios sin flujo de 15 y 20 llamadas simultáneas entrantes en `Shirl`

valor delta en 15 llamadas. El *jitter* máximo en 20 llamadas se incrementó en todos los escenarios respecto al valor en 15 llamadas, siendo el más notorio en el escenario no-WMM.

La tabla 6.2 muestra los parámetros que se obtuvieron en la computadora `Asterisk` de las 15 y 20 llamadas simultáneas. Estos resultados muestran una pequeña mejoría en los parámetros respecto a los ya presentados en la computadora `Shirl`.

En todos estos casos, los parámetros resultantes bajo el escenario no-WMM y WMM no distan por mucho a los resultados del escenario *lo*, pero se puede apreciar una distinción entre ellos: el escenario no-WMM presenta valores más cercanos a la prueba *lo* a diferencia del escenario WMM. Tómese en cuenta que en estos escenarios sólo existe el flujo VoIP.

Escenario	Paquetes recibidos	Paquetes perdidos	Delta max. (ms)	*Jitter* max. (ms)	*Jitter* prom. (ms)	Duración (s)
15 llamadas						
no-WMM	2001.67	0.0	37.51	2.86	0.99	40.01
WMM	2001.74	0.0	41.27	3.56	1.20	40.01
20 llamadas						
no-WMM	2002.7	0.0	40.92	10.32	1.31	40.03
WMM	2002.5	0.0	40.45	3.83	1.33	40.03

Tabla 6.2: Parámetros promedio en los escenarios sin flujo de 15 y 20 llamadas simultáneas entrantes en `Asterisk`

En la tabla 6.3 se muestran los resultados de los parámetros promedio VoIP, que se obtuvieron en los escenarios donde existe un flujo de transferencia de archivos (además del flujo VoIP), de las pruebas de 15 y 20 llamadas simultáneas entrantes en la computadora `Shirl`. En esta tabla, el escenario no-WMM con flujo con 15

llamadas presenta un promedio de paquetes recibidos de poco más de la mitad de los paquetes necesarios, para cubrir los 42.06 segundos promedio reales que duraron las conversaciones, perdiéndose el resto de paquetes. El valor promedio delta máximo es de 153.76 ms, significando que durante ése intervalo de tiempo no existió ningún otro paquete RTP. El valor de *jitter* máximo es de 104.09 ms, dicha cifra es la mayor de todas en los escenarios con flujo para 15 llamadas simultáneas. El *jitter* promedio registra un valor de 34.51 ms, significando que en promedio los paquetes tienen un espaciado o llegaron en desorden por más del tiempo en que el siguiente paquete debería llegar.

El escenario WMM* con flujo cumple con el número de paquetes requeridos para cubrir el tiempo real de duración de la llamada y con casi cero paquetes perdidos. A diferencia del escenario WMM con flujo, que muestra paquetes perdidos y el audio que crean el número de paquetes recibidos ($1930.53 * 0.020 = 38.6106$ segundos) menos el tiempo real de duración de la llamada (38.77 segundos), resulta en un faltante de 159.4 ms de audio. El tiempo de audio que crean los paquetes perdidos en el escenario WMM con flujo es de $37.53 * 20 = 750.6$ ms, cantidad de tiempo que supera el audio faltante en la conversación, significando que bajo este escenario los paquetes que fueron reenviados si llegaron a su destino minimizando el tiempo faltante en la conversación. El valor delta máximo resulta menor en el escenario WMM* con flujo, respecto al escenario WMM con flujo, el *jitter* máximo y *jitter* promedio resultan mejores en el escenario WMM con flujo.

En la prueba de 20 llamadas el escenario no-WMM con flujo tiene un menor número de paquetes recibidos y paquetes perdidos respecto a la prueba anterior. El tiempo de audio que pudo ser recibido es de 17.61 segundos y el tiempo de audio que se perdió fue de 15.72 segundos, por tanto, no se recibió ni la mitad del audio necesario que duró realmente la conversación y el tiempo de audio perdido es casi igual al audio recibido. Esto es reflejado directamente en los valores de delta máximo y *jitter* máximo. El valor de *jitter* promedio fue de 35.15 ms que no muestra un incremento considerable respecto a la prueba de 15 llamadas.

El escenario WMM* con flujo presenta valores normales, en cuanto paquetes recibidos y paquetes perdidos en la prueba de 20 llamadas. Los valores de delta máximo y *jitter* máximo se incrementan un poco, aunque el *jitter* promedio es mejorado respecto a la prueba de 15 llamadas.

El escenario WMM con flujo tiene un tiempo de audio recibido de $1965.85 * 0.020 = 39.297$ segundos y menos el tiempo real de la duración de la conversación 39.68, resulta en un tiempo de audio faltante de 383 ms. El tiempo de audio que generan los paquetes perdidos es de $24.95 * 20 = 499$ ms, notandose un traslape de $|383 - 499| = 116$ ms que pudo ser de paquetes re-enviados y recibidos.

En la tabla 6.4 se muestran los valores de los parámetros promedio de los escenarios con flujo con 15 y 20 llamadas entrantes en la computadora `Asterisk`. Estos resultados claramente muestran que el acceso al medio que tuvo la computadora `Shirl`, para enviar sus flujos VoIP es mejor que el acceso al medio que tiene el PA

Escenario	Paquetes recibidos	Paquetes perdidos	Delta max. (ms)	*Jitter* max. (ms)	*Jitter* prom. (ms)	Duración (s)
15 llamadas						
no-WMM cf	1138.4	838.47	153.76	104.09	34.51	42.06
WMM* cf	1999.87	0.07	72.37	16.27	12.33	39.96
WMM cf	1930.53	37.53	94.73	12.46	5.52	38.77
20 llamadas						
no-WMM cf	880.55	786.05	248.32	169.10	35.13	39.13
WMM* cf	2000.5	0.0	95.78	21.24	8.88	39.98
WMM cf	1964.85	24.95	64.63	8.16	4.65	39.68

Tabla 6.3: Parámetros promedio en los escenarios con flujo de 15 y 20 llamadas simultáneas entrantes en `Shirl`

para enviar los paquetes VoIP de la computadora `Asterisk` conectada vía *Ethernet*.

El escenario no-WMM con flujo cumple con el tiempo de audio recibido respecto al tiempo real de la conversación para 15 llamadas. Se registran muy pocos paquetes perdidos, aunque el valor delta máximo no es bajo. El *jitter* máximo y *jitter* promedio es de casi al doble respecto a la prueba WMM* con flujo.

El tiempo de audio generado por los paquetes recibidos en la prueba de 15 llamadas, en los escenarios WMM* con flujo y WMM con flujo cumplen con el tiempo real de duró la conversación y además presentan cero paquetes perdidos. Los valores delta en estos escenarios es mejor que en los resultados anteriores, el *jitter* máximo y el *jitter* promedio se mejora, siendo WMM con flujo el escenario que muestra mejores resultados.

En la prueba de 20 llamadas el escenario no-WMM con flujo presenta un número considerable de paquetes perdidos, aunque el *jitter* promedio no es alto. Los escenarios WMM* con flujo y WMM con flujo cumplen con el tiempo de audio requerido, el número de paquetes perdidos es despreciable en ambos escenarios. Los parámetros *jitter* máximo y *jitter* casi no se incrementan respecto a la prueba de 15 llamadas.

En los resultados de escenarios con flujo de transferencia de archivos, el escenario no-WMM con flujo fue quien tuvo un desempeño muy deficiente en el flujo VoIP en todos los parámetros. El escenario WMM* con flujo es quien mejor cumple en el número de paquetes recibidos, aunque no muy alejado del escenario WMM con flujo. El escenario WMM con flujo registra más paquetes perdidos que el escenario WMM* con flujo, pero la suma del número de paquetes perdidos más el número de paquetes recibidos en el escenario WMM con flujo, crean un tiempo de audio que supera la cantidad de tiempo que duró realmente la conversación. El escenario no-WMM con flujo registra un tiempo entre el inicio y término de llamada (duración de la conversación) que sobrepasa los 40 segundos establecidos en la prueba. El escenario WMM* con flujo fue quien más cumplió con este tiempo y el escenario WMM con flujo siempre

Escenario	Paquetes recibidos	Paquetes perdidos	Delta max. (ms)	*Jitter* max. (ms)	*Jitter* prom. (ms)	Duración (s)
15 llamadas						
no-WMM cf	2104.73	6.4	136.91	16.30	7.26	42.30
WMM* cf	1999.2	0.0	66.96	8.43	4.27	39.96
WMM cf	1939.73	0.0	81.80	9.25	3.50	38.77
20 llamadas						
no-WMM cf	1638.85	289.95	131.0	106.50	11.54	39.09
WMM* cf	2000.45	0.15	72.30	13.82	4.54	40.0
WMM cf	1930.75	0.0	56.62	12.40	3.62	38.59

Tabla 6.4: Parámetros promedio en los escenarios con flujo de 15 y 20 llamadas simultáneas entrantes en `Asterisk`

estuvo por debajo de dicho número. En su gran mayoría, los valores de *jitter* máximo y *jitter* promedio en el escenario WMM con flujo fueron los más bajos respecto a los los demás escenarios con flujo.

Capítulo 7

Conclusiones y trabajo a futuro

Conclusiones

Se obtuvo una plataforma de comunicación basada en SIP, resultando en un laboratorio en el cual se provee un servicio de telefonía a través de IP. Además, la plataforma converge dos dominios de comunicación, el dominio de IP y el dominio de la PSTN. La heterogeneidad de la plataforma no se refiere solamente a la diversidad de dispositivos que pueden ser intercomunicados, si no también, a la utilización de redes alámbricas e inalámbricas. A pesar de que la heterogeneidad, en redes alámbricas e inalámbricas, en IP se encuentra a nivel de configuración, la provisión de QoS en redes heterogéneas no resulta de una manera directa.

El servidor Asterisk es utilizado como un registrador y *proxy* en la plataforma. Asterisk es el servidor, que de manera centralizada, enlaza las comunicaciones VoIP junto con la PSTN. Para la instalación, configuración y administración de Asterisk se requieren de bastos conocimientos sobre el tema, aún contando con herramientas que ayuden a su instalación, es indispensable conocer los fundamentos básicos, tal fue el caso de configurar la tarjeta TDM, que no resultó una configuración directa, debido principalmente al contar con una línea digital requiriéndose una línea análoga en los puertos Oficina de Intercambio Foráneo (FXO).

Mediante la plataforma, son posibles comunicaciones VoIP entre terminales tan diversas como PC de escritorio, computadoras portátiles, dispositivos móviles (PDAs y *smartphones*) e IP-*phones*, que se encuentran conectadas por medio inalámbrico Wi-Fi como por el medio alámbrico Ethernet, teléfonos conectados al PBX del Cinvestav por medio de las líneas analógicas, conectadas a los puertos FXO y a teléfonos analógicos comunes de la PSTN conectados a los puertos FXS.

Se mostró la interoperabilidad de diversos sistemas operativos embebidos (`Symbian`, *Windows Mobile* y *Familiar Linux*) y no embebidos (*Linux* y *Windows*) mediante la biblioteca PJ que implementa varios protocolos para crear *softphones*. La comprensión y uso de esta biblioteca fue la actividad que tomó más tiempo, la cantidad de información y funcionalidad de la biblioteca es muy grande y que sólo fue posible probar la portabilidad entre distintos sistemas operativos además de agregar mecanismos

de provisión de QoS basado en DiffServ.

La provisión de QoS en la arquitectura se realizó mediante la utilización de dispositivos que cuentan con mecanismos de QoS, en el medio alámbrico el conmutador Ethernet se requirió configurar la cola de servició en la que un paquete será enviado. Para el medio inalámbrico se requirió de la instalación de un *firmware* diferente al provisto por el fabricante para incorporar otros mecanismos de QoS. El soporte WMM es el mecanismo de QoS para el medio inalámbrico, cuyos requerimientos fueron cumplidos principalmente al marcar los paquetes SIP y RTP con un DSCP apropiado hecho directamente por la aplicación.

Para realizar el marcado de los paquetes por la aplicación, se requirió de incorporar código en la biblioteca PJ, donde el principal problema fue la diversidad de dispositivos que la biblioteca soporta. En los sistemas como *Linux* y *Windows Mobile* 6.0 se incorporó la manera de poder marcar dichos paquetes por la biblioteca, mientras que para el sistema *Windows XP* no se realizó principalmente por la falta del mecanismo utilizado para marcar los paquetes. Bajo el sistema *Familiar Linux* utilizado en dispositivos móviles, resulta la imposibilidad de agregar QoS por medio del soporte WMM, debido al no contar con una tarjeta inalámbrica que cuente con el soporte WMM, caso contrario al módulo del kernel de Linux *iwlwifi* para computadoras personales o de escritorio, que agrega mecanismos de WMM.

Los protocolos más estudiados en esta tesis fueron SIP y RTP. El esquema funcional de SIP fue probado, al utilizar tecnologías como Asterisk y la biblioteca PJ, el conocimiento adquirido del protocolo sirvió para la elaboración de un *softphone* básico. Resultó indispensable una revisión del protocolo RTP pese a su existente implementación en la biblioteca PJ, debido a la necesidad de analizar la QoS que se encuentra en la plataforma. El entendimiento del protocolo RTP proporcionó información acerca de cómo obtener parámetros para el análisis de los flujos RTP y así medir de una manera objetiva la QoS en la plataforma. Tanto es así, que se requirió de todo un capítulo para el plantear las pruebas de QoS y de como se extraen los parámetros VoIP, y otro capítulo para el análisis de los resultados.

El nivel QoS en la plataforma solamente se analizó para el medio inalámbrico, principalmente porque es el eslabón más débil en la transmisión de los paquetes. El probar el nivel de QoS en el medio alámbrico quedó sujeto a la imposibilidad de saturar el conmutador Ethernet. El *firmware* instalado en el PA proporciona un medio por el cual es posible configurar los parámetros de las CA del soporte WMM.

Las pruebas en el medio inalámbrico se plantearon a manera de obtener información de cómo es el trato de paquetes, bajo diferentes escenarios. Los escenarios se distinguen principalmente por dos características, la primera de ellas es el algoritmo del CAM que utilizan, habiendo tres posibles configuraciones: al utilizar el CAM IEEE 802.11 (sin provisión QoS), al utilizar el CAM IEEE 802.11e mediante el soporte WMM y por último la utilización de ambos algoritmos. La segunda característica es cuando existe sólo el flujo VoIP y cuando existe además un flujo de baja prioridad, en tal caso el de una transferencia de archivos.

Los resultados de las pruebas muestran que tanto el CAM en 802.11 y en WMM, bajo condiciones de sólo un tipo de tráfico como el de VoIP, cumplen con los requerimientos de un flujo de RTP y cuando existe otro tipo de flujo además del flujo RTP, este último se ve seriamente afectado. Cuando se hace uso del CAM de IEEE802.11 tanto para el flujo VoIP como para el flujo de transmisión de archivos, se utiliza el total ancho de banda disponible, pero los flujos RTP se ven seriamente afectados, mostrándose un nivel de *jitter* considerable reflejándose directamente en la entrega no óptima de paquetes RTP e incluso existe un número elevado de paquetes perdidos.

Cuando se utiliza el CAM de 802.11 para el flujo de transferencia de archivos y el CAM de WMM para los flujos RTP en la mejor CA, se utiliza todo el ancho de banda disponible, pero el *jitter* todavía muestra una inestabilidad en la entrega de paquetes RTP y bajo este esquema el número de paquetes perdidos es de casi cero.

Al utilizar el CAM de WMM tanto para el flujo VoIP como para el flujo de transferencia de archivos, no se logra la utilización total del ancho de banda disponible, debido a que el flujo VoIP se encuentra categorizado como de mejor prioridad y el flujo de transferencia de archivos como el de menor prioridad. El flujo de transferencia de archivos sólo utiliza una porción del ancho de banda, definido por la caracterización de los parámetros de su CA configurado y al ser de baja prioridad sólo utiliza unos cuantos cientos de *kbits* del ancho de banda, cuando se encuentra un flujo de mayor prioridad. Bajo este esquema, los flujos de RTP cumplen con los requerimientos de RTP, al obtenerse bajos niveles de *jitter*, aunque ligeramente persiste una pérdida de paquetes RTP. Pese a la pérdida de paquetes RTP, la entrega de estos resulta óptima aún cuando un flujo que consume mucho ancho de banda se encuentra presente.

La plataforma presentada en esta tesis resulta atractiva en varios ámbitos, tanto comerciales como de investigación. La Coordinación General de Tecnologías de la Información y las Comunicaciones (CGSTIC) se ha planteado como un objetivo principal el apoyar al Cinvestav para que éste se convierta en una ciudad digital, científica y tecnológica mediante el uso del estado del arte de la tecnología. Dicha coordinación ha mostrado interés en la plataforma y se proponen la intercomunicación de varias unidades y departamentos del Cinvestav mediante VoIP.

Por parte del Cinvestav unidad Tamaulipas, se nos ha propuesto implantar un sistema como el presentado en la tesis, intercomunicando tanto dispositivos con capacidad VoIP como con el PBX presente en la unidad y sobre todo con vías a lograr una comunicación entre unidades o departamentos del Cinvestav.

Trabajo a futuro

Existen indicios de que en el sistema *Symbian*, mediante el uso apropiado de su API de *sockets*, es posible el marcado de los paquetes salientes, pero debido a ser un proceso muy elaborado, además de requerir una comprensión mayor tanto del sistema operativo como de la propia biblioteca PJ, son razones suficientes para proponer la integración del marcado de paquetes por la aplicación como trabajo a futuro.

La plataforma mostrada resultó ser a pequeña escala, principalmente debido a la poca infraestructura, pero mediante otras tecnologías como los son STUN, ICE y TURN es posible la comunicación a grandes distancias, además de sobrellevar el problema de atravesar redes NAT tanto como *firewalls*. Otra tecnología a considerar es el Hallazgo de Números Universales Distribuido (DUNDi) [22], un sistema punto a punto para localizar puertas de enlace en Internet (como por ejemplo otros servidores Asterisk) para proveer servicios telefónicos, al utilizar esta tecnología con múltiples servidores Asterisk es posible la descentralización del servicio VoIP.

La aportación del trabajo realizado encamina a la investigación y prueba de las tecnologías comentadas anteriormente, para ser incorporadas a la plataforma. Actualmente en la plataforma se brinda el servicio de VoIP y como trabajo a futuro se encuentra el incorporar un servicio de mensajería y presencia. Asterisk no es utilizado como un servidor de mensajería y deberá realizarse una investigación pertinente para la incorporación de otros servicios. Cabe mencionar que otros servicios como el de mensajería y presencia son logrados mediante SIP.

Bibliografía

[1] 3G-Americas. IP Multimedia Subsistem IMS Overview and applications. White paper, July 2004.

[2] IEEE Std 802.11-1999. Part 11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Reference number ISO/IEC DIS 8802-11, Reaffirmed June 2003.

[3] IEEE Std 802.11d 2004. *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*. June 2004.

[4] IEEE Std 802.11e 2005. *Wireless LAN MAC and Physical Layer specifications Amendment 8: MAC Quality of Service Enhancements*. November 2005.

[5] Wi-Fi Aliance. WiFi CERTIFIED for WMM - Support for Multimedia Applications with Quality of Service in Wi-Fi Networks. White paper, September 2004.

[6] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. RFC4594, 2006.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC2475, 1998.

[8] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC1633, 1994.

[9] Mihaela Cardei, Ionut Cardei, and Ding-Zhu Du. *Resource Management in Wireless Networking (Network Theory and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, and J. Segers. Megaco Protocol Version 1.0. RFC3015, 2000.

[11] B. Davie, A. Charny, J. C. R. Bennet, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC3246, 2002.

[12] Ericsson. Evolution towards converged services and networks. White paper, April 2005.

[13] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC2327, 1998.

[14] William C. Hardy. *VoIP Service Quality*, chapter 1, pages 10–18. McGraw-Hill, 2003.

[15] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC2597, 1999.

[16] Intel. Providing QoS in WLANs - How the IEEE 802.11e Standard QoS Enhancements Will Affect the Performance of WLANs. White paper, 2004.

[17] Y. Kondo and H. S. Matthews. Comparative analysis of traditional telephone and voice-over-Internet protocol (VoIP) systems. In *ISEE '04: Proceedings of the International Symposium on Electronics and the Environment*, pages 106–111, Washington, DC, USA, 2004. IEEE Computer Society.

[18] George Konstantoulakis and Morris Sloman. Call Management Policy Specification for the Asterisk Telephone Private Branch Exchange. In *POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 251–255, Washington, DC, USA, 2007. IEEE Computer Society.

[19] Linksys. *Switch Gigabit 10/100/1000 de 8 puertos Guía del usuario*. Cisco Systems, Inc., 2006.

[20] Indu Mahadevan and Krishna M. Sivalingam. Architecture and experimental framework for supporting QoS in wireless networks using differentiated services. *Mob. Netw. Appl.*, 6(4):385–395, 2001.

[21] Mario Marchese. *QoS over heterogeneous networks*. John Wiley & Sons, Ltd., England, 2007.

[22] Jim Van Meggelen, Jared Smith, and Leif Madsen. *Asterisk: The Future of Telephony*. O'Reilly Media, Inc., 2nd edition, 2007.

[23] Yongwan Park and Fumiyuki Adachi. *Enhanced Radio Access Technologies for Next Generation Mobile Communication*, pages 1–37. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[24] Radvision. IMS SIP and Signaling, The RADVISION Perspective. White paper, 2006.

[25] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC3031, 2001.

[26] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. draft-ietf-mmusic-ice-19, 2007.

[27] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). draft-ietf-behave-turn-10, 2008.

[28] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC3261, 2002.

[29] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC3489, 2003.

[30] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC3551, 2003.

[31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC3550, 2003.

[32] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC2326, 1998.

[33] Stardust. QoS protocols & architectures. White paper, July 1999.

[34] L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC2205, 1997.

Páginas electrónicas

[35] W. Borgert. Setting up a cross development environment on debian gnu/linux. 2008, [Online]. Available: http://people.debian.org/~debacle/cross/.

[36] Coms & Network. Using sip with nokia series60 and asterisk. 2008. [Online]. Available: http://www.newlc.com/Using-SIP-with-Nokia-Series60-and.html.

[37] Counterpath Corp. Counterpath. 2008. [Online]. Available: http://www.counterpath.com.

[38] Joseph Davies. Qos support in windows. 2008, [Online]. Available: http://technet.microsoft.com/en-us/magazine/cc162504.aspx.

[39] DD-WRT. dd-wrt.com. 2008. [Online]. Available: http://www.dd-wrt.com/dd-wrtv3/index.php.

[40] DD-WRT. Supported devices. 2008. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Supported_Devices.

[41] es.wikipedia.org. Interactive voice response. 2008. [Online]. Available: http://es.wikipedia.org/wiki/Interactive_Voice_Response.

[42] John Finlay. Pygtk 2.0 tutorial. 2008, [Online]. Available: http://pygtk.org/pygtk2tutorial/index.html.

[43] handhelds.org. Familiar build setup. 2008, [Online]. Available: http://familiar.handhelds.org/build-setup.html.

[44] Digium Inc. Asterisk. 2008. [Online]. Available: http://www.asterisk.org/.

[45] Digium Inc. Tdm400p documentation. 2008. [Online]. Available: http://www.digium.com/en/supportcenter/documentation/viewdocs/TDM400P.

[46] intellinuxwireless.org. Intel wireless wifi link drivers for linux. 2008, [Online]. Available: http://intellinuxwireless.org/?n=Info.

[47] Thenu Kittappa. Aruba advanced qos: Qos mechanisms. 2008, [Online]. Available: https://edge.arubanetworks.com/article/aruba-advanced-qos-qos-mechanisms.

[48] Forum Nokia. Increasing wlan power efficiency for full-duplex voip and video applications. 2008, [Online]. Available: http://wiki.forum.nokia.com/index.php/TSS000650_-_Increasing_WLAN_power_efficiency_for_full-duplex_VoIP_and_Video_applications.

[49] Zapata Telephony Organization. Welcome to zapata telephony! 2008. [Online]. Available: http://www.zapatatelephony.org/.

[50] pjsip.org. Download source code. 2008. [Online]. Available: http://www.pjsip.org/download.htm.

[51] pjsip.org. Getting started, python sip tutorial. 2008. [Online]. Available: http://trac.pjsip.org/repos/wiki/Python_SIP_Tutorial.

[52] pjsip.org. Pjsip. 2008. [Online]. Available: http://www.pjsip.org.

[53] Benny Prijono. Tos over transport udp for sip, rtp, rtcp. 2008. [Online]. Available: http://lists.pjsip.org/pipermail/pjsip_lists.pjsip.org/2008-April/002705.html.

[54] Richard Sharpe. Wireshark user's guide. 2008, [Online]. Available: http://www.wireshark.org/docs/wsug_html/.

[55] Handhelds.org Open source for handheld devices. Build cross tool chain howto. 2008. [Online]. Available: http://handhelds.org/moin/moin.cgi/BuildCrossToolchainHowto.

[56] Handhelds.org Open source for handheld devices. Gpe cross compilation. 2008. [Online]. Available: http://www.handhelds.org/moin/moin.cgi/GpeCrossCompilation.

[57] trac.pjsip.org. Building and debugging pjsip on symbian s60 3rd edition device using carbide c++. 2008. [Online]. Available: http://trac.pjsip.org/repos/wiki/DevelopingSymbianAppWithCarbide.

[58] uclibc.org. Toolchains. 2008, [Online]. Available: http://www.uclibc.org/toolchains.html.

[59] Steven J. Vaughan-Nichols. Linux hackers tackle wifi hassles. 2008, [Online]. Available: http://www.desktoplinux.com/news/NS4466236354.html.

[60] wiki.emqbit.com. Openembedded from scratch. 2008. [Online]. Available: http://wiki.emqbit.com/openembedded.

[61] Wikipedia. Dd-wrt. 2008. [Online]. Available: http://en.wikipedia.org/wiki/DD-WRT.

[62] Wikipedia. Linksys wrt54g series. 2008. [Online]. Available: http://en.wikipedia.org/wiki/Wrt54g.

[63] Wikipedia. Toolchain. 2008, [Online]. Available: http://en.wikipedia.org/wiki/Tool_chain.

[64] Wikipedia. Vmware. 2008, [Online]. Available: http://es.wikipedia.org/wiki/VMware.

Lista de acrónimos

3G	red de comunicación móvil de tercera generación
3GPP	Proyecto de Colaboración de Tercera Generación
ACELP	Predicción Lineal con Exitación por Código Algebraico
ACK	Reconocimiento
ADPCM	Modulación por Impulsos Codificados Diferencial y Adaptable
AF	Envío Asegurado
AGI	Interfaz de Puerta de Enlace
AIFS	Espacio Entre-Trama Arbitrario
AIFSN	Número AIFS
API	Interfaz de Programación de Aplicaciones
APSD	Entrega Automática de Ahorro de Enegía
ASCII	Código Americano Estándar para el Intercambio de Información
AU	Agentes de Usuario
AUC	Agente de Usuario Cliente
AUS	Agente de Usuario Servidor
BE	Mejor Esfuerzo
CA	Categoría de Acceso
CAM	Control de Acceso al Medio
CC	conteo CSRC
CELP	Predicción Lineal con Exitación por Código

CGSTIC	Coordinación General de Tecnologías de la Información y las Comunicaciones
CH	Coordinador Híbrido
CNAME	Nombre canónico
PC	Computadora Personal
CS	Clase de Servicio
CSeq	secuencia de comando
CSMA/CA	Acceso Múltiple por Detección de Portadora con Evasión de Colisiones
CSRC	Contribuidor fuente
CW	Ventana de Contienda
CWmax	Ventana de Contienda máxima
CWmin	Ventana de Contienda mínima
DCF	Función de Coordinación Distribuida
DHCP	Protocolo de la Configuración de *Host* Dinámico
DiffServ	Servicios Diferenciados
DIFS	Espacio de Entre-Trama Distribuido
DNS	Servidor de Nombres Dinámico
DS	DiffServ
DSCP	Código de Punto DiffServ
DUNDi	Hallazgo de Números Universales Distribuido
EDCA	Acceso Coordinado Distribuido Mejorado
EF	Envío Acelerado
EST	estación
ETSI	Instituto Europeo de Normas de Telecomunicaciones
FQDN	Nombre de Dominio Totalmente Calificado
FXO	Oficina de Intercambio Foráneo

FXS	Estación de Intercambio Foráneo
GSM	Sistema Global de Comunicaciones Móviles
GUI	Interfaz Gráfica de Usuario
HCCA	Acceso al Canal Controlado HCF
HCF	Función de Coordinación Híbrida
HTTP	Protocolo de Transferencia de Hipertexto
ICE	Establecimiento de Conectividad Interactiva
IEEE	Instituto de Ingenieros Eléctrico y Electrónicos
IETF	Fuerza de Trabajo de Ingenieros de Internet
iLBC	Codificador de tasa de Bit Bajo de Internet
IMS	Subsistema Multimedia IP
IntServ	Servicios Integrados
IP	Protocolo de Internet
ITU	Unión Internacional de Telecomunicaciones
ITU-T	Unión Internacional de Telecomunicaciones Sector de Telecomunicaciones
IVR	Respuesta de Voz Interactiva
LPC	Codificación Predictiva Linear
MEGACO	Protocolo de Control de Puertas de enlace de Media
MOS	Puntaje de Opinión Media
MPLS	Protocolo Múltiple de Conmutación por Etiquetas
MP-MLQ	Cuantificación Multipulso de Máxima Probabilidad
ms	milisegundo
MSDU	Unidad de Datos de Servicio CAM
NAT	Traducción de Dirección de Red
PA	Punto de Acceso

PBX Intercambio Privado de Ramificación

PCF Función de Coordinación de Punto

PCM Modulación por Codificación de Pulsos

PCMA PCM escalado *alaw*

PCMU PCM escalado μlaw

PCo Punto Coordinador

PCon Periodo de Contención

PDA Asistente Personal Digital

PHB Comportamiento por Salto

PLC Periodo Libre de Contención

P-QoS QoS percibida

PSTN Red pública de telefonía conmutada

QEST estación mejorada QoS

QoS Calidad de Servicio

QPA PA mejorado QoS

RFC Solicitud de Comentarios

RSVP Protocolo por Reservación

RTCP Protocolo de Control de Tiempo Real

RTP Protocolo de Tiempo Real

RTSP Protocolo de Flujo de Tiempo Real

SC Selector de Clase

SDK Kit de Desarrollo de Software

SDP Protocolo de Descripción de Sesión

SIFS Espacio Corto de Entre-Trama

SIP Protocolo de Inicio de Sesiones

SSRC	Sincronización fuente
STUN	Transversal Simple de UDP a través de NATs
TA	transferencia de archivos
TCP	Protocolo de Control de Transmisión
TLS	Seguridad de la Capa de Transporte
ToS	Tipo de Servicio
TSPECs	Especificaciones de Tráfico
TURN	Transversal que Utiliza Repetidores en torno a NAT
TXOP	Oportunidad de Transmisión
UDP	Protocolo de Datagrama de Usuario
UPC	Unidad de Procesamiento Central
URI	Identificador Uniforme de Recurso
VAC	Actividad de Compresión de Voz
VAD	Detección de Actividad de Voz
VoIP	Voz sobre IP
WCDM	Acceso Múltiple de Banda Ancha por División de Código
WLAN	Red de Área Local Inalámbrica
WMM	Multimedia Inalámbrica
WSM	Acceso WMM-Planificado
XML	Lenguaje Extensible de Marca