UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ESTUDIO E IMPLEMENTACIÓN DE UNA RED DE ÁREA PERSONAL (WPAN) DE TIPO "BLUETOOTH", UTILIZANDO LA TÉCNICA DE ESPECTRO DISPERSO

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

P R E S E N T A N

ERIC ALFREDO RÍOS ROCHA

DAVID ALONSO TLÁHUETL HERRERA

DIRECTOR DE TESIS: DR. FRANCISCO GARCÍA UGALDE

FEBRERO DEL 2003

AGRADECIMIENTOS

Nos preguntamos: "¿quién me he creído para ser brillante, espléndido, talentoso, sensacional?", pero en realidad, ¿quiénes nos hemos creído para no serlo?

Dedico este trabajo a todas aquellas personas que han contribuido a mi formación, tanto académica como personal. Ustedes han sido un ejemplo y una base para mis ideas, metas, compromisos y acciones.
El llegar a este nivel sólo puede ser posible cuando se tiene apoyo y confianza hacia alguien. Yo puedo afirmar que los he tenido, así agradezco que formen parte de mi vida.

A mis padres: Porque me debo a ustedes y porque soy por ustedes, por enseñarme tantas cosas, por aprender a quererlos, por admirarlos, por saber que soy parte de sus vidas y porque se preocupan mi. Sé que no he podido ser muy expresivo, pero los quiero demasiado y me preocupo mucho por ustedes. El haber concluido una carrera no es sino un compromiso que he adquirido para retribuirles todo lo que me han dado, ahora es mi turno. Les demostraré cuáles son mis metas y mis ideales. Es por ustedes.

A mis hermanos: Porque me gustaría que comprendieran que muchas de las cosas que hago o digo, son por ustedes, porque yo tuve los mismos problemas y los mismos miedos, pero he tratado de superarlos y demostrarles que sólo se necesita compromiso y convicción en todo lo que hagan. Los quiero mucho.

A mis amigos: Porque ustedes me ayudaron, me escucharon, me impulsaron e inspiraron en mi muchas metas. Gracias a todos aquellos que me estiman: yo he cumplido con demostrarles que una amistad sincera vale más que muchas otras cosas. Va a ser imposible olvidar las anécdotas, las risas, los corajes, y todo lo que hizo que nos conociéramos y nos entendiéramos.

A mis maestros: Porque son un ejemplo de dedicación, respeto y admiración. Gracias a ustedes he podido comprender el significado de la vocación. Sus enseñanzas son parte de mi formación.

A mi Universidad: Porque sólo estando aquí es donde se comprende y se aprende cuán grande y fabulosa es esta institución. Gracias a ella conocí mucha gente y logré muchas cosas. Te debo mi carrera.

En especial, agradezco al Dr. Francisco García Ugalde por su apoyo, consejo y paciencia para ayudarme en este trabajo. Su asesoría es un regalo que aprecio mucho.

Agradezco al programa de Alto Rendimiento Académico, en particular a Lidia Delgado: gracias a ti pude continuar. También valoro mucho el apoyo brindado por la Fundación UNAM, la Fundación Telmex, Inttelmex y el programa Probetel.

Finalmente, quiero que sepan que valoro mucho su apoyo, sus críticas, sus comentarios, y sus favores. Todo en conjunto ha nutrido mucho mi forma de pensar, actuar y decidir. Hoy me sumo a la tarea de construir un futuro y unas mejores condiciones de vida.

Por todo lo que falta...

Eric Ríos

Este trabajo de tesis representa la conclusión de una etapa, la cual tiene un valor especial para mí. La importancia de este ciclo radica tanto en los conocimientos científicos aprendidos, como en las experiencias vividas. A lo largo de este periodo hubo muchas personas que influyeron de manera significativa, y a las cuales me gustaría hacer un reconocimiento a través de las siguientes líneas.

En primer lugar, agradezco a mis padres por el apoyo incondicional que me han brindado, sus enseñanzas, su aliento, su confianza, su dedicación y paciencia; con todo ello, han sido un ejemplo y guía. Continuando con la familia, quiero hacer una mención especial a mis tíos, que han contribuido de manera definitiva para alcanzar esta meta.

A mis profesores por haber compartido conmigo sus conocimientos, su experiencia y su tiempo. En particular, agradezco al Dr. Francisco García Ugalde, por haber dirigido el desarrollo de esta tesis mediante su asesoría, así también, por proporcionar los medios necesarios para llevarla a cabo.

A mis amigos, que me ofrecieron su ayuda dentro y fuera del salón de clase, con quienes pasé momentos agradables, y de quienes también aprendí.

Mi gratitud a la UNAM, a toda la gente que con su trabajo y participación hace posible que exista esta gran institución, de la cual formo parte y me siento muy orgulloso.

Para finalizar, sólo me queda decir gracias nuevamente y espero corresponderles cuando así sea necesario.

David Alonso Tláhuetl Herrera

ÍNDICE

H

INTRODUCCIÓN

Una red inalámbrica puede definirse como una colección de dispositivos interconectados entre sí, capaces de intercambiar algún tipo de información; y que tiene como medio de transmisión el aire.

Actualmente la mayoría de las redes inalámbricas dependen de la aplicación y la naturaleza del proveedor de servicios de telecomunicaciones que las implemente. Por un lado, las redes inalámbricas pueden utilizarse para proporcionar movilidad, liberando al usuario de un enlace fijo. Por otro lado, pueden ser usadas por el proveedor como parte de su infraestructura cuando sea más conveniente o más barata la estructuración de una red inalámbrica en lugar de una cableada.

Muchos de nosotros nos hemos acostumbrado a las limitaciones que tenemos con una red cableada. Cuando queremos revisar nuestro correo electrónico o imprimir un reporte, nos encontramos confinados a cierto lugar o espacio. En los últimos años, la creciente popularidad de las comunicaciones inalámbricas llamó la atención de cuerpos corporativos, de manufactura y académicos. La red inalámbrica ha probado que puede cubrir los beneficios de una red cableada con la adición de los beneficios de libertad de movilidad. Algunos de los sistemas que utilizan las redes inalámbricas son los siguientes: sistemas celulares, sistemas inalámbricos como extensión de redes fijas, sistemas inalámbricos de acceso fijos, sistemas inalámbricos de dispositivos portátiles [12].

Los servicios de datos en estos sistemas inalámbricos se han desarrollado rápidamente durante los últimos cinco años. Estos sistemas han cambiado debido a muchos factores, principalmente técnicos, económicos y de regulación.

Actualmente es común portar numerosos dispositivos electrónicos como lo son: *laptops*, teléfonos móviles, PDA's (Asistentes Digitales Personales), cámaras digitales, reproductores mp3, minidisc, etc. Estos dispositivos se han vuelto cotidianos y representan tanto una herramienta de trabajo como de entretenimiento.

La habilidad de integrar más transistores en un área pequeña de silicio ha hecho posible la construcción de dispositivos embebidos pequeños, capaces de funcionar con protocolos

complejos. Los controladores embebidos en dispositivos pueden ser programados, controlados y usados de varias formas inteligentes. Así pueden encontrarse dispositivos inteligentes tanto en el trabajo como en la casa. Varias técnicas están disponibles para conectar estos dispositivos a la Internet, formando así la llamada *Internet embebida*. También, se ha hecho gran progreso desarrollando sensores pequeños y baratos que pueden captar señales útiles del entorno del usuario sin la interacción del usuario, o de una orden explícita [12].

Los sistemas de cómputo móviles, tales como PDA'S (Asistentes Personales Digitales) y agendas, han tenido un crecimiento muy rápido dentro de la industria de las computadoras. Con ellos se pretende tener una oficina portátil para poder enviar y recibir faxes, correo electrónico, archivos y acceder a máquinas remotas desde cualquier lugar.

La relación entre las redes inalámbricas y los dispositivos de cómputo portátiles es estrecha, sin embargo, no siempre se encuentra presente. Existen computadoras portátiles que requieren cableado para poder acceder a una red. También se pueden observar redes inalámbricas que no son portátiles, por ejemplo, aquellas que comunican a dos edificios mediante transmisores/receptores láser ubicados en los techos.

Los dispositivos portátiles que pueden establecer redes inalámbricas presentan el beneficio de movilidad y la posibilidad de conectarse a servidores centrales más inteligentes, sensores o actuadores. Un sistema de este tipo, facilita el intercambio de información y la automatización de procesos. Una forma de implementar un sistema como éste, dentro de un área personal de operación, es a través de la tecnología llamada *Bluetooth*.

La necesidad inmediata de *Bluetooth*, vino del deseo de conectar periféricos y dispositivos, sin el uso de cables. La tecnología inalámbrica disponible para tal fin, antes de la llegada de *Bluetooth*, era IrDA. Dichos sistemas están basados en enlaces infrarrojos, limitados a conexiones en línea de vista.

Bluetooth está fuertemente equipado para resolver la demanda de acceso móvil e inalámbrico a LANs, Internet sobre móviles y otras redes existentes, donde la parte principal se cablea pero la interfaz es libre de moverse. Esto no sólo hace que la red sea más fácil de usar sino que también extiende su alcance.

Las ventajas y la rápida proliferación de las redes LAN sugieren la puesta en marcha de redes de área personal tipo *Bluetooth* como un complemento para incrementar la interacción entre el usuario y la red LAN a través de interfaces que permitan enlaces dinámicos y sencillos.

La necesidad comercial de proporcionar capacidades informáticas en los dispositivos móviles, puede ser satisfecha a través de *Bluetooth*.

Una característica esencial de *Bluetooth* es su habilidad para manejar transmisiones de datos y voz simultáneamente, permitiendo soluciones innovadoras, tales como: un auricular de "manos libres" móvil para llamadas de voz, capacidad de impresión de fax, y sincronización automática con PDA's, *laptops*, y aplicaciones en teléfonos celulares.

I.1 Justificación

La revolución de las telecomunicaciones y el procesamiento de la información están creando una sociedad cuyas actividades y procesos se basan cada día más en la información electrónica. Los avances e interacciones de estos campos se han incrementado aceleradamente en los últimos años y se prevé un mayor desarrollo para los siguientes, con miras a penetrar más en la vida cotidiana de todos los entes que conforman nuestra comunidad. Por esta razón, se puede estimar un incremento considerable en la utilización de las redes de comunicaciones.

Las redes cableadas presentan muchos inconvenientes en conexiones a corta distancia, por lo que se ha planteado a *Bluetooth* como una alternativa para la gran cantidad de aplicaciones que requieren compartir información en dicho entorno. Esta tecnología encuentra justificación en la creciente necesidad de interacción, entre dispositivos portátiles, de una manera sencilla y dinámica.

Una consecuencia, de impacto general, relacionada con la comunicación eficiente entre este tipo de dispositivos es la mejor explotación de los recursos existentes en dichos aparatos, lo cual se puede traducir en un incremento en la productividad de las personas y optimización de los procesos, con todos los beneficios inherentes a este hecho.

De la importancia de *Bluetooth* se desprende el valor del presente trabajo de tesis. Con este trabajo, además de alcanzar los objetivos que se plantean a continuación, se pretende establecer un precedente para futuros proyectos que aspiren a mejorar dicha tecnología, o bien, para el desarrollo de sistemas afines.

I.2 Objetivos

El desarrollo del presente trabajo de tesis se enfoca en el análisis de la tecnología *Bluetooth* como parte de la implementación de una red inalámbrica de área personal.

El estudio de la tecnología *Bluetooth* comprende los protocolos utilizados desde la capa de radio hasta la capa de aplicación y tiene como finalidad conocer los procedimientos y parámetros necesarios en la implementación de la red.

Una vez analizada esta tecnología, se implementará una red inalámbrica de área personal, a través de un *kit* de desarrollo, con dos módulos *Bluetooth*. Dicha red constará de dos dispositivos, una computadora portátil y una de escritorio, los cuales podrán intercambiar información de forma dinámica y sencilla mediante la tecnología *Bluetooth*.

En suma, se pretende probar la utilidad y viabilidad de la tecnología *Bluetooth* dentro de los sistemas de comunicaciones inalámbricas, a través del estudio e implementación de una red inalámbrica de área personal.

I.3 Estructura del documento

El presente documento está conformado por seis capítulos, una sección de conclusiones, seis anexos, un glosario y un apartado bibliográfico. A continuación se describe brevemente el contenido de cada uno.

En la introducción de este documento se muestra la justificación y objetivos del trabajo, la estructura y el marco de referencia en el que se desarrolla la presente tesis. De una forma concisa, se explica la historia de las redes de comunicación, así como la evolución que han tenido para dar paso a las redes inalámbricas. Se resalta la importancia de las últimas en la vida cotidiana y su necesidad en la actualidad.

En el primer capítulo se trata el tema de las redes de área personal y sus características. Se estudian las redes *ad hoc*, el enfoque inalámbrico, y la necesidad de la creación de un estándar para las comunicaciones de área personal.

En el segundo capítulo se describe la tecnología *Bluetooth*: sus orígenes, sus objetivos, así como las necesidades que llevaron a su creación. Se destacan las características que hacen de esta tecnología una herramienta útil para la comunicación en la actualidad, su relación con otras tecnologías inalámbricas que se han utilizado y la proyección que tiene a futuro. Finalmente, se describe la pila de protocolos que la conforman y las aplicaciones que se pueden implementar para innovadoras soluciones a la comunicación inalámbrica dentro de un área personal.

En el tercer capítulo, se describe la parte de la pila de protocolos que corresponde al módulo *Bluetooth*. Se estudian las capas fundamentales para el establecimiento de la comunicación a nivel del hardware: *radio* y *bandabase*, *controlador de enlace*, *administrador de enlace*, y la *interfaz de control del host*. Se establecen las relaciones entre ellas y la estructura de los paquetes de información que utilizan.

En el cuarto capítulo se expone la parte de la pila de protocolos que corresponde al *host* que maneja al módulo *Bluetooth*. Se detallan las capas para poder dar servicio a aplicaciones y sus respectivos protocolos: L2CAP, RFCOMM y SDP. Se explica cómo se multiplexan los paquetes y como se registran y usan los servicios que se pueden brindar con un solo módulo *Bluetooth*.

En el quinto capítulo se expone la interacción de todas las capas de la pila de protocolos *Bluetooth*, la cual representa un mecanismo necesario para el establecimiento de *perfiles* para el desarrollo de nuevas aplicaciones, la seguridad y la calidad de servicio que se garantizan en la transmisión de información mediante esta tecnología.

En el sexto capítulo se describe la implementación de una red inalámbrica de área personal utilizando la tecnología *Bluetooth*. El objetivo es comprobar la utilidad y la gran versatilidad que ofrece la creación de aplicaciones basándose en el análisis previo de las características de dicha

tecnología, dando como resultado una aplicación que ilustra la interacción de las capas de la pila de protocolos para la transmisión de información entre dos dispositivos móviles.

En la sección de conclusiones, se expondrán de forma concisa los resultados obtenidos del análisis de la tecnología *Bluetooth*, así como de la red inalámbrica de área personal implementada.

Los anexos son un conjunto de resúmenes que le permiten al lector conocer con mayor profundidad algunos de los temas de esta tesis, sin necesidad de recurrir a otras fuentes. Se habla del espectro radioeléctrico, la técnica de espectro disperso (*spread spectrum*), las redes basadas en células, el protocolo WAP, otras tecnologías inalámbricas y las especificaciones de los módulos *Bluetooth* utilizados.

El glosario esta formado por los términos más utilizados dentro la tecnología *Bluetooth*, los cuales están acompañados de una breve descripción.

Finalmente se tiene la bibliografía y las referencias electrónicas usadas durante el desarrollo del presente trabajo de tesis.

I.4 Marco de referencia

I.4.1 Historia y desarrollo de las comunicaciones inalámbricas

La historia de las comunicaciones inalámbricas vía radio se puede dividir en tres grandes fases. La primera se refiere a la demostración y predicción teórica de las ondas de radio; la segunda está vinculada al desarrollo y evolución tanto de los equipos como de las técnicas; la tercera y última fase es donde los servicios de comunicaciones móviles son proporcionados en gran escala a todos los sectores de la población [4].

I.4.1.1 Fundamentos teóricos

Durante esta primera fase se establecen los principios fundamentales del fenómeno de radio, de tal manera que los sistemas están limitados al uso experimental en los laboratorios.

Este periodo comienza en 1678 con el trabajo de Huygens sobre los fenómenos de reflexión y refracción de la luz. Fue Fresnel, en 1819, quien demostró la naturaleza ondulatoria de la luz. En 1865, Maxwell estableció sus ecuaciones que unifican a los fenómenos eléctricos y magnéticos, pero no fue hasta 1887 que Hertz demostró por primera vez la posibilidad de transmitir y detectar una onda electromagnética entre dos puntos separados unos cuantos metros. Marconi, basándose en el trabajo previo, fue quien sacó la tecnología de radio fuera del laboratorio y le dio un enfoque práctico. Marconi observó que las ondas de radio de gran longitud se propagaban a través de largas distancias, así que en 1896 pudo establecer comunicación a lo largo de una distancia de 3 Km.

I.4.1.2 Desarrollo

En 1898 se mostró la comunicación vía radio entre un barco y una isla, tres años más tarde, Marconi estableció la primera transmisión de radio transatlántica entre Europa y Estados Unidos. En 1901, Marconi demostró con éxito la transmisión y recepción de la letra *S* en código Morse entre Inglaterra y Estados Unidos (3000 Km), utilizó una frecuencia menor a 1 MHz. Seis años después fue el mismo Marconi quien creó el primer servicio comercial de comunicaciones vía radio para ser usado en sistemas marítimos.

A pesar de los avances logrados durante esta fase, a principios del siglo XX, el tamaño de los transmisores y receptores de radio era bastante grande. Ha sido gracias a los avances tecnológicos y al mejoramiento en el desempeño de los equipos que los sistemas de radio se han vuelto realmente móviles. Entre otros, el desarrollo del diodo por Fleming y De Forest, y la invención del triodo por Lieben, hicieron posible reducir el tamaño del equipo y el desarrollo de los servicios de radio y telefonía en vez de las aplicaciones previas que permitían solamente servicios telegráficos.

Desde principios del siglo XX los servicios de radiocomunicación han experimentado un crecimiento en todas las áreas, tanto públicas como privadas. Primero fue reconocida su utilidad en los servicios marítimos, después en los servicios de seguridad pública como policía y bomberos. Más tarde las radiocomunicaciones cobraron importancia en el sector privado, se utilizaron principalmente en el sector energético y el de transporte.

En 1912, las frecuencias reguladas para uso comercial estaban debajo de los 3 MHz. Entre 1930 y 1960, el rango de frecuencias para los servicios de radio se incrementó gradualmente para incluir a las ondas cuyas longitudes eran métricas, decimétricas y centimétricas[1]. Al mismo tiempo se empezó a usar modulación en frecuencia.

Durante la segunda guerra mundial se aceleró el proceso de desarrollo de los sistemas de radio, y su uso fue extendido al sector civil en 1950. A pesar de ello, el equipo todavía era muy pesado y ocupaba gran espacio dentro de los vehículos en los que era instalado.

I.4.1.3 Servicios móviles para el público en general

En esta fase, el progreso técnico y el desarrollo de los sistemas de comunicaciones permitieron que los sistemas de comunicaciones móviles de radio fueran asequibles al público en general.

La primera red a gran escala de comunicaciones móviles fueron los sistemas celulares[2] analógicos, los cuales fueron desarrollados durante la década de los 70's. En 1979, el primer sistema celular llamado AMPS (*Advanced Mobile Phone Service*) fue instalado en Chicago, le siguió en 1980 el HCMTS (*High Capacity Mobile Telephone System*) en Tokio. Durante la década

[1] Una tabla con las bandas de frecuencias, servicios y aplicaciones se encuentra en el anexo A.
[2] Una descripción del funcionamiento de las redes basadas en células se encuentra en el anexo B.

de los 80's, los sistemas celulares analógicos se popularizaron en muchos países. En 1981, se introdujo en Escandinavia el NMT (*Nordic Mobile Telephone*). En 1985 *Radiocomm 2000* inició operaciones en Francia, TAC en el Reino Unido y el sistema C 450 en Alemania. Al mismo tiempo, el uso de teléfonos fijos inalámbricos registraba un gran incremento.

En los 90's, los sistemas de *paging* alcanzaron grandes tasas de crecimiento; sin embargo, la muestra más significativa del desarrollo y crecimiento de las comunicaciones móviles durante esa década estuvo representada por GSM. Con GSM se tuvo la posibilidad de obtener servicios tipo ISDN y disponibilidad de *roaming* internacional, esta tecnología representa a la segunda generación de telefonía celular. Esta generación agrupa a los sistemas digitales enfocados a telefonía, pero con la capacidad de transmitir datos a baja velocidad.

Los últimos años de la década de los 90's se caracterizaron, en el ámbito de las comunicaciones inalámbricas, por la existencia de múltiples aplicaciones con estándares incompatibles. Para hacer compatibles dichos estándares y crear sistemas de comunicaciones enfocados a un uso personal, se planteó la creación de una tercera generación de telefonía celular. Esta tercera generación pretende unificar estándares, tanto de voz como de datos, en una nueva tecnología basada en la conmutación de paquetes. Bajo la tercera generación se ofrecerán servicios de voz, video, correo electrónico, Internet, multimedia y comercio electrónico, desde cualquier lugar. Actualmente se ha empezado la transición de la segunda a la tercera generación, en algunos países desarrollados.

Hasta este punto se ha enfocado el desarrollo de las comunicaciones móviles al tráfico de voz; sin embargo, aunque se han extendido menos, las redes inalámbricas para la transmisión de datos también representan un grupo importante dentro de las comunicaciones móviles.

I.4.2 Redes de datos inalámbricas

Existe una gran variedad de implementaciones para los servicios de transmisión de datos de dos vías en redes móviles. Los sistemas que soportan dichas tecnologías pueden ser de dos tipos diferentes. El primer tipo fue originalmente diseñado para la transmisión de voz, mientras el segundo, fue desarrollado y específicamente optimizado para la transmisión de datos.

La primera y segunda generación de celulares y teléfonos inalámbricos pertenecen al primer grupo, mientras ARDIS y MOBITEX se encuentran en el segundo. Otros sistemas como TETRA, la tercera generación de celulares y *Bluetooth* soportan tanto voz como datos.

Las aplicaciones de transmisión de datos se pueden dividir en dos grupos. En el primero, se encuentran las aplicaciones que están principalmente enfocadas al desarrollo de un trabajo profesional; mientras en el segundo, están aquellas aplicaciones dirigidas al público en general.

Las aplicaciones del primer grupo pueden ser tales como: cómputo móvil, administración de flotas, manejo de inventarios, medición, señalización de alarmas, verificación de tarjetas de

crédito, seguridad pública, renta de autos, control de ventas remotas, control de tráfico, entre otras.

Entre las aplicaciones del segundo grupo están: sistemas de asistencia automovilística, correo electrónico, acceso a bases de datos y asistentes digitales personales.

Dentro de estos dos grupos, se pueden tipificar las redes según su extensión. Existen: redes de área ancha (WAN), la cuales pueden cubrir un país; redes de área local (LAN) que abarcan varias decenas de metros; y redes de área personal (PAN) cuyo rango de operación es de unos pocos metros.

I.4.2.1 Redes inalámbricas de área ancha

Las redes inalámbricas de área ancha poseen similitudes con los sistemas celulares, sin embargo, éstas se basan en una estructura más simple que ha sido optimizada para la transmisión de datos y que presenta un mejor desempeño que los sistemas de *paging*. Estas redes proporcionan servicio a dispositivos que se mueven con gran velocidad, pero a tasas de transmisión muy bajas.

Generalmente, el dispositivo terminal es una *laptop* equipada con un radio módem y software de comunicaciones. La conexión de la red móvil con los servidores a través de una red cableada permite el acceso remoto a bases de datos, actualizaciones, o procesamiento.

Existen varios tipos de tecnologías dentro de esta clasificación, las más importantes son: ARDIS, MOBITEX, CDPD Y GPRS [6].

Las velocidades de transmisión de estos sistemas son: ARDIS y CDPD hasta 19.2 Kbps; MOBITEX hasta 8 Kbps; GPRS puede llegar, en operación multislot en un solo móvil, hasta 170 Kbps [4].

I.4.2.2 Redes inalámbricas de área local

Una red de área local o WLAN (*Wireless* LAN) utiliza ondas electromagnéticas, ya sea ondas de radio o infrarrojo, para enlazar los equipos conectados a la red, en lugar de cables coaxiales, UTP, o fibra óptica que se utilizan en las LAN convencionales cableadas.

Las redes WLAN proporcionan las facilidades no disponibles en los sistemas cableados, por lo que no resultan una sustitución de las redes LAN convencionales, sino una extensión. Además se puede formar una red de mayor tamaño donde coexistan los dos tipos de sistemas, enlazando los diferentes equipos o terminales móviles asociados a la red. El objetivo principal de este tipo de redes es permitir una alta tasa de transmisión para usuarios con poca movilidad sobre un área limitada de cobertura, como un edificio o un campus.

Las redes WLAN que utilizan infrarrojos generalmente están dedicadas a las comunicaciones entre terminales fijas. En el caso de las WLAN que usan radio, es factible

ofrecer una mayor movilidad. Las bandas de frecuencia utilizadas en las WLAN vía radio, son: 18-19 GHz, bandas ISM (902-908 MHz, 2.4-2.5 GHz, 5.8-5.9 GHz), y 5 GHz.

Las redes inalámbricas que utilizan altas frecuencias permiten la reutilización de frecuencias debido a que las señales en dichas bandas sufren una alta atenuación debido a estructuras densas como el concreto. Por otro lado las redes que se ubican en las bandas ISM, generalmente, utilizan la técnica de *espectro disperso* para realizar CDMA.

Dentro de las WLAN existen dos estándares principales: IEEE 802.11 (Estados Unidos) e HIPERLAN (Europa). El primer estándar opera en la banda ISM de 2.4-2.5 GHz, utiliza la técnica de espectro disperso y alcanza velocidades de hasta 2 Mbps. El estándar HIPERLAN opera a 5 GHz, utiliza FDMA y alcanza velocidades de hasta 20 Mbps.

A lo largo de esta introducción se ha establecido el marco de referencia que nos permitirá ubicar a las redes inalámbricas de área personal dentro del contexto de las comunicaciones inalámbricas. El estudio de las redes inalámbricas de área personal (WPAN) se desarrollará en el siguiente capítulo.

PAGINACIÓN DISCONTINUA

CAPÍTULO 1
REDES DE ÁREA PERSONAL

El uso de dispositivos electrónicos portátiles tales como: *laptops*, teléfonos celulares, *palmtops*, reproductores de audio y video, se ha incrementado recientemente. Sin embargo, estos dispositivos generalmente se usan de manera separada, es decir, sus aplicaciones no interactúan. Una de las razones principales por la que estos aparatos se usan de manera separada es porque su interconexión debe realizarse vía cable, lo cual resulta un proceso tedioso y engorroso que consume mucho tiempo.

Para hacer que estos elementos interactúen de una manera sencilla y dinámica, es necesario el establecimiento de una red de área personal con características especiales de conectividad.

Las redes de área personal (PAN's) se distinguen de otro tipo de redes por su tamaño y su enfoque. La comunicación en las PAN's está normalmente confinada a un área personal que generalmente se extiende hasta 10 metros en todas direcciones de una persona u objeto, e involucra a dos o más dispositivos, estacionarios o en movimiento.

Este tipo de redes tienen gran aplicación tanto en el ambiente profesional como en el hogar. En el ambiente profesional, existe una gran gama de dispositivos y posibilidades en los que se puede establecer este tipo de redes para hacer más eficiente el intercambio de información. En el hogar, los dispositivos de una red PAN pueden interactuar con los aparatos electrodomésticos, tanto para intercambio de información como para control.

Desde el punto de vista operativo, se espera que en una red de área personal puedan participar tanto dispositivos pertenecientes a dicha red como a otras redes PAN, y se debe tener en cuenta que éstos pueden entrar o salir del área de cobertura continuamente, debido a que se trata de dispositivos móviles. De lo anterior se puede observar que se trata de redes muy dinámicas, en donde en cualquier momento, los elementos pueden incorporarse o separarse.

1.1 Redes tipo *Ad hoc*

Para tratar con redes de naturaleza volátil, el concepto de redes tipo *ad hoc* puede aplicarse para crear una conectividad flexible y robusta. La conectividad *ad hoc* se puede definir como un esquema que permite a los dispositivos establecer comunicación, espontánea y conveniente, en cualquier momento y lugar, sin la necesidad de una infraestructura central [12].

Los nodos que forman este tipo de redes son móviles y utilizan una interfaz inalámbrica para el envío de los paquetes de datos. Debido a que los nodos pueden tener la función tanto de enrutadores como de *hosts*, estos pueden dirigir paquetes en nombre de otros nodos, así como también ejecutar aplicaciones. Una diferencia fundamental con otro tipo de redes es que las redes *ad hoc* pueden operar en un ambiente en el que algunos o la totalidad de los nodos son móviles. Algunas características de estas redes son [6]:

- **Operación distribuida**. En estas redes un nodo puede no contar con el respaldo de la infraestructura central para el soporte de seguridad o enrutamiento, es por ello que estas funciones deben ser diseñadas de tal manera que puedan operar eficientemente bajo condiciones distribuidas.
- **Topología dinámica de la red**. La movilidad de los nodos tendrá como resultado variaciones en la topología, no obstante, la conexión en la red debe mantenerse para permitir a las aplicaciones y servicios operar ininterrumpidamente. En particular, esta característica afectará a los protocolos de enrutamiento.
- **Capacidad variable de enlace**. Los efectos de altas tasas de error pueden ser más severos en una red *ad hoc* en la que la información tenga que cruzar por varios elementos para llegar a su destino final (*multihop*), debido a que los errores de cada enlace se acumulan dando como resultado un efecto total mayor. Por lo anterior, se puede observar la conveniencia de crear redes punto a punto para los dispositivos que intercambien información frecuentemente. De esta manera se hace más eficiente el intercambio de datos entre los elementos en cuestión y se mejora el desempeño de la red en general.
- **Dispositivos de bajo consumo de potencia**. La característica de movilidad limita a los dispositivos en el uso de potencia tanto en las etapas de transmisión/recepción, como en el procesamiento y almacenamiento de los datos y señales.

1.1.1 Enrutamiento *Ad hoc*

El problema de enrutamiento de paquetes entre cualquier par de nodos en una red PAN *ad hoc* o entre redes PAN se complica porque los nodos se pueden mover aleatoriamente dentro de la red. Una ruta que se considere óptima en un momento, puede ya no serlo en otro. Además, las

propiedades estocásticas de los canales inalámbricos y del ambiente de operación añaden incertidumbre a la calidad de la ruta [12].

Tradicionalmente los protocolos de enrutamiento son proactivos, es decir, mantienen las rutas de todos los nodos, incluyendo los nodos a los que no se enviarán paquetes. Dichos protocolos reaccionan a cualquier cambio en la topología aunque no se afecte el tráfico debido a esa variación, para ello requieren mensajes de control periódicos para mantener las rutas a cada nodo en la red. Este enfoque utiliza una mayor cantidad de recursos, potencia y ancho de banda, a medida que la movilidad de los nodos aumenta.

Debido a las características de las redes PAN es preferible el uso de protocolos de enrutamiento reactivos, en donde las rutas entre nodos son determinadas únicamente cuando se requiere explícitamente enrutar algún paquete. De esta forma se previene que los nodos desperdicien recursos al actualizar constantemente todas las rutas de la red; en su lugar, los nodos se enfocan únicamente a las rutas que están siendo usadas, o a las que establecerán.

Las PAN tiene la posibilidad de interacción con redes de mayor escala, como son LAN y WAN, lo anterior se logra mediante la utilización de protocolos compatibles en todas las redes, tales como IP. Como resultado de la compatibilidad de protocolos, es posible conectar los dispositivos portátiles con redes de área ancha a través de sistemas celulares que permitan la transmisión de datos (2.5 G y 3 G). Esta interacción brinda la posibilidad de ofrecer acceso a Internet [14] de manera inalámbrica con los dispositivos portátiles[1].

1.2 WPAN (Wireless Personal Area Network)

WPAN es una red inalámbrica de área personal que está regulada por el estándar 802.15 de la IEEE. Entre otros estándares inalámbricos controlados por la IEEE se encuentran los siguientes [25]:

- 802.11 Estándar Base
  - 2.4 GHz Espectro Disperso con Saltos en Frecuencia (1 Mbps)
  - 2.4 GHz Espectro Disperso con Secuencia Directa (2 Mbps)
  - Infrarrojo (1Mbps)
- 802.11a 5GHz (mayor a 20 Mbps)
- 802.11b 2.4GHz (mayor a 8 Mbps)
- 802.15 Redes Inalámbricas de Área Personal (WPAN)
- 802.16 Redes Inalámbricas de Área Local de Banda Ancha (LMDS)

El estándar de las redes WPAN está enfocado al establecimiento de conexión inalámbrica para la interoperabilidad entre dispositivos que se encuentren dentro de un espacio personal de

[1] En el anexo E se detalla el protocolo WAP.

operación. Estos dispositivos de cómputo portátiles o móviles son aparatos pequeños, para trabajo en red, que involucran software y periféricos que son usados, o portados, por las personas para incrementar su productividad, así como también para proveer entretenimiento [7]. Dentro de ésta clase de dispositivos se encuentran las PC's, PDA (Asistente Personal Digital), periféricos, teléfonos celulares, localizadores personales (*pagers*), y aparatos electrónicos para comunicarse e interoperar con otro usuario.

Se define una red inalámbrica de área personal como un sistema inalámbrico de comunicación de datos tipo *ad hoc* que permite la comunicación entre diversos dispositivos [7]. WPAN usa la banda del espectro *industrial científica y medica* (ISM) que se ubica en 2.4 GHz. El uso de esta banda se debe principalmente a su disponibilidad mundial, ya que es una banda en la que no se requiere licencia para aplicaciones secundarias.

Una red WPAN tipo *ad hoc* también conocida como *piconet* es una red compuesta únicamente por estaciones que se encuentren dentro de un rango de comunicación mutuo y cuya conexión es inalámbrica. Una *piconet* se puede crear de manera espontánea y sencilla, de tal modo que pueden ser operadas por usuarios no especializados. Una de sus principales caracteristicas es su limitada existencia temporal y espacial, estas limitaciones permiten crear y disolver este tipo de redes de la manera más conveniente para el usuario. Una WPAN es capaz de operar en un área de 10 metros de diámetro, sin embargo puede alcanzar hasta 100 metros si se selecciona un módulo de potencia de 20 dBm, en lugar de uno de 0 dBm.

Existen diversos requerimientos de una red inalámbrica de área personal. A continuación se mencionan los más importantes.

1.2.1 Enfoque de una red WPAN

- Ubicación en una banda del espectro no regulada (ISM, 2.4 GHz).
- Posibilidad de establecer un esquema de comunicación asincrono no orientado a conexión (ACL), para la transmisión de datos.
- Posibilidad de establecer un esquema de comunicación síncrono orientado a conexión (SCO), para la transmisión de voz.
- Coexistencia de múltiples WPAN's en la misma área.
- Coexistencia con otros sistemas inalámbricos, como WLAN's, dentro de la misma área.
- Conectividad con otro tipo de redes de datos.
- Soportar la caracteristica de calidad de servicio (QoS) con la finalidad de manejar diferentes tipos de tráficos.
- Acceso muy rápido a la red WPAN para los dispositivos dentro del rango de comunicación.
- Control de acceso a la red WPAN.

- Cada dispositivo de la WPAN debe tener la posibilidad de comunicarse con cualquier otro dentro de esta.
- Soporte de red para diversos tipos de dispositivos.
- Tasa de transmisión relativamente baja (≤ 1 Mbps).
- Rango de operación típico de 0-10 metros.
- Número relativamente pequeño de dispositivos activos.

1.2.2 Características de los elementos de una WPAN

- Bajo consumo de potencia.
- Bajo costo.
- Tamaño reducido.

1.3 *Bluetooth* y el estándar 802.15 de IEEE

Una especificación robusta y bien detallada es solo el primer paso para el establecimiento de una tecnología. En el caso de la tecnología de redes inalámbricas de área personal (WPAN) se requiere de un proceso formal de estandarización para asegurar la proliferación de soluciones de conexión inalámbrica para los nuevos dispositivos portátiles. La IEEE provee foros para el desarrollo de estos estándares.

La diferencia entre un estándar y una especificación es tanto sutil, como profunda. Es sutil debido a que los dos describen las características técnicas de un sistema, lo cual no hace sencillo observar la diferencia. Sin embargo, la principal diferencia está en como y porque son construidos.

Una especificación describe el funcionamiento de uno o un pequeño grupo de implementaciones de una determinada tecnología. Generalmente, las especificaciones asumen características de las implementaciones particulares, con frecuencia son de carácter narrativo en donde no se utiliza un lenguaje formal, como en el caso de los estándares.

Y [illegible]ia de las especificaciones, usualmente, los estándares son creados antes de la existencia física del sistema donde se aplicarán. La falta de dichos sistemas hace que un estándar posea descripciones no ambiguas, a través de un lenguaje formal y un documento rigurosamente estructurado.

Existen ocasiones en donde una compañía o consorcio inventa una tecnología de gran aceptación y establece especificaciones para ella. Dichas especificaciones pueden generalizarse y estandarizarse para convertirse en un estándar que sirva para futuros desarrollos de la tecnología. Ejemplos de este proceso son *Ethernet* y *Bluetooth*, como veremos a continuación.

A pesar de que los esfuerzos por estandarizar las redes WPAN, por parte de IEEE, anteceden al anuncio de la creación de *Bluetooth* por un año, no fue hasta la liberación de la

primera versión de las especificaciones *Bluetooth* (v1.0b) en 1999, que se consolidó el grupo de trabajo IEEE 802.15 el cual tenía la finalidad de crear un estándar para la redes WPAN.

Una vez consolidado el grupo 802.15, este se dio a la tarea de hacer llamados a los grupos interesados en participar en la creación de un estándar WPAN. Aunque existían 2 consorcios involucrados, *HomeRF* y *Bluetooth*, solamente el grupo de interés especial de *Bluetooth* (SIG) creado en 1998 respondió al llamado. Después de negociaciones se acordó que la especificación de *Bluetooth* v1.0b, sería la base para el estándar WPAN (802.15). Dentro de las negociaciones se convino que el estándar fuera totalmente compatible con las especificaciones *Bluetooth*, y con ello se aseguró una interoperabilidad con la versión presente y las futuras.

Finalmente, el estándar WPAN basado en *Bluetooth*[4] fue etiquetado por IEEE como 802.15.1. Posteriormente al establecimiento de este estándar, surgieron otros grupos de IEEE relacionados con las redes inalámbricas de área personal. En febrero de 2000, se creo el grupo 802.15.2, con la finalidad de emitir recomendaciones para la coexistencia e interoperabilidad entre redes WPAN 802.15 y redes WLAN 802.11. En marzo de 2000 surgió el grupo 802.15.3, con la finalidad de crear un estándar para redes inalámbricas de área personal de alta velocidad WPAN-HR. Recientemente apareció el grupo 802.15.4, dedicado a las redes inalámbricas de área personal de baja velocidad WPAN-LR. En la figura 1.1 se muestra un esquema que resume la familia 802.15 [20].

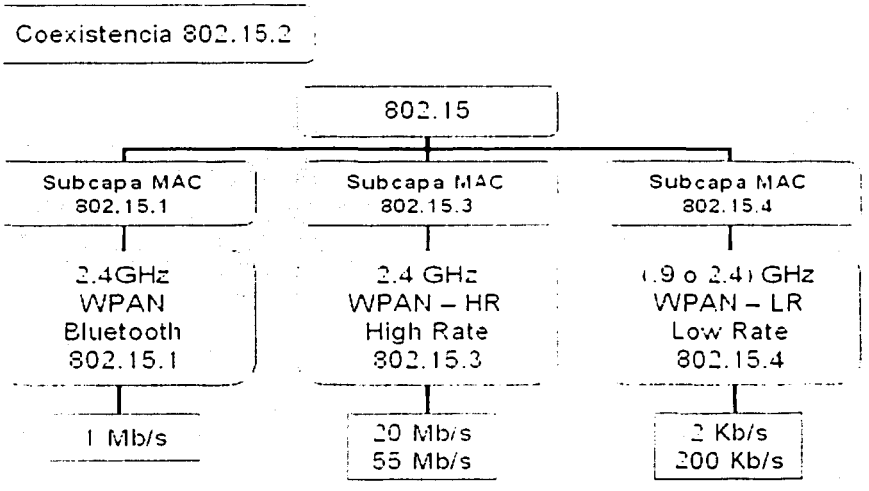


Figura 1.1 Familia 802.15 de IEEE

[4] Debido a que el estándar WPAN fue creado [illegible]

Una de las principales metas del estándar 802.15 de IEEE y de *Bluetooth* es el uso global. En contraste con las tecnologías WLAN que están específicamente diseñadas para dispositivos que se encuentran dentro y alrededor de la oficina o casa, un dispositivo WPAN/*Bluetooth* tiene la posibilidad de viajar de país en país, usarse en automóviles, aviones, barcos y está realmente diseñado para un uso internacional.

Debido a esto, la tecnología *Bluetooth* están enfocada en un estándar que reúne los requisitos internacionales dentro de las siguientes categorías: espectro, potencia y seguridad.

La seguridad es necesaria a causa de que el enlace de radio contendrá información privada de negocios o personal, datos o voz.

Respecto al espectro y la potencia, la tecnología requiere ser diseñada para viajar con el usuario, a diferencia de WLAN que se establece en un área y nunca se mueve, evitando así romper las leyes de gestión del espectro de cada lugar.

Bluetooth reúne las características anteriores para el trabajo en redes de área personal y es totalmente compatible con el estándar 802.15 de IEEE.

1.4 Coexistencia

En la banda ISM tanto las aplicaciones primarias como secundarias están permitidas. Las aplicaciones secundarias no requieren licencia pero deben seguir ciertas reglas relacionadas con el total de la potencia radiada y el uso de esquemas de modulación de espectro disperso. La interferencia entre las diversas aplicaciones no es significativa mientras se sigan las reglas establecidas. La desventaja de usar esta banda es que debe ser compartida, y por lo tanto, tolerar interferencia potencial.

Bajo circunstancias normales, los esquemas de espectro disperso y las reglas de máxima potencia radiada, tratan efectivamente el problema que existe cuando varias aplicaciones comparten la misma banda; sin embargo, si los dispositivos de diferentes aplicaciones que utilizan la misma banda se encuentran muy cercanos, interferirán entre si, de tal manera que dejarán de funcionar correctamente.

Entre los dispositivos que operan en la banda ISM están los hornos de microondas (2.45 GHz), WLAN's, *HomeRF*, *Bluetooth* y varios teléfonos inalámbricos. Estudios realizados indican que con una modesta separación física, todos estos dispositivos pueden coexistir con algún grado de degradación en su desempeño [15].

La coexistencia de diferentes aplicaciones depende del grado de interoperabilidad que existan entre ellas, es decir, de la capacidad de compartir información, de tal manera que el desempeño de las diferentes aplicaciones involucradas sea máximo.

En la figura 1.2 se muestra la ubicación, como función del rango de operación y ancho de banda, de varias aplicaciones para enlaces inalámbricos y tecnologías existentes.

Para hacer una comparación de las diversas tecnologías inalámbricas con la tecnología *Bluetooth*, la cual se explica en forma en el siguiente capítulo, se ha incluido un breve resumen en el anexo E, junto con referencias para su consulta.

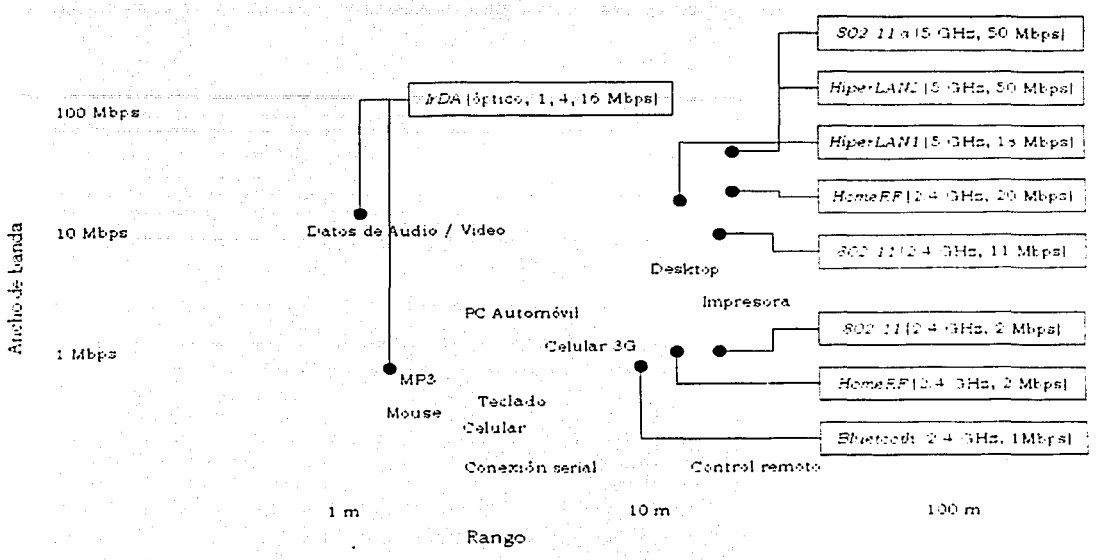


Figura 1.2. Aplicaciones inalámbricas y tecnologías

CAPÍTULO 2
LA TECNOLOGÍA INALÁMBRICA BLUETOOTH

Bluetooth es una tecnología para enlaces de radio de corto alcance, bajo costo y poco consumo de potencia. Originalmente *Bluetooth* fue diseñado para sustituir el cableado entre los dispositivos de comunicaciones móviles y las computadoras portátiles. Actualmente, el estándar define una estructura uniforme para que un amplio rango de dispositivos transfieran entre sí voz y datos, con un mínimo esfuerzo por parte del usuario.

Con esta nueva tecnología los usuarios no tendrán la necesidad de conectar, instalar, o configurar la conexión entre dos dispositivos, debido a que este estándar de comunicación inalámbrico provee un enlace sencillo y dinámico.

Bluetooth está encaminado a alcanzar una aceptación global tal que cualquier dispositivo con esta tecnología, en cualquier lugar en el mundo, pueda conectarse con otro que esté próximo sin importar el modelo, o fabricante.

Las especificaciones de *Bluetooth* definen de una manera general y abierta al sistema, desde la capa de *radio* hasta el nivel de aplicaciones. La pila de protocolos se implementa tanto en *hardware* como en *software* dentro de un microprocesador, las diferentes implementaciones asignan diferentes funciones a ambos.

2.1 Orígenes y desarrollo de Bluetooth

El desarrollo de esta tecnología empezó en 1994, cuando el grupo de comunicaciones móviles de *Ericsson* estudió las alternativas para reemplazar los cables que conectaban a los teléfonos móviles con sus accesorios.

Este estudio llevó a los enlaces de radio, debido a que estos no necesitan la gran directividad del infrarrojo (usado en equipos anteriores) por lo que tienen la ventaja de no requerir línea de vista para establecer comunicación. En esta investigación hubo que considerar varios requerimientos, tales como: el manejo de voz y datos para hacer posible

la conexión entre los teléfonos móviles, sus accesorios (*headsets*), y los dispositivos de cómputo.

Actualmente existe un grupo de interes especial (SIG), constituido por varias compañías encargadas de promover y definir las especificaciones de *Bluetooth*. El grupo se fundo en febrero de 1998, y la primera versión de las especificaciones de *Bluetooth* fue liberada en 1999. Debido a las ventajas de formar parte del SIG, éste ha tenido una gran aceptacion y muchas compañías se han vuelto miembros. En abril de 2000 se tuvieron registrados 1790 miembros.

El nombre de *Bluetooth* es en honor de un rey Vikingo del siglo décimo, Harald Blatand (*Bluetooth*), quien unificó y gobernó Dinamarca y Noruega. El nombre se adoptó debido a que esta tecnologia tiene como expectativa unificar la industria de las telecomunicaciones, con la de la computación

2.2 Objetivos de la tecnologia *Bluetooth*

La finalidad de *Bluetooth* es la implementación de una red inalámbrica de área personal, la cual no requiere una tasa de transferencia de datos grande, sin embargo exige un establecimiento dinámico y sencillo de la red.

A continuación se mencionan los objetivos particulares que debe cumplir *Bluetooth*, para que sea una tecnologia viable que satisfaga los requerimientos de una red inalámbrica de área personal:

- *Bluetooth* debe ser un estándar de aceptación universal que ofrezca los medios para el intercambio de información de un conjunto de dispositivos de diversos tipos de una manera sencilla y eficiente
- El estándar debe permitir a los dispositivos establecer conexiones *ad hoc*. "Conectividad espontánea", donde los dispositivos puedan conectarse a otros en su proximidad casi sin ninguna interaccion, o comando del usuario.
- Un objetivo importante en *Bluetooth* es la transmision tanto de voz como de datos de una manera segura.

Ademas de esto, es importante mencionar los retos técnicos más importantes que debe enfrentar *Bluetooth*:

- Al ser una tecnologia inalambrica que pretende reemplazar a los cables y sistemas basados en infrarrojo, *Bluetooth* requiere que su implementacion tenga un costo bajo, comparable con el precio de los dispositivos que va a reemplazar. Lo anterior obliga a que sea una tecnologia barata

- Debido a que *Bluetooth* esta diseñado para dispositivos móviles, debe ser capaz de consumir poca potencia y operar con voltajes bajos. Otra caracteristica que debe cumplir es un peso ligero y tamaño pequeño, para hacer posible su introducción dentro de aparatos portátiles compactos tales como teléfonos celulares, *headsets*, y asistentes personales digitales (PDA's).
- Finalmente otro aspecto importante dentro de *Bluetooth* es la seguridad, por lo que debe ser tanto, o más confiable que los dispositivos que reemplaza.

En suma, esta tecnologia pretende ser ampliamente disponible, conveniente, fácil de usar, confiable, segura, barata, pequeña y de bajo consumo de potencia.

2.3 Características

2.3.1 Física

Bluetooth opera en la banda industrial, científica y médica (ISM), la cual está disponible mundialmente sin requerir licencia para su uso. En esta banda se deben observar ciertas reglas relacionadas con la potencia y frecuencias trasmitidas, asi como con la interferencia.

Dadas las condiciones anteriores *Bluetooth* consta de un sistema robusto debido a que existen muchos usuarios y dispositivos "contaminantes" compartiendo esta banda. Para ello *Bluetooth* usa la técnica de espectro disperso[1], en la modalidad de saltos lentos en frecuencia. De esta manera se usa toda la banda ISM disponible y si una transmisión se encuentra comprometida por interferencia en un canal, la retransmisión se hará siempre en un canal diferente lo más alejado posible.

Además de utilizar la técnica de espectro disperso como acceso al medio, se utiliza multiplexaje por división de tiempo (TDM). Esto se realiza por medio de ranuras de tiempo en las cuales se transmite y recibe información, cada ranura tiene una duración de 625 μs [12].

Los dispositivos saltan a otro canal cada vez que transmiten un paquete, el cual puede tener una duración de 1, 3, o 5 ranuras de tiempo.

2.3.2 Arquitectura

Los dispositivos *Bluetooth* tienen una arquitectura basada tanto en *hardware* como en *software* [22].

[1] Un estudio detallado de la técnica de espectro disperso se presenta en el Anexo C.

El *hardware* que compone al dispositivo *Bluetooth* está compuesto por dos partes: un dispositivo de radio, encargado de modular y transmitir la señal; y un controlador digital. El controlador digital está compuesto por: un CPU, un procesador de señales digitales (DSP *Digital Signal Processor*) llamado *Link Controller* (controlador de enlace) y las interfaces con el dispositivo anfitrión. En la figura 2.1 se muestra la arquitectura de *hardware*.

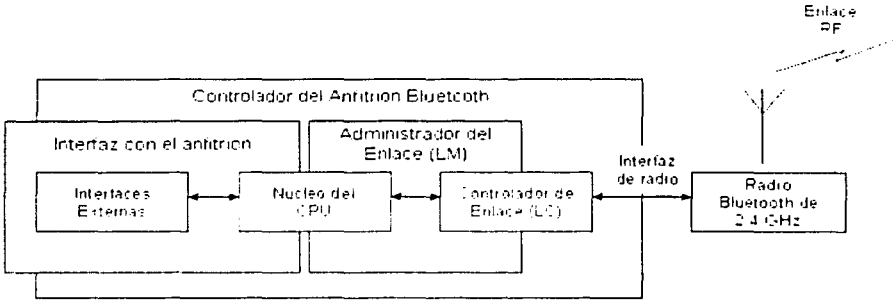


Figura 2.1. Arquitectura de *Hardware Bluetooth*.

En cuanto al *software*, buscando ampliar la compatibilidad de los dispositivos *Bluetooth*, los dispositivos que se apegan al estándar utilizan como interfaz entre el dispositivo anfitrión (*laptop*, teléfono celular, etc) y el dispositivo *Bluetooth* como tal (chip *Bluetooth*) una interfaz denominada HCI (*Host Controller Interface*).

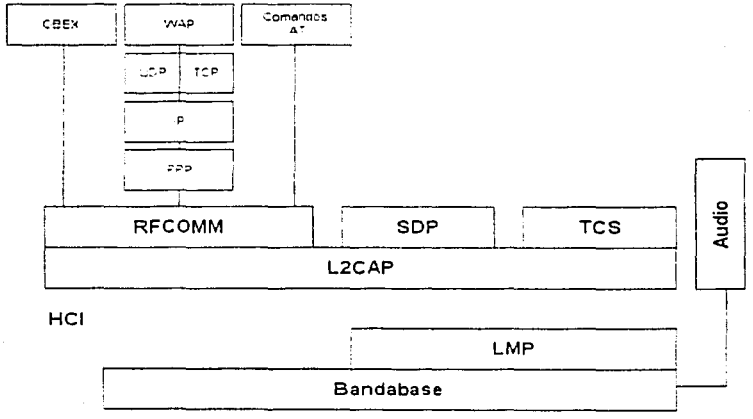


Figura 2.2. Arquitectura de *Software Bluetooth*.

Los protocolos de alto nivel como el SDP (protocolo utilizado para encontrar otros dispositivos *Bluetooth* dentro del rango de comunicación, encargado también, de detectar la función de los dispositivos en rango), RFCOMM (protocolo utilizado para emular las conexiones de puerto serial) y TCS (protocolo de control de telefonía) interactúan con el controlador de bandabase a través del Protocolo L2CAP (*Logical Link Control and Adaptation Protocol*). Más adelante, se detallan las funciones de estos protocolos ubicándolos en una serie de capas que conforman la pila de protocolos *Bluetooth*. En la figura 2.2 se muestra un esquema de la arquitectura de *software*.

2.3.3 Maestros, Esclavos, ranuras y saltos en frecuencia

Debido a que los dispositivos saltan a una nueva frecuencia después de transmitir cada paquete, es necesario que exista una coordinación con respecto a la secuencia de frecuencias que se usarán.

Los dispositivos *Bluetooth* puede operar en dos modos: Maestro o Esclavo. El Maestro establece la secuencia de saltos en la frecuencia, mientras el Esclavo se sincroniza al Maestro tanto en el tiempo, como en la frecuencia y sigue la secuencia de saltos de éste.

Cada dispositivo *Bluetooth* tiene una dirección y reloj únicos, a partir de los cuales se genera la secuencia de saltos.

Cuando los Esclavos se conectan al Maestro, éstos reciben la dirección y reloj de éste, con los que calculan la secuencia de salto y de esta manera se alcanza una sincronización con la secuencia del Maestro.

Además de controlar la secuencia de salto, el Maestro determina cuándo deben transmitir los demás dispositivos. El Maestro permite la transmisión a los Esclavos asignándoles ranuras de tiempo para tráfico de voz, o datos.

Cuando se trata de ranuras de tráfico de datos, los Esclavos solamente tienen permitido transmitir cuando en una ranura anterior el Maestro se haya comunicado con ellos, es decir, se limitan a responder al Maestro.

En la transmisión de voz, se requiere que los Esclavos transmitan periódicamente en ranuras reservadas, independientemente de que se trate o no de una respuesta al Maestro.

El Maestro controla la forma de distribución del ancho de banda disponible entre los Esclavos, al decidir cuándo y con qué periodicidad se comunica con cada uno de ellos.

El número de ranuras de tiempo que cada dispositivo utiliza depende de las necesidades de transmisión de datos.

2.3.4 Piconets y scatternets

Al conjunto de Esclavos que interactúan entre sí, y poseen un Maestro común, se denomina *piconet*. Todos los dispositivos dentro de la *piconet* están sincronizados con la frecuencia de salto y reloj del Maestro. Dentro de una *piconet* puede existir comunicación punto a punto, en el caso que sólo esté constituida por un Esclavo y el Maestro; también puede existir comunicación punto a multipunto, cuando existe más de un Esclavo. Dentro de la red los Esclavos establecen comunicación solamente con el Maestro, por lo que no existe una comunicación directa entre Esclavos. En la figura 2.3 se muestran una *piconet* punto a punto, y otra, punto a multipunto.

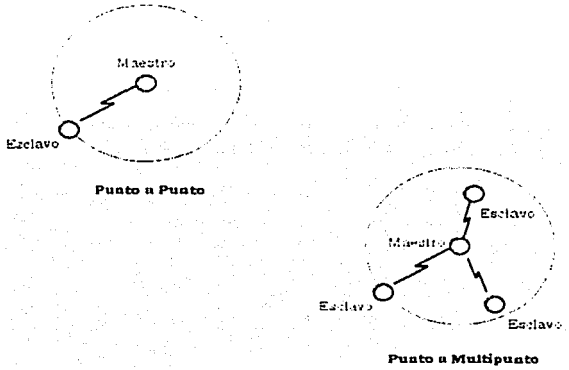


Figura 2.3. *Piconet* punto a punto y punto a multipunto

Bluetooth establece como límite máximo 7 Esclavos activos por *piconet*, sin embargo se puede formar una red más grande enlazando varias *piconets* dentro de una red de mayor tamaño, llamada *scatternet*, donde algunos dispositivos son miembros de más de una *piconet* al mismo tiempo (figura 2.4).

Si restringimos el caso a dos *piconets*, puede suceder que un dispositivo sea Maestro en una y Esclavo en otra (figura 2.4 lado derecho), o bien que sea Esclavo en ambas (figura 2.4 lado izquierdo), pero es imposible que sea Maestro en las dos. Un dispositivo no puede ser Maestro en más de una *piconet* debido a que, dado el supuesto caso, todos los Esclavos se sincronizarían con el Maestro, por lo que se trataría de la misma *piconet*, y no de dos diferentes.

La mayor fuente de interferencia para un dispositivo *Bluetooth* es otro que no esté sincronizado, por lo que *piconets* diferentes que comparten la misma área, tendrán colisiones aleatorias en la misma frecuencia. Si existe una colisión en un canal específico,

esos paquetes se perderán; en el caso de paquetes de datos, éstos serán retransmitidos en un canal diferente, lo más alejado posible; en el caso de paquetes de voz, serán ignorados.

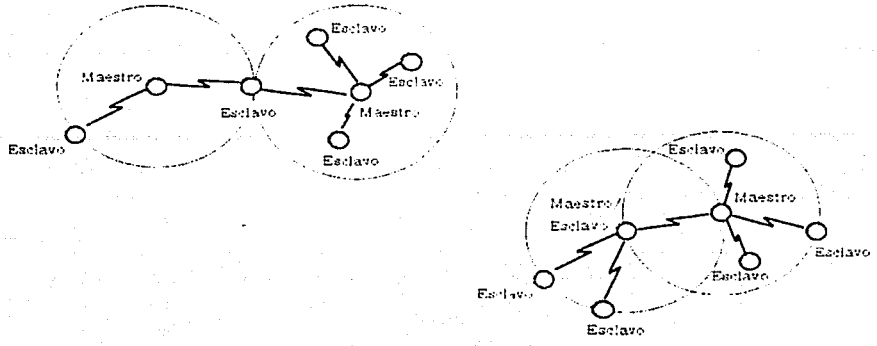


Figura 2.4. Redes *scatternet*

El número de *piconets* dentro de un área común afecta directamente la tasa de transferencia de datos, a mayor número de redes será menor el desempeño de cada sistema. Esto es independiente de la formación de *scatternets*, pues las *piconets* que la componen no están coordinadas.

Clasificación de emisión de potencia de radio

Existen 3 diferentes clases de emisión de potencia definidas para esta tecnologia, éstas proveen rangos de operación de 10 m, 20 m y 100 m. El menor rango de operación lo proporciona la clase 3, que emite 1 mW; a continuación se encuentra la clase 2, que emite 2.5 mW y por último la clase 1, con 100 mW.

Como en toda emisión de radio, un factor que afecta de manera importante el rango de operación son los obstáculos entre los dispositivos, por lo que no es conveniente trabajar en el límite de los rangos.

De la misma manera como existe un rango máximo, también existe una distancia mínima de trabajo de los dispositivos que generalmente es de 10 cm. Si la distancia entre los dispositivos se hace más corta que la mínima se corre el riego de saturarlos y tener un mal funcionamiento.

Es posible establecer comunicación entre dispositivos que posean diferentes clases de emisión de potencia. Este hecho se ilustra en la figura 2.5.

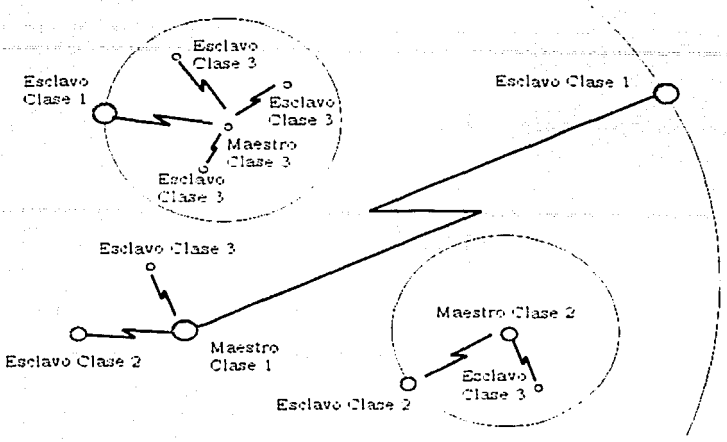


Figura 2.5. *Piconets* formadas con dispositivos de diferentes clases de emisión de potencia

2.3.5 Enlaces de voz y datos

Bluetooth permite la transmisión de voz y datos al mismo tiempo, para ello se definen dos tipos diferentes de enlaces: síncrono orientado a conexión (SCO) y asíncrono no orientado a conexión (ACL). El enlace ACL está enfocado a la transmisión de paquetes de datos no sensibles a retardos en el tiempo. Este tipo de enlace puede ser simétrico o asimétrico relativo a la tasa de transferencia. En el caso asimétrico, la máxima tasa de transmisión en un sentido es 723.2 Kbps y en el otro 57.6 Kbps; para el caso simétrico la tasa es de 433.9 Kbps en ambos sentidos. La capacidad máxima aproximada de datos útiles que puede manejar este enlace es 650 Kbps [12].

Los enlaces SCO están diseñados para la transmisión de datos de voz o audio, los cuales son sensibles a retardo en el tiempo. Este enlace trabaja a 64 Kbps, y permite tener hasta tres enlaces *full-duplex* de voz a la vez, o mezclar voz y datos. La calidad de voz que se puede obtener es la que se esperaría de un sistema de telefonía celular como GSM. *Bluetooth* puede soportar la transmisión de música si se utiliza algún tipo de compresión de datos adecuado, tal como MP3.

2.3.6 Seguridad

Bluetooth utiliza la técnica de espectro disperso que se basa en la generación de saltos pseudoaleatorios en frecuencia a una alta velocidad, por lo que es difícil de detectar, o interceptar la comunicación.

Por otro lado, para la encriptación y autenticación se usa un algoritmo de dominio público llamado SAFER+, el cual genera llaves de cifrado de 128 bits, a partir de una entrada de texto de 128 bits.

2.4 Aplicaciones

Debido a que *Bluetooth* es una tecnología inalámbrica capaz de transmitir voz y datos dentro de un espacio relativamente pequeño, se vuelve ideal para la conexión entre dispositivos móviles, o entre un dispositivo móvil y uno fijo. Algunas de las aplicaciones más comunes podrían ser [18]:

- Conexión de un teléfono celular a la red de telefonía pública (PSTN) a través de un punto de acceso *Bluetooth*.
- Conexión de un teléfono celular a una computadora portátil o *laptop*.
- Conexión de un teléfono celular con accesorios tales como *headsets*.
- Punto de acceso a redes LAN para *laptops* y *palmtops*.
- Puerto de salida a Internet a través de la red de telefonía pública.
- Conexión entre *laptops* y *palmtops*.
- Conexión entre *laptops* y computadoras de escritorio.

2.5 Implementación

El asunto clave para la implementación es la partición *hardware/software*, dicha separación de funciones dentro del subsistema de *Bluetooth* involucra desempeño, costo y consumo de potencia.

La partición se da entre el sistema del *host* y el subsistema de *Bluetooth*, una implementación con mayores funciones en *software* y menos en *hardware*, representa para el *host* una mayor carga de procesamiento; por el contrario si se hace una implementación mayor en *hardware*, esta incrementará la complejidad y el costo del sistema *Bluetooth*.[2]

La implementación de *Bluetooth* como un sistema embebido requerirá un diseño óptimo, que considere las características específicas de cada dispositivo, para evitar crear un producto no comercial.

2.6 Pila de protocolos

Una característica clave de este tipo de tecnología es que ofrece la posibilidad de conexión entre una gran variedad de dispositivos de diversos fabricantes. Con este fin, *Bluetooth*

[2] Se han fabricado ambos tipos de dispositivos Bluetooth y se encuentran disponibles en forma individual o junto con *kits* de desarrollo y prueba

define tanto un sistema de radio, como un conjunto de protocolos que permiten a las aplicaciones encontrar otros dispositivos *Bluetooth* en el área, descubrir qué tipo de servicios ofrecen y usar esos servicios.

La pila de protocolos *Bluetooth* esta definida como una serie de capas con funciones propias, sin embargo existen algunas funciones en las que intervienen varias capas.

A continuación se enuncian las principales funciones de cada capa y en la figura 2.6 se muestra la pila de protocolos. Un estudio detallado de los protocolos de *Bluetooth* se realiza en los dos capítulos subsecuentes.

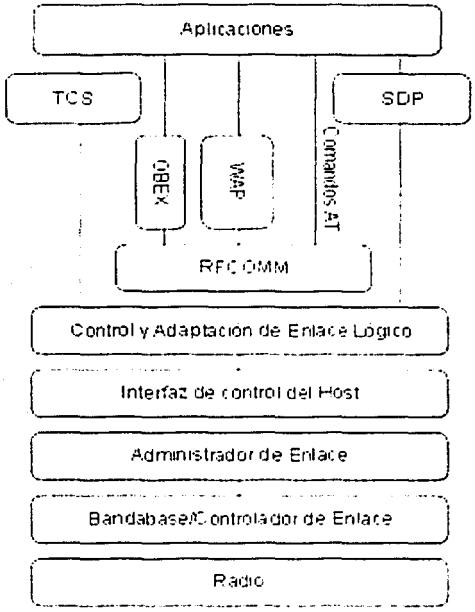


Figura 2.6. Pila de protocolos *Bluetooth*

La capa de radio modula y demodula los datos para la transmisión y recepción en el aire.

La bandabase y el controlador de enlace (LC) controlan los enlaces físicos vía *radio*, ensamblan paquetes y regulan los saltos en frecuencia. El manejo de los paquetes en el enlace inalámbrico es responsabilidad de la *bandabase*. Se pueden establecer dos tipos de conexiones:

- SCO (*Synchronous Connection Oriented*). Para información síncrona, como es la voz.

- ACL (*Asynchornous Connection Less*). Para aplicaciones de transferencia de información que no es susceptible a la variación de retardo en el tiempo.

La *bandabase* establece las funcionalidades para que los dispositivos sincronicen sus relojes y establezcan la conexión.

El protocolo de administración de enlace (LMP) controla y configura los enlaces con otros dispositivos. Básicamente, las funciones del LMP pueden clasificarse como:

- Administración de *Piconets*
- Configuración de enlace
- Funciones de seguridad

Este protocolo da la funcionalidad para añadir y quitar Esclavos, intercambiar papeles entre un Maestro y un Esclavo y establecer enlaces ACL/SCO. El LMP también maneja los modos de bajo consumo de potencia, diseñados para ahorrar energía cuando el dispositivo no tiene datos que enviar.

Las tareas de configuración de enlace incluyen la negociación de parámetros para el establecimiento de enlace, calidad de servicio y control de potencia en el caso de que el dispositivo lo soporte. También maneja los procesos de autenticación y administración de llaves de enlace.

La interfaz de control del host (HCI) se encarga de la comunicación entre un anfitrión (*host*) independiente y el módulo de *Bluetooth*. Para muchos dispositivos, el modulo *Bluetooth* se agrega como una tarjeta por separado, por ejemplo en una PC, o una laptop, etc. Los módulos *Bluetooth* implementan usualmente las capas inferiores como son *radio*, *bandabase*, LC y LMP. La información que se envía hacia dichas capas inferiores viaja a través de un bus físico, como por ejemplo, el USB. Un controlador para este bus se requiere en el anfitrión, esto es, la PC; y una interfaz de control se requiere en la tarjeta de *Hardware* de *Bluetooth* para aceptar datos del bus físico. Así, las capas superiores, como L2CAP y otras, residen en el *software*; y las inferiores, en el *hardware*.

El protocolo de control de enlace lógico y adaptación (L2CAP), multiplexa los datos de capas superiores, y realiza conversiones entre diferentes tamaños de paquetes. Las funciones básicas son las siguientes:

- Multiplexaje: El protocolo permite que múltiples aplicaciones usen una misma conexión entre dos dispositivos simultáneamente.
- Segmentación y reensamblaje: El protocolo adecua el tamaño de los paquetes que entregan las aplicaciones al tamaño de los paquetes aceptados por la *bandabase*. En sí, L2CAP acepta tamaños de paquetes de más de 64Kb, pero los paquetes de *bandabase* pueden aceptar una carga útil de solo 2745 bits. El procedimiento inverso, el de combinar los paquetes segmentados en el orden correcto, tiene que ser llevado a cabo con los paquetes recibidos.

- Calidad de Servicio: L2CAP permite aplicaciones que demanden QoS (Quality of Service), en ciertos parámetros como ancho de banda, latencia y variación de retraso. L2CAP verifica si el enlace es capaz de proveerlo, y si es posible, lo provee.

RFCOMM emula un RS232 como interfaz serial.

WAP y OBEX proveen interfaces para las capas altas de otros protocolos de comunicación.

SDP (*Service Discovery Protocol*) permite a los dispositivos *Bluetooth* descubrir, qué servicios soporta el otro aparato con el que se pretende establecer conexión.

TCS (*Telephony Control Protocol Specification*) provee servicios de telefonía.

La capa de Aplicación. La capa L2CAP debe ser alcanzada directamente por las aplicaciones, o a través de ciertos protocolos de soporte como lo son RFCOMM, TCS y SDP. Las aplicaciones usan otros protocolos como TCP-IP, o WAP, y *Bluetooth* permite que operen entre sí. Las aplicaciones se desarrollan con PPP (Protocolo Punto a Punto), FTP (Protocolo de transferencia de archivo), u otros protocolos específicos requeridos por la aplicación. Una aplicación debe usar el SDP (Protocolo de Descubrimiento de Servicios) para saber qué servicio se encuentra disponible.

Algunos modelos de uso, propuestos por muchos fabricantes, son los siguientes:

- Múltiples servicios en un teléfono: Un simple auricular puede funcionar como un intercomunicador en la oficina, como un punto de acceso a la red PSTN, o bien como un teléfono móvil.
- El maletín mágico: Debido a que el enlace de radiofrecuencia no necesita línea de vista, un teléfono móvil puede conectarse a una laptop aún cuando ésta se encuentre en un maletín, permitiendo el acceso a las facilidades como el correo electrónico.
- El sincronizador automático: La conectividad inalámbrica entre PDA's, laptop, PC y teléfonos móviles, permitirá a las aplicaciones ponerse al corriente y sincronizar calendarios y otros datos cuando se hacen modificaciones en algún dispositivo.
- Auriculares inalámbricos: Permitirán acceso a los teléfonos móviles de los usuarios, así como servicios de audio, mientras los dispositivos están en el bolsillo del usuario. Esto permite la operación sin necesidad de utilizar las manos.
- Equipo para automóviles: Dispositivos con "manos libres" permitirán a los usuarios acceso a sus teléfonos sin que intervengan las manos.

Además de todo esto, existe una gran variedad de otras aplicaciones en la automatización del hogar, intercambio de datos durante reuniones sin el uso de equipo extra, sistemas de seguridad, acceso a la red en lugares públicos, entre otros.

2.7 Comparación con el modelo de referencia OSI

La pila de protocolos de *Bluetooth* no se adapta exactamente al modelo de referencia OSI (*Open Systems Interconnect*), sin embargo, es posible realizar una comparación entre ambos sistemas a partir de las funciones en común. Dicha comparación se realiza a continuación y se muestra en la figura 2.7.

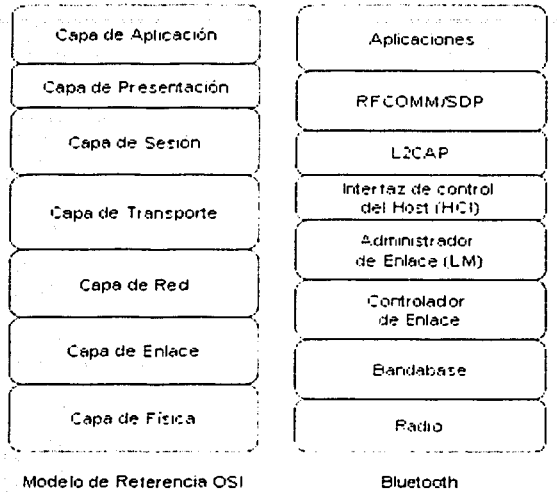


Figura 2.7. Comparación entre el sistema de referencia OSI y la pila de protocolos *Bluetooth*

En el modelo de referencia OSI **la capa física** es responsable de la interfaz entre la parte eléctrica y el medio, incluyendo la modulación y la codificación de canal. Por tanto cubre la capa de *radio* y parte de la *bandabase*.

La capa de enlace maneja la transmisión, los paquetes, y el control de errores sobre un enlace en particular, y es por eso que se traslapa con las tareas de *controlador de enlace* y la parte de control de *bandabase*, incluyendo la revisión y corrección de errores.

La capa de red controla la transferencia de datos a través de la red, independientemente del medio y la topología de la red. Ésta abarca la parte alta del *controlador de enlace*, estableciendo y manteniendo múltiples enlaces, y también cubre la gran parte de las tareas del *administrador de enlace*.

La capa de transporte es responsable de la integridad y multiplexaje de los datos a través de la red hasta el nivel de las aplicaciones, y por esto se traslapa con la parte alta

del *administrador de enlace* y cubre la *interfaz de control del host*, la cual provee los mecanismos de transporte.

La capa de sesión proporciona la administración y control de flujo de los datos. En *Bluetooth* estas funciones son realizadas por L2CAP y la parte baja de RFCOMM y SDP.

La capa de presentación provee una representación común para los datos de la capa de aplicación. Dentro de *Bluetooth* dicha tarea esta a cargo de RFCOMM/SDP.

Finalmente, **la capa de aplicaciones** es responsable de manejar las comunicaciones entre las aplicaciones de los *hosts*.

2.8 Bosquejo del establecimiento de una conexión Bluetooth

Para ejemplificar el establecimiento de una conexión *Bluetooth* consideremos a una persona que llega a un salón de exposiciones y quiere tener acceso a su correo electrónico mediante un dispositivo *Bluetooth*, el cual puede ser una *laptop*, o un asistente digital personal. El procedimiento específico a seguir depende de la implementación, sin embargo, se pueden generalizar los pasos que el dispositivo llevará a cabo. Estos son:

- *Inquiry*. El dispositivo que encuentre un nuevo entorno automáticamente iniciará una búsqueda para descubrir qué puntos de acceso se encuentran en su rango. Durante este proceso, todos los puntos de acceso cercanos responderán con sus direcciones y posteriormente, el dispositivo elegirá alguno.
- *Paging*. De este proceso resulta una sincronización del dispositivo con el punto de acceso, en términos de compensación de fase del reloj, secuencia de saltos de frecuencia y otros parámetros requeridos.
- Establecimiento del enlace. El protocolo de administración de enlace (LMP) establecerá un enlace con el punto de acceso. Como en este caso la aplicación es de correo electrónico, se usará un enlace asincrono (ACL). Antes de acceder al servicio se llevan a cabo los pasos que se describen a continuación:
- Descubrimiento de Servicios. El protocolo de administración de enlace (LMP) usará el protocolo de descubrimiento de servicios (SDP) para recabar información sobre los servicios que están disponibles desde el punto de acceso.
- Establecimiento del canal L2CAP. Con la información obtenida por el SDP, el dispositivo creará un canal L2CAP al punto de acceso. Este canal será utilizado directamente por la aplicación, o por otro protocolo como el RFCOMM.
- Canal RFCOMM. Dependiendo de la necesidad de la aplicación de correo electrónico un canal RFCOMM, u otro diferente, se creará a través del canal L2CAP. Esta característica permite a aplicaciones existentes diseñadas para puertos seriales, funcionar sin modificación en plataformas *Bluetooth*.

- Seguridad. Si el punto de acceso restringe su acceso a un grupo particular de usuarios, u ofrece comunicaciones seguras a gente con cierta prioridad, entonces el punto de acceso enviará una petición de seguridad para que se pueda hacer el establecimiento del enlace. La conexión se establecerá sólo si el usuario conoce la clave de acceso al servicio. Posteriormente, el mecanismo de encriptación será invocado si el modo seguro se está utilizando.
- Protocolo Punto a Punto (PPP). Asumiendo que un enlace PPP es utilizado a través de un módem serial en una red de *dial up*, la misma aplicación podrá trabajar PPP sobre RFCOMM (el cual emula un puerto serial). Este enlace permitirá al usuario ingresar a su correo electrónico.
- Protocolos de red. Los protocolos de red como TCP/IP, IPX, Appletalk pueden enviar y recibir datos sobre el enlace.

Es importante decir que para que se pueda establecer una conexión utilizando la tecnología inalámbrica *Bluetooth*, ambas terminales tienen que tener la disposición para conectarse. Cada dispositivo y la aplicación que lo maneje, cuentan con modos para ser descubiertos y modos para poder realizar la conexión. Estos modos se explicarán más detalladamente en los siguientes capítulos.

La capa de control de enlace hace que los dispositivos trabajen por medio de estados. Algunos de estos estados son los siguientes: *inquiry* o *búsqueda*, *inquiry scan* o *exploración de búsqueda*, *page* o *llamado* y, *page scan* o *exploración de llamado*. Los dos primeros sirven para que los dispositivos puedan ser descubiertos; mientras que los otros dos, sirven para que los dispositivos puedan establecer conexión. Todo esto se explicará más a detalle en los siguientes capítulos.

En este capítulo se ha presentado una visión general de *Bluetooth*, la cual incluye a la pila de protocolos que componen las especificaciones de esta tecnología. En el siguiente capítulo se realizará un estudio profundo sobre los procedimientos y parámetros necesarios para que cada capa realice sus funciones en la parte inferior de la pila de protocolos *Bluetooth*.

CAPÍTULO 3
PROTOCOLOS DEL MÓDULO BLUETOOTH

3.1 Radio

Los dispositivos *Bluetooth* operan en la banda Industrial, Científica y Médica ISM (2.4GHz), la cual está disponible mundialmente y no requiere de licencia para su uso. Dicha banda tiene restricciones en algunas regiones geográficas como son: Japón, España y Francia; por lo que, para armonizar estas regulaciones mundialmente, el SIG[1] se encuentra participando activamente con cuerpos políticos.

La banda ISM se encuentra ocupada por una gran cantidad de otros emisores de radiofrecuencia que abarcan desde aplicaciones inalámbricas, como: las técnicas de corto alcance (seguridad automotriz, teléfonos inalámbricos, etc.) y aplicaciones WLAN estandarizadas; hasta los diversos generadores de ruido como los hornos de microondas y las lámparas de vapor de sodio. En la tabla 3.1 podemos ver las frecuencias en las que se encuentra la banda ISM.

Área Geográfica	Banda ISM (GHz)	Banda de guarda inferior (MHz)	Banda de guarda superior (MHz)	Canales Disponibles
Francia, España y Japón	2.4465 a 2.4835	7.5	7.5	23
Resto del mundo	2.4000 a 2.4835	2	3.5	79

Tabla 3.1. Banda ISM.

Como se puede ver, la banda de 2.4 GHz no es un medio nada estable o confiable. Sin embargo, el hecho de que sea mundialmente disponible asegura una aceptación bastante razonable para el caso de la tecnología *Bluetooth*, la cual emplea varias técnicas para luchar

[1] Special Interest Group

contra esta interferencia, entre las que se encuentran principalmente: espectro disperso[2], control adaptable de potencia y paquetes cortos de datos.

3.1.1 Antenas

Los dispositivos *Bluetooth* requieren antenas cuyo patrón de radiación sea lo más cercano al esférico, de tal manera que puedan establecer conexión en cualquier dirección y ángulo.

A manera de referencia, para dispositivos LAN se pueden utilizar antenas más direccionales debido a que cubren áreas más especificas.

Para la elección de la antena óptima se deben considerar las características particulares del dispositivo *Bluetooth* donde se instalará. Además del espacio y costo, debe tenerse en cuenta que los elementos adyacentes, los empaques, el tipo de alimentación, los planos de tierra y la posición, afectan el desempeño de una antena [5].

El tipo de antenas más común para esta tecnología es la micro cinta o *microstrip*, la cual proporciona la ventaja de no ocupar volumen ya que es impresa sobre la tarjeta o chip.

3.1.2 La técnica de *salto de frecuencia*

Esta técnica ha sido utilizada en las comunicaciones de sistemas que requieren tanto seguridad como robustez, por lo que se convierte en una técnica ideal para *Bluetooth*, dado que siempre existe la posibilidad de que un canal se encuentre temporalmente bloqueado por una fuente de interferencia dentro de la banda ISM. Aunque *Bluetooth* provee un esquema de retransmisión para paquetes de datos perdidos, es definitivamente más eficiente y robusto retransmitir la información en un nuevo canal alejado del actual, el cual es más improbable que también se encuentre bloqueado. El algoritmo que se utiliza asegura una distancia máxima entre canales adyacentes de salto en la secuencia utilizada. Además, muchas *piconets* activas podrían estar dentro del rango de cualquier otra; en donde cada *piconet* hace el salto independiente con una secuencia pseudoaleatoria basada en el código de acceso/identidad correspondiente, las colisiones serán mínimas. Es importante mencionar que todos los dispositivos *Bluetooth* tienen 79 canales en los cuales operar.

3.1.3 La modulación

La banda operativa de *Bluetooth* va de 2.4000-2.4835 GHz, tiene una banda de guarda inferior de 2 MHz y una banda de guarda superior de 3.5 MHz, la banda *útil* esta dividida en 79 canales espaciados 1 MHz. Los datos son señalizados a 1 mega símbolo por segundo para obtener el máximo ancho de banda del canal. Con el esquema de modulación elegido, de GFSK (*Gaussian Frequency Shift Keying*), esto resulta en 1 Mbps. El uno binario da una desviación positiva en

[2] Un estudio detallado de la técnica de espectro disperso se encuentra en el Anexo C de esta tesis.

frecuencia de la frecuencia nominal de la portadora, mientras que el cero binario da una desviación negativa. En ambos casos la desviación está en el rango de 140-175 KHz [1].

Para obtener el uso más eficiente del ancho de banda mientras se mantiene una probabilidad de error aceptable, la secuencia digital de bits es modulada utilizando GFSK con un factor BT de 0.5 y un índice de modulación entre 0.28 y 0.35. El factor BT es el producto de la separación[1] en frecuencia de señales adyacentes (0.5MHz) y la duración de un símbolo de (1μs). Un producto BT de 0.5 corresponde a la mínima separación de la portadora para asegurar la ortogonalidad (sin correlación cruzada) entre señales de canales adyacentes. El índice de modulación representa la magnitud del pico de la frecuencia de desviación (f_d) y puede ser expresado como $2f_dT$, donde T es la duración del símbolo. Esto se traduce en un rango de desviación de frecuencia de 140 KHz a 175 KHz.

La modulación GFSK emplea un filtro gaussiano para suavizar las transiciones de frecuencia, así la frecuencia modulada de la portadora cambia suavemente con una envolvente de forma gaussiana. Esto mantiene una fase continua de la frecuencia de la portadora y reduce los lóbulos laterales espectrales emitidos, permitiendo una eficiencia espectral mucho mejor y una menor interferencia entre símbolos. El filtro gaussiano actúa en la transmisión de la secuencia de bits y puede ser implementado en la parte de radio como un filtro analógico, o implementado en la parte digital de la *bandabase* utilizando un filtro FIR.

3.1.4 Sincronización de símbolos

La especificación *Bluetooth* requiere de una sincronización de símbolos con una precisión de ±20 ppm. Es decir, que el reloj que maneja el procesamiento de símbolos en *bandabase* debe ser preciso en todas las posibles condiciones de operación y durante toda su vida operativa. Con la tecnología de cristal de cuarzo disponible, esto no representa ningún problema, ni incrementa el costo de los dispositivos *Bluetooth*.

El resultado de este requerimiento de precisión sobre el paquete más largo que se puede manejar (DH5), da como resultado 0.12μs, lo cual es menos que un octavo del periodo de símbolo. Durante la siguiente recepción, la sincronización de símbolo se hará nuevamente. Por esta razón esquemas de recuperación de sincronización son posibles para *Bluetooth*, sin la necesidad de un registro continuo de la trama de información que se encuentra transmitiéndose.

[1] Separación entre miembros del alfabeto de modulación o la separación de canal entre tonos modulados

3.1.5 Control y emisión de potencia

Las regulaciones de la FCC[4] permiten el uso de la banda ISM sin la utilización de la técnica de espectro disperso para emisiones menores a 0 dBm. Para emisiones de mayor potencia, el esquema de espectro disperso debe utilizarse. Por medio del salto de frecuencia, *Bluetooth* puede operar hasta los 20 dBm, permitiendo una distancia de operación de 100 m.

En la especificación *Bluetooth* se encuentran definidas tres clases de potencia. La clase 3 es la que más adoptan los fabricantes y por supuesto es la opción con menor potencia. La clase 1 tiene un requerimiento de control de potencia, mientras que en las clases 2 o 3 es opcional. Sin embargo, para un consumo mínimo de potencia, se prefiere la implementación de dicho control de potencia.

Clase de Potencia	Potencia de Salida (max)	Mínima potencia de Salida	Control de Potencia
1	100 mW (20 dBm)	1 mW (0 dBm)	Obligatoria: 4 dBm a 20 dBm
			Opcional: -30 dBm a 4 dBm
2	2.5 mW (4 dBm)	0.25 mW (-6 dBm)	Opcional: -30 dBm a 4 dBm
3	1 mW (0 dBm)	---	Opcional: -30 dBm a 4 dBm

Tabla 3.2. Clases de Potencia de Transmisión

El control de potencia opera por medio de un receptor RSSI[5] que monitorea la señal recibida, y envía comandos de control al transmisor de otro dispositivo, pidiendo que se reduzca la potencia de transmisión si el valor de potencia de la señal recibida es más alto que el necesario para mantener un enlace satisfactorio, y pidiendo que se aumente en caso contrario. La especificación indica que la potencia debe ser controlada en pasos de 2 a 8 dB, mientras que las mediciones del RSSI deben tener una precisión de ±4 dB a -60 dBm [22].

3.1.6 Parámetros de rendimiento de radio

Las especificaciones de *Bluetooth* establecen una operación con una máxima tasa de bits en error BER = $1*10^{-3}$, es decir 0.1%. Este BER es alcanzado con una sensibilidad de recepción de -70dBm. El establecimiento de la sincronización es un parámetro importante de desempeño en cualquier sistema. En la figura 3.1 se muestra la temporización de las operaciones durante cualquier proceso de Tx/Rx. El procesador encargado de los protocolos de la capa más baja debe decidir en qué estado poner al *controlador de enlace* de *bandabase* y programar varios parámetros asociados. Después, la *bandabase* puede calcular la frecuencia apropiada de un

[4] Federal Communications Commission
[5] Received Signal Strength Indication

canal y programar el sintetizador. Esto debe ocurrir mucho antes de que el inicio de la información llegue, para permitir que el sintetizador se sincronice. Todo esto impone un límite en la sincronización del sintetizador de aproximadamente 180 µs, eso no representa problema, pues algunos radios tienen un tiempo de sincronización mucho menor, que varía de 130 µs a 170 µs [5].

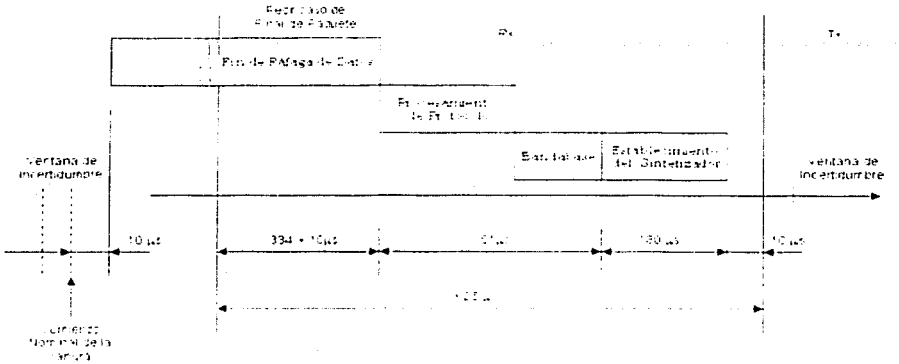


Figura 3.1. Requerimientos de sincronización del sistema

El mecanismo óptimo de recuperación depende de la implementación empleada. Dentro de *Bluetooth* existen dos variantes en la recuperación del reloj: la primera se orienta a la recuperación del reloj basándose en el conocimiento de los cuatro bits de preámbulo más el primer bit de la *palabra de sincronización* (syncword); la segunda opción se basa en la detección de los bordes del flujo de datos de entrada mediante un circuito PLL, a este tipo de sincronización suele llamársele "*early-late*".

En cualquier caso, el componente importante para la recuperación del reloj son los 5 bits de la secuencia inicial. En la práctica, la mayoría de los sistemas de radio suelen perder, incluso, los 3 primeros símbolos. Lo anterior ocasiona que se tengan que tomar más bits de la *palabra de sincronización* para llevar a cabo la recuperación del reloj, con lo cual se reduce la sensibilidad del enlace.

3.2 Bandabase

Para comenzar con la descripción de esta capa se hará una distinción entre las funciones del *controlador de enlace* (*Link Controller*) y la *Bandabase*.

El *controlador de enlace* (LC) es responsable de llevar a cabo las operaciones a nivel de enlace sobre varios paquetes de datos en respuesta a comandos de alto nivel del *administrador de enlace* (*Link Manager*). Las entidades locales y remotas del *controlador de enlace* manejarán

el proceso de establecimiento de enlace, paquete por paquete, una vez ordenado por el LM y mantendrán el enlace una vez establecido [5].

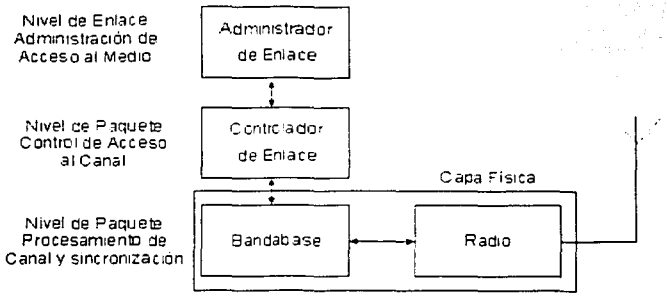


Figura 3.3. Diferencias entre el *controlador de enlace* y *bandabase*

La capa Física del modelo de referencia OSI esta representada por el *radio* y la *bandabase*. La *radio* es una interfaz entre el canal (aire) y la *bandabase* digital, dicha interfaz da formato a la información proveniente del LC, para una transmisión robusta y confiable en el canal y recupera información del canal para pasarla a la pila. La *bandabase* es responsable tanto de la codificación y decodificación de canal y el control de la sincronía a un nivel bajo, así como también del manejo del enlace dentro del dominio de la transferencia de un solo paquete de datos. En la figura 3.3 se ilustra la diferencia entre el *controlador de enlace* y la *bandabase*.

3.2.1 Dirección de dispositivos *Bluetooth*

Cada dispositivo *Bluetooth* tiene una dirección IEEE MAC[6] de 48 bits, conocida como *dirección Bluetooth* (BD_ADDR, *Bluetooth* Device Address). Esta dirección es usada en algunas de las operaciones de procesamiento serial de bits, y en particular para la derivación del código de acceso. La dirección *Bluetooth* se divide en: una parte de dirección no significativa (NAP), una parte de dirección alta (UAP), y una parte de dirección baja (LAP) como sigue[7]:

- BD_Addr[47:32] – NAP[15:0]
  Utilizada para iniciar el proceso de encriptación.
- BD_Addr[31:24] – UAP[7:0]
  Utilizada para iniciar los cálculos de HEC, CRC y salto de frecuencia.
- BD_Addr[23:0] – LAP[23:0]
  Utilizada por la palabra de sincronización y el salto de frecuencia.

[6] Media Access Control
[7] Los números entre corchetes indican el intervalo de bits que ocupa cada sección de la dirección *Bluetooth*.

3.2.2 Maestros, Esclavos y Piconets

Bluetooth es un sistema multiplexado por división en tiempo (TDM), donde la unidad básica de operación es una ranura (slot) de 625 μs. Cuando hay una conexión todas las operaciones transmitidas o recibidas ocurren en 1, 3 o 5 ranuras. En las operaciones de pre-conexión, la transmisión y la recepción pueden ocurrir algunas veces en medias ranuras (312.5 μs).

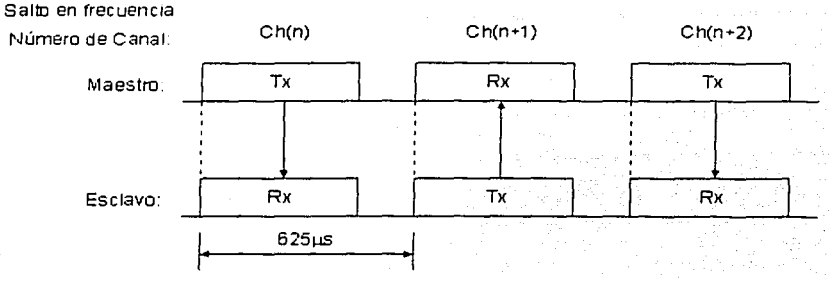


Figura 3.4. Temporización de paquetes de una sola ranura

Cada dispositivo *Bluetooth* puede ser Maestro o Esclavo en cualquier momento, aunque no simultáneamente. Estos dos papeles se definen como sigue:

- Maestro: Dispositivo que inicia un intercambio de datos.
- Esclavo: Dispositivo que responde al Maestro

Durante la comunicación el Esclavo utilizará la temporización de su Maestro, con la que sincronizará sus saltos. En la figura 3.4 se tiene que: primero el Maestro transmite al Esclavo. Esto ocurre cuando ambos dispositivos están sintonizados en el canal de radio C(n), 625 μs después, los dos dispositivos vuelven a sintonizar sus radios, o "saltan" al C(n+1), en este tiempo el Esclavo debe responder a pesar de que no haya sido exitosa la recepción del paquete anterior. Después de la ranura de recepción del Maestro, el Esclavo otra vez se pone en alerta de un mensaje. Sin embargo, el Maestro puede elegir entre transmitir a alguien más o no transmitir a nadie. El Esclavo no transmitirá de nuevo hasta que el Maestro le vuelva a transmitir.

Cada dispositivo hará un salto una vez por paquete. Esto es una parte fundamental del sistema *Bluetooth* y provee las siguientes caracteristicas [22]:

- Seguridad: Debido a que el salto es una secuencia pseudoaleatoria basada en la dirección del dispositivo Maestro.

- Confiabilidad: Debido a que si un radio interfiere y causa una pérdida de un paquete en el C(n), es poco probable que también interfiera en C(n+m), donde n+m es una distancia alejada desde n debido al algoritmo pseudoaleatorio de salto.

Bluetooth define paquetes de información los cuales tienen una longitud de 1, 3 o 5 ranuras, la figura 3.5 muestra cómo la temporización cambia un poco para esos paquetes multi-ranura. Usar un paquete de longitud más grande permite altas tasas transferencia de información, a costa de la confiabilidad. Es más probable que las ráfagas de errores o interferencia afecten a un solo paquete de larga duración en un solo canal que, a muchos paquetes de corta duración distribuidos en varios canales.

Todos los paquetes tienen la misma cabecera y la misma información de control (*overhead*), así los paquetes multi-ranura son más eficientes en cuanto al transporte de datos. Resulta poco práctico hacer saltos durante paquetes sencillos debido al largo tiempo de sincronización de radio, así en paquetes multi-ranura se usa la máxima cantidad de tiempo para enviar información.

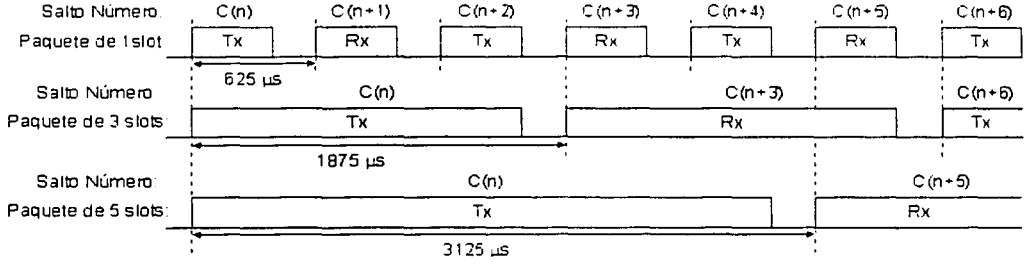


Figura 3.5. Temporización de paquetes multi-ranura

De esta manera, un Maestro controla qué transmitir (paquetes de 1, 3 o 5 tramas) y cuándo hacerlo. Los esclavos están esperando silenciosamente durante cierto tiempo y cualquier Esclavo que escuche un paquete con su dirección lo recibe y le responde.

Bluetooth sería una tecnología útil tan solo con facilitar la comunicación punto a punto; sin embargo, es mucho más que eso. *Bluetooth* crea un tipo de red LAN en miniatura, a la cual se le conoce como *piconet*.

Una *piconet* es una colección arbitraria de dispositivos *Bluetooth* los cuales están lo suficientemente cercanos para poder comunicarse e intercambiar información de forma regular. Una *piconet* esta formada por un dispositivo configurado como Maestro, al cual "le pertenece" la *piconet*, y entre uno y siete dispositivos que siempre actúan como Esclavos de su Maestro. Todos los Esclavos adoptan la sincronización del Maestro y responderán cualquier mensaje que incluya el *código de acceso* del Maestro [16].

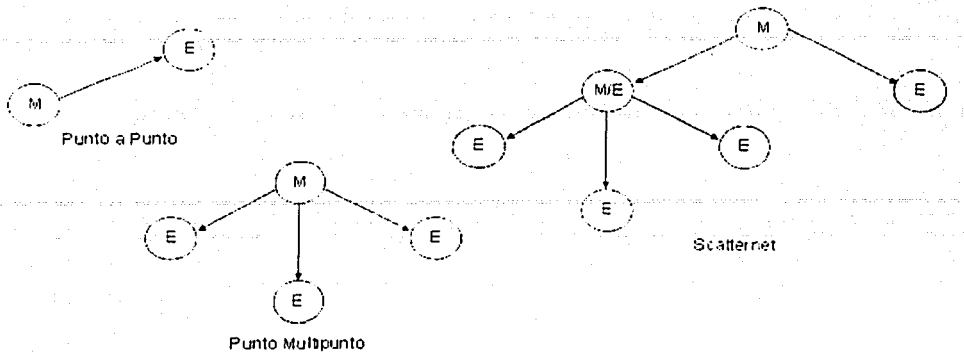


Figura 3.6. *Piconets y Scatternets*

La figura 3.6 ilustra la topología de un enlace punto a punto entre el Maestro y el Esclavo; en el segundo diagrama se muestra una *piconet* típica; y en el tercero tenemos un Maestro con tres Esclavos, pero el Maestro es también un Esclavo en otra *piconet*, obedeciendo a otro dispositivo Maestro, a esta última red se le conoce como *scatternet*. Como *Bluetooth* tiene un límite de siete Esclavos en una *piconet*, es deseable enlazar *piconets* para formar áreas de cobertura física mucho más amplias.

Como los dispositivos pueden moverse, pueden salirse fácilmente del área de cobertura y perder contacto con la *piconet*. Cada enlace tiene un tiempo de supervisión, el cual asegura que ese enlace sea terminado cuando ya no se encuentre el dispositivo dentro del área de cobertura.

3.2.3 Temporización del sistema

Como muchos protocolos de comunicación, *Bluetooth* sincroniza muchas operaciones con un reloj en tiempo real o con un contador. Esto asegura, por ejemplo, poder sincronizar intercambios de información de transmisión/recepción entre dispositivos, diferenciar entre paquetes perdidos y reenviados, y generar una secuencia pseudoaleatoria predecible y reproducible de números de salto de frecuencia. El reloj *Bluetooth* es un contador de 28 bits el cual se reinicia a 0 en el encendido y posteriormente comienza a correr libremente. Este reloj se incrementa cada media ranura, es decir, cada 312.5 µs. Esto sucede aproximadamente una vez por día. Cada dispositivo tiene su propio contador *nativo* que corre libremente y controla la temporización y operación de ese dispositivo. El reloj nativo es conocido como CLKN.

Si un dispositivo esta operando como Maestro, controla la *piconet*, y usa su CLKN como temporizador de referencia interno. En cambio si un dispositivo esta operando como Esclavo, su temporización debe estar sincronizada a su Maestro. La sincronización con el Maestro, asegura que las ranuras de transmisión y recepción del Esclavo estén alineadas correctamente

con las del Maestro, así como también que se escojan correctamente los números de salto de frecuencia. Para sincronizarse con el Maestro, un Esclavo debe agregar un valor de compensación a su reloj nativo (CLKN). Esta compensación deriva en un nuevo valor, el CLK, el cual es estimado a partir del CLKN del Maestro. Al CLK se le llama *reloj de la piconet*.

Existe otro reloj llamado CLKE, el cual se obtiene agregando otra compensación al CLKN. Este es usado por un Maestro para crear una estimación del CLK del Esclavo. Se usa en el caso del establecimiento de una conexión con el Esclavo antes de que el Esclavo se haya sincronizado con el Maestro. En la figura 3.7 se muestra un diagrama de los diversos relojes mencionados.

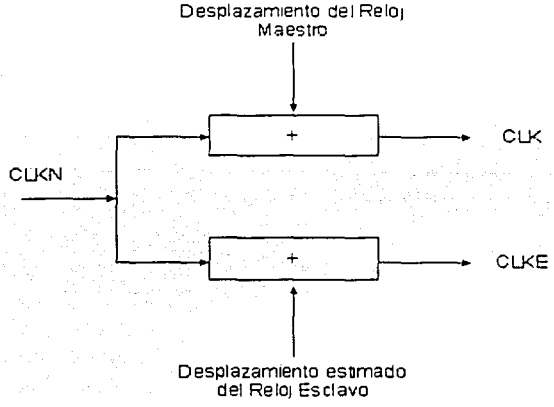


Figura 3.7. Relojes de *Bluetooth*

Los 2 bits menos significativos del CLK se usan para delimitar las ranuras y medias ranuras para la transmisión y recepción de los paquetes. También establecen el criterio para escoger sobre transmisión o recepción, dependiendo de si el dispositivo esta actuando como Maestro o como Esclavo. Una transmisión del Maestro en conexión siempre comienza cuando el CLK[1:0] = 00, y una transmisión del Esclavo en conexión siempre comienza cuando CLK[1:0] = 10.

Las figuras 3.8 y 3.9 ilustran el uso del CLK[1:0] y muestran la ventana de incertidumbre, la cual acomoda cualquier variación del sistema de temporización entre dos dispositivos que se comunican.

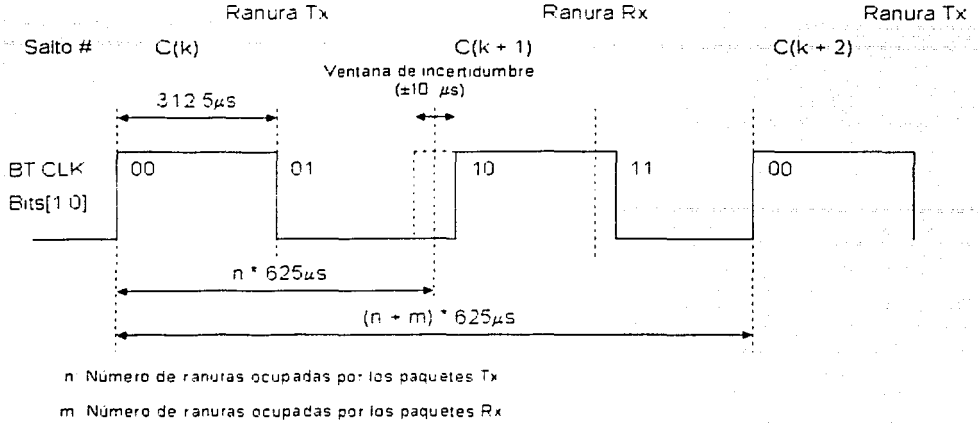


Figura 3.8. Ciclo de Tx – Rx del transmisor/receptor en el modo Maestro normal

El receptor en el Esclavo estará habilitado algún tiempo antes de que el Maestro inicie la transmisión, y en el caso de que no haya transmisión del Maestro, el radio del Esclavo se apagará después de un cierto tiempo de espera. Esta *desviación* del punto ideal de inicio de recepción, permite acomodar la desviación en la temporización de los sistemas, así como también deja que el Esclavo vuelva a sincronizar su reloj cada vez que recibe un paquete.

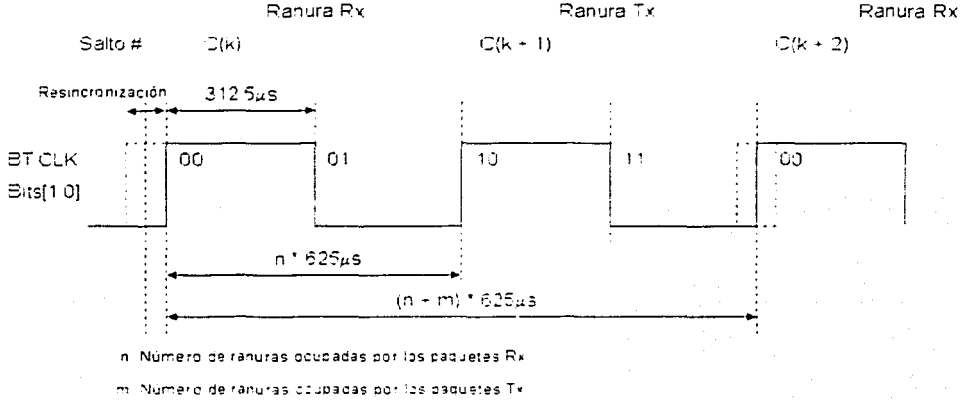


Figura 3.9. Ciclo de Rx – Tx del transmisor/receptor en el modo Esclavo normal

Todos los dispositivos realizan una correlación con la primera parte del paquete (palabra de sincronización), con el fin de realinear su temporización con el inicio de la transmisión del otro dispositivo. Un Maestro recibirá este paquete con la temporización resincronizada, mientras

que un Esclavo cambiará su sistema de temporización para acoplarse al del Maestro. Si un Maestro no ha transmitido a un Esclavo por algún tiempo, el Esclavo se resincronizará dentro de esta ventana. En una conexión, la ventana de incertidumbre se establece normalmente en ±10 μs. Esto permite a un Esclavo estar sin comunicación con el Maestro durante 800 ranuras, suponiendo el peor caso de precisión (±20 ppm) en ambas referencias de temporización (Maestro y del Esclavo).

3.2.3 Enlaces físicos SCO Y ACL

Una vez que se ha establecido el enlace, hay dos tipos básicos de paquetes de información que pueden ser intercambiados con una distinción importante entre ellos, caracterizada por el tipo de enlace [22]:

- Asíncrono No Orientado a Conexión (ACL, Asynchronous Connection-Less)
- Síncrono Orientado a Conexión (SCO, Synchronous Connection Oriented)

3.2.3.1 ACL

Un enlace ACL existe entre un Maestro y un Esclavo tan pronto como se ha establecido una conexión. Un Maestro podría tener cierto número de enlaces ACL a un número diferente de Esclavos en cualquier momento dado, pero sólo puede existir un enlace entre dos dispositivos[8]. Como ya se dijo, el Maestro no siempre transmite al mismo Esclavo de forma regular. Así el enlace ACL proporciona una conexión de paquetes conmutados donde la información es intercambiada esporádicamente en el momento en que los datos estén disponibles en la parte superior de la pila. La elección de a qué Esclavo transmitir o de cuál recibir es decisión del Maestro ranura por ranura, así los servicios isócronos y asíncronos son posibles. Muchos paquetes ACL facilitan la revisión de errores y retransmisión para asegurar la integridad de los datos.

Los enlaces portan información hacia y desde las capas L2CAP, o LMP. Todos los datos son enviados a través de L2CAP y pasan a *bandabase* como paquetes L2CAP. La configuración y el control de los enlaces se llevan a cabo por las entidades del *administrador de enlace* (LM) correspondientes, el comando asociado y los datos de control son enviados a *bandabase* como paquetes LMP. La información es llevada en paquetes DH (*Data High rate*) y DM (*Data Medium rate*), estos últimos paquetes portan menos información, pero proveen una protección extra contra los errores.

Un Esclavo puede responder únicamente con un paquete ACL en la siguiente ranura de transmisión Esclavo-Maestro si ha sido direccionado en la ranura anterior Maestro-Esclavo. Si

[8] Sobre el mismo enlace ACL se pueden tener varias conexiones en un esquema de conmutación de paquetes

el Esclavo falla en la decodificación de la dirección del Esclavo en la cabecera del paquete, no se tiene la certeza de si fue direccionado o no, y por lo tanto no le esta permitido responder.

Existen paquetes *broadcast* los cuales como su nombre lo indica están dirigidos a todos los Esclavos de la *piconet*.

3.2.3.2 SCO

Un enlace SCO es completamente diferente a un ACL, provee un enlace simétrico entre Maestro y Esclavo con un ancho de banda reservado para el canal y un intercambio periódico de datos mediante ranuras reservadas. Así, el enlace SCO provee una conexión de circuitos conmutados donde los datos son intercambiados regularmente, por lo que encuentra uso en aplicaciones donde es necesaria la transmisión periódica de datos, como en el caso del audio.

Un Maestro puede soportar hasta tres enlaces SCO al mismo Esclavo, o a diferentes Esclavos. Un Esclavo puede soportar hasta tres enlaces SCO del mismo Maestro. Debido a la naturaleza de continuidad de los datos SCO en el tiempo, los paquetes SCO no son retransmitidos nunca.

El Maestro transmitirá paquetes SCO al Esclavo a intervalos regulares, definiendo un parámetro llamado T_{SCO}, al cual se le conoce como *intervalo SCO* y es contabilizado en ranuras. El Esclavo siempre tiene permiso para responder con un paquete SCO en la ranura reservada para respuesta. Si el paquete fue incorrectamente decodificado debido a errores, entonces el Esclavo de todos modos responderá en la ranura reservada. En condiciones normales de operación, el Maestro no tiene permiso de transmitir hacia otro Esclavo diferente en la ranura reservada.

Un enlace SCO se establece por una orden del *administrador de enlace* (LM) desde el Maestro hacia el Esclavo. Este mensaje contendrá parametros de temporización, para especificar las ranuras reservadas, tales como el intervalo SCO (T_{SCO}) y el *offset* inicial, D_{SCO}

Una vez establecido el enlace SCO, lo datos en este enlace serán intercambiados cada T_{SCO} ranuras, y el tráfico ACL será transmitido en las ranuras que rodean a las reservadas. Esto asegura una continuidad de los datos SCO. Una excepción para la transmisión de datos diferentes a los SCO en las ranuras reservadas, es el envio de datos de control, con esto se asegura poder cerrar los enlaces SCO en el caso que ocuparan todo el ancho de banda.

3.2.4 Estructura de un paquete *Bluetooth*

Cada paquete consiste de un código de acceso, una cabecera, y una tributaria (*payload*) como se ilustra en la figura 3.10. El código de acceso es utilizado para detectar la presencia de un paquete y para direccionar el paquete hacia un dispositivo específico. Esto se lleva a cabo comparando el código de acceso recibido con una copia guardada del código de acceso del

Maestro. La cabecera contiene toda la información de control asociada con el paquete y el enlace, como la dirección del Esclavo para el cual el paquete es enviado. Finalmente, la tributaria puede contener información de un mensaje de capas superiores como L2CAP, o de LM, o bien los datos que circulan a través de la pila [5].

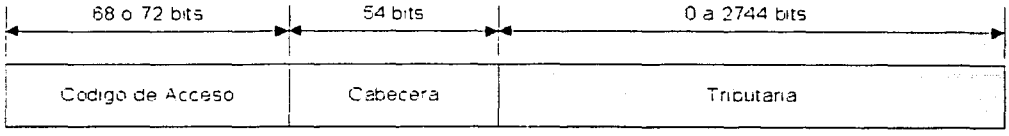


Figura 3.10. Estructura del paquete *Bluetooth*

3.2.4.1 Código de Acceso

Durante una conexión cuando un enlace está activo, el código de acceso identifica el paquete que viene de, o va hacia, un Maestro en específico, este código es formado con base en la *dirección Bluetooth* del Maestro. Para obtener códigos de acceso especiales (*Inquiry Access Code, IAC*), usados para descubrir dispositivos específicos durante el proceso de *inquiry*, se necesita usar una dirección especial del dispositivo *Bluetooth*.

La primera parte del código de acceso es un preámbulo de 4 bits el cual se usa para detectar los bordes de la información recibida. El preámbulo es una secuencia fija igual a 0101, o 1010, dependiendo del valor del primer bit de la palabra de sincronización[7] (syncword), con éste primer bit se forma una secuencia de 5 bits. Estos bits dan al circuito de recuperación de reloj únicamente 5 µs para crear una señal de reloj confiable, con la cual se puedan recuperar los datos restantes. Si sigue una tributaria, la última parte del código de acceso es un *trailer* de 4 bits, cuya función es similar al preámbulo, se usa para hacer una mejor recuperación del reloj. En la figura 3.11 se muestra la estructura del código de acceso de *Bluetooth*.

La palabra de sincronización se forma a partir de la *parte baja de la dirección Bluetooth* (*Lower Address Part, LAP*) mediante un algoritmo definido en las especificaciones. Primero, el algoritmo agrega una secuencia predefinida de 6 bits conocida como secuencia *Barker* a la LAP para mejorar las propiedades de autocorrelación de la palabra de sincronización. Esto da una palabra a la cual se le aplicará una operación OR-exclusiva de una longitud de 34 a 63 bits, con una secuencia PN de 64 bits P[63:0]. La secuencia resultante de 30 bits se codificad con un código de bloque BCH (*Bose Chaudhuri Hocquenghem*) (64,30) para resultar en una palabra de 64 bits, de los cuales 34 bits son de paridad. A los bits restantes de la secuencia PN, del 0 al 33, se les aplica una OR-exclusiva con la palabra BCH de paridad de 34 bits, para producir la

[7] El preámbulo es 0101 para una palabra de sincronización que empieza con un 0, y 1010 para el otro caso. El *trailer* es 1010 para una palabra de sincronización que termina con un 0, y 0101 en el otro caso.

palabra de paridad codificada final. La segunda XOR remueve las propiedades cíclicas del código de bloque del resultado.

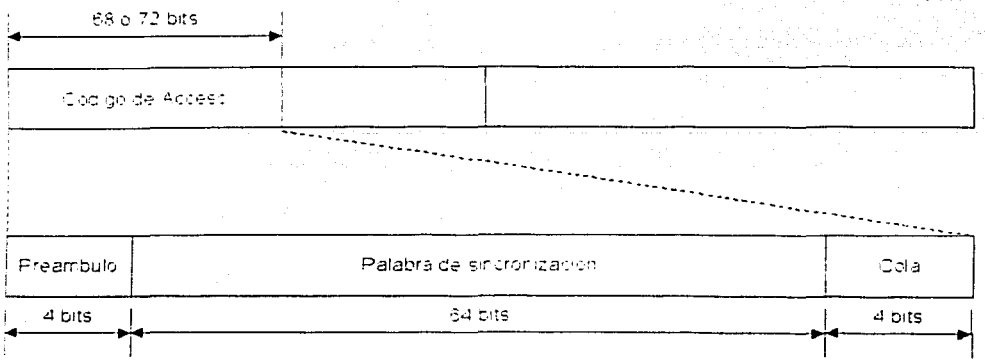


Figura 3.11. Estructura del código de acceso de *Bluetooth*

Como podemos ver en la figura 3.12, la palabra de paridad de 34 bits del código BCH es el elemento clave de la palabra de sincronización, pues tiene propiedades de muy alta autocorrelación y de muy baja correlación cruzada. Entonces, cuando un la palabra de un dispositivo se correlaciona con su palabra de sincronización esperada, se tendrá un pico grande exactamente en donde coincidan: la palabra de referencia y la palabra recibida. Otras palabras que no coincidan no producirán tal pico. El resto de la palabra de sincronización es construida por el LAP y la secuencia *Barker*, usada para generar la palabra de paridad. Las LAP's de dos dispositivos podrían diferir en sólo un bit; sin embargo, las dos palabras de paridad serán muy diferentes, a pesar de que el resto de las palabras de sincronización sean similares.

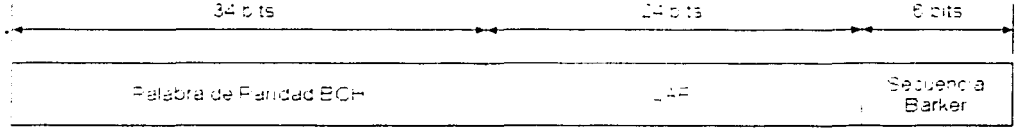


Figura 3.12. Estructura de la palabra de sincronización

El dispositivo receptor desplaza los datos hacia el correlador continuamente, el cual hace una correlación completa de 64 puntos contra la palabra de sincronización de referencia, o esperada. Una correlación grande indica que este paquete es el que el receptor espera y así seguirá recibiendo el encabezado del paquete, de otro modo, después de que la ventana de incertidumbre ha pasado el radio puede apagarse para poder conservar la potencia. De este modo, el Esclavo sólo recibirá paquetes que son transmitidos por su Maestro dentro de la

piconet que pertenece a dicho Maestro. En el caso de que dos *piconets* estén cerca, cada grupo de Esclavos recibirá sólo los paquetes de su propia *piconet*, la cual tendrá la palabra de sincronización de la *piconet*.

El instante de tiempo en el que la correlación ocurre es crucial, sucede exactamente a los 68μs dentro de la ranura del Maestro. Esto permite al Esclavo reajustar su propia temporización de ranura para coincidir con el Maestro en un intervalo de 1μs.

3.2.4.2 Tipos de Código de Acceso

Existen 4 tipos diferentes de códigos de acceso, lo cuales se usan de manera diferente como se describe a continuación.

- Código de acceso de canal (CAC, *Channel Access Code*). Se deriva del LAP del Maestro y lo utilizan todos los dispositivos en la *piconet* durante el intercambio de datos mientras está la conexión activa.
- Código de acceso a dispositivo (DAC, *Device Access Code*). Se deriva del LAP del dispositivo. Es utilizado cuando se hace *paging* a un dispositivo especifico y por el dispositivo en *Page Scan* mientras escucha por mensajes de búsqueda dirigidos a él.
- Código de acceso de búsqueda general (GIAC, *General Inquiry Access Code*). Es usado por todos los dispositivos durante los procesos de búsqueda, debido a que no se conocen las LAP de los dispositivos. El GIAC está fijado por la especificación.
- Código de acceso de búsqueda dedicada (DIAC, *Dedicated Inquiry Access Code*). Es un código que permite investigar la presencia de dispositivos *Bluetooth* de cierto tipo, por ejemplo, impresoras, o celulares. A los códigos de este mismo tipo cuya existencia se encuentra limitada por el tiempo se les llama *LIAC* (*Limited Inquiry Access Code*) o bien códigos de acceso de búsqueda limitada. Los *LIAC* se usan sobre una base temporal cuando la intervención de los usuarios provoca especificamente que dos dispositivos pretendan descubrirse entre si.

3.2.4.3 Cabecera del paquete

La cabecera del paquete contiene información de control asociada con el paquete. En total, la cabecera contiene 18 bits de información, los cuales son protegidos por un código de tasa 1/3, que opera bajo el principio FEC (*Forward Error Correction*).

Esta codificación hace una réplica de la información 3 veces, asi cada bit ocupa 3μs, o 3 periodos de bit al aire. La cabecera resultante tiene 54μs de duración. Esta redundancia de alto nivel y la codificación de *overhead* se emplea debido a que cada campo es crucial para la correcta operación del protocolo de *control de enlace* y por lo tanto, del enlace. Un campo CRC (*Cyclic Redundacy Check*) se incluye para que en el caso de que alguna parte del contenido sea

gravemente dañada, se pueda permitir que el paquete entero sea ignorado y el estado del enlace se mantenga. A continuación se describirá la estructura del encabezado del paquete *Bluetooth* mostrado en la figura 3.13.

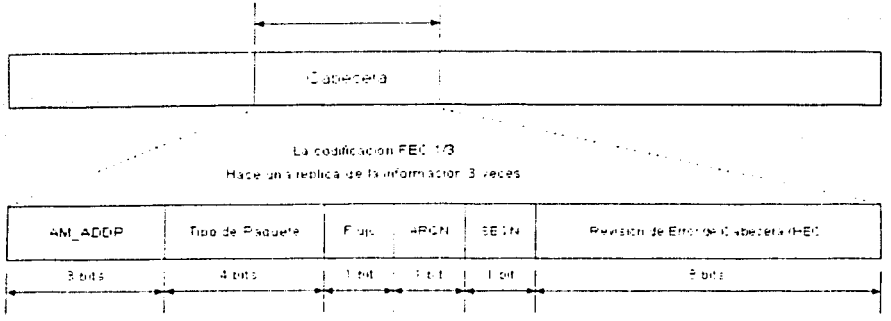


Figura 3.13. Estructura de la cabecera del paquete *Bluetooth*.

- **AM_ADDR**. Durante el proceso de *paging*, el Maestro asignará una dirección de miembro activo (*Active Member Address*, AM_ADDR) al Esclavo. Esto hará posible el manejo de conexión, la AM_ADDR se usa para direccionar todas las comunicaciones hacia un Esclavo; mientras que para el Maestro sirve para diferenciar respuestas de diferentes Esclavos (7 Esclavos, ubicados en un campo de 3 bits). Un AM_ADDR de cero corresponde a un paquete *Broadcast*, que proviene del Maestro y es dirigido a todos los Esclavos.
- **Tipo de Paquete.** El tipo de paquete es un campo que define qué tipo de tráfico es llevado por el paquete (SCO, ACL, NULL o POLL), el tipo de corrección de error usado por la *payload*, y cuántas ranuras dura esta *payload*.
- **Flujo**. Esta bandera es levantada por un dispositivo cuando no es capaz de recibir más datos, debido a que su *buffer* no ha sido vaciado.
- **ARQN y SEQN.** La bandera ARQN es levantada por un dispositivo para indicar que la recepción previa ha sido exitosa después de la validación del CRC. Si el ARQN de la cabecera regresada se pierde, el dispositivo que originó la transmisión asumirá que la bandera no fue establecida, esto indicará que el paquete anterior no fue reconocido, es decir, ARQN=NAK (*Negative-AcKnowledge*). Cada vez que un nuevo paquete es enviado, la bandera de secuencia (SEQN) se conmuta entre uno y cero. En el caso de que el paquete sea reenviado, la bandera SEQN se quedará igual y así el receptor sabrá que el paquete es el mismo.
- **Revisión de Error de Cabecera (*Header Error Check* HEC).** El campo HEC es simplemente una función CRC que se lleva a cabo en la cabecera.

3.2.4.4 Payload ACL

El campo *payload* de todos los paquetes ACL está dividido en tres partes: la cabecera del *payload*, la *payload* y el campo de CRC En la figura 3.14 se muestra la estructura de la *payload*, o tributaria de un paquete ACL.

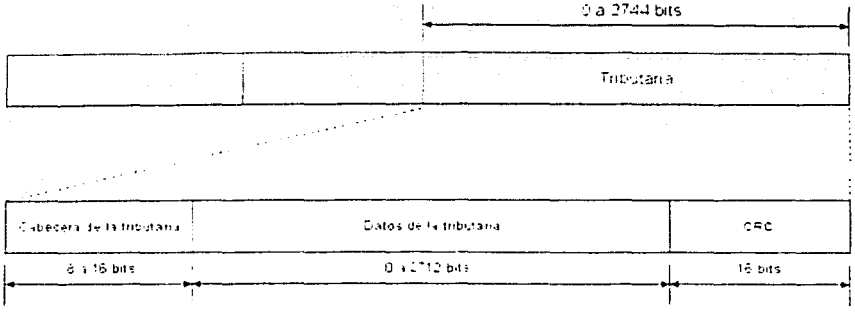


Figura 3.14. Estructura de la tributaria ACL

Cabecera del Payload ACL

El campo de la cabecera *payload* contiene la información de control de enlace siguiente:

- **El campo L_CH (*Logical Channel*).** Indica si la tributaria es el comienzo o la continuación de un mensaje L2CAP (debido a que los mensajes L2CAP pueden durar varios paquetes ACL), o un mensaje LMP (el cual es llevado totalmente en un paquete ACL de una sola ranura).
- **La bandera de flujo.** Controla la transferencia de datos a nivel de la capa L2CAP.
- **El campo de longitud.** Describe la longitud en bytes de los datos de la tributaria.

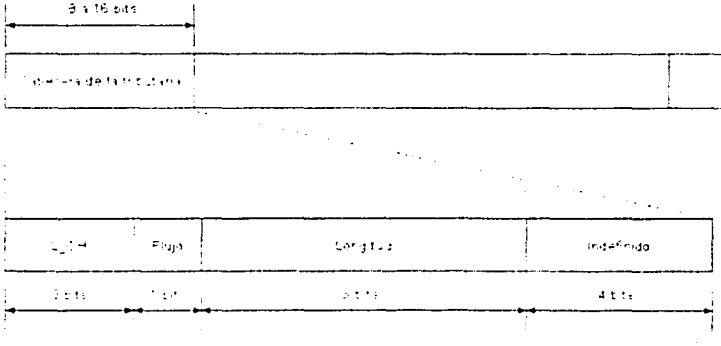


Figura 3.15. Estructura de cabecera de la tributaria ACL

3.2.4.5 Estructura de paquete SCO

Los paquetes SCO comparten el mismo código de acceso y la cabecera que los paquetes ACL, aunque los campos de flujo, ARQN y SEQN sean redundantes debido a que el control de flujo y retransmisión no aplican en enlaces SCO. El campo CRC se encuentra ausente. El tamaño de la *payload* se fija a 30 bytes, con una longitud de datos de origen de 10, 20 o 30 bytes, dependiendo del tipo de paquete, el cual selecciona la tasa FEC para utilizar (1/3, 2/3 o ninguna). En la figura 3.16 se muestra la estructura del paquete SCO.

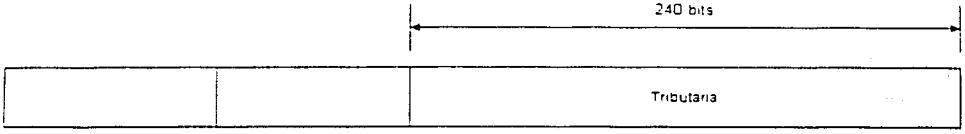


Figura 3.16. Estructura del paquete SCO

3.2.4.6 Estructura de paquete mixto SCO/ACL

Un caso especial del paquete SCO es el paquete DV, mostrado en la Figura 3.17, el cual, como los paquetes SCO, debe ser enviado en intervalos regulares. El campo de voz es lo suficientemente largo para enviar información de tipo HV1 de 10 bytes, la cual no es protegida por el FEC. Al igual que los paquetes SCO normales, el campo de voz no puede ser retransmitido. Sin embargo, el campo de datos está protegido con un codificador operando en modo FEC de tasa 2/3, un campo CRC y las banderas ARQ y SEQ. La retransmisión del campo de datos es posible, y esto ocurrirá junto con la transmisión del SCO que está saliendo en el campo de voz.

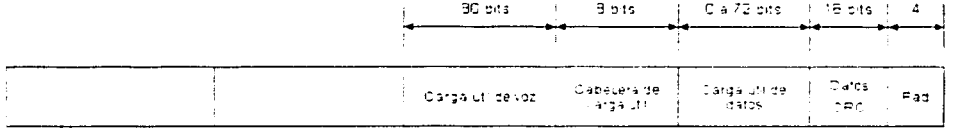


Figura 3.17 Estructura del paquete DV

El campo de carga útil de datos llevará 9 bytes (72 bits) de datos, por lo que el campo de datos se acerca mucho a un paquete DM1. El campo *pad* al final del paquete se agrega a la información original, cuando se requiere, para asegurar que sea un múltiplo de 10 bits antes de realizar la codificación con el código que operará en modo FEC.

3.2.4.7 Paquetes especiales ID, NULL, POLL y FHS

El paquete ID consiste sólo del código de acceso y se usa durante la operación de *preconexión*, donde un enlace todavía no se ha establecido y los temporizadores entre dispositivos no están relacionados. El paquete ID es un mecanismo de señalización altamente robusto, debido a que la única información que lleva es el código de acceso del dispositivo del cual proviene, o hacia el que se dirige. Es detectado y decodificado simplemente correlacionándolo contra el código de acceso seleccionado.

El paquete NULL consiste sólo del código de acceso y la cabecera de paquete y se usa para pasar una confirmación (ARQ) o control de flujo al dispositivo, después de la recepción de un paquete. Esto ocurre típicamente durante un enlace cuando el receptor de la transferencia de datos no tiene nada que enviar de regreso. Sin embargo, debe continuar confirmando que el paquete se ha recibido. En ausencia de cualquier información a transferir, el Esclavo envía paquetes NULL en el establecimiento inicial de un enlace entre dos dispositivos. El paquete NULL en sí mismo no requiere confirmación.

El paquete POLL tiene la misma estructura que el paquete NULL, pero mientras que el paquete NULL no tiene que ser confirmado, el paquete POLL sí debe ser confirmado independientemente de si el receptor tiene información que transmitir. Sin embargo, el paquete POLL, en sí mismo, no afecta la confirmación o el esquema de control de retransmisión gobernado por el ARQ y el SEQ. El uso típico del POLL es hecho por el Maestro para revisar la presencia de Esclavos en una *piconet*, los cuales deben responder si están presentes.

El paquete de sincronización de salto de frecuencia (FHS), mostrado en la figura 3.18, es utilizado durante el proceso de *inquiry*, durante el proceso de *page*, y también durante el intercambio de papeles entre un Maestro y un Esclavo. En todos los casos, el paquete FHS proporciona toda la información requerida por el receptor para sincronizarse con el emisor en términos de temporización y saltos de frecuencia, así como también obtener el código de acceso del dispositivo correcto. Los paquetes FHS contienen la siguiente información:

- El campo palabra BCH de paridad, que contiene la parte clave de la palabra de sincronía, y junto con el LAP y una secuencia *Barker*, formará el Código de Acceso del dispositivo transmisor.
- El LAP, UAP y NAP, que son parte de la dirección *Bluetooth* de 48 bits del dispositivo transmisor y se utilizan para diferentes propósitos en la codificación de paquetes y cálculo de saltos.
- El SR y el SP, que especifican los tiempos de *page scan repetition* y *scan period*[10], respectivamente, mientras el campo de *page mode* indica qué modo *page* es usado por el dispositivo transmisor.

[10] Parámetros de temporización en el proceso de *page* y *page scan*

- El campo clase, que indica a qué clase de dispositivo pertenece el remitente, por ejemplo, una impresora, un PDA, un celular, etc.
- El campo AM_ADDR, que indica la *dirección de miembro activo* que el destinatario usará, si el paquete fue enviado por un Maestro durante el proceso de *paging*, o debe ser cero si es regresado por un dispositivo en el proceso de *inquiry scanning*.
- El campo CLK, que es el reloj *Bluetooth* del remitente. Esto permite al destinatario sincronizar su reloj calculando la compensación y aplicándola a su reloj nativo.

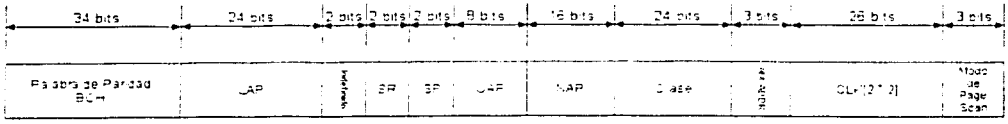


Figura 3.18. Formato del paquete FHS

3.2.5 Tipos de paquetes y construcción de paquetes.

En la tabla 3.3 se listan los distintos tipos de paquetes y sus características particulares.

Segmento	Tipo de Código	Número de slots	Enlace SCO	Enlace ACL	Cabecera de carga útil (bytes)	Carga útil de usuario (bytes)	FEC	CRC
-	-	1/2	ID		0	0	No	No
1	0000	1	NULL	NULL	No	No	No	No
	0001	1	POLL	POLL	No	No	No	No
	0010	1	FHS	FHS	No	18	2/3	No
	0011	1	DM1	DM1	1	0 - 17	2/3	Si
2	0100	1	Indefinido	Indefinido	1	0 - 27	No	Si
	0101	1	HV1	Indefinido	No	10	1/3	No
	0110	1	HV2	Indefinido	No	20	2/3	No
	0111	1	HV3	Indefinido	No	30	No	No
	1000	1	DV3	Indefinido	1	10 0 - 9	2/3	Si
	1001	1	Indefinido	AUX1	1	0 - 29	No	No
3	1010	3	Indefinido	DM3	2	0 - 121	2/3	Si
	1011	3	Indefinido	DH3	2	0 - 183	No	Si
	1100	3	Indefinido	Indefinido	-	-	-	-
	1101	3	Indefinido	Indefinido	-	-	-	-
4	1110	5	Indefinido	DM5	2	0 - 224	2/3	Si
	1111	5	Indefinido	DH5	2	0 - 339	No	Si

Tabla 3.3. Resumen de tipos de paquetes. Las cifras sombreadas indican parametros con respecto al elemento ACL de un paquete DV; el resto de las cifras de la fila son aplicables al elemento SCO del paquete DV. Los paquetes DV pertenecen a la familia SCO.

La columna *Segmento* separa los diferentes tipos de paquetes: control, datos de una sola ranura, datos de tres ranuras y datos de cinco ranuras. Los diferentes tipos de paquetes ACL negocian entre una mayor o menor codificación para el control de errores FEC y paquetes

largos o cortos para obtener bajas tasas de transmisión con alta robustez, o bien, altas tasas de transmisión con baja robustez. Una mayor protección contra errores se logra a expensas de más redundancia, por otro lado, los paquetes más largos presentan mayor susceptibilidad a errores [5].

Los paquetes SCO usan el incremento de redundancia para aumentar la protección contra errores y por lo tanto robustez.

Es responsabilidad del *administrador de enlace* (LM) monitorear la calidad y confiabilidad del enlace y decidir qué tipos de paquetes son apropiados en cualquier momento. Algunos dispositivos no soportarán todos los tipos de paquetes, por lo que durante el periodo de configuración se necesitará algún tipo de negociación.

Una tasa simétrica de transmisión se alcanza utilizando el mismo tipo de paquete en ambas direcciones; sin embargo, muchas aplicaciones tienen necesidades de ancho de banda asimétrico.

3.2.6 Canales lógicos

La especificación define cinco canales lógicos de información que son portados en enlaces físicos SCO y ACL. Son importantes en la diferenciación del contenido de los diversos paquetes que se transmiten y la información que llevan [22]. Los canales son los siguientes:

- LC (*Link Control* o Control de Enlace). Es llevado en la cabecera del paquete y consiste del ARQ, SEQ y los datos de control asociados. Como fue descrito previamente, estos datos son cruciales para mantener y controlar el enlace.
- LM (*Link Manager*, Administrador de Enlace). Es llevado a través de una carga útil ACL dedicada y contiene datos de control LM que se intercambian entre dos entidades LM. Típicamente llevado por paquetes DM, el canal LM se indica por el código L_CH = 11. Los paquetes DM1 son comunes a los enlaces SCO y ACL, esto permite transportar mensajes LMP a través de un enlace SCO.
- UA/UI (*User Asynchronous* o *User-Isochronous data*). Es llevado por la carga útil del ACL y contiene información de usuario L2CAP. La fragmentación es manejada por el valor apropiado de L_CH: 10 para los paquetes de inicio y 01 para los paquetes de continuación.
- US (*User-Synchronous data*). Los datos síncronos son llevados por la carga útil de los canales SCO.

3.2.7 Codificación de canal y procesamiento del flujo de bits

La figura 3.19 describe el flujo de datos en *bandabase* y cómo la codificación de canal se ajusta en todo el cuadro. Debido a que cada función de codificación se lleva a cabo en todo, o en parte

del flujo de datos al momento que pasa a través de la *bandabase*, nos referiremos a esto como *procesamiento del flujo de bits*.

La codificación de los datos antes de la transmisión es importante para protegerlos contra las perturbaciones en el canal. Debido a las caracteristicas de *Bluetooth*, como la baja tasa de transmisión, la codificación empleada no es muy compleja. La codificación aplicada al flujo de bits en *Bluetooth* consiste de los siguientes elementos[5][22]:

- **CRC (*Cyclic Redundancy Check*)** y **HEC (*Header Error Check*).** Un CRC se lleva a cabo en todas las cabeceras de los paquetes y sobre la carga útil ACL para validar la integridad de la información recibida. La decodificación en modo FEC tiene un alcance limitado para corrección y detección de errores, mientras que un CRC anunciará cualquier error en los datos. El HEC de 8 bits se calcula en la cabecera de carga útil antes de aplicar el control de errores FEC.
- **Encriptación.** Para llevar a cabo este proceso, al flujo de datos se le aplica la operación *OR exclusiva* con el flujo de cifrado proveniente del sistema de encriptación.
- ***Whitening.*** El *blanqueamiento* involucra la mezcla de una secuencia pseudoaleatoria de bits con los datos para que éstos se vuelvan aleatorios. Esto reduce la posibilidad de largas secuencias de ceros o unos, eliminando la componente de DC. Esta caracteristica mejora el desempeño en la recepción de una cadena de bits, en la parte de *radio*, debido a que evita *offsets*.
- **FEC (*Forward Error Correction*)** Al agregar bits de redundancia extra creados a partir de la información de entrada, cierto número de errores de bits en la información pueden ser detectados y corregidos. Hay tres opciones para FEC dependiendo del tipo de paquete: ninguno, 1/3 y 2/3: El que ofrece mayor protección de ellos es el 1/3, haciendo que se repita un mismo bit tres veces y se decodifica por "mayoria de votos" en el lado receptor. Esto es suficiente para detectar dos errores en cada tres bits de entrada y corregir uno en cada tres. La cabecera de paquete utiliza codificación 1/3, debido a que ahi se encuentra la información más importante del paquete. La FEC 2/3 es un código de Hamming (15, 10) implementado como un registro de corrimiento lineal con retroalimentación (LFSR). Por cada 10 bits de entrada al LFSR, se obtienen 15 bits a su salida. Esta protección es suficiente para detectar 2 bits erróneos en cada 15 bits y corregir 1 de cada 15.

El *whitening* y el HEC se aplican a todos los paquetes con al menos una cabecera. El FEC y el CRC se aplican dependiendo del tipo de paquete, como se ve en la tabla 3-3.

El orden del procesamiento del flujo de bits es importante. Considerando la ruta de transmisión mostrada en la figura 3.19, primero se realiza el CRC o el HEC para proveer la

validación de datos primitivos. Después, se aplica el *whitening*. Finalmente, se realiza la decodificación FEC.

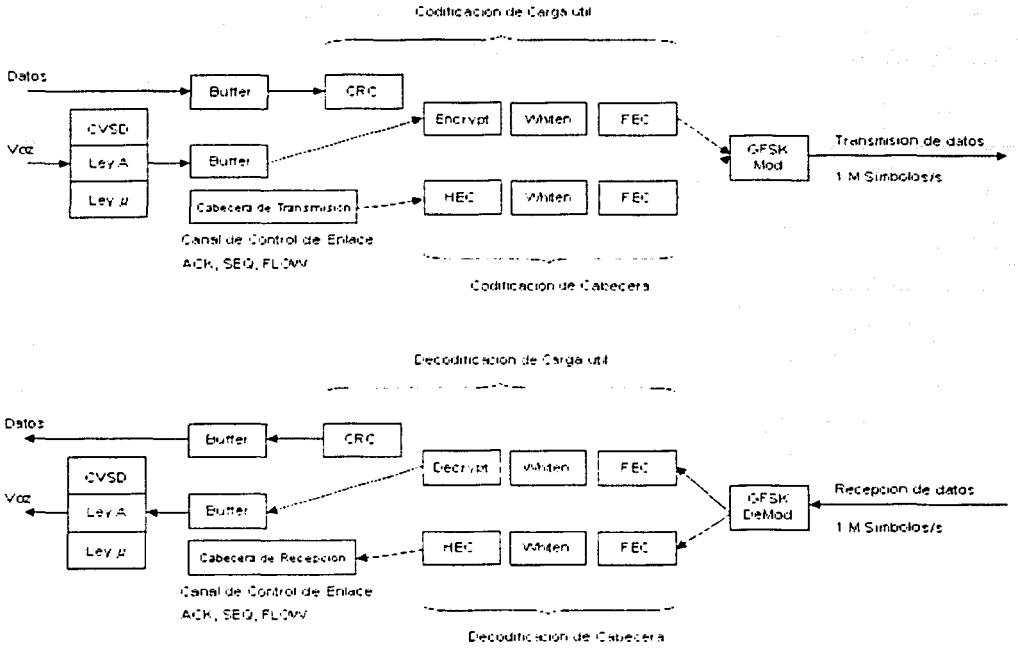


Figura 3.19. Flujo de datos de *Bandabase*

Los datos SCO se codifican o decodifican con uno de los tres esquemas de codificación de audio y se omite el CRC, debido a que la retransmisión del SCO no es posible. La función de los *buffers* de Tx/Rx es permitir escribir la información que será transmitida o leer la recibida, mientras se realiza el procesamiento del flujo de bits.

Cada enlace requerirá su propio *buffer* de transmisión debido a que cualquier paquete ACL puede requerir ser retransmitido en caso de errores en la recepción.

3.2.8 Sincronización del tiempo base y correlación de recepción

Un dispositivo *Bluetooth* mantiene la sincronización del tiempo base realizando una correlación periódica contra la palabra de sincronía, esto se muestra la figura 3.20. Nominalmente, la ventana de incertidumbre de *Bluetooth* es de ±10μs, o 20 símbolos, y requiere resincronización al menos cada 800 ranuras. Sin embargo, en otras circunstancias, como en el *inquiry scan*, o al regresar del modo *park*, la ventana puede ser más grande.

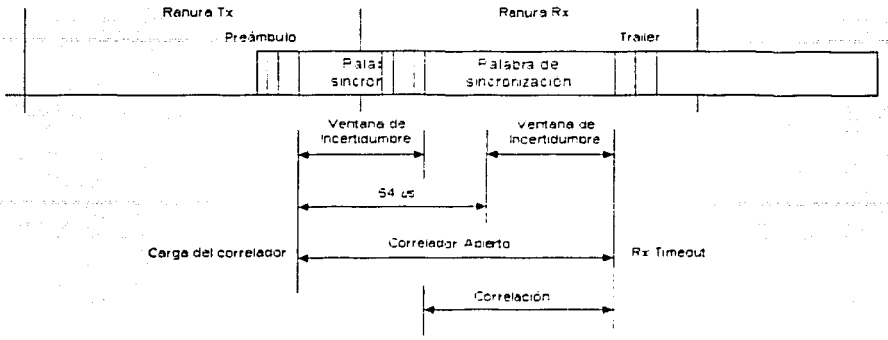


Figura 3.20. Temporización del correlador de recepción.

La correlación se lleva a cabo contra la palabra de sincronización de 64 bits guardada, por lo que el correlador debe primero introducir completamente los 64 bits recibidos antes de llevar a cabo esta correlación. Este proceso se comienza media ventana de incertidumbre antes del inicio de la ranura. Posteriormente se realiza una correlación de 64 puntos y se compara con un valor de umbral. Cada bit recibido es introducido sucesivamente al sistema y se repite la correlación hasta que un cierto número de correlaciones hayan sido hechas, o bien hasta que el valor del umbral haya sido sobrepasado.

En el punto en donde la correlación coincide, deben ocurrir dos operaciones:

- Se señala un evento para que la máquina de estado receptora comience la recepción.
- Se ajusta la sincronía de los dispositivos.

El punto de coincidencia es el punto en el que el último bit de la palabra de sincronización transmitida por el Maestro ha sido recibido. El comienzo de la cabecera de paquete será el siguiente bit en un tiempo de 1µs (código de acceso de 68 bits), o el quinto bit en un tiempo de 5µs (código de acceso de 72 bits). El *offset* de temporización entre el transmisor y el destinatario se mide en este punto. Si T_{Match} es el tiempo que abarca desde que el destinatario recibe la ranura, hasta que el correlador llega al punto de coincidencia (tomado del valor actual del tiempo base) el *offset* se calcula con T_{Match} menos la longitud de la palabra de sincronización (64) y menos la longitud del preámbulo (4), o $T_{Match} - 68$.

El CLK actualizado y el conteo de *subslot* se aplican inmediatamente, de este modo en la siguiente recepción de la cabecera y la carga útil, habrá una sincronía correcta. Si el dispositivo es Esclavo, entonces actualiza sus *offsets* para alinearse con el Maestro. Sin embargo, si el dispositivo es un Maestro, cada par de ranuras subsecuentes tendrá que ver con un Esclavo diferente, cada uno utilizando la temporización del Maestro. Por lo que aunque el Maestro

mantenga un *offset* de cero en cada caso, debe resincronizar su proceso de recepción para encontrar el comienzo de los datos siguientes a la coincidencia del correlador.

3.2.9 Saltos en frecuencia

Los saltos en frecuencia son el núcleo de *Bluetooth*, y como se verá en el siguiente capítulo forman una parte fundamental de los protocolos. La selección de salto de canal funciona como un algoritmo de mapeo que sigue una secuencia diferente dependiendo del estado de control de enlace (búsqueda, llamado, etc). Se selecciona una secuencia en particular dependiendo de las partes de la dirección *Bluetooth* provista (LAP y UAP). Posteriormente se indexa con base en esa secuencia utilizando el valor *Bluetooth* CLK. Debido a las restricciones en la parte de *radio*, el algoritmo podría truncarse a 23 canales (en lugar de 79) en algunos países [5].

El flujo de números de canal generado se pasa al subsistema de RF, donde se programan en el sintetizador de canal.

Por cada uno de los 79 canales, hay cuatro secuencias definidas que se describen a continuación. Las descripciones son específicas para la operación de salto de 79 canales y están basadas en el estado del LC.

- ***Page* o *Inquiry*.** Consiste en una secuencia rápida controlada por el reloj de media ranura usando 32 saltos igualmente distribuidos en todo el rango de transmisión. La secuencia de transmisión se divide en dos partes: A y B. La parte A contiene las 16 frecuencias de cada lado y lo más cercanas a la frecuencia esperada de *page* o *inquiry scan* que está escuchando el dispositivo explorador. La parte B contiene las siguientes 16 frecuencias ubicadas en cualquier lado del "tren" A. Dando como resultado la siguiente secuencia.

  f(k-8)...f(k)...f(k+7), f(k-8)...f(k-15), f(k-16)...f(k-9)

  donde f(k) es el canal que se espera, y que ha sido definido por el CLKE durante el *page*. Durante el *inquiry*, debido a que no hay estimaciones, la secuencia se controla por el CLKN del dispositivo investigador y se comienza en un punto aleatorio en la secuencia. La dirección BD correcta es importante para que las dos unidades tengan la misma secuencia. Respecto al tiempo, el orden dentro de los "trenes" se recorre para evitar una falta de coincidencia constante. Después de dos transmisiones de media ranura, el dispositivo escucha en dos mitades de ranura, que corresponden a dos frecuencias sucesivas en secuencia de respuesta (*page inquiry*) del patrón usado para transmitir. Este algoritmo facilita la operación de acoplamiento de frecuencias (*frequency handshaking*).

- ***Page Response o Inquiry Response.*** Durante el estado de respuesta, el CLKE (respuesta a *page*) o el CLKN (respuesta a *inquiry*) son congelados con el valor que tenían en el momento que el correlador señalizó la recepción de un paquete ID. Esto asegura que una vez que se ha encontrado un canal coincidente, este no se pierde. La primera respuesta se hace en la misma frecuencia, mientras que cada transmisión subsecuente, se hará en la siguiente frecuencia de la secuencia de *page/ inquiry*.
- ***Page Scan o Inquiry Scan.*** Esta es una secuencia muy lenta, que usa los mismos parámetros de la dirección *Bluetooth* que el esquema de *page/ inquiry*, y por tanto, sigue la misma secuencia pero salta más lento, un salto cada 1.28s.
- **Conexión.** Los saltos de canal se producen a una tasa de uno por ranura, excepto durante paquetes multi-ranura. Durante un paquete multi-ranura, la secuencia de salto no se detiene, sino que continúa corriendo durante la transmisión del paquete. La secuencia involucra todas las 79 o 23 frecuencias espaciadas igualmente, pero con la máxima distancia entre ellas, con el máximo intervalo posible entre repeticiones. No se muestran patrones repetidos en un periodo de tiempo corto.

3.3 Controlador de enlace

La capa de control configura y maneja el enlace a nivel de paquetes de *bandabase*, también lleva a cabo operaciones de nivel superior como *inquiry, paging*, y la administración de enlaces múltiples con diferentes dispositivos y hasta con diferentes *piconets*.

El protocolo de control es transportado en el canal de control de enlace (LC), y es responsable del mantenimiento del enlace una vez que éste se ha establecido [16].

Antes de detallar las funciones de *controlador de enlace*, se describirá brevemente el mecanismo que se usa en la retransmisión de datos dañados, debido a que forma una parte importante de la perspectiva general del *controlador de enlace*.

3.3.1 Mecanismo de retransmisión

El mecanismo de retransmisión esta basado en un esquema de "requerimiento automático de repetición" (ARQ). En cada encabezado de paquete, la bandera ARQN indica el estado de la recepción del paquete previo. ARQN=1 o ACK significa que el paquete ha sido recibido y decodificado correctamente. ARQN=0 o NAK significa que la recepción del paquete previo falló.

Un NAK puede ocurrir bajo las siguientes condiciones:

- Un Maestro falla en la detección de un código de acceso y la recepción no ocurre, en este caso sucede que la respuesta de un Esclavo a la transmisión del Maestro se pierde.
- Un código de acceso es detectado y el CRC falla, esta razón es la más común para que ocurra la retransmisión durante la conexión.

Cada vez que un nuevo paquete con un CRC se transmite, la bandera SEQN se alterna entre 0 y 1, mientras que para la retransmisión del mismo paquete, ésta permanece con el mismo valor. El mecanismo anterior permite al receptor reconocer la diferencia entre un paquete retransmitido debido a que el receptor envió la bandera de no confirmación de la recepción del paquete anterior (NAK), y un paquete retransmitido debido a que el transmisor falló en la recepción correcta de la bandera de confirmación de recepción (ACK) (enviada por el receptor). Es decir, si un paquete es recibido correctamente, pero la bandera SEQN es la misma que la del paquete anterior, el paquete actual es ignorado.

Las banderas ARQN y SEQN son iniciadas por el primer intercambio de paquetes después del establecimiento del enlace. Los valores iniciales son ARQN=NAK y SEQN=1, tanto para el Maestro, como para el Esclavo.

En los mensajes de *broadcast*, el Maestro enviará datos a todos los esclavos participantes en la *piconet*, este proceso se lleva a cabo mandando el paquete con la AM_ADDR establecida en cero. Debido a que el mensaje de *broadcast* es dirigido a más de un dispositivo, el mecanismo de confirmación de recepción (ARQ) descrito anteriormente no es aplicable.

El mecanismo alternativo utilizado consiste en retrasmitir los paquetes de *broadcast* un número especifico de veces. La bandera ARQN no se usa; sin embargo, la bandera SEQN si se utiliza para identificar los paquetes retransmitidos. Un mensaje de *broadcast* de larga duración se segmenta en un número de paquetes *Bluetooth*, en donde cada paquete es retransmitido N_{BC} veces sucesivamente.

3.3.2 Estados del *controlador de enlace.*

En un determinado tiempo, un dispositivo *Bluetooth* permanece en uno de los posibles estados, que se describen a continuación [22].

3.3.2.1 *Standby*

En el estado de *standby*, el dispositivo está inactivo, no hay transferencia de datos, y el sistema de radio está apagado. El dispositivo es incapaz de detectar algún código de acceso. Este estado se usa normalmente para habilitar la operación con bajo consumo de energía.

3.3.2.2 *Inquiry* (*Búsqueda*)

Inquiry es el proceso en donde un dispositivo intenta descubrir todos los dispositivos habilitados con *Bluetooth* dentro de su área. Durante este procedimiento, los dispositivos que se han encontrado proveerán paquetes FHS al dispositivo que hace la *búsqueda*. Los paquetes FHS permiten a dicho dispositivo construir una tabla de información esencial requerida para hacer la conexión. Esta información contiene: el CLKN, el cual controla la sincronía y la

secuencia de salto en frecuencia; BD_ADDR, el cual controla la secuencia de salto en frecuencia y forma parte del código de acceso; la palabra BCH de paridad, que también forma parte el código de acceso.

3.3.2.3 *Inquiry scan* (*Exploración de búsqueda*)

Inquiry scan es la otra mitad del procedimiento de *inquiry*. A pesar de que la *búsqueda* es opcional y depende de la aplicación, la mayoría de los dispositivos periódicamente entran en el estado de *inquiry scan* para hacerse disponibles a los dispositivos "buscadores". Los dispositivos en el estado de *inquiry scan* escuchan por un determinado periodo de tiempo, esperando un paquete de búsqueda que usa un código de acceso fijo, el cual puede ser: general o dedicado (GIAC, DIAC). El proceso anterior es necesario debido a que no existe conocimiento del reloj, ni del comportamiento de los saltos en frecuencia de ninguno de los dispositivos que estés realizando la *búsqueda*.

Cuando se recibe un mensaje válido de *búsqueda*, el dispositivo en el estado de *inquiry scan* pasa al subestado de respuesta a la búsqueda (*inquiry response*) y contesta con información mediante un paquete FHS. Los estados de *inquiry* e *inquiry scan* utilizan una secuencia especial de salto, la cual está diseñada para reducir la cantidad de tiempo transcurrido antes de que ocurra el acoplamiento en frecuencia. Se utiliza una secuencia rápida para el estado de *búsqueda* y una lenta para el estado de *exploración de búsqueda*.

3.3.2.4 *Page (Llamado)*

Para establecer conexión, el dispositivo que se convierte en Maestro recibe instrucciones de la aplicación para llevar a cabo los procesos de *llamado*. El Maestro entra en el estado de *llamado*, en donde transmitirá mensajes de *page* dirigidos al dispositivo que pretende ser Esclavo. En este proceso se usa el código de acceso y la información de reloj obtenida en el procedimiento de búsqueda. El Esclavo confirma la recepción del mensaje de *llamado* y el Maestro pasa al subestado de *respuesta*, en el que contesta con su paquete FHS.

3.3.2.5 *Page scan (Exploración de llamado)*

Un dispositivo debe entrar en el estado de *page scan* para permitir a otro dispositivo que se encuentra realizando el *llamado*, establecer conexión. Una vez que el dispositivo en el estado de *page scan* recibe un paquete de *llamado*, entra en el subestado de *repuesta del Esclavo*, contesta con un paquete que confirma la recepción y a continuación espera un paquete FHS.

Una vez que ha recibido el paquete FHS del Maestro, el dispositivo Esclavo sincroniza su reloj y actualiza su código de acceso con base en los datos recibidos en el paquete FHS. Posteriormente entra en el estado de conexión.

Al igual que en el estado de búsqueda, en este estado se utilizan secuencias especiales de salto.

Es posible llevar a cabo el procedimiento de *llamado* sin antes haber realizado el de *búsqueda* si se conoce la dirección del dispositivo al que se desea conectar. Este puede ser el caso de una conexión dedicada entre una laptop y un teléfono celular. Realizar el procedimiento de *llamado* sin el de *búsqueda* lleva más tiempo debido a que no se conoce el CLKE (estimación del reloj del Esclavo).

3.3.2.6 *Conexión activa*

Al entrar en el estado de conexión, el dispositivo Esclavo se sincroniza con el reloj del Maestro, lo que le permite seguir su secuencia de saltos en frecuencia y ranuras de tiempo. Para alcanzar sincronía el Esclavo introduce un *offset* a su reloj de tal manera que se ajuste con el del Maestro.

Posteriormente el Maestro envía un paquete POLL para verificar la conexión. Al recibir el paquete, el Esclavo debe responder enviando un paquete de cualquier tipo, generalmente envía un paquete NULL. Las banderas ARQN del Esclavo y del Maestro son establecidas en el valor inicial NAK, en la siguiente transmisión del Maestro empezará el funcionamiento descrito anteriormente para las banderas ARQN y SEQ.

Si la conexión no se realiza de la manera descrita anteriormente, después de un tiempo determinado, se regresa al estado de *page* o *page scan*.

En el estado de conexión activa, el Esclavo se mantiene sincronizado al reloj del Maestro aunque éste último no transmita nada al Esclavo durante varias ranuras de tiempo. El Esclavo realiza la correlación con el código de acceso del Maestro (CAC) en cada paquete que transmite, aunque no esté destinado a este Esclavo. Es por lo anterior que el Maestro debe transmitir periódicamente aunque no halla datos que enviar, en este caso se transmiten paquetes tipo NULL.

Una vez que el Esclavo se ha enganchado con el CAC y recibido el encabezado del paquete, generalmente encontrará que la *dirección de miembro activo (AM_ADDR)* no corresponde con la suya, y deberá abortar el resto de la recepción. Debido a que en el encabezado se encuentra la información de la duración del paquete, el Esclavo puede pasar al modo *estacionario de baja potencia (Low Power Sleep)*, durante la duración de la transmisión del Maestro.

Durante el estado de *conexión*, se intercambian varios tipos de datos y es posible el uso de diferentes tipos de canales lógicos. Sin embargo, existen periodos de tiempo en los cuales los dispositivos entran en los modos de *bajo consumo de energía*, como los descritos a continuación.

3.3.2.7 *Conexión hold (retenida)*

En este modo el dispositivo deja de soportar la recepción de tráfico ACL por un determinado periodo de tiempo, de esta manera libera ancho de banda para realizar otras operaciones como: *búsqueda* o *llamado*. El dispositivo conserva su dirección de miembro activo. Una vez que el periodo de tiempo ha terminado, el dispositivo se sincroniza con el CAC y empieza a escuchar el tráfico otra vez.

3.3.2.8 *Conexión de tipo sniff*

En el modo *sniff*, al dispositivo Esclavo se le asigna una ranura de tiempo predefinida y una periodicidad para escuchar tráfico. El Esclavo escuchará en la ranura de tiempo D_{sniff} cada T_{sniff} ranuras de tiempo por un periodo de N_{sniff}. Durante la recepción de un paquete en este tiempo, el dispositivo continuará escuchando hasta que los paquetes con su dirección de miembro activo terminen y el periodo N_{sniff} termine. Posteriormente, el dispositivo espera hasta el siguiente periodo T_{sniff}.

3.3.2.9 *Conexión park (estática)*

En este modo, el Esclavo cede su dirección de miembro activo y sólo escucha el tráfico ocasionalmente. La mayoría del tiempo el dispositivo puede entrar al estado *estacionario de baja potencia*, el dispositivo sólo necesita *despertarse* en un momento determinado de transmisión para sincronizarse con el código de acceso del canal (CAC), antes de regresar al modo de *bajo consumo de energía* nuevamente.

3.3.3 Operación del *controlador de enlace*

El diagrama de la máquina de estado de un dispositivo *Bluetooth* aparece en la figura 3.21. Este diagrama muestra rutas que van de los estados de *standby*, a través del *inquiry*, a la conexión. Pero en realidad no existe esa ruta directa. Para establecer una conexión, se debe ir a través del estado de *paging* o *page scanning*. El diagrama pretende mostrar que al estado de *inquiry* se puede acceder desde el estado de *standby*, o el de *conexión*. Se puede observar que este diagrama no muestra las rutas específicas a través de los estados para establecer conexiones [24].

En la figura 3.22 se presenta un diagrama de estado que muestra de una manera simple cómo se establece un enlace desde el principio. Sin embargo, en la práctica, la transición es mucho más compleja debido a que intervienen otros factores. Por ejemplo, un Maestro en estado de conexión ocupado con transmisión de tráfico síncrona, puede llevar a cabo periódicamente operaciones de *búsqueda* o *llamado* para buscar posibles nuevos esclavos, o

bien puede realizar los procedimientos de *exploración de búsqueda* o *exploración de llamado*, para formar una posible *scatternet*, mediante un intercambio de papeles Maestro/Esclavo.

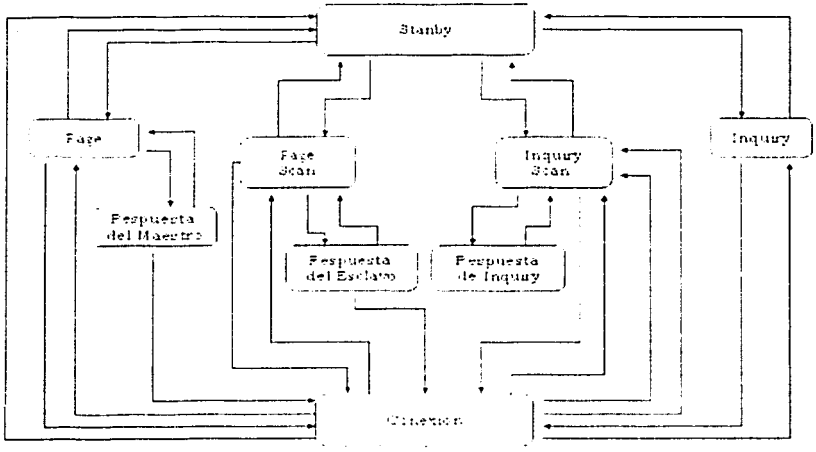


Figura 3.21. Diagrama de estado del controlador de enlace.

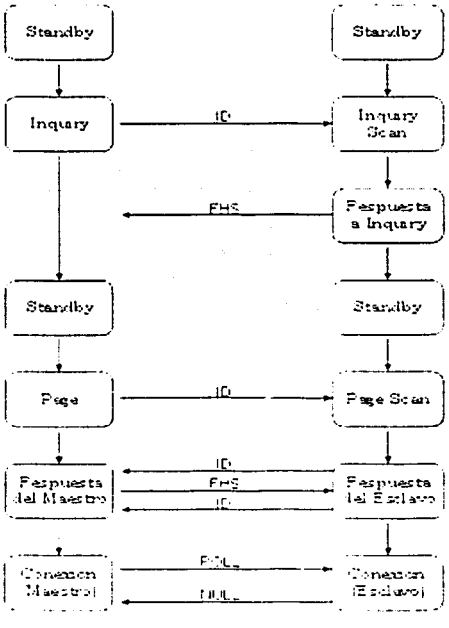


Figura 3.22. Diagrama de estado desde *standby* hasta la conexión

3.3.3.1 Búsqueda y descubrimiento de dispositivos

Debido a la naturaleza de las redes *Bluetooth*, en la que los dispositivos entran y salen del rango de comunicación constantemente, la topologia de la red y la de los elementos que la componen cambia frecuentemente. Es por lo anterior, que es necesario un mecanismo para el descubrimiento de los dispositivos habilitados con *Bluetooth* dentro del rango de comunicación.

El *controlador de enlace* usa los estados de *búsqueda (inquiry) y exploración de búsqueda (inquiry scan)* para manejar los procesos de descubrimiento de los dispositivos.

En la figura 3.23 se muestra el intercambio de mensajes entre dos dispositivos durante los estados antes mencionados.

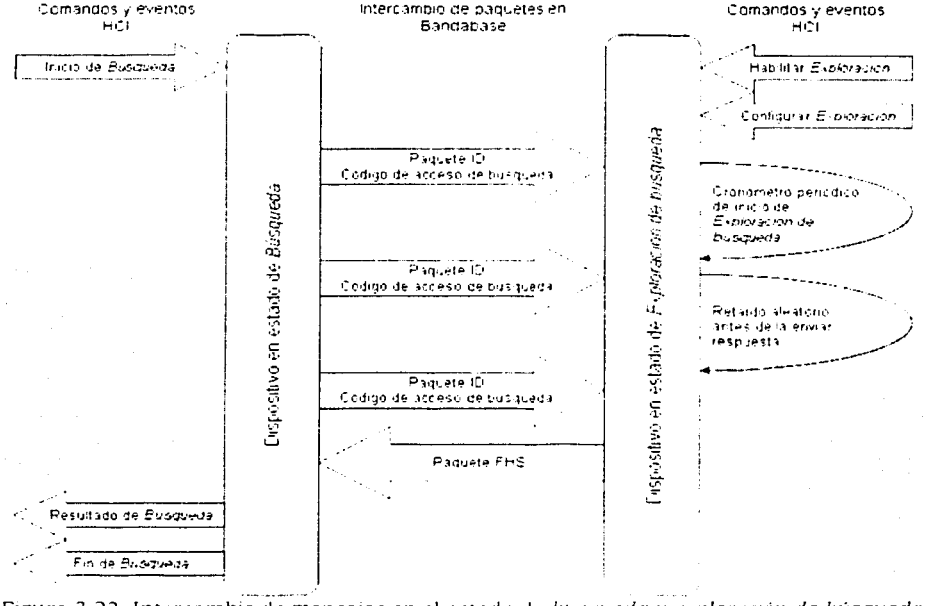


Figura 3.23. Intercambio de mensajes en el estado de *búsqueda y exploración de búsqueda*

Los dispositivos en estado de *búsqueda* transmiten paquetes ID que contienen un *código de acceso de búsqueda* (IAC), que frecuentemente es un *código de acceso de búsqueda general* (GIAC), el cual es común entre todos los dispositivos que realizan los procedimientos de *búsqueda*. Sin embargo, los dispositivos pueden usar un *código de acceso dedicado* (DIAC), que permite buscar solamente a cierto tipo de dispositivos.

Para optimizar el uso de energia y cumplir con otras actividades del enlace, los estados de *búsqueda y exploración de búsqueda* sólo se llevan a cabo cuando son requeridos por capas de protocolos superiores. Ambos estados se realizan en ventanas de tiempo cortas y periódicas.

Cuando un dispositivo en estado de *exploración de búsqueda* escucha a un dispositivo *buscador*, tiene la capacidad de responder inmediatamente, pero podría suceder que muchos dispositivos que hayan escuchado el mensaje hicieran lo mismo y el dispositivo *buscador* no recibiera correctamente ninguna respuesta, debido a que todas interferirían entre sí. Para evitar esto, se usa un periodo de tiempo aleatorio antes de contestar. Cuando un dispositivo escucha por primera vez un mensaje de *búsqueda*, en lugar de responder inmediatamente, espera un periodo de tiempo aleatorio (hasta 640 ms, o 1023 ranuras), al terminar este periodo vuelve al estado de *exploración de búsqueda*, cuando en esta ocasión escuche nuevamente el mensaje de *llamado* (paquete ID), responderá con un paquete FHS, 625 μs más tarde. El paquete FHS, contiene la información necesaria para que el dispositivo que inicio la *búsqueda* realice una estimación del reloj del dispositivo *explorador* (CLKE). Por parte del dispositivo *buscador*, es claro que deberá permanecer en el estado de *búsqueda* por un periodo de tiempo mayor que el periodo aleatorio.

El dispositivo *buscador* transmite dos veces por ranura de tiempo, en canales de salto consecutivos. En otras palabras, envía dos paquetes ID por ranura, en donde cada paquete se envía en una canal de salto diferente. El mecanismo anterior tiene la finalidad de encontrar en el menor tiempo posible a un dispositivo que se encuentre en *exploración de búsqueda*, el cual tiene una frecuencia de salto, más lenta, de 1.28 s o bien una vez cada 2.048 ranuras de tiempo. El dispositivo *buscador* transmite usando la secuencia de salto de *búsqueda* y escucha en la siguiente ranura de tiempo en los mismos canales usados anteriormente para transmitir.

Debido a que el *buscador* siempre escucha en el canal de salto de respuesta correspondiente al usado en la transmisión previa, cuando el dispositivo que está en *exploración de búsqueda* responde con un paquete FHS, el dispositivo *buscador* está listo para recibirlo. Sin embargo, a causa de que no hay una relación entre los relojes de ambos dispositivos el procedimiento no se lleva a cabo de una manera exacta y es necesario realizar varios intentos antes de completar el proceso. Los periodos de tiempo en ambos mecanismos están diseñados para maximizar la posibilidad de coincidencia del canal en el menor tiempo posible.

En caso de que el dispositivo *buscador* reciba un paquete FHS en la primera mitad de la ranura, éste no podrá recibir ningún paquete durante la segunda mitad debido a que no han tenido tiempo de ajustar su radio. Si el *buscador* recibe el paquete FHS en la segunda mitad de la ranura, éste no podrá transmitir nada durante la primera mitad de la ranura siguiente, por la misma razón.

Si el dispositivo en estado de *exploración de búsqueda* no recibe el primer o segundo paquete ID durante el periodo de duración de la *exploración*, entonces regresa al estado de *standby*, o de *conexión* según sea el caso. El estado de *búsqueda* termina cuando se han

recibido un número de respuestas predeterminado, o cuando ha acabado el tiempo establecido por el *host* para dicho procedimiento.

En la figura 3.24 se muestra la temporización del proceso de *inquiry*, en este ejemplo, el proceso termina con una sola respuesta FHS del dispositivo en *inquiry scan*. f(x) representa la frecuencia usada para transmitir o recibir un paquete.

En un ambiente libre de errores, el peor caso para la coincidencia en frecuencia con todos los dispositivos ocupando la secuencia de salto de *inquiry* es de aproximadamente 10 s [2]. Si el dispositivo *buscador* tiene algún enlace síncrono orientado a conexión (SCO) activo, entonces el tiempo necesario para la coincidencia en frecuencia aumenta, debido a la pérdida de tiempo por la transmisión regular de tráfico síncrono.

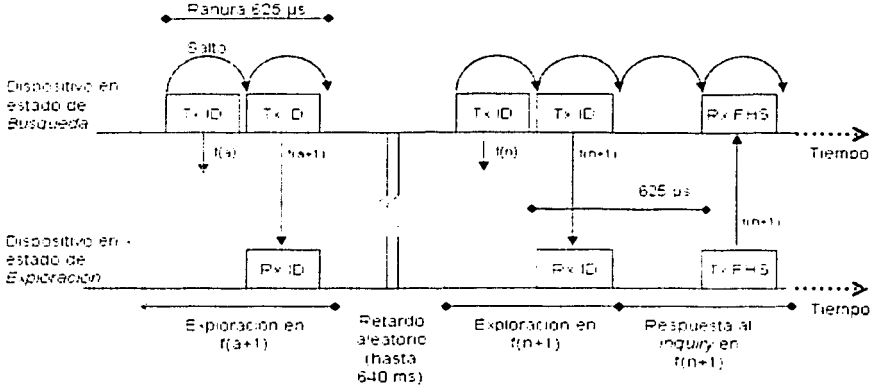


Figura 3.24. Temporización del proceso de *inquiry*

Es posible omitir todo el proceso de *búsqueda* si la dirección del dispositivo con el que se desea establecer conexión es conocida. El procedimiento de *llamado* puede ser realizado directamente, sin embargo puede tomar mayor tiempo establecer el enlace, pues no se tiene una estimación del CLKE, y por tanto la secuencia de salto de *llamado* no es predecible. Esto puede ser apropiado, por ejemplo, en el caso donde exista la combinación de un teléfono móvil de bajo costo y un *headset* (con una interfaz mínima) y sus respectivas direcciones sean preestablecidas por el fabricante para trabajar juntas.

3.3.3.2 *Llamado* y establecimiento de *conexión*

Para establecer *conexión* entre dos dispositivos se requiere de una previa invitación o *llamado*. La invitación realizada por un determinado dispositivo requiere ser escuchada por otro, al proceso de escuchar se le llama *page scan o exploración de llamado*. Al final del proceso

de *paging* el dispositivo que envió la invitación (*pager*) se convierte en el Maestro y el dispositivo que escuchó será el Esclavo. Sin embargo, a lo largo de la conexión los dispositivos pueden conmutar sus papeles.

En la figura 3.25 se muestra el intercambio de mensajes entre dos dispositivos durante el proceso de *llamado*. Los comandos de creación de *conexión* hacen que un dispositivo entre en el modo de *llamado*, en el que éste envía una serie de paquetes ID basados en la dirección del dispositivo al que se está invitando a la conexión. Mientras tanto, el dispositivo con el que se intenta establecer comunicación realiza *exploraciones* periódicas, en busca de dichos paquetes, con una duración específica.

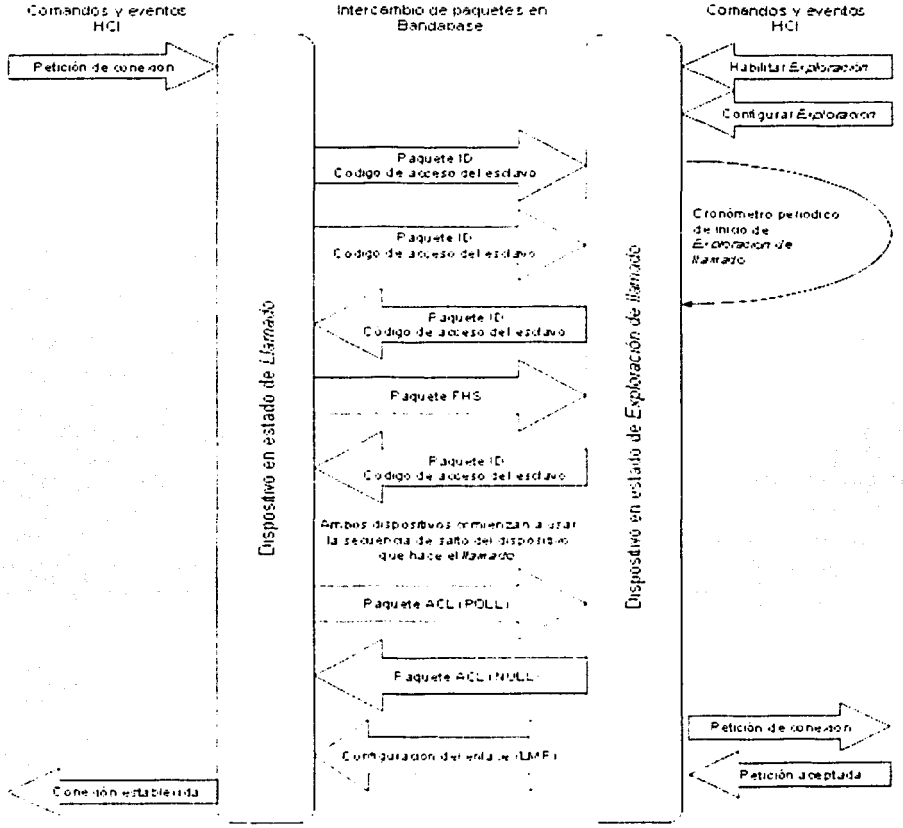


Figura 3.25. Intercambio de mensajes en el proceso de *paging*

El *pager* transmite paquetes ID con la dirección del dispositivo *explorador*, si el dispositivo *explorador* (*page scanner*) está escuchando en ese momento, responderá con otro paquete ID con su propia dirección. En este caso, a diferencia del proceso de *inquiry*, el paquete va dirigido únicamente al dispositivo *explorador*, ya que contiene su código de acceso, es por ello que no se requiere que transcurra un tiempo aleatorio para que el dispositivo *explorador* conteste.

Cuando el dispositivo que realizó la invitación recibe el paquete ID, transmitido por el explorador, inicia el envío del paquete FHS. En el momento que el *explorador* recibe el paquete FHS, envía un paquete ID para confirmar la recepción.

Una vez que el *explorador* tiene el paquete FHS, extrae de él la información necesaria para establecer *conexión*. El paquete FHS contiene información del *pager*, tal como su reloj y su dirección *Bluetooth*, con estos datos el *explorador* calcula la secuencia de salto del *pager* y empieza a utilizarla en lugar de la secuencia de salto especial de este proceso, que usaba anteriormente. En este punto inicia el estado de *conexión* donde el *pager* se convierte en el Maestro y el *explorador* en el Esclavo.

Después de que ambos dispositivos han cambiado a la nueva secuencia de salto, el Maestro envía un paquete POLL para verificar que la secuencia calculada haya sido la correcta. El Esclavo debe responder con un paquete ACL, que generalmente es un paquete NULL. A este proceso sigue el intercambio de paquetes LMP de configuración del enlace. En este momento, el Maestro y el Esclavo pueden acordar intercambiar papeles si se requiere. Si el paquete POLL, o el ACL, no son recibidos en un determinado tiempo, ambos dispositivos regresarán a los estados de *page* y *page scan*.

Durante este proceso, el *pager* hace una aproximación de la secuencia de salto del *explorador* mediante una estimación (CLKE) del reloj del *explorador* (CLKN), usando los datos recuperados en el proceso de *inquiry*. El dispositivo en estado de *page* recorre la secuencia de salto con una tasa de dos saltos por ranura de tiempo. Lo anterior es necesario debido a que el CLKE, es impreciso, o no existe en el caso que no se haya realizado anteriormente el proceso de *inquiry*. El dispositivo *explorador* salta sobre los mismos canales, pero con una tasa de 1.28 segundos por salto, o bien, una vez cada 2.048 ranuras de tiempo. Este mecanismo asegura que exista una coincidencia eventual en el menor tiempo posible.

A continuación se describirá la temporización del proceso del *llamado* mostrado en la figura 3.26. En esta figura f(x) representa la frecuencia, basada en el *explorador*, utilizada en la transmisión o recepción; F(x) representa la frecuencia, basada en el *Maestro*, que se utilizará en la transmisión o recepción durante el estado de conexión.

Una vez que el *explorador* ha recibido el primer paquete ID, congela su reloj de salto y empieza a saltar en secuencia con el dispositivo *pager*.

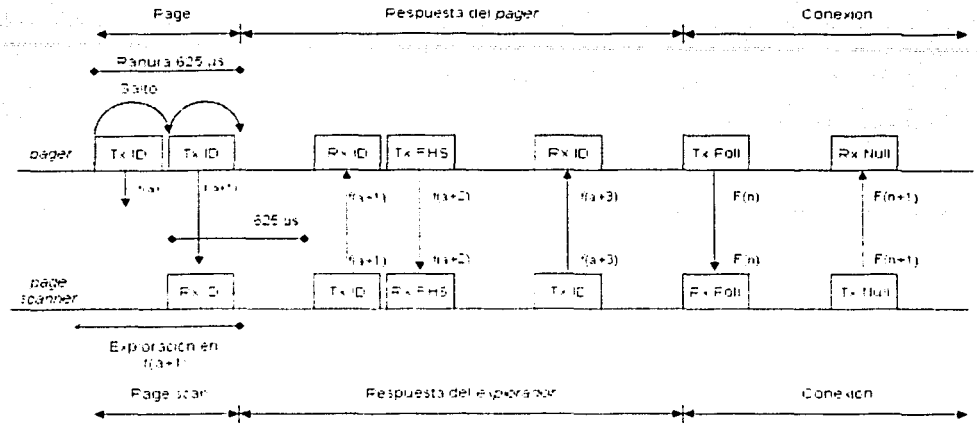


Figura 3.26. Temporización del proceso de *paging*

Cuando el *pager* recibe el ID, transmite un paquete FHS en el comienzo de la siguiente ranura, de esta manera permite al Esclavo ajustar su referencia de ranuras de tiempo. El Esclavo, por tanto, debe abrir su correlador media ranura después de haber enviado el paquete ID, y mantenerlo abierto por lo menos 625 µs, pues el paquete FHS puede llegar tanto en la próxima media ranura, o en la próxima ranura completa. Después de la recepción del paquete FHS, el Esclavo responde con un ID. Hasta este punto ambos dispositivos estaban usando el código de acceso (DAC) del Esclavo, la temporización del Esclavo (Esclavo: CLKN, Maestro: CLKE), y una secuencia de salto de *paging* derivada de la parte baja de la dirección *Bluetooth* (LAP) del Esclavo. La siguiente transmisión la realizará el Maestro usando el código de acceso del canal (CAC) y su propio CLKN, el Esclavo usará el CLK (CLKN del Maestro) y reconocerá el CAC derivado del Maestro, con lo que se habrá establecido la conexión. A continuación los dispositivos intercambiarán paquetes POLL y NULL para verificar el estado de la *conexión*.

3.3.4 Operación de una *piconet*

En una *piconet*, el Maestro transmite y recibe información de cada uno de los esclavos que se encuentren activos en ese momento, los cuales tienen asignada una dirección de miembro activo diferente. En caso que no exista nada que transmitir el Maestro puede omitir al Esclavo, o mandarle un paquete nulo (NULL). Si hay un enlace SCO en operación con cierto Esclavo, entonces se debe establecer comunicación con él de acuerdo al periodo de repetición T_{sco}. El protocolo de control de enlace y la información de la secuencia de retransmisión se deben mantener, a través de todos los demás enlaces, listos para la siguiente comunicación [24].

3.3.4.1 Enlaces de tipo ACL

El Maestro tienen la capacidad de comunicarse con varios esclavos subsecuentemente, las transmisiones hacia los esclavos se realizan en las ranuras impares, mientras que las respuestas de los esclavos son transmitidas en las ranuras pares.

El Maestro, en los enlaces ACL, sólo transmite cuando existe alguna información que enviar, mientras que el Esclavo solamente responde a los mensajes que el Maestro le envia. Debido a lo anterior, es posible que el Maestro omita la transmisión a alguno de los esclavos y en su lugar establezca comunicación con el siguiente.

3.3.4.2 Enlaces SCO

En los enlaces SCO, a diferencia de los ACL, existen ranuras de tiempo reservadas para la comunicación entre el Maestro y determinado Esclavo. Es decir, cada periodo de tiempo, T_{sco}, el Maestro está obligado a transmitir información hacia dicho Esclavo, y éste esta obligado a responder al Maestro, aunque no haya detectado alguna transmisión del Maestro hacia él en la ranura anterior.

Las ranuras de tiempo reservadas en los enlaces SCO son dos, una para recepción y otra para transmisión, éstas pueden estar espaciadas cada uno, dos o tres pares. El espaciamiento de las ranuras SCO se establece al inicio del enlace y dichas ranuras quedan reservadas hasta que el enlace termine. Debido a que el espaciamiento más grande entre las ranuras SCO es cada tres pares, el máximo número de enlaces SCO que puede soportar una *piconet* es tres.

En presencia de estos enlaces, la recepción y transmisión de información SCO continuará en las ranuras reservadas, aún durante los procedimientos de *inquiry*, *paging*, *inquiry scan* y *page scan*. Lo anterior reduce la oportunidad de descubrir a los dispositivos vecinos y conectarse con ellos.

Si se encuentra activo un enlace SCO cada tres pares de ranuras, se pierde aproximadamente el 60% del tiempo de *exploración*; cuando dos enlaces están activos cada tres pares de ranuras, entonces alrededor del 90% del tiempo de *exploración* se pierde. Para contrarrestar este efecto, se recomienda, para el primer caso, duplicar el número de veces que se realiza el procedimiento de *inquiry* y; triplicarlo para el segundo caso. Para la *exploración* es recomendable aumentar el tamaño de la ventana de 11.25 ms, establecido por *default*, a 36 ms para el primer caso y a 54 ms para el segundo caso [2]. Sin embargo, lo mejor para tener un proceso de *exploración* confiable, es detener los enlaces SCO durante este procedimiento.

Para el proceso de *page* y *page scan* la interferencia es mayor, debido a que la secuencia de paquetes intercambiados es tambien mayor.

3.3.5 Operación de una *scatternet*

Una *scatternet* se compone de dos o más *piconets*, donde al menos un dispositivo está activo en ambas *piconets*. La conmutación entre *piconets* requiere mantener dos bases de tiempo, pues cada *piconet* está sincronizada al reloj de un Maestro diferente. El dispositivo que se encuentre activo en dos *piconets* debe mantener y seleccionar entre dos relojes y dos códigos de acceso [24].

Debido a que las bases de tiempo de las *piconets* son asíncronas entre sí, la conmutación entre ellas requiere ranuras de tiempo de guarda, las cuales permiten la sincronización a la nueva base de tiempo. En el peor de los casos, las bases de tiempo podrían estar fuera de sincronía un par de ranuras. Lo anterior impone un límite al número de *piconets* que pueden conectarse entre sí.

Las ranuras de guarda mencionadas limitan el desempeño de un dispositivo presente en dos *piconets*, es decir, éste requerirá un mayor ancho de banda que un dispositivo presente en una sola *piconet*.

Cuando un dispositivo esta ocupado en otra *piconet*, no puede ser contactado por su Maestro y por ello debe advertirle a éste de los periodos que estará ocupado, de esta manera el Maestro no asumirá que se ha perdido la comunicación totalmente.

Para los enlaces SCO el máximo espaciamiento es cada tres pares de ranuras, lo cual no permite que un dispositivo tenga enlaces SCO en dos *piconets* a la vez. Sin embargo, sí es posible un enlace SCO en una *piconet* y un enlace ACL en otra.

3.3.6 Operación con baja potencia

Durante los estados de *Standby*, *Park*, *Hold*, *Sniff*, o incluso entre las operaciones de transmisión y recepción en modo *Activo*, un dispositivo puede entrar en el modo de operación con baja potencia, en el cual los protocolos y los elementos de procesamiento se apagan, los relojes son desactivados y el dispositivo entra en un modo de operación de muy bajo consumo de energía. Algunos datos estáticos deben mantenerse, como el protocolo de control de enlace, el *buffer* de contenidos y el reloj del dispositivo (CLKN). Debido a la necesidad de bajo consumo de energía el reloj disminuye su exactitud.

3.3.7 Relación del *controlador de enlace* con *bandabase*

Desde una perspectiva general, se puede decir que, la entidad de control de enlace controla y dirige la secuencia de las operaciones y funciones de *bandabase*. Este control se puede resumir, de una manera muy simple, en dos máquinas de estado: una para la recepción (figura 3.27) y otra para la transmisión (figura 3.28). La transmisión se dispara con base en el reloj CLK, mientras la recepción depende de la señalización del correlador [5].

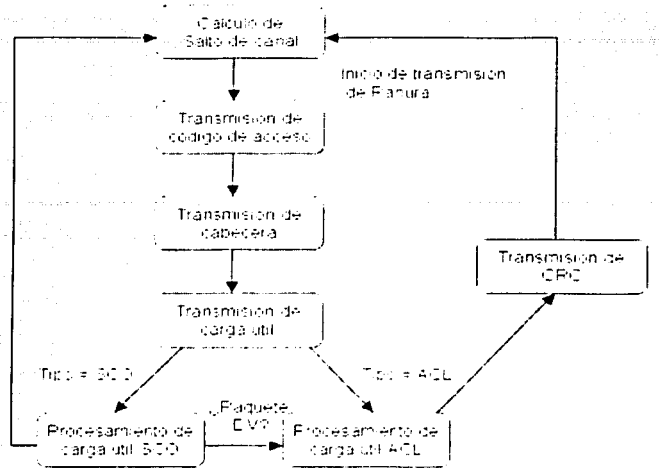


Figura 3.27. Máquina de estado de transmisión de la entidad de *controlador de enlace*

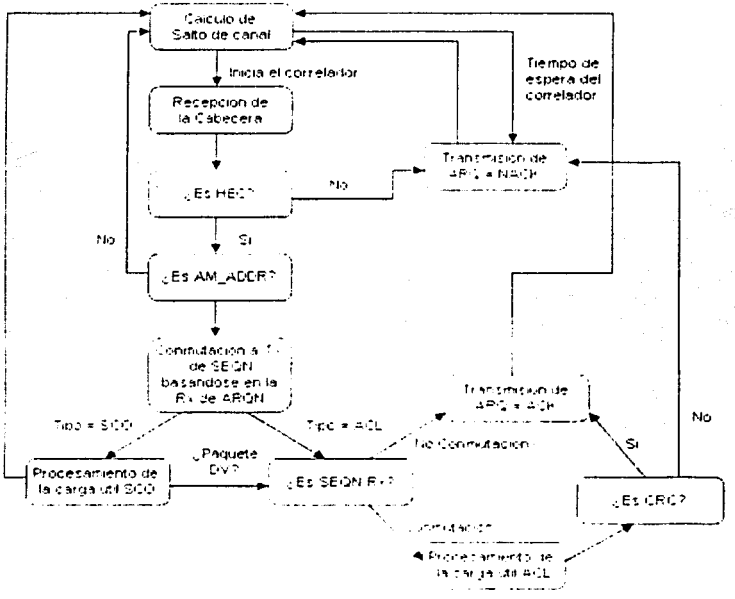


Figura 3.28. Máquina de estado de recepción de la entidad de *controlador de enlace*

3.4 Administrador de Enlace

El *host* controla un dispositivo *Bluetooth* a través de comandos de la *interfaz de control del host* (HCI), pero el que traduce estos comandos a operaciones al nivel de *bandabase* es el *Administrador de enlace* (LM), mediante las siguientes operaciones, [22]:

- Agregar Esclavos a una *piconet* y asignar direcciones de miembros activos.
- Terminar conexiones para separar Esclavos de una *piconet*.
- Configurar el enlace incluyendo el control de la conmutación Maestro/Esclavo.
- Establecer enlaces ACL (para datos) y SCO (para voz).
- Configurar conexiones en los modos de bajo consumo de potencia: *Hold, Sniff y Park*.
- Controlar los modos de prueba.

La entidad de administración de enlace de un dispositivo *Bluetooth* se comunica con las entidades de administración de enlace de otros dispositivos, utilizando el Protocolo de Administración de Enlace (LMP, *Link Management Protocol*).

La especificación *Bluetooth* define los mensajes LMP intercambiados entre administradores de enlace, pero no especifica en detalle cómo son llevadas a cabo las instrucciones del protocolo. Sin embargo, dado el contenido de los mensajes, se puede decir que el *administrador de enlace* controla la administración de una *piconet* (estableciendo y destruyendo enlaces), configura el enlace, y realiza funciones de seguridad.

Algunos mensajes LMP requieren respuesta, otros no. Si un *administrador de enlace* (LM) recibe un mensaje el cual no está permitido sobre el enlace y dicho mensaje requiere una respuesta, entonces se responde con un mensaje que indica que el mensaje no fue aceptado (LMP_*not_accepted*).

3.4.1 Unidades de datos del protocolo LMP (PDUs, *Protocol Data Units*)

Como ya se mencionó, los administradores de enlace se comunican con sus similares en otros dispositivos utilizando el protocolo de administración de enlace (LMP), el cual define una serie de mensajes los cuales son intercambiados entre administradores de enlace.

Cada mensaje LMP comienza con un identificador de un bit (TID, *transaction identifier*), el cual es 0 si el Maestro inicia la transacción y 1 si el Esclavo inicia la transacción. Una transacción puede tomar muchos intercambios de paquetes LMP, por ejemplo todos los intercambios requeridos para preparar el *acoplamiento (pairing)*, incluyendo la autenticación mutua después de la creación de un enlace, forman una sola transacción, así todos usan el identificador de transacción del primer paquete.

Al identificador de transacción (TID) le sigue un código de operación (OpCode) de 7 bits, el cual identifica el tipo de mensaje LMP que ha sido enviado. Al OpCode le siguen los parámetros de mensaje, cada uno de los cuales ocupan un número entero de bytes.

En la figura 3.29 se muestra la estructura de un PDU del protocolo de administración del enlace.

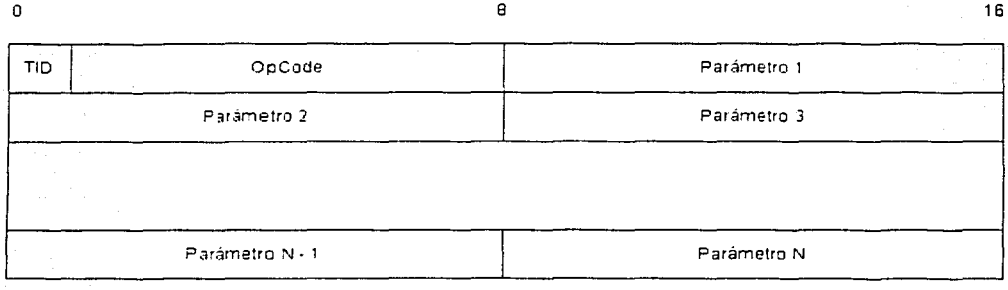


Figura 3.29. Estructura de un PDU del protocolo de administración de enlace

3.4.2 El canal de administración de enlace

Los PDUs del LMP son enviados como paquetes de una sola ranura sobre el canal lógico de administración de enlace. Este canal se identifica estableciendo el valor 11 en el campo de canal lógico ubicado en la cabecera de paquete.

El *administrador de enlace* envía sus mensajes al *controlador de enlace* para la transmisión. El *controlador de enlace* garantiza comunicarse sobre el enlace sólo una vez cada T_{sondeo} ranuras, es por ello que puede existir un tiempo de retraso considerable entre el envío de un mensaje del *administrador de enlace* hacia el *controlador de enlace* y la transmisión de dicho mensaje al aire. El *administrador de enlace* tiene que estar enterado de esta posibilidad de retraso en la transmisión, debido a que puede afectar la sincronía de las transacciones, la conmutación Maestro/Esclavo y el comienzo del modo *hold*.

Debido a que el control de flujo detiene la transmisión de los paquetes de datos de L2CAP, no lo utiliza el LMP. Para lograr esto, el bit de flujo en los paquetes PDU del LMP se pone siempre en 0, por el dispositivo que transmite. En cualquier caso, este valor no importa, pues el dispositivo que recibe lo ignora.

3.4.3 Establecimiento del enlace

El *administrador de enlace* es responsable del establecimiento y la administración de las conexiones en *bandabase* que enlazan a los dispositivos *Bluetooth*. El *administrador de enlace*

establece enlaces ACL por medio de la *bandabase*, entonces los mensajes LMP se pueden utilizar para establecer un enlace SCO (para voz) a través de una conexión ACL ya existente.

El *administrador de enlace* actualiza los datos de los Esclavos a los que ha asignado una dirección de miembro activo (AM_Addr).

En la figura 3.30 se muestran los mensajes que forman parte del establecimiento de una conexión ACL. La capa del *controlador de enlace* debe establecer un enlace entre dispositivos antes de que los mensajes LMP se intercambien.

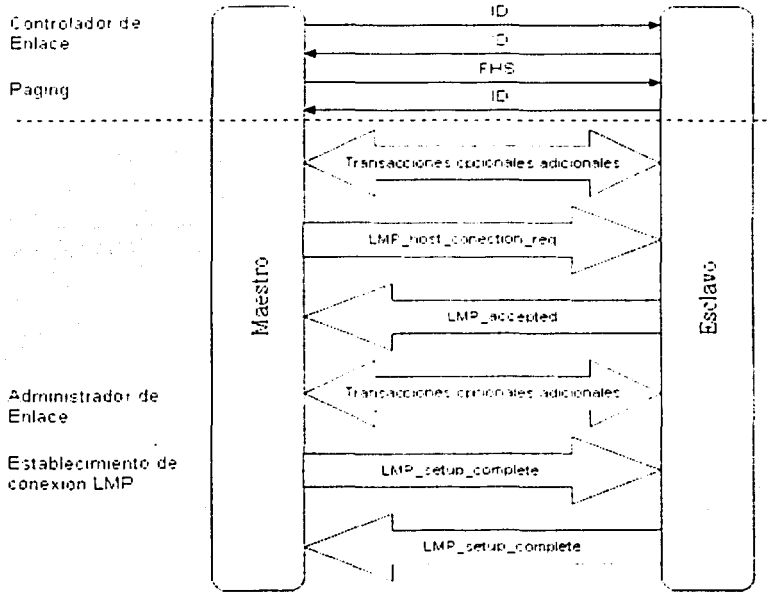


Figura 3.30 Secuencia de mensajes LMP para el establecimiento de un enlace ACL

Entre el establecimiento de la conexión del *controlador de enlace* y el envío de la petición de conexión a nivel LMP (LMP_*host_connection_req*) del *administrador de enlace*, se puede enviar cierto número de mensajes, tales como:

- LMP_*clkoffset_req*
- LMP_*name_req*
- LMP_*detach*
- LMP_*version_req*

Los primeros tres mensajes se pueden usar para conectarse temporalmente a un dispositivo, recuperar su nombre y reloj, y separarse antes de que el *host* sea informado del

enlace. El mensaje de *offset* del reloj es una parte útil de esta secuencia pues permite al Maestro acelerar el proceso de *paging* en la siguiente ocasión en que establezca una conexión prediciendo dónde, en su secuencia de salto, un Esclavo estará *explorando*.

El LMP tiene un tiempo máximo de espera de 30 segundos[11] entre un mensaje enviado y su respuesta recibida. Si se excede el tiempo de espera, el enlace se considera como perdido.

El mensaje LMP_*setup_complete* es generado de forma independiente por el Maestro o el Esclavo cuando cada uno está de acuerdo con el estado del enlace, en otras palabras, es un mensaje de respuesta satisfactoria a la transacción anterior

Debido a que cada *administrador de enlace* decide independientemente cuándo enviar un mensaje, cada LMP_*setup_complete* cuenta como una transacción por separado y lleva su propio identificador de transacción (TID). *Bluetooth* establece que cuando el Maestro envía el mensaje LMP_*setup_complete*, se debe poner el identificador de transacción a 0, y cuando el Esclavo envía el mensaje se debe poner el identificador de transacción a 1.

Una vez que el enlace ACL ha sido establecido, ya sea el Maestro o el Esclavo, puede pedir la preparación del enlace SCO a través del enlace ACL. Tanto el Maestro como el Esclavo pueden usar una petición LMP SCO para iniciar la preparación de una conexión SCO como se muestra en la figura 3.31.

Cuando el Maestro pide un enlace SCO, envía un LMP_SCO_req que contiene los siguientes parámetros del enlace:

- Manejador o identificador del enlace SCO.
- Banderas de control de temporización (utilizadas para pedir un cambio de temporización en un enlace establecido).
- D_{SCO}, que indica cuando se llevará a cabo la primera ranura SCO.
- T_{SCO}, que es el intervalo que separa a las ranuras SCO reservadas.
- Tipo de paquete SCO para usar: HV1, HV2, HV3 y DV.
- Modo de codificación: ley u, ley A y CVSD.

Un Maestro puede tener enlaces SCO con muchos Esclavos a la vez. Una vez que un Maestro ha preparado un enlace SCO, tiene cierta libertad para asignar otras ranuras para otros enlaces. Por lo tanto, tiene sentido dejar a los Maestros escoger los parámetros de temporización para los enlaces SCO. Generalmente, un Maestro manejará enlaces con muchos Esclavos, por lo que es el Maestro quien escoge los identificadores de conexión. Si un Esclavo escoge un identificador, existe el riesgo de tomar uno que se encuentre ya en uso por otro

---

Antes de la versión 1.1 de *Bluetooth*, hubo problemas con el establecimiento inicial del enlace debido a que no había un temporizador en marcha antes de que el tráfico LMP comenzara. Esto daba por resultado que si el administrador de enlace no comenzaba correctamente, no había forma de detectarlo y dar por terminado el enlace. La versión 1.1 resolvió este problema especificando que se debía aplicar un tiempo de espera de 30 segundos a la respuesta LMP, al tiempo entre el establecimiento de la conexión a nivel de control de enlace y el envío del LMP_*host_connection_req*.

enlace. Así, cuando un Esclavo envía una petición para establecer un enlace SCO (LMP_SCO_*request*), los parámetros de temporización y el identificador no son válidos. Si el Maestro está en disposición de establecer el enlace SCO, responde con un mensaje LMP_SCO_*request* que contiene parámetros válidos, y el Esclavo responderá con un mensaje LMP_*accepted*, tal y como lo hubiera hecho si la transacción hubiese sido iniciada por el Maestro.

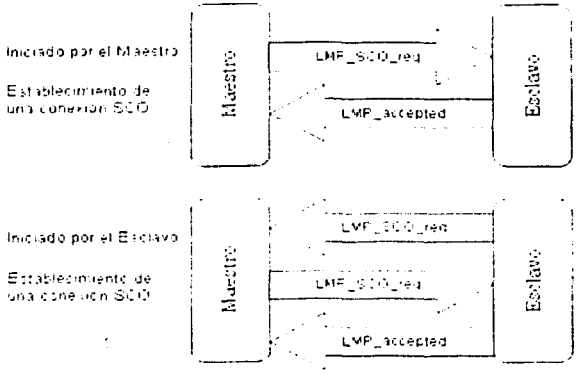


Figura 3.31. Secuencia de mensajes LMP para el establecimiento de una conexión SCO.

3.4.4 Término del enlace LMP

En cualquier momento en el que un Maestro o un Esclavo deseen terminar un enlace *Bluetooth*, se envía un comando LMP_*detach*. Entre las posibles razones para terminar el enlace, incluidas en el comando mencionado, están las siguientes:

- Conexión terminada por el usuario.
- Bajos recursos.
- Acercarse a la potencia de apagado.

Cuando un Esclavo se ha apartado de una *piconet*, el Maestro puede volver a utilizar la dirección de miembro activo de ese Esclavo para identificar un nuevo Esclavo. Si un Esclavo no se separó correctamente, puede haber problemas con dos Esclavos tratando de responder a la misma dirección de miembro activo (AM_ADDR). Existe un procedimiento de *separación* utilizado por el administrador para asegurar que un Esclavo ha sido separado correctamente, este es:

- El *Administrador de enlace* (LM) inicia el proceso de separación terminando de enviar el paquete ACL que se encuentra transmitiendo, con datos L2CAP.
- El LM inicia el proceso de *separación* suspendiendo el envío de paquetes L2CAP.

- El LM inicia el proceso de *separación*, preparando un mensaje LMP_*detach* para transmisión y comienza un *timer* de $6*T_{sondeo}$ ranuras, donde T_{sondeo} es el intervalo de sondeo para la conexión.
- Si el *timer* de $6*T_{sondeo}$ se termina antes de que la confirmación de *bandabase* del LMP_*detach* se reciba, el enlace se abandona y se inicia un tiempo de espera de supervisión de enlace. Si se recibe una confirmación de *bandabase*, se inicia un cronómetro de $3*T_{sondeo}$ slots.
- Cuando el segundo *timer* ($3*T_{sondeo}$) se termina, entonces el AM_ADDR puede utilizarse nuevamente.

También se utiliza un *timer* en el lado de la recepción: Si un Maestro recibe un mensaje LMP_*detach*, comienza un cronómetro de $6*T_{sondeo}$ ranuras; si un Esclavo recibe un mensaje de LMP_*detach*, comienza un cronómetro de $3*T_{sondeo}$ ranuras. Cuando este cronómetro se termina, el enlace puede ser abandonado y el AM_ADDR de dicho enlace puede volverse a usar.

Si el mensaje LMP_*detach* se pierde debido a interferencia o a que los dispositivos se han movido fuera de rango, entonces el tiempo de espera de supervisión de enlace terminará y el enlace será terminado de cualquier modo.

3.4.5 Cambio de papeles Maestro-Esclavo

Normalmente, el dispositivo *pager* se convierte en el Maestro de la *piconet*, y el dispositivo que hace la *exploración* de *llamado* se convierte en el Esclavo. El Esclavo sólo puede transmitir en respuesta a una transmisión del Maestro (excepto por las ranuras SCO reservadas). El Maestro también determina el tamaño de los paquetes, los intervalos SCO, y la temporización. Todo esto quiere decir que el Maestro controla el ancho de banda disponible para el Esclavo.

En algunos casos, los roles en los que se encuentren el Maestro y el Esclavo pueden no ser los apropiados. Por esto, los dispositivos pueden dar por terminado el enlace y comenzar a establecer el enlace con los roles invertidos, pero esto puede ser un procedimiento que lleve mucho tiempo, y es obvio que, durante el tiempo en el que el enlace es reestablecido, no se pueden transmitir datos. Para evitar está pérdida de tiempo, el LMP proporciona una forma para el intercambio de roles, sin dar por terminado el enlace.

El dispositivo que quiere iniciar el cambio de roles envía un LMP_*switch_req*. Este paquete contiene un parámetro que especifica exactamente cuándo debe realizarse el intercambio de papeles Maestro Esclavo. Lo anterior se debe a que algunas implementaciones no pueden reaccionar rápidamente a las peticiones de conmutación. Para asegurar que haya el suficiente tiempo para la inversión de papeles, esta debe suceder $2*T_{sondeo}$, o 32 ranuras después de la petición.

Si el Maestro envía el LMP_*switch_req*, el Esclavo responde con un mensaje LMP_*slot_offset*. Si el Esclavo inició el cambio, se envía el LMP_*switch_req*, después del mensaje LMP_*slot_offset*. La figura 3.32 muestra el intercambio de mensajes de este proceso.

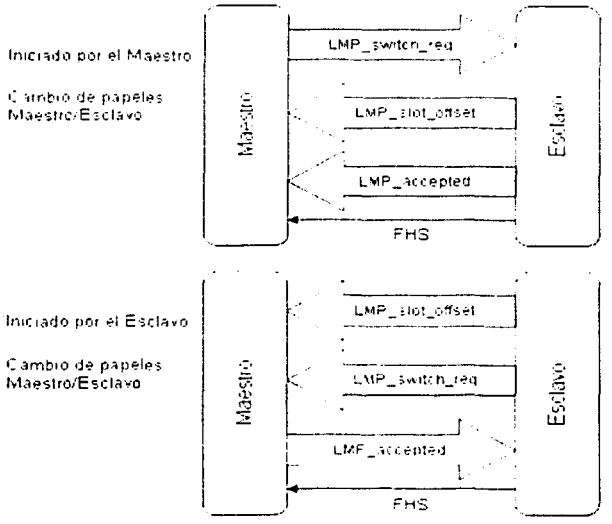


Figura 3.32. Secuencia de mensajes LMP para el intercambio de roles.

El mensaje LMP_*slot_offset* contiene un BD_ADDR y una compensación de tiempo de la ranura. El BD_ADDR es la dirección del Esclavo, que se va a convertir en Maestro después del cambio.[12]

Durante el cambio de papeles Maestro/Esclavo, se usa un paquete FHS para sincronizar los dos dispositivos; sin embargo, para obtener una mejor precisión en la sincronía, el dispositivo que comienza como Esclavo usa un mensaje de LMP *slot_offset* para enviar la diferencia entre su reloj, y el reloj del otro dispositivo. Si el Esclavo pide el cambio, éste envía dicho mensaje antes del LMP_*switch_req*. Si el Maestro pide el cambio, el Esclavo envía el LMP_*slot_offset* antes del mensaje LMP_*accepted*.

Si el cambio de roles es aceptado, el Esclavo deberá convertirse en Maestro. Esto significa que el Maestro debe sincronizarse al reloj *Bluetooth* del Esclavo. Sin importar qué lado inició el cambio, el Esclavo envía al Maestro un paquete FHS para permitirle sincronizarse a su reloj. Después de que el paquete FHS ha sido confirmado ambos dispositivos se cambian a la nueva sincronización.

[12] Se requiere de un cálculo para obtener dicha compensación de tiempo

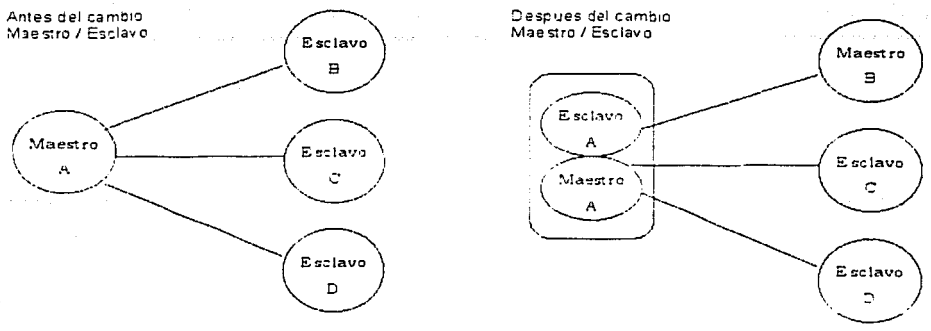


Figura 3.33. Configuración de una *piconet* antes y después del cambio de Maestro/Esclavo.

En la figura 3.33 se muestra qué le sucede a un Maestro con tres Esclavos cuando acepta un cambio de Maestro/Esclavo con uno de los Esclavos. La *piconet* del Maestro se convierte en dos *piconets* unidas en una *scatternet* por el Maestro que aceptó el cambio de papel. El Maestro *conmutado* (dispositivo A) ahora tiene un papel dual: es Maestro de los Esclavos con los que no hizo el cambio, y es Esclavo del otro dispositivo con el que realizó el cambio.

Si el dispositivo A no soporta *scatternets*, o no tiene los suficientes recursos para estar presente en dos *piconets*, puede escoger entre rechazar el cambio de Maestro/Esclavo, o desconectar los Esclavos D y C.

3.4.6 Control de paquetes multi ranura

Cuando un enlace comienza a prepararse usa paquetes multi ranura o *multislot*. Los paquetes *multislot* hacen más eficiente el uso del ancho de banda, pero hay circunstancias en las que no deben usarse, por ejemplo:

- En enlaces ruidosos, pues los paquetes *multislot* son más propensos a ser corrompidos por una ráfaga de interferencia.
- En enlaces SCO, pues no existe suficiente espacio entre las ranuras reservadas para el manejo de paquetes *multislot*.

3.4.7 Seguridad

La *bandabase* de *Bluetooth* incluye una generación de código de encriptación y una máquina de cifrado. Para la encriptación y des-encriptación, esta máquina debe estar configurada y controlada por las capas superiores, y ese control debe estar sincronizado en ambas terminales del enlace *Bluetooth* [22]. El LMP da el mecanismo para negociar los modos de encriptación y

coordinar los códigos de encriptación usados por los dispositivos que estén formando el enlace[13].

3.4.8 Control de potencia

El LMP soporta el control de modos de baja potencia de consumo. En estos modos, el Maestro y el Esclavo permanecen sincronizados, pero ahorran potencia dejando la conexión inactiva por periodos definidos.

La duración de las baterías estará radicalmente afectada por la potencia usada por el sistema de radio, debido a que prácticamente éste representa el mayor gasto de potencia en todo el sistema. Debido a que *Bluetooth* formará parte de dispositivos pequeños y portátiles, alimentados por baterías, es necesario mantener un bajo consumo de potencia para extender la vida de dichas baterías.

Si las *piconets Bluetooth* operan cerca una de la otra, las transmisiones de radio tenderán a interferirse. Mientras menor sea la potencia, menor será la interferencia que habrá entre *piconets* adyacentes; por lo que el uso de mínima potencia permite que más *piconets* coexistan en un cierto espacio.

La potencia mínima es importante. Sólo el dispositivo receptor sabe qué recepción es lo suficientemente buena como para reducir la potencia, por lo tanto es éste el que hace las peticiones para cambiar la potencia transmitida.

Un dispositivo que quiere cambiar los niveles de potencia envía un LMP_*incr_power* para aumentar la potencia, o un LMP_*decr_power_req* para disminuirla. No hay necesidad de una respuesta LMP_*accepted* a estas peticiones debido a que si la petición no es recibida, el dispositivo que la hace detectará que la potencia sigue siendo la incorrecta por lo que vuelve a hacer la petición.

Existen límites en el incremento, o decremento, de potencia en el radio de los dispositivos *Bluetooth*. Para el límite de incremento, llega un punto donde la potencia ya no puede incrementarse más, sin importar que tan mala sea la recepción. De la misma forma, un radio puede reducir su potencia hasta un límite antes de que el dispositivo se apague.

Para evitar que un dispositivo haga peticiones repetidamente si no se satisface su nivel de potencia, el LMP posee mensajes informativos. Estos mensajes se envían cuando se ha llegado al límite superior, o inferior, de potencia. Un mensaje LMP_*max_power* se envía al dispositivo que hizo la petición de incremento; y un mensaje LMP_*min_power* cuando se hizo la petición de decremento. Este procedimiento se muestra en la figura 3.34

[13] Una descripción más detallada de los métodos de encriptación se encuentra en el capítulo 6 ("Interacción entre las capas de la pila de protocolo *Bluetooth*") de esta tesis.

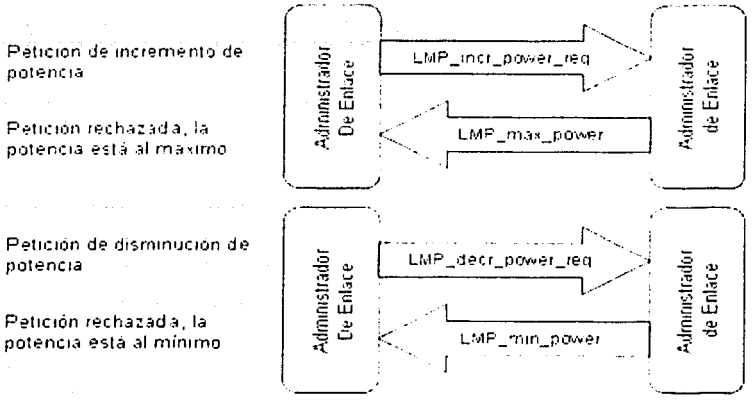


Figura 3.34. Secuencia de mensajes LMP para el cambio de niveles de potencia cuando la emisión de potencia se encuentra en el límite.

3.4.9 Calidad de servicio (QoS)

El LMP soporta mensajes para configurar la calidad de servicio en una conexión. También soporta mensajes que pueden configurar un canal para que automáticamente cambie los tipos de paquetes de acuerdo con la calidad del canal físico. De esta forma los datos pueden ser transferidos a una tasa de transmisión más alta cuando la calidad del canal físico sea buena, y a una menor tasa, con mejor protección a errores, cuando la calidad del canal se deteriore.[4]

3.5 Interfaz de Control del Host (HCI)

Algunos sistemas *Bluetooth* tienen las capas de *bandabase y administrador de enlace* en un procesador, mientras que las capas superiores (L2CAP, SDP, RFCOMM) y aplicaciones se encuentran en el procesador del *host*. Debido a este hecho es necesaria la existencia de una interfaz entre ambos grupos de capas. *Bluetooth* define la *Interfaz de Control de Host (HCI)* para satisfacer esta necesidad. En la figura 3.35 se muestra la ubicación del HCI dentro de la pila de protocolos [5].

En el caso de una PC y una tarjeta PCMCIA *Bluetooth* que contiene las capas de bajo nivel, el diseño anterior presenta las siguientes ventajas:

- El procesador del *host* (en este caso la PC) tiene capacidad de sobra para manejar las capas superiores y aplicaciones, lo que permite que el dispositivo *Bluetooth* requiera menos memoria y un procesador menos poderoso, con lo que se reduce su costo.

[4] En el capítulo se habla más a detalle de la calidad de servicio

- El procesador del *host* puede estar inactivo y ser activado por una petición del módulo *Bluetooth*.

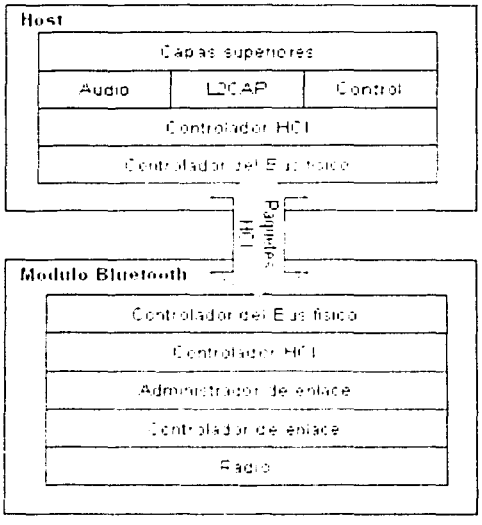


Figura 3.35. Ubicación del HCI dentro de la pila de protocolos

La separación de capas garantiza una respuesta rápida, del orden de microsegundos, necesaria en las capas de bajo nivel, debido a que no requiere interrupciones en el procesador del *host* cada vez que se reciba, o transmita, un mensaje entre las capas bajas de los dispositivos involucrados.

La estandarización de la *Interfaz de Control de Host* hace posible la compatibilidad entre los *host* y los módulos *Bluetooth* de diversos fabricantes.

A pesar de las consideraciones anteriores existen casos en los que es preferible tener todo el sistema en un solo procesador, como por ejemplo en un *headset*, donde se requiere que el sistema sea pequeño, ligero, de bajo consumo de energía y barato.

3.5.1 Tipos de paquetes HCI

El estándar *Bluetooth* para la *Interfaz de Control de Host (HCI)* define los siguientes tipos de paquetes:

3.5.1.1 Paquetes de comandos HCI

Los comandos HCI son usados por el *host* para controlar al módulo *Bluetooth* y para monitorear su estado. Los comandos HCI se transmiten usando los paquetes de comandos HCI. La estructura de los paquetes HCI se muestra en la figura 3.36. Estos empiezan con un campo de código (*Opcode*) de 2 bytes, el cual identifica el tipo de comando; sigue un campo de *longitud* que indica el número de bytes de los parámetros subsecuentes; luego siguen los parámetros, lo cuales ocupan un número entero de bytes cada uno.

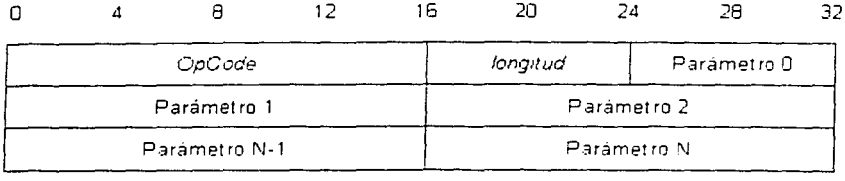


Figura 3.36. Estructura de un paquete de comando HCI.

3.5.1.2 Paquetes de eventos HCI

Estos paquetes son usados por el módulo para informar al *host* de cambios en las capas inferiores. Su estructura es similar a los paquetes de comandos. Poseen un *código de evento* que identifica al paquete, su función es la misma que el *OpCode*; sigue un campo de *longitud* que indica el número de bytes de los parámetros subsecuentes; luego siguen los parámetros, los cuales ocupan un número entero de bytes cada uno. Este tipo de paquetes se ilustra en la figura 3.37.

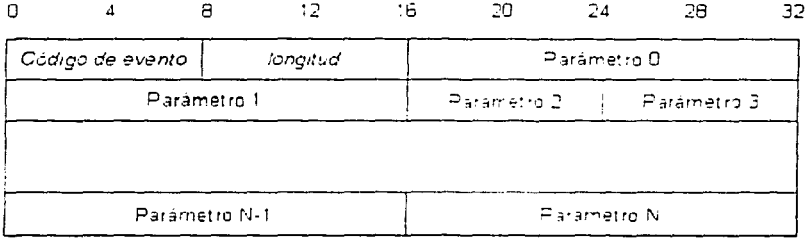


Figura 3.37. Estructura de un paquete de evento HCI.

3.5.1.3 Paquetes de datos HCI

Estos paquetes se usan para transmitir información de voz (SCO) y datos (ACL) a través del HCI, entre el *host* y el módulo. Los paquetes usados en ACL son diferentes a los paquetes SCO

tanto en tamaño como en formato. El paquete para un enlace SCO es más pequeño debido a que el tiempo de espera entre paquetes debe ser corto.

La estructura de un paquete HCI ACL contiene los siguientes campos: un identificador, o *manejador de conexión (Connection Handle)*, el cual identifica la conexión ACL a la que pertenecen los datos; una bandera PB (*Packet Boundary*) que determina si el paquete ACL representa el principio, o la continuación de un paquete L2CAP; una bandera BC (*BroadCast*), marca la diferencia entre los datos punto a punto y los datos de *broadcast*; un campo de *longitud* que indica la longitud de los datos en bytes. Esta estructura se muestra en la figura 3.38.

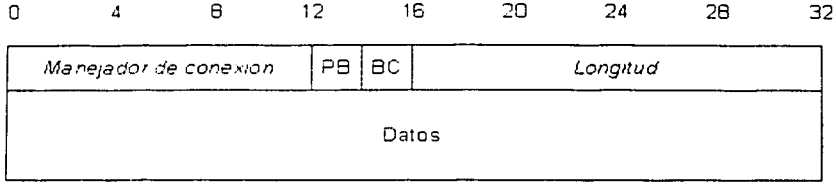


Figura 3.38. Estructura de un paquete de datos HCI ACL

El tamaño del campo de *longitud* es de 2 bytes, de esta manera, la máxima cantidad de datos que puede contener un paquete HCI es 65535 bytes. Cabe mencionar, que muchos módulos *Bluetooth* no poseen *buffers* con la capacidad para recibir paquetes con el máximo tamaño.

La estructura de un paquete HCI SCO es muy similar a la de los paquetes HCI ACL. Los paquetes SCO no poseen las banderas PB y BC; además, su campo de *longitud* es de sólo 1 byte, con lo que se limita su máxima cantidad de datos a 255 bytes.

3.5.1.4 Capa de transporte

Se necesita una capa de transporte para acarrear los paquetes HCI entre el *host* y el módulo *Bluetooth* (vease figura 3.39). Se definen tres tipos de capas de transporte:

- USB (*Universal Serial Bus*), mapea los diferentes tipos de paquetes HCI en las diferentes estructuras (*endpoints*) lógicas del estándar USB.
- RS-232, es una interfaz serial que permite la corrección de errores.
- UART (*Universal Asynchronous Receiver Transmitter*), es una interfaz serial que no permite la corrección de errores.

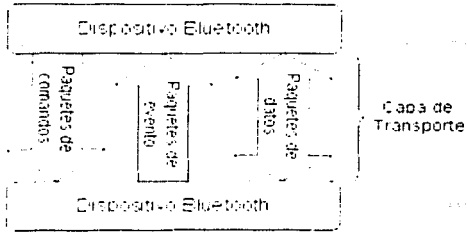


Figura 3.39. Dirección del flujo de paquetes HCI

3.5.2 Control de flujo

Algunas interfaces de transporte HCI ofrecen tasas de datos mucho mayores a las que puede manejar el módulo *Bluetooth* y la interfaz de radio, es decir, el *host* puede pasar datos al módulo más rápido de lo que éste los puede transmitir [22]. El módulo *Bluetooth* puede almacenar los datos recibidos en el *buffer* hasta que puedan ser transmitidos, sin embargo, debido a que el *buffer* es de capacidad limitada es necesario implementar mecanismos de control de flujo. El control de flujo hace posible la disminución de la tasa de transmisión del *host* cuando el *buffer* del módulo esté sobrecargado, y permite utilizar toda la velocidad el resto del tiempo.

3.5.2.1 Comandos de control de flujo

Cada vez que se envía un comando al módulo *Bluetooth*, éste responde con un mensaje HCI_*Command_Complete*, o HCI_*Command_Status*. HCI_*Command_Complete* se usa si el comando puede ejecutarse inmediatamente, de otra manera se utiliza HCI_*Command_Status* y cuando se ejecuta por completo el comando, se envía un HCI_*Command_Complete*.

3.5.2.2 Control de flujo de datos provenientes del *host*

Para el control del flujo de datos se utiliza un mecanismo diferente, el comando HCI_*Read_Buffer_Size* se usa para conocer cuánto espacio tiene el *buffer* del módulo. El módulo responde con un número de paquetes SCO y ACL que puede almacenar en su *buffer*, y con el tamaño máximo de los paquetes de datos HCI SCO y HCI ACL que soporta.

Generalmente los datos SCO no se almacenarán, debido a que éstos pasan con el menor retardo posible. Mientras tanto, los datos ACL se pueden almacenar en el *buffer* sin que se afecte a la aplicación.

Una vez que el *host* sabe cuantos datos SCO y ACL pueden almacenarse, este envía hasta esa cantidad de paquetes al módulo a través del HCI. Una vez que está lleno el *buffer*, el *host* debe esperar a que el módulo le indique cuando haya espacio disponible.

El módulo *Bluetooth* le dice al *host* que su *buffer* está disponible para recibir más datos mediante el evento HCI_*Number_Of_Completed_Packets*. Este evento le indica al *host* cuántos paquetes de datos HCI han sido removidos (limpiados, o enviados) desde el último evento HCI_*Number_Of_Completed_Packets* enviado.

Debido a que es posible manejar varias conexiones a la vez, cada una ellas se identifica con un *manejador*, o *identificador*, mediante el cual se puede saber con precisión a qué enlace pertenece la información del evento enviado.

3.5.2.3 Control de flujo de datos provenientes del módulo *Bluetooth*

Generalmente no hay problema con la capacidad del *host*, sin embargo, existen algunos casos en donde el *host* no puede procesar la información con la misma velocidad con la que el módulo la envía. En estos casos, se implementa un mecanismo de control de flujo igual al anteriormente descrito.

3.5.3 Configuración de los módulos

El HCI provee comandos de configuración para establecer las caracteristicas del módulo local. Debido a que estos comandos no involucran un enlace, entonces se envía de inmediato un evento HCI_*Command_Complete*. Estos comandos obtienen información y/o establecen los valores:

- Versión.
- Nombre local de los dispositivos.
- Clase de dispositivo.
- Controles de voz.
- Caracteristicas opcionales.
- Código del país.
- Dirección *Bluetooth*.

3.5.4 *Búsqueda* (*Inquiry*)

Todos los aspectos del proceso de *inquiry* pueden ser controlados a través del HCI. El comando HCI_*Inquiry* se usa para iniciar el proceso, éste tiene tres parámetros:

- *Inquiry* LAP. El código de acceso que se usará.
- Duración *Inquiry*. El total de la duración del proceso, en unidades de 1.28 s.
- Num_*responses*. El número de resultados del proceso de *inquiry* que se esperarán.

3.5.4.1 Manejo de respuestas

Las respuestas recibidas al proceso de *inquiry* son reportadas al *host* en un evento HCI_*Inquiry_Result*, el cual lleva la información necesaria para conectarse con el dispositivo que generó la respuesta.

Un dispositivo que realiza el proceso de *inquiry* puede no estar interesado en todos los dispositivos en el área. Para tratar con esto, se puede llevar a cabo un filtrado a nivel del módulo *Bluetooth* mediante el comando HCI_*Set_Event_Filter*. El filtrado también se puede llevar a cabo en el nivel de la aplicación, sin embargo, resulta más conveniente realizarlo en el módulo debido a que se optimiza el uso del ancho de banda del HCI, pues de esta manera no se envía información irrelevante al *host*.

3.5.4.2 Paro de la *búsqueda*

Una *búsqueda* puede detenerse porque un número determinado de dispositivos han contestado, o bien, porque el tiempo programado para el proceso ha finalizado. En el caso de que el *host* quiera detener la *búsqueda*, lo puede hacer en cualquier momento mediante el comando HCI_*Inquiry_Cancel*. De cualquier forma que termine la *búsqueda*, se generará un evento HCI_*Inquiry_Complete*.

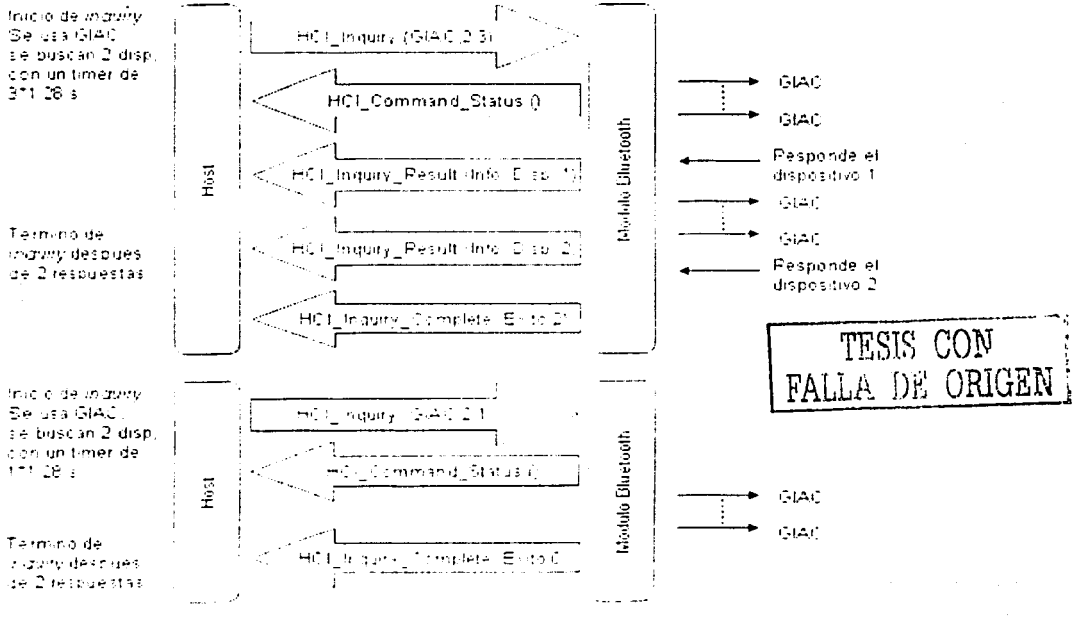


Figura 3.40. Secuencia de mensajes para la terminación del estado de *inquiry*

En la figura 3.40 se muestran dos maneras en que un proceso de *inquiry* puede finalizar, sin el envío del mensaje de cancelación.

En el diagrama superior, se inicia el *inquiry* usando el código de acceso general (GIAC), y se busca un máximo de dos dispositivos durante un tiempo de 3*1.28 s. Después que el dispositivo *Bluetooth* transmite un flujo de paquetes ID con el GIAC, dos dispositivos responden. Cada respuesta genera un evento HCI_*Inquiry_Result*, y después del segundo, el evento HCI_*Inquiry_Complete* se envía.

En el diagrama inferior, no se recibe ninguna respuesta, así que un *timer* de 1*1.28 s llega a su fin y un evento HCI_*Inquiry_Complete* se envía para informar que no se recibieron respuestas.

3.5.4.3 Temporización de la *búsqueda*

Es deseable que los dispositivos mantengan una imagen actualizada de su entorno, para ello sería necesario realizar una *búsqueda* continua, sin embargo, esto acabaría rápidamente con las baterías de los dispositivos móviles.

La solución es realizar *búsquedas periódicas* de corta duración. El *host* puede controlar esto pidiéndole al módulo que las realice cada determinado tiempo. Existe otra alternativa en donde el *host* configura al módulo para que automáticamente realice las *búsquedas* periódicamente.

Si las *búsquedas* se realizaran periódicamente con el mismo intervalo de tiempo, podría suceder que nunca se encontrara a los dispositivos adyacentes, debido a que estos podrían realizar la *exploración* en los intervalos donde no se busca. Para evitar este problema las *búsquedas* se inician en intervalos aleatorios, los cuales tienen un valor máximo y mínimo de tiempo, establecido por el *host*. Estos intervalos son múltiplos de 1.28 s.

3.5.5 *Exploración de búsqueda (Inquiry Scan)*

Un dispositivo *Bluetooth* permite a otros dispositivos descubrirlo mediante el proceso de *inquiry scan*. En este modo es posible responder a los dispositivos *buscadores*, y por tanto dejar que obtengan información del dispositivo explorador.

Cuando se inicia el proceso de *búsqueda* se envía un código *de acceso de búsqueda general* (GIAC), o limitado (LIAC), el cual puede ser determinado por el *host*. A su vez en el proceso de *exploración* se puede configurar al receptor para que sólo acepte cierto tipo de código. El comando para la configuración del dispositivo explorador es HCI_*Write_Current*_IAC_LAP. La utilización de códigos de acceso limitados ofrece un mejor desempeño en áreas donde existen diversos dispositivos *Bluetooth*.

3.5.5.1 Temporización de la *exploración*

Un dispositivo que quiere ser descubierto puede llevar a cabo una continua *exploración* hasta que escuche una *búsqueda*, luego responder, y posteriormente entrar en el proceso de *page scan*. Este mecanismo se enfrenta a un par de problemas: el primero es el ineficiente uso de las baterías del equipo y el segundo es que la pérdida de tiempo en el proceso continuo de *exploración* afecta a los demás procesos del sistema.

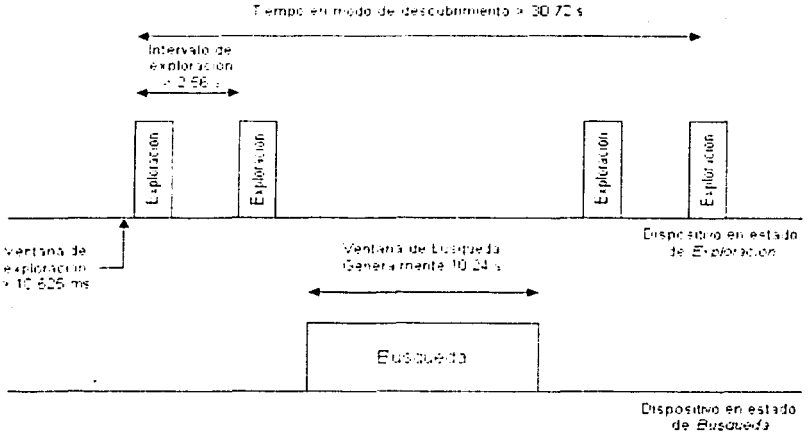


Figura 3.41. Tiempos recomendados para *búsqueda* y *exploración*

La solución es explorar en pequeñas ráfagas. Esto significa que mientas un dispositivo está realizando una *búsqueda*, los dispositivos que desean ser descubiertos llevarán a cabo varias *exploraciones*, de esta manera tendrán una buena oportunidad para ser descubiertos.

El perfil de acceso genérico[14] de *Bluetooth* recomienda que la *búsqueda* dure 10.24 s, mientras que el modo que posibilita el descubrimiento debe durar 3 veces más, es decir 30.72 s. La figura 3.41 muestra con detalle los tiempos recomendados por el estándar.

La *exploración* es habilitada usando el comando HCI_*Write_Scan_Enable*.

El intercambio de mensajes entre el *host* y el módulo *Bluetooth* durante el proceso de *investigación* y *exploración* de *búsqueda* se muestra en la figura 3.42.

[14] El perfil de acceso genérico, o GAP, se detalla en el capítulo llamado *Implementación de protocolos*, en la sección de *Perfiles*.

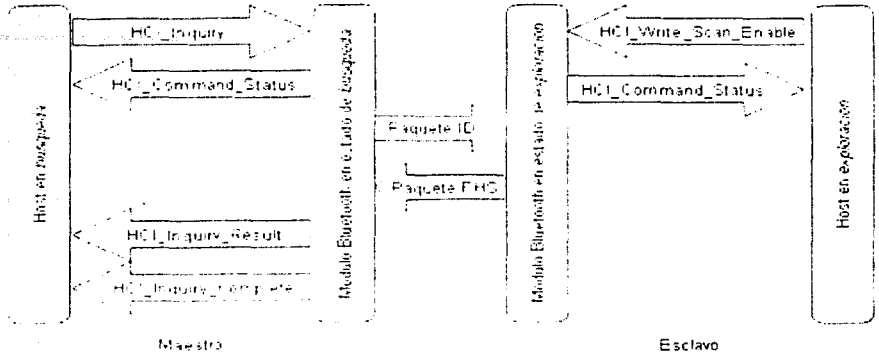


Figura 3.4.2 Intercambio de mensajes entre el *host* y el módulo *Bluetooth* durante los procesos de *investigación* y *exploración*

3.5.6 *Llamado (Paging)*

Un dispositivo *Bluetooth* se conecta con otro mediante el proceso de *paging*, en donde el primero manda su dirección *Bluetooth* en un paquete ID al segundo.

El *host* usa el comando HCI_*Create_Connecton* para iniciar el proceso. Este comando contiene la información necesaria para que el módulo pueda establecer conexión.

Un dispositivo le permite a otro establecer conexión mediante el modo de *page scan*, en este estado el dispositivo escucha paquetes con su propio ID y responde de una manera adecuada para establecer conexión.

El proceso de *page scan*, al igual que el de *inquiry scan*, se lleva a cabo mediante ráfagas cortas, y es habilitado usando el mismo comando: HCI_*Write_Scan_Enable*.

Para que el proceso de *page scan* sea exitoso, debe recibirse un paquete ID con la dirección *Bluetooth* del dispositivo que realiza la *exploración*. En este caso, el módulo explorador puede aceptar automáticamente la conexión, si está configurado así, o puede informar al *host* mediante un evento HCI_*Connection_Request*. El *host* puede aceptar la conexión mediante un comando HCI_*Accept_Connection_Request*, o rechazarla con el comando HCI_*Reject_Connection_Request*. Si el *host* no responde después de un determinado tiempo, la conexion se rechaza.

Si la conexión es aceptada por el *host*, entonces las capas de *bandabase* y LMP terminan de establecer la conexión. Una vez que está hecha, el *host* en ambos lados es notificado por su modulo mediante un evento HCI_*Connection_Complete*, el cual contiene el manejador de conexion (*connection_handle*) que es usado para identificar al enlace. El intercambio de mensajes se observa en la figura 3.4.3. Durante la conexion el *host* puede usar el comando HCI_*Disconnect* para terminar con la conexión.

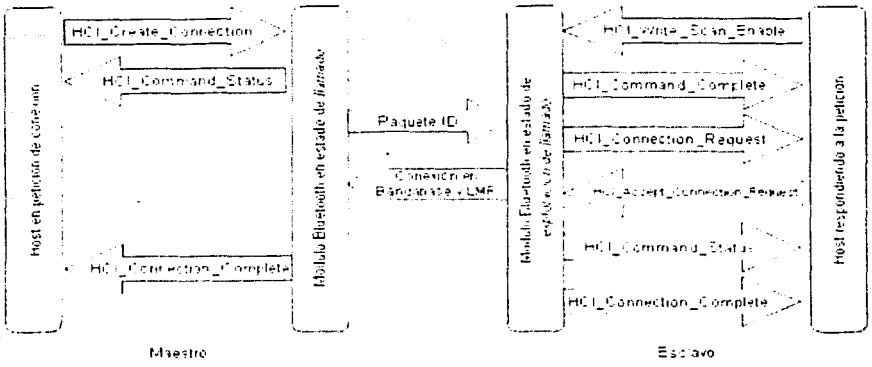


Figura 3.43. Intercambio de mensajes entre el *host* y el módulo *Bluetooth* durante los procesos de *Page* y *Page Scan*

En la tabla 3.4 se resume el comportamiento de los dispositivos de acuerdo a la combinación de las configuraciones de los modos de *exploración*: *page scan* e *inquiry scan*.

Configuración del modo de *Page Scan*	**Configuración del modo de *Inquiry Scan***	**Comportamiento**
Habilitado	Habilitado	Se realizan periodicamente los procesos de *page scan* e *inquiry scan*. Los dispositivos se pueden conectar independientemente de que sepan o no, las direcciones *Bluetooth*.
Habilitado	Deshabilitado	Se realiza periódicamente el proceso de *page scan*. El dispositivo solo puede ser conectado por otros que conozcan su dirección *Bluetooth*.
Deshabilitado	Habilitado	Se realiza periodicamente el proceso de *inquiry scan*. Si el dispositivo responde a un *inquiry* entonces llevara a cabo el proceso de *page scan*. El dispositivo puede conectarse solo con otros dispositivos que no conozcan su dirección *Bluetooth*, y por lo tanto, tengan que hacer primero una *búsqueda*.
Deshabilitado	Deshabilitado	No se realizan los procesos de *page scan* e *inquiry scan*. Los dispositivos no pueden establecer conexiones nuevas. Si actualmente no tienen ninguna conexion activa, pueden usar algun modo de bajo consumo de energia.

Tabla 3.4. Configuraciones de los modos de *exploracion*

3.5.7 Conexión SCO

Una vez activado un enlace ACL, se puede establecer una conexión SCO a través de él, mediante el comando HCI_*Add*_SCO_*Connection*. Los parámetros de este comando especifican el manejador ACL mediante el cual se establecerá la conexión sincrona.

Cuando se envía el comando HCI_*Add*_SCO_*Connection*, se recibe como acuse un evento HCI_*Command_Status*, y cuando se ha establecido la conexión sincrona se recibe un evento HCI_*Connection_Complete* el cual contiene el manejador de conexión SCO.

CAPÍTULO 4
PROTOCOLOS DEL HOST BLUETOOTH

4.1 Protocolo de control y adaptación de enlace lógico (*Logical Link Control and Adaptation Protocol*)

El protocolo de control y adaptación de enlace lógico (L2CAP) toma información de capas superiores de la pila *Bluetooth* y de aplicaciones, y la envía a las capas inferiores de la pila. El L2CAP pasa paquetes ya sea al *Host Controller Interface* (HCI), o directamente al *Administrador de Enlace* (en un sistema *hostless*).

En la figura 4.1 se muestra la posición del L2CAP en la pila de protocolo *Bluetooth* para los casos en los que se use (izquierda) o no (derecha), el *Host Controller Interface*. El L2CAP transfiere datos, pero no audio [5].

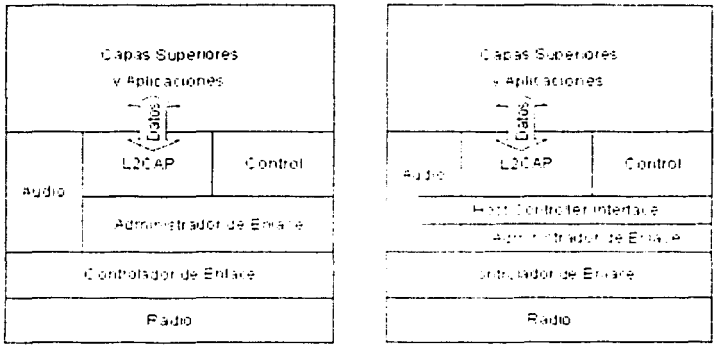


Figura 4.1. Posición del L2CAP en la pila del protocolo *Bluetooth*.

El L2CAP cumple con las siguientes funciones:

- Multiplexaje. Se hace entre distintos protocolos de capas superiores, permitiendo que compartan enlaces de capas inferiores.

- Segmentación y rearmado. Permite la transferencia de paquetes más grandes de lo que las capas inferiores soportan.
- Administración de grupo. Permite la transmisión de una vía a un grupo de otros dispositivos *Bluetooth*.
- Administración de calidad de servicio. Para protocolos de capas superiores.

El L2CAP depende de conexiones ACL para pasar información, de manera confiable, de terminal a terminal. Una función de control, independiente de L2CAP, debe preparar las conexiones ACL cuando éstas sean requeridas por el L2CAP y terminarlas cuando ya no lo sean. El L2CAP también depende de la calidad de servicio de las conexiones para proporcionar la calidad de servicio que se negoció con las capas superiores.

4.1.1 Multiplexaje usando canales

El L2CAP proporciona multiplexaje para permitir que pasen enlaces de capas superiores a través de una simple conexión ACL. Esto permite que distintos tipos de aplicaciones del usuario compartan un enlace ACL con las capas superiores de la pila de protocolo *Bluetooth* (como por ejemplo: el protocolo de descubrimiento de servicios, SDP).

El L2CAP utiliza números de canal para etiquetar paquetes, así cuando éstos se reciben, se les puede encaminar al lugar correcto. Debido a que las entidades L2CAP deben comunicarse entre sí para controlar canales, se reserva un número especial de canal para paquetes de señalización usados para controlar conexiones L2CAP; un segundo número de canal se reserva para recibir paquetes *multicast*. Un rango de números de canal está disponible para etiquetar conexiones L2CAP que podrán ser utilizadas por capas superiores. Esos números de canal se asignan al momento en que las conexiones se establecen [22].

El número de canal se lleva en un identificador de dos bytes, y viene después del campo de longitud en cada paquete L2CAP, como se observa en la figura 4.2.

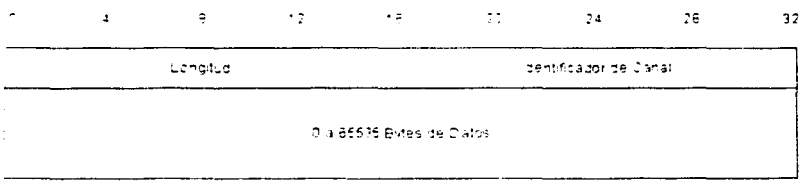


Figura 4.2. Estructura de un paquete L2CAP.

El resto del paquete L2CAP lleva la información proveniente de capas superiores. El tamaño del paquete es variable, y la longitud de los datos está limitada por un campo de dos bytes. Esto permite al L2CAP llevar hasta 65,535 bytes de datos por paquete. La gran capacidad de los paquetes L2CAP permite que se mantengan los paquetes de protocolos de capas superiores.

4.1.2 Señalización L2CAP

El canal de señalización L2CAP está localizado en el canal cuyo identificador (CID) es 0x0001. Dicho canal se usa para enviar información de control entre entidades L2CAP (de punto a punto), para mantener la conectividad, la configuración y la desconexión de conexiones L2CAP. *Bluetooth* tiene reglas para nombrar las señales las cuales siguen una convención comúnmente utilizada en protocolos de comunicaciones [22]. Estas reglas se ilustran en la figura 4.3.

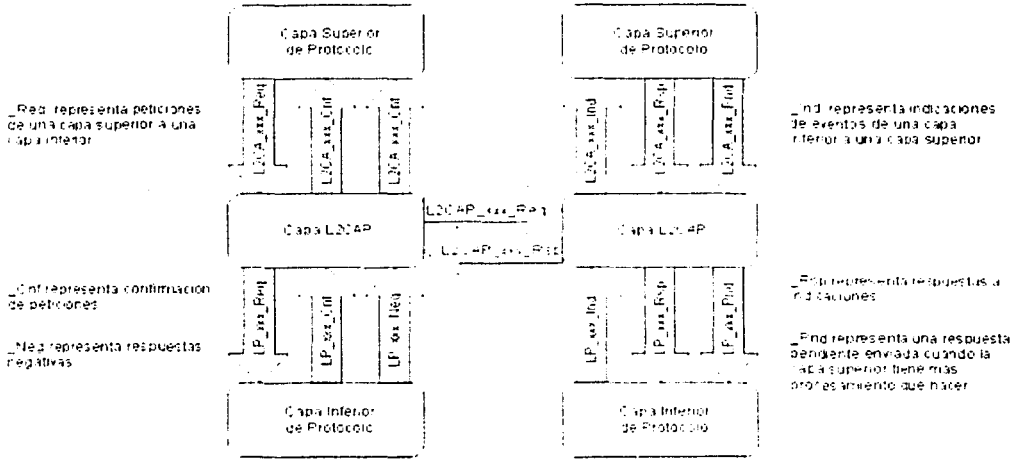


Figura 4.3. Convenciones para nombrar señales L2CAP.

Las reglas para nombrar las señales son las siguientes:

- Señales entre "punto a punto" del mismo nivel de capa, se les antepone un prefijo con las iniciales del protocolo (por ejemplo: L2CAP)
- Señales entre capas superiores e inferiores: se les antepone un prefijo con las iniciales de la capa inferior (por ejemplo: L2CA).
- Peticiones de una capa superior a una capa inferior, se les añade el sufijo "Req".
- Confirmaciones a peticiones enviadas desde una capa inferior a una capa superior, se les añade el sufijo "Cnf".
- Si una respuesta desde una capa inferior es negativa, el sufijo "Neg" debe usarse en lugar del sufijo "Cnf".
- Indicaciones de eventos enviados desde una capa inferior a una capa superior, se les añade un sufijo "Ind"

- Respuestas a indicaciones enviadas desde una capa superior a una capa inferior, se les añade al sufijo "Rsp".
- Si una respuesta a una indicación requiere procesamiento posterior, "Rsp" debe ser reemplazado por "Pnd", que es una abreviación de una respuesta pendiente.

Todas las peticiones desde una capa superior a una capa inferior deben ser confirmadas, pero no todas las indicaciones desde una capa inferior requieren una respuesta.

4.1.3 Estructuras de Señalización L2CAP

Como en otras capas de la pila de protocolo *Bluetooth*, las señales L2CAP usan la estructura común que se muestra en la figura 4.4. El primer byte tiene el *OpCode* identificando los contenidos de la señal, le sigue un campo de identificador. Algunos comandos podrían enviarse en un solo paquete, y las respuestas a los mismos podrían regresar repartidas en más de un paquete, haciendo difícil que puedan coincidir las respuestas con las peticiones originales. Para hacer más sencilla la coincidencia de respuestas y peticiones se usa un identificador nuevo por cada petición, y el identificador se copia de la petición a la respuesta.

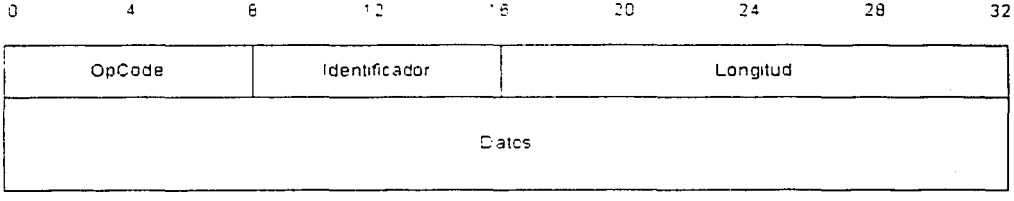


Figura 4.4. Estructura de un comando L2CAP.

El L2CAP inicia un cronómetro cuando se envía un mensaje y si ninguna respuesta regresa, el mensaje se reenvía. En este caso, el identificador original se vuelve a usar, así si el comando fue recibido y hubo una respuesta de que se perdió, el dispositivo que responde sabe que debería enviar otra respuesta, pero no puede ejecutar el comando dos veces. Además de las retransmisiones de los comandos, los identificadores no deben volver a usarse en un periodo de 6 minutos; esto permite que se puedan enviar 255 comandos en 6 minutos.

Después del campo de identificador viene un campo de longitud; éste proporciona la longitud del campo de datos.

Muchos comandos pueden ser enviarse dentro de un paquete de señalización L2CAP, poniendolos uno tras otro como se ve en la figura 4.5. Diferentes implementaciones de L2CAP pueden soportar diferentes longitudes de paquetes L2CAP. El máximo tamaño de la carga útil de paquete soportado se conoce como "Unidad de Transmisión Máxima" (MTU, *Maximum Transmission Unit*). Todas las implementaciones del L2CAP requieren soportar una longitud de

carga útil del paquete de al menos 48 bytes para señalización, así, a menos que el MTU se conozca, una carga útil de paquete de señalización debe limitarse a esta longitud.

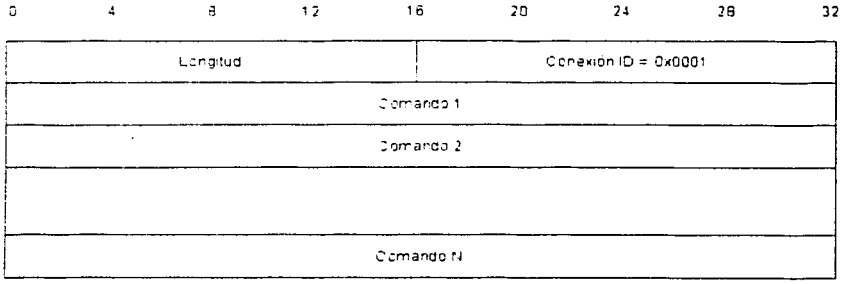


Figura 4.5. Paquete de señalización L2CAP conteniendo múltiples comandos.

Si se recibe un paquete de señalización y éste es más largo que una MTU, se rechaza. Un paquete especial de rechazo esta disponible y tiene un campo de *razón*, el cual dice el motivo por el cual el comando está siendo rechazado. El paquete de rechazo, mostrado en la figura 4.6, contiene un identificador para el comando que está siendo rechazado. Por supuesto que si muchos comandos están siendo rechazados debido a que están en una carga útil más grande que la MTU del dispositivo receptor, el dispositivo receptor puede no ser capaz de leer todos los comandos, por lo que esto representa un gran problema. Para resolverlo, si se recibe una carga útil de paquete de comandos y es más grande que la MTU del dispositivo receptor, todos los comandos en el paquete son rechazados usando un paquete de rechazo. En este caso, el identificador copiado en el paquete de rechazo es el que se tomó del primer comando en el paquete.

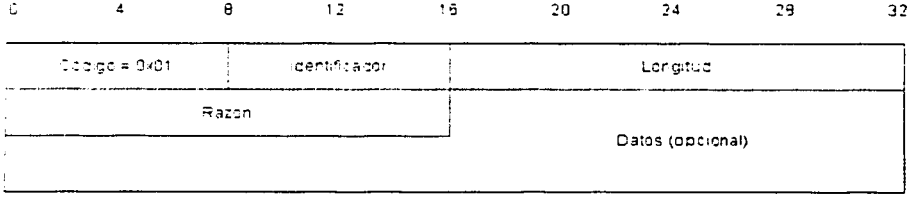


Figura 4.6. Estructura de un paquete L2CAP de rechazo

La versión 1.0 del estándar *Bluetooth* define tres razones para rechazar un paquete, y son las siguientes:

- Comando no entendido. En este caso, no se regresa información.

- La petición tiene un identificador de conexión inválido. El identificador se regresa en el campo de información.
- La carga útil del paquete de comandos fue más larga que la MTU del dispositivo que la rechazó. En este caso, el tamaño correcto de la MTU se regresa en el campo de datos.

El campo de *razón* es de 2 bytes de longitud, por lo que hay mucha libertad para futuras versiones del estándar para definir más razones de rechazo.

4.1.4 Establecimiento de una conexión

El L2CAP usa enlaces ACL para pasar datos confiablemente y sin errores entre las capas inferiores de la pila de protocolos *Bluetooth* [5].

Para establecer un enlace, el protocolo de capa superior enviará una petición a la capa L2CA[1] para conectarse. Si no hay una conexión ACL ya existente, se hace que el L2CAP envíe una petición a un protocolo de capa inferior para conectarse. Como se vio en la figura 4-1, el protocolo bajo la capa L2CA puede ser el HCI o el LM. La figura 4-7 ilustra el caso donde hay una capa HCI y el L2CAP usa el HCI para establecer una conexión ACL. En este diagrama, los mensajes en itálicas son dependientes de la implementación, pero usan nombre sugeridos por la especificación *Bluetooth*. Los mensajes que no están en itálicas tienen formatos completamente definidos por la especificación.

Como se puede ver en la figura 4-7, los pasos involucrados en preparar una conexión ACL son un poco complejos. Este diagrama ilustra cómo el L2CAP depende del resto de la pila de protocolo para proporcionar conexiones confiables de datos.

Los mensajes 8 y 9 muestran al HCI en la parte correspondiente a la aceptación de la conexión, indicando a la capa L2CAP que ha sido *llamada* y se requiere una decisión acerca de si quiere, o no, conectarse. En el protocolo L2CAP, no hay primitivas definidas para capas inferiores para indicar que una petición de conexion ha sido recibida. En su lugar, hay una primitiva LP *ConnectInd* (LM_*ConnectInd*), la cual indica que la conexión se ha puesto en marcha completamente. El estándar pretende que la *bandabase* acepte automáticamente las conexiones, por lo que sólo informara después de que la conexión ya se haya puesto en marcha completamente. En este caso, tiene sentido sólo tener una primitiva LP_*ConnectInd* (LM_*ConnectInd*), sin embargo, es posible que se quiera más control en las conexiones, y tener la habilidad de rechazarlas al principio, en el proceso de puesta en marcha, para permitir a un dispositivo ahorrar tanto ancho de banda como potencia.

[1] L2CA hace referencia a la capa, donde se implementa el protocolo L2CAP

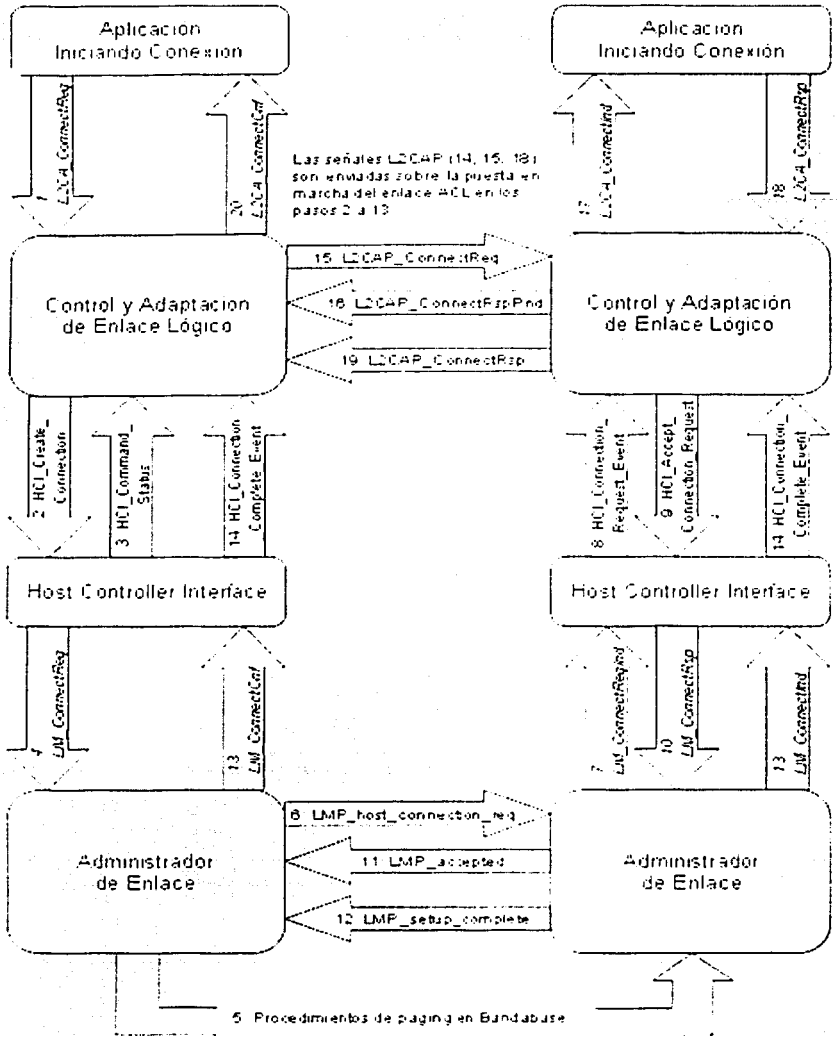


Figura 4.7. Mensajes para preparar una conexión L2CAP a través del HCI.

Si el dispositivo se configuró para aceptar automáticamente conexiones, los mensajes 8 y 9 se omiten; de otro modo, es lógico enviar el mensaje al L2CAP como se muestra.

También hay que hacer notar en el diagrama los dos mensajes 13 y 14, los cuales son enviados a capas superiores de la pila cuando la conexión esta completa. Las entidades L2CAP

en ambos lados deben saber que el enlace ACL está activo antes de que intercambien mensajes L2CAP. En un sistema con un HCI, el mensaje 14 (HCI_*connection_Complete*), indica a ambos lados que la conexión ACL está disponible para intercambiar mensajes L2CAP. El mismo evento HCI es usado tanto en el lado que inicia, como en el lado que acepta, y como es enviado casi simultáneamente en ambos lados, los dos mensajes tienen el mismo número.

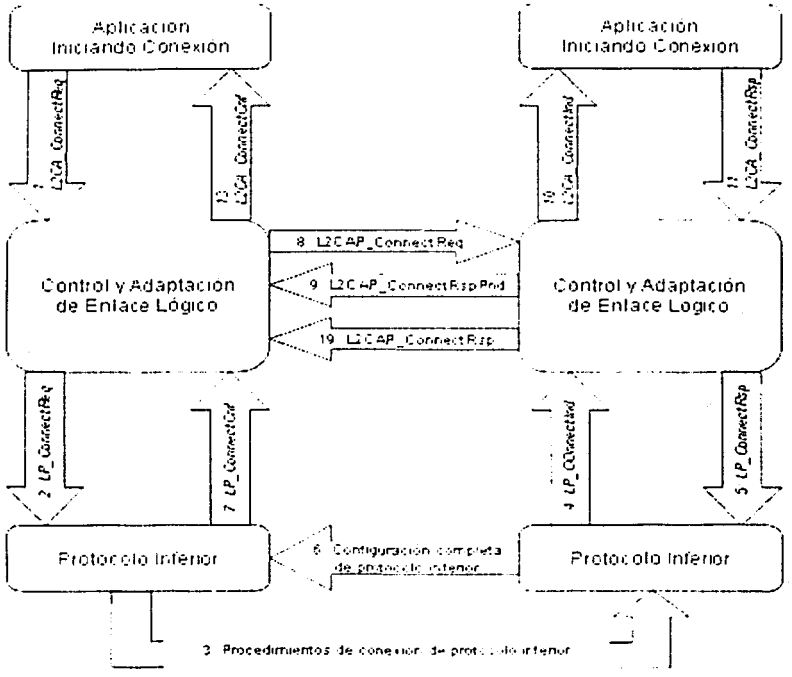


Figura 4.8. Mensajes para preparar una conexión L2CAP a través de un protocolo inferior.

La figura 4.8 muestra un caso general con el L2CAP conectandose vía un protocolo de capa inferior. En un sistema *hostless*, la capa inferior de protocolo podría ser el *Administrador de Enlace* (LM). Esto ilustra el establecimiento exitoso de una conexión a través de capas inferiores, pero si el protocolo inferior fallo para establecer la conexión, los mensajes LP_*ConnectCnfNeg* serían enviados, en lugar del LP_*ConnectCnf*.

En el diagrama de la figura 4.8, la etapa de aceptar la conexión, entre la recepción del paquete de *paging* y la configuración de la conexion, ha sido excluida. Así, de manera diferente

al diagrama anterior, se asume que la *bandabase* ha sido configurada para aceptar conexiones sin informar a capas superiores hasta que sean completadas.

Una vez que una conexión ACL se establece entre capas inferiores, los paquetes L2CAP pueden enviarse a través de ésta. En las figuras 4.3, 4.7, 4.8, los paquetes L2CAP se muestran intercambiados directamente entre entidades "punto a punto L2CAP", pero, son transferidos a través de capas inferiores de la pila en unidades de información de protocolo (PDUs *Protocol Data Units*) inferior. El primer mensaje enviado es un L2CAP_*ConnectReq* (Figura 4.9). Además de los campos *OpCode*, identificador, y longitud, el mensaje lleva los siguientes parámetros:

- Un valor de multiplexor de servicio de protocolo (PSM, *Protocol Service Multiplexer*) especificando el protocolo que está usando la conexión.
- Un identificador de canal de origen (CID, *Channel* ID), el ID de Canal asignado en la conexión por el dispositivo iniciador.

Un mensaje L2CAP_*ConnectionRsp* se envía en respuesta a la petición de conexión. La respuesta contiene los siguientes parámetros:

- El ID del canal destino (CID) que contiene el CID que va a utilizarse para la conexión por el dispositivo que responde. Éste debe guardarse para su posterior uso durante la conexión.
- El ID del canal de origen (CID), el cual es copiado del L2CAP_*ConnectionReq*.
- Un campo de resultado, el cual da el estado de la conexión: configuración exitosa, configuración pendiente, rechazo por razones de seguridad, o rechazo debido a falta de recursos.
- Si la conexión está pendiente, el campo de situación explica qué operación está esperando a ser completada. Ésta puede ser autenticación, autorización, o una operación no especificada.

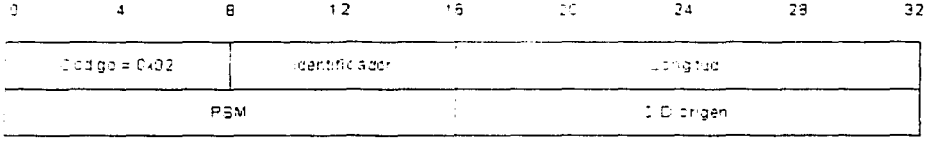


Figura 4.9. Estructura de un paquete de petición de conexión L2CAP.

La estructura del paquete L2CAP_*ConnectionRsp* se muestra en la Figura 4.10.

0	4	8	12	16	20	24	28	32
Código = 0x03		Identificador		Longitud				
CID destino				CID origen				
Resultado				Resultado				

Figura 4.10. Estructura de un paquete de respuesta de conexión L2CAP.

4.1.5 Configurando una conexión

Una vez que una conexión ha sido establecida, debe configurarse. El L2CA que inicia comienza a enviar peticiones de configuración en mensajes L2CAP_*ConfigReq*. Si son rechazados, se regresa un mensaje L2CAP_*ConfigNegRsp* (o un rechazo de comando se envía, si la petición no está hecha correctamente), y el dispositivo que inicia debe tratar una vez más con parámetros diferentes hasta que estos parámetros para la conexión sean aceptados con un mensaje L2CAP_*ConfigRsp*.

Una vez que el dispositivo que inicia ha configurado el canal que va al dispositivo que lo acepta, el dispositivo que acepta puede configurar el canal de regreso usando el mismo conjunto de mensajes, esto se observa en la figura 4.11.

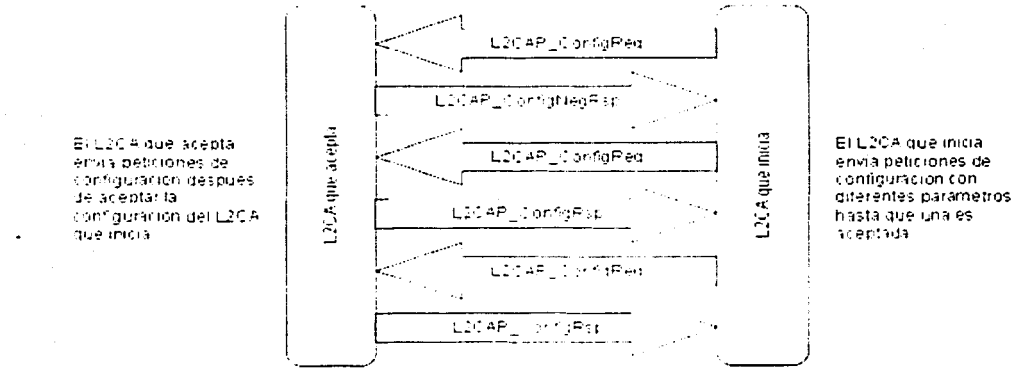


Figura 4.11. Mapa de secuencia de mensajes para una configuración L2CAP.

Si dos dispositivos tienen dificultades para decidir en un conjunto de parámetros mutuamente aceptados, los mensajes L2CAP_*ConfigReq* y L2CAP_*ConfigNegRsp* pueden intercambiarse por un periodo largo de tiempo. El tiempo en que los dispositivos abandonarán finalmente los intentos de configurar el canal, depende de la implementación. El estándar tiene

un tiempo máximo absoluto de 2 minutos; después de ese tiempo, los dispositivos trabajarán ya sea con los parámetros que tenían preestablecidos o terminarán la conexión.

Los parámetros que pueden ser configurados son:

- Unidad de Transmisión Máxima (MTU).
- Tiempo de espera de limpieza.
- Calidad de Servicio.

4.1.5.1 Unidad de Transmisión Máxima

El MTU especificado en un mensaje L2CAP es el máximo tamaño en bytes de carga útil de paquete que un dispositivo está dispuesto a aceptar. Como ya se mencionó, si el MTU que se solicita es más grande que el MTU del dispositivo en el otro extremo, éste rechazará la petición con un mensaje de rechazo que contiene el MTU que puede manejar. Entonces el dispositivo que está haciendo la petición decidirá si aceptar ese tamaño de MTU, o abandonar la conexión.

4.1.5.2 Tiempo de espera de limpieza

El tiempo de espera de limpieza proporciona la cantidad de tiempo en milisegundos que un dispositivo empleará tratando de transmitir un segmento de paquete L2CAP, antes de rendirse. Si un segmento de paquete no pasa antes de que el tiempo de espera se termine, entonces todo el paquete se limpiará (se desecha). Este tiempo determina cuántas veces la *bandabase* puede retransmitir un paquete.

El tiempo de espera se aplica al canal en la misma dirección en la que el L2CAP_*ConfigReq* viaja; es decir, el dispositivo que transmite le dice al dispositivo que recibe qué tiempo de espera de limpieza desea implementar. Si no se especifica un tiempo de espera, el valor de *default* es retransmitir hasta que el enlace se pierda.

4.1.5.3 Calidad de Servicio

La opción de calidad de servicio puede seleccionar un *mejor esfuerzo*, o una calidad de servicio garantizada. Tomando en cuenta que un enlace inalámbrico sujeto a interferencias no puede garantizar calidad de servicio, sólo se puede configurar el canal para no comprometer dicha calidad aceptando más tráfico del que puede manejar. Pueden negociarse valores como el *token rate, token bucket size, peak bandwith, latencia y variación de retraso.*

La calidad de servicio se aplica al canal en la misma dirección en la que viaja el L2CAP *ConfigReq*; es decir el dispositivo que transmite le dice al dispositivo que recibe qué calidad de servicio desea implementar.

4.1.6 Transferencia de datos

Una vez que el canal ha sido creado y configurado, puede usarse para transferir datos. El estándar sugiere que las señales L2CA_*DataWrite* y L2CA_*DataRead* se usen para la transmisión de datos, como se observa en la figura 4.12. Se usa un mensaje L2CAP_*Data* para llevar los datos entre las entidades L2CAP.

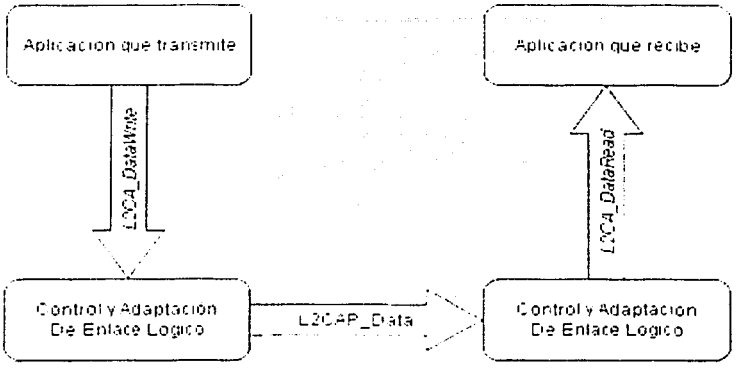


Figura 4.12. Intercambio de datos L2CAP.

Si una capa superior trata de enviar un paquete el cual podría exceder la MTU del receptor, entonces sólo se envían los primeros bytes correspondientes al tamaño de la MTU.

4.1.6.1 Segmentación y reensamblaje

Algunos protocolos de capas superiores usan tamaños de paquetes más grandes de los que *Bluetooth* puede manejar, por lo tanto L2CAP proporciona segmentación de paquetes de capas superiores que bajan a la pila, y reensamblaje de los mismos cuando suben hacia la pila (figura 4.13).

Los paquetes de datos L2CAP tienen una cabecera de 4 bytes la cual se transmite con los datos. Naturalmente, entre más datos puedan ser transmitidos con una cabecera, más eficiente es el sistema, así, los paquetes L2CAP son muy largos, pudiendo llevar hasta 65,535 bytes de datos.

El *Host Controller Interface* (HCI) usa paquetes similares. El tamaño del paquete depende de la implementación debido a que algunos sistemas embebidos tienen menos memoria y no son capaces de guardar paquetes grandes. Sin embargo, el HCI también tiene cabeceras, y si los paquetes son muy pequeños, sería ineficiente; por lo que la especificación dice que todas las implementaciones deben soportar paquetes que lleven arriba de 255 bytes de datos. Debido a que los paquetes HCI son más pequeños que los paquetes L2CAP, los paquetes L2CAP deben

ser segmentados en porciones de datos de varios paquetes HCI. Cuando los paquetes son reensamblados, la capa L2CAP tiene que saber dónde está el comienzo de cada paquete L2CAP, entonces la bandera de límite del paquete HCI (PB, *Paquet Boundary*) se usa para identificar qué paquete HCI tiene el comienzo del paquete L2CAP [5].

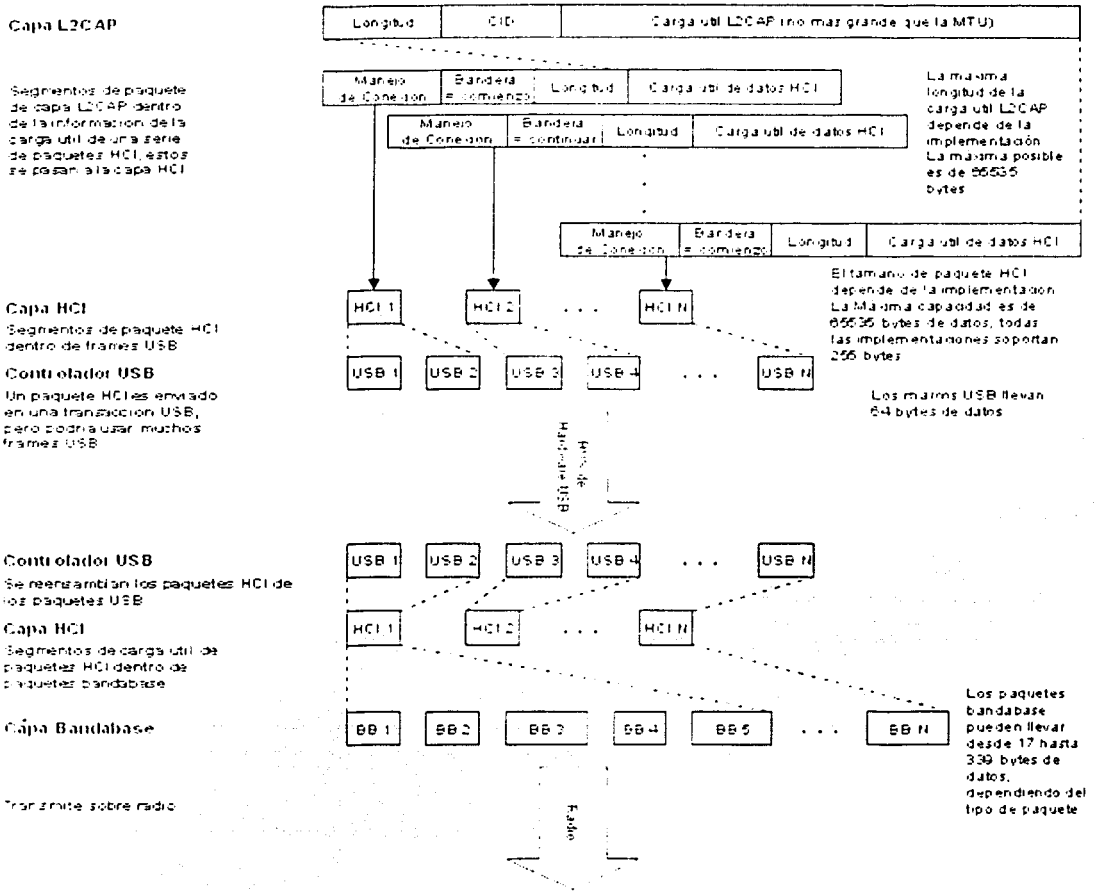


Figura 4.13. Segmentación y transporte de paquetes L2CAP.

El HCI puede usar diferentes capas de transporte. Si se usa el USB (*Universal Serial Bus*), cada paquete HCI se envía en una transacción USB. Una transacción USB puede llevarse a cabo mediante la transferencia de varias tramas USB de 64 bytes. Cualquiera que fuera la capa de transporte utilizada, los mismos paquetes HCI son reensamblados en el lado inferior de la capa HCI.

La *bandabase* usa una variedad de diferentes paquetes que varian en la capacidad de datos. Las capas inferiores de la pila de protocolo *Bluetooth* pueden escoger diferentes tipos de paquetes dependiendo de las circunstancias que se presenten ranura a ranura. Por ejemplo, algunas ranuras deben reservarse para la comunicación con esclavos en los modos de baja potencia como lo son el *Sniff* o el *Park*, así un enlace que normalmente usa paquetes de 5 ranuras, algunas veces sólo tendrá espacio para un paquete de una sola ranura, mientras las otras ranuras sigan reservadas. Debido a que las ranuras disponibles podrian cambiar momento a momento, sólo las capas inferiores de la pila pueden decidir qué tamaño de paquete *bandabase* usar. Por lo tanto, hay otro nivel de segmentación el cual se lleva a cabo en las capas inferiores de la pila, donde los paquetes HCI deben repartirse dentro de las cargas útiles de varios paquetes *bandabase*.

Todavia es importante que el comienzo del paquete L2CAP sea reconocible, por lo que el campo de canal lógico de *bandabase* identifica los paquetes que cargan el comienzo de un nuevo paquete L2CAP.

Cada paquete *bandabase* podría retransmitirse varias veces, pero a menos que el enlace se pierda completamente, la *bandabase* transferirá confiablemente paquetes en orden, y cualquier paquete repetido que se reciba será filtrado. Aún cuando los paquetes L2CAP podrian dividirse y reensamblarse varias veces a través de la pila de protocolo *Bluetooth*, pueden entregarse siempre de forma confiable y reensamblados en el orden correcto, identificando fácilmente el principio de cada paquete L2CAP.

4.1.7 Desconexión y tiempos de espera

Hay dos motivos para que un canal L2CAP se cierre: un protocolo de capa superior, o un servicio puede pedir que sea cerrado, o puede terminar su tiempo de vida.

Una vez que la transferencia de datos ha terminado, el protocolo, o servicio, que usa el canal puede enviar un L2CA_*DisconnectReq* para pedir una desconexión. El formato exacto de este mensaje depende de la implementación, pero debe contener el número ID del canal a desconectar.

Cuando el L2CAP recibe una petición de desconexion de alguna capa superior, envía un paquete L2CAP_*DisconnectReq* a través del enlace *bandabase* al L2CAP homólogo de la otra terminal del canal. Al mismo tiempo, L2CAP deja de enviar y recibir datos en el canal. Cualquier "cola" de datos a transmitir se vacia, y cualquier información que se haya recibido se descarta

El paquete L2CAP_*DisconnectReq* se muestra en la figura 4.14. Debido a que cada conexión L2CAP tiene un número ID de canal diferente en cada terminal de la conexión, la petición de desconexión contiene dos IDs de canal: el ID de la terminal de origen de la conexión puede

copiarse del L2CA_*DisconnectReq*, y el ID del destino fue enviado en la respuesta de conexión, y guardado por el L2CAP cuando el canal se puso en operación.

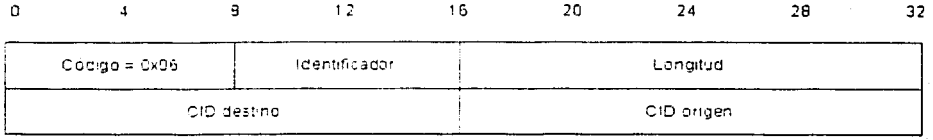


Figura 4.14. Estructura de un paquete de petición de desconexión L2CAP.

El dispositivo que recibe el L2CAP_*DisconnectReq* descarta cualquier dato en la "cola" para transmisión en ese canal, debido a que el dispositivo que envió el L2CAP_*DisconnectReq* está descartando todos los datos que recibe de cualquier forma. El dispositivo que recibe el L2CAP_*DisconnectReq* responde con un L2CAP_*disconnectRsp*, cuya estructura se muestra en la figura 4.15.

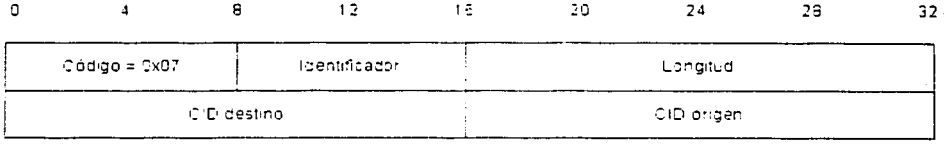


Figura 4.15. Estructura de un paquete de respuesta de desconexión L2CAP.

Cuando el L2CAP que envió el L2CAP_*DisconnectReq* recibe el L2CAP_*DisconnectRsp*, puede informar a la capa de protocolo superior del resultado de la petición de desconexión que hizo. Un resultado con el código 0x0000 se envía de regreso para indicar éxito. Si no se recibió una respuesta, se envía un 0xEEEE, para indicar que la petición ha expirado. El protocolo de capa superior o servicio puede entonces decidir si hace algo para reiniciar la petición. El intercambio de mensajes utilizados para desconectar un canal L2CAP se muestra en la Figura 4.16.

Los canales también pueden ser desconectados como resultado de la expiración de tiempos de espera. Cada vez que el L2CAP envía una señal, comienza un cronómetro conocido como *Response Timeout Expired* (RTX). La longitud del cronómetro RTX varía entre implementaciones, pero está entre un segundo y un minuto. Obviamente, no tiene sentido retransmitir un paquete mientras la *bandabase* está todavía intentando transmitirlo, por lo que el cronómetro RTX debe prepararse para que la retransmisión no comience hasta después de que la *bandabase* ha dejado de transmitir el paquete. Si el tiempo de espera de limpieza es infinito, la *bandabase* se mantiene tratando de retransmitir hasta que el enlace se pierda, entonces el L2CAP no debe retransmitir nada.

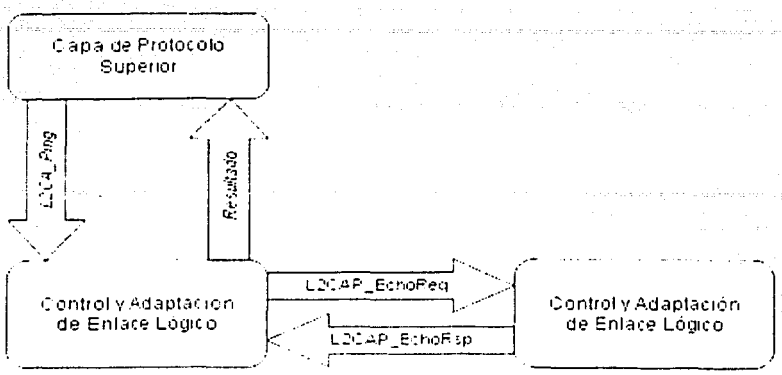


Figura 4.16. Mapa de secuencia de mensajes para desconectar un canal L2CAP.

Cuando el cronómetro expira antes de que una respuesta se reciba, se puede enviar un duplicado del mensaje, o desconectar el canal. Si se envía un duplicado del mensaje, el tiempo de espera RTX para el duplicado es el doble que el previo. Es decisión de los realizadores escoger cuántas veces retransmitir, pero obviamente no tiene sentido seguir tratando por siempre si el mensaje no está pasando. En esos casos, el enlace en *bandabase* probablemente se haya perdido, entonces si no se recibe ninguna respuesta en 60 segundos, el canal deberá ser desconectado sin enviar un L2CAP_*DisconnectReq*.

Hay una excepción para desconectar después de 60 segundos: cuando la terminal regresa una respuesta que indica que hay más procesamiento por hacer, entonces un cronómetro *Extended Response Timeout Expired* (ERTX) se deberá usar en lugar del cronómetro RTX. El cronometro ERTX es como el RTX, excepto por que el rango del ERTX está entre uno y cinco minutos.

Por cada señal enviada, se inicia un cronómetro RTX, y éste puede cambiarse a un cronometro ERTX cuando la terminal opuesta del enlace regresa una respuesta que indica un procesamiento pendiente.

4.1.8 Canales de datos no orientados a conexión

El L2CAP proporciona canales no orientados a conexión para conectar un dispositivo a un grupo de uno o mas dispositivos en una sola dirección.

Los canales de datos no orientados a conexión no pueden ser configurados para calidad de servicio. Esto se debe a que tratar de negociar la misma calidad de servicio con un grupo de dispositivos puede durar por siempre, por lo que es práctico usar un *mejor esfuerzo* (calidad de servicio por *default*). El L2CAP envía datos no orientados a conexión a todos los dispositivos en

un grupo, pero debido a que los canales no orientados a conexión son de tipo *mejor esfuerzo*, la información no está garantizada a llegar.

Los identificadores de canal (de transmisión) se configuran en el momento en que se establecen los canales no orientados a conexión, sin embargo, el mismo identificador de canal L2CAP (0x0002) se usa para todos los canales no orientados a conexión.

Con tráfico L2CAP orientado a conexión, un valor de *Protocol Service Multiplexer* (PSM) se asocia con un identificador de canal particular, pero no puede hacerse con tráfico no orientado a conexión, debido a que todo el tráfico no orientado a conexión es recibido sobre el mismo identificador de canal (0x0002). Para proporcionar una forma de asociar un valor PSM con tráfico no orientado a conexión, el valor PSM se agrega dentro de todos los paquetes L2CAP no orientados a conexión al principio del campo de datos de carga útil, como se muestra en la figura 4.17.

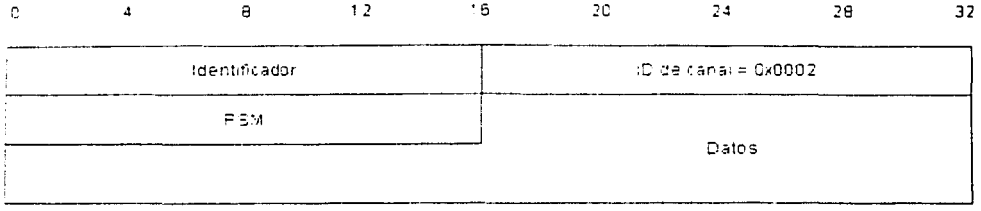


Figura 4.17. Estructura de un paquete L2CAP sin conexión.

El campo PSM toma al menos dos bytes. El bit menos significativo de cada byte en el campo PSM es una bandera de continuidad. Si es cero, hay mas bytes por venir; si es 1, entonces es el último byte del campo PSM.

Algunos valores PSM definidos por el estándar *Bluetooth* son:

- 0x0001 Protocolo de descubrimiento de servicio (SDP).
- 0x0003 Emulación de puerto serial, RFCOMM
- 0x0005 TCS-BIN, *Telephony Control protocol Specification* – binario.
- 0x0007 TCS-BIN-CORDLESS.

Los valores PSM mayores a 0x1001 están disponibles, para asignarse a servicios en función de como las conexiones L2CAP sean establecidas.

4.1.9 Habilitación y deshabilitación de tráfico no orientado a conexión

Es posible que aplicaciones de capas superiores quieran solo recibir tráfico direccionado especificamente para ellas, entonces el L2CAP define mensajes para deshabilitar tráfico no orientado a conexión.

El formato exacto del mensaje para deshabilitar el tráfico no orientado a conexión depende de la implementación, pero el mensaje debe incluir un parámetro *Protocol Service Multiplexer* (PSM). El tráfico destinado por un protocolo, o servicio, puede ser deshabilitado especificando su valor PSM. Por ejemplo, el mapa de secuencia de mensajes a la derecha de la Figura 4.18, muestra la deshabilitación de tráfico no orientado a conexión, destinado para RFCOMM, mediante un mensaje que especifica el PSM reservado para RFCOMM (0x0003).

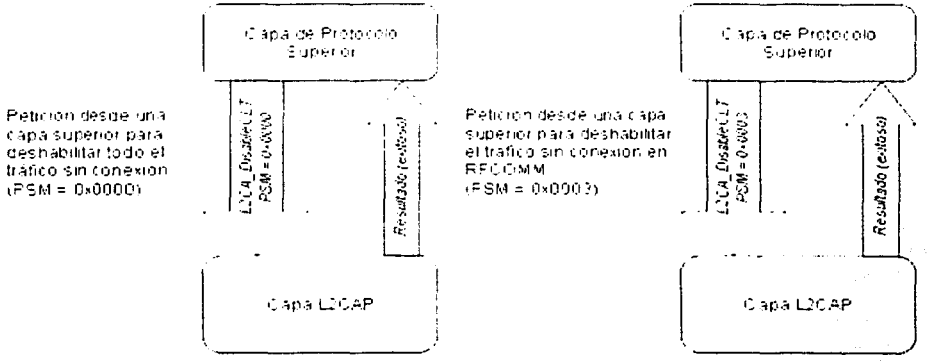


Figura 4.18. Mapa de secuencia de mensajes para deshabilitar tráfico sin conexión.

Para deshabilitar toda la recepción de tráfico no orientado a conexión, el parámetro PSM inválido se establece con el valor 0x0000, esto se muestra en la parte izquierda de la figura 4.18.

Existe un mensaje que trabaja de forma similar al L2CA_*DisableCLT* y habilita la recepción de paquetes sin conexión, este es L2CA_*EnableCLT*.

4.1.10 Manejo de grupos

Para enviar tráfico no orientado a conexión, primero se debe crear un grupo L2CAP. Así como con todos los mensajes entre L2CAP y protocolos de capas superiores, el formato exacto del mensaje es específico de la implementación, pero debe incluir un campo PSM para el protocolo o servicio que transmitirá al grupo. La respuesta lleva el número ID de conexión (CID), el cual será usado por el protocolo de capa superior para transmitir al grupo. Este ID de conexión también se usa para agregar dispositivos al grupo, esto se hace enviando el ID de conexión del grupo y la dirección del dispositivo *Bluetooth* que se va a agregar. Por ejemplo, el mapa de secuencia de mensajes a la izquierda de la figura 4.19 muestra un grupo que esta siendo creado por RFCOMM (PSM = 0x0003); el resultado regresa un ID de 0x0004 para el nuevo grupo. El mapa a la derecha muestra un dispositivo que esta siendo agregado al recién creado

grupo. Los parámetros de la petición L2CA_*GroupAddMember* son el ID de conexión y la dirección del dispositivo *Bluetooth* que se agrega; la respuesta es un 0x0000 que indica un resultado exitoso, o un 0x0001 que indica un resultado fallido.

También está disponible un comando L2CA_*GroupRemoveMember*, que es similar al L2CA_*GroupAddMemberCommand* y maneja los mismos parámetros: un ID para especificar el grupo del cual el dispositivo es removido y la dirección del dispositivo *Bluetooth* a ser removido. El resultado es ya sea un éxito (0x0000), o un fallo (0x0001) si el dispositivo especificado no era miembro del grupo especificado.

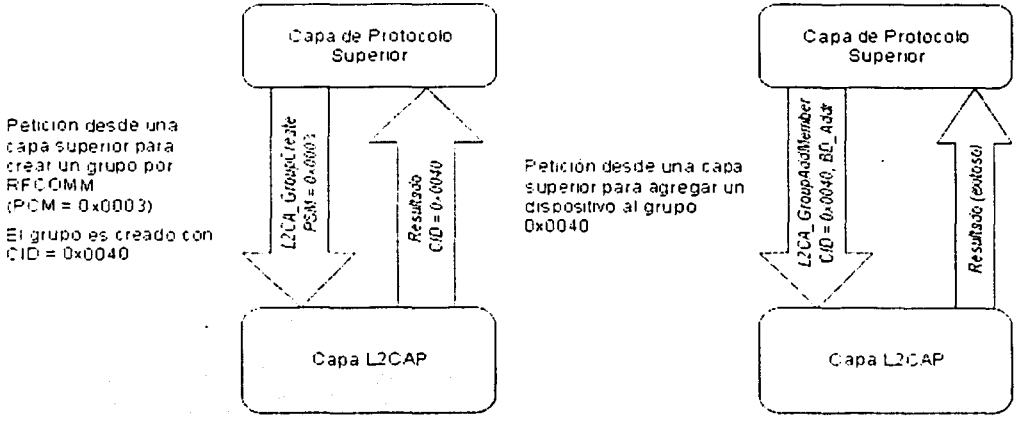


Figura 4.19. Mapa de secuencia de mensajes para crear un grupo no orientado a conexión y agregar un dispositivo.

El L2CA_*GroupClose* debe usarse para cerrar un canal sin conexión. Esta petición toma un solo parámetro: el número ID de la conexión a ser cerrada.

Finalmente, la petición L2CA_*GroupMembership* debe usarse para encontrar la dirección del dispositivo *Bluetooth* de los dispositivos en un grupo. El CID del grupo se pasa como un parámetro del L2CA_*GroupMembership*; el resultado es un éxito (0x0000), o un fallo (0x0001). Si es éxito, entonces una lista de direcciones de dispositivos *Bluetooth* se regresa en el resultado. La lista viene después del código de éxito.

4.1.11 *ECO Y PING*

Peticiones de *Echo* piden al dispositivo L2CAP enviar una respuesta de eco de regreso. Pueden usarse para probar un enlace, o para pasar comandos específicos de la implementación, debido a que incluyen un campo de datos como se muestra en la figura 4.20. Cualquier dispositivo que use el campo de datos de la petición de eco para extender la configuración del comando L2CAP

debe prepararse cuidadosamente para asegurar que haya una implementación similar en el otro extremo, de lo contrario sus comandos podrían tener resultados no deseados.

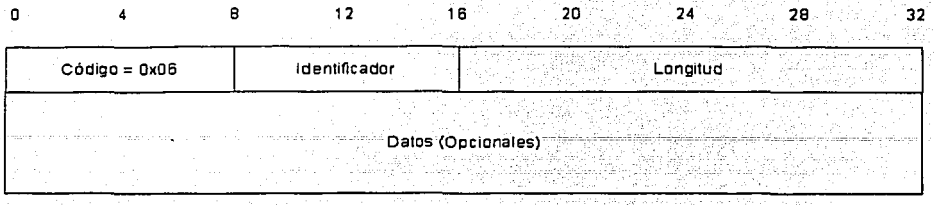


Figura 4.20. Estructura de un paquete L2CAP de petición de eco.

El paquete de respuesta al eco coincide con la petición de eco en todo menos en el código de paquete. El identificador de la petición de eco se copia a la respuesta de eco, como también el campo de longitud y cualquier información.

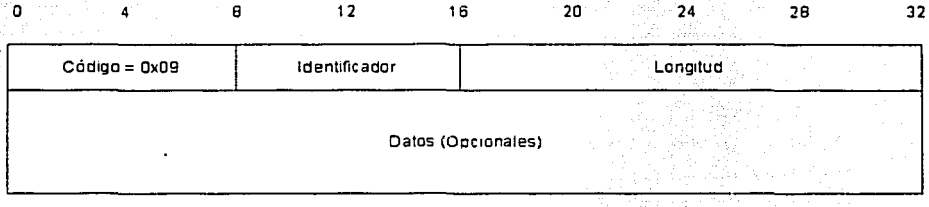


Figura 4.21. Estructura de un paquete L2CAP de respuesta de eco.

Una capa de protocolo superior puede iniciar el intercambio de mensajes de eco mediante la petición L2CA_*Ping*. Esta petición depende de la implementación, pero debe tomar en cuenta los siguientes parámetros:

- BD_ADDR – La dirección del dispositivo *Bluetooth* al cual se enviará el L2CAP_*EchoReq*.
- ECHO_DATA – Un apuntador a la información a ser enviada en el L2CAP_*EchoReq* (opcional).
- Longitud – Dos bytes que indican la longitud del ECHO_DATA (si no hay datos, la longitud es igual a cero).

El resultado comienza con un código de éxito (0x0000), o fallo (0x0001). El *ping* falla cuando una petición de eco enviada por el L2CAP expira antes de que la respuesta sea recibida. Esto se puede observar en la figura 4.22.

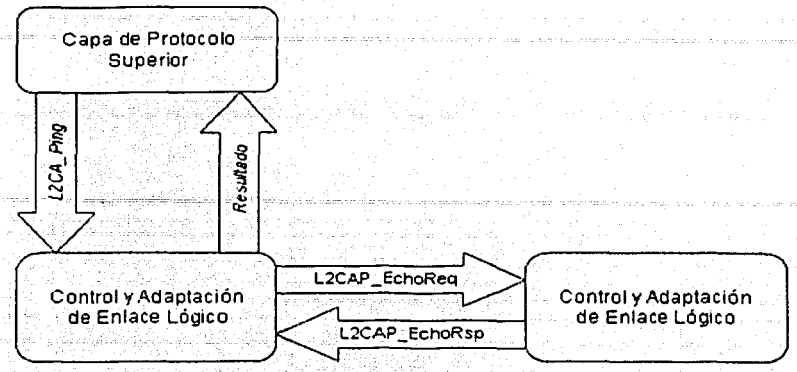


Figura 4.22. Mapa de secuencia de mensajes para un *ping* y un *echo* de L2CAP.

4.1.12 Obtención de información

El L2CAP tiene mensajes L2CAP_*InfoReq* y L2CAP_*Info_Rsp*, que pueden usarse para intercambiar información entre capas homólogas L2CA.

La figura 4.23 muestra el paquete L2CAP_*InfoReq*. El parámetro de tipo de información especifica el tipo de información de la que se hizo petición. La versión 1.0 del estándar sólo especifica un tipo de información que puede ser recuperada: el valor de la MTU sobre el canal no orientado a conexión.

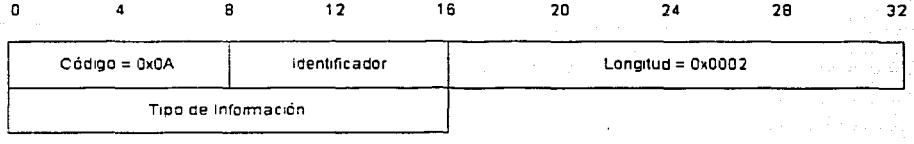


Figura 4.23. Estructura de un paquete L2CAP de petición de información.

La figura 4.24 muestra que el tipo de información L2CAP_*InfoRsp* coincide con el tipo de información enviada en la petición; el resultado es 0x0000 para éxito, o 0x0001 para fallo. Si la petición de información fue exitosa, el campo de datos tiene la información requerida. Para la MTU no orientada a conexión, toma dos bytes.

Si una capa superior de protocolo quiere obtener información del L2CAP, ésta envía una petición L2CA_*GetInfo*. El formato exacto de esta petición depende de la implementación, pero debe incluir la dirección de dispositivo *Bluetooth* al cual la petición va a ser enviada y el campo de tipo de información, para identificar la información que fue requerida.

El resultado de la petición L2CA_*GetInfo* lleva un parámetro de resultado, que puede tomar los siguientes valores:

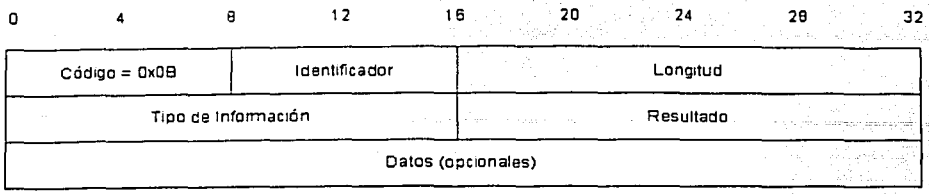


Figura 4.24. Estructura de un paquete L2CAP de respuesta de información.

- 0x0000 – Éxito, una respuesta se recibió; este resultado estará seguido por información de la MTU
- 0x0001 – La petición no fue soportada por el dispositivo local, por lo que no hay datos.
- 0x0002 – La petición no fue soportada por el dispositivo remoto, por lo que no hay datos.
- 0x0003 – La petición expiró antes de que recibiera la respuesta, por lo que no hay datos.

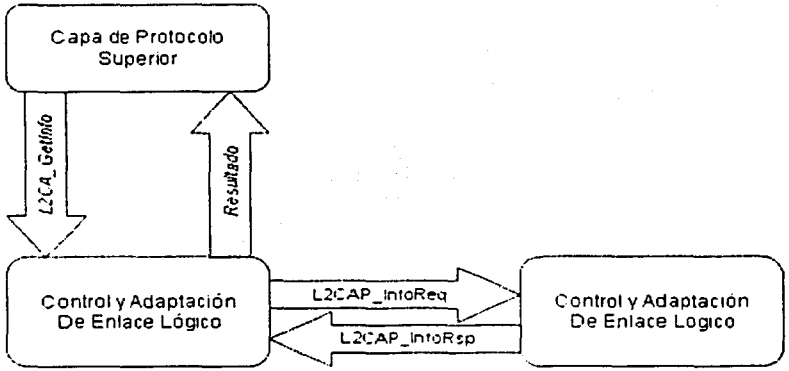


Figura 4.25. Mapa de secuencia de mensajes para una petición de información al L2CAP.

Si la petición fue exitosa, se regresa un apuntador a la información requerida y un campo de tamaño dando la longitud de la información. La figura 4.25 muestra cómo la petición L2CA_*GetInfo* dispara los mensajes L2CAP_*InfoReq* y L2CAP_*InfoRsp* entre puntos homólogos de capas L2CA.

4.1.13 Máquina de estado L2CAP.

Internamente, el L2CAP tiene una máquina de estado manejada por señales L2CA provenientes de capas superiores y señales L2CAP que son llevadas a través de capas inferiores, la figura 4.26 muestra la máquina de estado para preparar una conexión. En cualquier estado, el canal se cerrará, cuando las señales L2CAP no aparecen antes del *Response Timeout* (RTX).

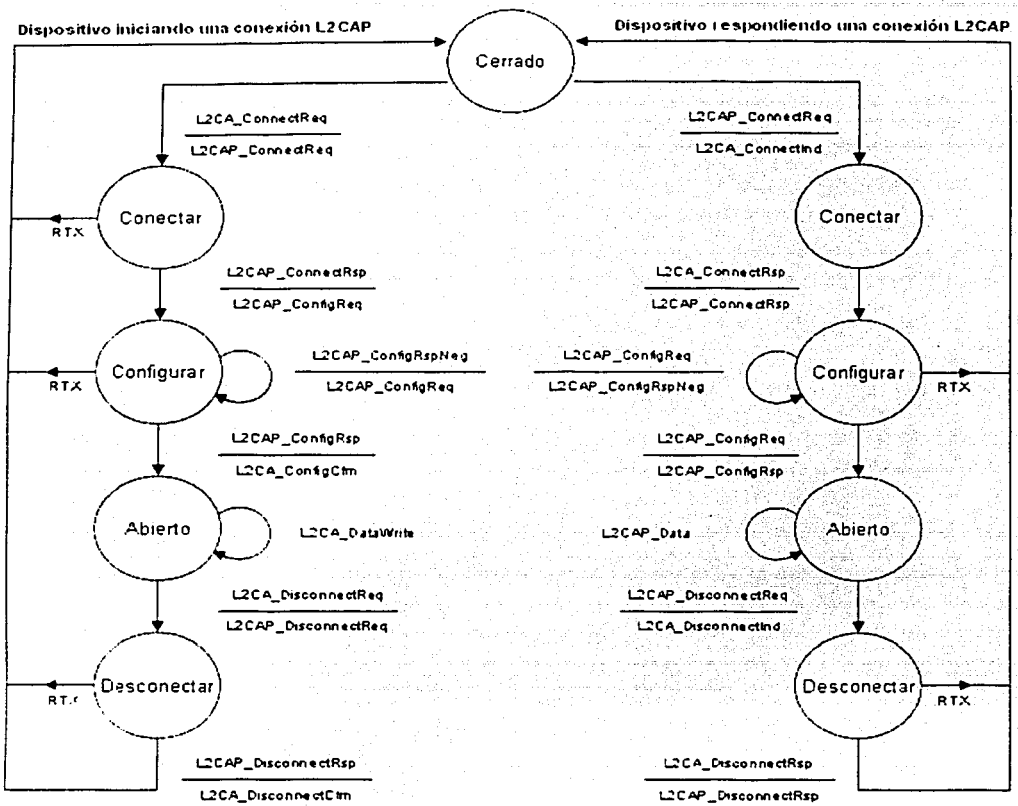


Figura 4.26. Máquina de estado L2CAP.

La figura 4-26 muestra cómo las capas superiores manejan el L2CAP, y cómo los puntos homólogos de L2CAP se comunican. En ese diagrama no se muestran señales que van a capas inferiores, por lo que la transferencia de datos en el estado abierto no enseña las señales usadas para pasar datos a través de capas inferiores. Como se puede observar ese diagrama

sólo muestra datos que pasan del dispositivo que inicia al dispositivo que responde, aunque en realidad, los datos pueden ir en ambas direcciones.

Hay que notar que el dispositivo que inicia la conexión espera señales de su aplicación de capa superior, de esta manera, tal capa lo maneja a través de los estados. En este ejemplo, el dispositivo que acepta la conexión espera señales L2CAP del dispositivo que inicia la conexión, y es manejado a través de los estados por señales L2CAP. Sin embargo, su aplicación también puede manejar la conexión L2CAP, por ejemplo, pidiendo reconfigurar el enlace o pidiendo la desconexión.

Es importante remarcar que todas las aplicaciones deben usar L2CAP para enviar datos. L2CAP también es usado por capas superiores de *Bluetooth* como son RFCOMM y SDP, de esta forma L2CAP se convierte en una parte imprescindible de cualquier sistema *Bluetooth*.

En las secciones siguientes se analizarán los protocolos superiores a L2CAP, primero se estudiará RFCOMM, luego SDP y por último OBEX.

4.2 RFCOMM

Los puertos seriales RS-232 tienen nueve circuitos, los cuales pueden usarse para transferir datos o señales de control. RFCOMM puede emular la configuración y estado de un puerto serial RS-232. RFCOMM provee conexiones concurrentes múltiples, debido a que se apoya en L2CAP para: manejar el multiplexaje de conexiones sencillas, y para proveer enlaces a múltiples dispositivos [22].

RFCOMM se apoya en la *bandabase* de *Bluetooth* para proveer una entrega confiable y en secuencia, de ráfagas de bytes. No tiene la habilidad de corregir errores.

Las tasas de transmisión se verán limitadas en dispositivos donde hay un puerto serial físico (dispositivos tipo 2).

RFCOMM es un protocolo de transporte, simple y confiable, que provee empaquetado, multiplexaje, y tiene las siguientes características:

- Estado del *modem*. Los comandos relacionados con esta función proveen la señalización adecuada en caso de que los datos RFCOMM fueran transferidos a través de cables en lugar de una conexión *Bluetooth*.
- Estado de linea remota. Mediante esta función se envían mensajes al otro extremo para informar sobre algún error. Los errores que pueden ser informados son: sobreescritura de caracter, error de paridad, error en la estructura de la trama.
- Configuración remota del puerto. Los comandos involucrados con esta función tienen la capacidad de cambiar la configuración del extremo remoto del enlace, durante la conexión.

- Negociación de parámetros. Se acuerdan los parámetros de una conexión antes de establecerla. Algunos de los parámetros que se negocian son: el número máximo de retransmisiones y la longitud máxima del paquete, limitada por el MTU del L2CAP.
- Control de flujo basado en crédito. En este tipo de control de flujo se tiene un campo de crédito que representa la cantidad de paquetes que puede enviar un dispositivo. Cuando dicho dispositivo envía un paquete, el crédito negociado inicialmente disminuye en una unidad. Cuando recibe un paquete y el campo de crédito es diferente de cero, éste se agrega al crédito que tenga actualmente.

La especificación RFCOMM de *Bluetooth* describe la emulación de los 9 circuitos de un puerto serial RS-232, y especifica cómo emular un flujo serial de datos. El *software* de RFCOMM maneja datos en forma paralela provenientes de las capas inferiores, y conecta a las capas superiores con las inferiores, de la pila de *Bluetooth*, a través del L2CAP.

4.2.1 Tipos de Dispositivos RFCOMM

RFCOMM soporta dos tipos de dispositivos:

- Tipo 1. Es un dispositivo que posee un puerto serial emulado internamente y es el final de la ruta de comunicaciones.
- Tipo 2. Es un dispositivo intermedio en la ruta de comunicaciones y requiere un puerto serial físico RS-232 para comunicarse con RFCOMM.

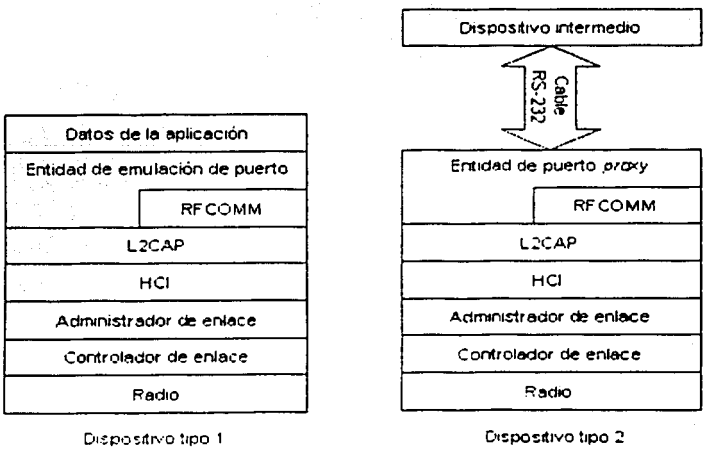


Figura 4.27. Tipos de dispositivos soportados por RFCOMM

En el primer tipo de dispositivo, la entidad de emulación de puerto mapea a una interfaz específica de comunicación del sistema (API) hacia los servicios RFCOMM, con este dispositivo es posible manejar los datos provenientes de una aplicación sobre RFCOMM, a través del puerto emulado internamente, o bien conectarse a aplicaciones específicamente escritas para *Bluetooth*, un ejemplo de este tipo de dispositivos puede ser una impresora, o una computadora.

En el segundo tipo de dispositivo la transferencia de datos entre el puerto físico y RFCOMM se realiza a través de una entidad de puerto *proxy*, un ejemplo de este tipo de dispositivos es un *modem*. En la figura 4.27 se muestran estos dispositivos descritos anteriormente.

4.2.2 Tipos de tramas RFCOMM

RFCOMM utiliza el modo de comunicación con tramas. Las tramas RFCOMM son la carga útil de datos en los paquetes L2CAP. Existen cinco tipos de tramas:

- SABM (*Start Asynchronous Balanced Mode*). Trama que transporta el comando de inicio.
- UA (*Unnumbered Acknowledgement*). Trama de respuesta en estado de conexión.
- DISC (*Disconnect*). Trama que transporta el comando de desconexión.
- DM (*Disconnected Mode*). Trama de respuesta en estado desconectado.
- UIH (*Unnumbered Information with Header check*). Trama que transporta datos.

Las tramas SABM, UA, DM y DISC son tramas de control. RFCOMM usa canales, cada uno de los cuales tiene un identificador de conexión de enlace de datos, DLCI (*Data Link Connection Identifier*). Cuando el DLCI = 0, las tramas UIH se usan para enviar mensajes de control; de otro modo, son usadas para enviar datos.

4.2.3 Conexiones RFCOMM

Para establecer una conexión RFCOMM, primero se debe establecer un enlace L2CAP, debido a que las tramas de RFCOMM se envían en el campo de carga útil de los paquetes de L2CAP.

RFCOMM tiene un valor reservado, el PSM, el cual se usa por el L2CAP para identificar tráfico RFCOMM. Dicho valor está definido por la especificación, como 0x0003. Cualquier trama L2CAP recibida con este valor en el campo PSM será enviada al RFCOMM para su procesamiento.

La primera trama que se envía en un canal RFCOMM es la trama SABM. Si el RFCOMM del dispositivo que responde quiere conectarse, se pone en modo asíncrono balanceado, ABM (*Asynchronous Balanced Mode*) y envía una trama UA; en el caso contrario, rechaza la conexión enviando una trama DM.

El RFCOMM tiene un cronómetro de 60s, el cual se inicia cuando se envía un comando. Si no se ha recibido una confirmación cuando el tiempo expira, la conexión se da por terminada. Para la trama SABM, dicho tiempo puede extenderse debido a procedimientos de seguridad. Si RFCOMM decide desconectarse, debe enviar una trama DISC con el mismo DLCI tal y como en el SABM original, esto en caso de que en el otro lado se haya salido de rango y se regrese considerando que la conexión sigue activa.

Si la conexión se lleva a cabo, el dispositivo que responde envía una trama UA a la trama SABM que se le envió. Después, el dispositivo que inicia envía un comando de negociación de parámetros, PN (*Parameter Negotiation*), y el dispositivo que responde envía una respuesta PN, tal como se observa en la Figura 4.28.

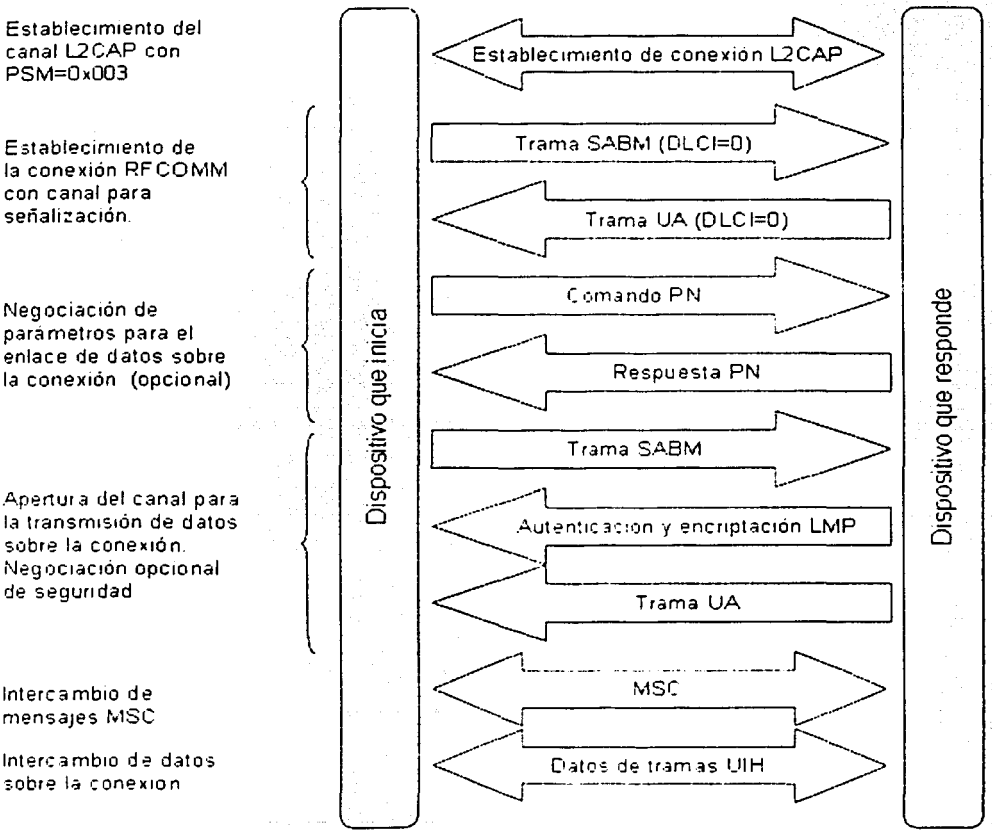


Figura 4.28. Secuencia mensajes para el establecimiento de una conexión RFCOMM.

Una vez establecida la conexión con un DLCI = 0, ésta se encuentra disponible únicamente para señalización. Para la transferencia de datos es necesario el establecimiento de otros canales, sobre la misma conexión RFCOMM, con un DLCI ≠ 0. En el caso de la figura, para el establecimiento del nuevo canal, el dispositivo que inicia envía una trama tipo SABM, y a continuación se lleva a cabo la autenticación mutua y encriptación; al terminar ésta, el dispositivo que responde envía una trama tipo UA. Una vez que el dispositivo que inicia ha recibido la trama tipo UA, se intercambian comandos MSC de *estado del modem*, los cuales indican el estado de las señales de control. Posteriormente, los datos pueden ser transmitidos.

A cada aplicación se le asigna un número de canal de servicio, el rango de números que se pueden asignar comienza en 1 y termina en 30. Con lo cual, teóricamente, se pueden establecer hasta 30 canales, para 30 servicios diferentes a la vez. En la práctica, la mayoría de los dispositivos *Bluetooth* no tienen los recursos suficientes para soportar estos 30 servicios.
El número de canal de servicio es parte del identificador de conexión DLCI (*Data Link Connection Identifier*) que es diferente para cada canal, dicho identificador se encuentra en el campo de dirección de los paquetes RFCOMM.

Para terminar una conexión RFCOMM se envía un comando DISC. Cuando el último enlace de datos ha sido terminado, se debe enviar un DISC, con el DLCI = 0, para apagar el multiplexor. Después de esto, el dispositivo es responsable de desconectar el canal L2CAP.

4.2.2 Estructura de la trama RFCOMM

En la figura 4.29 se muestra la estructura de una trama RFCOMM, se puede observar que se tiene un límite en el número de bytes en un paquete, debido al límite de tamaño de los paquetes L2CAP.

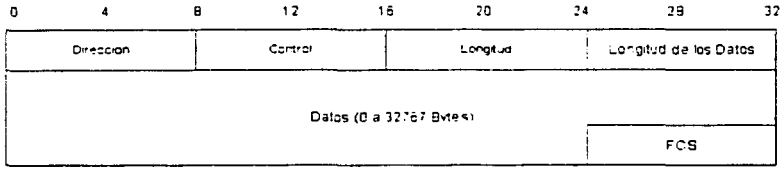


Figura 4.29. Estructura de una trama RFCOMM

Cuando se utiliza el control de flujo basado en crédito, el paquete cambia, teniendo entonces dentro de la trama, además del campo de dirección, el de control, longitud, crédito, datos y FCS, tal como se ve en la Figura 4.30.

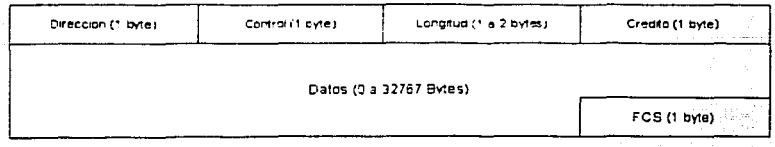


Figura 4.30. Estructura de una trama RFCOMM

La función principal del campo de dirección es identificar a cuál de todos los canales multiplexados pertenece dicha trama.

El campo de control, establece el tipo de trama.

El campo de longitud, nos dice el tamaño de la trama.

El campo de crédito es usado en el control de flujo, y sólo existe cuando se ha negociado esa característica.

Finalmente, el campo de datos sólo existe en las tramas tipo UIH, su tamaño máximo es de 32,767 bytes, sin embargo, puede encontrarse limitado por el MTU de L2CAP. El FCS es el campo con el cual se realiza la revisión de errores dentro de la trama.

RFCOMM esta basado en el estándar GSM 07.10, un protocolo asimétrico utilizado por los teléfonos celulares GSM, para multiplexar varios flujos de datos a través de un cable serial físico. Éste estándar fue ligeramente modificado para adaptarse a los dispositivos *Bluetooth*.

4.2.3 Registro de servicios

Los dispositivos *Bluetooth* que ofrecen servicios soportados por RFCOMM, deben tener un registro en su base de datos de descubrimiento de servicios, la cual les da información de como conectarse sobre RFCOMM.

Los canales de servicio de RFCOMM son dinámicos, es decir, no cambian mientras el servicio esté activo, pero pueden ser reasignados cuando el servicio no esté en uso.

La información mínima necesaria para conectarse a un servicio sobre RFCOMM es un nombre de servicio y un número de canal sobre el cual transmitir datos. Sin embargo, otros servicios pueden necesitar otros parámetros adicionales para la conexión. Mediante una búsqueda en los registros *SDP*, cualquier dispositivo puede encontrar toda la información necesaria para conectarse a un servicio a través de RFCOMM.

La tabla 4.1 muestra un registro de servicio mínimo que debe usarse para proveer la información necesaria para conectar a un servicio a través del RFCOMM. El parámetro ServiceClassIDList proporciona el nombre del servicio; el parámetro ProtocolDescriptorList da los protocolos soportados. Debido a que RFCOMM se apoya en L2CAP, el servicio L2CAP debe estar presente cada vez que RFCOMM esté presente.

Parámetro	Tipo	Valor	ID de Atributo
ServiceRecordHandle	Uint32	Asignado por el *Servidor*	0x0000
ServiceClassIDList			0x0001
ServiceClass()	UUID	Nombre del Servicio	UUID del Servicio
ProtocolDescriptorList			0x0004
Protocol0	UUID	L2CAP	0x0100
Protocol1	UUID	RFCOMM	0x0003
ProtocolSpecificParameter()	Uint8	Número de canal de *servidor*	
ServiceName	String	"Nombre en texto"	0x0000 + offset del idioma

Tabla 4.1 Atributos de Servicio, necesarios para conectarse a un servicio RFCOMM

4.3 Protocolo de descubrimiento de servicios (SDP)

Este protocolo provee los medios para que un dispositivo *Bluetooth* pueda encontrar los servicios que otros dispositivos ofrecen [23].

En la figura 4.31 se muestra la posición que ocupa SDP en la pila de protocolos *Bluetooth*. SDP se apoya sobre L2CAP, así cuando un enlace L2CAP se ha establecido, puede ser usado para encontrar servicios y obtener la información necesaria para conectarse a ellos. L2CAP no maneja las conexiones a los servicios por sí misma, sino que únicamente provee información.

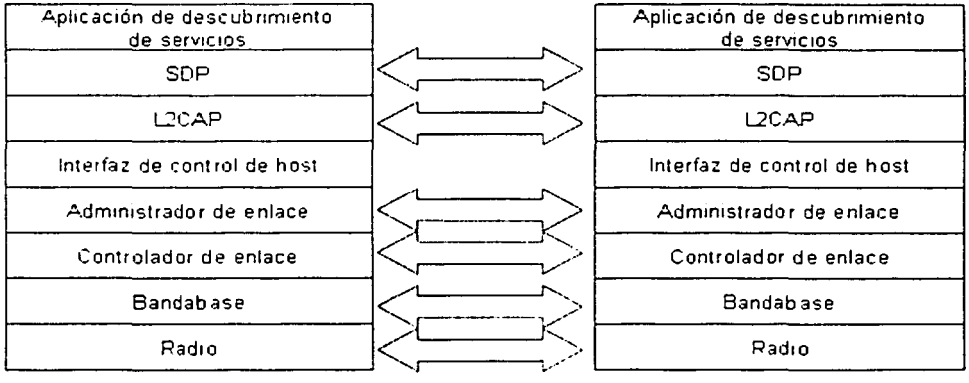


Figura 4.31. Posición de SDP en la pila de protocolos *Bluetooth*

Para encontrar y conectarse a un servicio ofrecido por un *servidor SDP*, un *cliente* debe seguir los siguientes pasos:

- Establecer una conexión L2CAP con el dispositivo remoto usando un identificador de canal especial (PSM=0X0001).
- Buscar una clase específica de dispositivo, o servicio.
- Obtener los *atributos* necesarios para conectarse al dispositivo seleccionado.
- Establecer una conexión diferente a la usada por el SDP, con el fin de utilizar el servicio.

El canal L2CAP usado por SDP puede ser cerrado una vez obtenida la información necesaria, sin embargo, puede dejarse abierto si el *cliente* quiere obtener información de otros servicios contenidos en la base de datos del *servidor*.

La importancia de conocer un identificador PSM radica en la posibilidad de que una vez establecido un enlace ACL, el *cliente* puede conectarse con el *servidor* SDP utilizando dicho número y de esta manera explorar los servicios que ofrece.

Bluetooth no ofrece una interfaz hombre-máquina para el descubrimiento de servicios, sino que únicamente define un protocolo para intercambiar datos entre un *servidor*, que ofrece servicios, y un *cliente* que desea usarlos.

4.3.1 Modelo *cliente/servidor*

Un *servidor* SDP se define como cualquier dispositivo *Bluetooth* que ofrece algún servicio, o servicios, a otros dispositivos *Bluetooth;* de manera complementaria, se define un *cliente* como un dispositivo que desea encontrar servicios en el área. Los dispositivos pueden ser simultáneamente *clientes* y *servidores*.

La información acerca de los servicios se encuentra en bases de datos SDP. Cada *servidor* SDP mantiene su propia base de datos, debido a que no hay una base de datos centralizada. Para hacer posible que los *clientes* utilicen los servicios ofrecidos por los *servidores*, ambos dispositivos intercambian información referente a dichos servicios, usando *registros de servicio.*

4.3.2 La base de datos SDP

La base de datos SDP es un conjunto de registros que describe todos los servicios que un dispositivo *Bluetooth* puede ofrecerle a otro. SDP provee los medios para realizar la exploración en dichos registros [23].

4.3.2.1 Atributos de Servicio

Un servicio SDP está descrito mediante *atributos*, los cuales brindan la información que el *cliente* necesita para usar el servicio. Cada atributo tiene un identificador de 16 bits y un valor. Los valores de los atributos pueden ser cadenas, valores booleanos, o enteros.

Los atributos se usan para pasar información sobre servicios y sobre la jerarquía de servicios disponibles [5]. Cada atributo en un registro describe un aspecto diferente del servicio. La versión 1.0 del estándar *Bluetooth* define veintiocho tipos de atributos que se enlistan a continuación:

- *ServiceRecordHandle*: Es un número de 32 bits que identifica un registro de servicio dentro de un *servidor*.
- *ServiceClassIdList*: El tipo de servicios cubiertos por este registro de servicio.
- *ServiceRecordState*: Un número de 32 bits que cambia si algún atributo del registro cambia. Permite al *cliente* obtener información y verificar si esta actualizada o no.
- *Service*ID: Un identificador único para esta instancia del servicio. El mismo servicio puede tener diferentes valores *Service*ID en diferentes *servidores*.
- *ProtocolDescriptorList*: Los protocolos necesarios para usar el servicio.
- *BrowseGroupList*: Una lista de grupos que se usa cuando se hace la búsqueda de servicios.
- *LanguageBasedAttributeList*: Una lista de idiomas que el registro de servicio soporta. Cada idioma listado tiene su identificador, un identificador de cómo los caracteres son codificados, y un ID de atributo.
- *ServiceInfoTimeToLive*: Un entero de 32 bits que proporciona un estimado del número de segundos hasta que el registro de servicio haga el siguiente cambio.
- *ServiceAvailability*: Un entero de 8 bits que indica el máximo número de *clientes* al cual el servicio está sirviendo.
- *BluetoothProfileDescriptorList*: Una lista de perfiles *Bluetooth* soportados por un servicio.
- *Documentation*URL: Un URL para documentación en el servicio.
- *ClientExecutable*URL: Un URL para un ejecutable que se necesita para usar el servicio.
- *Icon*URL: Un URL para el icono que la interfaz de usuario del *cliente* utilizará para representar el servicio.
- *ServiceName*: Una cadena con el nombre del servicio para el uso de la interfaz de usuario del *cliente*.
- *ServiceDescription*: Una cadena que describe el servicio, para el uso de la interfaz de usuario del *cliente*.
- *ProviderName*: Una cadena con el nombre del proveedor del servicio.
- *VersionNumberList*: Una lista de versiones soportadas.

- *ServiceDatabaseSet*: Un entero de 32 bits que cambia cuando un registro de servicio en el *servidor* cambia.
- *GroupId*: Identificador de un grupo de servicios, usado cuando se buscan servicios.
- *RemoteAudioVolumeControl*: Esto si el control de volumen remoto es soportado (para auriculares).
- Red Externa: Usada por el protocolo telefónico inalámbrico para identificar el tipo de red telefónica a la que el teléfono está conectado. El valor puede ser: PSTN = 1, ISDN = 2, GSM = 3, CDMA = 4, celular analógico = 5, conmutación de paquetes = 6, otro = 7.
- Versión del Servicio: Número de versión del servicio.
- Lista de almacenes de datos soportados: Usada por el perfil de sincronización.
- Lista de formatos soportados: Usada por el perfil de *object push*, una lista de formatos de objetos que pueden ser puestos. El valor del formato en Uint8 es: agenda = 1, calendario = 3, notas = 5, mensajes = 6.
- Soporte de fax clase 1: Usado por el perfil de Fax, un dato booleano que describe si se soporta a no el estándar de fax clase 1.
- Soporte de fax clase 2.0: Usado por el perfil de Fax, un dato booleano que describe si se soporta a no el estándar de fax clase 2.0.
- Soporte de fax clase 2: Usado por el perfil de Fax, un dato booleano que describe si se soporta a no el estándar de fax clase 2.
- Soporte de *Audio Feedback*: Usado por el perfil de fax, es un dato booleano que describe si se provee el *audio feedback* en un canal SCO durante el establecimiento de una llamada.

4.3.2.2 Elementos de datos

Los atributos tienen valores, y esos valores pueden tener varios tipos y tamaños. Entonces, un dispositivo que está recibiendo un atributo, sabe de qué tipo y tamaño es. Los atributos se envían en elementos de datos que comienzan con descriptores de elementos de datos que describen su tipo y tamaño.

El primer byte del elemento de datos contiene los descriptores del tipo de elemento de datos; los primeros cinco bits son el descriptor de tipo, y lo otros tres son el descriptor de tamaño.

El descriptor de tipo da el tipo de atributo en el elemento de datos. Hay nueve tipos diferentes:

0. Tipo nulo.
1. Entero no signado.
2. Entero signado con complemento a dos.

3. Identificador universalmente único, UUID (*Universally Unique Identifier*).
4. Cadena de texto.
5. Dato booleano.
6. Secuencia de elementos de datos, los cuales en conjunto forman la información.
7. Secuencia de elementos de datos, uno de los cuales debe ser escogido y conocido como una alternativa de secuencia de datos.
8. *Uniform Resource Locator* (URL)

El descriptor de tamaño proporciona el tamaño del atributo en el elemento de datos. Comienza con un índice de tamaño. Cuando el índice de tamaño tiene los valores: 0, 1, 2, 3 o 4, da la longitud del atributo como sigue:

0. Un byte, o cero bytes si es un elemento de datos nulo.
1. Dos bytes.
2. Cuatro bytes.
3. Ocho bytes.
4. Dieciséis bytes.

Si el índice de tamaño es 5, 6 o 7, le corresponde el tamaño:

5. El tamaño de datos está en el siguiente byte.
6. El tamaño de datos está en los siguientes dos bytes.
7. El tamaño de datos está en los siguientes cuatro bytes.

Podria ser posible, olvidar el tener que estar enviando el tamaño en un campo de 32 bytes, pero muchos atributos se acomodarían perfectamente en 1, 2, 4 u 8 bytes. Para estos atributos, el mandar el índice de tamaño y no un campo separado de tamaño, ahorra una cantidad considerable de ancho de banda. Para ilustrar esto, en la Figura 4.32, se tiene un ejemplo de elemento de datos que contiene un entero de 16 bits. Los primeros cinco bits tienen el tipo, esto es, 1, que indica que el atributo es un entero no signado; el índice de tamaño es 1, que indica que el atributo es de dos bytes de tamaño, y por último: los atributos de datos: el primero es el byte menos significativo y el segundo es el byte más significativo.

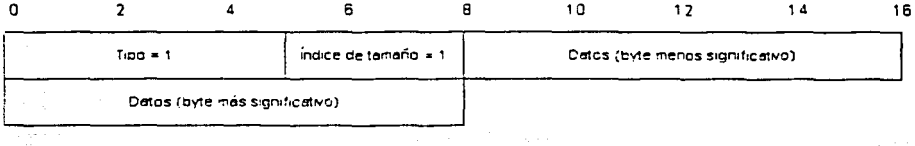


Figura 4.32. Elemento de datos que contiene un entero de 16 bits.

4.3.2.3 Registros de Servicios

Un *registro de servicio* tiene toda la información que describe a dicho servicio y está constituido por una serie de *atributos* que contienen valores. La *clase de servicio* define el significado de los *atributos*, de tal manera que un atributo puede significar una cosa diferente en un *registro de servicio* diferente. Los *registros de servicio* contienen información tal como: la lista de protocolos necesarios para usar el servicio, la lista de *perfiles* que el servicio soporta y el nombre del servicio, entre otros. Sumado a la lista de protocolos necesarios, se tiene la información necesaria para el uso de dichos protocolos. Por ejemplo, en el caso de RFCOMM, se tiene el número del canal del *servidor* RFCOMM.

Para encontrar fácilmente los servicios buscados, se ordenan en una estructura jerárquica tipo árbol. Los *clientes* empiezan por examinar la raiz del árbol, para después continuar con las hojas, donde se encuentran descritos los servicios individuales. Es decisión del proveedor de servicios elegir qué servicios hace disponibles y la estructura del árbol. La figura 4.33 muestra un ejemplo de esta estructura jerárquica de búsqueda para una computadora portátil.

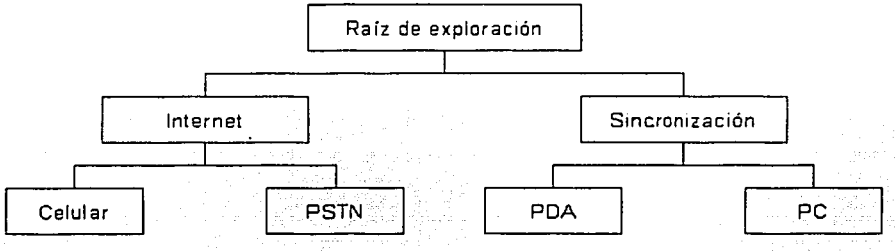


Figura 4.33. Posible estructura jerárquica de búsqueda para una LAPTOP

Cada servicio está asociado a un identificador universal único (UUID, *Universally Unique Identifier*), estos identificadores son transmitidos al *servidor* para investigar si éste soporta el servicio identificado con dicho UUID. Los servicios definidos por los perfiles *Bluetooth* tienen UUID asignados por el estándar, sin embargo, los proveedores de servicios pueden definir sus propios servicios y asignar sus propios UUIDs a esos servicios. Los UIDDs son generados mediante un método que garantiza que no se dupliquen, de esta manera no es necesario que un organismo central los asigne.

4.3.3 Mensajes SDP

Para buscar clases de servicios, o para obtener información de un servicio especifico, los *clientes y servidores* SDP intercambian mensajes. Estos mensajes van en unidades de datos de

protocolo SDP, conocidas como PDU (*Protocol Data Units*). Hay sólo siete tipos de PDU's SDP definidas, y cada una tiene su ID para identificarla:

- 0x01 = SDP_*ErrorResponse*.
- 0x02 = SDP_*ServiceSearchRequest*.
- 0x03 = SDP_*ServiceSearchResponse*.
- 0x04 = SDP_*ServiceAttributeRequest*.
- 0x05 = SDP_*ServiceAttributeResponse*.
- 0x06 = SDP_*ServiceSearchAttributeRequest*.
- 0x07 = SDP_*ServiceSearchAttributeResponse*.

4.3.3.1 Unidades de datos de protocolo SDP

SDP usa unidades de datos de protocolo (PDU's) con la estructura que se muestra en la figura 4.34. El primer byte es un ID, que identifica el mensaje en la PDU; los siguientes dos bytes corresponden al ID de transacción. Cuando un *cliente* envía una petición SDP dando el ID de transacción, el *servidor* copia este ID en la respuesta, por lo que si el *cliente* manda varias peticiones, puede discriminar qué respuesta va con cada petición.

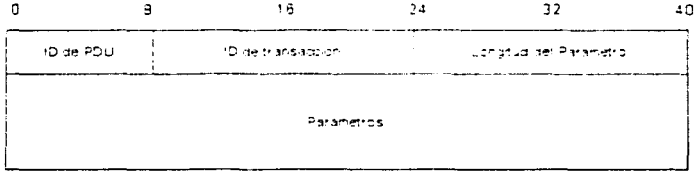


Figura 4.34. Estructura de una PDU SDP.

Con excepción del SDP_*ErrorResponse*, todos los PDU's SDP tienen un *ContinuationState* como último parámetro. Esto permite al mensaje ser repartido en más de una PDU.

4.3.4 Perfil de descubrimiento de Servicios

El perfil de descubrimiento de servicios (*Service Discovery Profile*) describe cómo las aplicaciones que se llevan a cabo en un *cliente* SDP, utilizan SDP, y otras características de la pila de protocolos *Bluetooth*, para descubrir servicios que los dispositivos *Bluetooth* brindan dentro del rango de operación.

El perfil de descubrimiento de servicios brinda:

- Una serie de primitivas de servicio que una aplicación puede usar para manejar el descubrimiento de servicios.

- Secuencias que muestran cómo una aplicación debe hacer la búsqueda de dispositivos, conectarse a ellos, y usar el descubrimiento de servicios en respuesta a una entrada del usuario.
- Mapas de secuencia de mensajes que muestran los estados por los que se pasa para preparar y usar una conexión SDP.
- Una especificación de que los dispositivos que soportan el perfil SDP deben soportar el apareamiento y la autenticación.
- Listas de caracteristicas que se requieren en el SDP, L2CAP, LMP y el LC.

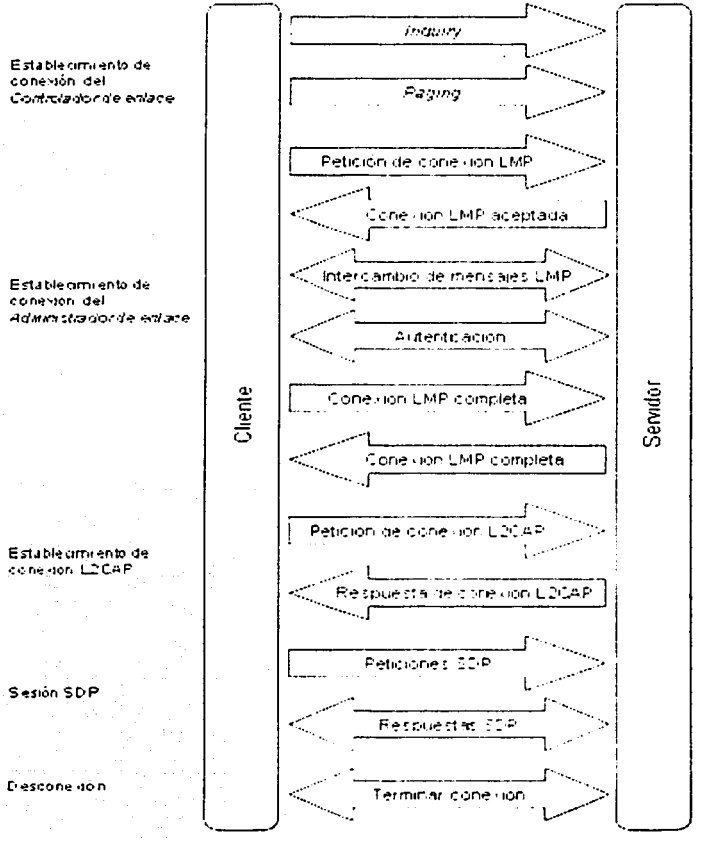


Figura 4.35. Etapas en el establecimiento de una sesion SDP.

La figura 4.35 muestra cómo las diversas capas de la pila deben conectarse para preparar una conexión SDP. El perfil de aplicación SDP da ejemplos de cuándo esas conexiones deben llevarse a cabo; por ejemplo, un *cliente* podría ya estar conectado cuando comienza una sesión SDP, o podría sólo conectarse en respuesta a una petición de información del usuario.

Las primitivas que se dan en el perfil de descubrimiento de servicios incluyen *ServiceBrowse* y *ServiceSearch*, que usan las capacidades del SDP para explorar y buscar. También hay una primitiva llamada *enumerate*RemDec, que causa un *inquiry* y hace que la aplicación diga si hay otros dispositivos *Bluetooth* en el entorno. Por último, hay una primitiva *terminate*, que provoca que un enlace se desactive una vez que la aplicación SDP ha terminado de utilizarlo.

CAPÍTULO 5
INTERACCIÓN ENTRE LAS CAPAS DE LA PILA DE PROTOCOLOS BLUETOOTH

En este capítulo se describirá la forma en que interactúan las capas de la pila de protocolos para realizar funciones enfocadas al nivel de la aplicación. Los puntos a tratar son los siguientes: *perfiles*, *seguridad* y *calidad de servicio* en las conexiones.

5.1 Perfiles

Un *perfil* provee una descripción clara de cómo debe usarse la especificación de un sistema para implementar una determinada función. La finalidad de un perfil es garantizar la interoperabilidad entre sistemas de un mismo estándar, indicando los procedimientos básicos para ello.

La ISO define la noción de *perfil* de acuerdo a las siguientes características [23]:

- Se reducen las opciones de implementación de tal manera que las aplicaciones compartan las mismas características.
- Se definen parámetros para que las aplicaciones operen de forma similar.
- Se definen mecanismos comunes para la convivencia entre diversos estándares.
- Se definen las bases para las interfaces con el usuario final.

Los perfiles en *Bluetooth* cumplen las mismas funciones, aseguran la interoperabilidad al proveer un conjunto de procedimientos bien definidos para capas superiores y métodos uniformes para el uso de las capas inferiores de *Bluetooth*. Al seguir los perfiles, se garantiza que los dispositivos *Bluetooth* sean capaces de interactuar a pesar de que el modelo, o fabricante, sean diferentes.

Para el diseñador, los perfiles permiten reutilizar las características básicas de otras soluciones en el desarrollo de nuevas aplicaciones, o en la optimización de aplicaciones anteriores. Con lo anterior, se acelera el desarrollo de la tecnología.

Los perfiles están organizados en una jerarquía en donde unos dependen de otros, en la figura 5.1 se presentan los grupos de perfiles *Bluetooth*.

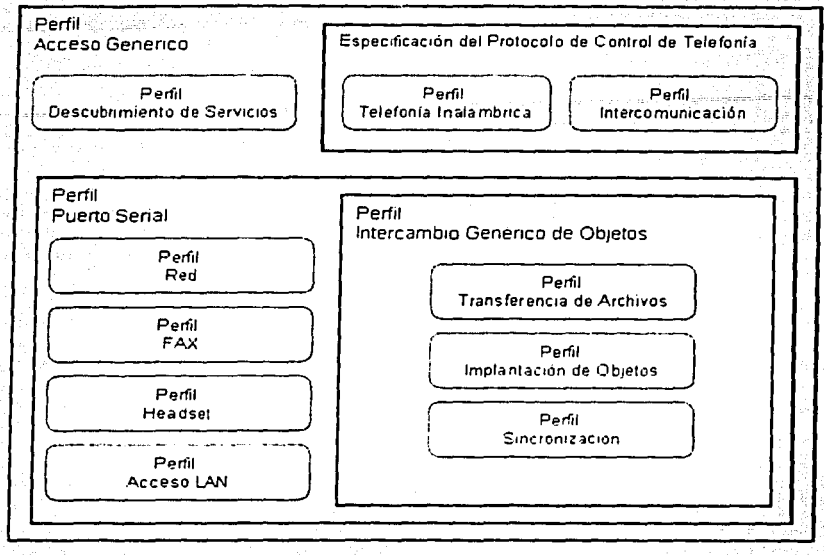


Figura 5.1. Grupos de Perfiles *Bluetooth*

Los perfiles *Bluetooth* son:

- Acceso Genérico. Define las reglas básicas para el uso de la pila de protocolos y es la base para todos los demás perfiles.
- Puerto Serial. Define cómo utilizar las características de RFCOMM en los dispositivos *Bluetooth*.
- Red. Define la conexión con un *modem*.
- FAX. Define cómo transferir un Facsímil sobre *Bluetooth*.
- *Headset*. Define la comunicación con un *headset*, controlado por una *gateway* de audio, como un teléfono, celular o fijo.
- LAN. Define un enlace con una LAN a través de *Bluetooth*.
- Intercambio Genérico de Objetos. Un conjunto de reglas para el uso de OBEX, las cuales soportan los perfiles de: Sincronización, Implantación de Objetos y Trasferencia de Archivos.
- Sincronización. Define la sincronización de objetos entre dispositivos *Bluetooth*.

- Implantación de Objetos. Define el intercambio de objetos entre un dispositivo *Bluetooth servidor* y un *cliente*.
- Transferencia de Archivos. Define la transferencia de archivos entre dispositivos *Bluetooth*.
- Telefonía Inalámbrica. Define la transferencia de llamadas telefónicas mediante dispositivos *Bluetooth*.
- Intercomunicación. Define una conexión orientada a voz entre dispositivos *Bluetooth*.

A continuación se profundizará en los perfiles relacionados con la aplicación propuesta en esta tesis [22][23][5].

5.1.1 Perfil de Acceso Genérico

El perfil de acceso genérico (*Generic Access Profile*, GAP) es el perfil básico dentro de *Bluetooth*, todos los demás se construyen sobre él, su propósito es asegurar que todos los dispositivos puedan establecer exitosamente un enlace en *bandabase*. En este perfil se definen:

- Requerimientos que deben ser implementados en todos los dispositivos.
- Procedimientos generales para el descubrimiento de dispositivos *Bluetooth*.
- Procedimientos generales del administrador de conexión para el establecimiento de enlace.
- Procedimientos relacionados con el uso de diferentes niveles de seguridad.
- Formatos de los parámetros accesibles para el usuario.

Este perfil describe los modos de operación, tanto obligatorios como opcionales, para los dispositivos *Bluetooth*. A continuación se mencionan estos modos.

5.1.1.1 Descubrimiento

Este modo gobierna el uso de *inquiry scan* y determina la manera en que otros dispositivos pueden descubrir a un dispositivo *Bluetooth* que entre en su área de cobertura.

Existen tres tipos diferentes de modos de descubrimiento: *no habilitado para descubrimiento*, *descubrimiento limitado* y *descubrimiento general*.

Un dispositivo *no habilitado para descubrimiento* no efectuará *inquiry scan*, y por lo tanto, no podrá ser encontrado por algún dispositivo que lleve a cabo el proceso de *inquiry*.

Un dispositivo en el modo de *descubrimiento limitado*, usa el código de acceso limitado (LIAC) al realizar *inquiry scan*, el cual le permite ser descubierto únicamente por dispositivos en el estado de *búsqueda* con este código.

Un dispositivo en el modo de *descubrimiento general* usa el código de acceso general (GIAC) al efectuar *inquiry scan*, esto le permite ser encontrado por los dispositivos que usen este código al efectuar es proceso de *inquiry*. El GIAC, es el código comúnmente utilizado.

Alguno de los modos anteriormente mencionados *debe* ser soportado. En el caso de ser el modo de *descubrimiento limitado*, es necesario soportar también el modo *no habilitado para conexión*.

5.1.1.2 Conexión

Este modo gobierna el uso de *page scan* y determina la manera en que otros dispositivos pueden conectase a un dispositivo *Bluetooth* que entre en su área de cobertura.

Existen dos modos de conexión: *habilitado para conexión* y *no habilitado para conexión*. Un dispositivo en el primer modo, realiza periódicamente *page scan* para permitir a otros dispositivos conectarse con él; mientras que un dispositivo en el modo *no habilitado para conexión* no realiza periódicamente *page scan*, de esta manera este dispositivo sólo puede establecer conexión con otro, iniciando el proceso de *page* por iniciativa propia. El modo *habilitado para conexión* es obligatorio, mientras el *no habilitado para conexión*, es opcional.

5.1.1.3 Apareamiento

Este modo establece el uso de las caracteristicas de *apareamiento* utilizadas para crear las llaves de enlace necesarias en los procedimientos de encriptamiento. Existen dos modos: *habilitado* y *no habilitado*. El primero es capaz de establecer una llave de enlace con otro dispositivo, mientras el segundo no.

Las funciones de apareamiento son usadas por capas superiores para el establecimiento de conexiones confiables (*Bonding*) entre dispositivos *Bluetooth*. Si los dispositivos soportan conexiones confiables entonces es obligatorio el uso del modo *habilitado*.

5.1.1.4 Seguridad

Este modo gobierna cuándo y cómo se lleva a cabo el encriptado sobre un enlace. Existen tres modos diferentes:

- *No seguro*: los procedimientos de seguridad nunca se inician.
- *Seguridad a nivel servicio*: la seguridad no se inicia hasta que un canal L2CAP se establece, una vez establecido dicho canal, los procedimientos de seguridad se inician de acuerdo a los requerimientos de los servicios.
- *Seguridad a nivel enlace*: la seguridad se inicia cuando un enlace ACL de *bandabase* se establece.

5.1.2 Perfil de Puerto Serial

Este perfil provee una emulación de un puerto serial RS-232 para los dispositivos *Bluetooth*. Con lo anterior, la información proveniente de las aplicaciones no tiene que modificarse para ser transmitida a través de *Bluetooth*, con lo cual estas aplicaciones tratan una conexión *Bluetooth* como si se tratase de un enlace serial.

Este perfil se basa en el *Perfil de Acceso Genérico*, y sobre él se construyen ocho perfiles más, los cuales son: Red, FAX, *Headset*, Acceso LAN, Intercambio Genérico de Objetos, Sincronización, Implantación de Objetos y Transferencia de Archivos.

El perfil de Puerto Serial usa RFCOMM para proveer la emulación. Al dispositivo que establece la conexión RFCOMM se le llama *iniciador* y al otro *respondedor*. Para establecer una conexión a través de un puerto serial emulado se requiere seguir un conjunto de pasos, los cuales se describen a continuación:

- El primer paso es encontrar la dirección del dispositivo en el otro extremo, para lo cual es posible realizar cualquiera de las siguientes acciones: llevar a cabo el proceso de *inquiry*, solicitar al usuario que introduzca la dirección del dispositivo *Bluetooth*, o bien, que la dirección esté predefinida en el dispositivo *iniciador*.
- El siguiente paso es realizar el proceso de *paging* para crear una conexión ACL.
- Posteriormente se crea un canal L2CAP para llevar a cabo el descubrimiento de servicios a través del SDP. El SDP obtiene la información del canal de servicio RFCOMM para el servicio de puerto serial.
- A continuación se crea un canal L2CAP para RFCOMM en el dispositivo *respondedor*, una vez hecho esto, se inicia una sesión RFCOMM a través del canal L2CAP. Si fuese necesario negociar parámetros de enlace RFCOMM, es el momento de realizarlo, y precisamente se debe de hacer antes de solicitar el enlace RFCOMM para datos.

El soporte de seguridad es obligatorio en el perfil de Puerto Serial. En este punto, para establecer una conexión *confiable* con otro dispositivo es necesario el intercambio de claves PIN entre los dispositivos.

Una vez realizado este proceso, las aplicaciones se pueden comunicar a través del puerto serial virtual utilizando el canal RFCOMM.

5.1.3 Perfil de Intercambio Genérico de Objetos

Este perfil define cómo debe usarse OBEX dentro de *Bluetooth*, también describe cómo establecer la comunicación *cliente/servidor*. No define la capa de aplicación, lo cual sí se hace en los perfiles de Sincronización, Implantación de Objetos y Trasferencia de Archivos. De esta

manera, los perfiles mencionados anteriormente, se basan en el perfil de Intercambio Genérico de Objetos.

Se definen dos papeles, estos son: *cliente y servidor*. El cliente es el dispositivo que implanta o retira objetos, mientras el servidor es el dispositivo donde se implantan o se retiran objetos.

Debido a que la información es transportada mediante OBEX, se define el formato básico para los encabezados. La autenticación no es obligatoria, sin embargo, en el caso de ser usada, debe realizarse el proceso antes de establecer una conexión OBEX.

5.1.4 Perfil de sincronización

Provee una manera estándar de sincronizar datos personales entre dispositivos *Bluetooth*. Este perfil permite mantener actualizada la información entre PDAs, celulares, LAPTOPs y PCs. La sincronización puede realizarse de manera transparente, una vez que se ha establecido la conexión con un dispositivo reconocido como *confiable*.

5.1.5 Perfil de Implantación de Objetos

Este perfil provee los procedimientos necesarios para el intercambio de objetos de formato limitado entre un *cliente* y un *servidor*, está enfocado al intercambio de tarjetas virtuales de negocios y la implantación, por parte del cliente, de objetos de formato específico en el dispositivo *servidor*. De esta manera es el *cliente* quien siempre inicia las operaciones. Para el intercambio de datos de formato libre, se usan otros perfiles.

Para proveer seguridad, es obligatorio para los dispositivos que utilicen este perfil, el soporte de características de autenticación y encriptamiento.

5.1.6 Perfil de Transferencia de Archivos

Este perfil a diferencia del perfil de Implantación de Archivos, permite el intercambio de archivos de formato libre entre diversos tipos de dispositivos *Bluetooth*. El perfil de Transferencia de Archivos se basa en las funciones de OBEX especificadas en el perfil de Intercambio Genérico de Objetos, y este a su vez descansa en el perfil de Acceso Genérico y Puerto Serial.

Este perfil es ideal para la conexión "PC a PC" durante reuniones, esta conexión es capaz proveer: exploración, creación y transferencia de carpetas y archivos entre los dispositivos.

5.2 Seguridad

En contraste con las comunicaciones basadas en cables, inherentemente seguras, las transmisiones inalámbricas requieren un nivel de seguridad mayor debido a que hay una gran probabilidad de que otros "escuchen" la información que se está transmitiendo.

El algoritmo adoptado por *Bluetooth* para la autenticación y cifrado esta basado en otro algoritmo llamado SAFER+, el cual genera llaves de cifrado de 128 bits a partir de entradas de texto de 128 bits. Dentro de *Bluetooth*, el texto se obtiene a partir de la combinación de un PIN del dispositivo, o una llave de unidad, y un número aleatorio. La llave resultante se carga junto con la dirección *Bluetooth*, el reloj del maestro y otro número aleatorio de 128 bits, en un banco de registros de corrimiento lineal con retroalimentación. La salida de estos registros se combina en una máquina de estado finita para producir un *cifrado de flujo*, el cual se usa para cifrar y descifrar los datos aplicando la operación or-exclusiva sobre el flujo de cifrado y los datos.

Existen tres operaciones generales a realizar dentro de todo el proceso: generación de número aleatorios, generación de llaves, encriptación.

5.2.1 Llaves y PINs

La generación de llaves se basa en el algoritmo SAFER+, resulta un procedimiento lento, por lo que no se realiza frecuentemente. Este proceso sólo se lleva a cabo durante la negociación LMP, por lo que no resulta crítico.

Existen diferentes tipos de llaves en *Bluetooth*, las cuales se describirán a continuación.

Llave de enlace

Las llaves de enlace son usadas como llaves de autenticación entre dispositivos *Bluetooth* y para generar llaves de encriptación. Éstas pueden ser: *semipermanentes*, usadas para varias sesiones; o *temporales*, usadas únicamente para una sesión. Cada vez que se genera una llave de enlace, es necesario hacer una verificación mutua.

Llave maestra

Este tipo de llave es para comunicación punto a multipunto, reemplaza a la llave de enlace por un determinado tiempo durante los mensajes de *broadcast*. Esta llave es únicamente temporal.

Llave de unidad

Es una llave *semipermanente* que generalmente es establecida por el fabricante, sin embargo, puede llegar a cambiarse en cualquier momento.

Llave combinada

Este tipo de llaves depende de dos unidades, donde cada unidad produce y manda un número aleatorio al otro dispositivo. Una nueva llave combinada de 128 bits se crea usando SAFER+ para cada nueva combinación. Una llave combinada se crea al final del proceso de *apareamiento*.

Llave de inicialización

Es una llave de 128 bits usada en una sola sesión, la cual se crea cada vez que se inicializa la unidad. Esta llave se usa únicamente cuando no han sido intercambiadas llaves combinadas, o de unidad entre los dispositivos. La creación de esta llave se requiere un PIN y se lleva a cabo al inicio del proceso de apareamiento.

Llave de encriptamiento

Esta llave se deriva de la llave de enlace en uso. El mecanismo de encriptación la usa para producir el *cifrado de flujo*.

PIN

Para usar la característica de encriptación, el maestro y el esclavo deben compartir la misma clave secreta, la cual nunca es transmitida al aire. La llave secreta puede ser establecida por el fabricante, o bien, puede crearse tomando como parámetro un PIN introducido por el usuario.

5.2.2 Apareamiento y vinculación

En el perfil de acceso genérico se dice que dos dispositivos están *vinculados* si se sabe que comparten la misma llave de enlace. Los procedimientos involucrados en crear una relación basada en una llave de enlace común se llaman *vinculación*.

La *vinculación* involucra el establecimiento de un enlace específicamente con el propósito de crear e intercambiar una llave de enlace común. Durante la *vinculación*, los administradores de enlace crean e intercambian una llave de enlace y posteriormente la verifican mutuamente. A los procesos en el nivel de enlace, de generación e intercambio de una llave de enlace, se les llama *apareamiento*.

La *vinculación* puede involucrar tanto apareamiento, en el nivel de enlace, como procesos en las capas superiores [5].

5.2.2.1 Autenticación

Es el proceso mediante el cual los dispositivos verifican que comparten la misma llave de enlace. Este proceso se lleva a cabo mediante un intercambio de mensajes usando el protocolo de administración de enlace.

La *autenticación* se realiza generalmente antes de la *encriptación*, sin embargo, puede realizarse de manera independiente en cualquier momento. Es decir, un dispositivo puede realizar la autenticación para comprobar que se está comunicando con la unidad adecuada, sin que ello implique un encriptado de los datos.

Como resultado de una *autenticación* exitosa, se generan parámetros que son usados para crear la llave de cifrado.

Antes de realizar la *autenticación*, ambos dispositivos deben inicializar su sistema de *encriptación* con un mismo número. Este número es un número aleatorio que es enviado dentro de un mensaje LMP, de esta manera ambos dispositivos usan el mismo número.

Posteriormente el dispositivo *verificador* envía un mensaje LMP que contiene otro número aleatorio que será autentificado por el dispositivo *verificado*. El dispositivo *verificado* encripta el número usando la llave de enlace, después lo regresa en otro mensaje LMP. El *verificador* encripta el número aleatorio que envió usando su llave de enlace y lo compara con el recibido. De esta manera el *verificador* puede comprobar si ambos dispositivos están usando la misma llave de enlace sin que ésta sea transmitida por el aire.

5.2.2.2 Generación de la llave de enlace

Para obtener una llave compartida (llave de enlace), cada unidad manda en un mensaje LMP su llave de unidad, o una llave combinada. Antes de transmitir la llave, cualquiera que sea, se realiza la operación or-exclusiva entre dicha llave y la llave de inicialización, con el objetivo de incrementar la seguridad.

Las reglas para la determinación de la llave compartida son las siguientes:

- Si ambos dispositivos envían la llave de unidad, se usa la llave de unidad del maestro.
- Si un dispositivo envía la llave de unidad y el otro una llave combinada, se usa la llave de unidad.
- Si ambos dispositivos envían una llave combinada, se genera una llave basada en ambas.

Como se puede observar, la llave de enlace puede ser bien una llave de unidad, o una llave basada en las llaves combinadas de las unidades. Es evidente, que el intercambio de llaves combinadas presenta un nivel de seguridad mayor.

Después de la generación de la llave de enlace, ambos dispositivos se autentican mutuamente. La figura 5.2 muestra el intercambio de mensajes LMP para la autenticación de dos dispositivos. Este diagrama representa un caso particular en donde la llave de enlace se genera a partir de dos llaves combinadas.

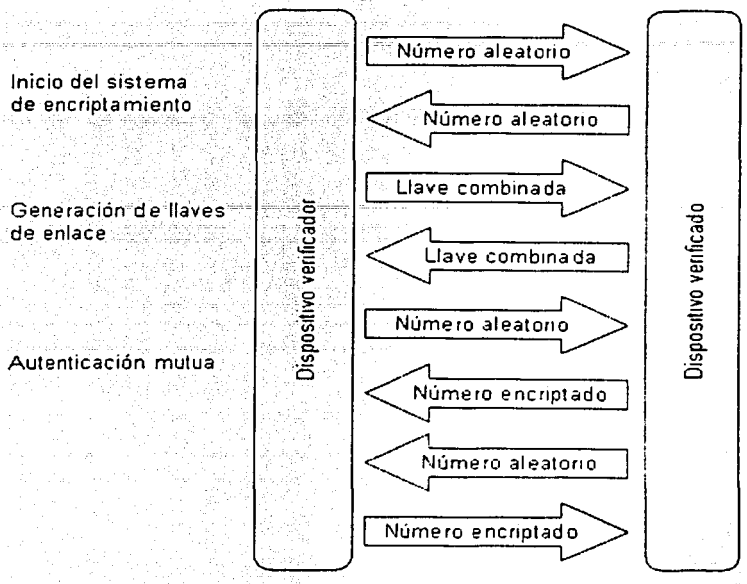


Figura. 5.2 Intercambio de mensajes LMP durante la autenticación

5.2.2.3 Cambio de llaves de enlace

Si por alguna razón el *host* decide cambiar la llave de enlace porque supone que ésta se encuentra comprometida, lo puede hacer. El procedimiento de cambio es el mismo que se usa para negociar la llave inicialmente. Debido a que cada conexión usa una llave diferente, se requiere un identificador para el manejo de cada llave de enlace.

5.2.2.4 Cambio a llaves temporales

Otro caso en el que se cambia la llave de enlace es cuando la información de *broadcast* requiere ser encriptada. Para ello se usa una llave temporal.

Un esclavo no sabe, hasta que recibe el paquete, si éste es de *broadcast*, o punto a punto, por lo que no tiene tiempo para conmutar entre una llave de enlace para *broadcast* y otra para punto a punto, es por esta razón que debe usar una misma llave para ambos tipos de paquetes.

Debido a que un mensaje de *broadcast* se envía a todos los dispositivos de la *piconet*, ahora ellos compartirán la misma llave de enlace y la seguridad quedará comprometida. Debido a este último punto, la llave de enlace debe regresar al modo normal tan pronto como se deshabilite el modo de encriptación para *broadcast*. Por ello se puede afirmar que la llave de enlace para *broadcast* es temporal y solo será válida para la sesión en uso, así que cada vez que una nueva

sesión de *broadcast* se inicie, una nueva llave temporal tendrá que generarse y posteriormente llevarse a cabo una autenticación mutua.

Debido a que el maestro es el único que puede transmitir mensajes de *broadcast*, es él quien crea la llave de enlace temporal.

5.2.2.5 Regreso a llaves semipermanentes

Las llaves semipermanentes de enlace representan las llaves de enlace usadas en las comunicaciones punto a punto. En el caso del uso de una llave temporal, la llave semipermanente de enlace es la que estaba en uso anteriormente, por lo que ambos dispositivos la conocen. Esto significa que no hay necesidad de realizar todo el proceso nuevamente, tan solo se requiere que el maestro envíe un mensaje informando al esclavo que regrese a la llave de enlace semipermanente. Al igual que con todos los cambios de llave de enlace, la encriptación de los datos debe detenerse e iniciar nuevamente con la nueva llave de enlace.

5.2.2.6 Almacenamiento de llaves de enlace

Hasta ahora se ha descrito la manera de generar las llaves de enlace por medio de negociación, sin embargo, el *host* tiene la capacidad de escribir la llave de enlace al módulo a través del HCI. Es decir, es posible que el *host* lea y almacene la llave de una sesión para su uso en otra sesión posterior.

El almacenamiento de las llaves de enlace requiere que el *host* posea memoria no volátil, a cambio, agiliza el proceso de *autenticación/encriptación* y hace que los módulos *Bluetooth* sean más baratos, debido a que en estos no se tiene que implementar memoria no volátil para soportar esta característica. En el caso de una *laptop* con el módulo *Bluetooth* implementado en una tarjeta PCM-CIA resulta ventajoso que las llaves de enlace se almacenen en el *host*, esto permite intercambiar las tarjetas sin ningún problema, mientras las llaves permanecen seguras en la computadora (donde se les puede proteger con algún *password*).

Cada vez que se genera una nueva llave de enlace, se envía una notificación al *host* que contiene la llave de enlace y la dirección *Bluetooth* del dispositivo en el otro extremo. Esta información puede ser guardada en el *host* para una posterior utilización. Cuando el módulo *Bluetooth* quiere recuperar una llave de enlace del *host* realiza una petición a través de HCI, utilizando como único parámetro la dirección *Bluetooth* del dispositivo que se encuentra en el otro extremo del enlace ACL.

5.2.2.7 Vinculación dedicada y general

El perfil de acceso genérico divide la *vinculación* en: *dedicada y general*. La *vinculación dedicada* es aquella en donde los dispositivos solamente crean e intercambian llaves de enlace, y tan pronto como la autenticación en el nivel de enlace ha sido terminada, el canal se libera antes de que las capas superiores establezcan conexión. La *vinculación general* puede involucrar el intercambio de datos de las capas superiores para inicializar sus parámetros de seguridad.

Las llaves de enlace para los dispositivos *vinculados* se almacenan, por lo que no se requiere generarlas cada vez que estos dispositivos establezcan conexión. Cabe destacar que al realizar la *vinculación*, se generan llaves de enlace, por lo que se borrarán las llaves de enlace anteriores antes de realizar una nueva *vinculación*.

Una vez que dos dispositivos han sido *vinculados*, comparten la llave de enlace, por lo que pueden establecer conexión sin necesidad de realizar los procedimientos de *apareamiento*.

5.2.3 Encriptación

Cuando dos dispositivos *Bluetooth* se han *autenticado* y han acordado la llave del enlace, pueden comenzar el procedimiento para la *encriptación*. Los pasos para iniciar la *encriptación* son: negociación del modo de encriptación, negociación del tamaño de la llave de encriptación, inicio de la encriptación [22].

Los posibles modos de encriptación son: *sin encriptación, encriptación tanto de paquetes punto a punto, como broadcast y encriptación sólo de paquetes punto a punto*. El modo de *encriptación* puede cambiarse en cualquier momento, para ello la transmisión de datos debe ser detenida, con el fin de evitar la pérdida de paquetes por indeterminación en el modo de *encriptación* durante la transición.

La negociación del tamaño de la llave la realiza el maestro. El maestro usa inicialmente el máximo tamaño de llave que puede soportar, si este tamaño está dentro de la capacidad del esclavo, ya no hay cambio. Si el esclavo no tiene la capacidad para ese tamaño de llave, el maestro debe intentar con otra más pequeña, hasta encontrar una que el esclavo pueda usar.

Cuando se ha completado la negociación del modo y el tamaño de llave, se puede seleccionar entre habilitar o deshabilitar la *encriptación*. Para conmutar entre estas dos posibilidades, es necesario detener la transmisión de datos antes de iniciar el proceso.

Cabe destacar que también es posible para el esclavo realizar la *autenticación, apareamiento, negociación de modos* y la *conmutación de papeles*; no así la negociación del tamaño de la llave.

5.2.4 Modos de seguridad

El perfil de acceso genérico define 3 modos de seguridad: *modo no seguro, modo de seguridad a nivel servicio* y *modo de seguridad a nivel enlace.*

El *modo seguridad 1*, o no seguro, es aquel bajo el cual los dispositivos nunca inician ningún procedimiento de seguridad. El soporte de *autenticación* es opcional para los dispositivos que sólo soportan este modo.

En el *modo de seguridad 2*, o en el nivel de servicio, el canal o servicio que usa una conexión L2CAP decide cuándo usar los procedimientos de seguridad. De esta manera, hasta el momento de haber establecido un canal L2CAP, es posible para un dispositivo en este modo, iniciar algún procedimiento de *autenticación*. Una vez establecido el canal L2CAP, el dispositivo decide si es necesario, o no, usar procedimientos de *autorización*, *autenticación*, o *encriptación* para acceder a algún servicio.

El *modo de seguridad 3*, o en el nivel de enlace, permite realizar los procedimientos de seguridad antes de iniciar la conexión, de tal manera que si fallan, la conexión no se establece. Mediante este método es posible configurar al dispositivo *Bluetooth* para que solamente acepte conexiones con dispositivos preestablecidos.

Además de los modos descritos anteriormente, existen otros dos modos relacionados con la seguridad, estos son: el *modo no habilitado para conexión* y el *modo no habilitado para descubrimiento*. El dispositivo que se encuentra en el *modo no habilitado para conexión*, no responde a los llamados de otros dispositivos (*paging*), de tal forma que ningún dispositivo se puede conectar con él, a menos que él empiece por iniciativa propia el proceso de *paging*. En el *modo no habilitado para descubrimiento*, el dispositivo no responde a las *búsquedas* (*inquiry*) de otros dispositivos, de esta manera solo se pueden conectar con él aquellos dispositivos que conozcan de antemano su dirección de *Bluetooth* [23].

5.2.5 Arquitectura

Debido a que el *modo de seguridad 2* está implementado en el nivel de servicio, éste no afecta la interoperabilidad entre los dispositivos. Por lo anterior, se sugiere que la arquitectura del sistema de seguridad esté basada en este modo de seguridad. En este tipo de arquitectura, se clasifican los dispositivos y los servicios dentro de una base de datos.

Los dispositivos se dividen en tres categorías:

- Dispositivos *confiables*. Son los dispositivos, *apareados y vinculados*, que han sido marcados como *confiables* dentro de la base de datos; éstos tienen acceso total a todos los servicios.
- Dispositivos *conocidos no confiables*. Estos dispositivos han sido *apareados y vinculados*, pero no marcados como *confiables*; el acceso a los servicios puede estar restringido.

- Dispositivos *desconocidos*. No se tiene información sobre la seguridad de los dispositivos, éstos son *no confiables*; el acceso a los servicios puede estar restringido.

Al igual que los dispositivos, los servicios también están clasificados. Esta clasificación se presenta a continuación:

- Servicios *abiertos*. Cualquier dispositivo puede acceder a ellos, no hay requisitos de seguridad.
- Servicios disponibles con *autenticación*. Cualquier dispositivo que haya aprobado la autenticación puede hacer uso de ellos. La autenticación garantiza que compartan la misma llave secreta el *cliente* y el *servidor*.
- Servicios disponibles con *autenticación* y *autorización*. Sólo los dispositivos *confiables* pueden acceder a ellos.

Cada servicio debe establecer su nivel de seguridad independiente de los otros servicios, de esta manera, un dispositivo con acceso a un servicio podría no tener acceso a otros.

El manejo de la base de datos que contiene la información de los dispositivos y servicios es realizado por una entidad llamada *administrador de seguridad*. Esta entidad gestiona todas las transacciones entre la base de datos y las diversas capas.

Es importante tener en cuenta que el aspecto de seguridad es de suma importancia dentro de *Bluetooth*, sin embargo, para que los dispositivos sean fáciles de utilizar, es necesario implementar la seguridad a un nivel confiable, pero escondiendo la complejidad de los procedimientos para el usuario final.

5.3 Calidad de servicio (QoS)

La *calidad de servicio (Quality of Service)*, en la tecnología *Bluetooth*, se enfoca en los parámetros de tasa de transmisión, variación del retardo y confiabilidad. Para obtener una *calidad de servicio* en un enlace *Bluetooth*, es necesaria la interacción de diversas capas del protocolo. Las capas relacionadas con esta función son: LM, HCI y L2CA [5][22].

La *calidad de servicio* que una aplicación requiere, está determinada por el tipo de tráfico. En el caso de tráfico asíncrono (ACL) es importante obtener la máxima tasa de transferencia de manera confiable, no importa que el enlace tenga características variables (por ráfagas, *bursty*), siempre y cuando los datos sean recibidos íntegramente. Por otro lado, el tráfico síncrono (SCO) requiere un enlace con características constantes, y a cambio, está dispuesto a perder cierto número de paquetes mientras que el retardo no sea muy grande.

Los parámetros que pueden ser configurados para ofrecer una QoS dentro de *Bluetooth* son:

- Tipo de QoS: Garantizado, *Mejor esfuerzo*, sin servicio.

- *Token rate.* Es la tasa de transferencia, en bytes por segundo, que se le puede garantizar a una aplicación que use el canal.
- *Token rate bucket size.* Representa el tamaño del *buffer* que debe estar disponible para la recepción de los datos. Los enlaces variables (*bursty*) requieren más capacidad de almacenamiento que los enlaces que fluyen constantemente.
- *Peak bandwidth.* Representa la tasa de transmisión máxima a la que pueden ser enviados los paquetes sobre un enlace, cuando no existen otros enlaces que interfieran.
- Latencia. Representa el retardo máximo que existe entre el momento en el que los datos están listos para ser enviados y el momento en el que son transmitidos al aire por primera vez.
- Variación del retardo. Es el espaciamiento que hay entre el máximo y el mínimo retardo a través del enlace.

En la figura 5.3 se muestran los mensajes utilizados para establecer y configurar la *calidad de servicio*. Algunos de los mensajes fluyen verticalmente a través de la pila de protocolos de *Bluetooth;* mientras otros, fluyen horizontalmente durante la negociación entre entidades homólogas entre los extremos. Tanto L2CA como LM, son las capas que realizan negociaciones entre entidades homólogas.

La capa L2CA realiza una petición de QoS mediante el protocolo L2CAP, en el extremo opuesto, su homóloga hace una petición al *administrador de enlace* (LM) para cumplir con la *calidad de servicio* solicitada. En los sistemas que poseen un HCI, la interacción entre la L2CA y el *administrador de enlace* (LM) se realiza a través de comandos y eventos HCI.

El *administrador de enlace* configura y controla los enlaces de *bandabase*, de tal manera que es él quien implementa las características solicitadas. El LM posee varios medios para tratar de cumplir con la QoS solicitada, los cuales incluyen: la selección de tipos de paquetes, establecimiento de *intervalos de emisión*, asignación de *buffers*, asignación de ancho de banda y decide cuándo realizar los procedimientos de *exploración*.

Para asegurar que los datos sean removidos del *buffer* de transmisión lo suficientemente rápido para satisfacer el *token rate* que fue garantizado en la conexión, el LM establece un *intervalo de emisión*. Este intervalo representa el máximo tiempo entre transmisiones subsecuentes de un mismo enlace, es decir, de un Maestro a un Esclavo en particular. Este *intervalo de emisión* afecta la característica de latencia y al ancho de banda de un enlace, por lo que debe ser constante para satisfacer una determinada QoS.

Es posible que el *administrador de enlace*, modifique el comportamiento de todo el sistema para tratar de cumplir con la *calidad de servicio* deseada. Por ejemplo, el LM puede rechazar

todas las peticiones de conexión de otros dispositivos, o bien, detener los procesos periódicos de exploración (*page* o *inquiry*).

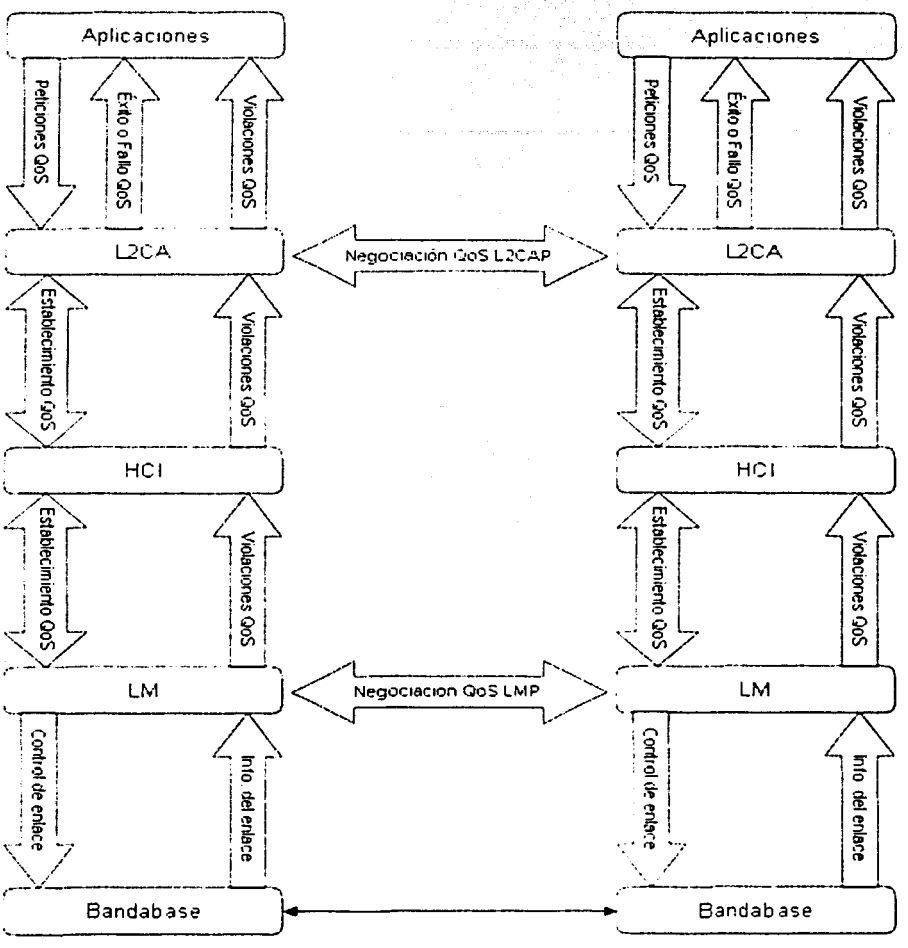


Figura 5.3. Mensajes usados para establecer y configurar la calidad de servicio

Es importante que el *administrador de enlace* que se encuentra en el extremo que recibe la petición de QoS, tome las medidas necesarias para llevar a cabo dicha solicitud. Puede resultar inútil que un LM transmita paquetes rápidamente, si su homologo en el extremo opuesto, no puede recibirlos y transmitirlos suficientemente rapido a las capas superiores. Es por esto, que

además de controlar las operaciones de *bandabase*, es necesario realizar negociaciones de QoS a nivel del *administrador de enlace*.

Cuando un L2CAP realiza una solicitud de configuración de QoS, los administradores de enlace llevan a cabo negociaciones. Si los LM están de acuerdo en que es posible alcanzar la *calidad de servicio* solicitada, se informa al L2CAP que realizó la petición, y éste a su vez informa a su homólogo L2CAP que la configuración QoS fue exitosa. De otra manera, si la configuración falla, el L2CAP (que realizó la petición) debe decidir entre darse por vencido o intentar nuevamente con otros parámetros. Esta decisión es en realidad tomada por la aplicación, sobre el L2CAP, que solicitó la QoS.

La *calidad de servicio* deseada puede no ser alcanzada debido a una interferencia en el enlace de radio, o bien, porque los dispositivos no posean los recursos necesarios.

5.3.1 Solicitud de calidad de servicio

Cuando un enlace se establece por primera vez, existe una etapa de negociación donde se configuran los parámetros del enlace. Es en este punto donde se configura la seguridad y la *calidad de servicio*. Al igual que la seguridad, la calidad de servicio puede cambiarse durante el enlace, esto puede deberse a que un nuevo servicio quiera usar dicho enlace, o a que las condiciones del ambiente hayan cambiado.

A continuación se mencionan los pasos involucrados en un el establecimiento de QoS.

1. Solicitud de *calidad de servicio* de una aplicación o protocolo superior hacia L2CA. El mensaje que contiene la petición de QoS tiene como parámetros un identificador de canal y las caracteristicas deseadas para dicho canal.
2. Negociación de *calidad de servicio* entre capas L2CA homólogas. Para iniciar la negociación de QoS, la capa L2CA local envía un mensaje a su homóloga en el otro extremo del canal. El mensaje enviado contiene los valores de todos los parámetros que determinan la *calidad de servicio* (tipo de servicio, *token rate*, *token rate buket size*, *peak bandwidth*, latencia, variación del retardo). Si el lado que recibe la petición está en desacuerdo con los valores de los parámetros, éste envía un mensaje indicando que la configuración ha fallado. Cuando la capa L2CA que inicio la petición de QoS recibe un mensaje de configuración fallida, puede seguir intentando con valores de parámetros diferentes hasta estar de acuerdo con la capa L2CA remota, o bien, puede decidir que el nivel de QoS ofrecido por el otro extremo es inaceptable y a continuación cerrar la conexión.
3. Establecimiento de QoS a través del HCI. En los sistemas con un *host* y módulo *Bluetooth* separados, el HCI proporciona los medios para que el L2CA controle al *administrador de enlace*. El *administrador de enlace* provee las caracteristicas necesarias para satisfacer la solicitud de QoS. En sistemas con HCI, las peticiones hacia

el LM se envían en comandos HCI. Cuando una capa L2CA recibe un mensaje de solicitud de QoS, dicha capa debe investigar si el *administrador de enlace* puede satisfacer dicha petición, para ello se lleva a cabo un intercambio de mensajes HCI.

4. Negociación de QoS a nivel del *administrador de enlace*. El *administrador de enlace* se vale de comandos LMP para configurar los valores de parámetros de *intervalo de emisión* y número de repeticiones para paquetes de *broadcast*. Debido a que los paquetes de *broadcast* no son *confirmados* es necesario enviarlos repetidamente para garantizar su correcta recepción. Es importante mencionar que con un mayor número de retransmisiones de los paquetes de *broadcast*, se obtiene mayor confiabilidad para su recepción, pero se utiliza más ancho de banda.
5. Finalización del establecimiento de QoS. Cuando el LMP concluye el establecimiento de QoS, éste envía un mensaje al L2CA local. El mensaje contiene un campo de estado que indica si la petición de *calidad de servicio* fue establecida de manera exitosa o fallida. La capa L2CA local transmite el resultado de dicho mensaje a su homólogo en el extremo opuesto. La L2CA que envió originalmente la petición QoS, manda un mensaje a la aplicación que inició la solicitud, en dicho mensaje informa el estado de la petición de *calidad de servicio*. Si la solicitud fue fallida, es decisión de la aplicación intentar nuevamente con otras características de *calidad de servicio*, o bien, darse por vencida. En el caso de que la solicitud haya sido exitosa, se procede a abrir el canal para la transmisión de datos con la *calidad de servicio* solicitada.

5.3.2 Violaciones de QoS

Una vez establecida una determinada *calidad de servicio* sobre un canal, es posible que la interferencia, o errores, del sistema no permitan alcanzarla. Este caso se presenta cuando el *buffer* de transmisión no puede vaciarse a la velocidad acordada, entonces, el LM envía un mensaje de *violación de QoS* hacia la L2CA, ésta a su vez informa a la aplicación. La aplicación es finalmente quien decide si debe reconfigurar la conexión, o tomar otro tipo de acción.

5.3.3 Retardos

El *administrador de enlace* establece los valores del *intervalo de emisión* para evitar las variaciones de retardo; sin embargo, es posible que la interferencia no permita la transmisión correcta de los datos. En este caso, los datos viejos se acumularán en el *buffer*, hasta que la *bandabase* logre transmitir correctamente el paquete actual. En sistemas con una HCI, se implementa un control del flujo hacia el *host*, esto permite que el módulo siga tratando de enviar los datos viejos, mientras que el *host* espera y no envía datos nuevos que desbordarían el *buffer* del módulo.

El sistema mencionado anteriormente resulta de utilidad para el tráfico asíncrono (ACL), donde se requiere que todos los paquetes se reciban íntegramente; por otra parte, para el tráfico síncrono (SCO) lo más importante es recibir información actualizada continuamente. Para el tráfico síncrono existe otro mecanismo, para garantizar una tasa de transmisión constante, basado en la *limpieza* del *buffer*.

Para los sistemas que poseen un HCI, existe un comando que el *host* envía al módulo para indicarle que limpie (tire) los datos que tiene en el *buffer*, esperando a ser transmitidos. Para reducir el número de comandos intercambiados entre el *host* y el módulo, es posible configurar al módulo para que limpie los datos viejos, que no han podido transmitirse, después de un cierto tiempo preestablecido.

La limpieza afecta a todo un paquete L2CAP y se vuelve a la transmisión normal cuando se detecta el principio de un nuevo paquete L2CAP.

Cuando se presentan varias operaciones de *limpieza* continuamente, es posible que el dispositivo en el otro extremo haya salido de rango, o haya sido apagado. Si esto ocurre, un temporizador de supervisión termina y el enlace se cierra, con lo que se libera la dirección de miembro activo, para así poder reutilizarla posteriormente. Este temporizador de supervisión, diferente para cada enlace, es establecido por el Maestro, debido a que es él quien determina la frecuencia con la que se comunica con cada esclavo.

5.3.4 Tasas de transmisión

La tasa de transmisión se puede controlar a través de la selección adecuada del tipo de paquete que llevará los datos. Para enlaces ACL, la tecnología *Bluetooth* permite el uso de paquetes DM y DH. Los paquetes DH alcanzan tasas de transferencia mayores usando menos corrección de errores en los paquetes, lo que deja más espacio para datos; por otro lado, los paquetes DM ofrecen una mayor protección contra los errores con una menor tasa de transferencia. Como se mencionó en el capítulo 3, en la sección de *bandabase*, existen paquetes DM y DH que ocupan una, tres, o cinco ranuras. Es importante aclarar que los paquetes que ocupan 1 ranura, independientemente del tipo de paquete (DM o DH), portarán menos datos que los paquetes que ocupan 3 ranuras, y éstos últimos, contendrán menos datos que los paquetes de 5 ranuras.

La elección del tipo de paquetes en dos dispositivos determina si el canal es simétrico o asimétrico.

Un canal simétrico usa el mismo tipo de paquetes en ambas direcciones. Este tipo canal resulta útil cuando el Maestro y el Esclavo requieren transferir la misma cantidad de información. Es común que existan aplicaciones, o sistemas, donde se necesite transferir información más rápido en uno de los sentidos. En estos casos se usan canales asimétricos.

En los canales asimétricos se utiliza un paquete de mayor tamaño en una dirección. Esta implementación permite obtener una mayor velocidad en el sentido que viajan los paquetes de mayor tamaño y una velocidad menor en el sentido de los paquetes pequeños.

En la tabla 6.1 se muestran las máximas tasas de transferencia teóricas para los diversos tipos de paquetes ACL, sobre canales simetricos.

Generalizando, para calcular las tasas de transferencia de un canal se utilizan las siguientes ecuaciones:

$$T_{AB} = \frac{B_B}{(R_A + R_B)*(625\,\mu S)}$$

$$T_{BA} = \frac{B_B}{(R_A + R_B)*(625\,\mu S)}$$

Donde T_{AB} representa la tasa de transferencia del dispositivo A hacia el B, y T_{BA} la tasa de transferencia del dispositivo B, hacia el A. El subindice A o B indica a qué dispositivo pertenece dicho parámetro. R es el número de ranuras por paquete y B es la cantidad de datos que transporta el paquete.

Tipo de paquete	Número de ranuras por paquete	Máxima carga útil (bytes)	Máxima tasa de transferencia simétrica (kbps)
DM1	1	17	108.8
DH1	1	27	172.8
DM3	3	121	258.1
DH3	3	183	390.4
DM5	5	224	286.7
DH5	5	339	433.9

Tabla 6.1. Máximas tasas de transferencia teóricas para paquetes ACL sobre canales simétricos

Se debe tener en cuenta que las tasas de transferencia en el nivel de aplicación son menores que las teóricas. Las principales causas de la disminución de la eficiencia son:

- La reducción de espacio disponible (en los paquetes) para datos útiles, debido a la introducción de información de control y encabezados usados por los protocolos.
- La retransmisión de paquetes debido a la interferencia.

Las condiciones del canal determinan la eficiencia en uso de los diversos tipos de paquetes. Por ejemplo, en un canal con mucha interferencia lo más conveniente es usar un paquete que

ofrezca una alta protección contra errores, a pesar de que tenga poco espacio para la carga útil, de esta manera se evitarán las retransmisiones del mismo paquete.

Para tratar con canales que tengan características variables, *Bluetooth* ofrece un mecanismo para la conmutación automática entre paquetes[1] DM y DH. La conmutación está basada en el parámetro de tasa de bits erróneos (BER) que hay en el enlace.

Hasta este momento se han estudiado detalladamente las diversas capas que componen la pila de protocolos de la tecnología *Bluetooth*. En el siguiente capítulo se describirá la implementación de estos protocolos en una red punto a punto de área personal.

Naturalmente, el establecimiento y operación de la red requiere de los conceptos tratados hasta ahora, por lo cual, en el siguiente capítulo se hará referencia a ellos únicamente, evitando así crear redundancia. La descripción de la red abordará aspectos específicos relacionados con la aplicación propuesta.

[1] Los paquetes DM y DH deben ser del mismo número de ranuras.

CAPÍTULO 6
IMPLEMENTACIÓN DE UNA RED WPAN DE TIPO BLUETOOTH Y DESARROLLO DE UNA APLICACIÓN.

En el presente capítulo se describirá el procedimiento empleado en el establecimiento de una red inalámbrica de área personal (WPAN) "punto a punto" de tipo *Bluetooth*, así mismo, se explicará el funcionamiento de una aplicación desarrollada para la red implementada. A través de la aplicación, se usarán los conceptos y protocolos mencionados a lo largo de esta tesis. Con esto, se mostrará la importancia del análisis y estudio detallado de todas las características de la tecnología *Bluetooth*, para demostrar su viabilidad y funcionalidad.

6.1 El *kit* de desarrollo *Bluetooth*

Para la implementación de la red y el desarrollo de la aplicación de esta tesis, se adquirieron dos *kits de desarrollo Bluetooth*. Mediante estos equipos de desarrollo es posible realizar aplicaciones para el intercambio de datos utilizando la transmisión inalámbrica.

Cada *kit de desarrollo* está constituido por una parte de *hardware* y otra de *software*. Ambas partes cumplen con las características que conforman el estándar 802.15 de IEEE.

6.1.1 Descripción del *hardware*

El *hardware* utilizado para la implementación de la red y el desarrollo de la aplicación consiste en dos tarjetas de circuito impreso equipadas con un buffer UART, un regulador de voltaje, elementos pasivos y el Módulo *Bluetooth* fabricado por ERICSSON. Dicho módulo (ROK101007) precalificado[1] incluye un dispositivo de Bandabase, una Memoria *Flash* y un dispositivo de Radio. Los detalles técnicos del módulo pueden consultarse en el anexo F. En conjunto, todos los componentes poseen características que requieren un

[1] El módulo está especificado y diseñado de acuerdo al Sistema *Bluetooth* v 1.0B

bajo consumo de energía para su uso en dispositivos operados con baterías. En la figura 6.1 se muestra una de las tarjetas, en donde podemos observar el módulo *Bluetooth* y los demás componentes del *kit*. La figura 6.2 muestra a detalle el módulo de radio *Bluetooth*.

Algunas características importantes del módulo son las siguientes:

- Es un Módulo precalificado con la versión 1.0B.
- Potencia de RF clase 2.
- Aprobado por FCC y ETSI.
- Incluye las siguientes interfaces para aplicaciones:
  UART para datos.
  PCM para voz.
  USB para voz y datos.
- Oscilador interno de cristal.
- Incluye HCI *firmware*
- Interfaz de LLC.
- Operación multipunto.
- Soporta todos los *perfiles Bluetooth*.

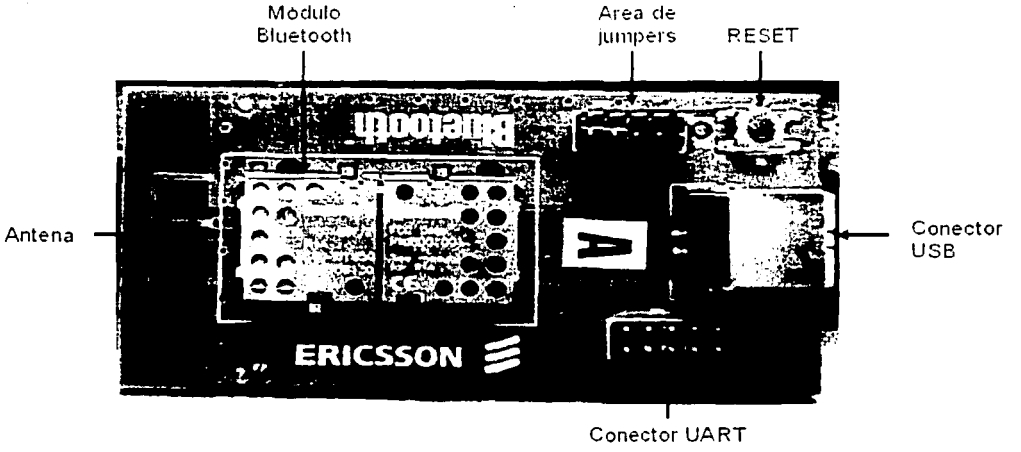


Figura. 6.1 Tarjeta del *kit Bluetooth*.

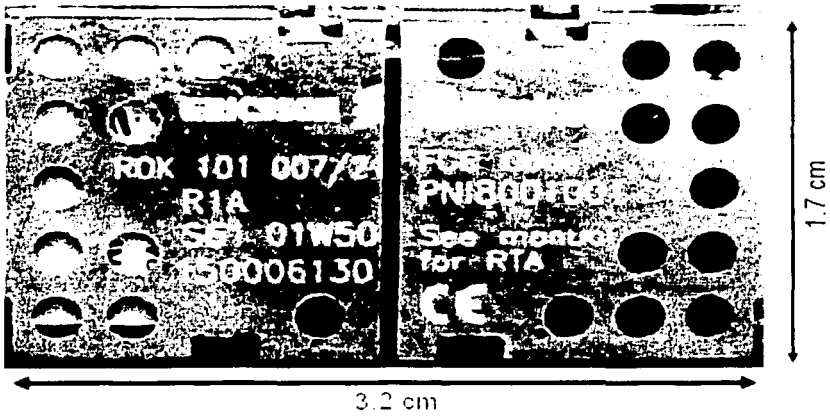


Figura 6.2 Detalle del modulo de radio *Bluetooth*.

La interfaz PCM permite la utilización de voz junto con un CODEC externo, pero no es soportado en la implementación del *kit*.

6.1.2 Descripción del *software*

El *kit* contiene el *software PC Referente Stack*, el cual contiene programados, en lenguaje C++, los comandos usados en los protocolos *Bluetooth* de alto nivel para que sean integrados en un entorno Win32. Al conjunto de protocolos, se le ha denominado "Pila de protocolos Ericsson" y han sido precalificados por el BQB (*Bluetooth Qualification Body*).

También dentro del *software*, existe un programa que funciona como servidor COM[2] (*Component Object Module*) el cual se comunica con las tarjetas por medio de la interfaz SERIAL, o el USB.

6.1.3 Descripción a nivel del sistema

Para explicar el funcionamiento del equipo utilizado, resulta útil representar los componentes mediante el esquema de la figura 7.3. El esquema se encuentra dividido en dos partes, separadas por una flecha que representa la interfaz serial o USB, que se utiliza para comunicar al *Host* y al módulo *Bluetooth*.

En la parte inferior tenemos el módulo *Bluetooth*, que son las tarjetas que contienen el módulo de radio. Esta parte es la equivalente a las capas de Radio y Bandabase de la pila de protocolos *Bluetooth*.

[2] Basado en componentes de *software* que permiten la construcción de aplicaciones utilizando un *software* autocontenido en pequeñas y potentes unidades.

En la parte superior tenemos la representación del *host*. De abajo hacia arriba, el primer bloque representa el Servidor COM de *Windows*, que contiene la pila de protocolos contenida en el *software* desarrollado por ERICSSON; el segundo bloque representa la parte del Cliente COM de *Windows*, que contiene la aplicación que maneja los protocolos *Bluetooth* para una solución determinada.

Esto equivale a las capas superiores de la pila de protocolos *Bluetooth*, y muestra cómo van a ser utilizadas para el desarrollo de la aplicación propuesta.

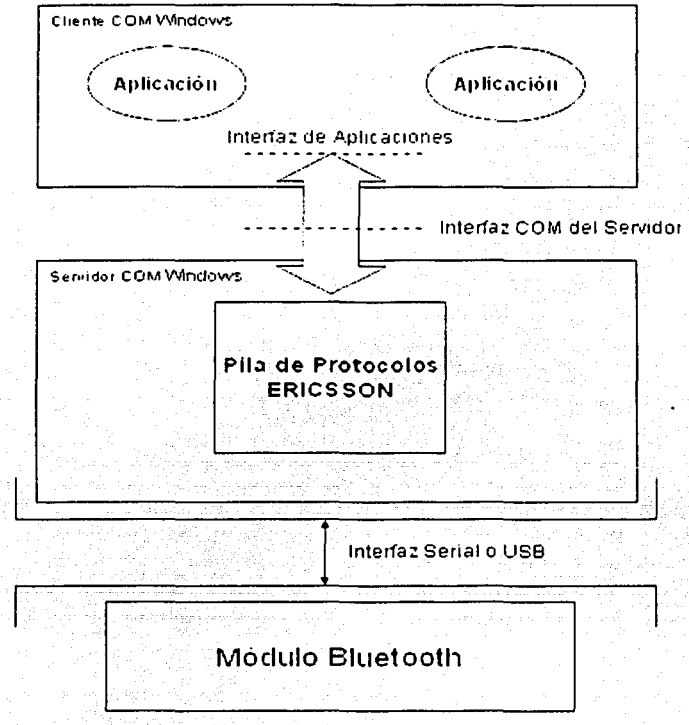


Figura. 6.3 Componentes del *PC Referente Stack*..

6.2 Definición de la aplicación

La aplicación desarrollada tiene por objeto la transmisión inalámbrica de datos entre una PC de escritorio y una *Laptop*, a través de una interfaz gráfica amigable para el usuario, usando la tecnología *Bluetooth*.

El escenario de la aplicación es una sala de reuniones u oficina, en donde resulta de utilidad el establecimiento automático de una red inalámbrica, para el intercambio de datos. El establecimiento espontáneo de una red inalámbrica es una necesidad en este tipo de escenario, debido a que los usuarios de los dispositivos presentan un gran dinamismo. Si el proceso de establecimiento de la red no fuera espontáneo o automático, resultaría poco práctico su uso.

El tipo de información que la aplicación permite intercambiar está conformada por archivos y mensajes de texto, mediante el modelo *cliente/servidor* y el perfil de puerto serial del estándar *Bluetooth*. La aplicación establece un servicio diferente para cada una de las funciones: transferencia de archivos e intercambio de mensajes.

Una vez que el dispositivo denominado CLIENTE, haga una petición de conexión con el dispositivo SERVIDOR, ambos dispositivos podrán enviar o recibir archivos y/o mensajes.

La aplicación habilita el cifrado y la autenticación. Debido a que, por razones de presupuesto, sólo se contó con dos módulos *Bluetooth* para el establecimiento de la red, el cifrado y la autenticación restringen la comunicación para que sólo se lleve a cabo con estos dos dispositivos aunque la presencia de otro dispositivo puede ser reconocida y se pueden hacer conexiones con el mismo, deshabilitando la autenticación.

Se utilizó el lenguaje de programación Visual C++ 6.0, con el cual se programó la aplicación que se describe más adelante y que se encuentra instalada tanto en la PC de escritorio como en la *Laptop*. Dicha aplicación está programada de manera tal que cualquier dispositivo que inicie el procedimiento de conexión se convierta en CLIENTE, dando lugar a que el otro dispositivo sea el SERVIDOR; sin embargo en cualquier momento pueden intercambiar papeles, terminando la conexión e iniciado otra nueva con los papeles invertidos.

6.3 Aspectos de programación

Para la programación de la aplicación, es necesario establecer primero cada uno de los procedimientos necesarios para la comunicación entre el *host* (PC) y el módulo *Bluetooth*.

6.3.1 Interfaz del servidor COM

Para comunicarse con la pila de protocolos, la aplicación programada debe funcionar como un cliente COM, ya que el PC Referente Stack es un servidor COM.

Una forma de simplificar esto es haciendo llamadas directamente a la pila de protocolos de cada aplicación. Para hacer esto, se deben incluir algunos archivos en la aplicación, los cuales son 'bt_interface.lib', 'serverevents.cpp', serverevents.h', y todas las cabeceras que vienen en el disco de instalación del *kit*. El archivo 'serverevents.cpp' crea un *shell* de cliente COM para posibilitar la comunicación con la pila de protocolos.

6.3.1.1 Métodos específicos del servidor COM

Todos los métodos implementados están relacionados con el proceso de comunicación en la pila de protocolos. Dichos métodos se enlistan en la tabla 6.1.

Como se puede ver, el proceso consiste en enviar un mensaje con los parámetros correspondientes, para especificar: que se desea realizar una petición a la pila de protocolos, la cual contesta con un evento que contiene una respuesta con los parámetros necesarios para aceptar, o rechazar, la petición realizada.

Método del servidor COM	Comentarios
EnviarMensaje(mensaje, IDReceptor)	Método utilizado para enviar un mensaje.
MensajeRecibido(mensaje, IDTransmisor)	Evento que notifica al cliente que un mensaje llega.
ObtenerIDProcesoActual()	Método que regresa el identificador del proceso actual en la pila de protocolos.
ObtenerIDProcesoExterno(nombreproceso)	Método que regresa el identificador del proceso especificado

Tabla 6.1 Métodos del servidor COM.

En los capítulos 3,4 y 5, utilizamos una serie de nombres para los comandos, mensajes, eventos y respuestas. Dichos nombres difieren un poco de los utilizados en la versión del *software* que se utilizó, pero las funciones son las mismas.

6.3.2 Proceso de conexión

A continuación se muestran las secuencias que se siguen en la aplicación para establecer la comunicación para la red de área personal. Se han hecho tres etapas que agrupan los diferentes procesos que se siguen para el funcionamiento del programa: inicio, búsqueda y conexión.

6.3.2.1 Inicio

La primera etapa consiste en el reconocimiento del dispositivo *Bluetooth* por el *Host*, para así iniciar y registrar las capas utilizadas en dicha etapa.

Debido a que existe una interfaz utilizada para la comunicación entre el módulo y el *Host*, es necesario iniciar una capa de interfaz serial. A este componente se le envían mensajes dependiendo de la tarea que se esté ejecutando.

El componente de la capa de interfaz serial, o SIL (*Serial Interface Layer*), ofrece la posibilidad de configurar el tipo de comunicación a usarse, ya sea por el puerto serial, o por el puerto USB. El diagrama se muestra en la figura 6.4, y muestra los mensajes y eventos que envuelven a este proceso. Cabe aclarar que para todos los mapas que se mostrarán, se identifica el componente respectivo con letras mayúsculas, seguido del mensaje en letras minúsculas, o del evento en letras mayúsculas.[3]

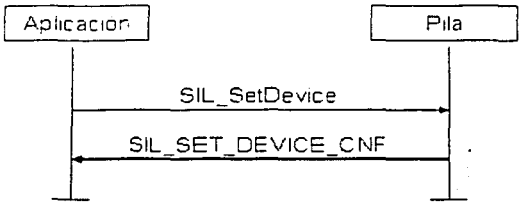


Figura. 6.4. Llamadas al componente SIL.

Para iniciar la comunicación, hacemos uso del SIL_*SetDevice* para establecer el tipo de interfaz a usar, la cual será el puerto USB. En el caso de que una llamada a cualquier componente provoque una respuesta positiva, se obtendrá un evento CNF; de lo contrario, se obtendrá un evento CNF_NEG.

El siguiente paso es el uso de la función HCI_*ReqConfigurePort* que prepara la configuración del puerto que se utilizará, en este caso el USB (figura 6.5).

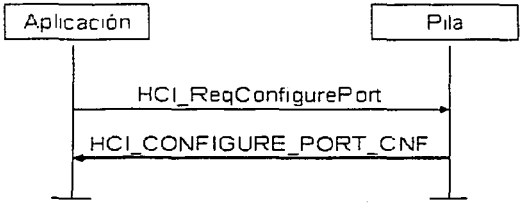


Figura. 6.5. Llamadas al HCI para configurar el puerto.

[3] Referirse al documento [18], para una lista detallada de todos los mensajes, eventos, parámetros, tipos de datos y estructuras utilizadas.

Una vez hecho esto, se comienzan las capas de RFCOMM y de Descubrimiento de Servicios llamando al COM_*ReqStart* y SDC_*ReqStart*, respectivamente. La capa RFCOMM es necesaria para la comunicación y la capa SD es necesaria para que otros dispositivos *Bluetooth* encuentren los servicios que contenga.

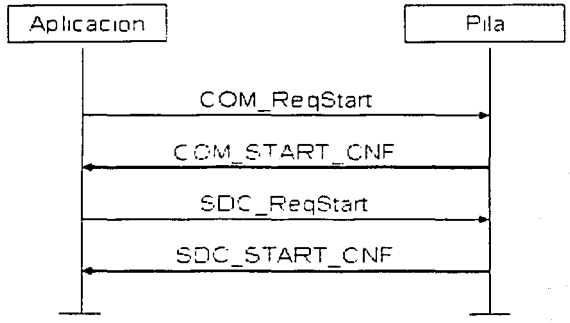


Figura. 6.6. Inicio de las capas RFCOMM y SD.

A continuación se prepara la funcionalidad de la capa HCI, llamando diversas funciones para el establecimiento de la configuración necesaria para cumplir los requisitos del *perfil* elegido (figura 6.7).

Las funciones que se utilizan se describen a continuación.

- HCI_*ReqWrEncryptionMode*. Habilita, o deshabilita, el cifrado (se habilita en esta aplicación).
- HCI_*ReqWrAuthMode*. Especifica si se requiere, o no, la autenticación (no se requiere en este caso, pero sí se implementa).
- HCI_*ReqConnectTmo*. Establece el parámetro de tiempo de espera para una conexión.
- HCI_*ReqWrPageTmo*. Establece el valor de tiempo de espera para el estado de *page*.
- HCI_*ReqWrVoiceSettings*. Establece los parámetros de voz (aunque no se utiliza en esta aplicación, es una función requerida).
- HCI_*ReqWrCod*. Establece la clase de dispositivo, del dispositivo local.
- HCI_*ReqWrName*. Cambia el nombre del dispositivo local *Bluetooth*.
- HCI_*ReqWrScanEnable*. Habilita el estado de *scan* (en la aplicación, tanto el *page scan*, como el *inquiry scan*, se habilitan).

Para poder responder a eventos, la aplicación necesita registrarse con el componente SCM (*Service Connection Manager*), por lo que se recurre a la función SCM_*ReqRegister*, la

cual es llamada en dos ocasiones. La primera es para registrar la aplicación como SCM_SECURITY_HANDLER, para aceptar, o rechazar, enlaces de datos y manejar los códigos de identificación durante el *apareamiento*; y la segunda sirve para registrar la aplicación como SCM_MONITOR_GROUP, para asegurar que la aplicación será notificada cuando se crean, o se destruyen, los enlaces.

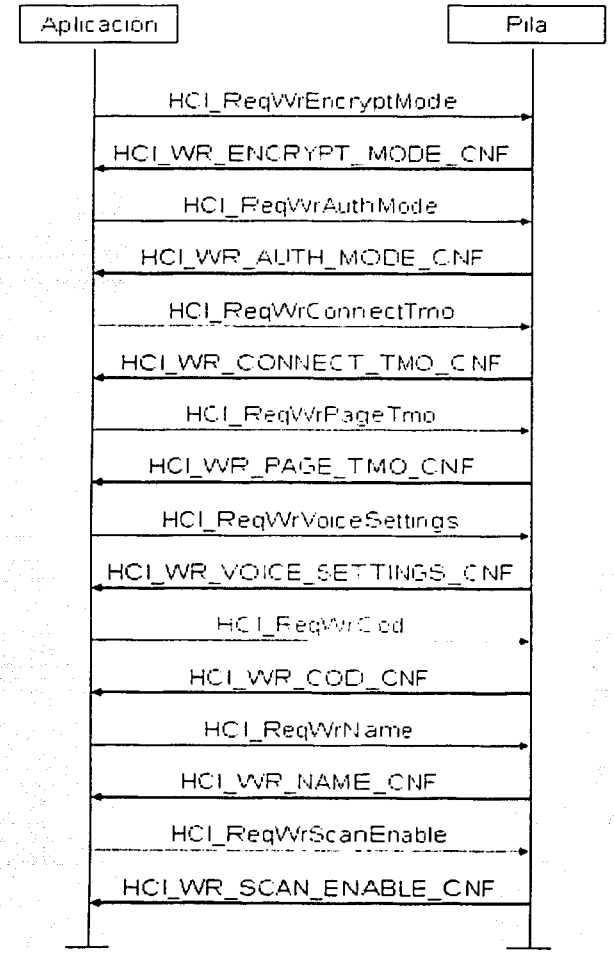


Figura. 6.7 Configuración de parámetros en la capa HCI.

Ahora se inicia el componente SDS (*Service Discovery Server*), el cual es fundamental para todos los modelos de uso que se le den a la aplicación. El protocolo de descubrimiento de servicios (SDP), hace posible a las aplicaciones el descubrimiento de qué servicios se encuentran disponibles y también sirve para determinar las características de dichos servicios, mediante una conexión L2CAP.

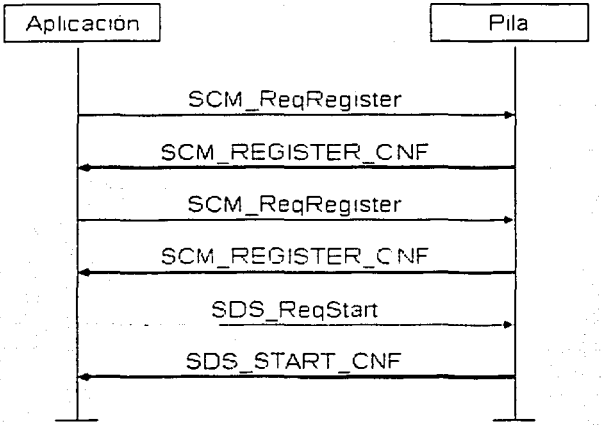


Figura. 6.8. Inicio de las capas SCM y SDS.

El siguiente paso es llenar el componente DBM (*Data Base Manager*), pues este es un mecanismo que proporciona acceso a la información de los servicios. En sí, el DBM contiene interfaces que permiten al usuario mantener su información.

Entonces, junto con todos sus parámetros, se comienza por agregar registros de servicios. Para esto se utiliza el DBM_*ReqRegisterService*, en el que se determina qué tipo de base de datos se utilizará para el registro de servicios: la base de datos del servidor, y/o la base de datos para registrar los servicios de los dispositivos remotos. Para el caso tratado en esta tesis, se necesitan las dos bases de datos, pero se comienza registrando la base de datos del servidor con los parámetros que se describen a continuación (figura 7.9).

Se comienza por llenar los atributos del servicio en la base de datos, por lo que se hace uso del DBM_*ReqAddDescr* para escribir la lista correspondiente al identificador de la clase de servicio a registrar; enseguida se usa el DBM_*ReqAddAttr* para escribir la lista correspondiente a los atributos de los descriptores del protocolo utilizado; y finalmente, otro DMB_*ReqAddAttr* para escribir los atributos del nombre del servicio.

Para registrar un servidor (aplicación) en el componente RFCOMM, se hace uso del COM_*ReqRegister*, para hacer posible que reciba indicaciones y eventos. Con este procedimiento reservamos un canal para el servidor.

Enseguida se debe llenar el registro de servicio en la base de datos del DBM, con parámetros específicos de protocolo del componente RFCOMM, así un cliente remoto que quiere conectarse al servidor, utiliza esta información. La función es COM_*ReqFillPdl*.

Ahora se hace una petición para saber cuál es la dirección *Bluetooth* del dispositivo local, por lo que se utiliza el HCI_*ReqLocalAddress*.

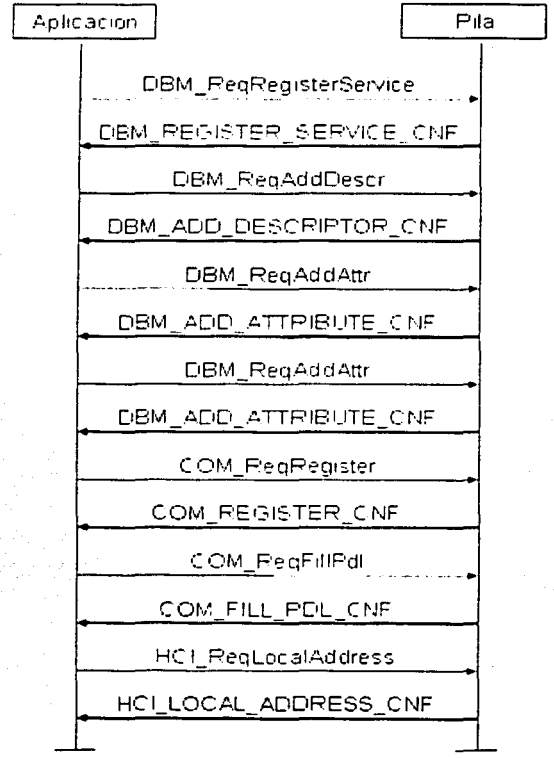


Figura. 6.9. Registro en el DBM, y petición de dirección *Bluetooth*.

Hasta este punto, el dispositivo *Bluetooth* local está completamente listo para usarse, ya que se han registrado e iniciado cada una de las capas correspondientes para el uso de las mismas, de acuerdo al perfil que se está usando, que es el de *PROFILE SERIAL*. Es importante recordar que dependiendo del perfil, se inician las capas de cierto modo. En sí

no hay una regla, o pasos, a seguir para iniciar los componentes, pero dependiendo de las necesidades, hay componentes necesarios que deben registrarse antes que otros.

En la aplicación, se decidió registrar dos servicios, por lo que se repiten los procedimientos de la figura 6.9, a excepción del HCI_*ReqLocalAddress*. Existe un manejador por cada registro de servicio en la base de datos, al igual que hay un manejador por cada conexión que se haga sobre RFCOMM; de este modo se puede hacer distinción de las funciones para cada servicio.

6.3.2.2 Búsqueda

El estándar Bluetooth permite a los dispositivos hacer búsquedas en su entorno para saber ¿qué dispositivos/servicios se encuentran disponibles? y así poder hacer uso de los mismos.

En la etapa de inicio, se configuran los dispositivos para que sean capaces de estar en los estados de *page scan* y de *inquiry scan*. En esta etapa, se ponen a los dispositivos en el estado de *inquiry* para que puedan encontrar dispositivos en su entorno, y como tienen habilitado el estado de *inquiry scan*, también estarán dispuestos a ser descubiertos ellos mismos por otros dispositivos.

Hay dos posibilidades para hacer el uso del *inquiry*, por ejemplo, cuando se quiere buscar dispositivos durante cierto periodo de tiempo y después del mismo se decide no hacerlo; y cuando se desea estar buscando regularmente durante periodos de tiempo hasta encontrar un dispositivo. En esta aplicación se hace uso de los dos tipos, por lo que es conveniente saber el funcionamiento de los mismos.

Para el caso de que sólo se quiera hacer un *inquiry* que durará cierto tiempo, se hace uso del HCI_*ReqInquiry*. El funcionamiento, de acuerdo a la figura 6.10, es el siguiente:

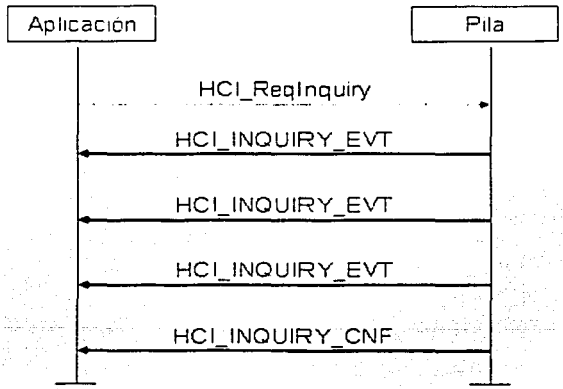


Figura. 7.10. Proceso de *Inquiry*.

La aplicación hace una petición para que inicie el proceso de *inquiry*, estableciendo un tiempo para ello y un número de respuestas que quiere obtener (número de dispositivos a encontrar). El evento HCI_INQUIRY_EVT indica a la aplicación que ha recibido una respuesta. El mensaje HCI_INQUIRY_CNF se genera cuando el tiempo ha expirado, o cuando ya se recibió el número de respuestas deseadas.

Para el caso de que se quiera un *inquiry* periódico, el cuál se mantendrá siempre activo hasta que se desee, se usa el HCI_*ReqPeriodicInquiry*. El funcionamiento, de acuerdo a la figura 6.11, es el siguiente:

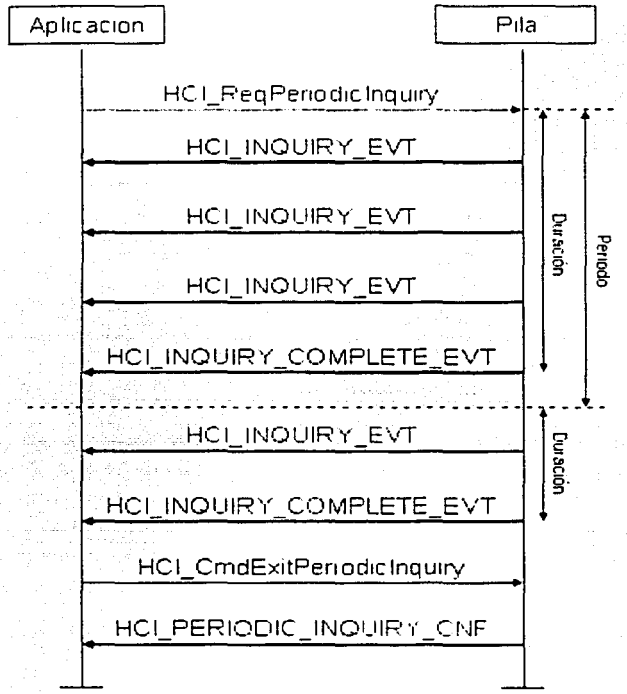


Figura. 6.11. Proceso de *Inquiry* Periódico.

En este caso, cada que termina un periodo de *inquiry*, se recibe el evento HCI_INQUIRY_COMPLETE_EVT. Para terminar este proceso, la aplicación puede hacer uso del HCI_*CmdExitPeriodicInquiry*, lo que hará que el dispositivo envie el mensaje HCI_PERIODIC_INQUIRY_CNF, indicando que se ha finalizado el proceso de búsqueda periódica.

Para el caso de la aplicación, al sólo haber un dispositivo al cual se va a encontrar, sólo es necesario recibir un solo HCI_INQUIRY_EVT, el cual contendrá información del dispositivo encontrado como lo es su dirección, sus modos: *PageScanRep*, *PageScanPeriod* y *PageScan*, la clase de dispositivo y la diferencia entre el reloj propio y el reloj del dispositivo remoto, es decir, el *offset*.

A este nivel, ya se tienen los datos que servirán para identificar al dispositivo remoto que contestó, por lo que en la aplicación se hacen los procedimientos necesarios para guardar los mismos y hacer uso de ellos cuando se quiera hacer una petición de conexión con el dispositivo remoto.

Finalmente se hace una petición para saber el nombre del dispositivo remoto, con los datos obtenidos y utilizando el HCI_*ReqRemoteName*.

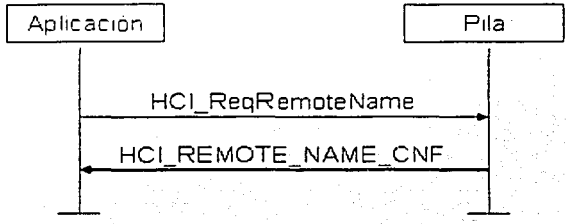


Figura. 6.12. Lectura del nombre del dispositivo remoto.

6.3.2.3 Conexión

En esta etapa ya se puede comenzar el proceso de conexión puesto que se tienen los datos necesarios para la misma. Entonces, para conectarse a un dispositivo específico, se entra en el estado de *page*, y como el otro dispositivo tiene habilitado el estado de *page scan*, entonces, la conexión es posible.

Para formar la *piconet*, se necesita de al menos dos dispositivos. Uno de ellos actuará como maestro de la misma, mientras que el otro será un esclavo. En la aplicación se conoce como SERVIDOR y CLIENTE y como ya se ha visto, se pueden tener múltiples clientes comunicándose con múltiples servidores, utilizando canales de multiprotocolos, y compartiendo el mismo enlace de datos.

Una de las funciones de la aplicación es la de actuar como manejador de seguridad, o *security handler*, teniendo como función la de aceptar, o rechazar, peticiones de enlace, además de que es el responsable de configurar los parámetros de configuración de cada enlace.

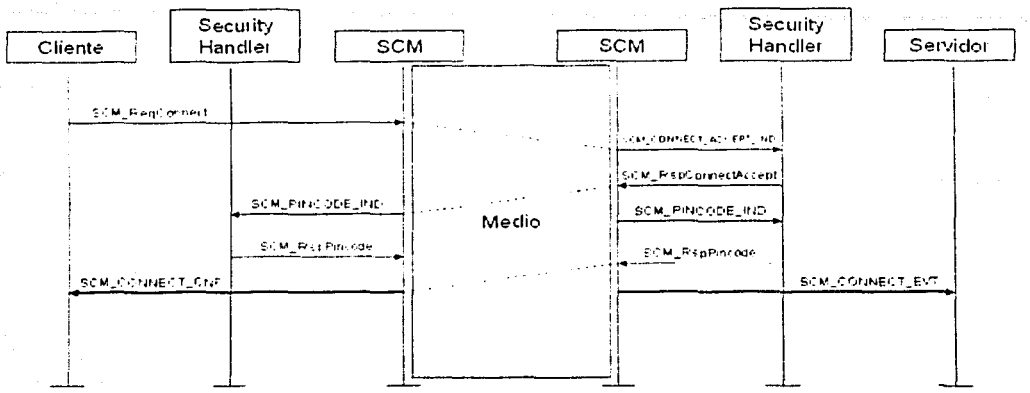


Figura. 6.13. Petición de conexión con el SCM.

Como se puede ver en la figura 6.13, al utilizarse la autenticación, se necesita enviar un SCM_*ReqConnect* para que en el otro dispositivo se reciba el indicador SCM_CONNECT_ACCEPT_IND, el cual va a responder con un SCM_*RspConnectAccept*, para que ambos dispositivos reciban el indicador SCM_PINCODE_IND, el cual informa que se está utilizando la autenticación y que se requiere de un código igual en ambos dispositivos, por lo que responden con un SCM_*RspPincode*. Finalmente el dispositivo que pidió la conexión recibe un mensaje SCM_CONNECT_CNF y el dispositivo que acepta la conexión, un evento SCM_CONNECT_EVT.

Hay que recordar que con el SCM se establecen parámetros como: tipo de paquete a utilizar, los modos *PageScanRep* y *PageScan*, el *offset*, y si se aceptará, o no, un cambio de papeles.

El siguiente procedimiento es el descubrimiento de servicios. Para ello, se necesita hacer un SDC_*ReqConnect*, para así poder hacer búsquedas en la base de datos del dispositivo remoto.

Luego, se empiezan las búsquedas con SDC_*ReqServiceSearch*, el cual contiene los parámetros necesarios para localizar registros de servicios y obtener sus manejadores, de acuerdo a ciertas características que dichos servicios deben cumplir.

El dispositivo remoto responde, y el dispositivo local recibe el mensaje SDC_SERVICE_SEARCH_CNF, el cual contiene una lista de manejadores de los registros de servicios que cumplen con las características dadas en la petición de búsqueda. Entonces, ya se tienen los manejadores, por lo que para cada servicio encontrado se usará su respectivo manejador para buscar sus atributos, usando SDC_*ReqServiceAttr* se obtienen valores específicos de dichos atributos, ver figura 6.14.

Una vez reunidos los atributos necesarios, se puede dar por terminada la conexión con el componente SDC, por lo que se recurre al SDC_*ReqDisconnect*.

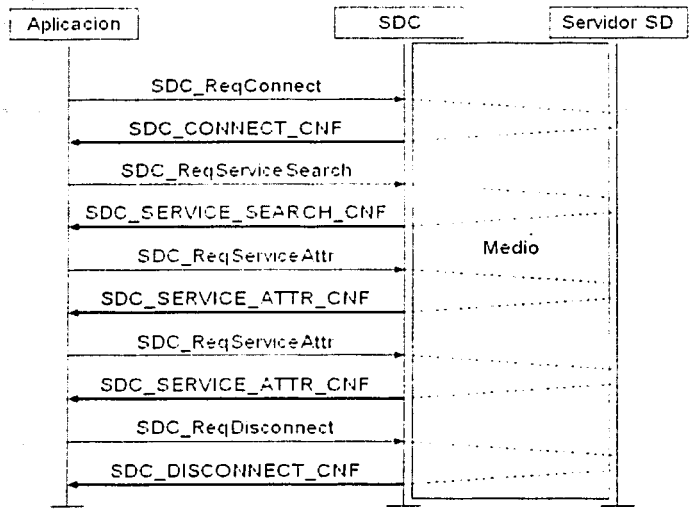


Figura. 6.14. Descubrimiento de servicios con el SDC.

Toda esta información se debe guardar en la base de datos de descubrimiento de servicios, por lo que en esta ocasión se volverá a hacer uso del DBM_*ReqRegisterService*, solo que ahora, se tratará de la base de datos en donde se registrarán los servicios descubiertos (figura 6.15).

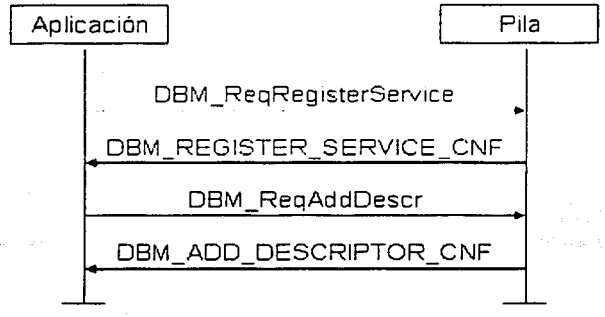


Figura. 6.15. Registro de los servicios encontrados en el dispositivo remoto.

Esto último se repetirá dependiendo de cuántos servicios se hayan encontrado, por lo que se tendrá completamente toda la información en los dos dispositivos y enseguida ya podrán conectarse.

Para establecer el canal, se hace una conexión con COM_*ReqConnect*. De acuerdo a la versión de la pila de protocolos que se esté usando, los mensajes COM_PAR_NEGO_IND y COM_*RspParNego* pueden ser necesarios, o no; para este caso sí lo son, por lo que en ellos se indica el control de flujo basado en créditos, el cual no se usa en esta aplicación (por lo que se deshabilita dicho valor) (figura 6.16).

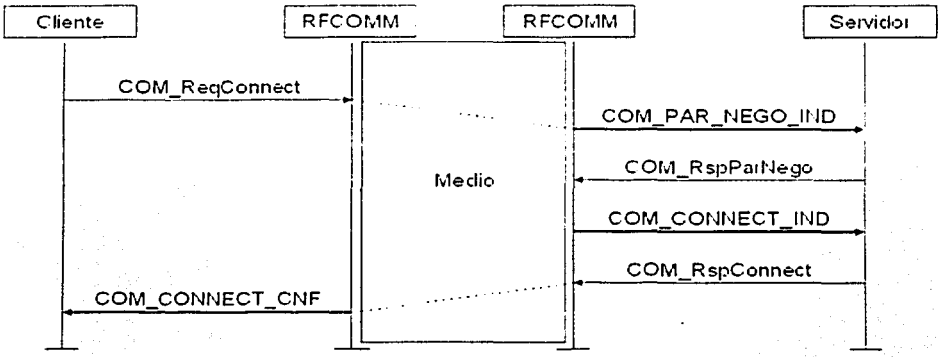


Figura. 6.16. Establecimiento de una conexión RFCOMM.

Si el dispositivo SERVIDOR acepta la conexión, entonces el dispositivo CLIENTE recibe un mensaje de COM_CONNECT_CNF.

Así se da por terminado todo el proceso de intercambio de datos y mensajes, para el establecimiento de conexión en nuestra piconet.

6.3.3 Intercambio de datos

El objetivo de la aplicación realizada es el de poder realizar intercambio de datos entre dos dispositivos: una PC de escritorio y una *Laptop*. Para dicho intercambio, se hace uso de dos servicios:

- Transferencia de Archivos.
- Intercambio de Mensajes.

Cada servicio ha sido registrado en la base de datos de los dispositivos, de esa manera cuando establezcan una comunicación, harán uso de ellos. Se creará una conexión RFCOMM individual para cada uno de los servicios.

A nivel RFCOMM, el intercambio de datos se hace a través de peticiones y mensajes, los cuales se ilustran en la figura 6.17. Como los dos servicios que se utilizan tienen conexiones en RFCOMM, utilizan el mismo modelo para el intercambio de información.

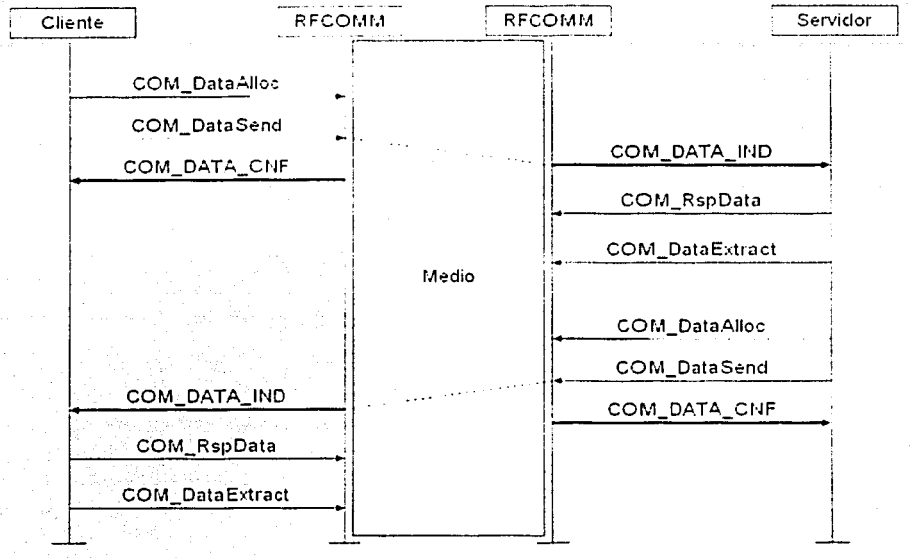


Figura. 6.17. Intercambio de datos mediante RFCOMM.

El mensaje COM_*DataAlloc* reserva memoria del sistema operativo con la estructura correspondiente para el envío de datos, especificando el número de bytes que se quieren reservar. Cuando dicha memoria se quiere liberar, entonces se usa el COM_*DataFree*.

Se envía la información mediante el COM_*DataSend*, especificando a qué conexión va dirigido para así hacer una distinción del tipo de datos que se está mandando.

El dispositivo remoto recibe un indicador COM_DATA_IND, para enterarse de que ha recibido datos. La aplicación debe responder a este mensaje lo más pronto posible. Para aceptar, o rechazar, los datos que han llegado, el dispositivo envía una respuesta COM_*RspData*, e inmediatamente, en caso de haber aceptado, extraer la información de la cabecera y los datos que llegaron en el COM_DATA_IND, con el comando COM_*DataExtract*.

La aplicación que recibió los datos puede responder al dispositivo que se los envió confirmándole que el paquete se ha recibido correctamente. Por lo que se repetirá el proceso, pero en sentido contrario.

Es cuestión de la aplicación la forma en la que se procese la información recibida, por lo que depende del servicio, el uso que se le dé a la misma.

6.3.3.1 Transferencia de Archivos

Para este servicio, los datos a transmitir/recibir corresponden a un archivo con la estructura que maneja el sistema operativo *Windows*.

Para procesar la información que llega y/o adecuarla para enviarla, se hace uso de una serie de pasos que denominamos control de protocolo de transferencia de archivos, o CPTA. Dichos pasos son los siguientes:

Paso	Transmisor	Receptor
1	Se envía un indicador NUEVO_ARCHIVO.	
2		Después de haber seleccionado la ruta en donde se guardará el archivo, se envía NUEVO_ARCHIVO_CNF.
3	Se envía un indicador INFORMACION_ARCHIVO.	
4		Se comienza el modo de recepción de información de archivo y se envía INFORMACION_ARCHIVO_CNF.
5	Se abre el archivo y se envía el nombre del archivo.	
6		Se crea el archivo y se envía un INFORMACION_ARCHIVO_CNF.
7	Se envía un indicador DATOS_ARCHIVO.	
8		Se comienza el modo de recepción y se envía DATOS_ARCHIVO_CNF.
9	Se lee un bloque de datos del archivo y se envía la información.	
10		Se escribe la información del archivo y se confirma enviando un DATOS_ARCHIVO_CNF.

	Se repiten los pasos 9 y 10 hasta que todo el archivo es leído (enviado).	
11	Se envía un FIN_ARCHIVO después de haber cerrado el archivo	
12		Se envía un FIN_ARCHIVO después de haber cerrado el archivo.

Tabla 7.2. Pasos a seguir en el control de protocolo de transferencia de archivo.

6.3.3.2 Intercambio de Mensajes

Para este servicio, los datos a transmitir/recibir corresponden a texto en código ASCII, por lo que no se necesita de un manejo especial, o de un procesamiento complejo, simplemente mostrando la información que llega y enviando la que se escribe tanto de un lado, como del otro.

6.4 Descripción de la aplicación

La aplicación realizada esta basada en el establecimiento de una piconet *Bluetooth*, en donde dos dispositivos intercambiarán información sin el uso de cables. Dichos dispositivos son una PC de escritorio y una PC portátil y fungirán como SERVIDOR y CLIENTE, respectivamente.

De acuerdo con las características de la red implementada, no es necesario el uso de la técnica de cifrado, pues no es necesaria para el modelo que se ha utilizado, pero es conveniente implementarla para proveer las características de seguridad correspondientes. El modo de autenticación si se habilita y provee seguridad, pero puede deshabilitarse si así se desea

La aplicación es un programa hecho con el lenguaje Visual C++ 6.0 y está basada en la programación del sistema operativo *Windows*. Se encuentra instalada en ambos dispositivos y aunque hemos definido un papel para cada uno, pueden tomar el otro papel si así se desea.

Debido a que sólo se cuenta con dos dispositivos, la *piconet* estará formada entonces por un maestro, que será conocido como el CLIENTE, ya que es el dispositivo que inicia el proceso de conexión; y un esclavo, que será conocido como SERVIDOR, pues es el dispositivo que acepta la petición de conexión.

La *piconet* formada es una red *ad hoc* propuesta en un escenario en donde se requiera obtener información vía inalámbrica, desde un dispositivo que actuará como SERVIDOR, por medio de un dispositivo que necesite tal información y que será el CLIENTE. Ambos

dispositivos se mantendrán en el estado de búsqueda periódica, para que una vez que se encuentren en el rango de comunicación, que en este caso es de 10m por tratarse de dispositivos clase 2, se encuentren y comiencen la negociación de la conexión.

Una vez que los dispositivos se hayan conectado pueden comenzar a intercambiar información, la cual no se interrumpirá aún cuando los mismos se muevan, o no tengan una línea de vista, y mientras no se salgan del rango de operación.

6.4.1 Interfaz gráfica

La aplicación tiene una interfaz gráfica principal, en la que se desarrolla el proceso de búsqueda y conexión, así como el uso del servicio de transferencia de archivos. Dicha interfaz se encuentra dividida en áreas como se muestra en la figura 6.18.

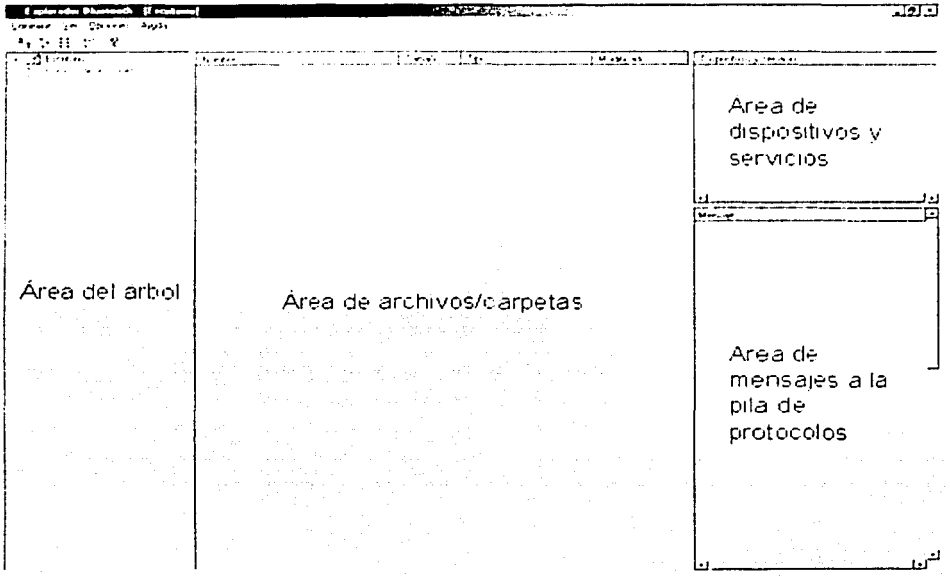


Figura. 6.18. Interfaz gráfica de la aplicación.

En el área del árbol se presenta el directorio de archivos de la PC correspondiente, además de un elemento identificado como "Entorno de Red *Bluetooth*", el cual permite buscar dispositivos *Bluetooth* que se encuentren en el área de operación.

En el área de archivos/carpetas se muestra en detalle el contenido de una carpeta, listando los archivos y carpetas que contenga, y permitiendo seleccionar archivos en el caso de que se quiera transmitir, o carpetas en el caso de que se quiera recibir.

En el área de dispositivos y servicios se muestra una lista de los dispositivos que se hayan encontrado en el proceso de *búsqueda* así como los servicios que se encuentren disponibles en los mismos.

En el área de mensajes a la pila de protocolos se muestra una lista de: mensajes, eventos, indicaciones y comandos, que se ejecutan desde y hacia la pila de protocolos del dispositivo *Bluetooth* local.

La barra de menús contiene lo siguiente:

- **Conexión**. Se puede seleccionar entre conectarse, o desconectarse, con un dispositivo dependiendo del caso, así como terminar la aplicación. También permite finalizar el proceso de búsqueda si se encuentra en ejecución.
- **Ver**. Se puede seleccionar si se quiere ver, o no, las barras de estado y herramientas.
- **Opciones**. Aquí se pueden configurar algunas opciones de conexión, dentro de un cuadro de diálogo. Además se puede abrir la ventana de mensajes, en donde se hace uso del servicio de intercambio de mensajes.
- **Ayuda**. Muestra información de la aplicación y la versión de la pila de protocolos que se está utilizando.

Los parámetros de conexión que se pueden configurar, se muestran en la siguiente figura:

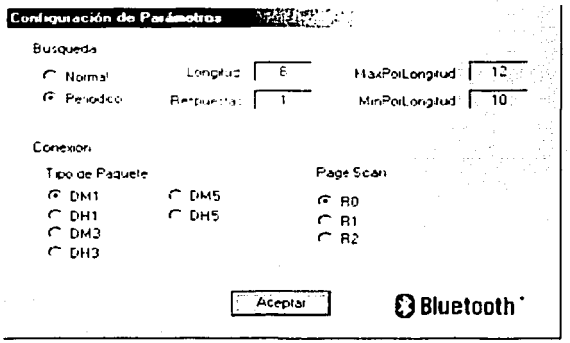


Figura. 6.19. Configuración de parámetros.

Para el conjunto de parámetros de búsqueda hay dos opciones: normal y periódico. Para el primero sólo se requiere especificar los valores longitud y respuestas, que establecen la máxima cantidad de tiempo antes de que la búsqueda termine y el máximo número de respuestas que se quieren recibir, respectivamente. Para el segundo, además

de los valores anteriores se requieren el MaxPorLongitud y MinPorLongitud, que establecen el máximo periodo, y el mínimo periodo, para la búsqueda periódica de dispositivos.

Para el conjunto de parámetros de conexión, se pueden establecer los valores de tipo de paquete y tipo de *page scan*, que son necesarios para el establecimiento de conexión vía SCM. De acuerdo a estos valores se hará la petición de conexión, por lo que sólo tienen efecto en el lado del CLIENTE y deben ser configurados antes de establecer la conexión.

Para el servicio de intercambio de mensajes, se cuenta con una ventana para su manejo. Dicha ventana se muestra en la figura 6.20.

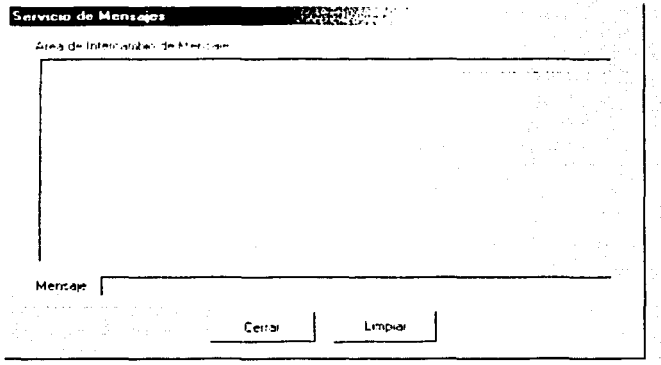


Figura. 6.20. Ventana de mensajes.

Como se puede apreciar es una ventana sencilla que muestra en el área de intercambio de mensajes una lista de los mensajes que se envían y reciben, además en el área de mensaje se puede ingresar el mensaje a transmitir.

La barra de estado que se muestra en la figura 6.21 muestra información importante sobre la conexión en la *piconet*: el papel que se tiene, ya sea CLIENTE o SERVIDOR, así como el nombre del dispositivo que lo identifica en la red; la acción que se esté realizando, y en el caso de transmisión de información, el número de bytes que se hayan transmitido; la dirección *Bluetooth* del dispositivo local y; la dirección *Bluetooth* del dispositivo remoto.

CLIENTE Nodo Móvil Bytes recibidos:151976 Dispositivo Local:0x00803716AF9C Dispositivo Remoto:0x00803716AF9B

Figura. 6.21. Barra de estado.

6.4.2 Proceso de conexión

Al iniciar la aplicación es necesario ponerle un nombre al dispositivo para identificarse en la red que se implementará. Dicho nombre será preguntado por el dispositivo que lo encuentre y facilitará su reconocimiento.

Cada dispositivo inicia un proceso de inicio, en donde se registran los parámetros necesarios y las capas de la pila de protocolos que están involucradas con el perfil que se está utilizando. La interfaz gráfica tiene la estructura mostrada en la figura 6.22.

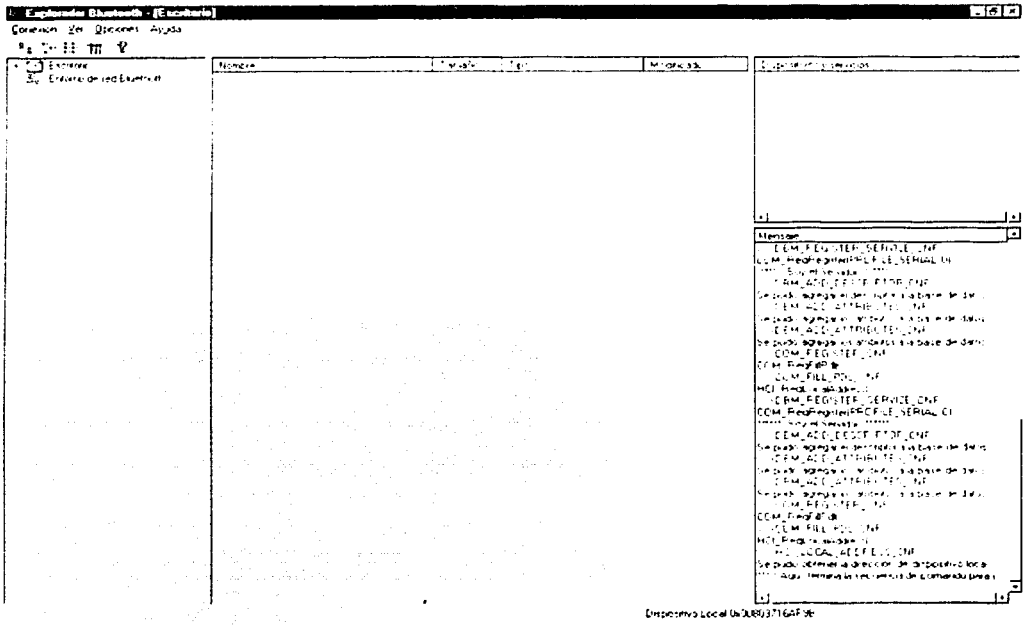


Figura. 6.22. Inicio de la aplicación.

La dirección del dispositivo local es única, y depende de cada dispositivo por lo que es el único parámetro que no se puede modificar, además de que con ella se forman los códigos necesarios para el establecimiento de un enlace vía RFCOMM.

Después, se comienza con el proceso de búsqueda y debido a que la aplicación ha sido programada para iniciar con búsqueda periódica, el dispositivo comienza con este proceso, mostrando cada vez que termina un periodo. Una vez que encuentra algún dispositivo, lo muestra en el área de dispositivos y servicios.

La aplicación elige por omisión el conectarse inmediatamente a los dispositivos que haya encontrado, por lo que este proceso se inicia de acuerdo a los parámetros de conexión que se establecieron previamente.

Una vez que se ha llevado a cabo la conexión, como se muestra en la Figura 6.23, se proporciona la lista de servicios disponibles del dispositivo en el área de dispositivos y servicios, y debido a que los dos dispositivos con los que se ha formado la *piconet* contienen los mismos servicios, entonces se puede comenzar a utilizar el servicio de transferencia de archivos.

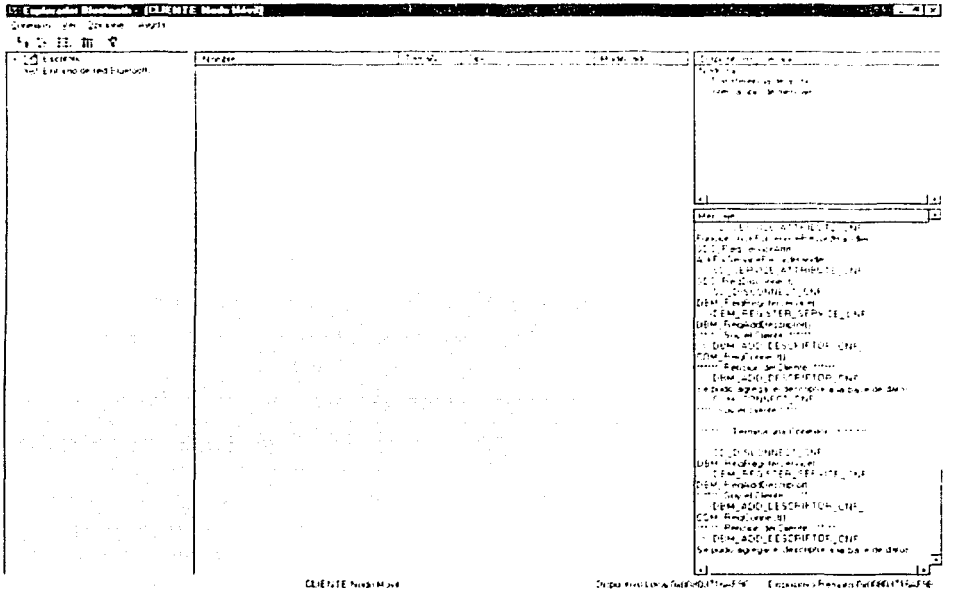


Figura. 6.23. Conexión establecida.

Para el servicio de transferencia de archivos, se elige un archivo del área de carpetas/archivos y se escoge la opción "Enviar archivo", entonces la aplicación se mantendrá en espera hasta que el dispositivo al que se le enviará el archivo haya aceptado, o no, (Figura 6.24). El dispositivo que recibirá el archivo indicará a la aplicación, en donde se debe elegir una carpeta del área de carpetas/archivos y se escoge la opción "Guardar archivo". Una vez que se haya aceptado, comienza la transferencia de información y en la barra de estado aparece el número de bytes transmitidos (Figura 6.25).

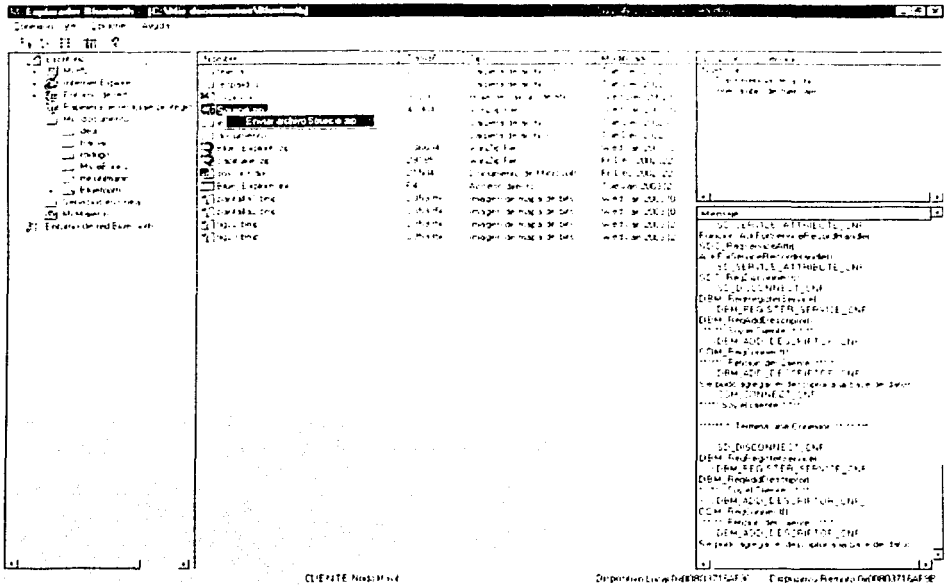


Figura. 6.24. Interfaz gráfica del CLIENTE al enviar un archivo

Cuando el proceso de transferencia termina, en ambas aplicaciones se notifica. Si se desea enviar un archivo en el orden opuesto, entonces se repite el procedimiento. Cabe señalar que el dispositivo CLIENTE, que fue el que inició la conexión, podrá enviar información con una tasa de transmisión mayor que la que utiliza el dispositivo SERVIDOR, porque el enlace es asimétrico y el CLIENTE puede usar paquetes de mayor tamaño.

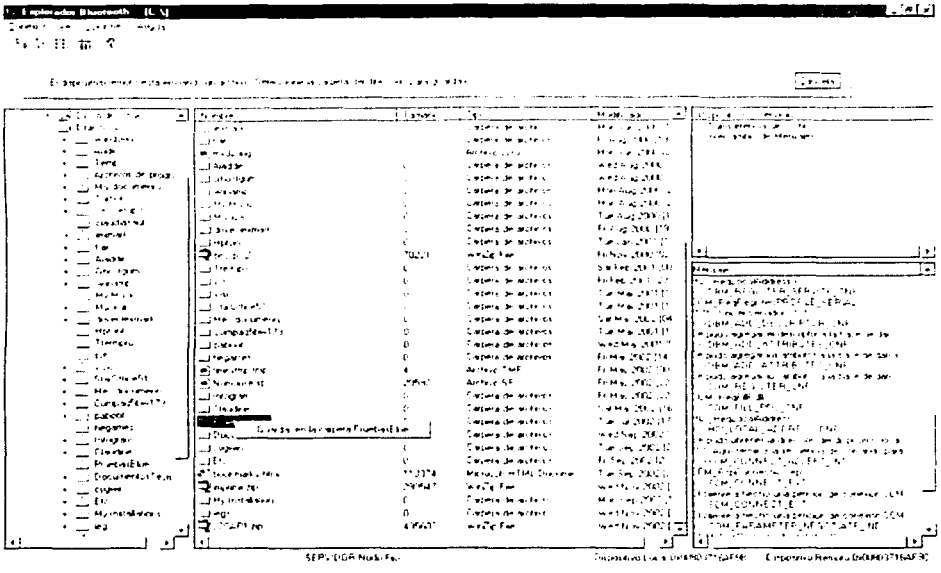


Figura. 6.25. Interfaz gráfica del SERVIDOR al recibir un archivo.

En cualquier momento se puede cancelar la recepción del archivo si así se desea, lo que implicaría notificar en ambas aplicaciones la realización de esta acción.

Para el servicio de intercambio de mensajes, basta abrir la ventana de mensajes en cualquiera de los dispositivos y escribir el mensaje a enviar. Una vez que se haya mandado, si en el otro dispositivo no se tiene abierta dicha ventana, se abre y se muestra el mensaje que se le ha enviado y si se desea, se puede responder.

Debido a que se estableció una conexión RFCOMM por separado para cada servicio, la transmisión de información puede ser simultánea, no interrumpiendo el funcionamiento de cualquiera de ellos.

Hay que recordar que aunque la comunicación por medio de la tecnología *Bluetooth* no es full dúplex, sí provee un mecanismo de división de tiempo para garantizar una tasa de transmisión óptima.

Según las especificaciones, la tasa máxima de transmisión puede ser de 723.2 kbps para un enlace asimétrico, utilizando paquetes de 5 ranuras de un lado y de una ranura del otro. Dicha tasa de transmisión máxima depende de los protocolos de capa superior y

de la configuración que se establezca para el envío de información. Para esta aplicación se hicieron varias pruebas con paquetes DH5 y un enlace asimétrico.

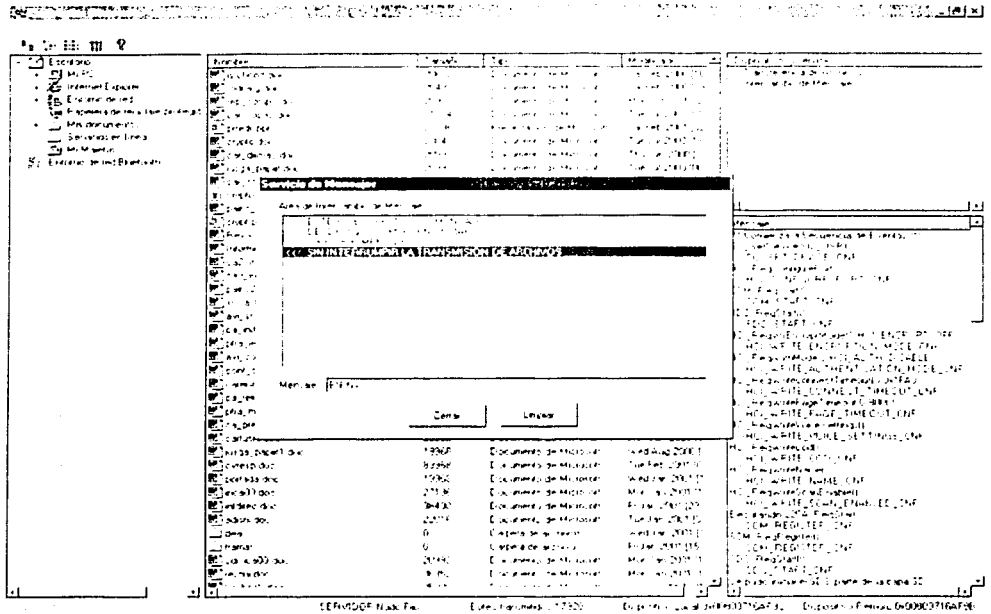


Figura. 6.26 Servicio de intercambio de mensajes.

Como la comunicación se lleva a cabo a través de un enlace vía RFCOMM, el máximo tamaño de un paquete es de 32772 bytes, debido al tamaño del paquete en L2CAP, siendo la carga útil de hasta 32767bytes. En la aplicación, para evaluar la tasa de transmisión se utilizó el servicio de transferencia de archivos, donde se utilizaron paquetes de datos de longitud 28672 bytes para transmitir; y paquetes de longitud 6 bytes para confirmar la recepción.

Además del procedimiento necesario en *Bandabase* donde, por cada paquete o paquetes enviados se debe confirmar si se recibieron correctamente, en el algoritmo de transferencia de archivos que utilizó la aplicación también se confirma por cada paquete de datos RFCOMM enviado, por lo que la tasa de transmisión se reduce. También debe tomarse en cuenta que hay un tiempo que involucra el procesamiento de la información a enviar/recibir, por ello, sólo se alcanzó una tasa de transmisión de 460kbps.

La distancia entre los dispositivos también afecta la tasa de transmisión, pues cuando más cercanos estén los dispositivos puede haber interferencia y errores en la transmisión,

lo que involucra más tiempo de transmisión; también cuando se alejan casi hasta el límite de cobertura, la tasa se reduce considerablemente por cuestiones de pérdida de paquetes.

El proceso de intercambio de información puede seguir de un lado a otro sin problemas de desconexión. Los dispositivos pueden seguir conectados aún cuando no se esté transmitiendo datos.

Debido a que el medio de alimentación de los módulos *Bluetooth* es el puerto USB de las respectivas computadoras, no se requiere ahorrar energía, por lo que los modos de bajo consumo no son utilizados aquí. Pero en caso de que se desearan habilitar, basta con usar los comandos HCI correspondientes.

Lo mismo sucede con el cambio de papeles entre MAESTRO y ESCLAVO, cuya única utilidad es esta aplicación sería la del intercambio de papeles CLIENTE/SERVIDOR, tomando en cuenta que es el CLIENTE quien posee la mayor tasa de transmisión.

El evento de desconexión se puede presentar si alguno de los dispositivos se sale de rango, o si se desea terminar la conexión. En el primer caso, la aplicación mostrará un mensaje como el que se muestra en la figura 6.27, y se volverá al estado de búsqueda; para el segundo, se comienza un proceso de cierre de las capas que se utilizaron en la conexión y ya no se hace un proceso de búsqueda, pudiéndose conectar nuevamente al dispositivo, ya que se tienen guardados sus datos.

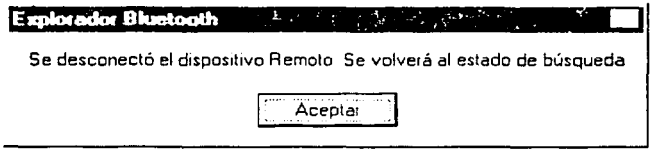


Figura 6.27. Mensaje de desconexión.

Como se puede ver, la *piconet* implementada cumple con las características de ser una red *ad hoc*, con una conexión transparente al usuario y una transmisión de información eficiente y segura. Seguimos paso a paso las especificaciones de la tecnología inalámbrica *Bluetooth* e hicimos uso de los *perfiles*, *capas*, *mensajes*, *peticiones*, y demás procedimientos para llevar a cabo el establecimiento de la red.

CONCLUSIONES

A continuación se presentan las conclusiones obtenidas con base en el estudio e implementación de una red inalámbrica de área personal tipo *Bluetooth*.

En el desarrollo de esta tesis y para el estudio de la tecnología *Bluetooth* analizamos la técnica de espectro disperso (véase Anexo C). Encontramos que es un esquema muy completo y eficiente para la transmisión de información evitando al máximo la interferencia y protegiendo la información para que no sea captada por otros dispositivos ajenos al sistema.

Utilizamos el lenguaje de programación Visual C++ 6.0 para programar la aplicación que establece la red de área personal entre los dispositivos *Bluetooth*. Aprendimos a manejar la comunicación entre los módulos y el sistema operativo, así pudimos intercambiar los mensajes correspondientes para el control de la comunicación entre los dispositivos.

Durante este estudio nos enfocamos en un nuevo concepto de redes: las redes *ad hoc*, que se presentan como una solución excelente para todas aquellas aplicaciones que requieran el establecimiento de una red que no necesite de una planeación: que sea móvil, y que se forme de manera espontánea y que tenga una topología que pueda cambiar.

Para este tipo de redes, *Bluetooth* es una tecnología viable, con grandes posibilidades de éxito y crecimiento, debido a que es capaz de brindar soluciones convenientes para las necesidades de comunicación a corta distancia. La conveniencia del uso de esta tecnología radica en las características siguientes: fácil manejo, compatibilidad mundial, tamaño, bajo costo, movilidad, desempeño y seguridad.

Se ha presentado esta tecnología no como un sustituto de las actuales redes inalámbricas, sino como un complemento para éstas y también para las redes cableadas, pues se han implementado protocolos lo suficientemente poderosos como para brindar una interconexión eficiente y un establecimiento de red con dispositivos heterogéneos pero compatibles

Para la implementación de una red de área personal, utilizamos dispositivos que nos brindaron la posibilidad de probar las funciones de las capas superiores de la pila de protocolos

Bluetooth. Decidimos desarrollar una aplicación que probara la facilidad que brinda la tecnología inalámbrica *Bluetooth*, aplicada en una red de área personal debido a la utilidad de un sistema que requiere el control por parte de un *host*, pues es en éste en donde se hace el procesamiento de la información para las aplicaciones.

La aplicación desarrollada sobre la red *Bluetooth* permitió demostrar la capacidad de ésta tecnología para el establecimiento dinámico y espontáneo de una red de área personal. Esta particularidad hace extremadamente atractiva a esta tecnología debido a que representa gran facilidad de operación para el usuario final. Por otra parte, se observó que *Bluetooth* presenta una mayor complejidad para el desarrollador, a medida que se incrementa la facilidad para el usuario.

Otro punto a destacar, es la robustez que presenta esta tecnología con respecto a los errores. Se puede afirmar que *Bluetooth* maneja de forma efectiva la información en presencia de ruido e interferencia, causantes de errores en el flujo de bits, a través de la técnica de espectro disperso. Las capas inferiores proveen una poderosa infraestructura que permite la detección de errores y el nivel de potencia de la señal que reciben, para pedir la retransmisión de información en caso de ser necesario.

Con respecto a la tasa de transferencia de datos, decimos que resulta adecuada para el enfoque que tienen las redes de área personal, dado que las posibles aplicaciones no requieren un gran ancho de banda. En la red implementada, hicimos uso de la transmisión de datos de forma asimétrica, para asegurar que se hiciera eficiente el uso del ancho de banda cuando un *cliente* hacia una petición de transmisión al *servidor*.

Aún cuando las especificaciones mencionan una tasa de transferencia de hasta 723.2 kbps con el esquema de enlaces asimétricos, nuestra implementación, debido al procesamiento que tiene que realizar con la información que envía y recibe, sólo alcanzó la tasa transmisión promedio de 460kbps. Hay que recordar que el uso de paquetes DH5 no garantiza que todos los paquetes van a ser recibidos sin la necesidad de que sean retransmitidos, a costa de una mayor tasa de transmisión.

Se observó que la tasa de transferencia se veía seriamente afectada por la distancia entre los módulos, es decir, a mayor distancia menor tasa de transferencia. Es importante remarcar, que en el límite del rango de operación, la tasa de transferencia desciende dramáticamente.

Una de las razones primordiales que hacen atractiva a esta tecnología, es la seguridad, y la garantía de que la información que se transmitirá estará protegida. Por esto, *Bluetooth* brinda una comunicación bastante segura, debido a la técnica de espectro disperso y al algoritmo de cifrado SAFER+. En conjunto, estos dos mecanismos proveen una gran confiabilidad, la cual es comparable con la que se encuentra en una red cableada inherentemente segura.

Los enlaces que establecimos nunca se interrumpieron, aún cuando los dispositivos se estuvieran moviendo; tampoco encontramos interrupción en la transmisión de datos o errores en la información decodificada.

Decidimos hacer uso del descubrimiento de servicios en los dispositivos, así pudimos configurar la aplicación para que el sistema *host-modulo Bluetooth* fuera versátil y se mostrara frente al otro sistema, ofreciéndole servicios para el intercambio de información. En este punto las aplicaciones que se implementen pueden ser muy poderosas y diversas.

En el escenario de la aplicación notamos la utilidad de los modelos de implementación. Hay muchas configuraciones que pueden desarrollarse, pero lo más importante es hacer que dichos modelos sean compatibles y operen correctamente.

Pensando en todos los aparatos electrónicos que pueden utilizar la tecnología *Bluetooth* para comunicarse con otros, dentro de un área relativamente pequeña, salta a la vista la importancia de un diseño poderoso, que tenga integrado todas las funciones de procesamiento necesarias para el fin deseado. El tamaño del módulo *Bluetooth* y su programación para que haga tareas específicas, juegan un papel muy importante para que este nuevo concepto de redes de área personal tenga éxito y sea parte de la vida cotidiana y productiva.

Aunque los algoritmos de cifrado y detección de errores se encuentran establecidos por el estándar, se podría utilizar un DSP para programar nuevos algoritmos y así fortalecer el nivel de seguridad. Todo esto requiere de un completo estudio y planeación de nuevas soluciones en criptografía.

Para el establecimiento de redes de más cobertura, que requieran poco ancho de banda, es necesario integrar algoritmos de enrutamiento que sean eficientes y que sean comparables en funcionalidad con los algoritmos utilizados en las redes actuales de gran capacidad.

Hay algunos detalles que necesitan perfeccionarse, como el hecho de que no se ofrezca un *hand over* que permita a los dispositivos decidir a quién o quiénes conectarse dependiendo de la potencia de la señal recibida. Esto motiva a seguir trabajando en el mejoramiento de la tecnología.

Aprovechando que hay muchas compañías que se encuentran trabajando activamente para hacer que las soluciones que actualmente se brindan sean más potentes, es factible participar en el desarrollo de nuevos procesos, algoritmos y aplicaciones, que nos ayuden en tareas para nuestra vida cotidiana y para proporcionar herramientas que se puedan aplicar en instituciones educativas y así aportar nuestros conocimientos y habilidades para aplicaciones que se puedan utilizar en cualquier parte del mundo.

El participar en el desarrollo y mejoramiento de tecnologías como la expuesta en este trabajo de tesis, produce una mayor eficiencia y beneficios a los miembros de una sociedad

como la nuestra y a todo el país, de aquí la importancia del desarrollo del presente trabajo y la consecución de los objetivos planteados inicialmente.

ANEXO A
BANDAS DE FRECUENCIAS, SERVICIOS Y APLICACIONES

A continuación se presenta una tabla con la nomenclatura convencional del espectro radioeléctrico, asi como también algunos servicios y aplicaciones [4].

Nomenclatura	Rango de frecuencias	Rango de longitudes de onda	Servicios y aplicaciones
ELF (Extremely Low Frequency)	Menor a 3 kHz	Mayor a 100 km	Comunicaciones militares y submarinas
VLF (Very Low Frecuency)	3 kHz – 30 kHz	100 km – 10 km	Trafico aéreo, sonar, navegación
LF (Low Frecuency)	30 kHz - 300 kHz	10 km – 1 km	Radio faros marítimos y aéreos
MF (Medium Frecuency)	300 kHz – 3 MHz	1 km – 100 m	Radiodifusión de onda media, radio faros aéreos y marítimos del sistema LORAN
HF (High Frecuency)	3 MHz – 30 MHz	100 m – 10 m	Radiodifusión de onda corta, comunicaciones para barcos y aviones, banda civil (CB), radio amateur
VHF (Very High Frecuency), banda métrica	30 MHz – 300 MHz	10 m – 1 m	Radiodifusion de TV y FM, comunicaciones para barcos y aviones, sistemas de *paging*, comunicaciones móviles terrestres, PMR
UHF (Ultra High Frecuency), banda decimétrica	300 MHz – 3 GHz	1 m – 100 mm	Radiodifusión de TV, sistemas celulares, radar aéreo, radar climático, telefonia inalámbrica, radiocomunicaciones para uso personal, *Bluetooth*.
SHF (Super High Frecuency), banda centimétrica	3 GHz – 30 GHz	100 mm – 10 mm	Enlaces de microondas, comunicaciones satelitales.

EHF (Extra High Frecuency), banda milimétrica	30 GHz – 300 GHz	10 mm – 1mm	Radar, radio astronomía, experimental
Ondas Micrométricas	300 GHz – 3 THz	1 mm – 100 micro m	Medición remota, comunicaciones láser, comunicaciones ópticas espaciales

ANEXO B
REDES BASADAS EN CÉLULAS

Las redes inalámbricas tradicionales (primera generación) que tratan de proveer cobertura a grandes áreas incrementando la potencia de transmisión de cada estación base sólo son capaces de dar servicio a un número limitado de suscriptores debido al ancho de banda usado. En estas redes el canal de radio asignado es retenido tanto como sea posible, a pesar de que el receptor se haya movido a otra área de servicio. Debido a que las fronteras de las áreas de servicio no están definidas con precisión, las áreas de servicio vecinas deben usar canales de radio diferente para evitar interferencia. Cuando la densidad de usuarios es elevada, esto resulta en una alta demanda de frecuencias, las cuales están muy restringidas debido a la escasez de espectro disponible.

La ineficiente utilización del espectro en estas redes y el incremento en el número de usuarios llevó al desarrollo de redes celulares.

Las redes celulares se basan en dividir el área total de operación en celdas o células, cada una de las cuales tiene una estación base que le proporciona el servicio. Cada estación base sólo puede usar un cierto número de canales de frecuencia del total disponible, los cuales se pueden volver a usar únicamente después de un espacio suficiente tal que evite interferencia con las celdas vecinas.

En las redes celulares, la baja transmisión de potencia de la estación base permite asignar frecuencias solamente en el área definida por los límites de la celda, de esta manera se permite que estas frecuencias se vuelvan a usar después de una distancia determinada.

Las células se representan generalmente como hexágonos regulares, pero debido a las condiciones topográficas y ambientales, esto es sólo una aproximación. En realidad las radioceldas tienen una forma muy irregular y están diseñadas para traslaparse con sus vecinas en un intervalo de 10 % a 15 %. Esto permite a los dispositivos móviles que operan cerca de la frontera de una célula, escoger con cual estación base comunicarse.

Las radio células están distribuidas en grupos (*clusters*), y cada frecuencia es usada una vez por grupo. Las celdas de un grupo vecino a otro pueden volver a usar todas las frecuencias ocupadas en el otro grupo, y por tanto es posible repetir el grupo. Debido a que los grupos

deben cubrir una cierta área total, el número de células agrupadas no es arbitrario, sino que tiene ciertos patrones. Dentro de un sistema completo de radiocomunicación asignado a una determinada banda, se observa que a menor cantidad de células en un grupo, se puede usar una mayor cantidad de canales de radio frecuencia por célula. Una reutilización de frecuencias a corta distancia incrementaría la interferencia co-canal entre grupos.

Este tipo de redes se usa en su mayoría en los sistemas móviles de telefonía digital, sin embargo, también se aplica en redes LAN y PAN inalámbricas, al igual que en sistemas satelitales (por ejemplo, la huella de un satélite LEO como parte de una constelación).

Cuando este tipo de sistemas se aplica a WLAN's y WPAN's, la red se compone de micro-células o *picocélulas* (*piconets*) respectivamente. Generalmente las LAN's y PAN's utilizan un enfoque no orientado a conexión para compartir el medio, mientras las redes de telefonía móvil y otras redes WAN inalámbricas utilizan un enfoque de conmutación de circuitos [7].

Para el caso de la telefonía móvil, el conjunto total de células del sistema cubrirá completamente el área de servicio. En estas redes las células están agrupadas en áreas, ésta agrupación se realiza conectando las estaciones base de cada célula mediante enlaces terrestres. Dentro de cada célula, los usuarios móviles se comunican con un transmisor/receptor dentro de la celda, cuando el usuario se aproxima a la frontera, la intensidad de la señal se desvanece, y el enlace pasa al transmisor/receptor de otra celda. Este proceso se llama *hand-off*. Dentro de la célula existe un número de canales que está disponible, los cuales generalmente están separados tanto en la frecuencia (FDM) como en el tiempo (TDM). Cuando un móvil inicia una llamada, se le asigna un canal dentro de la célula en la que se encuentra, este canal es utilizado dentro de la celda hasta que se alcanza la frontera, y entonces se asigna al usuario un nuevo canal disponible dentro de la otra celda.

El principal problema de las comunicaciones móviles es la variación de la intensidad de la señal a medida que los dispositivos en comunicación se mueven. Esta variación se debe a que la interferencia no es constante, por lo cual provoca desvanecimientos (*fading*). Los desvanecimientos son causa de rápidas variaciones en la intensidad de la señal. La solución normal para los desvanecimientos es el incremento en la potencia transmitida, pero en los sistemas celulares no es posible. En los sistemas basados en células se utilizan esquemas de modulación especiales para tratar con este problema. La presencia de desvanecimientos tiene serios efectos en la tasa de bits erróneos (BER), ésta se puede incrementar en el orden de miles de veces cuando está presente el fenómeno de desvanecimiento. Por ejemplo, si en un canal sin desvanecimiento se encuentra un BER de 10^{-6}, en el mismo canal con presencia de desvanecimiento se puede encontrar un BER de 10^{-3} [8].

Como se mencionó anteriormente, los sistemas celulares pueden utilizarse también en redes inalámbricas de área local y personal. En estos casos, es común que se use un sólo canal

con gran ancho banda. Lo anterior esta en concordancia con las altas tasas de transferencia de datos que los usuarios de LAN manejan.

En un ambiente de WLAN o WPAN, el tamaño de las células es generalmente más pequeño que en los sistemas de telefonía móvil. Cuando existe más de una celda por cuarto se habla de micro-células para WLAN y pico-células para WPAN. El proceso de *Hand-off* se lleva a cabo con base en la detección de potencia, o bien por medio de protocolos de solicitud/anuncio, apoyados en mecanismos de detección de potencia.

ANEXO C
LA TÉCNICA DE ESPECTRO DISPERSO

Un sistema de comunicaciones de espectro disperso es aquel que cumple con las siguientes caracteristicas:

- La señal ocupa un ancho de banda mucho mayor al mínimo necesario para ser transmitida.
- La dispersión se lleva a cabo mediante una señal de esparcimiento llamada señal de código, la cual es independiente de los datos.
- En el receptor se lleva a cabo la recuperación de la señal esparcida mediante la correlación de la señal recibida con una réplica sincronizada de la señal de esparcimiento usada en el transmisor.

Los esquemas de modulación estándar como modulación en frecuencia y PCM también esparcen el espectro de una señal de información, pero no se clasifican como sistemas de espectro disperso porque no cumplen con todas las condiciones mencionadas anteriormente.

C.1 Beneficios

C.1.1 Supresión de interferencia

El ruido blanco gaussiano tiene, por definición, potencia infinita distribuida uniformemente a lo largo de todas las frecuencias. Es posible la comunicación en presencia de este ruido con potencia infinita debido a que solamente las componentes de ruido, con potencia finita, que están presentes en el ancho de banda de la señal pueden interferir con ella. De lo anterior parte el concepto de supresión de interferencia de los sistemas de espectro disperso, y se puede explicar de la siguiente manera.

Considérese que varias señales (coordenadas) ortogonales, o dimensiones, están disponibles para un enlace de comunicación y que sólo un pequeño subconjunto de ellas se usa a la vez, en un determinado tiempo. Se asume que el dispositivo generador de interferencia (*jammer*), no

puede determinar el subconjunto de coordenadas que se encuentran en uso en un momento determinado.

Para señales de ancho de banda W y duración T, el número de dimensiones es aproximadamente 2*W*T [1]. Dado un diseño específico, los errores en el sistema son solamente función de $\frac{E_b}{N_0}$. Contra el ruido blanco gaussiano, con potencia infinita, el uso de la técnica de espectro disperso no ofrece mejoría en el desempeño. Sin embargo, el sistema tiene un buen desempeño cuando el ruido tiene potencia constante y finita, y se distribuye de manera independiente a la localización de la señal de información dentro del sistema de coordenadas. Existen dos opciones para el generador de ruido dadas las condiciones anteriores [1]:

1. Introducir ruido a todas las coordenadas de la señal con igual cantidad de potencia para cada una, lo que resulta en poca potencia de ruido para cada coordenada. En este caso se define la densidad espectral de ruido como $J_0 = \frac{J}{W_{ss}}$ donde J es la potencia del ruido y W_{ss} es el ancho de banda del sistema de espectro disperso.
2. Introducir ruido a algunas coordenadas de la señal, lo que resulta en mayor potencia de ruido para las coordenadas afectadas. Para este caso la densidad espectral J_D se modifica de la siguiente manera $\frac{J_0}{\rho}$ donde $(0 < \rho \leq 1)$ es la porción de la banda de espectro disperso elegida para ser afectada.

Se puede observar, para ambos casos, que un sistema de espectro disperso con un mayor número de coordenadas para la transmisión de la señal presenta una mejor efectividad ante la presencia de ruido con potencia fija finita e independiente de la señal. La presencia de ruido o interferencia con estas características no siempre es un acto intencional, también puede ser producto de fenómenos naturales y en ocasiones puede tratarse de interferencia propia causada por multitrayectorias de la señal recibida (*multipath*).

C.1.2 Acceso Múltiple

La técnica de espectro disperso se puede usar como técnica de acceso múltiple, la cual permite compartir un recurso de comunicaciones entre varios usuarios de manera coordinada.

En la técnica de acceso múltiple por división de código (CDMA), cada sistema usa una señal de código de espectro disperso diferente, lo que permite el uso de la misma banda de frecuencias al mismo tiempo para diferentes sistemas. Lo anterior es posible debido a que las trasmisiones se efectúan con códigos ortogonales que sólo pueden ser interpretados por el receptor que posea una réplica sincronizada del código usado, de esta manera es posible la

comunicación en la misma banda de frecuencias y al mismo tiempo sin interferir con transmisiones que utilicen otro código ortogonal diferente.

C.1.3 Otras Características

- Baja probabilidad de detección.
- Baja probabilidad de intercepción.
- Buena precisión en sistemas utilizados para la localización de objetos.

C.2 Modelo de rechazo de interferencia

En el modulador, la señal de información con una taza de R bits/s se multiplica por la señal de código de esparcimiento, la cual tienen una taza de símbolos de código de R_p chips/s. Considérese que el ancho de banda de la señal de información es R hertz y el de la señal de código es R_p hertz, donde $R << R_p$.

La multiplicación en el dominio del tiempo representa una convolución en la frecuencia; por lo tanto, si la señal de información es de banda angosta comparada con la señal de esparcimiento, el resultado de la multiplicación tendrá aproximadamente un ancho de banda igual a la señal de código. Es decir, se lleva a cabo un esparcimiento de la señal de información.

En el demodulador, la señal recibida se multiplica por una réplica sincronizada de la señal de código, lo que resulta en una recuperación de la señal esparcida. Un filtro con ancho de banda R se usa para remover cualquier componente espuria de alta frecuencia. Si hay una señal no deseada en el receptor, la multiplicación por la señal de código la esparcirá de la misma manera como sucedió con la señal de información en el transmisor. Debido a lo anterior, la señal de información se verá únicamente afectada por las coordenadas de la señal interferente que estén en el mismo espacio.

C.3 Secuencias Pseudoaleatorias

Existen dos enfoques utilizados en la técnica de espectro disperso, uno se llama *referencia transmitida* (TR) y el otro *referencia almacenada* (SR).

En el enfoque de *referencia transmitida* puede utilizarse una señal de código realmente aleatoria para el esparcimiento y recuperación de la señal, debido a que la señal modulada y la de código son trasmitidas simultáneamente en diferentes regiones del espectro.

En el enfoque de *referencia almacenada* no se puede usar una señal de código realmente aleatoria, debido a que el código requiere estar almacenado, o generado, en el receptor. En este caso se usa una señal *pseudoaleatoria*.

La diferencia entre una señal aleatoria y una pseudoaleatoria es que la primera no puede predecirse, sus variaciones sólo pueden describirse de manera estadística; a diferencia de la

anterior, una señal pseudoaleatoria es realmente determinística y periódica, y es conocida tanto por el transmisor como por el receptor. Aunque la señal es determinística, aparenta poseer las propiedades estadísticas de una muestra de ruido blanco. Para un detector que no posea el código representa una señal realmente aleatoria.

Existen tres propiedades básicas que deben cumplirse en una secuencia pseudoaleatoria [1]:

- Propiedad de balance. Hace referencia al número de 1 y 0 lógicos, un buen balance requiere que en la secuencia el número de 1 y 0 lógicos varíe a lo más en un dígito.
- Propiedad de corrida. Una corrida se define como una secuencia de dígitos binarios de un solo tipo. La aparición de un dígito diferente inicia una nueva corrida. La longitud de la corrida es el número de dígitos de la corrida. Entre las corridas de 1 y 0 en cada periodo, es deseable que cerca de la mitad de las corridas de cada tipo sea de longitud 1, cerca de un cuarto sea de longitud 2, un octavo de longitud 3 y así sucesivamente.
- Propiedad de correlación. Si un periodo de la función se compara término a término con un corrimiento cíclico de la misma función, es recomendable que el número de coincidencias difiera del número de discrepancias en no más de uno.

C.4 Generación de Secuencias Pseudoaleatorias

Las secuencias pseudoaleatorias más conocidas son las secuencias de máxima longitud generadas por registros de corrimiento, también llamadas "*m-sequences*". Este tipo de secuencias tiene una longitud $L = 2^m - 1$ bits y es generada por un conjunto de m registros de corrimiento con retroalimentación lineal, como el que se muestra en la figura C.1.

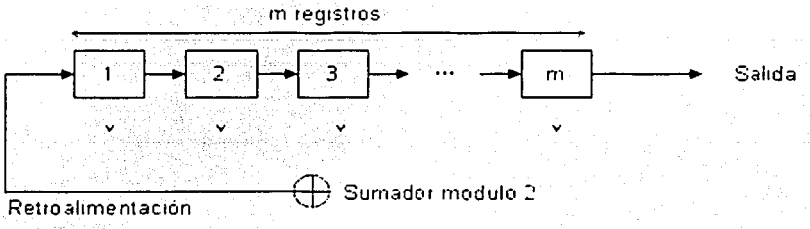


Figura C.1. Esquema general de un registro de corrimiento lineal de m etapas

Estas secuencias tienen periodo L. Cada periodo tiene una secuencia de 2^{m-1} unos y $2^{m-1} - 1$ ceros. Las secuencias pseudoaleatorias generadas dependen del número de registros de corrimiento, las conexiones, y las condiciones iniciales.

La secuencia pseudoaleatoria x(t) es una función periódica con forma de onda pulsar, donde a cada pulso se llama simbolo de código o *chip*. Para una secuencia pseudoaleatoria de duración unitaria de chip y periodo igual a p chips, la función de autocorrelación esta dada por la siguiente expresión:

$$R_x(\tau) = \frac{1}{p} * (\text{número de concordancias - número de discrepancias}) \tag{1}$$

La comparación para determinar las concordancias o discrepancias, entre la secuencia y su réplica que se recorrerá, se hace a lo largo de un periodo completo de la secuencia, τ representa la posición del corrimiento cíclico de la réplica.

El resultado de la correlación de una secuencia pseudoaleatoria del tipo *m-sequence* es:

$$R_x(\tau) = \begin{cases} 1 & para\ \tau = 0 \\ -\frac{1}{p} & en\ cualquier\ otro\ caso \end{cases} \tag{2}$$

Se puede interpretar que $\tau = 0$ es el estado donde la secuencia pseudoaleatoria y su réplica, que se desplaza, están perfectamente alineadas y es por ello que se obtiene un valor de autocorrelación máximo. Para cualquier otro caso, el valor de autocorrelación es constante y menor.

A continuación se presenta una secuencia pseudoaleatoria del tipo *m-sequece* para ejemplificar las caracteristicas antes descritas.

La secuencia que se muestra es de quinto orden, es decir, se utilizan 5 registros de corrimiento linear para generarla. La conexión de los registros al sumador módulo 2 se determinó a partir del polinomio primitivo (45_8) [2]. Los valores iniciales de los registros fueron asignados aleatoriamente. En la figura C.2 se muestra un esquema del registro usado y el polinomio primitivo.

Valores iniciales : [1 0 1 1 0]

Valores de la secuencia (2 periodos) : [0 1 1 0 1 1 1 0 1 0 1 0
0 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0
1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1
0 0 1 1 1 1 1 0 0 1]

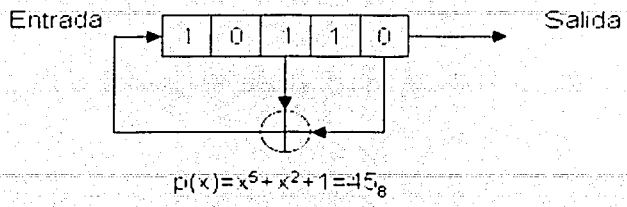


Figura C.2. Registro de corrimiento lineal con retroalimentación, basado en el polinomio de primitivo 45_8

En esta secuencia se pueden verificar las siguientes propiedades:

Propiedad de balance

Número de 1's	2^{m-1}	16
Número de 0's	$2^{m-1}-1$	15

Propiedad de autocorrelación

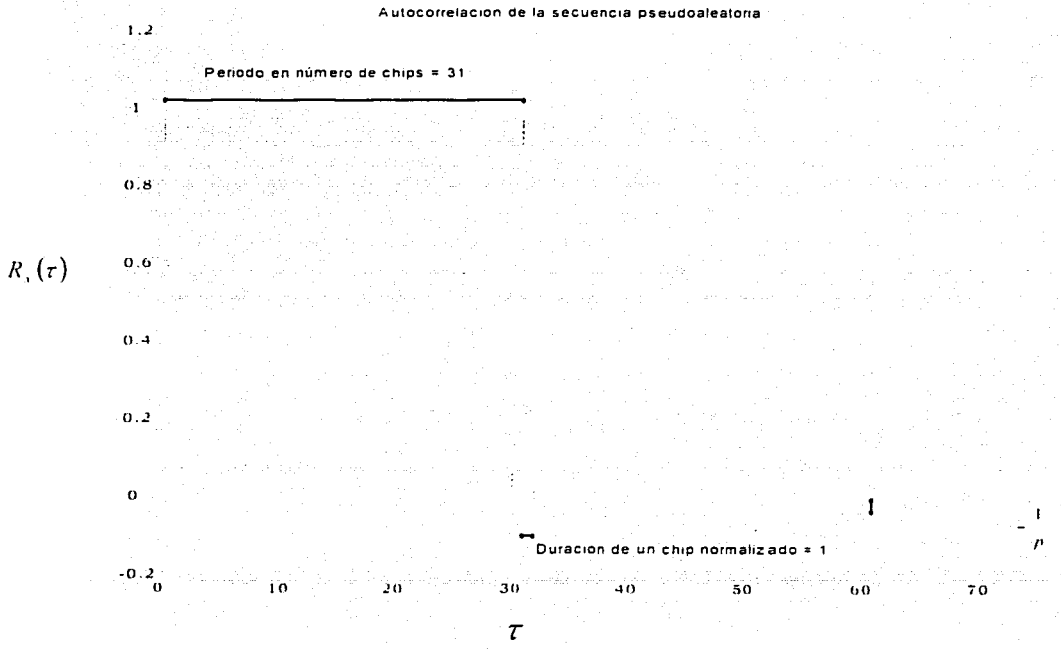


Figura C.3. Función de autocorrelación de una secuencia pseudoaleatora del tipo *m-sequence*

Propiedad de corrida

El número total de corridas es 17, de las cuales 8 son de unos y 9 son de ceros. La tabla de abajo muestra que el número de corridas de cada tipo, se aproxima al deseable descrito anteriormente.

	Longitud de las corridas				
	1	**2**	**3**	**4**	**5**
Número de corridas de 1's	4	2	1	---	1
Número de corridas de 0's	5	3	---	1	---

Idealmente, una secuencia pseudoaleatoria debe tener una función de correlación que tenga propiedades similares a las del ruido blanco, esto es:

$$R_{\tau}(\tau) = \begin{cases} 1 \ para \ \tau = 0 \\ 0 \ en \ cualquier \ otro \ caso \end{cases} \tag{3}$$

Es evidente que las *m-sequences* no cumplen cabalmente con las características del caso ideal, sin embargo, se puede observar que se aproximan mucho a él a medida que la longitud de la secuencia crece.

En el caso de CDMA resulta importante que las secuencias pseudoaleatorias sean mutuamente ortogonales, esto quiere decir que la correlación cruzada, entre diferentes códigos, debe ser 0 y la autocorrelación (mismo código) debe ser 1. El uso de las propiedades anteriores en sistemas de espectro disperso tiene la finalidad de que la interferencia ocasionada en las transmisiones con códigos diferentes sea cero.

Se ha observado que las *m-sequences* no cumplen con las propiedades de correlación cruzada necesarias para CDMA, debido a que presentan picos grandes al compararlas con otras secuencias con el mismo periodo mediante la correlación cruzada. La magnitud de los picos se hace mayor a medida que la longitud de la secuencia aumenta.

A pesar del fenómeno anterior, es posible seleccionar un pequeño subconjunto de *m-sequences* que tienen valores más pequeños de correlación cruzada. Sin embargo, este número de secuencias es muy pequeño para su uso en CDMA. A estas secuencias se les llama Gold sequences y se generan a partir de un par de secuencias de máxima longitud que tienen propiedades especiales [3].

C.5 Sistemas de Espectro Disperso

C.5.1 Secuencia directa

Un sistema de espectro disperso de secuencia directa (DS) es aquel en donde una forma de onda portadora es primero modulada por una señal de datos x(t), y posteriormente modulada por una señal de esparcimiento de alta velocidad o gran ancho de banda g(t).

La modulación de la señal portadora se hace en fase, y se puede expresar matemáticamente de la siguiente manera:

$$s(t) = \sqrt{2P}\cos[\omega_0 t + \theta_1(t) + \theta_g(t)] \tag{4}$$

De la ecuación (4) se observa que la fase de la portadora tiene dos componentes que modulan a la señal: $\theta_1(t)$ que depende de los datos y $\theta_g(t)$ que depende de la secuencia de esparcimiento.

La modulación en fase de los sistemas de secuencia directa es generalmente binaria (BPSK), es decir, se realizan cambios de π radianes a la fase de la portadora de acuerdo a la secuencia de datos.

Si se considera que la señal x(t) y la señal g(t) son trenes de pulsos antipodales con valores de +1 y −1, se puede expresar la ecuación (4) de la siguiente forma:

$$s(t) = \sqrt{2P}\, x(t)\, g(t)\cos(\omega_0 t) \tag{5}$$

De acuerdo a la ecuación (5), la cadena de pulsos de datos se multiplica en primer lugar con la cadena de pulsos de esparcimiento y posteriormente este producto modula a la señal portadora. Si se asigna el valor positivo del pulso, 1, al 0 lógico y el negativo, -1, al 1 lógico, la multiplicación de x(t) por g(t), se puede realizar mediante una suma módulo dos de las secuencias x(t) y g(t).

En la figura C.4 se muestra el esquema básico de un trasmisor de secuencia directa.

El primer paso para llevar a cabo la demodulación es correlacionar o modular nuevamente la señal recibida con una réplica de la señal de código.

La correlación se efectúa multiplicando la señal recibida r(t) con una réplica sincronizada de la señal de esparcimiento $g(t - T_{\Delta})$. Donde T_{Δ} es la estimación del tiempo de retardo en la propagación de la señal, realizada por el receptor.

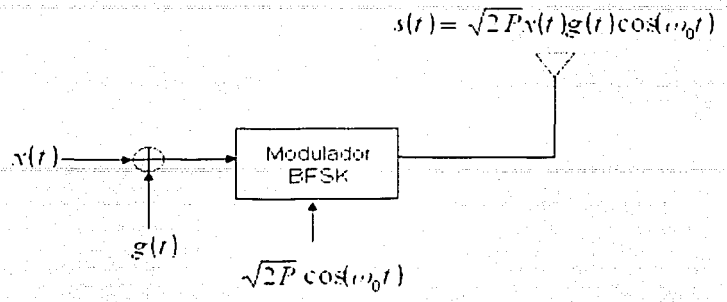


Figura C.4. Esquema básico de un transmisor de secuencia directa BFSK.

En ausencia de ruido e interferencia, la señal a la salida del correlador se puede escribir como:

$$c(t) = A \quad 2P\, x(t - T_d)\, g(t - T_d)\, g(t - T_{ds}) \cos(\omega_0 (t - T_d) + \phi) \tag{6}$$

donde A es un parámetro que modela la ganancia del sistema y ϕ es un ángulo de fase aleatorio en el rango de (0 a 2π). Debido a que $g(t) = \pm 1$, el producto $g(t - T_d)\, g(t - T_{ds})$ será la unidad únicamente si $T_{ds} = T_d$, lo que significa que el código en el receptor está exactamente sincronizado con el código en el transmisor. Cuando se cumple la condición anterior, la salida del correlador en el receptor es la señal de banda angosta modulada en BPSK únicamente por los datos.

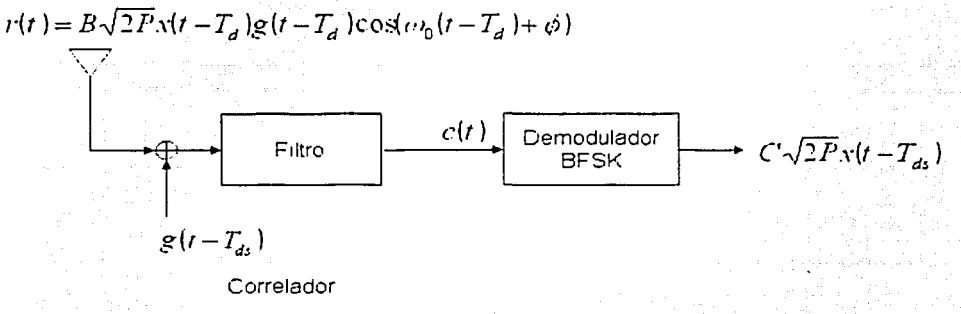


Figura C.6. Esquema básico de un receptor de secuencia directa BFSK.

La siguiente etapa después del correlador es un demodulador convencional de BPSK, el cual recupera los datos transmitidos.

En la figura C.5 se muestra un esquema básico de un receptor de secuencia directa BPSK.

C.5.1.1 Ganancia de Procesamiento

Una característica fundamental de las técnicas de espectro disperso es la protección que pueden proveer contra señales de interferencia de potencia finita. Las técnicas de espectro disperso distribuyen una señal de relativa baja dimensionalidad en un espacio de coordenadas de dimensiones mucho mayores.

Se puede considerar que la señal de información está escondida para el generador de ruido o interferencia (*jammer*) debido a que éste no sabe en que coordenadas la señal se está transmitiendo en un momento determinado. El generador de interferencia puede intentar afectar la señal introduciendo interferencia o ruido a lo largo de todo el espacio con potencia total finita, lo que resulta en una cantidad limitada de interferencia para cada señal coordenada; otra forma puede ser interferir una parte del espacio con su potencia total, dejando las coordenadas restantes libres de interferencia.

En este tipo de sistemas, la manera en que el *jammer* escoja distribuir su potencia es independiente de la relación señal a interferencia (SJR) obtenida. La expresión de la relación SJR [1] es la siguiente:

$$SJR = \frac{E_s N}{E_u D} \tag{7}$$

donde

- D Número de coordenadas ortogonales usadas para representar la información mediante señales de banda angosta.
- N es el número de coordenadas ortogonales del espacio donde se esparce la señal.
- E_s es la energía de cada forma de onda de la señales ortogonales de banda angosta usadas para representar la información.
- E_u es la energía de la señal de interferencia, medida en el mismo periodo de tiempo que E_s
- $N >> D$

De la ecuación (7) se puede concluir que el esparcimiento le da a la señal una factor de ventaja de $\frac{N}{D}$ sobre el *jammer*. A dicho factor se le llama *ganancia de procesamiento (processing gain)*, la cual se representa como G_p.

Debido a que la dimensionalidad de una señal con ancho de banda W y duración T es aproximadamente $2WT$, la *ganancia de procesamiento* se puede expresar de la siguiente manera:

$$G_p = \frac{N}{D} \approx \frac{2W_{ss}T}{2W_{min}T} = \frac{W_{ss}}{W_{min}} \tag{8}$$

donde

- W_{ss} es el ancho de banda donde se esparce la señal o ancho de banda del sistema de espectro disperso
- W_{min} es el ancho de banda mínimo requerido para la transmisión de los datos, es decir, el ancho de banda de la señal de banda angosta.

Para el sistema de secuencia directa G_p se puede expresar también como:

$$G_p = \frac{R_p}{R} \tag{9}$$

C.5.2 Saltos en Frecuencia.

La técnica de espectro disperso llamada saltos en frecuencia (FH) generalmente usa una modulación M-aria de frecuencia (MFSK), donde $k = \log_2 M$ es el número de bits de información usados para determinar cual de las M frecuencias será trasmitida. La posición del conjunto de las M frecuencias es recorrido pseudoaleatoriamente por el sintetizador de frecuencia, sobre un ancho de banda de salto W_{ss}.

En un sistema MFSK estándar los datos modulan a una frecuencia portadora fija; mientras que en un sistema de espectro disperso con saltos en frecuencia, los datos modulan a una portadora cuya frecuencia es determinada pseudoaleatoriamente. En ambos casos sólo se trasmite un tono.

El sistema de saltos en frecuencia puede ser conceptualizado como un proceso de dos etapas: modulación de los datos y modulación de saltos en frecuencia. En la práctica, las dos etapas anteriores se realizan en un solo paso debido a que el sintetizador de frecuencias produce un tono de transmisión basado simultáneamente en el código pseudoaleatorio y en los datos. En la figura C.6 se muestra un esquema básico de un sistema de saltos en frecuencia.

Para cada salto de frecuencia, un generador pseudoaleatorio alimenta al sintetizador de frecuencias con una palabra de L bits o chips, la cual establece una de 2^L posiciones en el espectro en las cuales puede saltar la frecuencia portadora. El mínimo número de chips por

palabra está determinado por el ancho de banda del sistema de espectro disperso de saltos en frecuencia, y el espaciamiento entre las posiciones de saltos consecutivos, Δf .

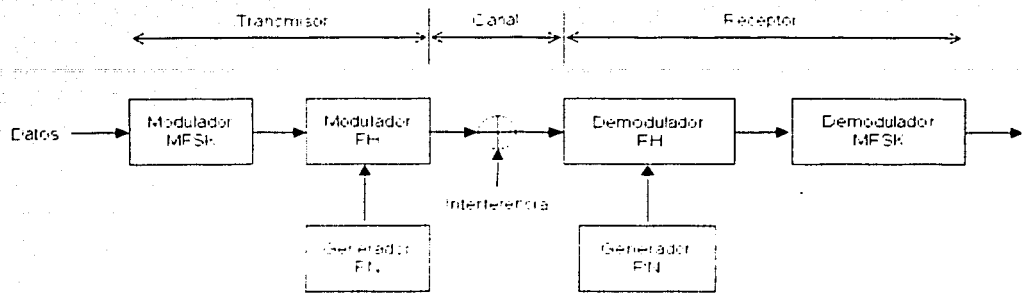


Figura C.6. Esquema básico de un sistema de saltos en frecuencia.

Para un salto en particular, el ancho de banda ocupado por la señal es idéntico al de una modulación convencional MFSK, el cual es mucho menor que W_{ss} . Sin embargo, al promediar muchos saltos resulta que se ocupa todo el ancho de banda del sistema de espectro disperso, W_{ss} .

La tecnologia actual permite anchos de banda para sistemas de saltos en frecuencia del orden de varios Giga Hertz, lo cual es mucho más de lo permitido para sistemas de secuencia directa. Debido a lo anterior, los sistemas de saltos en frecuencia tienen una ganancia de procesamiento mayor que los sistemas de secuencia directa. A causa del gran ancho de banda manejado por los sistemas de saltos en frecuencia es difícil mantener coherencia en la fase de salto en salto, por ello, generalmente se utiliza una demodulación no coherente.

La demodulación se realiza básicamente en dos pasos: el primero es recuperar la señal esparcida y el segundo es demodular la señal MFSK. La recuperación de la señal esparcida se lleva a cabo mezclando la señal recibida con la misma secuencia pseudoaleatoria utilizada para determinar los saltos en frecuencia; la demodulación MFSK se realiza con un banco de M detectores de energia no coherente convencionales, de esta manera, se selecciona el símbolo de mayor similitud.

En el caso de secuencia directa el término *chip* se refiere al un símbolo del código pseudoaleatorio, es decir, al símbolo de menor duración del sistema. De forma similar, para los sistemas de saltos en frecuencia, el término *chip* se usa para nombrar a la mínima forma de onda ininterrumpida.

Los sistemas de saltos en frecuencia se clasifican como sistemas de *saltos lentos (SFH)* y sistemas de *saltos rápidos (FFH).*

C.5.2.1 Saltos lentos en frecuencia

Los sistemas de espectro disperso con saltos en frecuencia lentos son aquellos en los que hay varios símbolos de modulación por salto en frecuencia. Para estos sistemas, la mínima forma de onda ininterrumpida, chip, es la del símbolo del dato.

En la figura C.7 se muestra una forma de onda con tres símbolos de modulación por salto. Cada símbolo corresponde a un chip, y representa una secuencia de bits. En el caso de BFSK, un símbolo representa un solo bit.

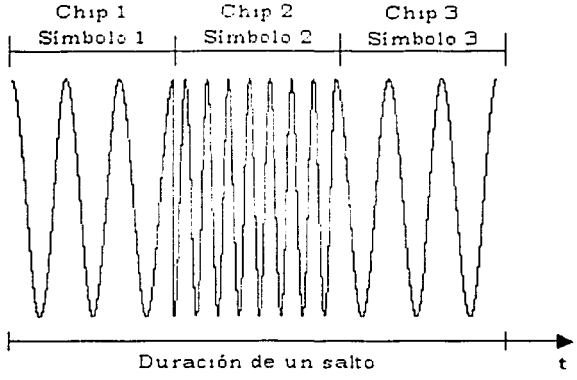


Figura C.7. Forma de onda de una señal de espectro disperso con saltos lentos en frecuencia que presenta tres símbolos de modulación por cada salto en frecuencia.

En la figura C.8 se muestra el funcionamiento del sistema en el dominio de la frecuencia. La modulación es binaria, BFSK, una desviación positiva de la frecuencia representa un 1 lógico y una desviación negativa representa un 0 lógico. En cada salto se agrupan 3 símbolos de modulación, que por tratarse de BFSK, corresponden a 3 bits por salto.

Del análisis anterior se puede verificar que en los sistemas de saltos lentos en frecuencia, la tasa de bits por segundo es mayor o igual a la tasa de saltos por segundo.

C.5.2.2 Saltos rápidos en frecuencia

Los sistemas de espectro disperso con saltos en frecuencia rápidos son aquellos en donde hay varios saltos en frecuencia por símbolo de modulación. En estos sistemas, la mínima forma de onda ininterrumpida, *chip*, es la determinada por el salto.

En la figura C.9 se muestra una forma de onda con cuatro saltos por símbolo de modulación. Cada salto corresponde a un *chip*.

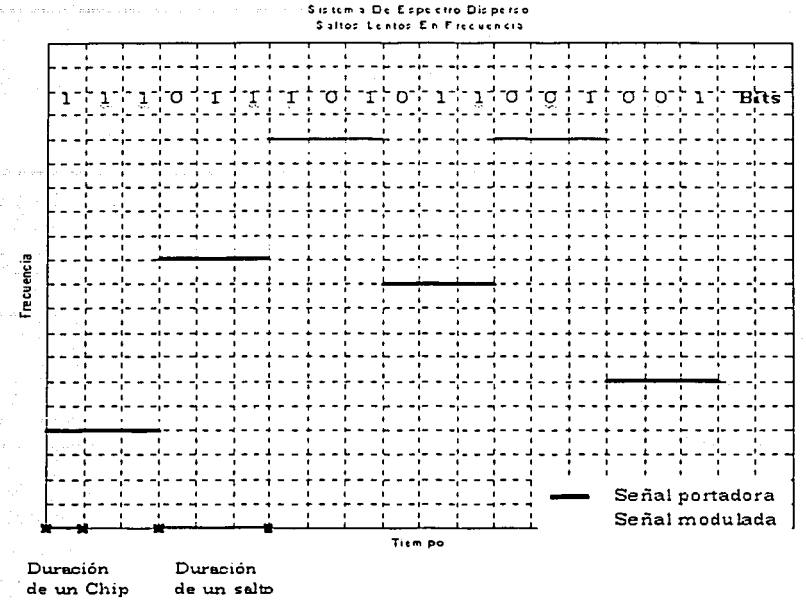


Figura C.8. Esquema del funcionamiento, en el dominio de la frecuencia, de un sistema de espectro disperso con saltos lentos.

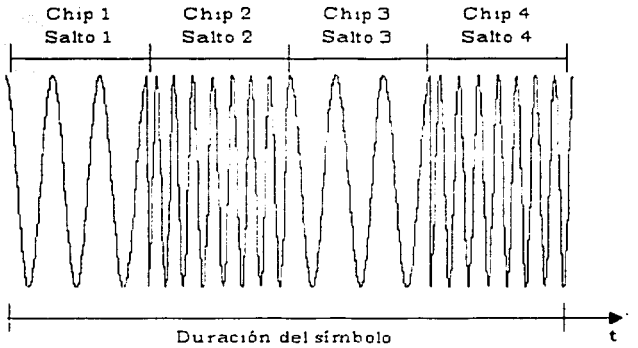


Figura C.9. Forma de onda de una señal de espectro disperso con saltos rápidos en frecuencia que presenta cuatro saltos por cada símbolo de modulación.

En la figura C.10 se muestra el funcionamiento del sistema en el dominio de la frecuencia. La modulación es binaria, BFSK, una desviación positiva de la frecuencia representa un 1

lógico y una desviación negativa representa un 0 lógico. Debido al tipo de modulación, BFSK, un símbolo de modulación representa un sólo bit. Cada símbolo de modulación ocupa 4 saltos en frecuencia, es decir, se transmite 4 veces.

Del análisis anterior se puede verificar que en los sistemas de saltos rápidos en frecuencia, la tasa de bits por segundo es menor a la tasa de saltos por segundo.

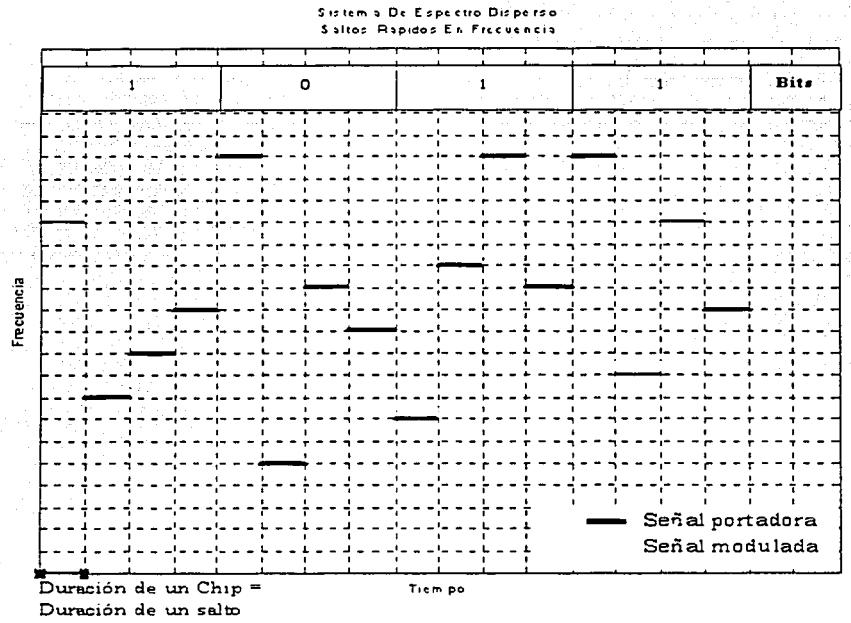


Figura C.10. Esquema del funcionamiento, en el dominio de la frecuencia, de un sistema de espectro disperso con saltos rápidos.

ANEXO D
WAP

En 1997, Motorota, Nokia, Ericsson y Openwave se unieron para desarrollar y utilizar el protocolo de aplicación inalámbrica, o WAP (*Wireless Aplication Protocol*). Su objetivo era el de crear un estándar poderoso para especificar la manera en cómo la Internet es adecuada para comunicaciones móviles. Desde entonces, un número sorprendente de compañías se ha unido a dicha tarea.

WAP puede trabajar con una variedad de tecnologías inalámbricas diferentes, cada una de las cuales se conecta en la parte inferior de la pila de protocolos WAP como soporte. *Bluetooth* da la posibilidad de ser otro soporte para la pila WAP.

De la misma forma que *Bluetooth* tiene el SIG, el cual define los estándares y ayuda a asegurar la interoperabilidad de los dispositivos *Bluetooth*, WAP tiene el foro WAP. El foro WAP permite a las compañías de todas partes de la industria inalámbrica, definir estándares WAP y ayudar a asegurar la interoperabilidad entre productos WAP.

WAP soporta las siguientes redes: CDPD, CDMA, GSM, PDC, PHS, TDMA, FLEX, ReFLEX, iDEN, TETRA, DECT DataTAC y Mobitex.

WAP también soporta una arquitectura cliente/servidor. El cliente se comunica con un servidor (o un *proxy*) utilizando los protocolos WAP. Aquellos dispositivos habilitados con WAP y que actúan como clientes, pueden usar *microbrowsers*, los cuales están especialmente diseñados para ser compatibles con dispositivos móviles tales como los teléfonos celulares móviles. Un *microbrowser* está diseñado para trabajar con una pantalla pequeña, y utilizar menos memoria que un *browser* que corre en una PC de escritorio. WAP soporta estas facilidades a través del *Wireless Markup Language* (WML).

El WML trabaja de forma similar que el HTML, el cual es utilizado para diseñar páginas en el *World Wide Web*; entonces las páginas WML trabajan también de forma similar que las páginas HTML, pero están diseñadas para caber en pequeñas pantallas.

Los proveedores de servicio de Internet están brindando páginas WAP debido a que eso les permite tener acceso a un basto mercado de consumidores móviles, muchos de los cuales navegan en la red en casa, en la oficina, pero no lo pueden hacer en movimiento.

El mercado de servicios WAP se extiende mucho más allá de lo que se cree, pues los dispositivos móviles se están vendiendo alrededor de dos veces más que las PC's de escritorio (poner una referencia).

WAP no sólo trabaja en teléfonos. Hay dispositivos como los PDAs y los *pagers*, así como cualquier dispositivo con una pantalla y con la habilidad de soportar conexión inalámbrica, los cuales pueden convertirse en dispositivos WAP.

La combinación de la tecnología inalámbrica *Bluetooth* y la transferencia de datos vía WAP proveen una herramienta potencial para transferencia de datos local en nuevos escenarios. Hay muchos casos en los que la transferencia de datos de un área local pequeña es útil, y el rango limitado de los enlaces *Bluetooth* puede ser una ventaja, asegurando que la información que se transfiere es relevante.

ANEXO E
BLUETOOTH Y OTRAS TECNOLOGÍAS RELACIONADAS

En la figura E.1 se da una idea de las diversas aplicaciones de la conectividad inalámbrica y sus limitantes dependiendo del rango y ancho de banda requeridos. En un extremo se encuentra el home AV setup, donde se puede transmitir video desde una cámara de video digital a un centro de entretenimiento casero o videocasetera. Esto requiere un pequeño rango de unas cuantas decenas de centimetros, pero las tasas de transferencia pueden ser muy altas.

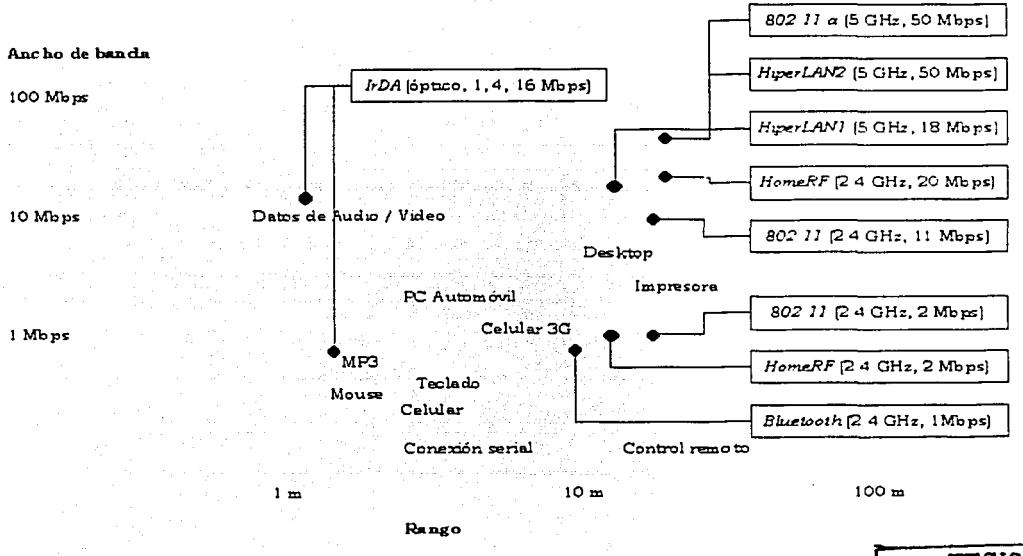


Figura E.1. Aplicaciones inalámbricas y tecnologías relacionadas

En el otro extremo, los teléfonos inalámbricos o el control remoto necesitarán una tasa de transmisión digamos modesta, pero el rango puede ser de decenas de metros para

trabajar en un ambiente casero. En medio, tenemos los dispositivos típicos de oficinas como lo son: impresoras y PC's y por supuesto la clásica WLAN donde tanto una tasa alta de transferencia como un rango razonable son importantes.

Recientemente, el surgimiento sistemas de voz sobre IP (VoIP) han complicado un poco el escenario, pues si la tecnología es aceptada y mejorada volviéndose popular, podría traducirse en una importante carga para las redes que se usan sólo para datos. El tráfico de voz es extremadamente sensible a retrasos y necesita de una completa calidad de servicio en la red, más que los datos.

Las otras tecnologías importantes que se presentan en la figura se presentan a continuación:

E.1 IrDA (Infrared Data Association)

La IrDA creó un sistema de comunicaciones basado en luz infrarroja, por lo que está limitado a línea de vista y no puede penetrar muebles o paredes tal como un sistema basado en radio lo puede hacer. La ventaja que se ha propuesto es que se provee un intercambio de datos controlado y privado.

El conjunto de estándares para IrDA fueron publicados a finales de 1993 incluyendo: la especificación de enlace infrarrojo serial (SIR), la especificación del protocolo de acceso a enlace (IrLAP) y la especificación del protocolo de administración de enlace. En 1995, IrDA liberó extensiones para el estándar SIR para la operación de 4Mb/s, y desde entonces se ha expandido el estándar para incluir operación a 1.152Mb/s, 4.9Mb/s y 16Mb/s utilizando la modulación de posición de pulso (PPM).

En la siguiente tabla se muestra una comparación de las tecnologías Bluetooth e IrDA:

Parámetro	IrDA	Bluetooth
Medio	Óptico Direccional	RF Omnidireccional
Tasa de transmisión neta (Mb/s)	1.152, 4, 16	1[1]
Máximo Rango	20cm 1.2cm	10m 100m
Voz y datos	Sólo datos	Voz y datos

Tabla E.1 Comparación de IrDA y Bluetooth

E.2 Digital Enhanced Cordless Telecommunications (DECT)

El estándar DECT fue desarrollado durante 1992 dentro del ETSI como ETS 300 175 y 300 176 como sucesor de los sistemas de telefonía inalámbrica digital CT2 y CT3 en Europa. Hoy en día, el DECT tiene una cobertura más amplia para el mercado a nivel

[1] Esta tasa se refiere a la tasa de símbolos; las tasas de transmisión de datos en aplicaciones son menores.

mundial y también ha sido considerado como una solución *Wireless Local Loop* (WLL) en áreas rurales y países en desarrollo.

Basándose en el esquema multi portadora TDMA TDD, DECT utiliza diez frecuencias portadoras en el rango de 1.88 a 1.9GHz. Cada vez que un paquete dura 10 ms y las 24 ranuras de tiempo se esparcen en dos mitades: 12 ranuras para el *downlink* y 12 para el *uplink*, con cada ranura conteniendo hasta 32kb/s de información de voz codificada con ADPCM[2], Esto da por resultado cerca de doce enlaces de voz simultáneos y full dúplex. Debido a la flexibilidad de la especificación, estos canales múltiples pueden combinarse en un solo soporte de n* 24kb/s (tasa de datos después de la protección a errores) de un máximo de 522kb/s para aplicaciones de datos.

El sistema DECT comprende una o más partes fijas, o estaciones base, y una o mas partes portátiles. No hay límite en el tamaño de la infraestructura una vez que se han establecido el número de estaciones base o terminales inalámbricas. El estándar base sólo cubre la interfase entre la parte fija y la parte portátil, brindando una serie de herramientas con protocolos y mensajes de los cuales se puede hacer selecciones (de forma similar a como trabajan los perfiles Bluetooth) para tener acceso a un cierto tipo de red. Los perfiles DECT han sido definidos para aplicaciones de radio en el *Local Loop* (RAP), Internetworking ISDN (IAP), e internetworking GSM (GIP).

Una estación base tipo DECT transmite continuamente por lo menos en un canal, mandando identificadores para que los portátiles puedan analizar la transmisión, que es de tipo *broadcast*, y así saber si las características de los servicios coinciden con las suyas para realizar el enlace.

La selección dinámica de canales y reservación de espacio y la capacidad de *handover* permiten a los portátiles escapar de una conexión de radio interferida estableciendo un segundo enlace (en un nuevo canal) ya sea con la misma estación base o con otra.

Otro aspecto de este proceso de *handover* es que debido a que los portátiles se mueven fuera de rango de las estaciones base tienen que hacer el *handover* a otra estación base cercana, permitiendo que el rango efectivo incremente agregando más partes fijas al sistema, algo que las versiones 1.0 y 1.1 de Bluetooth no puede hacer.

Sin el *handover*, Bluetooth no puede ofrecer las características de rango de la instalación inalámbrica DECT, lo que hace del DECT un sistema de telefonía muy escalable.

La siguiente tabla hace una comparación de DECT contra *Bluetooth*.

[2]La Modulación Adaptativa Diferencial de codificación de pulso es un esquema de codificación de voz de alta calidad que hace uso de la función auditiva humana para realizar una compresión de alto nivel.

Parámetro	DECT	Bluetooth
Banda de Frecuencias (GHz)	1.88 - 1.9	2.4
Tasa de transmisión neta (Mb/s)	1.152	1
Máximo Rango	300m	10m / 50m
Handover	Intercélula e intracélula	No se soporta entre piconets
Máximo número de enlaces de voz Full Dúplex	12	3

Tabla E.2 Comparación de DECT y Bluetooth

E.3 IEEE 802.11

El grupo de trabajo 802.11 del cuerpo de estándares de IEEE en los Estados Unidos es responsable de definir y mantener la especificación y estandarización de las WLANs. La especificación de estándar 802.11 en 1997 definia tres especificaciones de capa física (PHY) y una especificación de control de acceso al medio (MAC). Sin embargo, desde entonces, se ha trabado en extender las especificaciones originales de PHY para brindar tasas de transmisión superiores. Esto llevó a los estándares 802.11a y 802.11b.

EL MAC trabaja con dos configuraciones de red:

- Configuración independiente. Las estaciones se comunican directamente entre si sin apoyo de una infraestructura (lo que se conoce como establecimiento de redes ad-hoc), fácil de operar, pero con un área limitada de cobertura.
- Configuración de infraestructura. Las estaciones se comunican via puntos de acceso, que son parte de un sistema con un área de distribución más amplia.

El MAC brinda un mecanismo básico de acceso con evaluación y sincronización de canal, y se evita las colisiones utilizando el esquema de *Carrier Sense Multiple Access* (CSMA)[3]. También brinda servicio de exploración (similar a la *exploración* y *búsqueda* de *Bluetooth*), establecimiento del enlace, fragmentación de datos, autenticación, encriptamiento, administración de potencia y facilidades de *roaming*.

La especificación 802.11 define tres PHYs asociadas:

- Espectro disperso de salto de frecuencia (FHSS), con 2.4GHz en la banda ISM, 1 y 2Mb/s; modulación de dos y cuatro niveles GFSK y; saltando a 50saltos/s en 79 canales.
- Espectro disperso de secuencia directa (DSSS), con 2.4GHz en la banda ISM, 1 y 2Mb/s, modulación DBPSK y DQPSK y; secuencia Barrer de 11 chips para la expansión.

[3] Un dispositivo escucha el canal antes de transmitir y sólo transmite si el canal no está utilizado.

- IR bandabase, con infrarrojo difuso; transmisión de 1 y 2Mb/s y; modulación 16 PPM y 4 PPM.

E.3.1 IEEE 802.11b

Este esquema extiende de 802.11 DSS PHY para brindar 5.5 y 11 Mb/s, además de los 1 y 2Mb/s de datos usando un *Complementary Code Keying* (CCK) de 8 chips como esquema de modulación. El nuevo PHY se llama Secuencia Directa de espectro Disperso de Alta tasa de transferencia (HR/DSSS). Debido a que usan el mismo preámbulo y cabecera, tanto el 802.11 DSSS y el 802.11b HR/DSS pueden coexistir en la misma red.

E.4 El grupo de trabajo HOMERF (HRFWG)

EL HRFWG se formó en marzo de 1998 para crear una especificación para comunicación digital inalámbrica entre PCs y dispositivos electrónicos en cualquier lugar dentro del hogar.

El grupo tiene más de 90 miembros incluyendo Intel, IBM, Compaq y Microsoft.

Ha desarrollado una especificación para comunicaciones inalámbricas en el hogar, conocida como el protocolo de acceso inalámbrico compartido (SWAP) para combinar telefonia con distribución de datos en un ambiente de hogar.

E.4.1 SWAP

El SWAP está diseñado para soportar tanto tráfico de voz como de datos combinando DECT y 802.11 FHSS. Soporta TDMQ para brindar la entrega se servicios isócronos, como voz interactiva, y un servicio de paquetes de alta velocidad utilizando el esquema 802.11 CSM/CA.

Un sistema SWAP puede operar ya sea como una red ad-hoc o como una red administrada bajo el control de un punto de conexión. Cada nodo puede operar como:

- Un punto de conexión que soporta servicios de voz y datos.
- Una terminal de voz que sólo usa el servicio de TDMA para comunicarse con su estación base.
- Un nodo de datos que usa el servicio CSMA/CA para comunicarse con la estación base u otros nodos de datos.
- Un nodo de voz y datos que usa ambos tipos de servicios.

En la siguiente tabla se muestra una comparación entre SWAP y Bluetooth.

Parámetro	SWAP	Bluetooth
Frecuencia (saltos/segundo)	50	1600
Potencia transmitida (mW)	100	10/100
Tasa de transmisión neta (Mb/s)	1 (mod 2FSK) 2 (mod 4FSK)	1 (mod 2FSK)
Máximo Rango	50m	10/50/100m
Máximo número de nodos	127 dispositivos por red	8 por piconet, muchos más via scatternet o parking
Máximo número de enlaces de voz Full Dúplex	6	3
Algoritmo de Seguridad	Blowfish	SAFER+

Tabla E.3 Comparación de SWAP y Bluetooth

E.5 HIPERLAN

El estándar *High Performance Radio Local Area Network* (HIPERLAN) fue desarrollado dentro de ETSI durante el periodo de 1991 a 1996. El grupo de trabajo HIPERLAN concluyó que el espectro compartido como el ISM no facilitaba las altas tasas de transmisión y calidad de servicio que ellos consideraban necesarias para un networking inalámbrico basado en multimedia, por lo que tanto HIPERLAN Tipo 1 (H/1) e HIPERLAN Tipo 2 (H/2) usan un espectro dedicado a 5GHz.

Mientras que el H/1 es muy parecido al Ethernet inalámbrico, hubo un requerimiento para seguir un desarrollo similar a una versión inalámbrica de ATM, así se desarrolló el H/2.

El conjunto completo de especificaciones del H/2 ofrece opciones para tasas de transmisión de 54, 36, 16 y 6Mb/s, el PHY adopta un esquema multi portadora OFDM usando 48 frecuencias portadoras por cada símbolo OFDM. Cada portadora podría modularse usando BPSK, QPSK, 16-QAM o 64-QAM para brindar diversas tasas de transmisión.

En la tabla siguiente se muestra una comparación entre H/1 y H/2.

Parámetro	HIPERLAN Tipo 1	HIPERLAN Tipo 2
Aplicación	Ethernet LAN inalámbrico	ATM inalámbrico
Potencia transmitida (mW)	10/100/1000	10/100/1000
Tasa de transmisión (Mb/s)	23 5	6, 16, 36, 54
Tasa de transmisión neta (Mb/s)	> 18	50 (max)
Máximo Rango	50m	50m
Servicios	Saltos en tiempo y asincrono	Saltos en tiempo y asincrono
Acceso al canal	Dinámico, con prioridad	Reservado

Modulación	Gaussian Minimum Shift Keying (GMSK)	Orthogonal Frequency Division Multiplexing (OFDM) + BPSK, QPSK, y QAM modulación de portadora

Tabla E.4 Comparación de HIPERLAN Tipo 1 e HIPERLAN Tipo 2

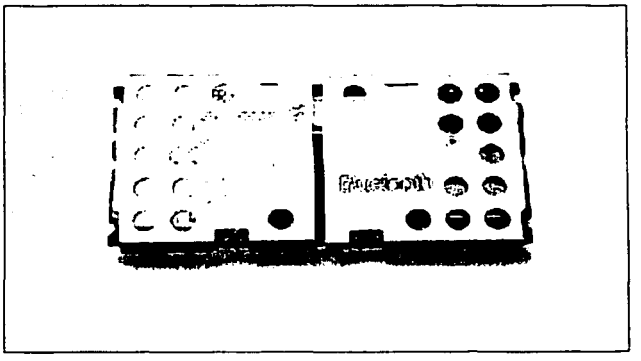
E.6 Direcciones Web útiles

IrDA:	http://www.irda.org
DECT:	http://www.dectweb.com
IEEE 802.11:	http://grouper.ieee.org/groups/802/11
IEEE 802.15:	http://grouper.ieee.org/groups/802.15
HomeRF:	http://www.homerf.org
H/1:	http://www.HIPERLAN.com
H/2:	http://www.H/2.com
ETSI BRAN	http://www.etsi.org/bran
MMAC:	http://www.arib.or.jp/mmac

ANEXO F
ESPECIFICACIONES DEL MÓDULO BLUETOOTH

Key Features

- Pre qualified Bluetooth 1.0B Module
- RF output power class 2
- FCC and ETSI approved
- 460 kb/s max data rate over UART
- Multiple interface for different applications
  - UART for data
  - PCM for voice
  - USB for voice and data
- I²C interface
- Internal crystal oscillator
- HCI firmware included
- Multi Point Operation
- Built in shielding



Suggested Applications

- Computers and peripherals
- Handheld devices and accessories
- Access points

Description

ROK 101 007 is a short range module for implementing Bluetooth functionality into various electronic devices. The module consists of three major parts: a baseband controller, a flash memory, and a radio that operates in the globally available 2.4 - 2.5 GHz free ISM band.

Both data and voice transmission is supported by the module. Communication between the module and the host controller is carried out using a high speed USB interface compliant with USB Specifications 1.1 or an UART/PCM interface. When using the USB interface, the module appears as a USB slave device and therefore requires no PC resources.

ROK 101 007, which is compliant with Bluetooth version 1.0B, is a Class 2 Bluetooth Module (0 dBm) and is type approved. The module supports all Bluetooth profiles.

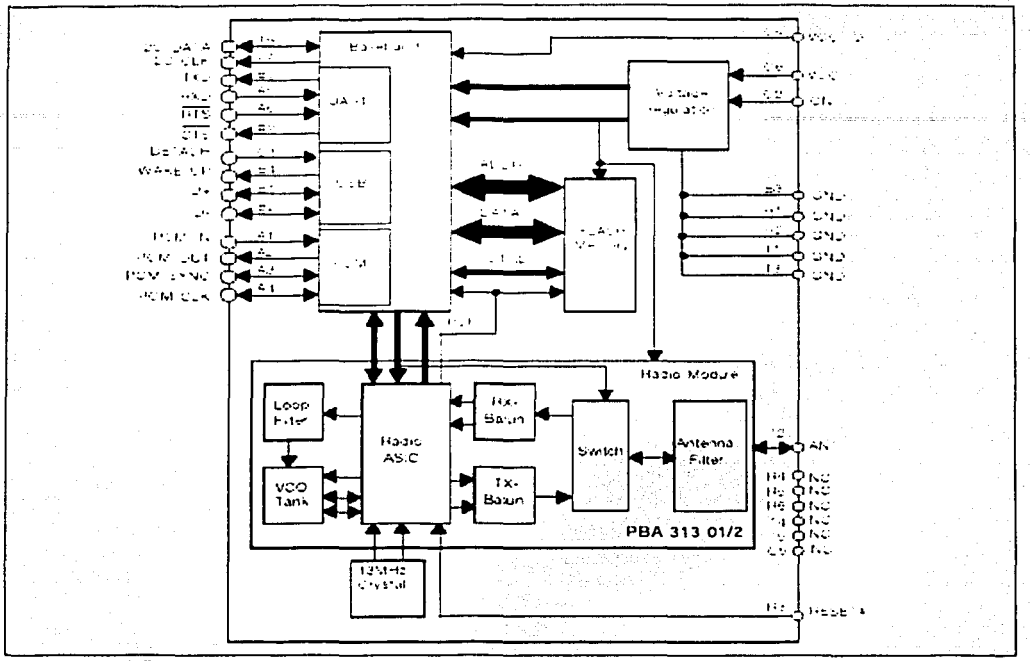


Figure 1. Block Diagram

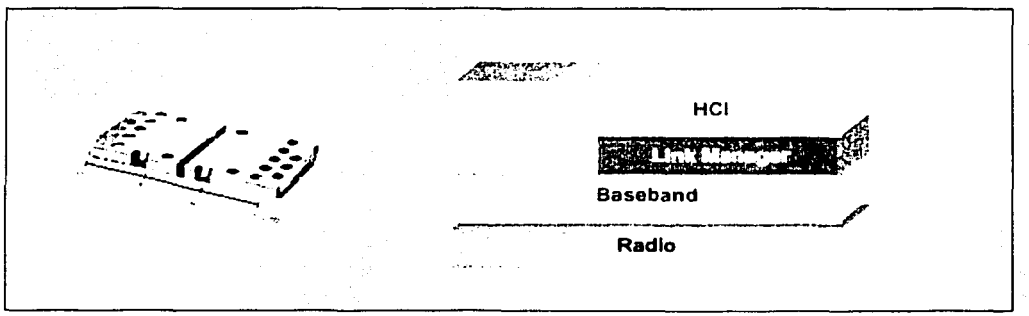


Figure 2. Actual size of the Ericsson Bluetooth Module and also showing the HCI and RF stack

Absolute Maximum Ratings

Parameter	Symbol	Min	Typ	Max	Unit
Temperature					
Storage temperature	T_{stg}	-30		+85	C
Operating temperature	T_a	0		+75	C
Power Supply					
V	V	-0.3		+5.25	V
V	V	-0.6		+3.6	V
Digital Inputs					
Input low voltage	V_{il}	-0.5			V
Input high voltage	V_{ih}			V_{io} +0.3	V

Recommended Operating Conditions

Temperature				
Ambient temperature, Test	T	+23		C
Power Supply				
Positive Supply Voltage	V	+3.3		V
I/O Ports Supply Voltage	V	+3.3		V

Electrical Characteristics

DC Specifications

Unless otherwise noted, the specification applies for T_a = 0 to +75 C, 3.175 < V < 5.25V

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Power Supply						
Supply Voltage		V	3.175	3.3	5.25	V
I/O Ports Supply Voltage	See note 10	V	2.7	3.3	3.6	V
Digital Inputs						
Logical Input High	Except ON signal	V_{ih}	0.7 x V		V	V
Logical Input Low	Except ON signal	V_{il}	0		0.3 x V	V
Logical Input High	ON signal only	V	2.0		V	V
Logical Input Low	ON signal only	V	0		0.4	V
Digital Outputs						
Logical Output High		V	0.9 x V		V	V
Logical Output Low		V	0		0.1 x V	V

Parameter	Condition	Symbol	Min	Typ	Max	Unit
Average Current Consumption	Average I [illegible]					
Standby		I [illegible]		5.35		mA
Shutdown: SW		I [illegible]		2.35		mA
Shutdown: HW	See note 1	I [illegible]		1		uA
Page Scan Mode P0	Page Scan Enable Page scan window: continuous Page scan interval: n.a.(0s)	I [illegible]		50		mA
Page Scan Mode P1	Page Scan Enable Page scan window: 11.25ms Page scan interval: 1.28s	I [illegible]		6.35		mA
Page Scan Mode P2	Page Scan Enable Page scan window: 11.25ms Page scan interval: 2.56s	I [illegible]		6.15		mA
Inquiry Scan with Page Scan Mode P0	Inquiry Scan Enable Page Scan Window: 2.56s 11.25ms Page Scan interval: 2.56s Inquiry Scan Window: 11.25ms Inquiry Scan interval: 2.56s	I		50		mA
Inquiry Scan with Page Scan Mode P1	Inquiry Scan Enable Page Scan Window: 11.25ms Page Scan interval: 1.28s Inquiry Scan Window: 11.25ms Inquiry Scan interval: 2.56s	I [illegible]		6.55		mA
Inquiry Scan with Page Scan Mode P2	Inquiry Scan Enable Page Scan Window: 11.25ms Page Scan interval: 2.56s Inquiry Scan Window: 11.25ms Inquiry Scan interval: 2.56s	I [illegible]		6.35		mA
Connect State	Established connection with data transfer	I [illegible]		26		mA
V Current	Only V	I [illegible]		TBD		mA

RF Specifications

Parameter	Condition	Symbol	Min	Typ	Max	Unit
General						
Frequency Range			2400		2483	GHz
Double Sided IF Bandwidth						MHz
Antenna load				50		Ω
VSWR	RX mode			3:1		
VSWR	TX mode, see note 2			3:1		
Receive Performance						
Sensitivity level	P_{in} -70dBm, 75kHz offset (max)				0.1%	BER
Maximum input level	P_{in} -20dBm, 75kHz offset (max)				0.1%	BER
C/I	C/I 6dB				14	dB
C/I $_{1}$	C/I 6dB				14	dB
C/I $_{2}$	C/I 6dB				30	dB
C/I $_{3}$	C/I 6dB				40	dB
Blocking C/I $_{n}$	See figure 6					
Out of band blocking	30 - 1910 MHz		-4			dBm
	1910 - 2000 MHz		10			dBm
	2000 - 2399 MHz		27			dBm
	2484 - 3000 MHz		27			dBm
	3.00 - 12.75 GHz		10			dBm
Spurious Emissions	30 MHz to 1 GHz				-57	dBm
Spurious Emissions	1 GHz to 12.75 GHz				-47	dBm
Transmitter Performance						
Frequency deviation	see notes 3,4 and figure 3	f_d	140		175	kHz
Initial frequency error	see note 5		-46		+46	kHz
TX power			2	1.5	4	dBm
TX carrier drift in 1 slot (366 µs)		f_1	-25		25	kHz
TX carrier drift in 3 slots (1539 µs)		f_2	-40		40	kHz
TX carrier drift in 5 slots (2862 µs)	see figure 4	f_3	-40		40	kHz
20 dB bandwidth	Measured with RBW: 10 kHz and peak detector				1000	MHz
Spurious Emissions	30 MHz - 1GHz				-36	dBm
Spurious Emissions	1 GHz - 12.75 GHz				-30	dBm
Spurious Emissions	1.8 GHz - 1.9 GHz				-47	dBm
Spurious Emissions	5.15 GHz - 5.3 GHz				-47	dBm
Timing performance						
LPO CLK frequency	Trimmed, see note 6	f_{LPO}		3.2		kHz
Tolerance of LPO CLK	see note 6		-250		+250	ppm
System clock frequency		f_{sys}		13.0000		MHz
Tolerance of system clock	see note 5		-20		+20	ppm
Channel switching time	see figure 5			150		µs
Received Signal Strength Indicator						
RSSI	input power = -40 dBm		15		25	n/a
RSSI	input power = -60 dBm		3		17	n/a

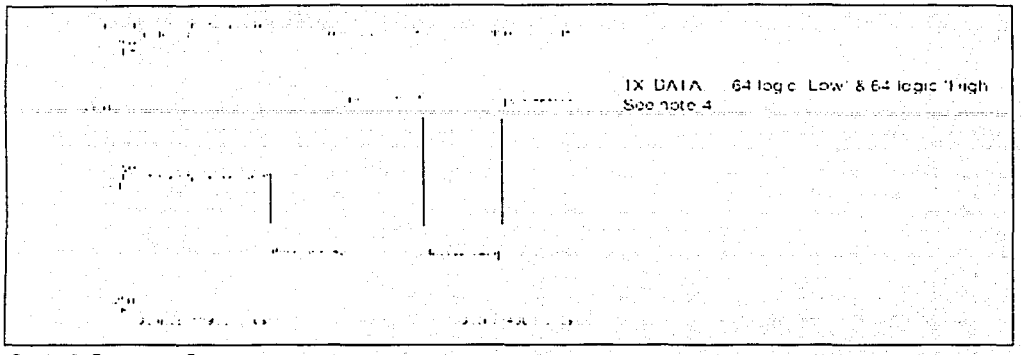


Figure 3. Frequency Deviation

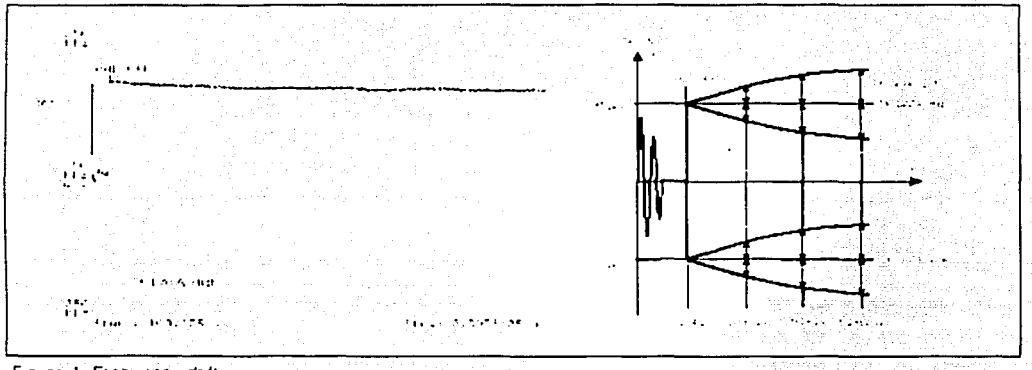


Figure 4. Frequency drift

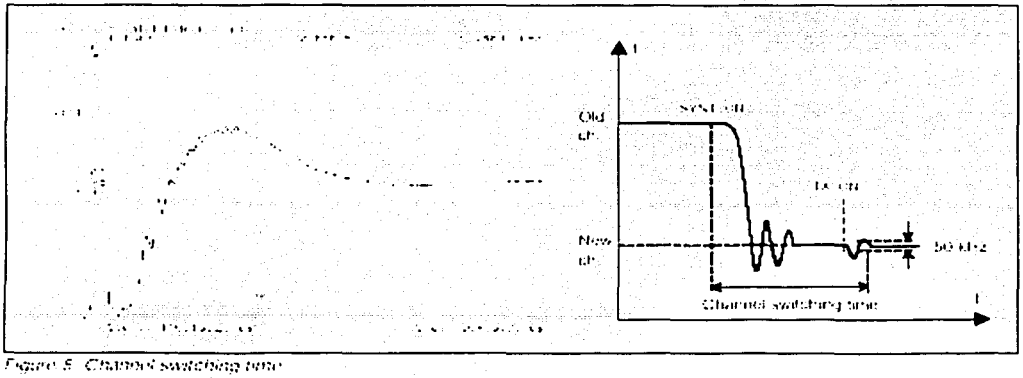


Figure 5. Channel switching time

RF Specifications continued...

C/I Blocking

The blocking characteristics can be basically split into two regions: In band and Out of band. Blocking is performed both on the chip and on the module level.

- **In-band**
  Filtering on chip
  C/I = 2MHz: -30 dB: < 0.1% BER
  C/I = 3MHz: -40 dB: < 0.1% BER

- **Out-of-band**
  Antenna filter: DC to 1.9 GHz and 3rd harmonic
  Switch: low freq. and 2nd harmonic
  RX balun: low freq. and 2nd harmonic
  On chip IF filter

Figure 6 shows the combination blocking effect of the antenna switch, antenna filter and RX balun. In addition to the blocking characteristics shown in figure 6, there is antenna isolation and filtering on the chip.
Marker 1 shows the region where the Bluetooth band is located. Markers 2 - 4 show the blocking at the telecom frequency bands.
An example of the total blocking characteristics can be seen in figure 7.

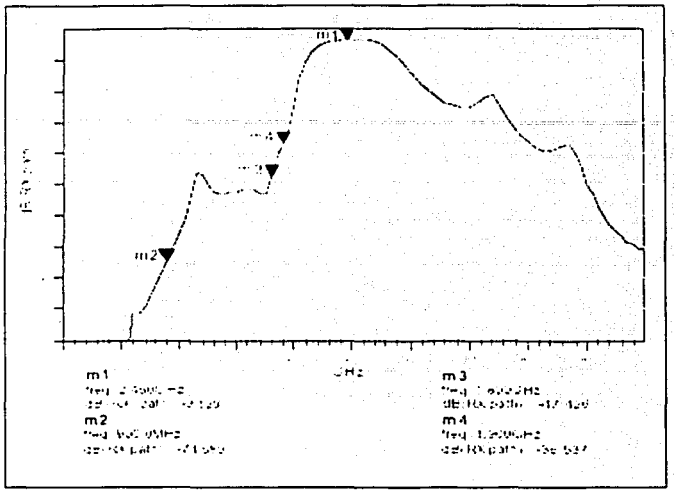


Figure 6 Typical blocking characteristics excluding antenna switch, RX balun and on chip filtering

Example 1		Example 2	
• Antenna isolation	15 dB	• Antenna isolation	25 dB
• Antenna filter, Antenna switch, RX-balun	27 dB	• Antenna filter, Antenna switch, RX-balun	36 dB
• Interference level before IF filter +33-15-27=	-9 dBm	• Interference level before IF filter +33-25-36=	-28 dBm
• 0.1% BER carrier level -40 + (-9)=	-49 dBm	• 0.1% BER carrier level -40 + (-28)=	-68 dBm

Figure 7 Blocking example

Pin Description				
Pin	Pin Name	Type	Direction	Description
A1	PCM IN	CMOS	In	PCM data (see note 7,9)
A2	PCM OUT	CMOS	Out	PCM data (see note 7,9)
A3	PCM SYNC	CMOS	In/Out	Sets the PCM data and clock rate (see notes 7,9)
A4	PCM CLK	CMOS	In/Out	PCM clock that sets the PCM data rate (see notes 7,9)
A5	RXD	CMOS	Input	Receive data for UART (see note 9)
A6	RTS	CMOS	Input	Ready to send is requested to send data from UART (see notes 7,9)
B1	D1	CMOS	In/Out	User data I/O (see notes 9,10)
B2	D0	CMOS	In/Out	User data I/O (see notes 9,10)
B3	GND	Power	Power	Signal ground
B4	WAKE UP	CMOS	Output	Indicates that the module wants to be attached to the LSB Active high (see notes 9,10)
B5	TXD	CMOS	Output	TX data from the UART (see note 9)
B6	CTS	CMOS	Output	Ready to send signal (clear to send data from UART, see note 9)
C1	DETACH	CMOS	Input	Indicates that the LSB host wants to detach the module Active high (see note 10)
C2	ON	Power	Input	Allows host to switch the module on and off
C3	I2C CLK	CMOS	Output	I2C clock signal (see note 9)
C4	VDD IO	Power	Power	External supply rail for the Input / Output pins
C5	NC			Do not connect
C6	VDD	Power	Power	Supply Voltage
P1	GND	Power	Power	Signal ground
P2	GND	Power	Power	Signal ground
P3	RESET#	CMOS	Input	Active low reset (see note 7,9)
P4	NC			Do not connect
P5	NC			Do not connect
P6	NC			Do not connect
*1	GND	Power	Power	Signal Ground
*2	ANT	RF	In/Out	50Ω Antenna connection
*3	GND	Power	Power	Signal Ground
*4	NC	Power	Power	Test point, internal voltage regulator. Do not connect
*5	NC			Do not connect
*6	I2C DATA	CMOS	In/Out	I2C data signal (see note 9)

Notes

1 Current consumption is based upon when the module is when 'ON' is low and VDD IO is high under
2 During the TX mode, the VSWR specification states the limits that are acceptable before any other RF parameters are strongly effected i.e. frequency deviation and initial frequency error.
3 Frequency deviation measurements are now recorded differentially, if Mod1 - Mod0 : : 2.
4 Provided that the TX INV register bit 0 has been set in the enable register at startup.
5 Tolerance for the system clock takes into account both the complete temperature range and aging of the crystal.
6 LPO CLK frequency is pre-trimmed within a tolerance of +/- 250ppm
7 100kΩ pull up resistors to V are used on the module. PCM signals direction is programmable
8 RESET# signal must be fed from an open drain output
9 CMOS buffers are low voltage TTL compatible signals
10 To be compliant with the LSB specification VDD IO = 3.11V

Mechanical Specification

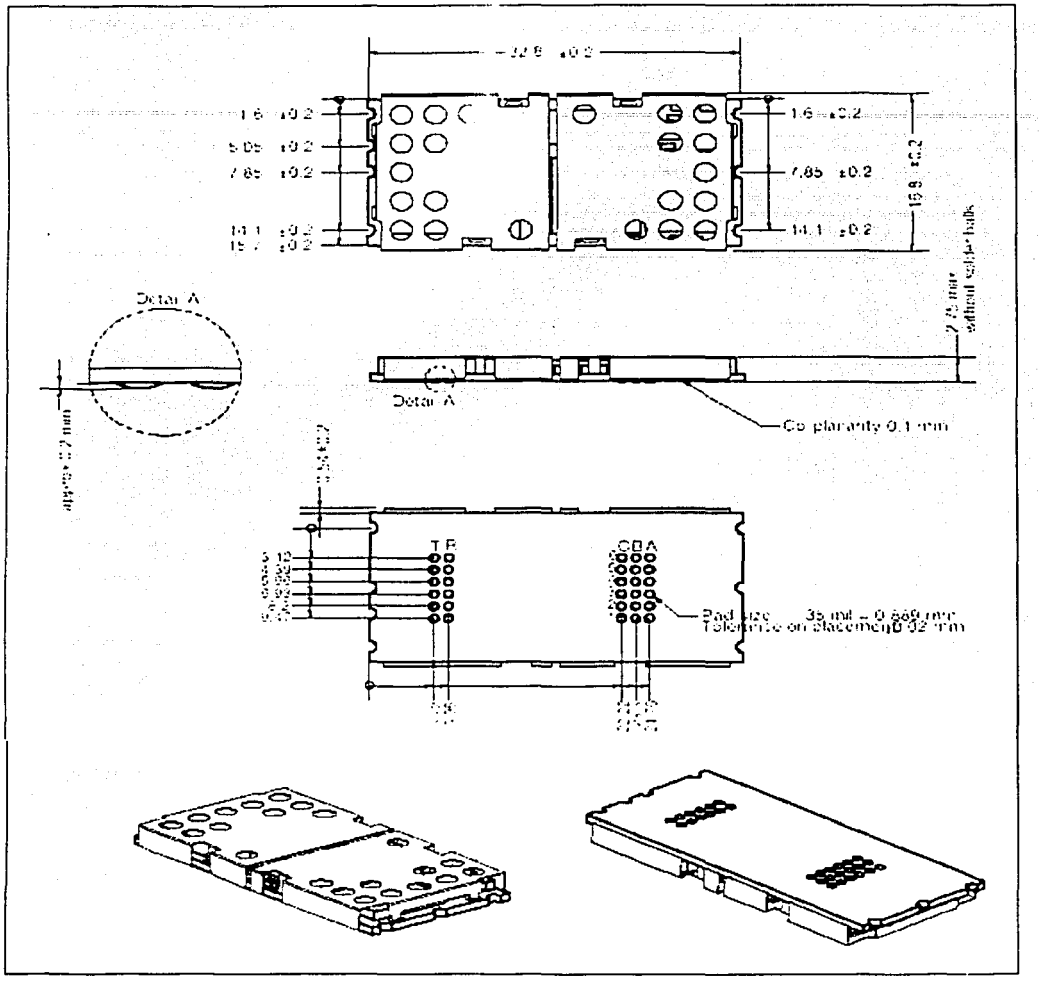


Figure 8. Mechanical dimensions

Application Block Schematics

USB Application

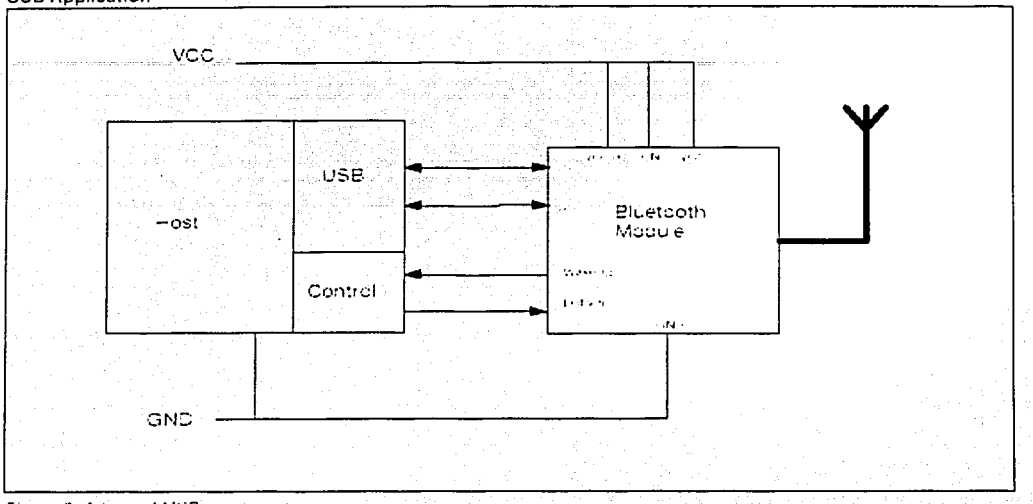


Figure 9: A typical USB configuration

UART and PCM Application

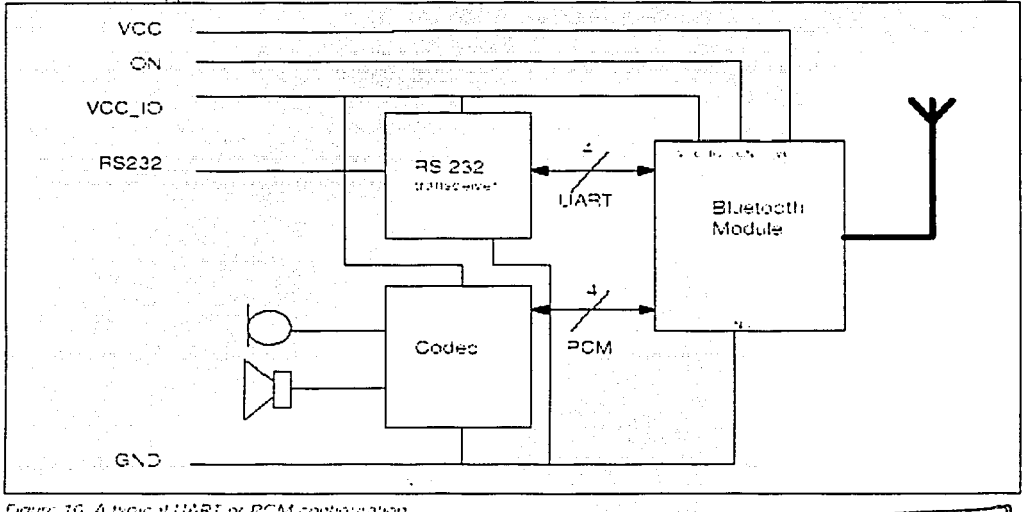


Figure 10: A typical UART or PCM configuration

Functional Description

The ROK 101 007 is a complete Bluetooth module that has been specified and designed according to the Bluetooth System v1.0B. Its implementation is based on a high performance integrated radio transceiver (PBA 313 01 2) working with a baseband controller, a flash memory and surrounding secondary components. Features: low energy consumption for use in battery operated devices.

Block Diagram

ROK 101 007 has five major operational blocks. Figure 1 illustrates the interaction of the various blocks. The functionality of each block is as follows:

1. Radio functionality is achieved by using the Bluetooth Radio PBA 313 01 2. Six operational blocks are shown for this section and their operation is as follows:

1a) VCO tank is a part of the phase locked loop. The modulation is performed directly on the VCO. To ensure high performance the VCO tank is laser trimmed.

1b) Loop filter: filters the tuning voltage of the VCO tank.

1c) RX balun: handles transformation from unbalanced to balanced transmission.

1d) TX balun: handles biasing of the output amplifier stage and transformation from balanced to unbalanced transmission.

1e) Antenna switch: directs the power either from the antenna filter to the receive parts or from the ASIC output ports to the antenna filter.

1f) Antenna filter: band pass filters the radio signal.

2. The baseband controller is an ASIC. This is the basic chip that controls the operation of the radio transceiver via one of the interface methods, USB or UART. Additionally, the baseband controller has a PCM voice interface and I²C interface.

3. A Flash memory is used together with the baseband controller. Please, refer also to the Firmware section.

4. The power management block regulates and filters the supply voltage. V_{BAT} is typically 3.3V and five regulated voltages are produced: 2.6V and 2.2V.

5. An internal clock is mounted on the module. The clock frequency is 13MHz and is generated from a crystal oscillator that guarantees a tuning accuracy within 20ppm.

Bluetooth Module stack

The Host Controller Interface (HCI) handles the communication by the transport layer through the UART or USB interface with the host. The Baseband and radio provides a secure and reliable radio link for higher layers.

The following sections describe the Bluetooth module stack in more detail. It is implemented in accordance with and complies with the Specification of the Bluetooth System v1.0B.

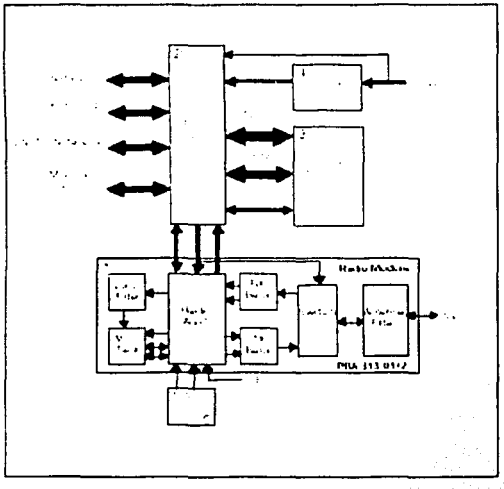


Figure 11. Simplified Block Diagram

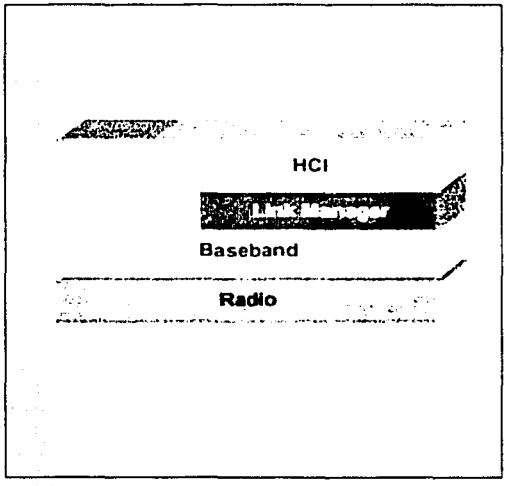


Figure 12. HW/FW parts included in the Ericsson Bluetooth module

Bluetooth Radio Interface

The Bluetooth module is a class 2 device with 4dBm maximum output power with no power control needed. Nominal range of the module with a typical antenna is up to a range of 10 m (at 0 dBm). It is compliant with FCC and ETSI regulations in the ISM band.

Baseband

Bluetooth uses an ad hoc net structure with a maximum of 8 point active units in a single piconet. By default the first unit setting up a connection is the master of the point to point link. The master transmits in the even timeslots and the slave transmits in the odd timeslots.

For full duplex transmission a Time Division Duplex (TDD) scheme is used. Packets are sent over the air in timeslots, with a nominal length of 625 µs. A packet can be extended to a maximum of 5 timeslots (DV3 and DH5 packets) and is then sent by using the same HF channel for the entire packet.

Two types of connections are provided. Asynchronous Connectionless (ACL) for data and the Synchronous Connection Oriented (SCO) for voice. Either 64kb/s data channels can be supported simultaneously. Further 16 slot packets are also packages used for the control purposes.

A variety of different packet types with error correction schemes and data rates can be used over the air interface. Asymmetric connection with variable packet length and speed can be established in each direction.

The Baseband provides the link setup and control routines for the layers above. Furthermore the Baseband also provides Bluetooth security like encryption, authentication and key management.

Please refer to the Specification of the Bluetooth System v1.0B part B for in depth information regarding the Baseband.

Firmware (FW)

The module includes firmware for the host controller interface (HCI) and the link manager, LM.
The FW resides in the Flash and is available in object code format.

Link Manager (LM)

The Link Manager in each Bluetooth module can communicate with another Link Manager by using the Link Manager Protocol (LMP) which is a peer to peer protocol.
The LMP messages have the highest priority and are used for link setup, security, control and power saving modes. The receiving Link Manager

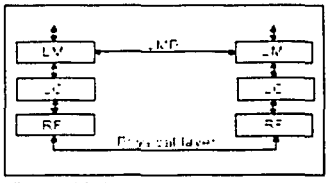


Figure 13: Link manager

filter out the message and does not need to acknowledge the message to the transmitting LM due to the reliable link provided by the Baseband and radio.

LM to LM communication can take place without actions taken by the host. Discovery of features at other Bluetooth enabled devices nearby can be found and saved for later use by the host.

Please refer to the Specification of the Bluetooth System v1.0B part C for in depth information regarding the LMP.

Type	Payload (bytes)	FEC	CRC	Symmetric Max. Rate	Asymmetric Max. Rate
DM1	0-17	2/3	yes	108.8	108.8
DH1	0-27	no	yes	172.8	172.8
DM3	0-121	2/3	yes	387.2	54.4
DH3	0-183	no	yes	585.6	86.4
DM5	0-224	2/3	yes	477.8	36.3
DH5	0-339	no	yes	723.2	57.6
AUX1	0-29	no	no	185.6	185.6

Link control packets

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)	Asymmetric Max. Rate (kb/s)	
						Forward	Reverse
DM1	1	0-17	2/3	yes	108.8	108.8	108.8
DH1	1	0-27	no	yes	172.8	172.8	172.8
DM3	2	0-121	2/3	yes	387.2	54.4	54.4
DH3	2	0-183	no	yes	585.6	86.4	86.4
DM5	2	0-224	2/3	yes	477.8	36.3	36.3
DH5	2	0-339	no	yes	723.2	57.6	57.6
AUX1	1	0-29	no	no	185.6	185.6	185.6

ACL packets

Type	Payload (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)
HV1	10	10	1/3	no	64.0
HV2	20	20	2/3	no	64.0
HV3	30	30	no	no	64.0
DV	10+(0-9)	10	2/3	yes	64.0+64.0

SCO packets

Table 1: Link Control Packets Table, ACL Packets Table, SCO packets

Host Control Interface (HCI)

The HCI provides a uniform command interface to the Baseband and Link Manager and access to HW status registers.

There are three different types of HCI packets:

- HCI command packets - from host to Bluetooth module (HCI).
- HCI event packets - from Bluetooth module (HCI) to host.
- HCI data packets - going both ways.

It is not necessary to make use of all different commands and events for an application. If the application is aimed at a pre-specified profile, the capabilities of such a profile is necessary to adjust to. See Specification of the Bluetooth System v1.0B Profiles.

a) With the HCI UART Transport Layer on top of HCI, the module will communicate with a host through the UART I/F. The PCM I/F is also available for communicating voice.

b) With the HCI USB Transport Layer on top of the HCI, the module will communicate with a host through the USB. Detach and Wake-up signals are also available for notebook implementations.

Please refer to the Specification of the Bluetooth System v1.0B part H:4 for in depth information regarding the HCI and different transport layers.

Module HW Interfaces

UART Interface

The UART implemented on the module is an industry standard 16C450 and supports the following baud rates: 300, 600, 900, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400 and 460800 bits/s. 128 byte FIFOs are associated with the UART.

Four signals will be provided for the UART interface. TxD & RxD are used for data flow, and RTS & CTS is used for flow control.

Please refer to the Specification of the Bluetooth System v1.0B part H:4 regarding the HCI and UART transport layers.

PCM Voice Interface

The standard PCM interface has a serial rate of 8 kHz (PCM_SYNC). The PCM clock is variable between 200 kHz and 2.0MHz. The PCM data can be linear PCM (13- 16bits), A-Law (8bit) or A-Law (8bit).

The PCM I/F can be either master or slave - providing or receiving the PCM_SYNC and the PCM_CLK. The PCM_CLK and PCM_SYNC can be accessed as well.

Over the air the encoding is programmable to be CVSD. A-Law or µ-Law. Preferably, the robust CVSD encoding should be used.

USB Interface

The module is a USB high speed class device (12Mbps) that has the full functionality of a USB slave and is compliant to the USB 1.1 specification. Data transfer occurs on the bi-directional ports D+ & D-.

Additionally, there are two side band signals for a host/device application. These side band signals Wake-up and Detach are used to control the state from which the notebook resumes. When the host is in a power down mode, Wake-up wakes the host up when the Bluetooth system receives an incoming connection. The host indicates that it is in Suspend mode by using the Detach signal.

I²C Interface

A master I²C I/F is available on the module. The control of the I²C pins are performed by Ericsson specific HCI commands available in the HW implementation - see Appendix C.

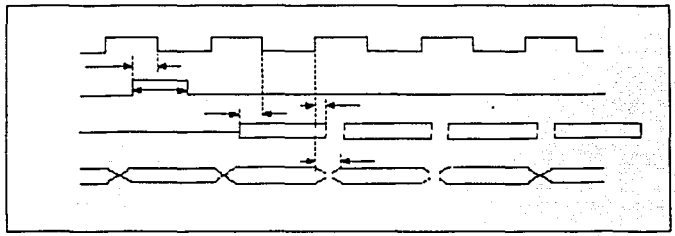


Figure 14. PCM timing

Name	Description	Min	Typ	Max	Unit
[illegible]	PCM data transfer clock frequency	128		2000	kHz
[illegible]	PCM sample rate sync frequency		8		kHz
[illegible]	PCM_CLK high period	200			ns
[illegible]	PCM_CLK low period	200			ns
[illegible]	PCM_SYNC setup to PCM_CLK fall	100			ns
[illegible]	PCM_SYNC hold time	100			ns
[illegible]	PCM_IN setup to PCM_CLK fall	100			ns
[illegible]	PCM_IN hold time PCM_CLK fall	100			ns
[illegible]	PCM_OUT valid from PCM_CLK rise	100			ns

Table 2. PCM parameters

Antenna

The ANT pin should be connected to a 50Ω antenna interface, thereby supporting the best signal strength performance. Ericsson Microelectronics can recommend application specific antennas – see Appendix C.

Power-up Sequence

There is no need for a power up sequence if VCC_ON and VCC_IO are tied together.

A power up sequence, if used, shall be applied accordingly. Connection of the supply rails, GND and then V_____ then the ON signal should be applied in order to initiate the internal regulators and finally the V_____ supply rail can be activated.

The power down sequence is similar to the power up procedure but in the reverse format. Therefore, the disconnection of the signals shall be as follows: V_____, ON, V_____ and finally GND.

RESET#

The assignment of the RESET# input is to generate a reset signal to the complete Bluetooth module. During power up the reset signal is set low automatically so that power supply glitches are avoided. Therefore no reset input should be required after power up.
When implementing an external RESET# the signal should be fed from an open drain output.

Power

There are three inputs to the Voltage Management section: V_____, V_____ ON. V_____ is the supply voltage that is typically 3.3V.

A separate power supply rail (V_____) is provided for the I/O ports, UART, PCM and USB. To be compliant with the USB 1.1 specification V_____ = 3.3 V. V_____ can either be connected to V_____ or to a dedicated supply rail, which is the same as the logical interface of the host.

Shielding / EMC Requirements

The module has its own RF shielding and is approved according to the standards by FCC and ETSI.

* The approval number is printed on the outside label of the module. When used in the final product, an external label must state that there is a transmitter module inside the product.

Ground

Ground should be distributed with very low impedance as a ground plane. Connect all GND pins to the ground plane.

Assembly Guidelines

Solder Paste

The ROK 101 007 module is made for surface mounting and the SSP connection pads have been formed after printing eutectic tin/Lead solder paste. The solder paste to use is not critical as long as this is a normal eutectic solder paste. A preferred solder paste height is 150µm.

Soldering Profile

It must be noted that the module should not be allowed to be placed upside down in the reflow operation. This means that the module has to be assembled on the side of the PCB that is soldered last.

The reflow process should be a regular surface mount soldering profile (full convection soldering preferred). The ramp up should not be higher than 2°C/s and with a peak temperature of 210-235°C during 20-60 seconds.

Pad Size

It is recommended that the pads on the PCB should have a diameter of 0.7-0.9 mm. The surface finish on the PCB pads should be Nickel Gold or a flat Tin/Lead surface or OSP (Organic Surface Protection).

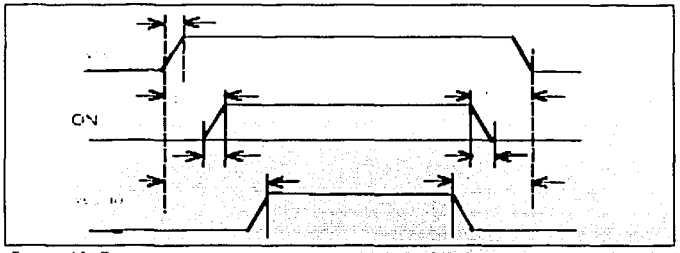


Figure 15. Power up sequence

Parameter	Min	Nom	Max	Unit
t			-	ms
t			60	ms
t			60	ms
t	-			us
t				ms
t	t			us
t	t			us

Table 3. Power up characteristics

Placement

The placement machine should be able to recognize odd BGA combinations (all ball recognition preferred) and be able to pick the component asymmetrical. The module contains a flat pick area of 10mm diameter minimum. The weight of the module is typically 2.8gr.

Storage

Keep the component in its dry pack when not yet using the reel. After removal from the dry pack ensure that the modules are soldered on the PCB within 48 hours.

Marking

Every module is marked with the following information on the

a) Component designation "ROK 101 007"
b) Ericsson's name and logotype.
c) Manufacturing code (pack, year, week and batch number)
d) CE - sign/type
e) Type approval (PTA no. See manual

Packaging

All devices will be delivered in a package protecting them from electrostatic discharges and mechanical shock. The package will be marked with the following information:

a) Delivery address.
b) Purchase order number
c) Type of goods and component designation.
d) Ericsson's name and logotype
e) Date of manufacture and batch number
f) Number of components in the package

Abbreviations

ASIC	Application Specific Integrated Circuit
BER	Bit Error Rate
CMOS	Complementary Metal Oxide Semiconductor
C/I	Carrier to Interference Ratio
DCE	Data Circuit terminating Equipment
GP	Gold Print
HC	Host Controller Interface
ISM	Industrial Scientific and Medical
PCB	Printed Circuit Board
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
PtP	Point to Point
Rx	Receive
SIG	Special Interest Group
SSP	Screen Solder Print
Tx	Transmit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VCO	Voltage Controlled Oscillator

GLOSARIO

- **ACK** (*ACKnowledge*): Bit usado en los paquetes de bandabase de *Bluetooth* para indicar que el paquete anterior ha sido recibido correctamente.
- **ACL** (*Asynchronous Connection Less, asíncrono no orientado a conexión*): Enlaces usados por *Bluetooth* para transmitir datos, a partir de los cuales se establecen los enlaces para transmitir voz (SCO).
- **AM_ADDR** (*Acitve Member ADDRess, dirección de miembro activo*): Dirección asignada por el maestro a cada esclavo activo en una piconet. Es usada para identificar al esclavo en particular al que se transmite la información.
- **Apareamiento** (*pairing*): Procedimiento de seguridad que involucra el intercambio de paquetes del administrador de enlace, con la finalidad de establecer una llave de enlace que se usará entre dos dispositivos Bluetooth.
- **ARQ** (*Automatic Repeat reQuest*): Bit en los paquetes de bandabase de Bluetooth usado para pedir la retransmisión de cualquier paquete recibido con errores.
- **Autenticación**: Procedimiento de seguridad durante el cual, dos dispositivos verifican que ambos poseen la misma llave secreta.
- **Autorización**: Procedimiento de seguridad, donde un dispositivo otorga o recibe permiso para acceder a un determinado servicio.
- **Bandabase**: Parte de la pila de protocolos que controla el sistema de radio.
- **BCH** (Bose, Chaudhuri, Hocquenghem): Familia de códigos de paridad cíclica, los cuales se añaden antes de la transmisión, con la finalidad de detectar y corregir errores en el receptor. Los códigos BCH mejoran las propiedades de autocorrelación en la *palabra de sincronización* de Bluetooth.
- **BD_ADDR** (*Bluetooth Device ADDRess, dirección de dispositivo Bluetooth*): Es un número único usado para identificar a un dispositivo Bluetooth. Se usa en la encriptación y generación de las secuencias de salto.
- **BER** (*Bit Error Rate, Tasa de bits erróneos*): Es un parámetro usado para medir la calidad del enlace.

- **BPSK** (*Binary Phase Shift Keying*): Modulación donde un bit es señalizado mediante dos posibles valores de la fase.
- **Broadcast**: Dentro de la tecnologia Bluetooth representa la acción de transmitir información a todos los dispositivos de la piconet.
- **Buffer:** Componente electrónico que sirve como memoria de datos
- **CAC** (*Channel Access Code, código de acceso del canal*): Es un código derivado de la dirección Bluetooth del Maestro, se usa al principio de los paquetes para identificar su piconet.
- **Canal Lógico**: Tipo de canal que puede ser transportado a través de un enlace físico de bandabase.
- **CID** (*Channel Identifier, identificador de canal*): Es usado en los paquetes L2CAP para identificar qué entidad de capa superior está usando el canal.
- **Cliente**: Un dispositivo que desea usar los servicios de otro dispositivo (servidor). Generalmente el cliente manda peticiones, y el servidor responde.
- **CLK**: Reloj del Maestro que define la temporización en la piconet Bluetooth.
- **CLKE**: Estimación del reloj de otro dispositivo.
- **CLKN**: Reloj nativo de un dispositivo Bluetooth, un Esclavo debe añadir una compensación a su propio CLKN para sincronizarse con el CLK.
- **CoD** (*Class of Device*): Campo en un paquete FHS que identifica la clase de dispositivo Bluetooth que está enviando el paquete.
- **CODEC** (*Coder DECoder*): Circuito para el procesamiento de señales de voz.
- **Correlador**: Circuito que explora los datos entrantes en busca de un patrón en particular. El correlador genera una señal cuando encuentra el patrón buscado.
- **CRC** (*Cyclic Redundancy Checksum*): el parámetro de revisión de redundancia ciclica, es un parámetro generado por un polinomio que permite verificar la integridad de los datos.
- **DAC** (*Device Access Code, código de acceso del dispositivo*): Código que identifica a un dispositivo en particular. Este código se deriva de la dirección Bluetooth del dispositivo y se usa en el proceso de *paging*.
- **Descubrimiento de servicios**: Proceso de descubrimiento de servicios y aplicaciones ofrecidas por otros dispositivos.
- **DH** (*Data High rate*): Categoria de paquetes Bluetooth que alcanzan altas tasas de transmisión reduciendo la revisión de errores.
- **DIAC** (*Dedicated Inquiry Access Code, código de acceso dedicado de investigación*): Código enviado en los paquetes ID por los dispositivos que quieren descubrir a otros dispositivos Bluetooth, que han sido configurados para buscar este tipo de código.
- **Dispositivo Bluetooth**: Equipo que consta de un módulo Bluetooth y un Host Bluetooth.

- **Dispositivo confiable**: Aquel dispositivo que ha sido apareado (comparte una clave de enlace) con otro dispositivo y éste lo a marcado como confiable dentro de una base de datos.
- **DLC** (*Data Link Connection*): Un canal RFCOMM.
- **DLCI** (*Data Link Connection Identifier*): Un número de canal RFCOMM.
- **DM** (*Data Medium rate*): Categoria de paquetes Bluetooth que poseen alta confiabilidad debido a que añaden caracteristicas para la revisión y corrección de errores. Esta confiabilidad es alcanzada a expensas de una baja tasa de transmisión.
- **Duplexaje por división de tiempo** (*TDD, Time Division Duplexing*): Acción de compartir el canal en dos direcciones, dejando que cada dirección transmita en un turno.
- **DV** (*Data Voice*): Tipo de paquetes bluetooth enviados en los enlaces SCO, los cuales pueden llevar tráfico de voz y datos.
- **Enlace físico**: Representa el nivel más bajo de conexión entre dispositivos. Dentro de la tecnología Bluetooth, el enlace físico es un enlace de radio.
- **Esclavo**: Dispositivo enlazado a un Maestro, el Esclavo sigue la secuencia de salto en frecuencia y la temporización del Maestro. El Esclavo sólo tienen permitido transmitir cuando después que Maestro le haya enviado un paquete o en las ranuras SCO reservadas.
- **Evento**: Mensaje enviado del módulo Bluetooth al Host a través de la interfase de control del host (HCI).
- **FCS** (*Frame Check Sequence, secuencia de revisión de trama*): Secuencia usada para detectar errores en un paquete.
- **FEC** (*Forward Error Correction, corrección de errores por adelantado*): Código de corrección de errores usado para proteger los datos en algunos paquetes Bluetooth.
- **FHS** (*Frequency Hop Synchonisation, sincronización de salto en frecuencia*):Paquete usado para pasar la información necesaria que permita a un dispositivo Bluetooth sincronizarse con la secuencia de salto de otro.
- **FHSS** (*Frecuency Hop Spread Spectum, espectro disperso con saltos en frecuencia*): Técnica de modulación que esparce los datos a lo largo de toda la banda de transmisión, mediante el envio de datos sucesivos sobre diferentes canales. A lo anterior se le conoce como *saltos en frecuencia.*
- **FSK** (*Frequency Shift Keying*): Modulación donde los valores son representados por un cambio en frecuencia.
- **Gateway**: Dispositivo que actúa como intermediario, permitiendo que sistemas incompatibles puedan comunicarse.
- **GIAC** (*General Inquiry Access Code*, código de acceso general de investigación): Código estándar fijo usado para investigar la presencia de otros dispositivos Bluetooth.

- **GSM** (*Global System for Mobile comunications, sistema global para comunicaciones móviles*): Estándar de telefonía celular digital.
- **GSM07.10**: Estándar para la emulación de múltiples cables seriales usado en los sistemas GSM.
- **HCI** (*Host Controller Interface, interface de control de host*): Interface que enlaza un host Bluetooth co un módulo Bluetooth. Datos, comandos y eventos pasan a través de esta interfase.
- **Headset**: Dispositivo que provee una entrada y salida de audio remoto para un gateway de auido. Este dispositivo consta de un auricular y un micrófono.
- **HEC** (*Header Error Check, revisión de error de cabecera*):Código corto usado por el dispositivo receptor para detectar la presencia de errores en la cabecera del paquete.
- **HOLD**: Modo donde el dispositivo Bluetooth está inactivo un periodo de tiempo.
- **Host** (*anfitrión*):Dispositivo que implementa las capas altas de Bluetooth y controla un módulo independiente que provee las funciones de las capas bajas. Por ejemplo, si una computadora posee una tarjeta Bluetooth, la computadora es el host y la tarjeta Bluetooth es el módulo.
- **HTML** (*HyperText Markup Languaje*): Lenguaje usado para definir páginas en la red mundial WWW.
- **HTTP** (*HyperText Transfer Protocol*): Protocolo usado para transferir páginas WEB escritas con HTML. Este protocolo es usado para seguir los enlaces entre páginas WEB.
- **HV** (*High quality Voice, voz de alta calidad*): Paquetes usados para transportar voz sobre enlaces de audio (SCO) Bluetooth. Existen tres tipos de paquetes HV: HV1, HV2 Y HV3. Estos son enviados cada par de ranuras, cada dos pares o cada tres pares, respectivamente.
- **IAC** (*Inquiry Access Code, código de acceso de investigación*): Código enviado en paquetes ID por los dispositivos que quieren descubrir otros dispositivos Bluetooth en el área.
- **IEEE** (*Institute of Electronic and Electrical Engineers, instituto de ingeniero eléctricos y electrónicos*): Organización que promueve la ingeniería eléctrica en todo el mundo. IEEE ha desarrollado muchos estándares entre ellos el 802.11 de redes LAN inalámbricas.
- **Implementar**: Llevar a cabo, realizar, aplicar, poner en práctica.
- **Inquiry** (*Investigación*):Procedimiento en el cual un dispositivo Bluetooth transmite mensajes de investigación o inquiry para encontrar a otros dispositivos dentro del área de influencia de su radio.
- **Inquiry scan** (*exploración de investigación*): Procedimiento mediante el cual los dispositivos exploran en busca de mensajes inquiry, a los cuales responden con la finalidad de dar a conocer su presencia.

- **IP** (*Internet Protocol, protocolo de internet*): Protocolo que provee de direccionamiento, ruteo, segmentación y reensamblaje.
- **IrDA** (*Infrared Data Association*): Organización que define los protocolos de comunicación para infrarrojos, usados en muchas laptops y teléfonos celulares para el intercambio de datos a corta distancia.
- **IrOBEX**: Especificación IrDA para el intercambio de objetos mediante infrarrojo.
- **ISM** (*Industrial, Scientific and Medical; Industrial, Científica y Médica*): Bandas de frecuencias disponibles sin licencia, ubicadas en los rangos: 902-908 MHz, 2.4-2.5 GHz, 5.8-5.9 GHz.
- **Isócrono**: Información que debe ser transmitida dentro de un tiempo fijo. El video comprimido es isócrono, debido a que su calidad se afecta si el retardo es variable.
- **L_CH** (*Logical Channel*): Par de bits en los paquetes de bandabase Bluetooth que se usan para identificar si un paquete ACL contiene el principio o la continuación de un paquete L2CAP; o si se trata de un paquete LMP.
- **L2CA** (*Logical Link Control and Adaptation, control y adaptación de enlace lógico*): Capa de la pila de Bluetooth donde se implementa L2CAP.
- **L2CAP** (*Logical Link Control and Adaptation Protocol, protocolo de control y adaptación de enlace lógico*): Protocolo que provee servicios de segmentación y reensamblaje que permite a paquetes largos pasar a través de los enlaces Bluetooth, también provee multiplexaje para los servicios y las capas superiores.
- **LAP** (*Lower Address Part*): Son los 24 bits menos significativos de la dirección Bluetooth del dispositivo.
- **Laptop**: Computadora portátil.
- **LC** (*Link Controller, controlador de enlace*): Capa de la pila de Bluetooth que controla los enlaces ACL y SCO, decidiendo que paquete se enviará. También maneja las funciones de retransmisión y de secuencia de paquetes.
- **LIAC** (*Limited Inquiry Access Code, código de acceso limitado de investigación*): Código de acceso de investigación que los dispositivos acuerdan usar por un corto periodo, en lugar de código de acceso general (GIAC).
- **LM** (*Link Manager, administrador de enlace*): Capa de Bluetooth donde se implementa LMP.
- **LMP** (*Link Manager Protocol, protocolo de administración de enlace*): Protocolo que maneja la configuración y el control de los enlaces bandabase de Bluetooth.
- **Maestro**: Dispositivo que controla y coordina un grupo de dispositivos Bluetooth. el Maestro establece la secuencia de saltos en frecuencia y la temporización, además otorga permiso a los Esclavos para transmitir.

- **Módulo Bluetooth**: Unidad que implementa las capas bajas de la pila Bluetooth hasta el HCI.
- **MTU** (*Maximum Transmisión Unit, unidad máxima de transmisión*): Es el tamaño más largo de paquete que una determinada capa puede manejar.
- **Multiplexaje por división de tiempo** (*TDM, Time División Multiplexing*): Acción de dividir el canal de comunicaciones entre diversos usuarios, de tal manera que primero lo use uno, después otro y así sucesivamente. Para llevar a cabo esta técnica se requiere el uso de ranuras de tiempo.
- **NAK** (*Negative ACKnowledgement*): Bit usado para señalar que un paquete ha sido recibido con errores, o que un paquete esperado no ha sido recibido.
- **NAP** (*Nonsignificant Address Part, parte no significativa de la dirección*): Son 16 bits de la dirección Bluetooth. Se encuentran en la parte media, entre LAP y UAP.
- **Números pseudoaleatorios**: Una serie de números que aparentan ser aleatorios, sin embargo, se repiten después de un largo periodo.
- **OBEX** (*Object Exchange protocol, protocolo de intercambio de objetos*): Protocolo que permite a los dispositivos intercambiar objetos de datos arbitrarios.
- **OpCode** (*Operational Code, código operacional*): Código usado en los paquetes para identificar que tipo de información lleva el mensaje.
- **OSI** (*Open Systems Interconnect*): Modelo de referencia cuya finalidad es garantizar la interoperabilidad entre sistemas.
- **Page** (*llamado*): Conjunto de procedimientos mediante los cuales se establece conexión.
- **Page scan** (*exploración de llamado*): Acción mediante la cual un dispositivo Bluetooth escucha sus propios ID. Si un dispositivo escucha su propio ID, entonces se conecta con el dispositivo que lo envió.
- **Paging**: Acción de transmitir el ID de otro dispositivo para establecer conexión con éste.
- **PAN** (*Personal Area Network, red de área personal*): Término que describe una red pequeña tipo *ad hoc* implementada a través de Bluetooth, la cual agrupa un conjunto de dispositivos dentro del espacio personal de operación, típicamente hasta 10 m.
- **Paquete**: Conjunto estructurado de datos que se transfiere como una unidad.
- **PARK**: Modo donde un Esclavo Bluetooth se encuentra activo solamente durante determinadas ranuras. Durante estas ranuras, el Esclavo puede solicitar su regreso al modo de conexión activa, o bien, el Maestro puede ordenarle abandonar el modo de Park.
- **Payload** (*tibutaria*): Representa la carga útil dentro de un paquete.
- **PDA** (*Personal Digital Assistant, asistente digital personal*): Dispositivo de cómputo pequeño y portátil. Su principal función es la de un organizador personal, incluye las características

de agenda, almacenamiento de direcciones, correo electrónico, almacenamiento de documentos. Ejemplo de un PDA es una Palm.

- **PDU** (*Protocol Data Unit, unidad de datos de protocolo*): Paquete de datos usado para intercambiar información en un formato especificado por un protocolo de comunicaciones.
- **Perfil** (*profile*): Dentro de Bluetooth representa un conjunto de reglas que indican como usar la pila de protocolos en un dispositivo. El objetivo de los perfiles es asegurar una interoperabilidad entre diversos dispositivos.
- **Piconet**: Grupo de dispositivo reunidos en una red de corto alcance implementada mediante la tecnología Bluetooth. Este grupo está sincronizado a la temporización y secuencia de salto de un dispositivo llamado: Maestro.
- **Pila de protocolo** (*protocol stack*): Un conjunto de unidades funcionales divididas en capas, entre las cuales implementan un protocolo. Cada capa de un protocolo tiene bien definidas sus tareas y responsabilidades, así como también las interfases con las capas vecinas.
- **PIN**: (*Personal Identification Number, número de identificación personal*): Número introducido por el usuario o almacenado dentro del dispositivo, el cual se usa para revisiones de seguridad.
- **POLL**:
- **Protocolo**: Conjunto de reglas que determinan un comportamiento. Un protocolo de comunicaciones es un conjunto de reglas que indican cómo debe comportarse un dispositivo o sistema con respecto a otro cuando se comunican. Es esencial que los dispositivos sigan el mismo protocolo si éstos pretenden comunicarse apropiadamente.
- **Proxy**: dispositivo que actúa como intermediario permitiendo que sistemas incompatibles originalmente, se comuniquen. Un proxy representa al dispositivo para el cual está traduciendo, es por ello que los dispositivos que están hablando con el proxy no están enterados de su presencia y actúan como si estuvieran comunicándose directamente.
- **PSK** (*Phase Shift Keying*): Técnica de modulación que representa un bit por medio de una desviación en fase.
- **PSM** (*Protocol/Serice Multipexer, multiplexor de protocolo/sevicio*): Usado por L2CAP para identificar al tipo de entidad de capa superior que esta usando la conexión L2CAP.
- **PSTN** (*Public Switched Telephony Network*): Red telefónica pública conmutada
- **QoS** (*Quality of Service, calidad de servicio*): Define el ancho de banda y retardo que un dispositivo puede esperar en una conexión.
- **Ranura** (*slot*): Ranura de tiempo. Lapso de tiempo utilizado para transmitir o recibir datos, en la tecnología Bluetooth una ranura tiene una duración de 625 μs. El uso de ranuras es necesario en el multiplexaje por división de tiempo para compartir el canal de comunicaciones entre varios dispositivos.

- **Registro de servicio**: Información acerca de los servicios que es necesaria para conectarse a ellos y usarlos. El SDP es se usa para acceder al registro de servicios.
- **RF** (*Radio Frequency*): Radio frecuencia.
- **RFCOMM**: Protocolo para la emulación de un cable serial RS-232.
- **RSSI** (*Recive Signal Strength Indication, indicación de magnitud de la señal recibida*): Se usa para determinar si la potencia en el extremo lejano del enlace de radio debe ser incrementada o disminuida.
- **Rx**: Receptor o recepción.
- **Scatternet**: Grupo de piconets reunidas por algún dispositivo que pertenece a más de una piconet.
- **SCO** (*Synchronous Connection-Oriented, síncrono orientado a conexión*): Enlaces usados por Bluetooth para enviar voz, estos enlaces usan ranuras reservadas.
- **SDP** (*Service Discovery Protocol, protocolo de descubrimiento de servicios*): Protocolo Bluetooth que permite a un cliente descubrir los servicios que presta un servidor.
- **Secuencia de salto**: Se llama así a la secuencia pseudo-aleatoria de frecuencias sobre la cual los dispositivos Bluetooth sintonizan sus sistemas de radio. En una piconet activa, la secuencia de salto es establecida por el reloj y la dirección del Maestro. Existen diversos tipos de secuencias, usadas para inquiry, paging, inquiry scan y page scanning
- **SEQN**: Bit de secuencia que permite que el dispositivo receptor descubra la pérdida de paquetes debido a interferencia de radio o desvanecimiento de la señal.
- **Servicio**: Es una característica de un determinado dispositivo que puede ser usada por otro.
- **Servidor**: Dispositivo que ofrece servicios que serán usados por otro dispositivo llamado: cliente. Generalmente, el cliente envía peticiones y el servidor las responde.
- **SIG** (*Special Interes Group, grupo de interés especial*): El SIG de Bluetooth es un grupo constituido por varias compañías encargadas de definir y promocionar las especificaciones de Bluetooth.
- **Sincronización**: Conjunto de procedimientos que tienen la finalidad que dos o más dispositivos o sistemas trabajen con una misma base de tiempo.
- **SNIFF**: Es un modo de bajo consumo de energía donde el dispositivo sólo se encuentra activo en ciertas ranuras llamadas: *ranuras sniff*.
- **Temporización** (*timming*): Secuencia de periodos de tiempo en que se realizan acciones diferentes.
- **Timer** (*temporizador*): Reloj de cuenta regresiva que indica que debe producirse una determinada acción al finalizar dicha cuenta.
- **Tx**: Transmisor o transmisión.

- **UAP** (*Upper Address Part, parte alta de la dirección*): Son los 8 bits más significativos de la dirección Bluetooth.
- **UART** (*Universal Asynchronous Receiver Transmiter*): Chip usado en las interfases seriales para transmitir y recibir datos.
- **USB** (*Universal Serial Bus*): Estándar de conexión de alta velocidad ampliamente usado para conectar dispositivos a la PC.
- **UUID** (*Universal Unique IDentifier, identificador universal único*): número de 128 bits generado a través de un método que garantiza su unicidad.
- **Vinculación** (*Bonding*): Procedimiento de alto nivel usado para establecer una relación confiable entre dos dispositivos. Una vez vinculados dos dispositivos, estos pueden intercambiar información sobre un enlace encriptado.
- **WLAN** (*Wireless Local Area Network*): Red inalámbrica de área local.

BIBLIOGRAFÍA

Libros

1. Bernard Sklar, "Digital Communications Fundamentals and Aplications", Prentice Hall, E.U., 1988.
2. Roger L. Peterson, et. al., "Introduction To Spread Spectrum Communications", Prentice Hall, E.U., 1995.
3. John G. Proakis, et. al., "Contemporary Communication Systems", PWS Publishing Company, E.U., 1997.
4. Sami Tabbane, "Handbook Of Mobile Radio Networks", Artech House Publishers, E.U., 2000.
5. Jennifer Bray, Charles F. Sturman, "Bluetooth 1.1, Connect Without Cables", Prentice-Hall, E.U., 2002.
6. John Walker, "Advances In Mobile Information Systems", Artech House Publishers, E.U., 1998.
7. Walke H. Bernhard, "Mobile Radio Networks Networking and Protocols", John Wiley and Sons, E.U. 1999
8. Saleem N. Bahatti, "Wireless and Mobile Connectivity", Apuntes University College London, U.K., 2001.

Artículos

9. Johansson Per, Sörensen Ericsson Johan, "Ad-hoc IP Networks over Bluetooth", IEEE Communications Magazine, 2001.
10. Richard C. Braley, Ian Gifford, "Wireless Personal Area Networks: An Overview of the IEEE P802.15 Working Group", IEEE transactions, 2000.
11. Naveen Chandran, Matthew C. Valenti, "The generations of cellular wireless systems", IEEE Communications Magazine, 2001.
12. Ahlgren Bengt, Feeney Laura, Westerlund Assar, "Spontaneous networking: An application-oriented approach to ad hoc networking", IEEE Communications Magazine, Junio 2001.

13. Toh C.K., "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networking", IEEE Comunications Magazine, Junio 2001.
14. Macker Joseph, Park Vincent, "Mobile and wireless internet services: putting the pieces together", IEEE Comunications Magazine, Junio 2001.
15. Chandrashekhar M.V.S., Choi Pedro, Maver Keith, "Evaluation of interference between IEEE 802.11b and Bluetooth in a typical office environment", IEEE Transactions 2001
16. Chatschik Bisdikian, "An overview of the Bluetooth Wireless Technology", IEEE Communications Magazine. Diciembre 2001.
17. Thomas M Siep, Ian C. Guifford, "Paving the Way for Personal Area Network Standards: An Overview of the IEEE P802.15 Working Group for Wireless Personal Area Networks", IEEE Personal Communications, Febrero 2000.
18. Per Johansson, Mantho Kazantzidis, "*Bluetooth*: An Enabler for Personal Area Networking", IEEE Network, Octubre 2001.
19. Ian C. Gifford, "*Bluetooth*, A Cable Replacement Overview", IEEE Communications Magazine, Agosto 2000.

Tutoriales y manuales

20. Ian Gifford, "Wireless Personal Area Network, Study Group", IEEE, 1999.
21. "Manual de usuario módulos Bluetooth", Ericsson, 2000.
22. Bluetooth Special Interest Group, "Specification Volume 1, Specification of the Bluetooth System, Core", versión 1.1, 2001.
23. Bluetooth Special Interest Group, "Specification Volume 1, Specification of the Bluetooth System, Profiles", versión 1.1, 2001.
24. Aman Kansal, "Bluetooth What and Why?", http://www.ee.iitb.ac.in/uma/~aman/bluetooth/tut1.html
25. Bob Heile, "Working Group for Wireless Personal Area Networks", IEEE, 1999.

Referencias electrónicas.

26. Palowireless, "Bluetooth Glossary", http://www.palowireless.com/infotooth/glossary.as
27. USB organization, "USB 2.0 Specifications", http://www.usb.org/developers/docs.html, versión 2.0, 2000.