TRABAJO ESPECIAL DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DE MENSAJERÍA INSTANTÁNEA PARA CELULARES CON SOPORTE PARA WIFI® BASADA EN CÓDIGO ABIERTO CON PROTOCOLO SIP

Tutor académico: Prof. Carlos Moreno

Presentado ante la Ilustre
Universidad Central de Venezuela
por el Br. Coronel M., Gabriel G.
para optar al Título de
Ingeniero Electricista

Caracas, 2011

CONSTANCIA DE APROBACIÓN

Caracas, 02 de noviembre de 2011

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Gabriel G. Coronel M. titulado:

"DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DE MENSAJERÍA INSTANTANEA PARA CELULARES CON SOPORTE PARA WIFI BASADA EN CÓDIGO ABIERTO CON PROTOCOLO SIP"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención de Comunicaciones, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Luis Fernández
Jurado

Prof. Zeldivar Bruzual
Jurado

Prof. Carlos Moreno
Prof. Guía

AGRADECIMIENTOS

A mi familia que a lo largo de mi vida me ha ayudado a superar obstáculos y a cumplir mis metas. A todos les agradezco enormemente el apoyo que me han brindado, y su paciencia ante las interminables horas de estudio que esta carrera ha supuesto.

A mi novia que desde el segundo semestre ha estado a mi lado compartiendo todo tipo de momentos; de éxito, de estrés, de cansancio, de alegría. El apoyo de ella ha sido fundamental durante cada semestre de la carrera.

A mis amigos que han hecho mucho más grato el período universitario, y en especial a aquellos con los que he podido contar desde el primer semestre.

A mi tutor Carlos Moreno por haberme permitido ser parte de este proyecto, y por su valiosa ayuda y sabios consejos en la consecución de este trabajo.

A cada uno de los profesores que ayudó en mi formación estudiantil y personal, en especial a las profesoras Norma Guzmán y Gabbriela González.

Coronel M., Gabriel G.

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DE MENSAJERÍA INSTANTÁNEA PARA CELULARES CON SOPORTE PARA WIFI® BASADA EN CÓDIGO ABIERTO CON PROTOCOLO SIP

Tutor Académico: Carlos Moreno. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Institución: U.C.V. Opción: Comunicaciones. Trabajo de Grado 2011. 117 h. + anexos

Palabras claves: Session Initiation Protocol; SIP; Protocolo de Inicio de Sesiones; Java; J2ME; Código abierto; WiFi®; Mensajería instantánea; BlackBerry®; 802.11.

Resumen. El presente trabajo plantea el diseño y la implementación de una arquitectura de mensajería instantánea para celulares con WiFi® basada en código abierto con protocolo SIP. Los componentes de dicha arquitectura son evaluados con el fin de identificar cuáles de ellos pueden ser diseñados e implementados bajo código abierto. Inicialmente, se estudiaron las arquitecturas y protocolos de mensajería instantánea para redes inalámbricas basadas en telefonía celular y WLAN existentes, entre ellas la plataforma de mensajería instantánea de BlackBerry® cuya arquitectura propietaria se ha hecho muy popular hoy en día. La arquitectura creada se programó en lenguaje Java y debió constar de dos partes: una escrita en J2ME para implementarla en un celular y otra escrita en J2SE para implementarla en una computadora. La aplicación de la computadora fue diseñada para que esta última, actuara como intermediaria entre el celular y un servidor SIP. Una vez que ésta fue desarrollada, se observó su compatibilidad con otro cliente propietario. De igual forma, se estudió la compatibilidad de la aplicación de celulares con equipos de diversas marcas. Finalmente, se evaluó la posible implementación de la arquitectura en casos de estudios reales.

ÍNDICE GENERAL

Pág.

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

Cuando los comités de Alemania y Francia en una de las reuniones de los comités de los países europeos para fijar los estándares GSM propusieron un servicio para enviar mensajes entre dos equipos móviles bajo el nombre Servicio de Mensajes Cortos (SMS, Short Message Service), no habrán tenido idea del impacto que dicho servicio causaría en la humanidad. Tal es el impacto de SMS, que hoy en día cualquier móvil que esté en el mercado soporta dicho servicio, y no sólo eso, sino que es sumamente difícil encontrar a una persona que tenga un celular y no use tal servicio. La popularidad de SMS junto con la popularidad de Internet y de las redes de datos, dio aparición a otros sistemas que incorporaban ambas ideas. Todos estos servicios son conocidos popularmente como sistemas de mensajería inalámbrica y permiten la transmisión de mensajes en tiempo real, con una latencia determinada por todos los elementos que componen al sistema telemático. Tales retardos se deben, a la arquitectura de los dispositivos de red, en relación a la conmutación, el enrutamiento, procesamiento, propagación y tamaño de los mensajes.

Algunos proveedores de servicios de Internet inalámbrico ofrecen sistemas de mensajería sobre redes Internet, tal como lo constituye la plataforma BlackBerry®, basándose en un código informático propietario en los equipos celulares y equipos servidores también propietarios. Adicionalmente, algunos equipos portátiles, incorporan funcionalidades de redes inalámbricas, como por ejemplo WiFi® o BlueTooth®.

Por otro lado, el protocolo SIP, es un protocolo utilizado ampliamente en sistemas de voz sobre IP y mensajería en el mundo entero, dada su fácil implementación, y su sencillez, a diferencia de otros como por ejemplo H.323 que es más denso.

El presente trabajo se centra en la creación de un sistema de mensajería instantánea basado en código abierto para celulares con WiFi® para que sirva como una herramienta a implementarse en la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Central de Venezuela para apoyar al sistema de educación a distancia, permitiendo la interacción de un profesor con sus estudiantes o tesistas con una alta disponibilidad en forma independiente de las aplicaciones propietarias del proveedor de servicios de telefonía celular. De igual forma, se sientan las bases para que sea posible en un futuro implementar la tecnología IP Multicast en la arquitectura desarrollada y poder optimizar el ancho de banda enviando mensajes a un grupo de usuarios.

Para la obtención de dicha arquitectura, se requirió el diseño y desarrollo de un software cliente de mensajería instantánea basado en Java. El software diseñado está constituido por dos subprogramas, uno para ejecutarse en computadoras y el otro para ejecutarse en equipos móviles con soporte del lenguaje Java Móvil (J2ME) y conexión a una red WiFi®.

En este trabajo inicialmente se estudian el estándar IEEE 802.11 (conocido como WiFi®), el protocolo SIP, la tecnología IP multicast, algunas de las más importantes arquitecturas de mensajería instantánea para redes inalámbricas basadas en telefonía celular y WLAN existentes, entrando en detalles acerca de la plataforma BlackBerry® y su servicio BlackBerry® Messenger, y el lenguaje Java. Seguidamente, se describe la metodología que permitió obtener la arquitectura propuesta, así como ciertas variaciones y consideraciones que debieron realizarse a lo largo de su desarrollo e implementación. Finalmente, tras un análisis detallado de la arquitectura construida, se establecen las conclusiones y recomendaciones.

PLANTEAMIENTO DEL PROBLEMA

Un sistema de mensajería instantánea con características propietarias, tal como es el perteneciente al sistema BlackBerry®, produce alta latencia innecesaria si los usuarios del servicio se encuentran en una misma red WiFi® o si utilizan la misma red celular. Inclusive, puede haber restricciones de seguridad por parte del proveedor de servicios para promover el uso de la aplicación propietaria de mensajería instantánea.

Por otro lado, la utilización del protocolo SIP y de la tecnología IP multicast en un producto, asegurarían su interoperabilidad con otro software cliente y/o servidor.

Es necesario realizar un diseño del sistema cliente contemplando: protocolos involucrados, definición de clases y configuración del servidor SIP. También se debe estudiar el rendimiento que tendría dicho sistema cliente, implementado sobre la red WiFi® de la Escuela de Ingeniería Eléctrica de la U.C.V.

OBJETIVOS

OBJETIVO GENERAL

El objetivo general del presente trabajo es diseñar e implementar una arquitectura de mensajería instantánea para celulares con soporte WiFi® basada en código abierto con protocolo SIP.

OBJETIVOS ESPECÍFICOS

Para el cumplimiento del objetivo general se debieron realizar los siguientes objetivos específicos:

- Estudiar las arquitecturas y protocolos de mensajería instantánea para redes inalámbricas basadas en telefonía celular y WLAN existentes.
- Comprender el funcionamiento del sistema BlackBerry® mediante el análisis de su arquitectura y de su plataforma de mensajería instantánea.
- Identificar los elementos de la plataforma que se pueden diseñar e implementar bajo código abierto.
- Diseñar el software cliente para mensajería instantánea utilizando Java Móvil (J2ME) bajo protocolo SIP. Adicionalmente, deben diseñarse unas clases que permitan usar multicast.
- Comprobar la interoperabilidad entre clientes de mensajería SIP y el software cliente desarrollado.
- Configurar el software diseñado y desarrollado, para varios casos de estudio reales. Ejemplo: reunión entre profesores, envío de información a alumnos, etc.

CAPÍTULO II

MARCO TEÓRICO

1. ESTÁNDAR IEEE 802.11 (WIFI®)

La necesidad de movilizarse sin perder los beneficios de la tecnología ha conllevado al desarrollo de numerosos avances tecnológicos. Uno de los más importantes y de los que mayor impacto ha tenido en la sociedad (tanto a nivel de hogar como a nivel empresarial), es el estándar IEEE 802.11, popularmente conocido como WiFi®.

El estándar 802.11 fue creado por un comité del IEEE (*Institute of Electrical and Electronics Engineers*) y tiene dos modos de trabajo:

1. En presencia de una estación base.
2. En ausencia de una estación base.

En el primer caso, toda la comunicación se realiza a través de una estación base, que en la terminología del 802.11 se conoce como punto de acceso (Figura 1-(a)). En el segundo caso, sin una estación base, las computadoras deben ser capaces de enviarse mensajes directamente entre sí. Cuando una red tiene esta última capacidad, es conocida generalmente como red ad hoc (Figura 1-(b)).

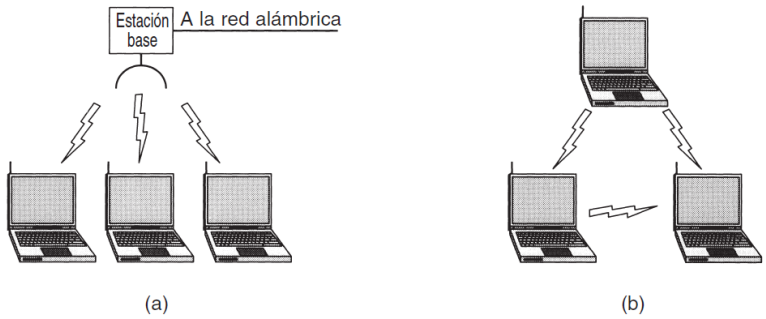


Figura 1. (a) Red con un punto de acceso. (b) Red ad hoc. [1]

Al diseñar este estándar se debían superar ciertos retos como: encontrar una banda de frecuencia adecuada, de preferencia mundial; enfrentar el hecho de que las señales de radio tienen un rango finito; asegurarse de que se mantuviera la privacidad de los usuarios; tomar en cuenta la vida limitada de las baterías; preocuparse por la seguridad humana; comprender las implicaciones de la movilidad de las computadoras y, por último, construir un sistema con suficiente ancho de banda para que fuese económicamente viable.

A mediados de la década de 1990, Ethernet era el estándar que dominaba las redes de área local (LAN) a nivel mundial. Debido a ello, el comité de la IEEE decide hacer que el nuevo estándar 802.11 fuese compatible con Ethernet sobre la capa de enlace de datos. Se diseñó entonces al estándar 802.11 de manera tal, que una red inalámbrica desde el exterior, fuese vista como una red Ethernet. La zona de cobertura que brinda un punto de acceso es conocida como celda, y la conexión entre un sistema 802.11 y el mundo exterior es conocida como portal (ver Figura 2). [1]

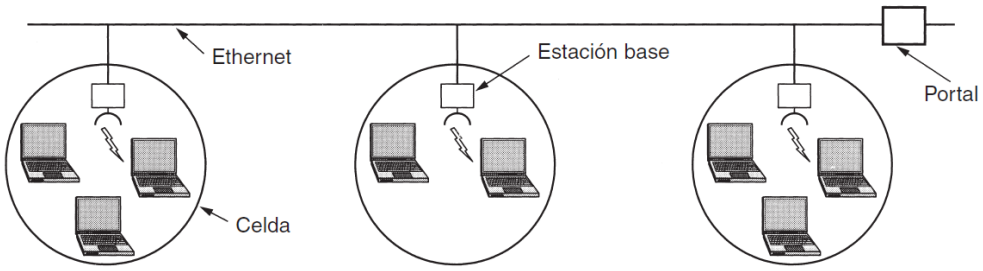


Figura 2. Red 802.11 de varias celdas conectada a un portal. [1]

1.1. Detalles técnicos

El estándar 802.11, define las características que deben tener las dos capas inferiores de una red inalámbrica. Esas capas son la capa física y la capa de enlace de datos.

1.1.1. Capa física

El primer estándar 802.11 (1997) especificaba tres técnicas de transmisión permitidas en la capa física: el método de infrarrojos, y dos métodos de radio de corto alcance que utilizaban las técnicas de Espectro Disperso con Salto de Frecuencia (FHSS, *Frequency Hopping Spread Spectrum*) o de Espectro Disperso de Secuencia Directa (DSSS, *Direct Sequence Spread Spectrum*). En 1999, se introdujeron dos nuevas técnicas para incrementar el ancho de banda, la de Multiplexación por División de Frecuencias Ortogonales (OFDM, *Orthogonal Frequency Division Multiplexing*) y la de Espectro Disperso de Secuencia Directa de Alta Velocidad (HR-DSSS, *High Rate Direct Sequence Spread Spectrum*). En 2001, una segunda modulación OFDM es introducida, pero a una banda de frecuencias distinta respecto a la primera. La diferencia entre esas técnicas está en la tecnología y en las velocidades alcanzables. Una breve descripción de las mismas se dará a continuación[1]:

(a) Infrarrojos

Utiliza transmisión difusa, es decir, no requiere línea visual, a una longitud de onda de 0.85 ó 0.95 µm. Permite alcanzar velocidades de 1 ó 2 Mbps y utiliza código Gray para realizar la codificación.

Los siguientes problemas hicieron que esta tecnología no fuese muy utilizada: las señales de infrarrojos no pueden penetrar paredes, se tenía un bajo ancho de banda, y la luz solar afecta este tipo de señales.

(b) FHSS

Utiliza un generador de números pseudoaleatorios para producir una secuencia que permite el salto en frecuencia. Todas las estaciones deben tener la misma secuencia para permanecer sincronizadas. El tiempo de permanencia en cada frecuencia es un parámetro ajustable, siempre y cuando sea menor a 400 ms. Estas últimas características proporcionan seguridad ya que un intruso que no conozca la secuencia ni el tiempo de permanencia, no es capaz de espiar las transmisiones. FHSS ofrece buena resistencia al desvanecimiento por múltiples trayectorias y es relativamente insensible a interferencia por lo que generalmente es utilizada en enlaces entre edificios. La desventaja de esta técnica es el bajo ancho de banda que provee. Al igual que la técnica de infrarrojos permite alcanzar velocidades de 1 ó 2 Mbps.

(c) DSSS

Al igual que las técnicas anteriores, DSSS también está limitada a 1 ó 2 Mbps. Cada bit se transmite como 11 chips y utiliza una modulación por Desplazamiento de Fase (PSK, *Phase Shift Keying*).

(d) OFDM

Permite transmisiones de hasta 54 Mbps en la banda de 5 GHz. Se utilizan 52 frecuencias distintas, 48 para datos y 4 para sincronización. Es considerada también como una técnica de espectro disperso ya que las transmisiones están presentes en múltiples frecuencias al mismo tiempo. Mejora la inmunidad a la interferencia, y utiliza un sistema de codificación basado en la modulación PSK para velocidades de hasta 18 Mbps, y en la

Modulación de Amplitud de Cuadratura (QAM, *Quadrature Amplitude Modulation*) para velocidades mayores. Actualmente también funciona en la banda de 2.4 GHz.

(e) HR-DSS

Es una técnica de espectro disperso que utiliza 11 millones de chips/s para alcanzar 11 Mbps en la banda de 2.4 GHz. Soporta velocidades de transmisión de 1, 2, 5.5 y 11 Mbps que se obtienen mediante modulación PSK. Bajo esta técnica, la tasa de datos puede ser adaptada de manera dinámica durante la operación para tener la velocidad más alta posible según las condiciones de carga y ruido.

1.1.2. Capa de enlace de datos

Se encuentra dividida en dos subcapas. La subcapa inferior es conocida como subcapa de Control de Acceso al Medio (MAC, *Media Access* Control) y la superior, como subcapa de Control de Enlace Lógico (LLC, *Logic Link* Control).

(a) Subcapa MAC [1]

La subcapa MAC se encarga de determinar la forma en que se asigna un canal, o dicho en otras palabras, de asignar a quién le toca transmitir.

El estándar 802.11 soporta dos modos de funcionamiento. Uno llamado Función de Coordinación Distribuida (DCF, *Distributed Coordination Function*) que no utiliza ningún tipo de control central, mientras que el otro, llamado Función de Coordinación Puntual (PCF, *Punctual Coordination Function*) emplea una estación base para controlar toda la actividad en su celda. El modo DCF es soportado por todas las implementaciones del estándar 802.11 mientras que el modo PCF es opcional.

En el modo DCF se utiliza el protocolo de Acceso Múltiple con Detección de portadora y Prevención de Colisiones (CSMA/CA, *Carrier Sense*

Multiple Access with Collision Avoidance). Este protocolo utiliza dos métodos: el de detección del canal físico y el de detección del canal virtual.

Cuando se utiliza el método de detección del canal físico, una estación al momento previo de hacer una transmisión, analiza el canal sobre el cual realizará la misma. Si éste está inactivo, inicia la transmisión. Mientras se transmite, no se detecta el canal, por lo que la trama completa es enviada sin importar si es destruida en el receptor debido a interferencia. Si en cambio el canal está ocupado, el emisor espera hasta que pase a estado inactivo para transmitir. En el caso de que se detecte una colisión, las estaciones involucradas en ella, al igual que en Ethernet, esperan un tiempo aleatorio y vuelven a intentar la transmisión más tarde.

El método de detección del canal virtual, se basa en el protocolo Inalámbrico de Acceso Múltiple con Prevención de Colisiones (MACAW, *Multiple Access with Collision Avoidance for Wireless*) y consiste en que el emisor estimula al receptor a enviar una trama corta, de manera que las estaciones cercanas puedan detectar esa transmisión y eviten ellas mismas hacerlo durante la siguiente trama de datos (que es la que contiene la verdadera información). Este método puede apreciarse en la Figura 3.

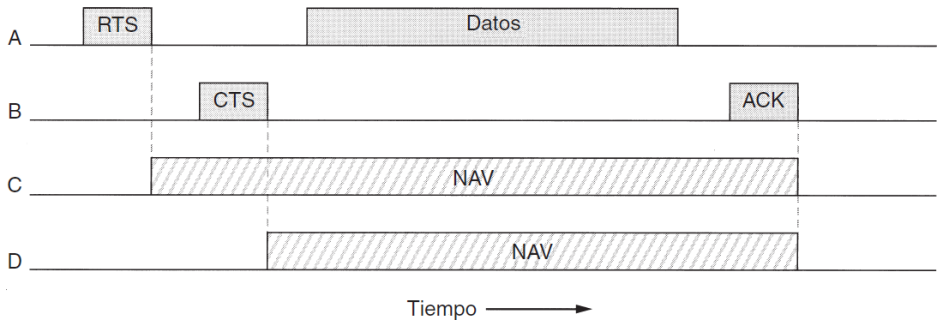


Figura 3. Método de detección de canal virtual. [1]

En este caso, “A” desea enviar datos a “B”; “C” es una estación que está dentro del alcance de “A”; y “D” es una estación dentro del alcance de “B” pero no dentro del de “A”. Cuando “A” decide enviar los datos es que se inicia el protocolo. Comienza entonces “A” enviando una trama de Solicitud

de Envío (RTS, *Request To Send*) a "B" donde le solicita permiso a ésta para enviarle una trama. Cuando "B" recibe esta solicitud decide si otorgar el permiso o no. En caso afirmativo, le envía de regreso a "A" la trama Libre para Envío (CTS, *Clear To Send*). Al recibir la CTS, "A" envía su trama e inicia un temporizador de reconocimiento (ACK, *ACKnowledgment*). Una vez que "B" recibe correctamente la trama de datos, responde con una trama ACK. Si llegase a ocurrir que el temporizador de ACK de la estación "A" termina, se ejecuta de nuevo toda la secuencia. Desde el punto de vista de las estaciones "C" y "D", se sabe que "C" está dentro del alcance de "A", por lo que es posible que reciba también la trama RTS. En este caso, al darse cuenta que otra estación transmitirá, desiste de transmitir cualquier cosa antes que dicho intercambio se complete. La solicitud RTS proporciona cierta información que permite a las estaciones receptoras estimar cuánto tiempo tardará la secuencia, incluyendo el ACK final. Es entonces cuando la estación se coloca en un estado de canal virtual ocupado que es indicado por el Vector de Asignación de Red (NAV, N*etwork Allocation Vector*). Por otro lado, la estación "D" a pesar de no escuchar el RTS, sí escucha la trama CTS por lo que también coloca la señal NAV para sí misma. Las señales NAV no se transmiten, simplemente son recordatorios internos que indican a las máquinas que se colocan en dicho estado, mantenerse en silencio durante un cierto periodo.

Debido a que el estándar 802.11 y sus modificaciones trabajan sobre una banda de frecuencias que no necesita licencia en ninguna parte del mundo, conocida como banda ISM (*Industrial, Scientific and Medical*), sobre la cual trabajan numerosos equipos como hornos microondas, teléfonos inalámbricos, portones de garajes, etc.; las redes inalámbricas son ruidosas e inestables. Como consecuencia de ello, la probabilidad de una trama de llegar a su destino se reduce a medida que la longitud de la misma aumente. Para darle solución a este problema, el estándar 802.11 permite dividir las tramas en fragmentos, cada uno con su propia suma de verificación.

Adicionalmente, cada fragmento se numera de manera individual y su recepción se confirma utilizando un protocolo de parada y espera (el emisor no transmite el fragmento *k+1* hasta no haber recibido confirmación de recepción del fragmento *k*). Una secuencia de fragmentos que es enviada se conoce como ráfaga de fragmentos. Considerando esta fragmentación de las tramas y el método de detección de canal virtual, el mismo ejemplo anterior donde "A" deseaba enviar datos a "B", se vería como lo indica la Figura 4.

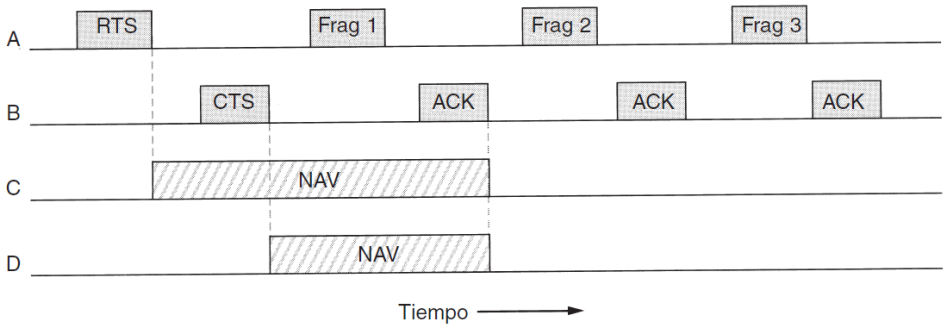


Figura 4. Método de detección de canal virtual con fragmentación. [1]

La fragmentación entonces permite incrementar la velocidad real de transporte y restringir las retransmisiones únicamente a fragmentos erróneos y no a una trama completa.

Es importante conocer que el tamaño de un fragmento no es fijado por el estándar, sino por quien lo implemente, y que el mecanismo NAV, como se observa en la Figura 4, únicamente mantiene en silencio a las otras estaciones del entorno hasta la siguiente confirmación de recepción. Para que una ráfaga de fragmentos completa pueda ser enviada sin problemas es utilizado otro mecanismo. Sin embargo, antes de que sea explicado se describirá el otro modo de funcionamiento del estándar 802.11: PCF.

En el modo PCF, la estación base sondea a las demás, preguntándoles si tienen alguna trama para enviar. De esta manera, la estación base tiene control total sobre el orden de las transmisiones, evitando así las colisiones. El estándar 802.11 define el mecanismo de sondeo pero no define ni su frecuencia, ni el orden en que debe realizarse. El mecanismo básico de

sondeo consiste en que la estación base difunde una trama de beacon (guía o faro) de manera periódica (de 10 a 100 veces por segundo). Dicha trama contiene parámetros del sistema, secuencias de salto, tiempos de permanencia, sincronización de reloj, invitaciones para que otras estaciones se suscriban al servicio de sondeo, etc.

Los modos DCF y PCF están diseñados para coexistir. Tal coexistencia se logra definiendo cuatro intervalos de tiempo con propósitos distintos que se describen a continuación (ver Figura 5):

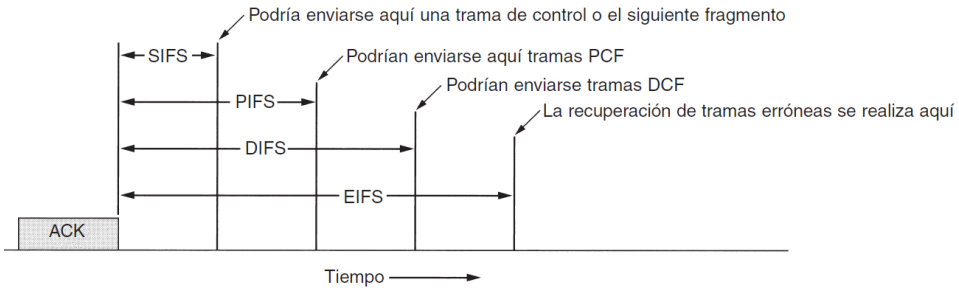


Figura 5. Intervalos de tiempo del estándar 802.11.

- Espaciado Corto Entre Tramas (SIFS, *Short Interframe Space*): el tiempo SIFS se utiliza para permitir que las distintas partes de un diálogo transmitan primero. Después de este intervalo se permite que el receptor envíe un CTS para responder a una RTS o un ACK para responder a la transmisión correcta de un fragmento o trama. También se permite que el emisor de una ráfaga de fragmentos transmita el siguiente fragmento sin tener que enviar una RTS nuevamente. Sólo una estación debe responder después de un intervalo SIFS. Todas las demás estaciones se mantienen escuchando de manera de conocer cuándo en realidad pueden transmitir.
- Espaciado Entre Tramas PCF (PIFS, *PCF Interframe Space*): si se llega al tiempo PIFS sin que se escuche una nueva trama, la estación base puede transmitir una trama de beacon, una trama de datos, o una secuencia de fragmentos, sin que ninguna otra estación interfiera. PIFS podría también definirse como un mecanismo para que la estación base tenga la primera oportunidad de tomar el canal apenas un emisor haya culminado su

transmisión. De esta manera, no tendrá la necesidad de competir con otras estaciones de menor jerarquía.

- Espaciado Entre Tramas DCF (DIFS, *DCF Interframe Space*): de llegar al tiempo DIFS sin que se escuche alguna transmisión de la estación base, cualquier estación puede intentar adquirir el canal para enviar una nueva trama. Las nuevas tramas se envían con las técnicas y mecanismos ya explicados.
- Espaciado Entre Tramas Extendido (EIFS, *Extended Interframe Space*): si la estación receptora recibe una trama errónea o desconocida, a partir del tiempo EIFS es que podrá reportar dicho error. Al darle menor prioridad a este evento, se evita interferir con un diálogo en curso entre otras dos estaciones.

Una vez comprendido cómo funciona el estándar WiFi®, puede ser explicada la estructura de una trama 802.11. Existen tres tipos de tramas 802.11: de datos, de control y de administración.

A continuación en la Figura 6 se muestra la estructura de la trama de datos.

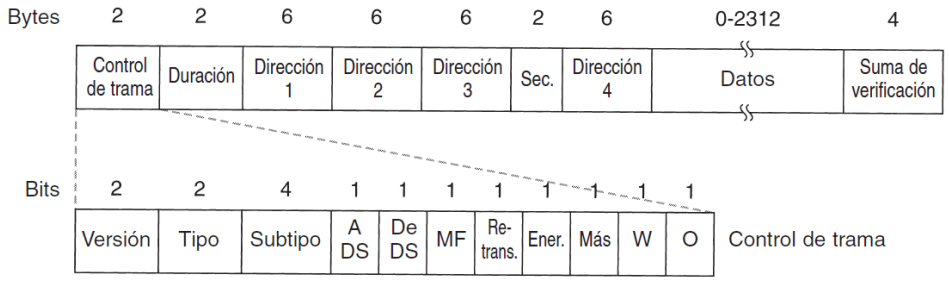


Figura 6. Trama de datos del estándar 802.11. [1]

La trama de datos del estándar 802.11 inicia con el campo de "Control de trama", quien a su vez está conformado por 11 subcampos. El primero, es la "Versión de protocolo" y permite la coexistencia de dos versiones diferentes en una misma celda en un mismo momento. Luego viene el subcampo de "Tipo", que indica si la trama es de datos, de control o de administración, y seguidamente el de "Subtipo", que indica por ejemplo si la trama es de RTS o CTS. A continuación van los bits de "A DS" y "De DS" que indican el

sistema de distribución hacia dónde va o desde dónde viene la trama (por ejemplo Ethernet). El bit denotado como “MF” se utiliza para indicar que siguen más fragmentos mientras que el de “Retransmisión” es utilizado para indicar que la trama es una retransmisión de una que fue enviada anteriormente. El bit de “Administración de energía” es utilizado por una estación base para poner al receptor en estado de hibernación o sacarlo del mismo. El bit “Más” indica al receptor que aún hay tramas pendientes en el emisor. Seguidamente está el bit “W” que indica si el campo de la trama está codificado usando el algoritmo de Privacidad Inalámbrica Equivalente (WEP, *Wired Equivalent Privacy*). Por último, el bit “O” indica al receptor si debe procesar con orden estricto una secuencia de tramas específica.

El segundo campo de una trama de datos es el de “Duración”, que indica cuánto tiempo ocuparán el canal la trama y su confirmación de recepción. Hay cuatro campos de “Direcciones” enumeradas del 1 al 4, las dos primeras son las direcciones de las estaciones de destino y origen, y las últimas dos son las direcciones de las estaciones base de origen y destino (en casos por ejemplo de tráfico entre celdas). Entre los campos de las direcciones 3 y 4 se encuentra el campo de “Secuencia”, que permite enumerar a los fragmentos. Doce de los 16 bits disponibles, identifican a la trama, y los 4 restantes al fragmento. Luego del campo de la dirección 4, viene el campo de “Datos” que contiene la carga útil y tiene un límite de hasta 2312 bytes. Finalmente, la trama posee un campo de “Suma de verificación” para conocer el estado de la recepción.

La trama de administración es similar a la de datos. Sin embargo, tiene sólo 3 campos de direcciones y se debe a que este tipo de tramas son utilizadas únicamente dentro de una celda.

Finalmente, la trama de control también es similar a las tramas ya mencionadas, sin embargo, son más cortas ya que poseen únicamente una o dos direcciones, y no tienen ni campo de datos ni de secuencia. El campo de “Duración” contiene la información necesaria para que las demás estaciones

puedan utilizar el mecanismo NAV, y el campo de “Control de trama”, específicamente el subcampo “Subtipo”, contiene la información clave como los RTS, CTS o ACK.

(b) Subcapa LLC [1]

Tiene como función proporcionar una interfaz con la capa de red de manera que se oculten las diferencias entre los diferentes estándares 802. Bajo este esquema, la capa de red siempre podrá interactuar de igual forma con las capas inferiores.

Permite brindar tres tipos de servicios: servicio no confiable de datagramas, servicio de datagramas sin confirmación de recepción y servicio confiable orientado a la conexión.

Por otro lado, la subcapa LLC agrega un encabezado a las tramas que se tienen en la subcapa MAC. Ese encabezado está formado por tres campos: un punto de acceso de destino, un punto de acceso de origen y un campo de control. Los puntos de acceso indican de cuál proceso proviene la trama y a dónde se va a enviar la misma. Estos campos se utilizan principalmente cuando se necesita una conexión confiable en el nivel de enlace de datos. Para Internet, la confiabilidad de la capa de red (IP) es suficiente, por lo que no se requieren confirmaciones de recepción en el nivel LLC.

La arquitectura del estándar 802.11 se muestra en la Figura 7.

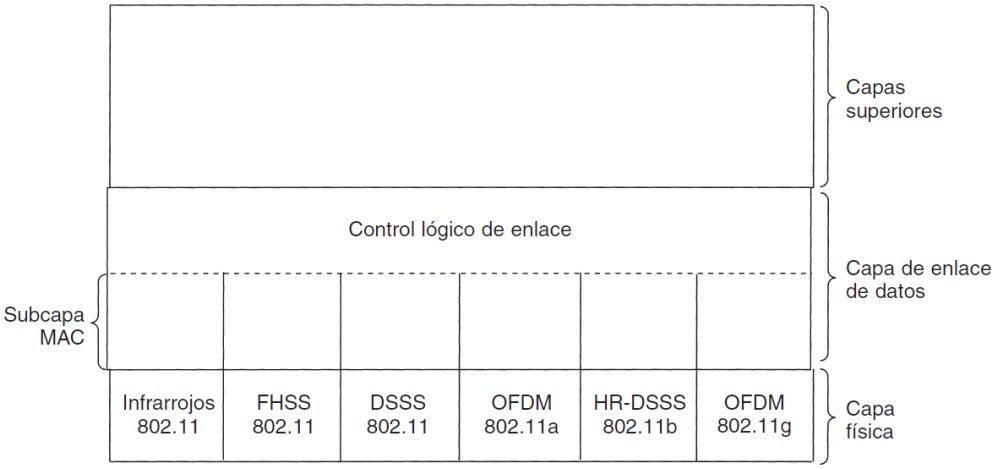


Figura 7. Arquitectura del estándar 802.11. [1]

1.2. Servicios

El estándar 802.11 establece que una red de este tipo debe proporcionar nueve servicios, cinco de distribución y cuatro de estación. Los servicios de distribución están relacionados con la movilidad de una estación conforme entra y sale de las celdas, así como con su conexión y desconexión a las estaciones base de dichas celdas. Por otro lado, los servicios de estación están relacionados con acciones dentro de una celda. [1]

1.2.1. Servicios de distribución

Así como se mencionó antes, estos servicios están relacionados con la movilidad entre celdas y con las conexiones (o desconexiones) de estaciones móviles con estaciones bases. Cinco servicios son de este tipo y se detallan a continuación[1]:

(a) Asociación

Lo utilizan las estaciones móviles para conectarse por sí mismas a las estaciones bases que están en el alcance. Una vez que llegan a dicha zona, anuncian su identidad y sus capacidades como la tasa de datos soportada (necesaria para los servicios PCF de sondeo), y los requerimientos de

administración de energía. La estación base decide si aceptar o rechazar a dicha estación móvil. Si la acepta, esta última debe autenticarse.

(b) Disociación

Es utilizado para romper la relación, bien sea por la estación base o las estaciones móviles. Este servicio puede usarse por una estación móvil antes de salir de la zona de cobertura o antes de apagarse, o por una estación base antes de realizar su mantenimiento.

(c) Reasociación

Es un servicio usado por una estación móvil para cambiar su estación base preferida. Es muy útil cuando las estaciones móviles se mueven de una celda a otra, ya que utilizado de manera correcta, evita la pérdida de datos al realizar el handover (aunque el estándar 802.11 no se encarga de garantizar ello).

(d) Distribución

Sirve para determinar cómo enrutar tramas enviadas a la estación base. Si el destino es local, las tramas pueden enviarse a través del aire; de lo contrario, se envían a través de una red cableada.

(e) Integración

Maneja la traducción de las tramas del formato 802.11 al formato requerido por una red de destino que no sea 802.11.

1.2.2. Servicios de estación

Están relacionados con acciones dentro de una celda y son utilizados únicamente después de que una asociación ha ocurrido. Los cuatro servicios de este tipo son [1]:

(a) Autenticación

Es utilizada como mecanismo de seguridad ante el hecho de que el medio (aire) es compartido por toda aquella estación que se encuentre en el alcance

de un punto de acceso. Una vez que una estación se autentica es que se le permite enviar datos. Al momento que una estación base asocia a una estación móvil, le envía una trama especial de desafío para conocer si dicha estación sabe la contraseña que se le ha asignado a esa red. Para probar que la estación móvil la conoce, ésta codifica la trama de desafío y la regresa a la estación base. Si el resultado es correcto, la estación móvil se vuelve miembro de la celda. Es un servicio que inicialmente no estaba planteado en el estándar 802.11, por lo que cualquier estación móvil podía conectarse a cualquier estación base. Tan pronto los especialistas del protocolo, se dieron cuenta de los problemas que ello ocasionaba (robo de información, suplantación de identidad, etc.), este servicio fue añadido.

(b) Desautenticación

Se usa cuando una estación, que anteriormente fue autenticada, desea abandonar la red.

(c) Privacidad

Para aumentar la seguridad, al igual que lo hizo el servicio de autenticación, fue creado el servicio de privacidad. Éste permite que la información enviada a través de una red WLAN sea codificada y por ende, confidencial.

(d) Entrega de datos

Este servicio es el que proporciona a una estación (móvil o base) una forma de transmitir y recibir datos. Hay que mencionar, que como el estándar 802.11 está basado en Ethernet, y en este último no se garantiza que la transmisión sea 100% confiable, el estándar 802.11 tampoco garantiza la confiabilidad total en sus transmisiones. Debe implementarse entonces con capas superiores que sean capaces de tratar con la detección y la corrección de errores.

2. PROTOCOLO SIP

2.1. Historia

El borrador del Protocolo de Inicio de Sesiones (SIP, *Session Initiaton Protocol*) fue presentado en diciembre de 1996 ante la IETF (*Internet Engineering Task Force*). Era un protocolo basado en HTTP y podía utilizar para el transporte de datos tanto el protocolo UDP como el TCP. Utilizaba al protocolo SDP para describir las sesiones. En 1999, el protocolo alcanza el nivel de estándar propuesto por la IETF, y es publicado como la RFC 2543. [2]

En 1999, se crea un grupo para seguir desarrollándolo. Este grupo en 2002, redefine las especificaciones del protocolo SIP propuestas en la RFC 2543, haciéndola obsoleta. La nueva RFC del protocolo SIP pasa a ser la RFC 3261. [3][4]

Las especificaciones dadas en la RFC 3261 son en su mayoría compatibles con las de la RFC 2543. Sin embargo, la nueva especificación corrige múltiples errores que fueron descubiertos en la anterior y provee información acerca de situaciones que tampoco fueron detalladas en ella. [5]

2.2. RFC 3261

2.2.1. Descripción

La RFC 3261 describe al Protocolo de Inicio de Sesiones, como un protocolo de control (señalización) que opera en la capa de aplicación para crear, modificar y terminar sesiones con uno o más participantes. Una sesión se refiere al intercambio de datos entre un conjunto de participantes, para propósitos de comunicación. Estas sesiones pueden ser llamadas telefónicas por Internet, distribución multimedia (como textos, fotografías, videos, sonidos, etc.), y conferencias multimedia. [5]

Las invitaciones SIP usadas para crear sesiones, llevan consigo descripciones de dichas sesiones que permiten a los participantes coincidir en un conjunto de parámetros que hace posible la compatibilidad entre sus equipos. SIP también provee funciones de registro que permiten a los usuarios actualizar su ubicación.

Funciona independientemente del protocolo de transporte sobre el cual esté y sin depender en el tipo de sesión que se esté estableciendo. Sin embargo, comúnmente SIP se ejecuta sobre los protocolos de transporte UDP y TCP, primordialmente sobre UDP ya que la RFC3261 lo define como el protocolo de transporte por defecto. El uso de un protocolo de transporte sobre otro traerá tanto beneficios como desventajas al sistema, por lo que queda a criterio de quien desee implementar el protocolo SIP elegir el que considere más adecuado. [5][6]

El protocolo SIP puede utilizar cualquier puerto. No obstante, en la mayoría de las implementaciones y por recomendación de la RFC 3261, se utiliza el puerto 5060 cuando los protocolos de transporte son UDP, TCP o SCTP (*Stream Control Transmission Protocol*). En el caso que se deseen conexiones SIP seguras, que se utiliza el protocolo TLS (*Transport Layer Security*), el puerto utilizado es el 5061.

SIP puede invitar participantes a sesiones ya existentes, como ocurre para conferencias multicast. De igual forma pueden añadirse o removerse medios de una sesión existente. SIP soporta servicios de redireccionamiento, permitiendo a los usuarios mantener un único identificador sin importar su localización.

Este protocolo soporta cinco facetas del establecimiento y cierre de comunicaciones multimedia:

1. Localización del usuario.
2. Disponibilidad del usuario.
3. Capacidades del usuario: determinación de parámetros soportados.

4. Configuración de la sesión: establecimiento de los parámetros de la sesión en ambas partes (parte llamada y parte llamante).
5. Administración de la sesión: incluye la transferencia y terminación de sesiones, modificación de parámetros de sesión, y la invocación de servicios.

Se puede decir que SIP no es un sistema de comunicación, sino que más bien es un componente (o módulo) que puede ser utilizado con otros protocolos IETF para construir una arquitectura multimedia completa. Típicamente las arquitecturas de este tipo incluyen protocolos como el Protocolo de Transporte en Tiempo Real (RTP, *Real-time Transport Protocol*) para transportar datos en tiempo real y proveer información de la Calidad de Servicio (QoS, *Quality of Service*); el Protocolo de Streaming en Tiempo Real (RTSP, *Real Time Streaming Protocol*) para controlar la entrega de trenes de datos; el Protocolo de Control de la Puerta de Acceso al Medio (MGCP, *Media Gateway Control Protocol*) para controlar la puerta de acceso a la red PSTN; y el Protocolo de Descripción de Sesiones (SDP, *Session Description Protocol*) para describir sesiones multimedia. Por lo mencionado anteriormente, se entiende que SIP debe ser utilizado en conjunto con otros protocolos para poder proveer servicios completos a los usuarios. Sin embargo, no depende de ninguno de ellos. [5]

2.2.2. Entidades SIP

El protocolo SIP define las siguientes entidades junto con sus roles dentro de cualquier arquitectura que utilice dicho protocolo:

- Agente de Usuario Cliente (UAC, *User Agent Client*): entidad lógica que crea y envía peticiones.
- Agente de Usuario Servidor (UAS, *User Agent Server*): entidad lógica que genera respuestas a peticiones SIP. La respuesta puede aceptar, rechazar o redireccionar la petición.

- Agente de Usuario (UA, *User Agent*): entidad lógica que puede actuar como UAC o como UAS.
- Registrar: es un servidor SIP que acepta peticiones con el método REGISTER.
- Servidor de Ubicación: estos servidores no son entidades SIP, sin embargo son bastante utilizados ya que permiten almacenar y retornar la ubicación de un usuario cuando sea necesario. No es una entidad SIP ya que la comunicación entre un servidor de este tipo y un servidor SIP no es establecida mediante el protocolo SIP.

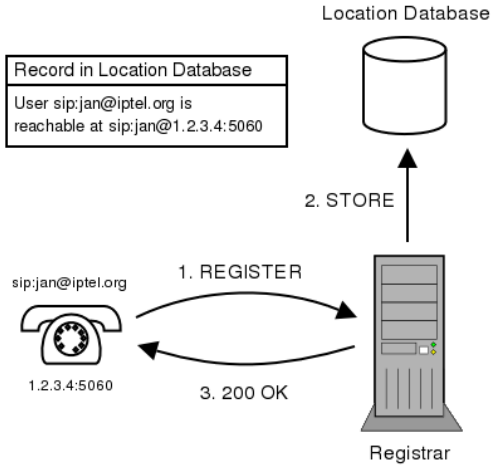


Figura 8. Comunicación entre un cliente SIP, un registrar y un servidor de ubicación.[7]

- Servidor Redireccionador: es un UAS que genera respuestas de redirección (3xx) a las peticiones que recibe. De esta manera, le dice al cliente que envió la petición que intente contactar a unas URIs alternas.

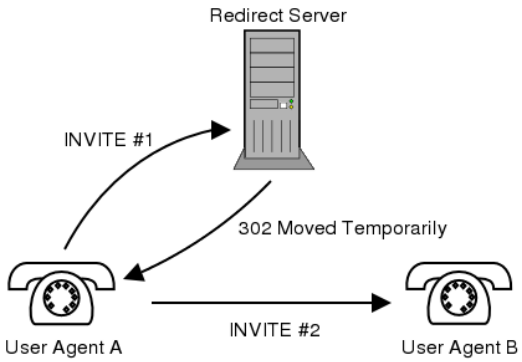


Figura 9. Servidor Redireccionador. [7]

- Servidor Proxy: entidad intermedia que puede actuar como servidor o cliente con el propósito de realizar peticiones en nombre de otros clientes. Su función principal es enrutar las peticiones de manera que la misma sea recibida por el usuario de destino o al menos esté más cerca de él. Un servidor proxy es capaz de interpretar, y de ser necesario reescribir, una parte específica de un mensaje antes de reenviarlo.

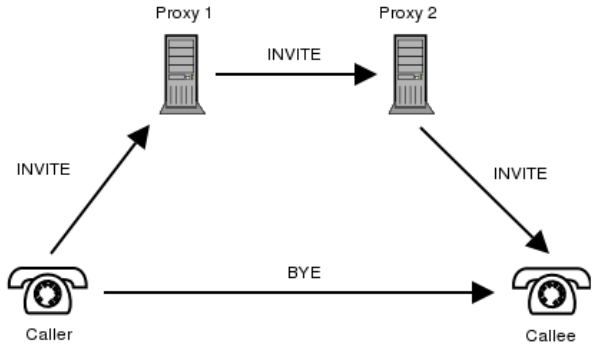


Figura 10. Comunicación trapezoidal mediante proxies. [7]

En la actualidad, los servidores proxies son los que se implementan ya que además de cumplir con sus funciones ya mencionadas, son capaces de cumplir funciones de registrars, servidores de ubicación y servidores redireccionadores. Por ello, en este trabajo cuando se hable de servidor proxy se estará hablando de un servidor con el máximo de sus capacidades. [5]

2.2.3. Mensajes SIP

SIP funciona con IPv4 e IPv6, y está basado en un modelo de transacción similar al de petición/respuesta del protocolo HTTP. Las transacciones son un componente fundamental en SIP y están formadas por las peticiones enviadas desde un cliente hacia un servidor junto con la respectiva respuesta que éste último le da a cada una de ellas. En otras palabras, cada transacción consiste en una petición que invoca un método o una función particular en el servidor. Del servidor, el cliente obtiene al menos una respuesta. Esto puede verse en la Figura 11.

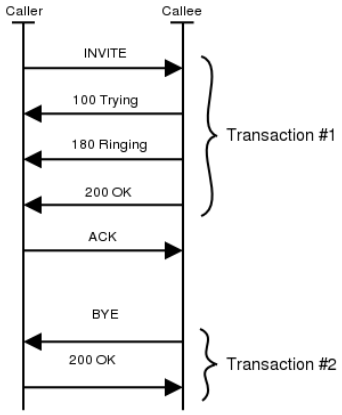


Figura 11. Transacción SIP. [7]

Cualquiera de las partes envía un mensaje (basado en la codificación UTF-8) que consiste en una línea de inicio, unas cabeceras (*headers*) con información de los parámetros de la sesión, una línea vacía indicando el final de las cabeceras y un cuerpo (message-*body*) con información propia de la sesión o tal vez su descripción (por ejemplo con el protocolo SDP). El cuerpo del mensaje es opcional, sin embargo aunque éste no exista, la línea vacía debe incluirse en los mensajes SIP. La estructura de un mensaje SIP puede verse en la Figura 12.

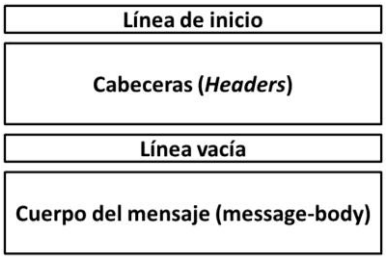


Figura 12. Estructura de un mensaje SIP.

A pesar que los mensajes SIP tienen una estructura general, su contenido depende si el mensaje es de petición o de respuesta. [5]

(a) Línea de Inicio

(a.1) Línea de Inicio en Peticiones

Las peticiones se caracterizan por tener una línea llamada “Request-Line” como línea de inicio. Una Request-Line, como se observa en la Figura 13,

contiene el nombre del método, una Request-URI, y la versión del protocolo. Todas separadas por un espacio simple.

Figura 13. Request-Line de un mensaje de petición. [5]

La RFC 3261 define 6 métodos: INVITE, ACK, BYE, OPTIONS, CANCEL y REGISTER.

- INVITE: usado para establecer una sesión. Las cabeceras de la segunda y las siguientes líneas, describen las capacidades, los tipos de medios soportados y los formatos del invocador.
- ACK: la conexión se realiza utilizando un acuerdo de 3 vías, de modo que el invocador puede responder con un mensaje de este tipo para terminar el protocolo y confirmar la recepción de un mensaje OK (próximamente descrito).
- BYE: utilizado para la terminación de una sesión. Una vez que el otro lado confirma su recepción, se termina la sesión.
- OPTIONS: se usa para consultar las capacidades de una máquina. Es utilizado por lo general, antes de iniciar una sesión por lo menos para conocer si una máquina tiene capacidad para transmitir voz sobre IP, o si soporta el tipo de sesión que se pretende establecer.
- CANCEL: es utilizado para cancelar una solicitud generada.
- REGISTER: este mensaje se envía a un servidor registrar con base de datos de ubicación para indicar que un usuario se está registrando, de manera que se tenga un registro de quién está conectado y desde dónde lo hace.

En la figura anexa se resume la descripción de cada método mencionado anteriormente.

Método	Descripción
INVITE	Solicita el inicio de una sesión
ACK	Confirma que se ha iniciado una sesión
BYE	Solicita la terminación de una sesión
OPTIONS	Consulta a un *host* sobre sus capacidades
CANCEL	Cancela una solicitud pendiente
REGISTER	Informa a un servidor de redireccionamiento sobre la ubicación actual del usuario

Figura 14. Métodos definidos en la RFC 3261. [1]

Una URI es una cadena de caracteres que identifica a cada usuario. Su nombre completo es Identificador Uniforme de Recurso (URI, *Uniform Resource Identifier*) y tiene la siguiente forma: "sip:usuario1@midominio.com" para un usuario de nombre "usuario1" registrado al dominio "midominio.com". También pueden utilizarse direcciones IPv4, IPv6, o incluso números telefónicos. Por ejemplo una URI válida utilizando una dirección IPv4 sería la siguiente: "sip:usuario1@192.168.1.106". Una URI del protocolo SIP es conocida como SIP URI o SIPS URI (en el caso que use una conexión segura). La Request-URI de la línea de inicio de una petición, se refiere entonces al identificador del usuario o del servidor al cual la petición está siendo direccionada.

La Versión-SIP indica qué versión del protocolo en SIP está en uso. La versión actual es la 2.0. Entonces se coloca en la Request-Line la cadena de caracteres "SIP/2.0".

(a.2) Línea de Inicio en Respuestas

Por otro lado, los mensajes de respuestas, se diferencian de los de peticiones ya que tienen una línea llamada "Status-Line" como línea de inicio. Esta línea, como se observa en la Figura 15, consiste en la versión del protocolo, un código numérico denominado Código de Estado (*Status*-Code) y una frase denotada como Reason-Phrase. Todas separadas por un espacio simple.

Versión-SIP Código-de-Estado Reason-Phrase

Figura 15. Status-Line de un mensaje de respuesta. [5]

La versión SIP al igual que para las peticiones, indica qué versión del protocolo está en uso. Se coloca entonces la cadena de caracteres "SIP/2.0".

Los códigos de estado (*Status-Code*) son números enteros de tres dígitos utilizados para responder ante solicitudes realizadas mediante los métodos anteriormente expuestos. Se dividen en 6 categorías según su primer dígito, ya que éste define la clase de respuesta:

- Respuestas provisionales (1xx): son conocidas como respuestas informativas, e indican que el servidor contactado está realizando alguna acción particular y por lo tanto no tiene aún una respuesta definitiva. Un servidor envía este tipo de respuestas si espera tardar más de 200 ms en obtener una respuesta final. Estas respuestas 1xx pueden o no tener un cuerpo para describir la sesión.
- Respuestas de éxito (2xx): indica que la solicitud fue recibida de manera exitosa.
- Respuestas de redirección (3xx): este tipo de respuestas dan al invocador información acerca de la nueva ubicación del usuario invocado, o acerca de servicios alternativos que están disponibles para satisfacer la petición.
- Respuestas de fallas en el cliente (4xx): indican que hay fallas como datos erróneos del usuario invocador o del invocado, usuarios no autorizados, el invocador no está autenticado, el invocado no puede ser alcanzado, métodos no aceptados, etc.
- Respuesta de fallas en el servidor (5xx): son enviadas cuando el servidor se da cuenta que ha cometido un error en el procesamiento de una solicitud.
- Respuesta de fallas globales (6xx): indican que un servidor tiene información sobre el usuario al cual la URI de la solicitud hace referencia, aunque no puede obtener el estado de él por lo que no puede procesar la solicitud de la manera adecuada. También sirven para indicar que el usuario invocado ha rechazado su ingreso a la sesión. [1]

En la tabla anexa pueden verse las respuestas más comunes especificadas en la RFC 3261 junto con su descripción.

Tabla 1. Respuestas más comunes definidas en la RFC 3261. [5]

Conjunto	**Código específico**	**Palabra Asociada**	**Descripción**
1xx	100	Trying	La solicitud ha sido recibida por el servidor más cercano y está intentando darle respuesta
	180	Ringing	El UA de destino que recibió una solicitud con método INVITE está alertando a su usuario acerca de la llamada
2xx	200	OK	La solicitud fue recibida de manera adecuada. La información que compone el resto del mensaje OK depende del método de la solicitud
3xx	301	Moved Permanently	El usuario que se está intentando contactar no puede ser conseguido en la URI indicada ya que cambió definitivamente de dirección. Arroja la nueva ubicación para que sea contactado
	305	Use Proxy	Debe utilizarse un servidor proxy para contactar la URI deseada
4xx	400	Bad Request	La solicitud tiene algún error en su sintaxis y no puede ser entendida por el servidor
	401	Unauthorized	El usuario no está autorizado a conectarse al servidor
	404	Not Found	El usuario no existe en la base de datos del servidor
	407	Proxy Authentication Required	El usuario debe autenticarse ante el servidor proxy para poder ser aceptado por éste
5xx	500	Server Internal Error	El servidor encontró que está en una condición no esperada por lo que evita tener que procesar la solicitud recibida. El cliente puede reintentar la solicitud luego de un tiempo

6xx	600	Busy Everywhere	El usuario invocado ha sido contactado pero se encuentra ocupado y por lo tanto no desea aceptar la solicitud
	603	Decline	El usuario invocado ha rechazado unirse a la sesión

Por último, la Reason-Phrase es una palabra o frase que se usa para mostrar una corta descripción del código de estado de la respuesta. Mientras que un código de estado es creado con la finalidad de ser usado de forma automática por una máquina, la Reason-Phrase es creada con la finalidad de ser utilizada por un usuario, por ejemplo para darse cuenta dónde está presentándose una falla algún sistema SIP que haya implementado.

La Figura 16 representa un ejemplo de cómo se utiliza el protocolo SIP para establecer una sesión. En ella puede observarse el flujo de mensajes SIP entre un usuario de nombre "user" registrado en el servidor proxy perteneciente al dominio "ibm.com" y un usuario también de nombre "user" pero que pertenece al servidor proxy del dominio "example.com". Dicho flujo de mensajes conforma un diálogo. Un diálogo es una relación SIP "peer-to-peer" entre dos User Agents (bien sea entre un cliente y un servidor, entre dos clientes, o entre dos servidores) que se mantiene por un tiempo. Un diálogo facilita la secuencia de mensajes y permite un adecuado enruteo de peticiones entre User Agents. El INVITE es el único método de esta especificación que permite establecer un diálogo. En la Figura 17, puede observarse qué es un diálogo y su diferencia con una transacción.

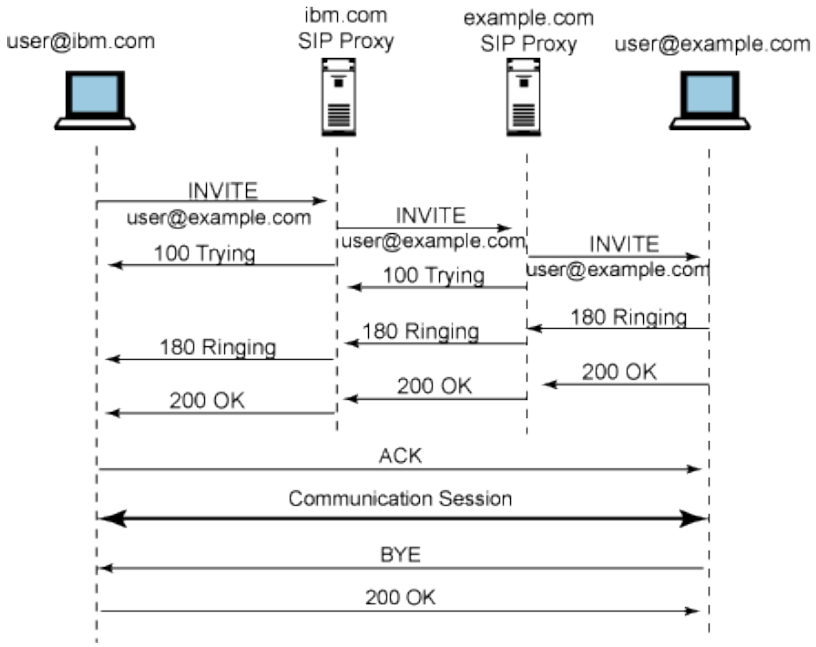


Figura 16. Comunicación mediante mensajes SIP. [8]

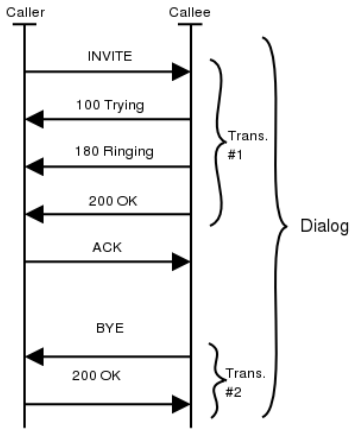


Figura 17. Representación de un diálogo y su diferencia con una transacción. [7]

(b) Cabeceras

Ya fue descrita la primera línea de los mensajes SIP, por lo que ahora se describirán las líneas de cabeceras. Toda cabecera incluye el nombre de la misma seguido por dos puntos, y posteriormente tras un espacio simple se coloca su valor. Si una cabecera tiene más de un valor, éstos se separan por comas. Hay ciertas cabeceras cuyos valores no deben ser combinados en una misma línea. Ellas (y también cualquier otra cabecera con múltiples valores)

pueden ser escritas como múltiples cabeceras de igual nombre y cada una con su valor respectivo. Es importante destacar, que a pesar que la RFC3261 especifica usar espaciado simple en algunos casos, no poner espacios o poner espacios múltiples, no afecta el funcionamiento del protocolo. A continuación puede verse cómo sería una cabecera con un valor simple o con valores múltiples.

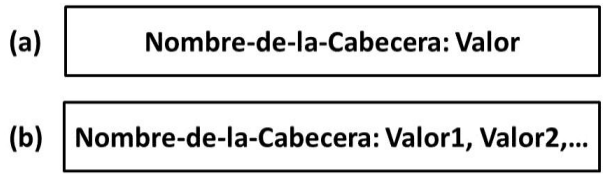


Figura 18. (a) Cabecera de un único valor. (b) Cabecera con múltiples valores. [5]

Las cabeceras pueden extenderse en múltiples líneas. Sin embargo, cada línea extra debe ser antecedida por al menos un espacio simple o una sangría. El orden de las cabeceras con nombres distintos no es significante, pero sí el de aquellas con un mismo nombre.

El formato de un valor de cabecera se define según el nombre de la cabecera. Siempre será una combinación de espacios, símbolos, separadores y caracteres de texto. Todos ellos bajo el formato de codificación UTF-8. Algunas cabeceras añaden a la forma general seguida por un punto y coma, un par de parámetros formado conformado por el nombre del mismo y su valor. En este caso la cabecera quedaría como lo muestra la Figura 19.

Figura 19. Otra forma de escribir una cabecera SIP. [5]

Pueden añadirse los pares de parámetros que se necesiten, siempre y cuando todos sean de distinto nombre. Vale la pena destacar que las cabeceras en SIP son case-insensitive, es decir, no se diferencia entre mayúsculas y minúsculas.

Al igual que como ocurre con la línea de inicio, algunas cabeceras sólo tienen sentido en peticiones o respuestas. En estos casos se llaman cabeceras

de peticiones o cabeceras de respuestas, respectivamente. Si una cabecera aparece en un mensaje que no coincide con su categoría, es sencillamente ignorada.

A continuación una breve descripción de las cabeceras SIP más utilizadas:

- To: especifica el recipiente lógico de la petición, o el recurso que es el objetivo de la petición. Puede que él sea o no el último receptor de dicha solicitud. La cabecera To puede contener una SIP URI o una SIPS URI (con seguridad), pero también puede contener otros tipos de URI (por ejemplo la URL de un teléfono). La cabecera To permite colocar un nombre que se mostrará a los demás usuarios (*display name*). El valor de la cabecera To debe ser siempre igual al del Request-URI a excepción de cuando se utilice el método REGISTER. Una cabecera To tiene la siguiente forma:

  To: Carol <sip:carol@chicago.com>

- From: indica la identidad lógica de quien inicia la petición. Como el To, contiene un URI y un display name opcional. Esta cabecera debe contener un parámetro "tag", que es propio del UAC que genera la petición. La forma de la cabecera From tiene entonces la siguiente forma:

  From: "Bob" <sips:bob@biloxi.com>;tag=a48s

- Call-ID: actúa como un único identificador para asociar una serie de mensajes. Debe ser el mismo para todas las peticiones y respuestas enviadas dentro de un diálogo. En cada petición nueva creada por un UAC fuera de cualquier diálogo, la cabecera Call-ID debe ser seleccionada por el UAC como un único identificador en tiempo y espacio. Todos los UAs deben asegurar que los Call-ID que produzcan no sean generados por otro UA, por lo que generan de manera aleatoria un identificador con la forma "identificador@host", donde host es la dirección lógica de quien genera por primera vez la cabecera. El campo Call-ID es case-sensitive, es decir que distingue entre minúsculas y

mayúsculas. Se compara byte a byte para determinar la igualdad de dos cabeceras Call-ID. La cabecera Call-ID tiene la siguiente forma:

Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com

- CSeq: sirve para identificar y ordenar transacciones. Consiste en un número secuencial y un método. El método debe ser igual al de la petición. Para peticiones que no sean REGISTER y que sean fuera de diálogo, el número es arbitrario. Dicho número debe ser expresable como un entero de 32-bits sin signo y debe ser menor que 2^{31}. No hay más restricciones con dicho número por lo que puede ser generado por cualquier mecanismo que se desee. Un ejemplo de esta cabera es:

  CSeq: 7411 INVITE

  Cuando se reintenta una petición, la misma constituye una nueva transacción y puede tener los mismos valores de Call-ID, To y From de la petición anterior. Sin embargo, el valor de la cabecera CSeq debe tener un número inmediatamente superior al de la petición anterior.
- Max-Forwards: sirve para limitar el número de saltos que una petición puede hacer en la vía hacia su destino. Consiste en un entero que es reducido en uno por cada salto. Si el valor alcanza 0 antes de que la petición alcance su destino se genera una respuesta de error con el código 483 y la descripción "Too Many Hops". Por recomendación del protocolo, esta cabecera debe tener un valor de 70. Este número fue elegido de manera que se garantice que una petición no se pierda en una red SIP sin haber hecho redundancias (*loops*), pero que sí lo haga cuando ocurran. Si se conoce la topología de la red, este número puede ser reducido o aumentado. El máximo número que puede tener es 255. Esta cabecera será por lo general de la siguiente manera:

  Max-Forwards: 70
- Via: indica qué transporte es usado para la transacción y a donde es enviado el mensaje. Debe contener el nombre y la versión del protocolo, que como ha sido mencionado, debe ser SIP y 2.0, respectivamente.

Seguidamente contiene un valor que indica el transporte que se está utilizando y la dirección lógica a la cual se está enviando el mensaje. Los tipos de transporte permitidos son UDP, TCP, TLS y SCTP. La cabecera Via debe contener un parámetro “branch” que es una cadena de caracteres que agrega quien envía la petición (sea cliente o servidor) utilizada para identificar una transacción. El parámetro branch debe ser único en espacio y tiempo para cada petición enviada por el UA. Las excepciones a esta regla son las peticiones con los métodos CANCEL y ACK para respuestas no-2xx. Una petición CANCEL tiene el mismo parámetro branch que la petición que cancela, mientras que una petición ACK para una respuesta no-2xx tendrá el mismo branch que el INVITE al que está haciendo reconocimiento (ACKnowledgment). El branch ID insertado por un elemento SIP debe siempre empezar con los caracteres z9hG4bK. Son 7 caracteres que permiten asegurar que la antigua especificación no tome un valor igual a ese. Esto es una característica que permite a los servidores que reciben las peticiones identificar que se está usando la especificación 3261 y no la 2543. Una cabecera Via toma la siguiente forma:

Via: SIP/2.0/UDP 192.168.1.103:49152;branch=z9hG4bKc0a801

- Contact: esta cabecera debe estar presente y contener únicamente un SIP o SIPS URI en cualquier petición que pueda resultar en el establecimiento de un diálogo (para esta especificación ocurre únicamente en la petición INVITE). Para estas peticiones, el objetivo de la cabecera Contact es global, es decir, esta cabecera contiene el URI al cual el UA desea recibir las peticiones, y el mismo debe ser válido incluso si es usado en peticiones subsecuentes fuera de cualquier diálogo. Puede tener un display name opcional. Un ejemplo sería:

Contact: “Luis” <sip:10@192.168.1.105:14634>

- Allow: esta cabecera sirve para informar qué métodos soporta el UA que genera el mensaje. Tiene la forma:

  Allow: INVITE, ACK, OPTIONS, CANCEL, BYE

- Content-Length: esta cabecera indica el tamaño del cuerpo del mensaje. El tamaño es mostrado como un número decimal y representa cuántos octetos fueron enviados al receptor. Puede tener cualquier valor mayor o igual a cero. Cero en el caso que el mensaje no tenga cuerpo. Un ejemplo:

  Content-Length: 349

- Content-Type: indica qué tipo de datos fueron enviados dentro del cuerpo del mensaje SIP. Si el cuerpo está vacío no hace falta esta cabecera, de lo contrario sí debe estar presente. Un ejemplo de una cabecera de este tipo, perteneciente a un mensaje que contiene la descripción de una sesión (protocolo SDP) sería de la siguiente manera:

  Content-Type: application/sdp

  Si por ejemplo el cuerpo del mensaje contiene texto simple la cabecera Content-Type sería:

  Content-Type: text/plain

- Expires: indica el tiempo después del cual, el mensaje enviado expirará. Es un número decimal que expresa un tiempo en segundos. Puede tener valores desde 0 hasta 2^{32}-1. Ejemplo:

  Expires: 3600

- Proxy-Authenticate: es igual a la cabecera utilizada en HTTP y contiene un reto de autenticación de manera prevenir conexiones de usuarios no autorizados. Debe incluirse como parte de una respuesta con código 407 (Proxy Authentication Required). El método para generar una respuesta a un reto de autenticación es llamado Digest. Para más información debe leerse la RFC 2617 (HTTP Authentication: Basic and Digest Access Authentication).[9][10]

  Ejemplo:

  Proxy-Authenticate: Digest realm="atlanta.com",
  nonce="f84f1cec41e6cbe5aea9c8e88d359",algorithm=MD5

A partir de los parámetros realm y nonce colocados por el servidor en la respuesta 407, y según el algoritmo indicado en el parámetro algorithm, el UAC deberá generar una cabecera Proxy-Authorization con los datos necesarios para ser aceptado.

- Proxy-Authorization: es la cabecera que coloca un UAC para identificarse ante el servidor que le ha enviado un response 407. Tiene la forma:

  Proxy-Authorization: Digest username="Alice", realm="atlanta.com", nonce="c60f3082ee1212b402a21831ae",response="245f23415f11432b3434341c022"

  Los parámetros nonce y realm son iguales a los enviados por el servidor con la respuesta 407. A partir de ellos, y el algoritmo también indicado en dicha respuesta, el UAC genera un código para contestar al reto (parámetro response). Adicionalmente, envía su nombre de usuario (parámetro username).

  El uso de las dos cabeceras previas puede entenderse con la siguiente imagen donde un usuario desea registrarse ante un registrar, que le envía una respuesta 407. El UA deberá responder de nuevo con un mensaje REGISTER pero con las credenciales correspondientes.

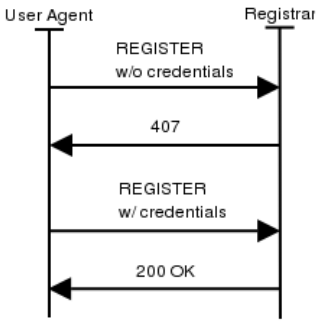


Figura 20. Uso de las cabeceras Proxy-Authenticate y Proxy-Authorization. [7]

- Record-Route: esta cabecera es insertada por un servidor proxy y se usa para forzar a futuras peticiones dentro de un diálogo a que pasen a través de él. Ejemplo:

  Record-Route: <sip:server10.biloxi.com;lr>

- User-Agent: contiene información acerca del UAC que inició la petición. Un ejemplo sería:

  User-Agent: X-Lite 4 release 4.1 stamp 63214

(b.1) Cabeceras en Peticiones

Una petición SIP debe contener al menos las siguientes cabeceras: To, From, CSeq, Call-ID, Max-Forwards y Via. Estas 6 cabeceras proveen en conjunto la información crítica para el enruteo de servicios (dirección de mensajes, enruteo de respuestas, propagación límite de un mensaje, orden de los mensajes y un identificador de transacciones.

(b.2) Cabeceras en Respuestas

Las cabeceras From, Call-ID, CSeq deben ser iguales a las de las peticiones. También debe serlo la cabecera Via. Ésta debe también estar ordenada de igual forma que en la petición. La cabecera To de una respuesta debe ser igual a la de la petición, aunque debe añadir un parámetro "tag" si la misma no lo contiene para identificar al UAS que está respondiendo.

Las cabeceras de una respuesta dependen en qué tipo de respuesta deba generarse.

(c) Cuerpo del mensaje

El cuerpo de un mensaje SIP es utilizado para enviar información. Por ejemplo, sirve para enviar texto, descripciones de sesiones (usando el protocolo SDP), etc.

(d) Ejemplos de mensajes SIP

- INVITE

  INVITE sip:bob@biloxi.com SIP/2.0
  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
  Max-Forwards: 70
  To: Bob <sip:bob@biloxi.com>
  From: Alice <sip:alice@atlanta.com>;tag=1928301774
  Call-ID: a84b4c76e66710@pc33.atlanta.com
  CSeq: 314159 INVITE
  Contact: <sip:alice@pc33.atlanta.com>

Content-Type: application/sdp

Content-Length: 142

(El cuerpo SDP no se muestra)

- 200 OK

SIP/2.0 200 OK

Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bKnashds8

Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1

Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds

To: Bob <sip:bob@biloxi.com>;tag=a6c85cf

From: Alice <sip:alice@atlanta.com>;tag=1928301774

Call-ID: a84b4c76e66710@pc33.atlanta.com

CSeq: 314159 INVITE

Contact: <sip:bob@192.0.2.4>

Content-Type: application/sdp

Content-Length: 131

(El cuerpo SDP no se muestra)

- REGISTER

REGISTER sip:registrar.biloxi.com SIP/2.0

Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7

Max-Forwards: 70

To: Bob <sip:bob@biloxi.com>

From: Bob <sip:bob@biloxi.com>;tag=456248

Call-ID: 843817637684230@998sdasdh09

CSeq: 1826 REGISTER

Contact: <sip:bob@192.0.2.4>

Expires: 7200

Content-Length: 0

2.2.4. Arquitectura multicapa

Existe una manera interesante de estudiar al protocolo SIP, ya que a pesar que es un protocolo que funciona a nivel de aplicación con respecto al modelo OSI, tiene una arquitectura interna basada en etapas (o capas), de manera que es posible describirlo internamente mediante un modelo multicapa.

La capa más baja de SIP corresponde a su sintaxis y su codificación. Esta última usa una gramática llamada "Augmented Backus-Naur Form". Por encima de ella, está la capa de transporte encargada de manejar cómo un cliente (o un servidor) envía peticiones y recibe respuestas (o recibe peticiones y envía respuestas, respectivamente). Sobre ambas capas se encuentra la capa de transacciones. En esta tercera capa se manejan retransmisiones, se asocian respuestas a peticiones, y se manejan temporizadores. Cualquier tarea que un agente de usuario cliente (UAC, *User Agent Client*) lleva a cabo, se realiza usando una serie de transacciones. Finalmente, sobre la capa de transacciones está la capa de Usuario de Transacción (TU, *Transaction User*). Esta última capa permite enviar peticiones a la dirección IP y al puerto de destino. También permite crear transacciones entre un cliente y un servidor. A continuación se muestra gráficamente el orden de las capas mencionadas.

Figura 21. Arquitectura multicapa del protocolo SIP.

La siguiente tabla resume las características más importantes del protocolo SIP:

Tabla 2. Características más importantes de SIP. [1]

Elemento	Descripción
Diseñado por	IETF
Compatibilidad con PSTN	Ampliamente
Compatibilidad con Internet	Sí
Arquitectura	Modular
Integridad de la sesión	SIP sólo maneja el establecimiento
Negociación de parámetros	Sí
Protocolo de Transporte	Generalmente UDP o TCP
Formato de mensajes	UTF-8
Llamadas de múltiples partes	Sí
Direccionamiento	URI
Conferencias multimedia	No
Terminación de llamadas	Explícita o mediante un temporizador
Mensajes instantáneos	Sí
Encriptación	Sí
Tamaño del estándar	269 páginas

2.3. RFC 3428

Gracias a que el protocolo SIP es modular, permite la extensión del mismo, de manera que añadiéndole funciones es posible ampliar su alcance. Una extensión del protocolo SIP para agregar mensajería instantánea a SIP, está descrita en la RFC 3428.

La RFC 3428, creada en diciembre de 2002, añade a SIP el método MESSAGE para permitir la transferencia de mensajes instantáneos. Como es una extensión, hereda todas las características de la RFC 3261.

El método MESSAGE no inicia por sí mismo un diálogo y puede ser utilizado separado de cualquiera de los métodos de la RFC 3261. Se utiliza en

peticiones. La respuesta que se espera a una petición MESSAGE es una 200 OK. Esta respuesta debe ser generada por el UA que reciba el mensaje, indicando que lo ha aceptado. La aceptación del mensaje no implica la lectura del mismo.

Todo UAC que soporte al método MESSAGE debe ser capaz de enviar mensajes de texto simples (mensajes de tipo text/plan). La cabecera Contact no debe ser añadida por ningún UA a una petición MESSAGE. [11]

En la próxima figura puede verse el flujo de mensajes SIP con peticiones de tipo MESSAGE:

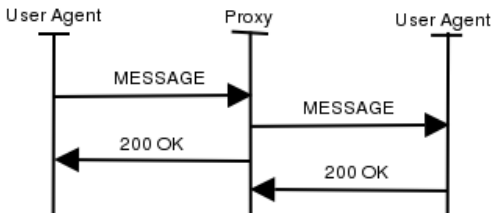


Figura 22. Mensajería instantánea con protocolo SIP. [7]

Un mensaje MESSAGE tiene la siguiente forma:

```
MESSAGE sip:user2@domain.com SIP/2.0
Via: SIP/2.0/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse
Max-Forwards: 70
From: sip:user1@domain.com;tag=49583
To: sip:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18
Watson, come here.
```

Su respuesta tendrá el siguiente formato:

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse;
received=1.2.3.4
From: sip:user1@domain.com;tag=49394
To: sip:user2@domain.com;tag=ab8asdasd9
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Length: 0
```

La RFC 3428 no toma en cuenta aspectos adicionales de seguridad. Por lo tanto, para realizar implementaciones seguras deben añadirse módulos de otros protocolos que sí lo hagan. [11]

2.4. Software SIP

2.4.1. Clientes SIP

A continuación se mencionan algunos famosos clientes SIP y sus características más importantes:

- X-Lite®: es un cliente SIP gratuito que combina llamadas de voz, video llamadas y mensajería instantánea. Tiene la opción de ser pago y añade funciones de seguridad, videoconferencia, soporte, etc. Su código es privado ya que es un software propietario. Opera en Windows. [12]
- SJPhone®: cliente SIP que puede ser descargado gratuitamente para realizar únicamente llamadas de voz. Es un software propietario y por lo tanto su código no puede ser visto. Opera en Windows. [13]
- Peers: es un cliente SIP completamente gratuito, basado por completo en lenguaje Java. Únicamente permite realizar llamadas. Es un software de

código abierto y por lo tanto su código puede ser descargado. Opera en Windows. [14]

- Jitsi: conocido previamente como SIP Communicator, es un cliente SIP OpenSource para sesiones de audio, video y mensajería instantánea. Opera en Windows y está basado completamente en Java. [15]

2.4.2. Servidores SIP

A continuación se mencionan algunos famosos servidores SIP y sus características más importantes:

- 3CX®: tiene dos versiones, una gratuita y una paga. La gratuita no tiene límite de usuarios registrados pero sí de conexiones simultáneas. Es un servidor propietario por lo que no se tiene acceso a su código. Adicionalmente, está diseñado para operar en Windows. Viene con ciertos parámetros preconfigurados al instalarlo.[16]
- OpenSIPS: anteriormente conocido como OpenSER, es un servidor gratuito de código abierto que se ejecuta en sistemas operativos abiertos (GNU, Linux, etc.). Permite programar todos los parámetros para que tenga exactamente el funcionamiento que se desee. Como todo software OpenSource, puede ser utilizado con fines comerciales o no comerciales.[17]
- Asterisk: es una aplicación para controlar y gestionar comunicaciones de cualquier tipo, ya sean analógicas, digitales o de VoIP. Es una aplicación OpenSource que puede ser utilizada con fines comerciales o no comerciales. Puede ser programada para tener los parámetros que se deseen, se ejecuta en sistemas libres como Linux y trabaja con estándares abiertos como SIP, H.323, etc. [18]

2.5. Importancia del protocolo SIP

Actualmente, se compara el futuro de SIP en las telecomunicaciones con lo que fue y es HTTP para la Web. Es por ello que las empresas de celulares han decidido normalizar sobre SIP todas las aplicaciones futuras.

Las características más importantes que el protocolo SIP ofrece son [19]:

- Estabilidad: ya que es utilizado desde hace años.
- Velocidad: el hecho que trabaje con el protocolo UDP, lo hace rápido para entregar los mensajes. A pesar de trabajar con UDP es extremadamente eficiente.
- Flexibilidad: está basado en texto, lo que lo hace fácilmente extensible.
- Normalización: el lograr que las empresas de telecomunicaciones emigren hacia él le da superioridad con respecto a cualquier otro estándar.

3. IP MULTICAST

3.1. Servicio de IP Multicast

Multicast o multidifusión, es la acción de enviar datos a un grupo de estaciones. Sin embargo, cuando se utilizan mecanismos de la capa de transporte para realizar dicha función, tal servicio es llamado IP multicast (o multidifusión IP).

En IPv4, el soporte de la multidifusión IP es opcional. A pesar de ello, la mayoría de las estaciones y enrutadores son capaces de soportar dicho servicio. En términos de IP multicast, las estaciones son aquellos miembros de un grupo. Los enrutadores tienen como funciones principales: enrutar los datos de la multidifusión hacia todos los miembros de un grupo o hacia otros enrutadores que estén más cerca de los receptores.

Una dirección multicast es una dirección IP (IPv4) en el rango que va desde 224.0.0.0 hasta 239.255.255.255. Cada dirección multicast es específica de un grupo multicast. Una estación se une a un grupo multicast iniciando un proceso de sondeo para escuchar aquellos paquetes que vayan dirigidos a la dirección IP asociada a dicho grupo. Un grupo IP multicast es abierto y dinámico, es decir, cualquier estación puede ingresar y dejarlo cuando lo desee. Así mismo, no tienen un límite en cuanto a número de miembros. Por otro lado, las estaciones pueden asociarse a múltiples grupos en un mismo momento.

Un paquete de multidifusión es un datagrama que tiene en su dirección de destino una dirección IP multicast. Es entregado a toda estación que sea miembro del grupo asociado a dicha dirección. El transporte en aplicaciones multicast se realiza únicamente con el protocolo UDP. Por lo tanto, si se requieren implementaciones con transferencia de datos confiables, la capa de aplicación debe proveer dicha confiabilidad. [20]

3.2. Direcciones IP Multicast

Como fue mencionado, el rango de las direcciones de multidifusión IP va desde 224.0.0.0 hasta 239.255.255.255. Este rango corresponde con las direcciones IP de clase-D según el esquema de clases IP.

Las direcciones clase-D empiezan siempre con los bits 1110. Los 28 bits restantes sirven para identificar al grupo multicast. [20]

4. MENSAJERÍA INALÁMBRICA

La mensajería instantánea, desde su nacimiento, ha adquirido un importante interés tanto en el mundo de la telefonía como en el de la computación. Podemos mencionar algunas de las arquitecturas de mensajería para redes inalámbricas basadas en telefonía celular más populares en la actualidad como lo son SMS, MMS, Email, BlackBerry® Messenger, etc. A continuación se ofrece una breve descripción de las mismas.

4.1. Servicio de Mensajes Cortos (SMS, *Short Message Service*)

Fue el primero de los servicios de mensajería que fue implementado en la telefonía celular. La idea surgió en una de las reuniones de los comités de los países europeos para fijar los estándares del sistema GSM, cuando los comités de Alemania y Francia propusieron incluir un nuevo servicio llamado "Transmisión de Mensajes Cortos" (*Short Message Transmission*). Dicho servicio se pensaba fuera para enviar mensajes cortos desde la red fija hasta un suscriptor móvil. Sin embargo, mientras discutían dicho servicio se dieron cuenta que era novedoso y podía marcar una diferencia con respecto a la red PSTN. Es entonces cuando deciden integrar por completo el Servicio de Mensajes Cortos (SMS, *Short Message Service*) en el portafolio GSM. Desde ese momento, todas las arquitecturas celulares que han surgido, han incluido SMS entre sus servicios, convirtiéndolo en el servicio de mensajería más importante de la telefonía móvil. Hoy en día, todo teléfono móvil soporta este servicio, conocido simplemente como Mensajería de Texto. [21]

Cuando se recibe una llamada, la torre envía un paquete de datos al teléfono notificándole acerca de la misma. Dicho paquete, también puede ser utilizado como un paquete de mensaje, básicamente significando para la operadora la oportunidad de

prestar un servicio adicional sin necesidad de gastar en infraestructura extra. Una vez que un celular recibe un paquete de este tipo, extrae su contenido y crea el mensaje de texto que se muestra en pantalla o se almacena en la bandeja de entrada del móvil.

Al enviar un mensaje de texto, éste se empaqueta y se envía a la torre, quien pasa el paquete al Centro de Conmutación Móvil (MSC, *Mobile Switching Center*). El MSC es un dispositivo que se encarga normalmente de enrutar las llamadas de voz, sin embargo cuando recibe un paquete de mensaje lo reenvía al Centro de Servicio de Mensajes Cortos (SMSC, *Short Message Service Center*). El SMSC revisa el mensaje para determinar a dónde se debe enviar. Si el receptor es miembro de otra red inalámbrica, el SMSC observa a qué operadora pertenece y pasa el mensaje al SMSC de ella. En algunos casos, el mensaje puede estar dirigido a un correo electrónico por lo que el SMSC envía el mensaje directamente sobre Internet. Una vez que un SMSC tiene un mensaje que va dirigido a un miembro de su red, lo primero que intenta es ubicar al teléfono de destino utilizando las mismas técnicas que se utilizan para ubicar a un teléfono cuando es solicitado para una llamada. Si el receptor está disponible, el mensaje es enviado; de lo contrario el mensaje es alojado en el SMSC hasta que el suscriptor vuelva a estar disponible. El tiempo que el mensaje es almacenado depende de la configuración del mensaje y de la del servidor. Una vez que el tiempo expira, el mensaje se vuelve inválido y es removido del servidor. [22]

Todos los paquetes de mensajes SMS, como se muestra en la Figura 23, contienen una sección de cabeceras SMS y una carga útil (*payload*) de 140 bytes. Las cabeceras SMS llevan información de la dirección de destino, de la confirmación de entrega, de los períodos de validez, etc. Pueden variar entre operadoras. Por otro lado, el payload, está más estandarizado y es lo que permite que haya compatibilidad entre los mensajes de diferentes operadoras.

Figura 23. Estructura de un paquete SMS. [22]

Los mensajes SMS son comúnmente codificados usando codificación US-ASCII (también conocida como la ASCII Original) o codificación GSM. Ambas son codificaciones de 7 bits por símbolo, y permiten que 160 caracteres se ajusten a los 140 bytes del payload. Si los receptores usan lenguas eslavas o arábigas, los mensajes pueden ser codificados como Unicode (codificación de 16 bits), pudiéndose enviar hasta 70 caracteres. Si por ejemplo el mensaje es enviado a un correo electrónico debe añadirse tal información de igual forma en el Payload, reduciéndose así los caracteres disponibles. Desde hace un tiempo, se pueden escribir mensajes de mayor longitud a la mencionada. Este servicio se conoce como Servicio de Mensajes Extendidos (EMS, *Extended Message Service*). Sin embargo, para el sistema, el mensaje es tratado como múltiples mensajes SMS.

El Servicio de Mensajes Cortos no incluye ningún tipo de garantías, y a pesar que el SMSC está diseñado para dar el mayor esfuerzo en el envío de mensajes, existen numerosas razones por la que la entrega podría fallar.

SMS ofrece las siguientes ventajas: rapidez en la transmisión, posibilidad de intercambiar mensajes en una misma red a pesar de que el servicio de Internet no esté disponible, y alta interoperabilidad entre operadoras. Las desventajas del servicio de SMS son: cada mensaje enviado tiene un costo (en algunos casos hasta por recibirlos), no está disponible a través de una red WiFi®, y el tamaño de su carga útil es bastante limitado. [22]

4.2. Servicio de Mensajería Multimedia (MMS, *Multimedia Messaging Service*)

Con la aparición de los celulares con cámara, también apareció el deseo de compartir fotos. Para satisfacer ello, fue creado el servicio MMS que mediante la

transmisión de datos permitió enviar entre teléfonos tonos, animaciones, y otros archivos multimedia. Nació en la era del Internet por lo que opera sobre este último, es decir, necesita una conexión IP para enviar y recibir mensajes MMS.

El núcleo de un sistema MMS es el Centro de Servicios de Mensajería Multimedia (MMSC, *Multimedia Messaging Service Center*). Éste se encarga de enrutar todos los mensajes multimedia que van hacia o vienen desde sus suscriptores. El MMSC permite comunicarse con los celulares de su red, con servidores (por ejemplo de correo), o con una MMSC de otra operadora.

Un mensaje MMS es transmitido dentro de una unidad de datos de protocolo (PDU, *Protocol Data Unit*). Una PDU es un término utilizado en arquitecturas multicapas para unidades de datos que se envían a una capa equivalente en el equipo de destino. La PDU contiene un conjunto de cabeceras que contienen la información del destino, el tamaño del mensaje, las peticiones para confirmar la entrega, etc. Luego de las cabeceras vienen los archivos adjuntos, cuyos formatos dependerán de los que la operadora haya configurado como permitidos. La estructura de una PDU MMS puede verse a continuación en la Figura 24. [22][23]

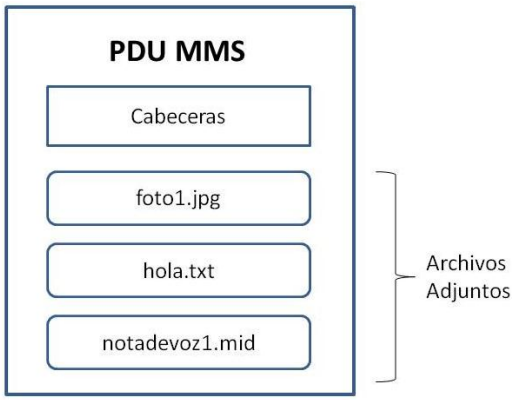


Figura 24. Estructura de una PDU MMS. [22]

Este servicio MMS tiene como ventajas: soporta el envío y la recepción de contenido multimedia, funciona entre equipos de diferentes marcas y soporta largos payloads (a diferencia de SMS). Como desventajas se presentan: la baja

interoperabilidad entre operadoras, requiere conexión de datos a través de una red de telefonía celular (no sirve en WiFi®), cada mensaje es cobrado, y al usar transcoders es probable que el mensaje original sea modificado y no logre verse de la manera adecuada. [22]

4.3. Correo Electrónico (*Email*)

El correo electrónico es uno de los servicios de red más populares de hoy en día. Los sistemas de Email son basados en el esquema cliente-servidor y utilizan los siguientes protocolos para brindar sus servicios: Protocolo Simple de Transferencia de Correo (SMTP, *Simple Mail Transfer Protocol*), Protocolo de Oficina de Correo (POP, *Post Office Protocol*), y el Protocolo de Acceso a Mensajes de Internet (IMAP, *Internet Message Access Protocol*). El protocolo SMTP es un protocolo de la capa de aplicación que trabaja sobre las capas TCP/IP y permite enviar mensajes desde un cliente de correo a un servidor de correo donde se almacenan. Para obtener dichos mensajes del servidor, un cliente de correo utiliza el protocolo POP (actualmente versión 3, POP3) para descargar los mensajes que van dirigidos a él, o el protocolo IMAP (actualmente versión 4, IMAP4) para verlos de manera remota.

Un cliente de correo que funciona con este protocolo, genera un proceso que abre una conexión TCP hacia un proceso en el servidor SMTP remoto que se mantiene a la espera de nuevas conexiones en un puerto específico (puerto 25). Una vez que la conexión TCP es exitosa se presenta un diálogo de tipo petición-respuesta que contiene las direcciones de correo de origen y destino. Si el servidor las acepta, le da la posibilidad al cliente que envíe el mensaje.

El esquema de la comunicación mediante correos electrónicos puede verse en la Figura 25. El usuario emisor tiene en su terminal (celular o computadora) un Agente de Usuario (UA, *User Agent*) que se conecta a un Agente de Transferencia de Mensajes (MTA, *Message Transfer Agent*). Éste es quien realiza el procedimiento anteriormente explicado. Adicionalmente, un MTA mantiene un email en memoria de

manera que pueda repetir su transmisión o realizarla posteriormente en caso que el servidor no sea alcanzable al momento de la transmisión. El servidor también posee un MTA que se encarga de almacenar el correo y enviarlo a la bandeja de entrada del destino, quien mediante la UA será capaz de mostrarlo al usuario final. [24]

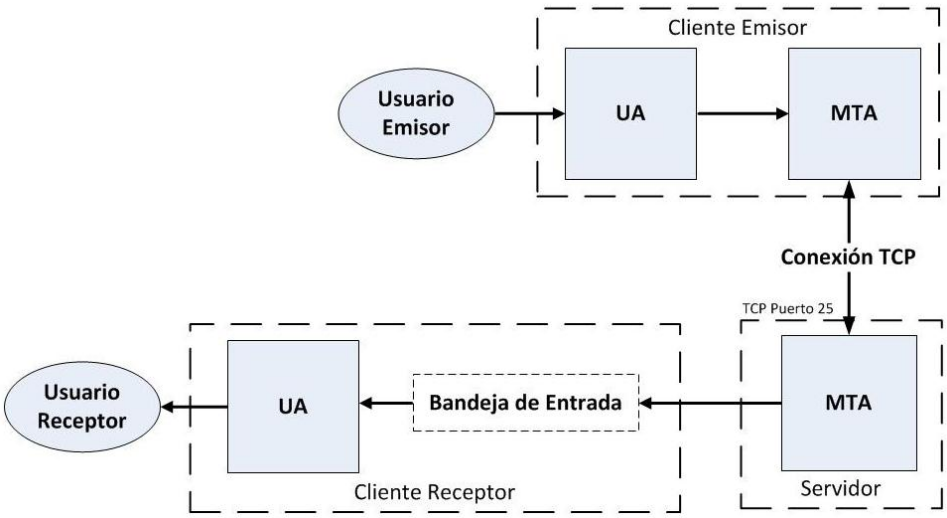


Figura 25. Comunicación mediante correo electrónico. [24]

Las ventajas de utilizar el servicio de correo electrónico son: almacenamiento de mensajes que permite al receptor recibirlo aunque no esté conectado a la red al momento que le es enviado, se pueden enviar o recibir mensajes de gran tamaño, hay buen soporte de texto y otros archivos adjuntos (fotos, animaciones, videos, música, etc.), y si es implementado en celulares su interoperabilidad es mucho mejor que cualquiera de los servicios anteriormente explicados. La desventaja fundamental de este servicio es que no se sabe cuándo el usuario leerá el mensaje, ni si lo recibirá en su móvil o dónde. [22]

4.4. BlackBerry® Messenger

Este servicio de mensajería será descrito en el siguiente punto, ya que antes de entrar en detalles en él, vale la pena describir la plataforma sobre la cual trabaja.

5. PLATAFORMA BLACKBERRY®

La plataforma BlackBerry®, creada por la compañía canadiense Research In Motion® (RIM), es una plataforma que junto con una línea de teléfonos inteligentes (smartphones) brinda servicios de navegación por Internet, correo electrónico, mensajería instantánea, comunicación empresarial, etc. Se hizo muy popular en Venezuela en los últimos años y para el 2010 según el diario El Universal[25] llegó a tener el 63% del mercado de los dispositivos inteligentes que circulaban por el país (aproximadamente unos 800000 equipos de los 1200000 circulantes). También a nivel mundial se convirtió en la marca de móviles más utilizada en América del Norte y América Latina, y la segunda a nivel mundial, con más de 50 millones de usuarios activos y aproximadamente 115 millones de equipos vendidos para Junio de 2010.[26]

5.1. Arquitectura BlackBerry®

La arquitectura de la plataforma BlackBerry® se comporta de la siguiente manera: un celular de esta marca se comunica con la torre de comunicación más cercana de la operadora a la que pertenezca el usuario. La torre está conectada al Centro de Operaciones de la Red de dicha operadora (CNOC, *Carrier Network Operating Centre*). Por otro lado, cada operadora configura una Red Privada Virtual (VPN, *Virtual Private Network*) que le permita conectar su CNOC al Centro de Operaciones de la Red de Research In Motion® (RIM NOC, *RIM Network Operating Centre*). Toda comunicación que ocurra entre esta última y un móvil se realiza de manera segura a través de esas conexiones VPN. Todos los móviles BlackBerry® poseen un único Número Personal de Identificación (PIN, *Personal Identification Number*) que los identifica ante el RIM NOC. Cuando un móvil BlackBerry® es encendido, o cuando entra en una zona de cobertura inalámbrica, se identifica a sí

mismo al RIM NOC quien sabrá cómo comunicarse con él. El RIM NOC sabe sobre cuál portadora está el móvil, y además, cuál conexión VPN debe usar para comunicarse con el mismo. En caso que el móvil esté en roaming, y tenga un Acuerdo de Roaming (GRX, GPRS Roaming Exchange) con la operadora local, el móvil podrá conectarse al RIM NOC, incluso si la operadora extranjera no soporta dicha conexión al NOC de RIM. Específicamente, la data del móvil viajará por la red de datos de la portadora foránea, luego por la de la red de la portadora local y finalmente al RIM NOC a través de la conexión VPN. Todas las comunicaciones en estas redes son realizadas mediante paquetes IP por lo que aplicaciones en el dispositivo ven simplemente a una red de capa 3. [27]

El RIM NOC tiene dos conexiones. Hacia un Servidor Empresarial BlackBerry® (BES, *BlackBerry® Enterprise Server*) o hacia un servidor llamado BlackBerry® Internet Bundle (BIB). A continuación se describen ambos:

(a) BlackBerry® Enterprise Server (BES) [27]

Es implementado en empresas y permite a cada una de ellas tener una conexión directa con sus móviles asociados. Así mismo, permite la conexión entre esos equipos y sus servidores (de aplicaciones, de Internet, de mensajería y de correo electrónico). Cada compañía programa su BES y sus equipos, para que estos realicen únicamente las funciones que se consideren adecuadas.

(b) BlackBerry® Internet Bundle (BIB) [28]

Es el servidor al cual se conectará por defecto cada dispositivo BlackBerry®. Permite comunicarse con servidores de correo electrónico, enviar y recibir mensajes instantáneos entre dispositivos BlackBerry® mediante un servicio llamado BlackBerry® Messenger, y conectarse a Internet para realizar actividades como explorar páginas Web, conectarse a servicios de mensajería instantánea y a servidores de correo, etc.

En la Figura 26, puede verse de manera general cómo es la arquitectura BlackBerry® descrita anteriormente.

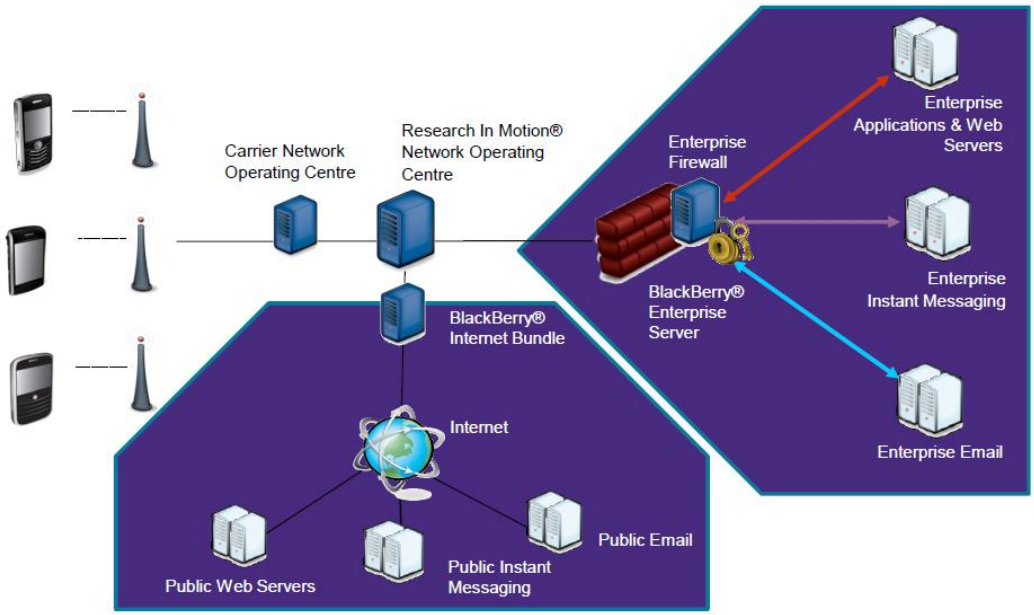


Figura 26. Arquitectura BlackBerry®. [26]

Tanto el BES como el BIB, tienen componentes para: comprimir y encriptar datos, determinar si un móvil debe enrutar datos sobre la red celular o sobre una red WiFi®, asegurarse que los datos se muevan por el camino más rápido posible, hacer posible la visualización de archivos adjuntos de correos electrónicos, y crear un conducto seguro para la comunicación que se establece con un móvil BlackBerry®.

5.2. Servicios brindados por la plataforma BlackBerry®

La plataforma BlackBerry® brinda 3 servicios que han hecho a RIM® una de las compañías de celulares más importantes del mundo. Ellos son: Internet, Email y mensajería instantánea; esta última bajo el nombre de BlackBerry® Messenger. Estos 3 servicios están disponibles en cualquier móvil BlackBerry®, sin importar si su uso es personal o corporativo. Básicamente, la diferencia entre estos dos usos es que un móvil corporativo tendrá las limitaciones o restricciones que la compañía que lo adquirió considere pertinentes.

5.2.1. Internet

El servicio que permite a los usuarios de equipos BlackBerry® conectarse a Internet desde sus móviles es llamado BlackBerry® Internet Service – Browser (BIS-B). Utiliza un protocolo propietario llamado protocolo IPPP (*IP Proxy Protocol*) que permite la transferencia de paquetes TCP y HTTP entre un dispositivo BlackBerry® y el servicio de destino adecuado [29]. Es el servicio que ha hecho fuerte a RIM en el mundo de los celulares con conexión a Internet y dado a que es propietario no existe mucha información acerca de él.

Se pueden describir 3 situaciones en las que un móvil utiliza este servicio para acceder a Internet:

1. El móvil se conecta a la red celular a través de la torre más cercana, que como fue explicado, está conectada a la infraestructura BlackBerry®, específicamente al RIM® NOC. En este caso el BIB de la Figura 26 contiene un servidor BIS-B que a su vez está conectado a Internet y por lo tanto a los diversos servidores que sobre él se alojan.
2. El móvil se conecta a un punto de acceso WiFi® ubicado en la casa del usuario. El punto de acceso directamente está conectado a Internet. A través de Internet el móvil se conecta a la arquitectura BlackBerry® (primero al BIB y luego al RIM® NOC). Luego de nuevo como en la primera situación, el RIM® NOC se conecta al servidor BIS-B para brindar el acceso a Internet.
3. El móvil se conecta a un punto de acceso WiFi® corporativo. El punto de acceso tiene un Firewall, que puede permitir o no el acceso a Internet. Si lo permite, el móvil se conecta a Internet y luego al RIM® NOC, tal cual como en la situación 2.

Las 3 situaciones expuestas pueden ser vistas en la Figura 27.

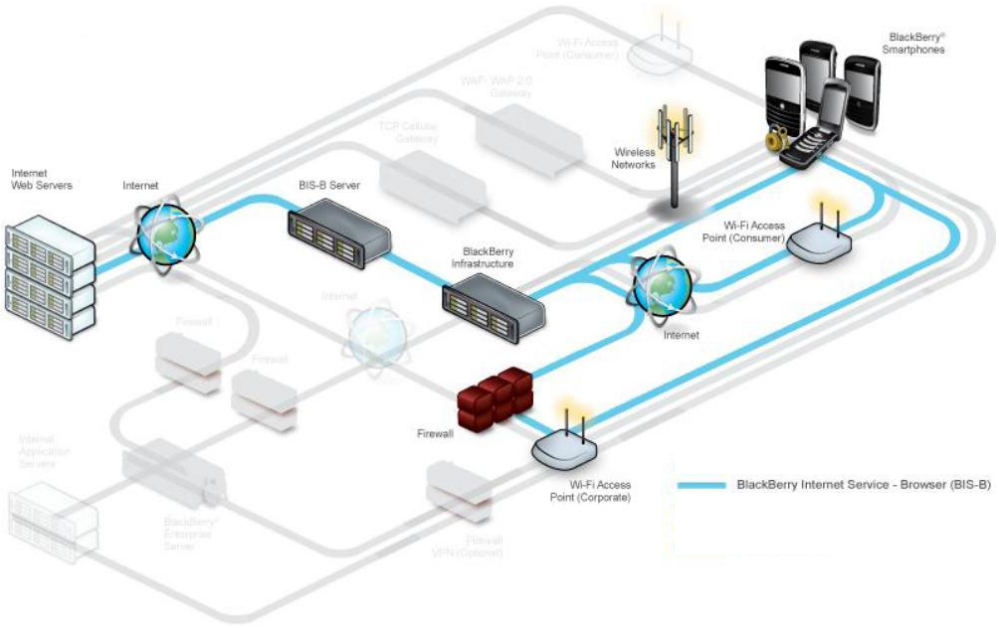


Figura 27. Conexión mediante el Servicio de Internet BlackBerry (BIS). [26]

5.2.2. Email

Cuando un usuario compone y envía un email desde su móvil, el mismo es enrutado a través de la infraestructura BlackBerry® (bien sea a través de la red celular o de una red WiFi®) hasta un servidor SMTP. Una vez que el mensaje llega a dicho servidor, es tratado exactamente como si fuera un correo electrónico enviado desde una computadora. Lo anterior puede verse reflejado en la Figura 28.

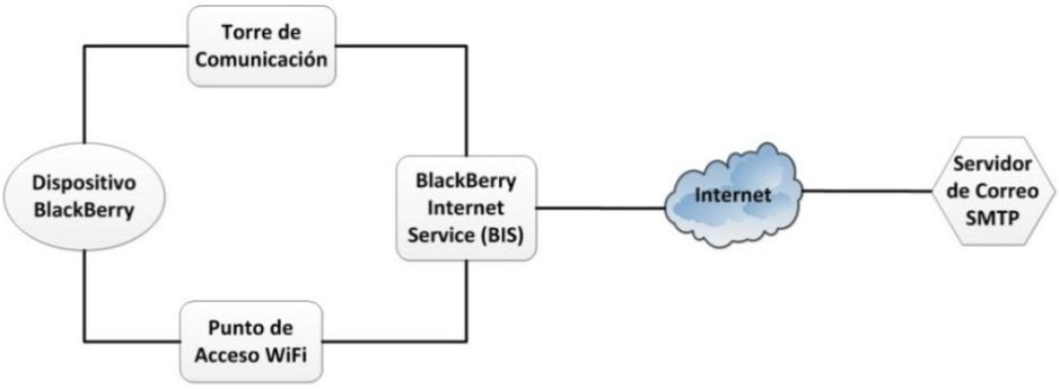


Figura 28. Conexión entre un dispositivo BlackBerry® y un servidor de correo SMTP. [22]

Para recibir correo electrónico, los dispositivos están configurados para conectarse a los servidores utilizando los protocolos POP3 o IMAP. El dispositivo chequea periódicamente, o cuando el usuario lo desee, en el servidor de correo para ver si recibió algún mensaje, y de encontrarlo lo descarga automáticamente al móvil.[22]

5.2.3. BlackBerry® Messenger

El servicio que más impacto causó en su llegada, al menos a los venezolanos, fue el BlackBerry® Messenger. El BlackBerry® Messenger es un servicio de mensajería instantánea diseñado por la propia compañía RIM cuyo funcionamiento se explica a continuación.

Todo celular posee varios números asociados a él, como por ejemplo el Número de Dispositivo Móvil (MDN, *Mobile Device Number*) que representa el número al que discamos desde cualquier otro teléfono para comunicarnos con él, el número de Identificación de un Equipo Móvil (MEID, *Mobile Equipment IDentifier*) que es identifica físicamente a cada celular de otro, entre otros. Los dispositivos BlackBerry® tienen la peculiaridad de poseer un Número de Identificación Personal (PIN, *Personal Identification Number*). El PIN permite a la infraestructura BlackBerry® reconocer cada dispositivo BlackBerry® y conectarse a él. Cualquier teléfono BlackBerry® conectado a Internet y a la infraestructura BlackBerry® podrá entonces enviar mensajes directamente a otro dispositivo BlackBerry®, simplemente conociendo su PIN. Bajo este sistema, no es necesario conocer ni el número ni el correo de una persona para comunicarse con ella. Los mensajes PIN se envían sobre el canal de datos a través de la red, incluso si el móvil no tiene conexión a la red celular (sino únicamente a una red WiFi®), o si se está utilizando el canal de voz.

La comunicación mediante mensajes PIN puede verse en la siguiente figura:

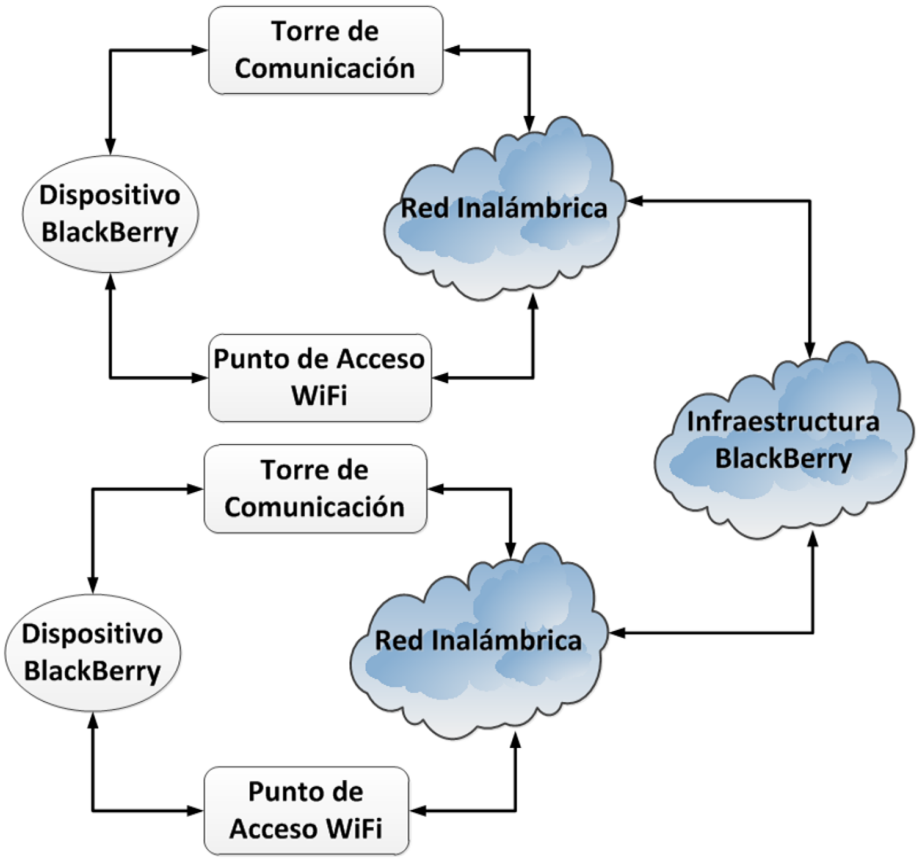


Figura 29. Comunicación mediante mensajes PIN. [22]

Las ventajas de la mensajería instantánea a través del BlackBerry® Messenger son: permite la comunicación entre cada persona que posea un BlackBerry® de manera fácil y prácticamente instantánea sin importar su ubicación, y es un servicio que no es cobrado por mensajes, sino que se cobra la cantidad de datos utilizados. Como desventaja tiene que es un servicio propietario por lo que no es compatible con celulares de otras marcas. [22]

6. JAVA

6.1. Descripción

Java es un lenguaje de programación basado en C++ que fue diseñado en Sun Microsystems en 1991. Actualmente, es bastante utilizado para desarrollar aplicaciones empresariales a gran escala, para mejorar el funcionamiento y funciones de servidores Web, para proporcionar aplicaciones para dispositivos domésticos como celulares, etc.

Los programas en Java son realizados mediante bloques. Cada bloque se conoce como clase, y cada clase está formada por métodos. Un método realiza tareas y devuelve alguna información al completarla. Existen entonces dos maneras de programar con Java: realizando cada una de las clases necesarias para un determinado programa, o aprovechando miles de colecciones de clases disponibles a través de Internet. Estas colecciones se consiguen en bibliotecas virtuales de clases conocidas como Interfaces de Programación de Aplicaciones (API, A*pplication Programming Interface)*. Utilizar APIs reduce el tiempo de desarrollo y aumenta la portabilidad de los programas. [30]

Actualmente, el lenguaje Java se encuentra en su versión 2. A su vez, ésta se divide en 3 ediciones distintas: J2SE, J2EE y J2ME. [31]

6.1.1. J2SE

Es la edición comúnmente utilizada ya que posee un conjunto básico de herramientas que permiten el desarrollo de numerosas aplicaciones de usuario final: aplicaciones con interfaz gráfica, multimedia, comunicaciones en red, etc.

6.1.2. J2EE

Está orientada al entorno empresarial ya que está pensada para ser ejecutada sobre una red de ordenadores de manera distribuida y remota. Con esta versión se desarrollan también servicios web.

6.1.3. J2ME

Está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas reducidas, como teléfonos móviles. Inicialmente utilizaba una máquina virtual denominada KVM (*Kilo Virtual Machine*) que funcionaba a partir de unos pocos kilobytes. En la actualidad, existe una máquina virtual llamada CVM (*Compact Virtual Machine*) con una mayor capacidad para dispositivos con más de 2MB de memoria RAM.

Según la máquina virtual que un teléfono soporte, éste tendrá una configuración específica. Una configuración es un conjunto mínimo de APIs de Java que el dispositivo soporta. Existen entonces dos configuraciones: CLDC y CDC.

- Configuración de Dispositivos Limitados con Conexión (CLDC, *Connected Limited Device Configuration*): está orientada a dispositivos dotados de conexión pero con altas limitaciones en cuanto a capacidad gráfica, de cómputo y de memoria. Estos dispositivos utilizan la máquina virtual KVM.
- Configuración de Dispositivos con Conexión (CDC, *Connected Limited Configuration*): está orientada a dispositivos con cierta capacidad computacional y de memoria. Utilizan la máquina virtual CVM.

Un dispositivo no está únicamente caracterizado por su configuración. Existe un parámetro llamado perfil que también lo define. Específicamente, un perfil asocia a un dispositivo con una funcionalidad específica, por ejemplo si es un celular, un electrodoméstico, etc. Los perfiles existentes son: Foundation Profile, Personal

Profile, RMI Profile, PDA Profile y MIDP Profile. Para este trabajo es importante conocer el perfil MIDP.

- Perfil para Dispositivos de Información Móvil (MIDP, *Mobile Information Device Profile*): está construido sobre la configuración CLDC por lo que está orientado a dispositivos de capacidad reducida. Fue la primera configuración definida para J2ME, y por lo tanto bajo este perfil es que se desarrollan las aplicaciones para celulares. Estas aplicaciones reciben el nombre de MIDlets.

La configuración CLDC tiene dos versiones desde su creación, la versión CLDC-1.0 y la versión CLDC-1.1. De igual forma, el perfil MIDP tiene tres: MIDP-1.0, MIDP-2.0 y MIDP-2.1. A la hora de programar un MIDlet es necesario conocer la versión de ambos parámetros del dispositivo celular para el cual se esté desarrollando el programa. Para los celulares BlackBerry®, por ejemplo, la configuración debe ser la CLDC-1.1 y el perfil el MIDP-2.0. Un programa no funcionará en un celular si está desarrollado en una versión superior a la que dicho teléfono soporte.

6.2. Herramientas de Redes

Java incluye herramientas de comunicaciones de redes basadas en flujos de datos y en paquetes.

Es posible obtener un servicio orientado a la conexión a través de una herramienta llamada “socket”. Utilizando un flujo de sockets es posible establecer una conexión entre dos procesos (por lo general procesos que se presentan en dos terminales distintos). Mientras la conexión esté activa, los datos fluyen entre los procesos continuamente. Un socket utiliza el protocolo TCP para establecer una conexión y permite a las aplicaciones ver a las redes como un archivo, de manera que puede leer o escribir sobre ellas sencillamente.

Para obtener un servicio no orientado a la conexión, Java permite transmitir paquetes individuales de información, conocidos como “datagramas”. En este caso, dichos paquetes son transmitidos utilizando el protocolo UDP. Este protocolo de transporte, ya que no garantiza que los paquetes lleguen en orden adecuado, e incluso ni que lleguen, o que lleguen duplicados, en algunos casos necesita programación adicional para lidiar con esos problemas. [30]

6.3. JSIP API Specification v1.2

Se mencionó anteriormente que Java se caracteriza por permitir la programación por bloques. De esta manera, se pueden obtener a través de Internet, o de otros programadores, bibliotecas de clases llamadas APIs que contienen herramientas que a la larga facilitan la programación.

La API de nombre “JSIP API Specification v1.2” es una biblioteca de clases totalmente libre diseñada para programar aplicaciones basadas en el protocolo SIP. También es conocida como “JainSip”.

Soporta todas las funcionalidades especificadas en la RFC 3261, el método MESSAGE de la RFC 3428, y varias de las demás extensiones asociadas al protocolo SIP que existen hoy en día.

Está diseñada para funcionar en la edición J2SE de Java de manera asíncrona. Contiene clases para crear mensajes de peticiones, mensajes de respuestas, y cabeceras SIP. Define una herramienta llamada “SipProvider” para el envío de mensajes a través de la red y otra de nombre “SipListener” para recibirlos. [32]

6.4. JSR180

Al igual que la JSIP API para J2SE, existe una API para J2ME. Esta biblioteca tiene el nombre de JSR180 y permite enviar y recibir mensajes SIP desde

un equipo de recursos limitados, como por ejemplo un celular. Fue diseñada en 2004 por Nokia Corporation y tiene como requisito que el dispositivo tenga una configuración CLDC-1.0 o superior. A pesar que está disponible para su descarga en Internet, es una librería totalmente diseñada por Nokia, y por lo tanto es propietaria. Cualquier aplicación que utilice esta API, para que sea distribuible, debe gozar con el consentimiento de Nokia Corporation. [33]

6.5. Entornos de Desarrollo Integrado (IDE, *Integrated Development Enviroment*)

Existen numerosos programas para desarrollar aplicaciones en Java. Lo más importante de los entornos de programación de Java, es que son software libre. Entre los más comunes están NetBeans y Eclipse.

6.5.1. NetBeans

NetBeans es un entorno de desarrollo integrado (IDE, *Integrated Development Enviroment*) de código abierto fundado por Sun MicroSystems en junio de 2000. Dicha compañía continúa siendo la patrocinadora principal de sus proyectos.

Sirve para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso. Puede ser utilizado tanto para uso comercial como para no comercial.

Al descargar e instalar NetBeans, es posible programar en cualquier de las ediciones Java 2: J2SE, J2EE y J2ME, e incluso en otros lenguajes de programación. [34]

6.5.2. Eclipse

Eclipse, al igual que NetBeans, es un entorno de desarrollo integrado (IDE) de código abierto creado por IBM en noviembre de 2001. También está escrito en Java y permite escribir, compilar, depurar y ejecutar programas. A diferencia de NetBeans, al ser descargado e instalado, únicamente podrá programarse para J2SE y J2EE. Para poder programar por ejemplo en J2ME, es necesario descargar paquetes de software adicionales que son utilizados en conjunto con el software base para extender sus capacidades (*plugins*). A pesar de que suene contraproducente, esa es una de sus cualidades más atractivas para los programadores, ya que pueden incrementar las capacidades del entorno tanto como lo deseen. [35]

Como prueba de lo anterior, está el plugin que permite desarrollar aplicaciones específicas para dispositivos BlackBerry®. Tiene como nombre "The BlackBerry® Java Plug-in for Eclipse" y provee herramientas específicas para desarrollar, depurar y simular aplicaciones con la totalidad de las capacidades que dichos dispositivos poseen. [36]

METODOLOGÍA

Previo a la realización de la arquitectura, fue necesario seleccionar algunos aspectos que constituirían las bases sobre las cuales se construyó la misma. Entre ellos se pueden mencionar la elección de software utilizado para programar, la elección del software servidor, determinación de los parámetros de programación, etc. A continuación una explicación más detallada de tales aspectos.

- Lenguaje de programación: como lenguaje de programación se utilizó Java. Primordialmente por dos razones, la primera es que el software para programar en Java es libre, y la segunda, es que Java es un lenguaje bastante popular por lo que se puede conseguir bastante documentación acerca del mismo.
- Software de programación: debido a que Java es libre, existen numerosos entornos de programación que permiten escribir códigos basados en su lenguaje. Dos de los más populares son NetBeans y Eclipse. King [22] recomienda la utilización del entorno Eclipse cuando se realicen programas que vayan a ser ejecutados en dispositivos BlackBerry®, ya que en el mismo pueden ser instalados componentes de depuración y simulación nativos de dicha marca. Eclipse fue entonces el software utilizado. Específicamente, la versión Galileo 3.5 (build ID 20090621-0832) de la edición "Eclipse IDE for Java EE Developers"[37]. Para instalar Eclipse es necesario instalar un conjunto de utilidades conocidas como "Java SE Runtime Enviroment"[38]. Fue instalada específicamente la versión 1.6.0.27 (build 1.6.0_27-b07).

  Adicionalmente a Eclipse, para que fuese posible generar aplicaciones para celulares, fue necesario instalar una extensión (o plug-in) para él. Dicha extensión es llamada "The BlackBerry® Java Plug-in for Eclipse"[36]. Se usó específicamente la versión 1.1.2.201004161203-16. Este plugin trae por defecto un simulador del equipo táctil BlackBerry® Storm 9500. Como el

equipo que se disponía era un BlackBerry® Curve 8520, debió descargarse el simulador del mismo[39]. Se descargó con el sistema operativo BlackBerry® OS 5.0.0.681.

- Tipo de aplicación: como lo indica el título del trabajo, debía diseñarse e implementarse una aplicación de mensajería instantánea para celulares con soporte WiFi®. Al hablar de celulares con soporte WiFi®, no se está haciendo énfasis en algún modelo o marca específica, y por lo tanto la aplicación diseñada debía ser capaz de ejecutarse en cualquiera de ellos, o al menos en la mayoría. Por tal razón, dicha aplicación se realizó en formato MIDlet. Se debe tener atención a la hora de diseñar un MIDlet, ya que debe conocerse el perfil y la configuración del teléfono (o del grupo de teléfonos) para el cual se quiere implementar. Usando el plugin de Eclipse ya mencionado, no hizo falta seleccionar ni el perfil ni la configuración, ya que los mismos están configurados por defecto para funcionar en dispositivos BlackBerry® (perfil MIDP-2.0 y configuración CLDC-1.1). En el caso que se desee utilizar otro IDE, se debe estar atento con dichos parámetros.
- Arquitectura y APIs necesarias: inicialmente se quería lograr una arquitectura mediante la cual dos equipos celulares se enviaran entre ellos mensajes SIP a través de una red WiFi®. Esto se lograría a través de un servidor SIP proxy ubicado dentro de esa misma red como lo muestra la Figura 30.

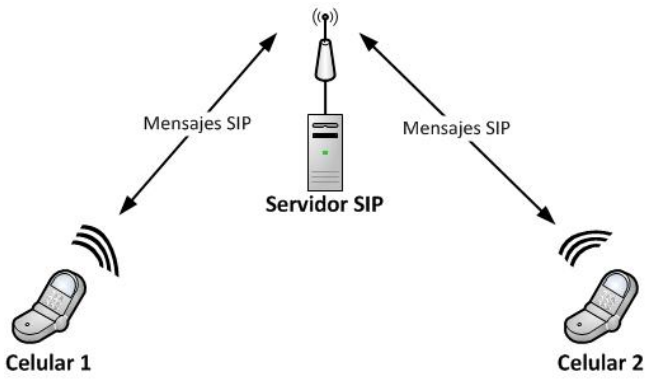


Figura 30. Arquitectura SIP inicialmente pensada.

Debía entonces diseñarse una aplicación para los celulares (programando en J2ME) que implementase dicho protocolo. Existían dos maneras de lograrlo; la primera, implementando datagramas UDP o sockets para construir desde cero las clases necesarias para cumplir con las especificaciones de la RFC 3261; y la segunda, a partir de la ya existente API JSR180.

Dada la complejidad de la primera, y debido a que se requería un alto grado de conocimientos (y dominio de ellos), en especial acerca de las herramientas de red en Java (sockets y datagramas UDP), de programación en J2ME, y por supuesto del protocolo, fue desechada inmediatamente. Quedaba entonces la implementación de la API JSR180 como la única opción disponible. Sin embargo, debido a que es una API propietaria que requiere permiso de Nokia Corporation para ser distribuible, y a que se tenía como fin depender en la menor medida de servicios propietarios para que el resultado fuese una aplicación altamente portable, el funcionamiento de la misma en dispositivos de diversos modelos y marcas debía ser probado. Se utilizó el código de ejemplo proporcionado en la página de Nokia donde se describe la JSR180[33]. Fue probado en cuatro equipos, dos de marca BlackBerry® (modelos Curve 8520 y Pearl 8120) y dos de marca Nokia (modelos E71 y N8). La aplicación únicamente se ejecutó en el Nokia E71 y utilizando el analizador de protocolos Wireshark®, fue posible verificar que los mensajes SIP eran enviados correctamente. Ante esos resultados, se hacía constar que utilizar la API JSR180 también ocasionaba problemas de portabilidad. Como esto se quería evitar a toda costa, se decidió no implementar tampoco dicha opción. Ante esto, debía pensarse en una nueva forma de implementar SIP en la aplicación de mensajería instantánea. Surgió entonces la idea definitiva de reducirle las responsabilidades a los celulares, y otorgárselas a computadoras, donde la portabilidad no es un problema. Entonces, se diseñó una aplicación para computadoras en lenguaje J2SE en la cual el protocolo SIP fue implementado con la librería JSIP v1.2, y que sirviera como intermediario entre un celular y un servidor SIP. La

comunicación entre un celular y su computadora asociada se realizó mediante sockets, aumentando la portabilidad de la aplicación. Bajo este esquema, en prácticamente todo celular con soporte para WiFi® y para Java, la aplicación puede ser ejecutada y funcionar de manera correcta. La siguiente figura permite entender fácilmente cómo quedó la arquitectura SIP realizada:

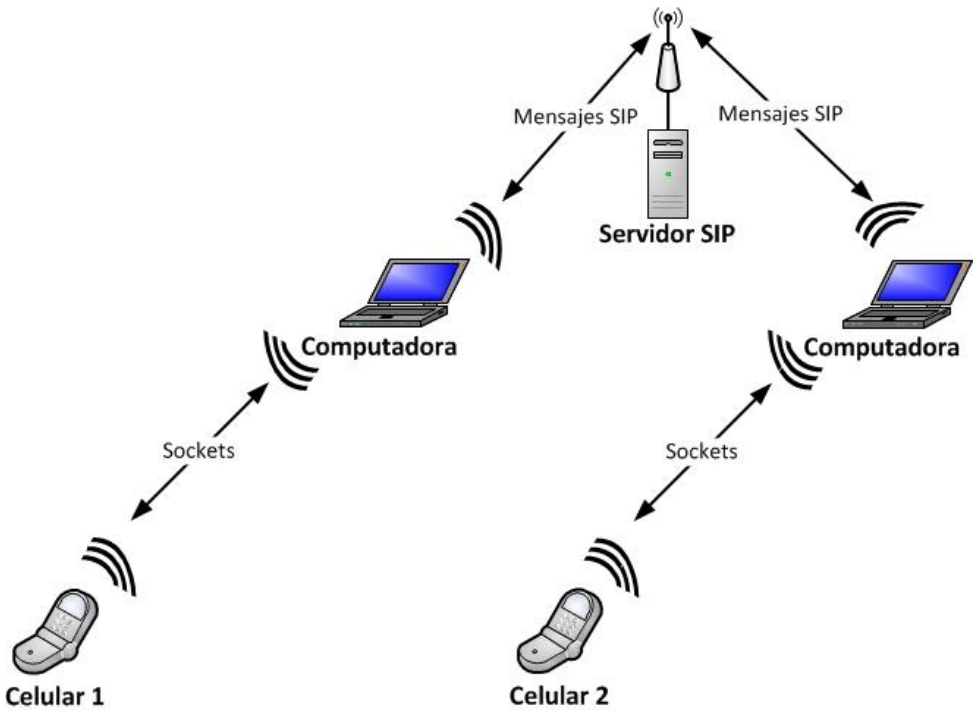


Figura 31. Arquitectura SIP final.

- Números de puertos: de acuerdo con la RFC 3261 un cliente SIP debe comunicarse con el puerto 5060 de un servidor. Los puertos mediante los cuales recibirían los sockets el celular y la computadora sí debían ser elegidos. Para garantizar la disponibilidad de esos puertos se eligieron en el rango de los puertos libres que define la Agencia de Asignación de Números de Internet (IANA, *Internet Assigned Numbers Authority*)[42]. Dentro de ese rango (49152-65535) se decidió que la aplicación del celular debía recibir en el puerto 62528 y que el cliente SIP lo hiciera en el 62527. Ambos números fueron elegidos de manera arbitraria.

- Servidor SIP: el IDE Eclipse y el analizador de protocolos fueron instalados en Windows porque la computadora que se tenía a disposición operaba bajo dicho sistema operativo (específicamente bajo Windows 7). Entre los servidores SIP más populares mencionados anteriormente, se eligió el único compatible con Windows, el servidor 3CX® [26]. Teniendo el servidor, el IDE y el analizador de protocolos ejecutándose en una misma computadora en un determinado momento, bastaría para simular y verificar el desenvolvimiento del código que se iba escribiendo. Las desventajas de utilizar dicho software eran que se estaría haciendo uso de un software propietario y que además, éste tiene de manera predefinida unos parámetros que podían afectar el funcionamiento de la aplicación. Sin embargo, se observó que servidores como el OpenSIPS y el Asterisk, permiten la configuración de cada uno de tales parámetros, de manera que si la aplicación era capaz de funcionar en el servidor 3CX® fácilmente podría hacerlo en los otros servidores de código abierto, al menos imitando dichos parámetros. Incluso, si se determina que alguno de los parámetros del 3CX® afecta el comportamiento de la aplicación, es posible corregirlo en estos servidores basados en Linux.

Una vez determinado el software requerido, se procedió a la instalación y configuración del mismo.

Como ya se mencionó, el software requerido fue: Java SE Runtime Enviroment, Eclipse IDE for Java EE Developers, The BlackBerry® Java Plug-in for Eclipse, y el servidor 3CX®. Esos programas fueron instalados de la manera tradicional, a excepción de Eclipse que no requirió instalación. Simplemente, se descargó como un archivo comprimido (.zip) que al extraerse contenía una carpeta con el archivo ejecutable (.exe).

Por otro lado, el servidor 3CX inmediatamente después de ser instalado solicitó ser configurado. El primer parámetro que se configuró fue el idioma. Se

seleccionó el idioma español y se hizo click en siguiente. Apareció una segunda ventana para configurar la dirección IP local predeterminada como se muestra en la Figura 32.

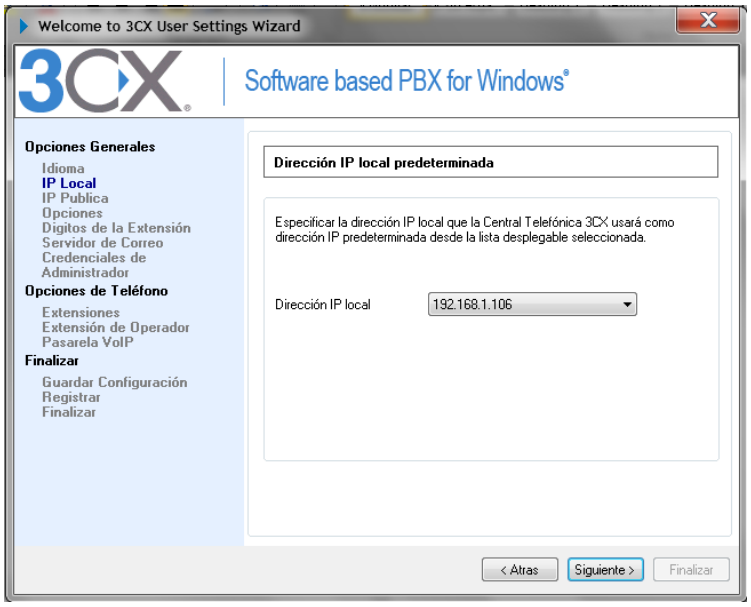


Figura 32. Configuración de dirección IP local predeterminada.

En ella aparecía una lista desplegable con varias direcciones IP, por lo que se debió ejecutar el símbolo del sistema para conocer cuál IP utilizar. En el símbolo del sistema se escribió el comando "ipconfig". Entre los datos que aparecieron se encontraba la dirección IP perteneciente al adaptador de LAN Inalámbrica como lo muestra la Figura 33. En este caso, el punto de acceso inalámbrico (Router TPLink MR-3420) de dirección IP 192.168.1.1, estaba configurado para asignar direcciones IP dinámicas y asignó a la computadora donde se instaló el 3CX la dirección IP 192.168.1.106. Posteriormente se configuró el punto de acceso para que esa dirección fuese estática y se mantuviera asociada a dicha computadora.

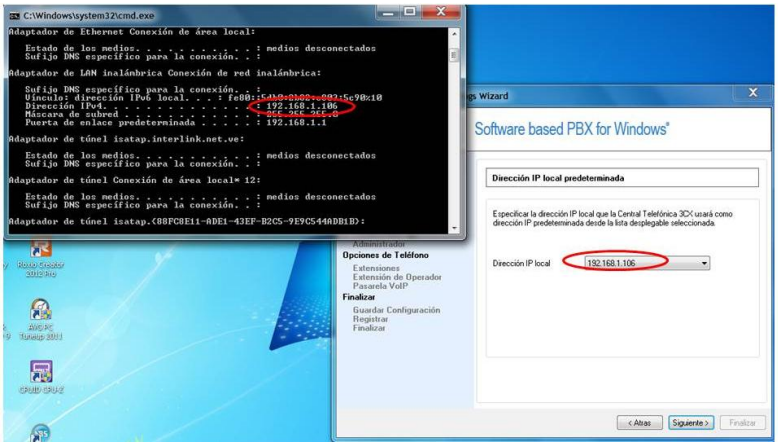


Figura 33. Selección de la IP local.

Una vez seleccionada la IP correcta se hizo click en Siguiente. Apareció entonces una ventana para la configuración de la IP Pública del servidor. Como la arquitectura planteada en este trabajo, tenía como fin implementarse dentro de una red WiFi® y no entre diferentes redes, se colocó en este campo la misma dirección seleccionada como IP Local.

Figura 34. Configuración de IP Pública.

En la siguiente ventana se seleccionó la opción “Crear Nueva Central” y se hizo click en Siguiente. Ahora aparecía una ventana para configurar el número de

dígitos que tendrían los números de extensión. La opción seleccionada fue la de 2 dígitos.

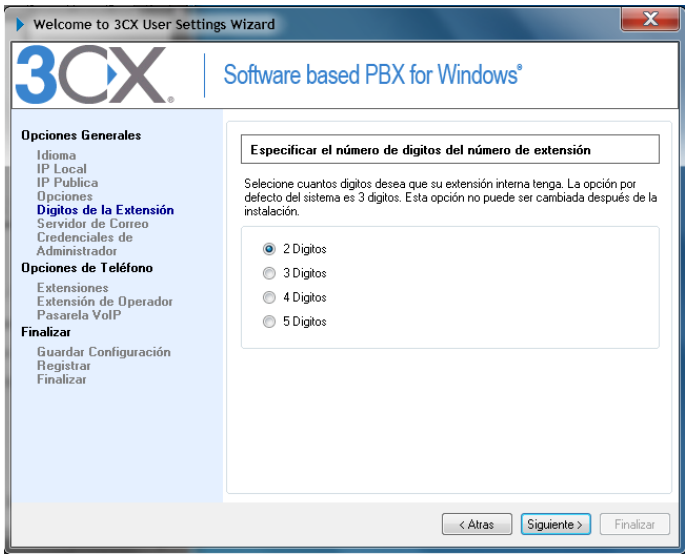


Figura 35. Selección del número de dígitos de las extensiones.

La siguiente configuración era para los parámetros de correo de voz. Como no iba a ser implementada fue dejada como estaba por defecto. Seguidamente, surgió la ventana de configuración de nombre de usuario y contraseña del administrador. Se eligió “admin” como nombre de usuario y contraseña, y se hizo click en siguiente.

En la ventana de configuración de extensiones de usuario fueron añadidas tres extensiones:

- Extensión 10 con el parámetro ID igual a “10” y sin contraseña.
- Extensión 11 con el parámetro ID igual a “11” y sin contraseña.
- Extensión 12 con el parámetro ID igual a “12” y sin contraseña.

En las próximas ventanas hasta la de “Pasarelas / Proveedores VoIP”, se hizo click en Siguiente sin modificar los valores por defecto. En esta última en cambio, se hizo click en “Saltar>>” y finalmente quedó correctamente configurado el servidor 3CX.

Seguidamente de la instalación y configuración del software requerido, se pasó a una etapa de programación que debió realizarse en dos partes: la primera para desarrollar la aplicación para móviles que fue escrita en J2ME, y la segunda para obtener al programa para computadoras que fue escrito en J2SE.

La interfaz gráfica y la utilización de los sockets en la aplicación J2ME, tienen como base un pequeño código de ejemplo de un MIDlet multipantalla[31]. Mientras que el cliente SIP elaborado, tiene como base el código de ejemplo que proporciona la descripción de la API JSIPv1.2[32]. A partir de dicho ejemplo y con la documentación de esa API[40], fue posible obtener una clase que manejara al protocolo SIP (creación de cabeceras, escucha de respuestas, envío de peticiones, etc.). Adicionalmente, se implementó una clase encargada de generar un response de autenticación ante un reto solicitado por una respuesta 407 (método de autenticación Digest), el código de ésta fue encontrado en Internet y únicamente debió ser modificado para trabajar sin contraseña [43]. Las clases de multicast fueron diseñadas también a partir de un código de ejemplo conseguido en Internet. [44]

A pesar que en el párrafo anterior se describe superficialmente, vale la pena entrar en detalle de cómo fue creada la aplicación J2SE a partir del código de ejemplo de la API JSIPv1.2. Dicho código generaba una aplicación que permitía enviar y recibir mensajes instantáneos utilizando protocolo SIP. La interfaz gráfica de la misma se puede ver en la siguiente figura:

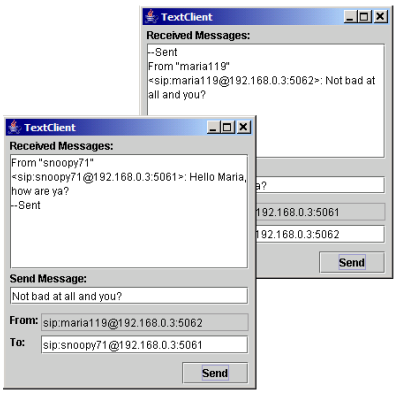


Figura 36. Aplicación ejemplo de la API JSIPv1.2. [32]

El código producía un problema para la arquitectura planteada ya que para que un cliente enviara un mensaje, debía conocer el nombre de usuario, la dirección IP y el puerto que usaba la aplicación del receptor. La mayoría de los puntos de acceso implementados hoy en día asignan direcciones IP dinámicas, de igual forma la mayoría de las aplicaciones que envían datos lo hacen por puertos seleccionados de manera aleatoria. Esto hacía imposible implementar dicho código en la arquitectura sin antes modificarlo. Ahora la pregunta era cómo establecer el envío de mensajes sin la necesidad de conocer dichos parámetros. La solución a ello la proporcionaría un servidor proxy, en este caso como ya se había decidido, sería el servidor 3CX®. Para conectarse a él se debió añadir al código el método REGISTER según las especificaciones propuestas en la RFC 3261. Una vez implementado, se consiguió un inconveniente con el servidor 3CX®. Éste respondía al REGISTER con un mensaje de respuesta 407 y por lo tanto el cliente debía autenticarse. Mediante el analizador de protocolos Wireshark® se observó la cabecera Proxy-Authorization de dicho mensaje. En ella decía que el esquema de autenticación llevaba como nombre "Digest". Cuando se investigó acerca de este esquema se consiguió un código en lenguaje Java que permitía al cliente autenticarse con dicho método. Se agregó entonces esa clase al código que se tenía y se corroboró mediante el Wireshark® que los usuarios se autenticaban de manera correcta y que por ende el servidor los aceptaba. Ya teniendo todas las partes de código necesarias se procedió a la organización de las mismas para la obtención del programa final. El programa implementaba los métodos REGISTER y MESSAGE, y tenía definido cómo actuar ante respuestas 200 OK y 407 Proxy Authentication Required. Posteriormente, se añadió cómo actuar ante respuestas 404 Not Found, cómo desconectarse de un servidor y cómo actuar ante otras respuestas que no fueron definidas.

Otra información respecto a cómo programar un MIDlet en J2ME con Eclipse, cómo añadir una API a un proyecto, cómo pasar un programa realizado a un dispositivo BlackBerry® (o cualquier otro celular), o cómo crear un archivo ejecutable .jar, se puede encontrar en los Anexos 1, 2, 3 y 4.

Tras la obtención de los programas, se procedió a evaluarlos y determinar si cumplían con los requerimientos inicialmente planteados. La evaluación se realizó con el software Wireshark® para captar los mensajes involucrados en las siguientes situaciones: registro y desconexión de la aplicación J2SE con el servidor, intercambio de mensajes entre dos computadoras con la aplicación J2SE (Figura 37), registro con el servidor SIP bajo un nombre de usuario no existente, y conexión de la aplicación J2ME con la aplicación J2SE (Figura 38).

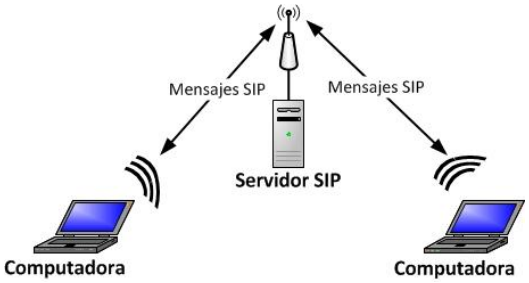


Figura 37. Arquitectura final sin celulares.

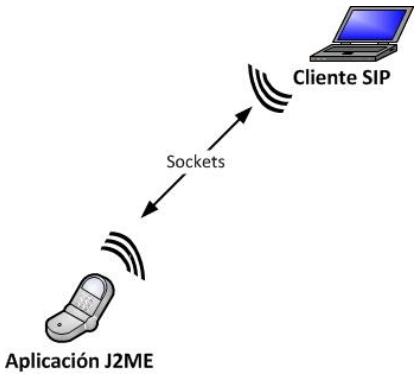


Figura 38. Conexión entre la aplicación J2ME y el cliente SIP.

Se estudió también la compatibilidad de la aplicación J2SE con el cliente SIP propietario X-Lite® (versión 4) y la compatibilidad de la aplicación J2ME con un equipo BlackBerry® Pearl 8120 y con un equipo Nokia N78.

Finalmente se plantearon posibles implementaciones de la arquitectura diseñada en la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería de la U.C.V.

CAPÍTULO III

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

La arquitectura de mensajería fue planteada inicialmente como se mostró en la Figura 30. Bajo tal planteamiento, bastaba con desarrollar una aplicación para celulares en lenguaje J2ME, que permitiera la conexión de ellos con un servidor SIP a través de los métodos establecidos en dicho protocolo. A través de ese servidor serían enviados los mensajes entre los dispositivos móviles. Por las complicaciones ya expuestas para implementar SIP en los celulares, esta arquitectura debió ser modificada hasta obtener la de la Figura 31.

La arquitectura de mensajería instantánea final está constituida por dos aplicaciones, una escrita en leguaje J2SE para computadoras, que sirve como intermediaria entre un celular y un servidor SIP, y otra escrita en J2ME que ejecutada sobre celulares, permite la comunicación entre ellos y la aplicación J2SE.

Cuando una persona envía un mensaje con la aplicación J2ME, éste va a la aplicación J2SE de su computadora asociada (mediante sockets a través de la red WiFi®). Esta aplicación J2SE lo recibe, lo introduce en un mensaje SIP y envía este último al servidor SIP, quien reenvía el mensaje a la aplicación J2SE de destino. Ésta, extrae el contenido del mismo y determina cuál es el verdadero mensaje a entregar. Una vez determinado, lo envía mediante sockets a través de la red WiFi® a su celular asociado.

Es importante mencionar, que la conexión entre una computadora (con la aplicación J2SE) y el servidor SIP puede realizarse por una red inalámbrica o alámbrica, sin que ello afecte el comportamiento del sistema.

Esta última arquitectura fue analizada en búsqueda de ventajas o desventajas con respecto a la primera. Las siguientes, son consideradas como ventajas:

- Confiabilidad en la entrega de paquetes a través de la red WiFi®: la comunicación entre un celular y una computadora (o viceversa) se establece mediante sockets, haciéndola confiable. Como fue explicado anteriormente, los sockets permiten prestar un servicio orientado a la conexión. Se basan en el protocolo TCP para establecerla y enviar datos a través de ella. La arquitectura inicial se pensaba implementar usando UDP para el envío de datos entre las computadoras y los celulares de manera que no hubiera existido confiabilidad en la entrega de paquetes a través de la red inalámbrica.
- Portabilidad: los sockets son una de las dos herramientas básicas de red del lenguaje Java. En el caso de la aplicación para celulares, ello implica que prácticamente cualquier teléfono que soporte J2ME, y que sea capaz de conectarse a una red WiFi®, pueda ejecutarla sin problemas. El único requerimiento para el teléfono es que sea de perfil MIDP-2.0 y de configuración CLDC-1.1, o superiores. Por otro lado, la portabilidad de la aplicación de computadoras es incluso mayor, ya que prácticamente todas ellas, sin importar su sistema operativo (Windows, Linux, MAC, etc.), son capaces de ejecutar aplicaciones escritas en Java. Bajo la primera arquitectura planeada se hubiese requerido implementar la API JSR180 en la aplicación J2ME, ocasionando así altos problemas de compatibilidad como el que ocurrió probando el código de ejemplo de dicha API en los equipos Nokia y BlackBerry® ya mencionados.
- Doble funcionalidad: además de cumplir con la función para la cual fue creada (intercambio de mensajes entre celulares), la estructura de la arquitectura permite comunicarse de otras dos maneras, entre computadoras (Figura 39), o entre un celular y una computadora (Figura 40).

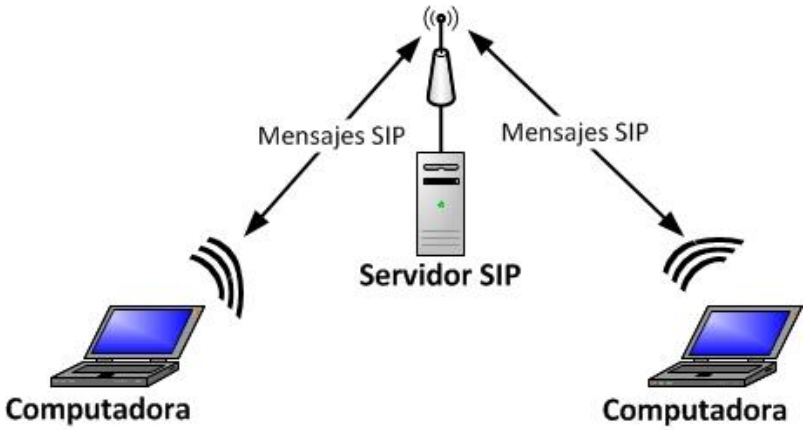


Figura 39. Comunicación entre computadoras a través de la arquitectura final.

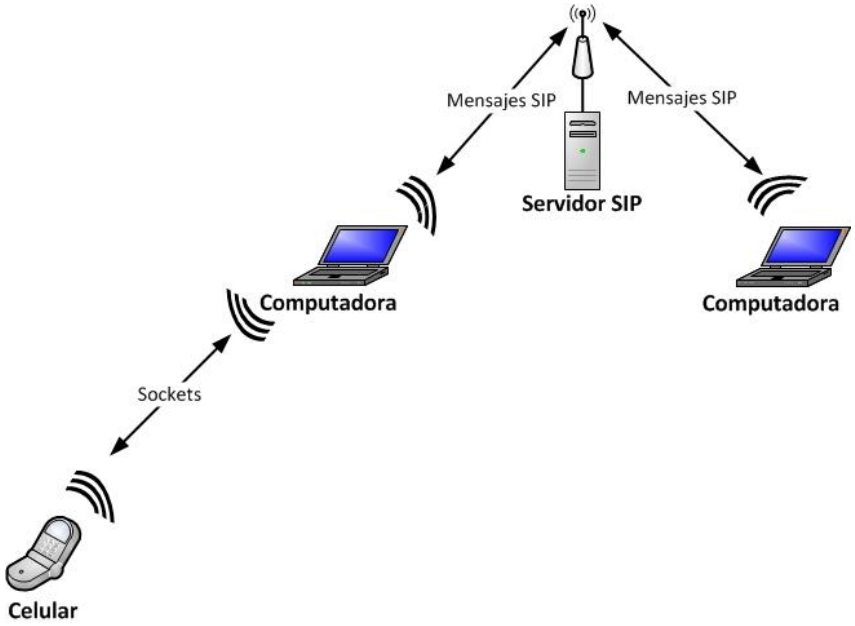


Figura 40. Comunicación entre un celular y una computadora a través de la arquitectura final.

Por otra parte, esta nueva arquitectura tiene ciertas desventajas respecto a la primera arquitectura planteada. Ellas son:

- Menor velocidad: usar sockets para la comunicación entre los celulares y las computadoras a pesar de que es confiable (ya que se usa el protocolo TCP), es un mecanismo lento en comparación con otros que utilicen UDP como protocolo de transporte. Esto ocurre ya que dicho paquete no es enviado

hasta no establecer la conexión. La primera arquitectura hubiera permitido un envío de mensajes más rápido debido a que usaría UDP.

- Dependencia de una computadora: la nueva arquitectura hace depender a la aplicación del celular de una computadora que funcione como intermediaria entre ella y un servidor SIP. Un usuario que no posea una computadora no podrá conectarse a ella.

Ya descritas las ventajas y las desventajas de la arquitectura diseñada e implementada, se pasará a describir las aplicaciones realizadas mediante la Tabla 3:

Tabla 3. Características de las aplicaciones realizadas.

	Aplicación para las computadoras	**Aplicación para los celulares**
Funciones	-Comunicarse con un celular mediante sockets. -Comunicarse con un servidor SIP. -Servir como cliente SIP. -Servir de intermediaria entre un celular y un servidor SIP.	-Comunicarse con una computadora mediante sockets.
Lenguaje	Java	Java
Edición Java	J2SE	J2ME
Requisitos Java	Java SE Runtime Enviroment Versión 1.6.0.27, o superior	Perfil MIDP-2.0 y Configuración CLDC-1.1, o superiores
Tamaño	1.07 MB	8.704 kB
Archivos .java	8	7
Puerto de envío	Para Sockets: 62528 Para mensajes SIP: Aleatorio*	62527
Puerto de recepción	Para Sockets: 62527 Para mensajes SIP: igual al de envío	62528

*Entre 49152 y 65536

En la tabla previa se indicaron cuántos archivos .java fueron creados por aplicación. Esos archivos permiten realizar una programación modular y por ende más sencilla. Así mismo, este esquema de programación modular permite a los

programadores añadir o quitar funciones de un programa para adaptarlo a sus necesidades. Los archivos .java en conjunto constituyen todo el código fuente.

El código fuente de la arquitectura desarrollada, fue entregado al Jefe del Departamento de Comunicaciones de la Escuela de Ingeniería Eléctrica de la Universidad Central de Venezuela (Prof. Carlos Moreno). La descripción de los archivos .java generados se encuentra en la siguiente tabla:

Tabla 4. Descripción de los archivos .java de las aplicaciones elaboradas.

Aplicación J2SE		**Aplicación J2ME**	
Nombre del archivo	**Descripción**	**Nombre del archivo**	**Descripción**
ClienteTexto	Contiene el código de la interfaz gráfica y de las acciones que el usuario haga sobre ella. Funcionando como cliente se comunica con CapaSIP para enviar y recibir mensajes SIP. Si funciona como intermediaria entre el celular y el servidor SIP cede sus funciones a ConectadoCelular.	MidletMensajeria	Contiene el código que permite iniciar y cerrar el MIDlet. También controla la creación de ventanas de interfaz gráfica.
CapaSIP	Contiene el código que hace posible el envío y la recepción de mensajes SIP. Esto lo realiza mediante las herramientas SipProvider y SipListener de la API JSIPv1.2. Permite configurar los parámetros de dichos mensajes, como por ejemplo las cabeceras, el protocolo de transporte, etc.	Enviar	Contiene el código que permite enviar datos mediante sockets hacia una computadora.
MessageProcessor	Permite a CapaSIP modificar parámetros de la interfaz gráfica de ClienteTexto en tiempo real. Por ejemplo al recibir un mensaje.	Recibir	Contiene el código que permite al celular recibir sockets provenientes de una computadora.
Client Authentication Method	Permite a CapaSIP utilizar a DigestClientAuthenticationMethod	EnvioFORM	Contiene el código que genera la forma de envío.
DigestClient Authentication Method	Contiene el código que sirve para generar el response necesario para autentificarse si se presenta un mensaje que solicite un reto de autenticación (mensajes 407).	MenuLIST	Contiene el código que genera una lista utilizada como menú.

ConectadoCelular	Contiene el código que controla la comunicación con el celular. De igual forma, maneja a CapaSIP para enviar y recibir mensajes SIP.	RegistroFORM	Contiene el código que genera la forma de registro donde se indica la IP de la computadora con la cual se conectará el celular.
EnviarSocket	Contiene el código que permite a la computadora enviarle datos mediante sockets al celular.	RegistroFORM2	Contiene el código que genera la segunda forma de registro donde se indica la IP del servidor SIP y el nombre de usuario.
RecibirSocket	Contiene el código que permite a la computadora recibir sockets provenientes de un celular.		

Los diagramas UML de las aplicaciones J2SE y J2ME, se encuentran en la sección de Anexos (Anexos 5 y 6, respectivamente). En ellos puede observarse la interacción entre las clases que las conforman.

Las clases para Multicast fueron diseñadas bajo los nombres EnvioMulticast y RecepcionMulticast. Como puede inferirse de sus nombres, la primera contiene el código para enviar datos hacia una dirección de multidifusión IP y la segunda para recibirlos. Estas clases no fueron implementadas pero gracias a que la programación en Java es modular, las mismas pueden ser añadidas en cualquier momento futuro a las aplicaciones ya existentes. Cuando se deseen implementar se necesitará modificar ciertas partes del código existente para ajustar la interfaz gráfica a los nuevos requerimientos y obtener de ella la dirección IP del servidor. En el código fuente de estas clases puede verse dónde irán la dirección IP y el mensaje cuando desee enviarse, o la dirección IP cuando desee recibirse. Dichas posiciones están marcadas como “AQUÍ VA LA DIRECCIÓN IP DEL SERVIDOR” o “AQUÍ VA EL MENSAJE A ENVIAR POR MULTICAST”.

A continuación se explica cómo utilizar las aplicaciones elaboradas.

- Aplicación J2SE

1. Se ejecuta el archivo .jar haciendo doble click como si fuera un ejecutable. Emergerá la ventana principal del programa (Figura 41).

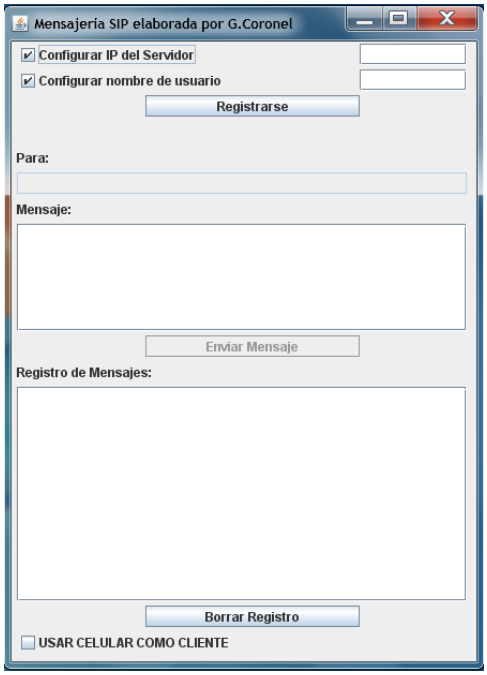


Figura 41. Ventana principal de la aplicación J2SE.

La interfaz está conformada por cuatro campos editables de texto, tres botones, tres CheckBox y un área de texto no editable (área de registro). Los campos editables son para ingresar información del servidor, del usuario local, del usuario de destino y el mensaje. El área de registro muestra todos los mensajes que la aplicación envía y recibe. También se muestran mensajes de error si la aplicación J2SE recibe un código de respuesta desconocido. Los CheckBox superiores (CheckBoxs 1 y 2) activan o desactivan la escritura en los campos que tienen a su lado. El CheckBox inferior (CheckBox 3) es utilizado para dejar de usar la aplicación como cliente SIP y transformarla en intermediaria entre el celular y el servidor. Los botones sirven para registrarse (o desconectarse), enviar mensajes o limpiar el área de registro. Los

números y nombres asignados a cada uno de esos componentes pueden verse en la Figura 42.

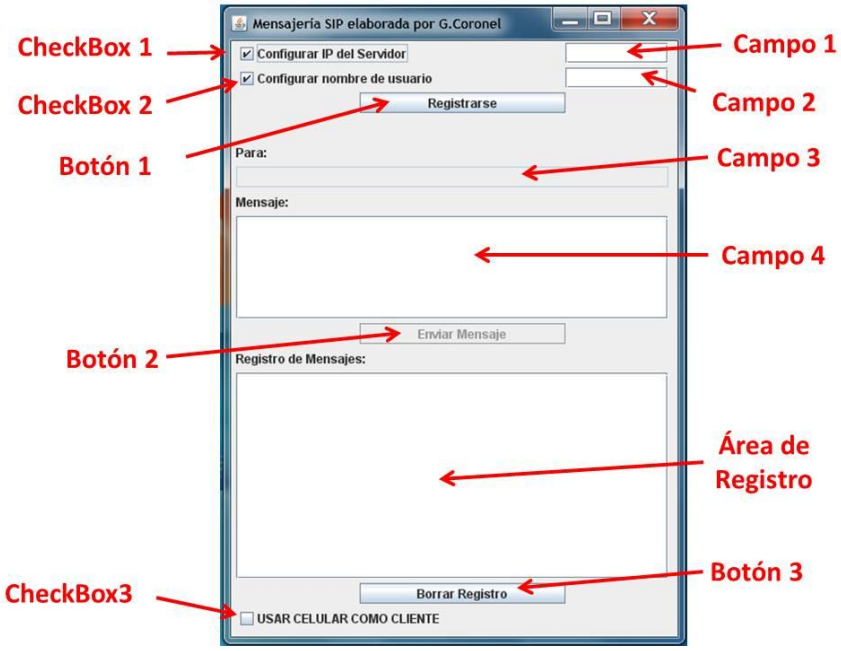


Figura 42. Componentes de la interfaz gráfica de la aplicación J2SE.

2. Se coloca en el campo 1 la dirección IP del servidor SIP y en el campo 2 el nombre de usuario.
3. Se hace click en el botón 1 etiquetado con la palabra "Registrarse".

   Al hacer click en el botón 1 se deshabilitan los CheckBox 1 y 2, y los campos 1 y 2. Si alguno de los campos, o ambos, están vacíos se muestra un mensaje de error como el de la Figura 43 y se permite la escritura en el campo vacío.

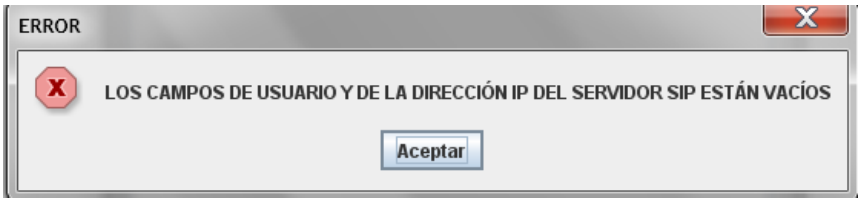


Figura 43. Mensaje de error por campos vacíos.

Si los datos ingresados son correctos, en la ventana aparecerá una etiqueta con la frase "REGISTRADO COMO [NÚMERO DE EXTENSIÓN DEL USUARIO] AL SERVIDOR EN [DIRECCIÓN IP DEL SERVIDOR

SIP]" como lo muestra la Figura 44. De igual forma, en el área de registro se podrán observar los mensajes SIP de REGISTER que intercambiaron la aplicación y el servidor. También se habilitan los campos 3 y 4, y el botón 2. El botón 1 ahora es etiquetado con la palabra "Desconectar".

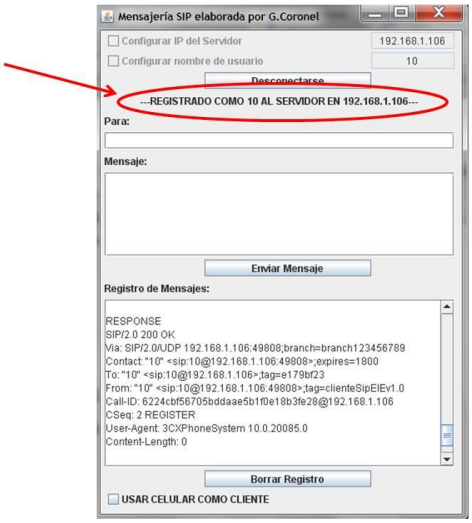


Figura 44. Conexión exitosa al servidor SIP.

Si la IP del servidor SIP es colocada de manera incorrecta, la aplicación enviará los mensajes SIP para registrarse. Sin embargo, no se obtendrá respuesta de parte del servidor. Al hacer click en el botón 1, en el área de registro aparecerá un mensaje como el siguiente:

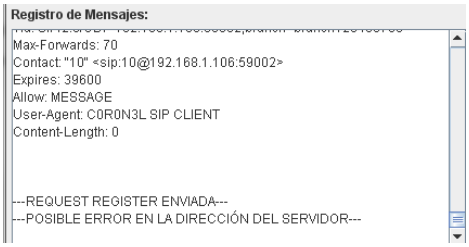


Figura 45. Error en la conexión con el servidor SIP.

El mensaje muestra la frase "POSIBLE ERROR EN LA DIRECCIÓN DEL SERVIDOR" ya que podría también estarse dando el caso de que la conexión WiFi® sea la que impida la comunicación entre la aplicación y el servidor. Se activa entonces el CheckBox 1 que a su vez habilita al campo 1 para modificar la IP del servidor.

Si el usuario introducido es incorrecto, el servidor envía una respuesta 404 Not Found a la aplicación. Se habilita entonces el CheckBox 2 que a su vez habilita al campo 2 para modificar el nombre de usuario. En el área de registro se muestra:

Figura 46. Error en el usuario.

4. En el campo 3 se escribe el número de extensión del usuario de destino, y en el campo 4 el mensaje.

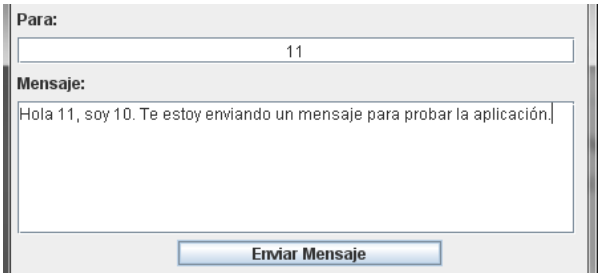


Figura 47. Escribiendo un mensaje.

5. Se hace click en el botón 2 etiquetado como “Enviar Mensaje”.

   Al hacer click en enviar, el mensaje escrito en el campo 4, será enviado a la extensión de usuario colocada en el campo 3. Si alguno de los campos está vacío se muestra un mensaje de error. Al igual que antes los mensajes SIP enviados y recibidos se muestran en el área de registro.

   El área de registro de quien envía se observará como lo muestra la Figura 48-(a), mientras que el área de registro de quien recibe se observará como lo muestra la Figura 48-(b).

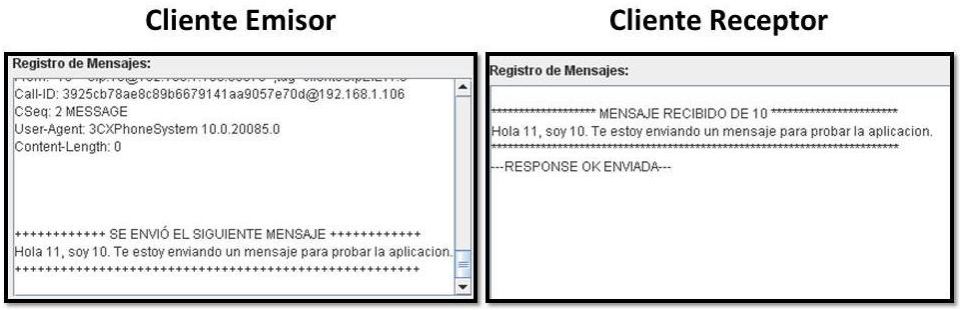


Figura 48. Área de registro de: (a) quien envía un mensaje (b) quien recibe un mensaje

6. Para desconectarse, o se hace click en el botón 2, o se cierra la ventana de la aplicación. En este último caso, de igual forma se envían mensajes SIP para desconectarse del servidor.
7. Bajo los pasos ya explicados es posible establecer una conversación de mensajería instantánea entre computadoras. Para poder hacerlo desde el celular, en primer lugar se debe marcar el CheckBox 3. Aparecerá una ventana como la mostrada en la Figura 49 que indica la dirección IP de la computadora. Esta ventana no debe cerrarse, o de lo contrario no podrá establecerse la comunicación desde el teléfono.

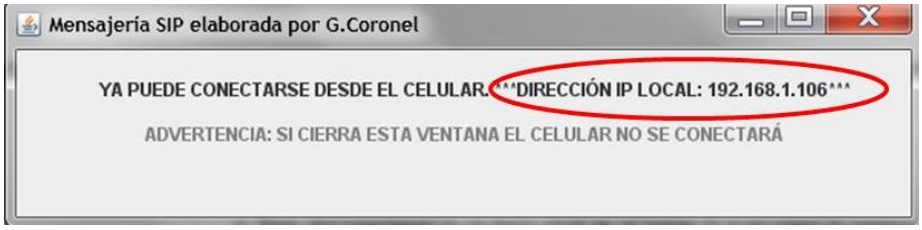


Figura 49. Interfáz gráfica de la aplicación J2SE actuando como intermediaria.

- Aplicación J2ME

Si es necesario instalar la aplicación, estos son los pasos a seguir:

1. Tener los archivos .jad y .jar de la aplicación en una misma carpeta y hacer click sobre el archivo .jar. Se mostrará en el teléfono la siguiente pantalla:

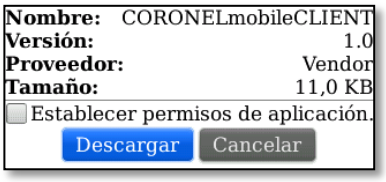


Figura 50. Ejecución del archivo .jar.

2. Al hacer click en Descargar se mostrará una advertencia como se muestra en la Figura 51. Debe seleccionarse Sí.

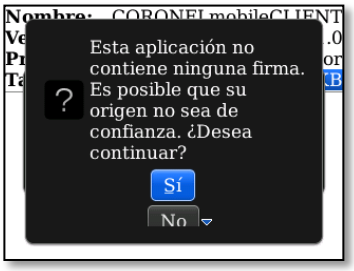


Figura 51. Advertencia de falta de firma.

El teléfono muestra esta advertencia ya que el MIDlet fue programado sin firma. Una firma es un archivo que se añade al entorno con el que se escribe un programa y sirve para certificar a un programador. Para adquirir ese archivo es necesario inscribirse y pagar una cierta cantidad de dólares. La falta de firma no afecta el funcionamiento de la aplicación.

3. Una vez seleccionado Sí, empezará la instalación del MIDlet (Figura 52). Al finalizar, la aplicación estará disponible en la carpeta de Descargas como lo muestra la Figura 53.

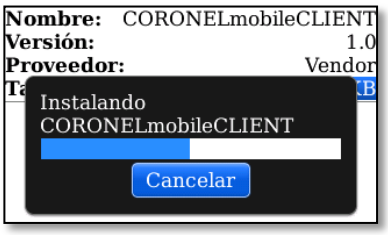


Figura 52. Instalación del MIDlet.

Figura 53. MIDlet en la carpeta de Descargas.

Ya instalada la aplicación pueden seguirse los siguientes pasos:

1. Se ejecuta la aplicación haciendo click en el ícono de la misma (ver Figura 53). Aparecerá en pantalla la interfaz gráfica programada en el archivo RegistroFORM.java (Figura 54). En esa interfaz se debe colocar la dirección IP de la computadora que posee la aplicación J2SE (la dirección IP puede ser vista en la aplicación J2SE como lo señala la Figura 49).

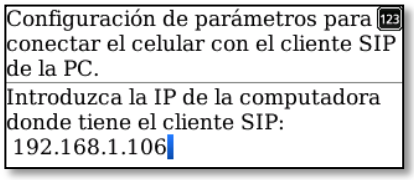


Figura 54. Configuración de la IP del cliente SIP en el RegistroFORM.

2. Se abre el menú de opciones (en el BlackBerry® es la tecla con el símbolo de dicha marca) como lo muestra la Figura 55.

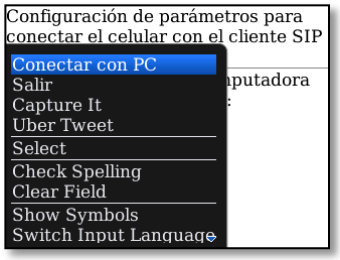


Figura 55. Opciones del RegistroFORM.

Tal menú permite la conexión con la computadora haciendo click en "Conectar con PC" o salir de la aplicación seleccionando la opción "Salir". Debe seleccionarse la opción "Conectar con PC".

3. Al seleccionar la opción "Conectar con PC" saldrá de nuevo una advertencia acerca de la falta de firma del MIDlet como lo muestra la Figura 56.

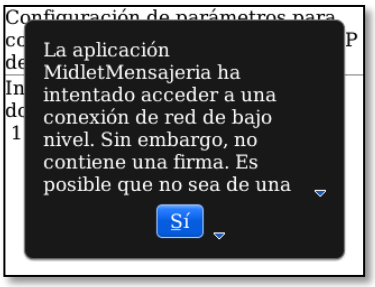


Figura 56. Advertencia al conectarse por falta de firma.

De nuevo debe omitirse esta advertencia y seleccionar la opción Sí.

El teléfono enviará un mensaje mediante sockets a la computadora solicitando el estado del cliente SIP. El cliente puede estar en dos estados: conectado, si el CheckBox 3 de la aplicación J2SE fue marcado después de estar registrado con el servidor SIP, o desconectado.

Si la IP es errónea o si hay problemas con la conexión se mostrará un error de TimeOut.

En el caso que el estado de la aplicación J2SE sea "desconectado" aparecerá en pantalla la interfaz gráfica del archivo RegistroFORM2.java (Figura 57). En ella se debe ingresar la dirección IP del servidor SIP y el nombre de usuario.

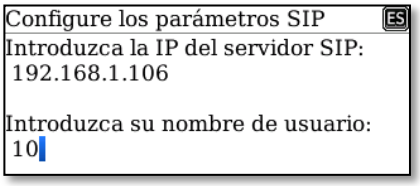


Figura 57. Configuración de la IP del servidor SIP y del nombre de usuario en el RegistroFORM2.

Al extender el menú de opciones se podrá conectar con el servidor SIP haciendo click en “Conectar al Servidor SIP” o salir de la aplicación seleccionando la opción “Salir”. Debe seleccionarse la opción “Conectar al Servidor SIP”.

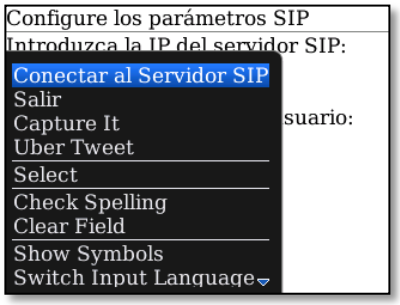


Figura 58. Opciones del RegistroFORM2.

Al seleccionar esa opción el celular enviará a la computadora la orden de enviar mensajes SIP al servidor. Si la aplicación de la computadora recibió exitosamente la petición, se mostrará en pantalla en siguiente mensaje:

Figura 59. Petición de conexión al servidor SIP exitosa.

Se mostrará entonces la interfaz gráfica del archivo MenuLIST.java (Figura 60) y cuando la conexión con el servidor SIP sea exitosa se mostrará el mensaje “Conectado al Servidor SIP” (Figura 61).

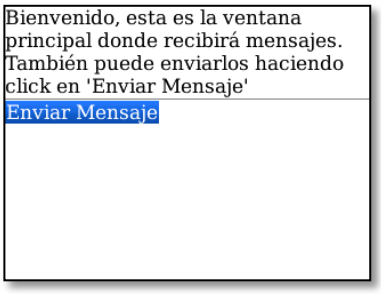


Figura 60. Interfaz gráfica de la clase MenuLIST.

Figura 61. Conexión exitosa al servidor SIP.

Si no fue exitosa porque la IP del servidor no es correcta no se mostrará ningún mensaje. Se debe estar atento entonces al ingresar ese parámetro. Si en cambio, la conexión no fue exitosa por un error en el campo de usuario, se mostrará el siguiente mensaje:

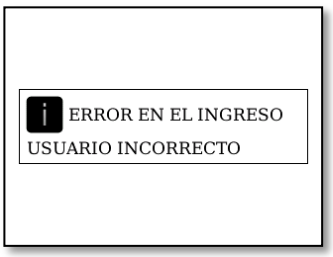


Figura 62. Error en el ingreso por usuario incorrecto.

Fue explicado el caso en que la aplicación del celular se comunicaba con la aplicación J2SE, y ésta estaba en estado de desconexión. Si en cambio la aplicación J2SE está conectada al servidor SIP y por lo tanto está en el estado “conectado”, se pasará directamente a la interfaz gráfica MenuLIST (Figura 60).

4. En el MenuLIST cualquier mensaje que se reciba será mostrado en pantalla como en la siguiente figura. Cada mensaje dice “RECIBIDO de” y luego el nombre del usuario que lo envió. En la siguiente línea se muestra el mensaje. En la Figura 63 el contenido del mensaje era “Probando” y fue enviado por el usuario de nombre “10”.

Figura 63. Mensaje recibido.

Para enviar mensajes estando en el MenuLIST basta con hacer click o extender el menú (Figura 64) y seleccionar la opción “Enviar Mensaje” para ir a la pantalla de envío.

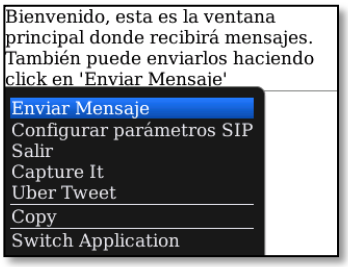


Figura 64. Opciones de MenuLIST.

En el menú de opciones también puede seleccionarse “Configurar parámetros SIP” para volver a la pantalla de RegistroFORM2 y cambiar la dirección IP del servidor SIP o el nombre de usuario. En este caso se envía un mensaje a la aplicación J2SE para desconectarse del servidor SIP. Cuando la petición es recibida por la computadora de manera exitosa, ésta envía una notificación de recepción (Figura 65) al celular.

Figura 65. Petición de desconexión al servidor SIP exitosa.

Al completarse la desconexión del servidor SIP, se muestra en pantalla el siguiente mensaje:

Figura 66. Desconexión del servidor SIP exitosa.

Cuando se selecciona la opción “Salir” la aplicación se cierra y envía un mensaje a la aplicación J2SE para desconectarse del celular.

5. La pantalla de envío (EnvioFORM) se muestra a continuación:

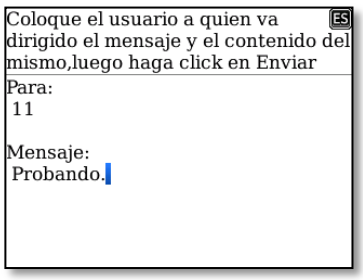


Figura 67. Pantalla para enviar mensajes.

En ella se escribe el nombre del usuario de destino debajo de la etiqueta “Para:” y el mensaje debajo de la etiqueta “Mensaje:”. Posteriormente, para enviar un mensaje se abre el menú de opciones (Figura 68) y se selecciona “Enviar”. Las otras opciones de dicho menú son: “Atrás” para volver a la pantalla MenuLIST y “Salir” para cerrar la aplicación. Al igual

que antes, si se selecciona “Salir” se envía un mensaje a la aplicación de la computadora para desconectarse del servidor SIP.

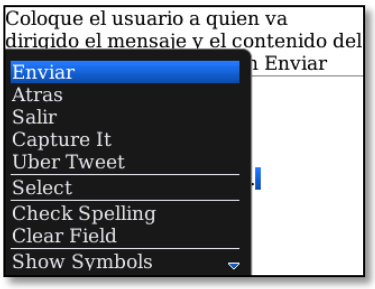


Figura 68. Menú de opciones de la pantalla EnvioFORM.

Cuando se envíe un mensaje del celular y éste sea recibido correctamente por la aplicación de la computadora, se mostrará en el teléfono la siguiente pantalla:

Figura 69. Envío exitoso de un mensaje desde el celular hasta la aplicación J2SE.

Debe mencionarse que mientras la aplicación de la computadora esté ejecutándose como intermediaria, la aplicación del celular puede ser cerrada y abierta cuantas veces se desee. Sin embargo, cada vez que sea abierta deberán introducirse de nuevo todos los parámetros ya mencionados. En algunas de las pruebas realizadas donde la aplicación del celular se conectaba y se desconectaba repetidas veces, ocurrió que la aplicación J2SE no liberaba el puerto para recibir sockets y por lo tanto el celular no podía conectarse. En estos casos se debió en la computadora cerrar la aplicación, entrar en el “Administrador de tareas” y finalizar el proceso “javaw.exe”. Luego se iniciaban de nuevo ambas aplicaciones y funcionaban sin ningún problema. Una posible opción es no cerrar la

aplicación mientras se desee permanecer conectado, sin embargo esto podría ocasionar que se descargue la batería del celular o que el mismo se ponga lento al intentar utilizar otra aplicación.

También debe decirse que las capturas fueron realizadas en un BlackBerry® Curve 8520 con sistema operativo 5.0.0.1067. Si se usa otro dispositivo, puede darse que las pantallas varíen con respecto a las mostradas. Sin embargo, el comportamiento del programa debe mantenerse.

Para la evaluación de los programas realizados, al servidor SIP le fue asignada de manera fija la dirección IP 192.168.1.106. El estudio realizado bajo las situaciones antes expuestas arrojó los siguientes resultados:

- Situación 1: Registro de la aplicación J2SE con el servidor.

  En esta situación la computadora que tenía la aplicación J2SE estaba asociada a la dirección IP 192.168.1.103. El flujo de mensajes SIP bajo esta situación se muestra en la Figura 70. La aplicación J2SE envía una petición con el método REGISTER al servidor quien le responde con un mensaje 407 Proxy Authentication Required. Entonces la aplicación J2SE reenvía la petición añadiéndole la información de sus credenciales.

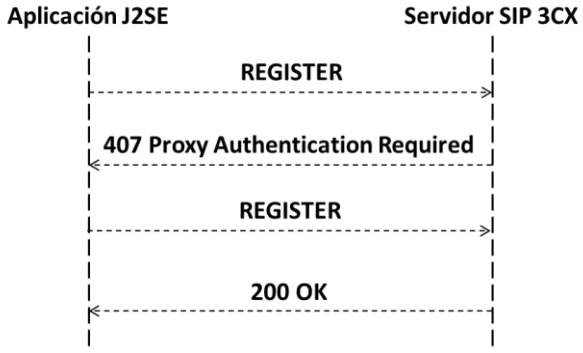


Figura 70. Flujo de mensajes SIP en el registro de la aplicación J2SE con el servidor.

En la Figura 71 se muestra cómo está conformado el mensaje de petición REGISTER de la aplicación creada. Está compuesto por la línea de inicio, 6

cabeceras obligatorias (To, From, CSeq, Call-ID, Max-Forwards y Via) y 5 cabeceras opcionales (Contact, Expires, Allow, User-Agent y Content-Length).

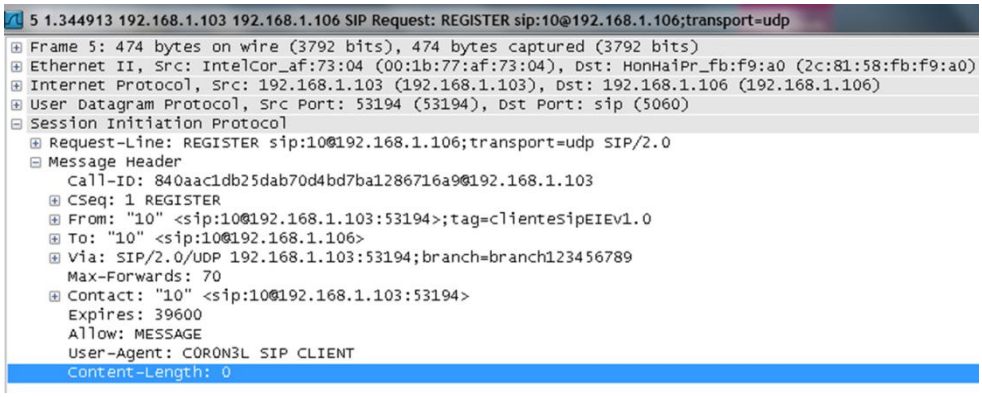


Figura 71. Mensaje REGISTER para conectarse al servidor.

A pesar que la cabecera Contact no es obligatoria según la RFC 3261[15], cuando era omitida producía un error en el servidor. El servidor anunciaba su falla con una respuesta 503 (Servicio no disponible). Como fue mencionado, una respuesta del grupo 5xx aparece cuando existe una falla en el servidor que le impide procesar los mensajes. Si se implementa un servidor con código abierto cuyos parámetros puedan ser configurados, la cabecera Contact debería poder omitirse sin problemas. La cabecera Expires fue añadida de manera tal que una vez que un cliente se registre a un servidor no necesite volver a conectarse por 11 horas. Se colocó ese valor para que por ejemplo un profesor que trabaje en una institución de 7 am a 6 pm pueda mantenerse conectado y enviar los mensajes directamente sin tener que registrarse nuevamente en dicho lapso. Se debe mencionar que el servidor 3CX® acepta un tiempo máximo de 1800 segundos, es decir, que a los 30 minutos de que un cliente se conecte necesitará registrarse de nuevo para enviar un mensaje. Este problema también se evitaría al utilizar un servidor SIP configurable. La cabecera Allow permite a otros clientes (propietarios o no) conocer que la aplicación J2SE está configurada para recibir de ellos únicamente mensajes con el método MESSAGE. La cabecera User-Agent

identifica a la aplicación creada bajo el nombre de “C0R0N3L SIP CLIENT”. Por último la cabecera Content-Length indica que el mensaje no tiene cuerpo.

El mensaje de respuesta donde el servidor solicita la autenticación de la aplicación J2SE se muestra a continuación en la Figura 72. El servidor indica los parámetros nonce, realm y algorithm que debe utilizar el esquema de autenticación Digest para generar una respuesta al reto. Las cabeceras Via, From, Call-ID, y CSeq son exactamente iguales a las de la petición original. En la cabecera To el servidor añade su tag.

Figura 72. Respuesta al REGISTER solicitando a la aplicación J2SE que se autentique.

La nueva petición de REGISTER es igual a la primera a excepción de la cabecera CSeq que incrementa su número de secuencia y de la cabecera Call-ID cuyo valor cambia por completo. También se añade la cabecera Proxy-Authorization que envía un código de respuesta al reto solicitado por el servidor (Figura 73).

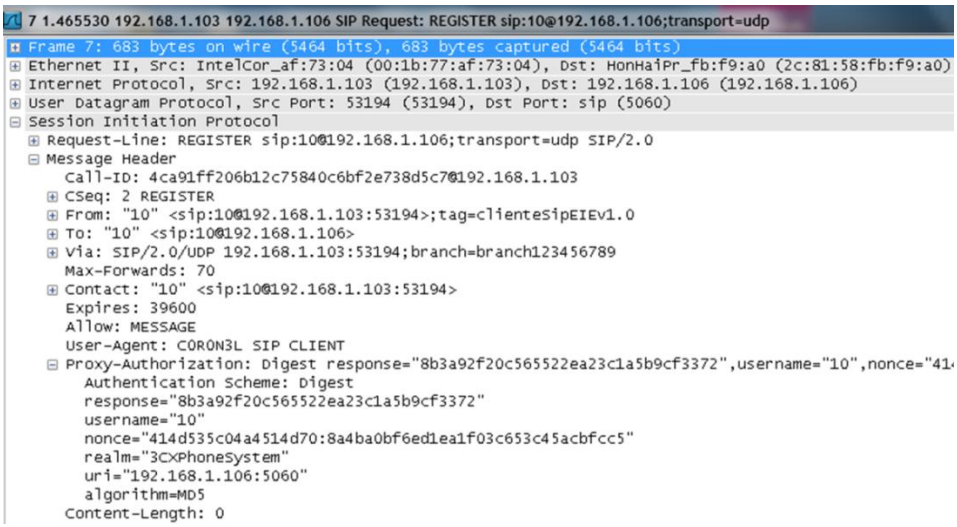


Figura 73. Mensaje REGISTER con datos de autenticación.

Cuando el cliente envía de manera correcta los datos de autenticación, el servidor le responde con un mensaje OK (Figura 74). En el mensaje OK puede verse que el servidor 3CX® modifica el valor de la cabecera Expires a 1800. Esto lo realiza añadiendo el parámetro "expires" en la cabecera Contact. El mensaje OK mantiene los valores de Call-ID y CSeq de la petición REGISTER que contenía los datos de autenticación.

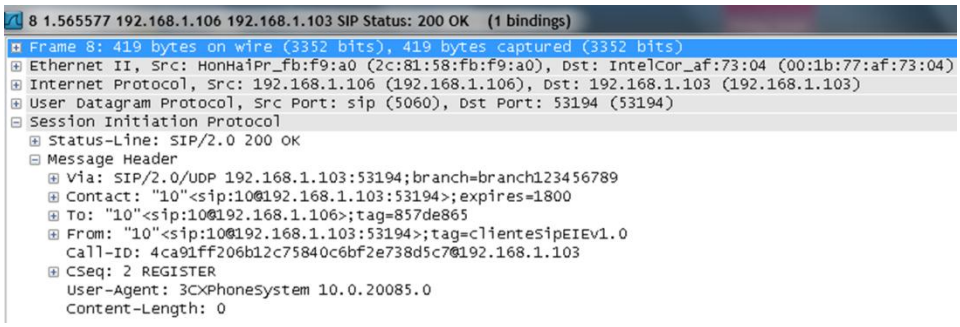


Figura 74. Mensaje OK a la solicitud de registro.

- Situación 2: Desconexión de la aplicación J2SE con el servidor.

  El protocolo SIP no posee ningún método directo para desconectarse de un servidor. Sin embargo, el método REGISTER puede ser utilizado para lograr dicho propósito. El flujo de mensajes SIP será entonces de la misma forma que el de la situación 1 (Figura 75) La desconexión se logra colocando el valor de la cabecera Expires igual a cero.

Figura 75. Petición SIP para desconectarse del servidor.

Los mensajes de respuestas que envía el servidor a la aplicación J2SE (407 Proxy Authentication Required y 200 OK) serán iguales a los mostrados en la Figura 72 y Figura 74, respectivamente.

El mensaje REGISTER con las credenciales para autenticarse en este caso, también tendrá la misma estructura que el mensaje REGISTER mostrado en la Figura 73 a excepción de la cabecera Expires que como se dijo tendrá un valor igual a cero.

Al igual que en la situación 1, la computadora con la arquitectura J2SE tenía asociada la dirección IP 192.168.1.103.

- Situación 3: Intercambio de mensajes entre dos computadoras con la aplicación J2SE.

  El estudio fue realizado luego de que las aplicaciones de ambas computadoras estuviesen conectadas al servidor. A la computadora 1 se le asignó la dirección IP 192.168.1.103 y a la computadora 2 la IP 192.168.1.107. El flujo de mensajes bajo esta situación se presenta como lo muestra la Figura 76.

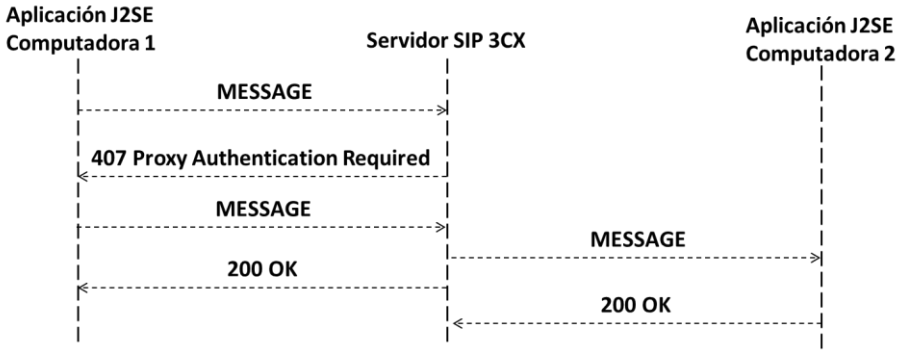


Figura 76. Flujo de mensajes SIP cuando se envía una petición MESSAGE de una computadora a otra.

La computadora 1 envía un mensaje con el método MESSAGE al servidor SIP quien le solicita autenticarse. Luego de generar el código de autenticación, la computadora 1 envía de nuevo la petición MESSAGE (añadiendo la cabecera Proxy-Authorization y cambiando las cabeceras Call-ID y CSeq al igual que con las peticiones REGISTER). Si la nueva petición es correcta, el servidor reenvía la solicitud a la aplicación de la computadora 2. Inmediatamente después de enviarla, le comunica a la aplicación de la computadora 1 que el mensaje fue correctamente reenviado a través de un mensaje 200 OK. Finalmente, la aplicación de la computadora 2 al recibir el mensaje, también responde con un mensaje 200 OK. Esta respuesta OK es enviada al servidor y no a la computadora 1. A continuación se muestra la estructura de una petición MESSAGE:

```
128 88.683236 192.168.1.103 192.168.1.106 SIP Request: MESSAGE sip:11@192.168.1.106;transport=udp (text/plain)
Frame 128: 479 bytes on wire (3832 bits), 479 bytes captured (3832 bits)
Ethernet II, Src: IntelCor_af:73:04 (00:1b:77:af:73:04), Dst: HonHaiPr_fb:f9:a0 (2c:81:58:fb:f9:a0)
Internet Protocol, Src: 192.168.1.103 (192.168.1.103), Dst: 192.168.1.106 (192.168.1.106)
User Datagram Protocol, Src Port: 64170 (64170), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: MESSAGE sip:11@192.168.1.106;transport=udp SIP/2.0
  Message Header
    Call-ID: 31ada87a0e911cc3e090bc979d394981@192.168.1.103
    CSeq: 1 MESSAGE
    From: "10" <sip:10@192.168.1.103:64170>;tag=clienteSipEIEv1.0
    To: "11" <sip:11@192.168.1.106>
    Via: SIP/2.0/UDP 192.168.1.103:64170;branch=branch1
    Max-Forwards: 70
    Content-Type: text/plain
    Allow: MESSAGE
    User-Agent: C0R0N3L SIP CLIENT
    Content-Length: 48
  Message Body
    Line-based text data: text/plain
      Hola 11, te estoy enviando un mensaje de prueba.
```

Figura 77. Estructura de una petición MESSAGE.

Está constituida por las 6 cabeceras obligatorias (To, From, CSeq, Call-ID, Max-Forwards y Via) y a diferencia de una petición REGISTER tiene sólo 4 cabeceras opcionales (Content-Type, Allow, User-Agent y Content-Length). Adicionalmente, el mensaje tiene cuerpo.

La cabecera Contact no se coloca en este tipo de peticiones ya que la RFC 3428 así lo especifica. La cabecera Expires no es necesaria ya que no se espera que el mensaje quede almacenado en alguna parte de la arquitectura. Ella indica la duración de una sesión. La cabecera Allow y User-Agent cumplen la misma función que tenían cuando fueron enviadas en una petición REGISTER. La cabecera Content-Length indica cuántos caracteres tiene el cuerpo del mensaje. Content-Type indica el formato del contenido que se está enviando. En este caso se observa que el tipo de contenido es texto simple (text/plain). En el cuerpo de la petición se envía el mensaje. Si se cuentan los caracteres de dicho mensaje ("Hola 11, te estoy enviando un mensaje de prueba.") puede verificarse que son 48 como lo indica la cabecera Content-Type. Los espacios son tomados como caracteres.

Las respuestas 407 y 200, mantienen las formas ya explicadas en las peticiones REGISTER. La única cabecera que varía es la de CSeq ya que un método es distinto el que se está utilizando.

En la Figura 78 se muestra la petición MESSAGE con las credenciales para autenticarse y en la Figura 79 se observa la estructura del mensaje que el servidor la reenvía a la computadora 2. Hay que destacar, que cuando el servidor realiza el reenvío, actúa como un cliente SIP que genera la petición por primera vez (reinicia la secuencia de CSeq y añade las cabeceras que su configuración establezca). Sin embargo, para que la computadora 2 reconozca el verdadero origen del mensaje, mantiene la cabecera From de la petición realizada por la computadora 1.

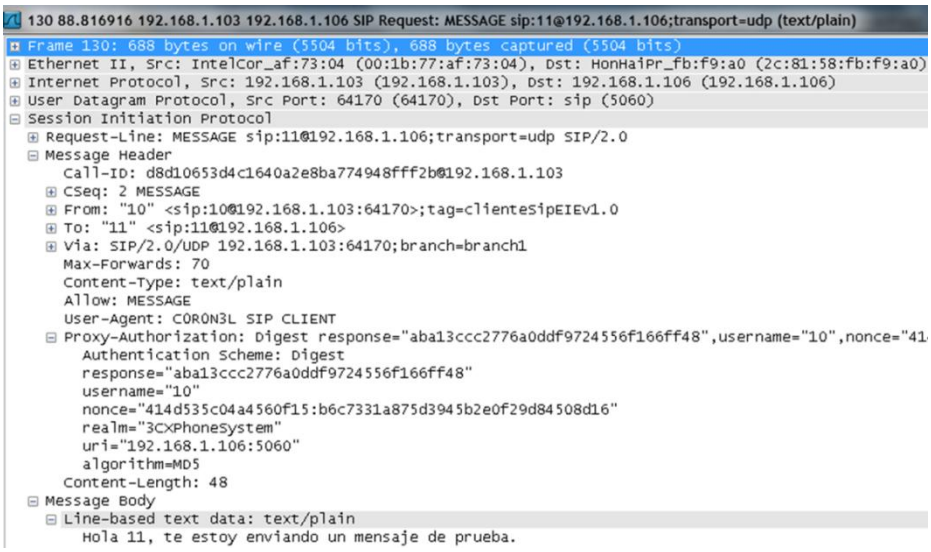


Figura 78. Petición MESSAGE con credenciales para autenticarse.

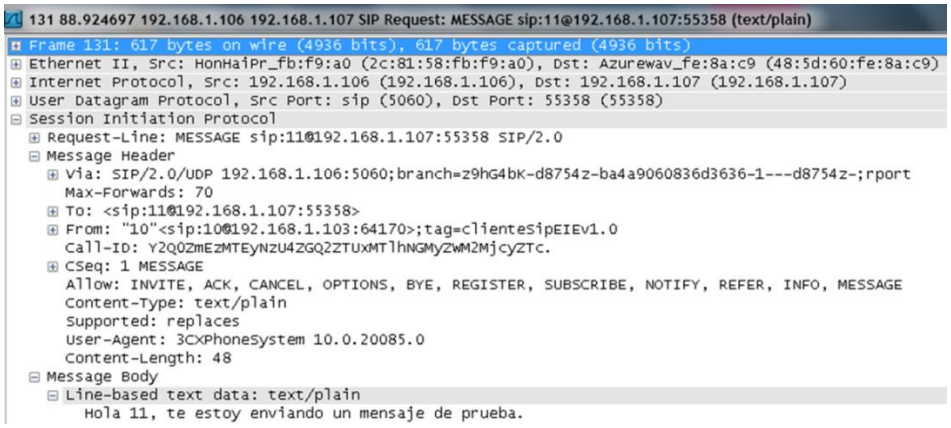


Figura 79. Reenrutamiento de la petición MESSAGE.

En la Figura 80, puede verse la estructura de la respuesta 200 OK que genera la aplicación desarrollada. Ella copia de la petición MESSAGE del servidor las cabeceras Via, From, Call-ID y CSeq. A la cabecera To le añade el tag. El tag que agrega la aplicación es el “888” seleccionado sin ninguna razón en particular.

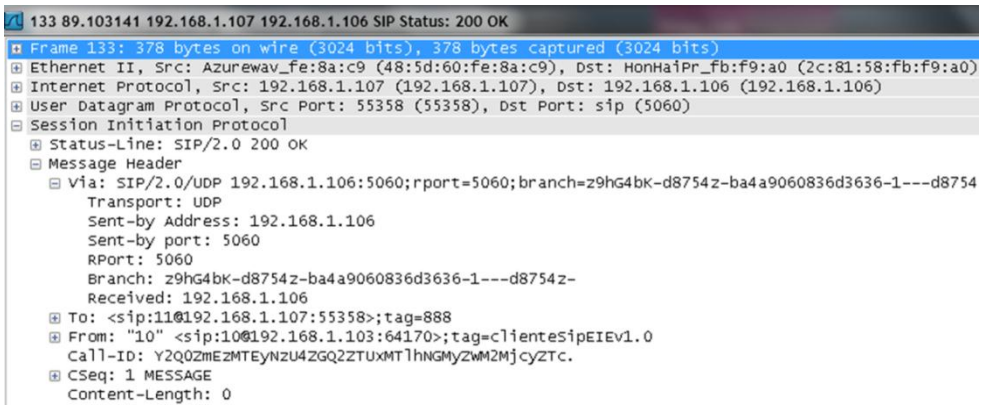


Figura 80. Respuesta a la petición MESSAGE del servidor.

- Situación 4: Intento de registro con un nombre de usuario no existente.

  En este caso la aplicación cumple con el mismo procedimiento que una petición REGISTER normal, pero recibe del servidor una respuesta de código 404 (Usuario no encontrado). El flujo de mensajes SIP que ocurre en esta situación puede verse en la Figura 81 y la estructura del mensaje de respuesta 404 en la Figura 82.

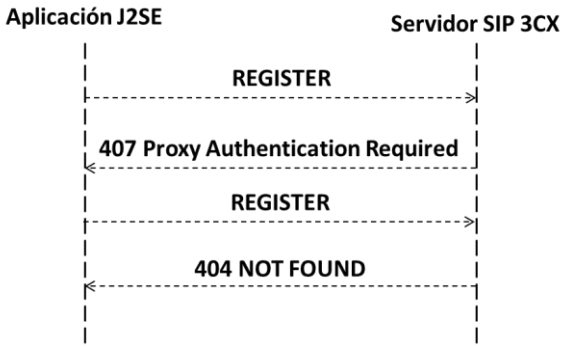


Figura 81. Flujo de mensajes SIP cuando el usuario es incorrecto.

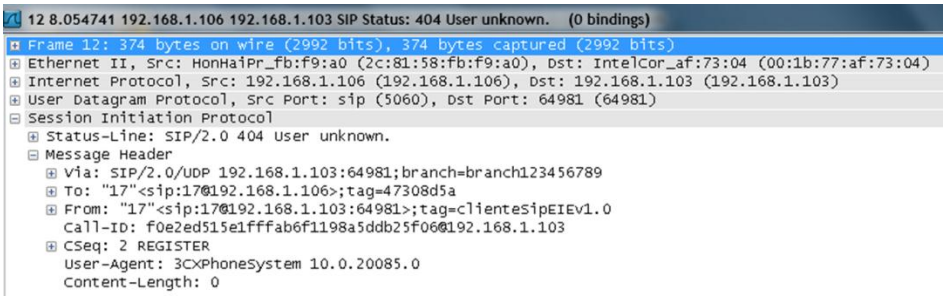


Figura 82. Estructura de un mensaje de respuesta 404.

- Situación 5: Conexión de la aplicación J2SE con la aplicación J2ME.

  Como la conexión se establece mediante Sockets, y éstos utilizan como protocolo de transporte a TCP, la captura arroja cientos de paquetes. La mayoría de esos paquetes son utilizados para control y mantener el servicio orientado a la conexión. Se captó el tráfico desde una computadora distinta a la que se conectaba el celular y los datos internos de cada paquete TCP no pudieron ser descifrados. Puede decirse entonces que la comunicación mediante Sockets se realiza de manera segura aunque no haya sido diseñado para ello.

  Sin embargo, cuando el tráfico de datos (con el analizador de protocolos Wireshark®) fue colocado en la misma computadora donde el celular se conectaba fue posible observar el mensaje enviado. Este mensaje captado se muestra en la Figura 83.

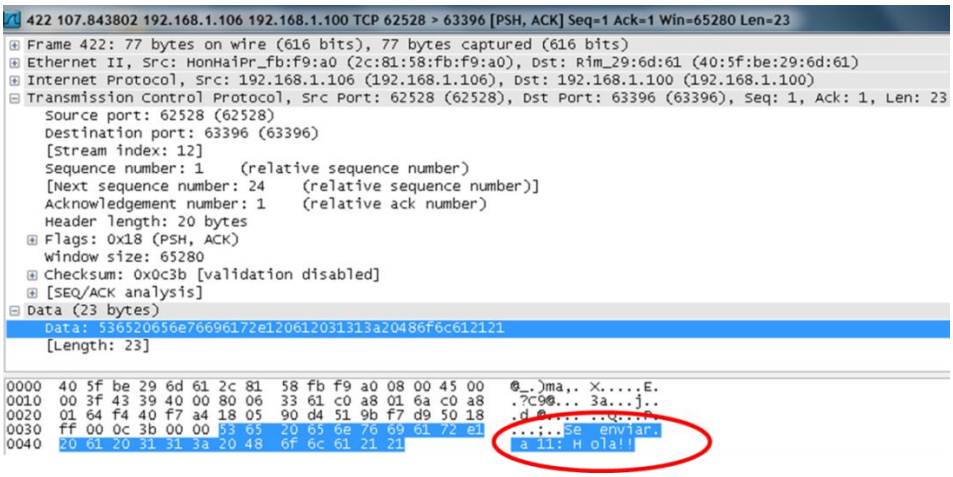


Figura 83. Paquete TCP con el mensaje enviado.

En todas las situaciones anteriormente explicadas puede observarse que toda petición que generaba la aplicación J2SE, requería ser autenticada por el servidor 3CX®, a pesar que en el mismo los usuarios estaban configurados para ingresar sin contraseña. Si se hiciera una implementación sin contraseña en un servidor de código abierto que sea configurable, no debería necesitarse el envío de esa nueva petición con credenciales. Esto beneficiaría a la arquitectura ya que se dejarían de involucrar

dos mensajes por cada transacción que se realice, lo que significaría una reducción en el tiempo empleado en cada de una de ellas.

Cuando se realizó el estudio de compatibilidad de la aplicación J2SE con el cliente propietario X-Lite® pudo comprobarse la compatibilidad de ambas. En la siguiente figura puede verse la conversación establecida entre ambos clientes:

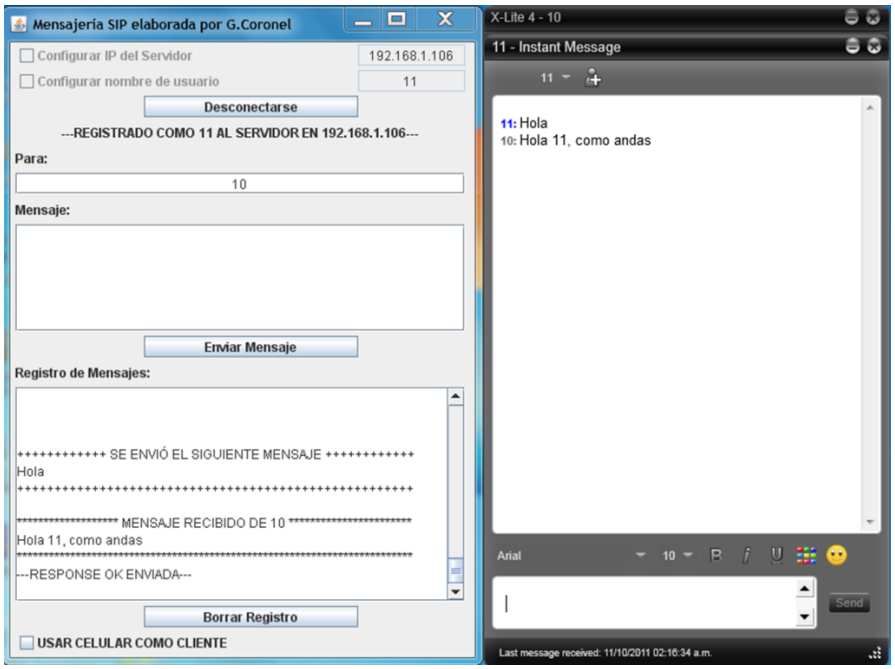


Figura 84. Compatibilidad de la aplicación J2SE con el cliente X-Lite® 4.

La aplicación J2ME probada en un BlackBerry® Pearl 8120 de software 4.0, fue 100% compatible con éste y funcionó exactamente igual que con el BlackBerry® Curve 8520. Cuando se instaló y ejecutó sobre el Nokia N78, el envío de mensajes hacia la computadora fue correcto pero al recibir respuesta de ella se presentaban problemas. Lo que ocasionó el problema es que la aplicación fue programada para mostrar una ventana que en J2ME es conocida como “Alert” y por lo que se podía interpretar del error obtenido en el equipo Nokia N78, éste no es capaz de soportar ese tipo de ventana. Para evitar ello es recomendable que para una implementación futura se modifique el modo en el que el teléfono muestra el mensaje recibido.

En la siguiente figura (Figura 85) puede verse el resultado final de la arquitectura, y el verdadero propósito con el que fue diseñada, una conversación entre dos equipos celulares. En el lado izquierdo de ella, se observa la captura de la pantalla del celular que emitió el mensaje (en este caso perteneciente a un usuario cuya extensión es la número 10). La misma dice "MENSAJE RECIBIDO" porque la aplicación J2ME ha recibido un mensaje de la aplicación J2SE que le comunica que ha recibido su petición para enviar al usuario "11" el mensaje "Probando". Por otro lado, en el lado derecho de la figura se observa la pantalla del teléfono receptor (en este caso perteneciente al usuario cuya extensión es la número 11), cuando ha recibido el mensaje "Probando" que envió el usuario "10".

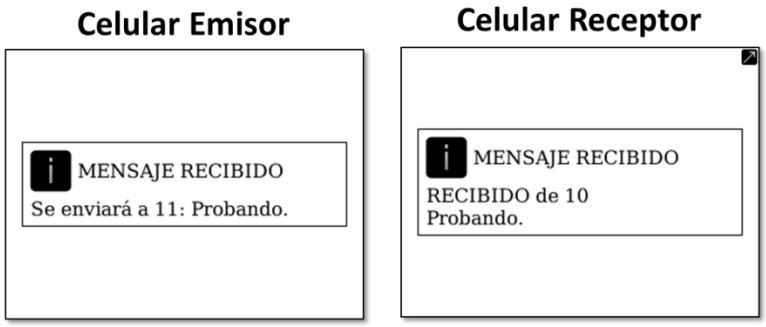


Figura 85. Conversación entre dos celulares.

La arquitectura diseñada puede ser implementada en la Escuela de Ingeniería Eléctrica de la Faculta de Ingeniería. Traería como ventaja evitar la latencia que normalmente introducen la gran cantidad de equipos por los que deben pasar los datos. Bajo esta arquitectura los datos pasarían en el peor de los casos por unos pocos puntos de acceso. De igual forma si sucediese un problema con el servidor no habría que esperar por la solución del mismo, teniendo a alumnos capacitados se podría solucionar el problema casi de inmediato. Algunos de los usos que podría tener la arquitectura en la escuela serían:

- Consulta de dudas de un estudiante con el profesor de la asignatura.
- Solicitud de notas de exámenes o asginaciones de un estudiante con el profesor de la asignatura.
- Información de horarios de revisión de exámenes por parte de un profesor.

- Notificación de cambio de salón de la clase del día por parte de un profesor.
- Notificación de asambleas por parte de la Dirección de la Escuela o de la Jefatura del Departamento de Comunicaciones, o de algún otro departamento tras la implementación de las clases de multicast.

CAPÍTULO IV

CONCLUSIONES

- La implementación del protocolo SIP en celulares trae consigo problemas de portabilidad. Es por ello, que la solución fue implementar al protocolo SIP en una aplicación para computadoras, que permitiera a ésta actuar como intermediaria entre un celular y un servidor SIP.
- La arquitectura de mensajería instantánea para celulares con soporte para WiFi® basada en código abierto con protocolo SIP quedó constituida por dos aplicaciones, una escrita en leguaje J2SE para computadoras, que sirve como intermediaria entre un celular y un servidor SIP, y otra escrita en J2ME que ejecutada sobre celulares, permite la comunicación entre ellos y la aplicación J2SE.
- Los mensajes enviados entre dos teléfonos celulares a través de la arquitectura fueron exitosamente recibidos. Cada mensaje que se envía desde un celular realiza el siguiente recorrido: desde la aplicación J2ME pasa a la aplicación J2SE de su computadora asociada (mediante sockets a través de la red WiFi®). Esta aplicación J2SE lo recibe, lo introduce en un mensaje SIP y envía este último al servidor SIP, quien reenvía el mensaje a la aplicación J2SE de destino. Ésta, extrae el contenido del mismo y determina cuál es el verdadero mensaje a entregar. Una vez determinado, lo envía mediante sockets a través de la red WiFi® a su celular asociado donde el usuario receptor puede leerlo.
- La utilización de sockets en los celulares además de aumentar la portabilidad de la aplicación J2ME, permitió tener confiabilidad en la entrega de paquetes entre el celular y la computadora a través de la red WiFi®. Sin embargo,

hace que el envío de paquetes sea más lento en comparación con una implementación que se base en UDP.

- La conexión entre una computadora (con la aplicación J2SE) y el servidor SIP puede realizarse por una red inalámbrica o alámbrica, sin que ello afecte el comportamiento del sistema.
- Los componentes de la arquitectura diseñada pueden ser todos implementados bajo código abierto (aplicación para celulares, aplicación para computadoras y servidores SIP), y bajo la estructura en la que fue concebida permite la comunicación entre celulares, entre computadoras o entre celulares y computadoras.
- La arquitectura es compatible con software propietario como fue comprobado con el cliente X-Lite®.
- Teniendo una red WiFi® que cubra toda la Escuela de Ingeniería Eléctrica de la Universidad Central de Venezuela y tras haber agregado las clases diseñadas para multicast, se puede implementar la arquitectura para que los estudiantes consulten dudas o notas, los profesores envíen notas, información de horarios de revisión, o notificaciones de cambios de salón, y por último, para que el director de escuela y los jefes de departamentos notifiquen acerca de futuras asambleas

RECOMENDACIONES

- Implementar la arquitectura planteada en este trabajo pero con un servidor SIP de código abierto que permita configurar sus parámetros.
- Complementar las aplicaciones J2SE y J2ME con las clases de multicast propuestas, para que la implementación de esta arquitectura en un futuro en la Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Central de Venezuela pueda cumplir con todas las características que se esperan (envío de notas, notificación de asambleas, etc.).
- Inicialmente, y en paralelo a la adición de las clases de Multicast, se puede trabajar con la aplicación J2SE para añadirle soporte de llamadas de voz y videoconferencia.
- Puede añadirse a la aplicación J2ME soporte para enviar y recibir archivos, por ejemplo archivos pdf, Word, etc.
- Para que la arquitectura no dependa de una computadora, pueden diseñarse a partir de sockets o datagramas las clases de Java necesarias para implementar SIP directamente en los celulares. Dado el alto grado de conocimientos que se requeriría para realizar la programación en este caso, se propone trabajar en conjunto con estudiantes o profesores de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela.

REFERENCIAS BIBLIOGRÁFICAS

[1] Tanenbaum, A. (2003). *Redes de computadoras* (4a ed.). México: Pearson Educación.

[2] Camarillo, G. (2002). *SIP Demystified*. Estados Unidos de América: McGraw-Hill Companies, Inc.

[3] Handley, M. et al. (1999). *SIP: Session Initiation Protocol (RFC2543)* [En línea]. Recuperado el 29 de noviembre de 2010, del sitio Web del Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2543.txt.pdf

[4] RFC Editor (2008). *Internet Official Protocol Standards* [En línea]. Recuperado el 6 de septiembre de 2010, del sitio Web de RFC Editor: http://www.rfc-editor.org/std/std1.txt

[5] Rosenberg, J. et al. (1999). *SIP: Session Initiation Protocol (RFC3261)* [En línea]. Recuperado el 17 de agosto de 2010, del sitio Web del Internet Engineering Task Force: http://www.ietf.org/rfc/rfc3261.txt

[6] Camarillo, G. et al. (2003). *Evaluation of transport protocols for the Session Initiation Protocol*. IEEE Network, 17 (5), 40-46. Recuperado el 27 de agosto de 2011, de la base de datos del IVIC.

[7] Janak, J. (2003). *SIP Introduction*. [En línea]. Recuperado el 2 de octubre de 2011, del sitio Web de iptel.org: http://ftp.iptel.org/pub/ser/0.8.14/doc/html/sip_introduction.html

[8] IBM WebSphere Developer Technical Journal: Session Initiation Protocol in WebSphere Application Server V6.1--Part 1. (s.f.) [En línea]. Recuperado el 15 de septiembre de 2011, del sitio Web: http://www.ibm.com/developerworks/websphere/techjournal/0606_burckart/0606_burckart.html

[9] Fielding, R. et al. (1999). *Hypertext Transfer Protocol – HTTP/1.1 (RFC2616)* [En línea]. Recuperado el 20 de septiembre de 2011, del sitio Web del Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2616.txt

[10] Franks, J. et al. (1999). *HTTP Authentication: Basic and Digest Access Authentication (RFC2617)* [En línea]. Recuperado el 20 de septiembre de 2011, del sitio Web del Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2617.txt

[11] Rosenberg, J. et al. (2002). *Session Initiation Protocol (SIP) Extension for Instant Messaging (RFC3248)* [En línea]. Recuperado el 3 de agosto de 2011, del sitio Web del Internet Engineering Task Force: http://www.ietf.org/rfc/rfc3428.txt

[12] *X-Lite welcomes you to the world of softphones!* (s.f.). [En línea]. Recuperado el 3 de agosto de 2010, del sitio Web de Counterpath: www.counterpath.com/x-lite.html

[13] *Free downloads* (s.f.). [En línea]. Recuperado el 5 de agosto de 2010, del sitio Web de SJ Labs: http://www.sjlabs.com/sjp.html

[14] *Documentation* (2011). [En línea]. Recuperado el 6 de agosto de 2010, del sitio Web de Peers: http://peers.sourceforge.net/

[15] *The Basics of the Protocol Provider Service* (s.f.). [En línea]. Recuperado el 7 de agosto de 2010, del sitio Web de Jitsi: http://www.jitsi.org/index.php/Documentation/PPBasics

[16] *3CX* Central Telefónica (s.f.). [En línea]. Recuperado el 5 de agosto de 2010, del sitio Web de 3CX: http://www.3cx.es/centralita-telefonica/3CXPhoneSystem_brochure_es.pdf

[17] *About* (s.f.). [En línea]. Recuperado el 17 de enero de 2011, del sitio Web de openSIPS: http://www.opensips.org/Main/About

[18] *Introducción* (s.f.). [En línea]. Recuperado el 10 de agosto de 2011, del sitio Web de Asterisk-ES: http://comunidad.asterisk-es.org/index.php?title=Introducci%C3%B3n

[19] Fernández, A. (2009). *Autorización de Prestaciones Médicas* [Tesis en línea]. Trabajo de Maestría, Universidad de Mendoza República Argentina. Recuperado del sitio Web de la Universidad de Mendoza: http://www.um.edu.ar/web/imagenes-contenido/UM-MTI-FernandezA.pdf

[20] Liebeherr, J. (2004). *Mastering Networks: An Internet Lab Manual* [Libro en línea]. Universidad de *Virginia*, Estados Unidos de América. Consultada el 20 de julio de 2011 en: *http://www.cs.virginia.edu/~itlab/book/pdf/Ch10_v1.pdf*

[21] Hillebrand, F. (2010). *Short message service*. Chichester, Reino Unido: John Wiley & Sons Ltd.

[22] King, C. (2009). *Advanced BlackBerry Development*. Nueva York, Estados Unidos de América: Apress.

[23] Brazee, J. (1996). *LAN Communications Standards* [En línea]. Recuperado el 12 de agosto de 2011, del sitio Web del Departamento de Ingeniería, Tecnología y Distribución Industrial de la Universidad de Texas A&M: http://etidweb.tamu.edu/cdrom3/CHAPTER3.PDF

[24] Riabov, V. (2005). *SMTP (Simple Mail Transfer Protocol)* [En línea]. Recuperado el 10 de mayo de 2011, del sitio Web de Rivier College: http://www.rivier.edu/faculty/vriabov/Information-Security-SMTP_c60_p01-23.pdf

[25] *Dos de cada diez usuarios de BlackBerry en Venezuela son niños o adolescentes* (2010, 23 de marzo). El Universal [En línea]. Recuperado el 29 de septiembre de 2011, de: http://www.eluniversal.com/2010/03/23/cyt_ava_dos-de-cada-diez-usu_23A3629771.shtml

[26] *Getting Started with the Blackberry Development Platform*. (2010, Noviembre). Ponencia presentada en el BlackBerry Developer Day 2010. Caracas.

[27] Johnston, C. (2005). *Professional BlackBerry*. Indianapolis, Estados Unidos de América: Wiley Publishing, Inc.

[28] BlackBerry (s.f.). *BlackBerry Messenger* [En línea]. Recuperado el 8 de octubre de 2011, del sitio Web de BlackBerry Venezuela: http://ve.blackberry.com/services/blackberrymessenger/

[29] BlackBerry (s.f.). *RIM proprietary protocols* [En línea]. Recuperado el 8 de octubre de 2011, del sitio Web de BlackBerry: http://docs.blackberry.com/en/admin/deliverables/16558/RIM_propietary_protocols_1107871_11.jsp

[30] Deitel, P. y Deitel, J. (2008). *Cómo programar en Java* (7ª ed.). México: Pearson Educación.

[31] Universidad Tecnológica Nacional (2011). *Aplicaciones para móviles* [En línea]. Recuperado el 26 de julio de 2011, del sitio Web del Instituto CBTech: http://www.aprender21.com/cursos/file.php/40/Unidad7.pdf

[32] Proulx, E. (2007). *An Introduction to the JAIN SIP API* [En línea]. Recuperado el 5 de agosto de 2010, del sitio Web de Oracle: http://www.oracle.com/technetwork/articles/entarch/introduction-jain-sip-090386.html

[33] Nokia Corporation (2004). *Overview JSR180 SIP API for J2ME Version 1.0.1* [En línea]. Recuperado el 10 de mayo de 2011, del sitio Web de Desarrolladores Nokia:http://www.developer.nokia.com/document/Java_Developers_Library_v2/GUID-2508C2ED-C0BE-4512-9302-6805AB7ACB0E/overview-summary.html

[34] NetBeans (s.f.). *Bienvenido a NetBeans y www.netbeans.org* [En línea]. Recuperado el 1 de octubre de 2011, del sitio Web de NetBeans: http://netbeans.org/index_es.html

[35] Eclipse (s.f.). *Mobile Tools for Java (MTJ)* [En línea]. Recuperado el 3 de marzo de 2011, del sitio Web de Eclipse: http://www.eclipse.org/mtj/

[36] BlackBerry (s.f.). *BlackBerry Java Plug-in for Eclipse* [En línea]. Recuperado el 3 de marzo de 2011, del sitio Web de BlackBerry Estados Unidos: http://us.blackberry.com/developers/javaappdev/javaplugin.jsp

[37] Eclipse Community (2009). *Eclipse IDE for Java EE Developers* [En línea]. Recuperado el 11 de noviembre de 2010, del sitio Web de Eclipse: http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/galileor

[38] Java (s.f.). *Descargar Java para Windows* [En línea]. Recuperado el 20 de agosto de 2011, del sitio Web de Java: http://java.com/es/download/windows_xpi.jsp?locale=es

[39] BlackBerry (s.f.). *BlackBerry Smartphone Simulators* [En línea]. Recuperado el 15 de noviembre de 2010, del sitio Web de BlackBerry Estados Unidos: http://us.blackberry.com/developers/resources/simulators.jsp

[40] ICT Backyard (2006). *JSIP API Specification v1.2* [En línea]. Recuperado el 5 de junio de 2011, del sitio Web de Information and Communications Technology Backyard: http://www.ictbackyard.com/resources/javadoc/jcp/32/

[41] Blue Jimp Company (2006). *Jitsi: the OpenSource Java VoIP and Instant Messaging client* [En línea]. Recuperado el 3 de marzo de 2011, del sitio Web: http://bluejimp.com/sip-communicator/api/

[42] IANA (2011). *Service Name and Transport Protocol Port Number Registry* [En línea]. Recuperado el 6 de mayo de 2011, del sitio Web de la Autoridad de Asignación de Números en Internet: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numb

[43] Cohen, M. (2007). *DigestClientAuthenticationMethod.java* [En línea]. Recuperado el 10 de junio de 2011, del sitio Web: http://www.koders.com/java/fidF6DCE5B1D37398A250ECDF7A500271FD913105A7.aspx?s=AddrUtil

[44] Parnes, P. (s.f.). *Java Multicasting Example* [En línea]. Recuperado el 7 de junio de 2011, del sitio Web de Luleå University of Technology: http://staff.www.ltu.se/~peppar/java/multicast_example/

ANEXOS

[ANEXO 1]

CÓMO ELABORAR Y SIMULAR UN MIDLET EN ECLIPSE [4]

Para elaborar un programa se deben seguir los siguientes pasos:

1. Ejecutar Eclipse.
2. Aparecerá la ventana Workspace Launcher donde se debe elegir la dirección de destino del espacio de trabajo (Workspace), o simplemente darle click a OK y se guardará en una dirección por defecto.
3. Dar click en la barra de herramientas sobre File → New → Project.
4. Expandir en la ventana New Project la carpeta BlackBerry, seleccionar BlackBerry Project y dar click en Next.
5. Asignarle un nombre a la aplicación en el campo Project Name y dar click en Finish.
6. Se creará una ventana sobre el editor con nombre BlackBerry Application Descriptor. Sobre la misma hay campos para asignar el título de la aplicación, versión, creador, descripción, tipo de aplicación, entre otras características. Estos campos pueden ser llenados según el deseo del programador. Sin embargo, para este trabajo de grado, todas las aplicaciones que se elaboren serán realizadas como aplicaciones tipo MIDlet ya que el objetivo es que éstas sean compatibles con la mayor cantidad posible de dispositivos móviles.
7. En el Project Explorer (parte izquierda de Eclipse, o en caso de no aparecer, buscarlo sobre la barra de herramientas en Window → Show View → Project Explorer) buscar la carpeta correspondiente al proyecto, expandirla y hacer click derecho sobre la carpeta "src", luego New y por último en Class.
8. Asignar un nombre a la clase en el campo Name. En el campo Package indica la dirección donde se guardará el archivo .java, por ejemplo si se coloca a.b.c, dentro de la carpeta src del proyecto se encontrará la carpeta "a", luego la carpeta "b", y por último dentro de ésta estará la carpeta "c" que contendrá el archivo .java. El campo Superclass es llenado con "javax.microedition.midlet.MIDlet".
9. Activar la casilla "Inherited abstract methods" y hacer click en Finish. Aquí en este punto se tiene un MIDlet vacío.
10. Se escribe el código del programa.
11. Sobre la barra de herramientas hacer click en Run → Debug As → BlackBerry Simulator.
12. Buscar en el dispositivo que se está simulando, la aplicación creada. Por lo general, aparecerá dentro de la carpeta Downloads del mismo; otras veces puede aparecer en alguna parte del menú principal.
13. Ejecutar la aplicación.

[ANEXO 2]

CÓMO AÑADIR UNA API A UN PROYECTO EN ECLIPSE

1. Es común que las APIs sean descargadas como archivos comprimidos (.zip o .rar). Dentro de ellos se encuentran los archivos .jar. Basta con extraerlos y almacenarlos en una carpeta que posteriormente en el paso 5, necesitará ubicar.
2. Con Eclipse abierto y un proyecto creado, se hace click derecho sobre la carpeta de dicho proyecto (en el Package Explorer) y se selecciona la opción Properties.
3. Aparecerá la ventana Properties. Se debe seleccionar en la sección izquierda la opción Java Build Path, conformada por cuatro pestañas.
4. Seleccionar la pestaña Libraries.
5. Dar click en el botón Add External JARs.
6. Agregar los archivos .jar de la API.
7. Dar click en OK.
8. Ya podrá utilizar las clases contenidas en dicha API. Para conocer las funciones de los método que se encuentran dentro de las clases, es recomendable leer sus especificaciones. Éstas pueden conseguirse a través del buscador google colocando el nombre de la API descargada seguida por las palabras API Specification.

[ANEXO 3]

CÓMO GENERAR, DESCARGAR, INSTALAR Y EJECUTAR UN PROGRAMA EN UN BLACKBERRY [4]

Con el archivo .java abierto en el editor de Eclipse de la aplicación creada, hacer click en la barra de herramientas sobre Project → BlackBerry → Package All.

1. Buscar en la computadora la carpeta del espacio de trabajo (carpeta Workspace) la carpeta con el nombre del proyecto.
2. Hay 2 maneras de instalar la aplicación en el móvil:
    a. Desde la PC utilizando el BlackBerry Desktop Software:
        i. Se ejecuta el programa BlackBerry Desktop Software.
        ii. Se hace click sobre Aplicaciones en la parte izquierda del programa.
        iii. Se hace click en "Importar Archivos" en la esquina superior derecha del programa.
        iv. Se ubica la carpeta con el nombre del proyecto y dentro de la misma se debe abrir la carpeta deliverables → Standard.
        v. En esta carpeta se selecciona el archivo de extensión .axl y se hace click en Abrir.
        vi. Se hace click en Aplicar (parte inferior derecha). La aplicación se instalará en el móvil.
    b. Desde el móvil:
        i. En la PC se ubica la carpeta con el nombre del proyecto y dentro de la misma se debe abrir la carpeta deliverables → Standard → 5.0.0.
        ii. Se copian todos los archivos de la carpeta en el teléfono. La manera más fácil es conectar el teléfono en modo de almacenamiento a la PC, pasar los archivos a un mismo destino del teléfono y desconectarlo de la PC.
        iii. Desde el teléfono se ubican los archivos transferidos y se da click sobre el archivo con extensión .jad.
        iv. Se hace click sobre Descargar para que la aplicación se instale.
    3. Una vez finalizada la instalación, se busca el archivo en el menú principal o en la carpeta Descargas.
    4. Se ejecuta la aplicación.
5. Con el archivo .java abierto en el editor de Eclipse de la aplicación creada, hacer click en la barra de herramientas sobre Project → BlackBerry → Package All.
6. Buscar en la computadora la carpeta del espacio de trabajo (carpeta Workspace) la carpeta con el nombre del proyecto.
7. Hay 2 maneras de instalar la aplicación en el móvil:
    a. Desde la PC utilizando el BlackBerry Desktop Software:
        i. Se ejecuta el programa BlackBerry Desktop Software.
        ii. Se hace click sobre Aplicaciones en la parte izquierda del programa.

iii. Se hace click en “Importar Archivos” en la esquina superior derecha del programa.
iv. Se ubica la carpeta con el nombre del proyecto y dentro de la misma se debe abrir la carpeta deliverables → Standard.
v. En esta carpeta se selecciona el archivo de extensión .axl y se hace click en Abrir.
vi. Se hace click en Aplicar (parte inferior derecha). La aplicación se instalará en el móvil.

b. Desde el móvil:
i. En la PC se ubica la carpeta con el nombre del proyecto y dentro de la misma se debe abrir la carpeta deliverables → Standard → 5.0.0.
ii. Se copian todos los archivos de la carpeta en el teléfono. La manera más fácil es conectar el teléfono en modo de almacenamiento a la PC, pasar los archivos a un mismo destino del teléfono y desconectarlo de la PC.
iii. Desde el teléfono se ubican los archivos transferidos y se da click sobre el archivo con extensión .jar.
iv. Se hace click sobre Descargar para que la aplicación se instale.
En otros equipos con Java, basta con tener en una misma carpeta los archivos .jad y .jar. De igual forma se ejecuta el archivo .jar para realizar la instalación.

8. Una vez finalizada la instalación, se busca el archivo en el menú principal o en la carpeta Descargas.
9. Se ejecuta la aplicación.

[ANEXO 4]

CÓMO GENERAR UN ARCHIVO EJECUTABLE (.JAR)

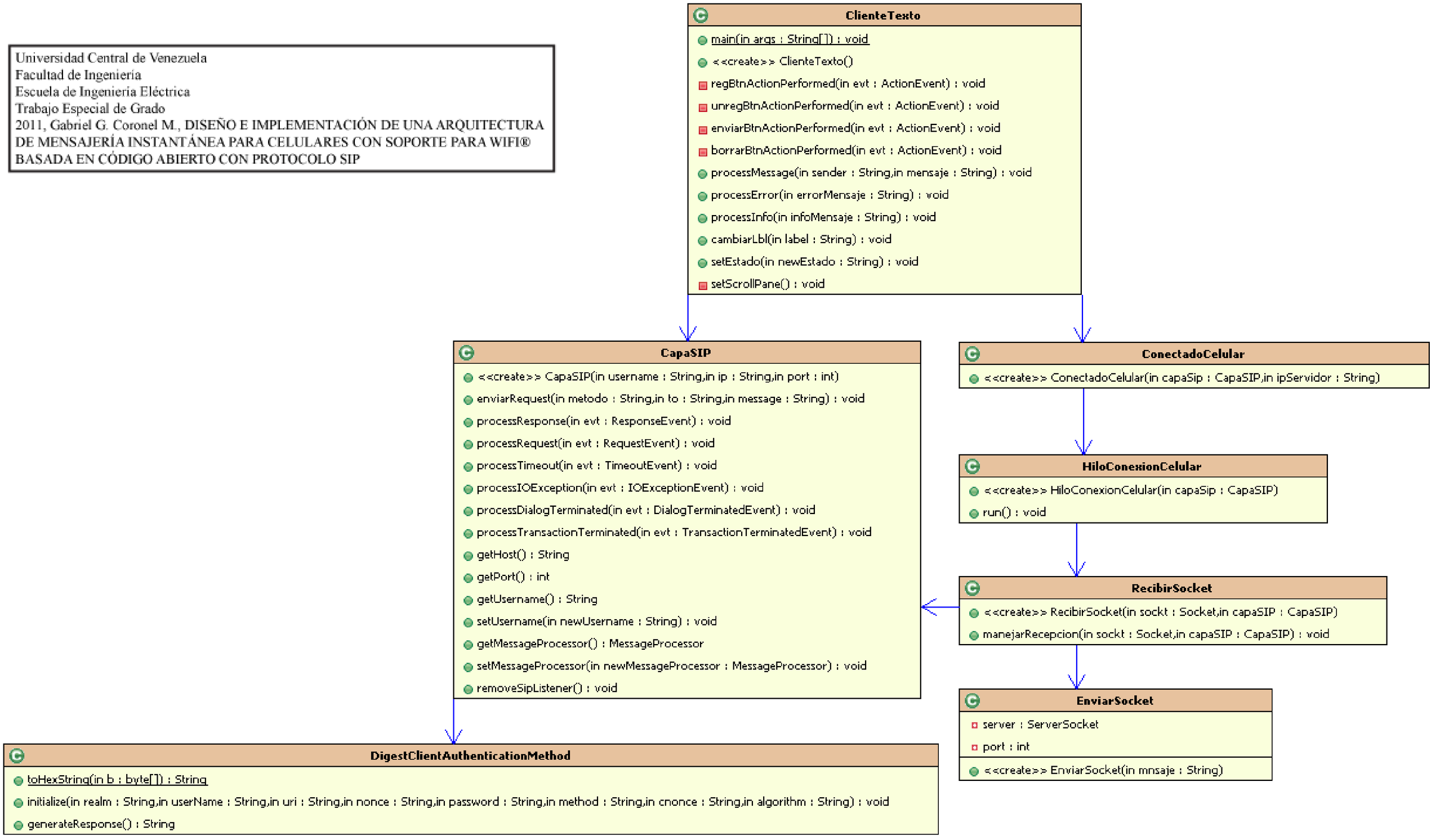
Estas instrucciones son válidas para generar un archivo .jar que puede ser ejecutado en una computadora con Java como si fuera un archivo ejecutable .exe.

1. Hacer click en File en la barra de herramientas de Eclipse.
2. Seleccionar la opción Export.
3. Abrir la carpeta Java y seleccionar Runnable JAR file.
4. En el campo Launch configuration se debe seleccionar dentro de la lista que allí aparece, la opción que coincida con el nombre de la clase principal y con el nombre del proyecto.
5. En el campo Export destination debe ubicar la carpeta de destino y el nombre que le asignará al archivo .jar
6. En las opciones Library handling se selecciona la opción Extract required libraries into generated JAR.
7. Finalmente, se hace click en el botón Finish.
8. En caso que le aparezca una ventana de advertencia, seleccione OK.
9. El archivo .jar es exitosamente creado y se encuentra en la carpeta que previamente seleccionó.
10. Para ejecutarlo haga doble click, si tiene problemassólo bastará que la computadora tenga Java Runtime Enviroment instalado y

[ANEXO 5]

DIAGRAMA UML DE LA APLICACIÓN PARA COMPUTADORAS

Universidad Central de Venezuela
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Trabajo Especial de Grado
2011, Gabriel G. Coronel M., DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DE MENSAJERÍA INSTANTÁNEA PARA CELULARES CON SOPORTE PARA WIFI® BASADA EN CÓDIGO ABIERTO CON PROTOCOLO SIP



[ANEXO 6]

DIAGRAMA UML DE LA APLICACIÓN PARA CELULARES

Universidad Central de Venezuela
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Trabajo Especial de Grado
2011, Gabriel G. Coronel M., DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA DE MENSAJERÍA INSTANTÁNEA PARA CELULARES CON SOPORTE PARA WIFI® BASADA EN CÓDIGO ABIERTO CON PROTOCOLO SIP