ANEXO A. STACK DE PROTOCOLOS BLUETOOTH

TABLA DE CONTENIDO

ANEXO A STACK DE PROTOCOLOS BLUETOOTH

Como se puede observar en la Figura A.1, la comunicación sobre *Bluetooth* se divide en varios niveles.

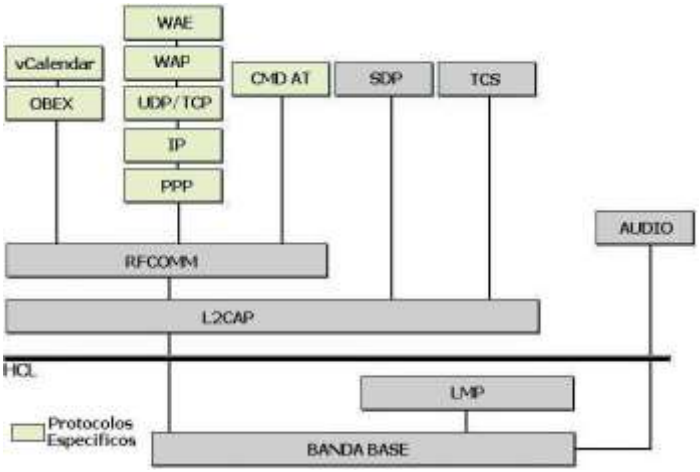

Figura A.1 Stack de Protocolos Bluetooth

Existen protocolos específicos para Bluetooth y otros que han sido adoptados con el fin de facilitar la adaptación de aplicaciones que utilizan estos protocolos ya existentes. De acuerdo a su propósito, la pila de protocolos de Bluetooth se puede dividir en cuatro capas como se muestra en la tabla A.1:

Grupo de Protocolos	Protocolos en la Pila
Protocolos Centrales Bluetooth	Banda Base, Protocolo de Gestión del Enlace, L2CAP y SDP
Protocolo Reemplazo de Cables	RFCOMM
Protocolo de Control Telefónico	TCS Binario
Protocolos Adoptados	PPP, UDP/TCP/IP, OBEX, WAP

Tabla A.1 Protocolos y capas en la pila de protocolos Bluetooth

Equivalencia entre el Modelo de Referencia OSI y Protocolos de Bluetooth

La Figura A.2 ilustra una comparación del stack de protocolos de *Bluetooth* con el modelo de referencia estándar *Open Systems Interconect OSI,* para stacks de protocolo de comunicaciones. A pesar de que *Bluetooth* no concuerda exactamente con el modelo, esta comparación es muy útil para relacionar las diferentes partes del stack *Bluetooth* con las capas del modelo *OSI.* Dado que el

modelo de referencia es un stack ideal y bien particionado, la comparación sirve para resaltar la división de funciones en el stack *Bluetooth.*

Figura A.2 Comparación entre el stack de protocolos de *Bluetooth y el Modelo de Referencia OSI*

- El *nivel físico* es responsable de la interfaz eléctrica al medio de transmisión e incluye también procedimientos como la modulación y la codificación del canal. Esta parte está bajo responsabilidad de la interfaz de *Radio* y parte de la *Banda Base* en el conjunto de protocolos de Bluetooth.
- La capa de *enlace* es responsable de la transmisión, formación de tramas y control (detección y corrección) de errores sobre un enlace particular. Estas funciones recaen sobre el segmento de control de *Banda Base* y la capa denominada *Link Manager*.
- La capa de *red* es responsable de la transferencia de datos a través de la red, independientemente del medio de transmisión y la topología. De estas funciones se encargan la parte superior del *Link Controler* y parte adicional del *Link Manager*.
- La capa de *transporte* debe garantizar la confiabilidad en la transferencia de la información y multiplexación de los datos a través de la red según el nivel determinado por la aplicación. En Bluetooth estas funciones estarán a cargo de una porción del *Link Manager* y la capa llamada *HCI* la cual posee los mecanismos para el transporte de datos.
- La capa de *sesión* proporciona servicios de gestión y control sobre el flujo de datos lo cual se logra mediante la implementación de la capa *L2CAP* y la parte inferior de las capas *RFCOM*M y *SDP*.
- La capa de *presentación* proporciona una representación común de los datos a la capa de aplicación agregando la estructura de servicio a las unidades de información. Esta es tarea del *RFCOMM* y *SDP*. Y finalmente la capa de aplicación es la que hace uso de la información transmitida entre dos unidades Bluetooth.

En las siguientes secciones se describen uno a uno los niveles de protocolos exclusivos de Bluetooth. Los protocolos que no son propios de la pila de protocolos no están cubiertos por este documento.

A.1 BANDA BASE

Bluetooth soporta un canal de datos asíncrono de hasta tres canales de voz simultáneos. El canal asíncrono soporta comunicación simétrica y asimétrica. En la comunicación asimétrica pueden ser enviados 723.3 kb/s desde el servidor y 57.6 kb/s hacia el servidor, mientras que en la comunicación simétrica pueden ser enviados 433 kb/s en ambas direcciones. *Bluetooth* brinda conexión *punto-a-punto* o conexión *punto-a-multipunto*. Dos o más unidades compartiendo el mismo canal forman una *piconet.* Cada *piconet* debe tener un maestro y puede tener hasta siete esclavos activos, además pueden haber muchos más esclavos en estado *parked*. Estos esclavos no están activos en el canal sin embargo están sincronizados con el maestro con el fin de asegurar una rápida iniciación de comunicación. La interconexión de varias *piconets* forma una *scatternet*[1]*.* En la Figura A.3 se puede observar una *piconet* donde el PC actúa como maestro y los otros dispositivos son conectados como esclavos.

Figura A.3 Diagrama de una Piconet

A.1.1 Canal físico

El canal físico contiene *79 frecuencias de radio diferentes*, las cuales son accedidas de acuerdo a una secuencia de saltos aleatoria. La velocidad de saltos estándar es de *1600 saltos/s.* El canal está dividido en *timeslots* (ranuras de tiempo), cada *slot* (ranura) corresponde a una frecuencia de salto y tiene una longitud de 625 us. Cada secuencia de salto en una *piconet* está determinada por

[1] Los conceptos de Piconet y Scatternet se explican en el Capitulo1.

la dirección del maestro de la *piconet.* Todos los dispositivos conectados a la *piconet* están sincronizados con el canal en salto y tiempo. En una transmisión, cada paquete debe estar alineado con el inicio de un *slot* y puede tener una duración de hasta cinco *timeslots***.** Durante la transmisión de un paquete la frecuencia es fija. Para evitar fallas en la transmisión, el maestro inicia enviando en los *timeslots* pares y los esclavos en los *timeslots* impares. En la Figura A.4 se puede observar este esquema de transmisión.

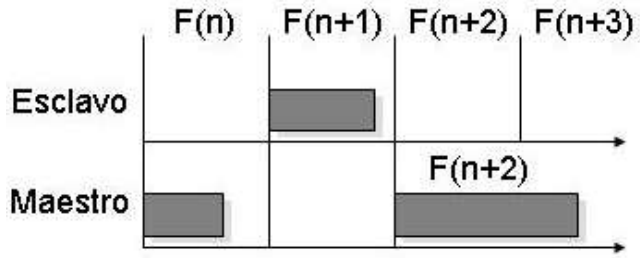

Figura A.4 Transmisión en una Piconet

A.1.2 Enlace físico

La comunicación sobre *Bluetooth* esta especificada tanto para enlaces *SCO como para enlaces ACL*. El enlace *SCO* es una conexión simétrica *punto-a-punto* entre el maestro y un esclavo específico. Para lograr la comunicación, el enlace *SCO* reserva *slots* en intervalos regulares en la iniciación, por esto el enlace puede ser considerado como una conexión de conmutación de circuitos. El enlace *ACL* es un enlace *punto-a-multipunto* entre el maestro y uno o más esclavos activos en la *piconet*. Este enlace de comunicación es un tipo de conexión de conmutación de paquetes. Todos los paquetes son retransmitidos para asegurar la integridad de los datos. El maestro puede enviar mensajes *broadcast* (de difusión) a todos los esclavos conectados dejando vacía la dirección del paquete, así todos los esclavos leerán el paquete.

A.1.3 Paquetes

Los datos enviados sobre el canal de la *piconet* son convertidos en paquetes, éstos son enviados y el receptor los recibe iniciando por el *bit* menos significativo. Como se observa en la Figura A.5, el formato de paquete general consta de tres campos: *código de acceso, cabecera y carga útil*.

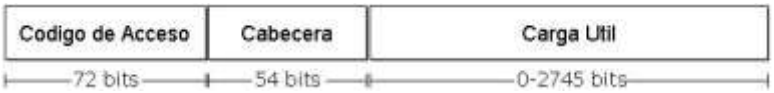

Figura A.5 Formato de Paquete

- Código de acceso Es usado para sincronización e identificación. Todos los paquetes comunes que son enviados sobre el canal de la *piconet* están precedidos del mismo código de acceso al canal. Existen tres tipos diferentes de código de acceso:
    - *Código de acceso al canal* - Para identificar los paquetes sobre el canal de la *piconet.*
    - *Código de acceso de dispositivo* – Para procedimientos de señalización especiales, *paging* (servicio para transferencia de señalización o información en un sentido), entre otros.
    - *Código de Acceso de Búsqueda (IAC)* – llamado *IAC general* cuando se quiere descubrir a otras unidades *Bluetooth* dentro del rango, o *IAC dedicado* cuando se desea descubrir unidades de un tipo específico.

- Cabecera de paquete Como se observa en la Figura A.6, la *cabecera del paquete* consta de seis campos:
    - *Dirección* – Una dirección de dispositivo para distinguirlo de los demás dispositivos activos en la *piconet.*
    - *Tipo* – Define qué tipo de paquete es enviado.
    - *Flujo* – El *bit* de control de flujo es usado para notificar al emisor cuándo el buffer del receptor está lleno.
    - *ARQN – Acknowledge Receive Data* o reconocimiento de datos recibidos.
    - *SEQN – Sequential Numbering* o numeración secuencial para ordenar los datos sobre el canal.
    - *HEC* – Chequeo de redundancia cíclica de cabecera.

Figura A.6 Formato de la Cabecera del Paquete

- Carga útil La carga útil de un paquete puede ser dividida en dos campos:
    - *Campo de Voz* – Consta de datos de voz de longitud fija y existe en paquetes de alta calidad de voz y paquetes combinados de datos/voz. No es necesaria ninguna cabecera de carga útil.
    - *Campo de Datos* – Consta de tres partes, cabecera de carga útil, datos de carga útil, y código *CRC.*

A.1.4 Corrección de errores

En una comunicación *Bluetooth* existen varios esquemas de corrección de errores:

- En la cabecera, cada *bit* es repetido tres veces.
- En la carga útil se usa un esquema de *código Hamming*. Los *bits* de información son agrupados en secuencias de 10 *bits*, éstos son enviados como 15 *bits* y el algoritmo corrige todos los errores de un *bit* y detecta los errores de dos *bits.*
- Para garantizar una recepción correcta, todos los paquetes de datos son retransmitidos hasta que el emisor reciba una confirmación. La confirmación es enviada en la cabecera de los paquetes retornados.
- Los paquetes *broadcast* son paquetes transmitidos desde el maestro a todos los esclavos. No hay posibilidad de usar confirmación para esta comunicación, sin embargo, para incrementar la posibilidad de recibir correctamente un paquete, cada *bit* en el paquete es repetido un número fijo de veces.
- El chequeo de redundancia cíclica (*CRC*) se usa para detectar errores en la cabecera. La suma de comprobación *CRC* está contenida en el campo *HEC* de la cabecera de paquete. Los chequeos de redundancia cíclica también se aplican sobre la carga útil en la mayoría de los paquetes.
- Para asegurar que no desaparezcan paquetes completos, *Bluetooth* usa números de secuencia. Actualmente sólo se usa un número de secuencia de un *bit*.

A.1.5 Transmisión/Recepción

Como se mencionó antes, el maestro de la *piconet* empieza enviando en los *timeslots* pares y el esclavo en los impares. Solamente el último esclavo direccionado está autorizado para enviar en el *timeslot* de los esclavos. Esto no causa problemas ya que el maestro siempre está inicializando todas las conexiones y transmisiones nuevas. Cada esclavo espera las oportunidades de conexión dadas por el maestro. Los paquetes pueden ser más grandes que un *timeslot***,** debido a esto el maestro puede continuar enviando en los *timeslots* impares y viceversa. El sistema de reloj del maestro sincroniza a toda la *piconet.* El maestro nunca ajusta su sistema de reloj durante la existencia de una *piconet,* son los esclavos quienes adaptan sus relojes con un *offset* de tiempo con el fin de igualarse con el reloj del maestro. Este *offset* es actualizado cada vez que es recibido un paquete desde el maestro.

A.1.6 Control de Canal

El control de canal describe cómo se establece el canal de una *piconet* y cómo las unidades pueden ser adicionadas o liberadas en la *piconet.* La dirección del maestro determina la secuencia

de saltos y el código de acceso al canal. La *fase* de la *piconet* está determinada por el sistema de reloj del maestro. Por definición, la unidad *Bluetooth* que inicia la conexión representa al maestro.

En *Bluetooth*, la capa de control de enlace se divide en dos estados principales:
Standby y conexión. Además existen siete sub-estados: *page*, *page scan*, *inquiry* (indagación)**,** *inquiry scan, respuesta de maestro*, *respuesta de esclavo* y *respuesta a inquiry*. Los sub-estados son usados para agregar nuevos esclavos a una *piconet.* Para moverse de un estado a otro se usan comandos de capas más altas o señales internas.
En *Bluetooth* se define un procedimiento de búsqueda que se usa en aplicaciones donde la dirección del dispositivo de destino es desconocida para la fuente. Esto puede ser usado para descubrir qué otras unidades *Bluetooth* están dentro del rango. Durante un sub-estado de *inquiry* o indagación, la unidad de descubrimiento recoge la dirección del dispositivo y el reloj de todas las unidades que respondan al mensaje de búsqueda, entonces la unidad puede iniciar una conexión con alguna de las unidades descubiertas. El mensaje de búsqueda difundido por la fuente no contiene información de ella, sin embargo, puede indicar qué clase de dispositivos deberían responder. Una unidad que permita ser descubierta, regularmente entra en un sub-estado de *inquiry scan* para responder a los mensajes de búsqueda.

Existen dos formas de detectar otras unidades. La primera, detecta todas las otras unidades en el rango de cobertura, y la segunda, detecta un tipo específico de unidades. Los esclavos que se encuentran en el sub-estado de *page scan*, escuchan esperando su propio código de acceso de dispositivo. El maestro en el sub-estado *page* activa y conectan a un esclavo. El maestro trata de capturar al esclavo transmitiendo repetidamente el código de acceso de dispositivo en diferentes canales de salto. Debido a que los relojes del maestro y del esclavo no están sincronizados, el maestro no sabe exactamente cuándo y en qué frecuencia de salto se activará el esclavo.
Después de haber recibido su propio código de acceso de dispositivo, el esclavo transmite un mensaje de respuesta. Este mensaje de respuesta es simplemente el código de acceso de dispositivo del esclavo. Cuando el maestro ha recibido este paquete, envía un paquete de control con información acerca de su reloj, dirección, clase de dispositivo, etc... El esclavo responde con un nuevo mensaje donde envía su dirección. Si el maestro no obtiene esta respuesta en un determinado tiempo, él reenvía el paquete de control. Si el esclavo excede el tiempo de espera, entonces retorna al sub-estado de *page scan.* Si es el maestro quien lo excede, entonces retorna al sub-estado de *page* e informa a las capas superiores.

Cuando se establece la conexión, la comunicación inicia con un paquete de sondeo desde el maestro hacia el esclavo. Como respuesta se envía un nuevo paquete de sondeo y de esta forma

se verifica que la secuencia de salto y la sincronización sean correctas. La Figura A.7 muestra la inicialización de la comunicación sobre el nivel *banda base*.

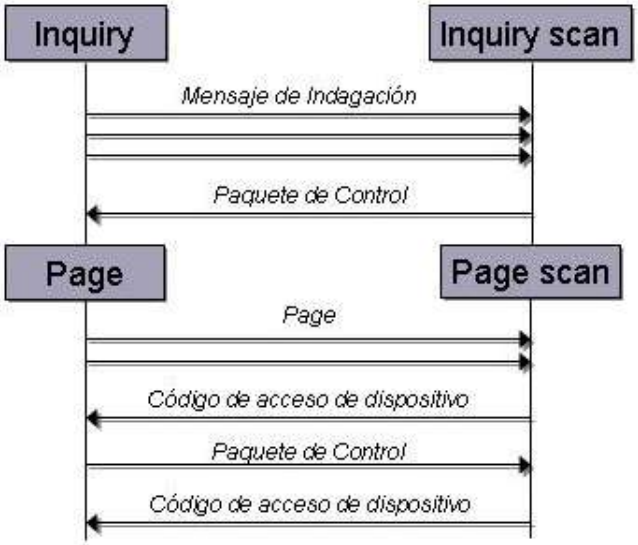

Figura A.7 Inicialización de la comunicación sobre el nivel Banda Base

Cada *transceiver* (receptor-transmisor) *Bluetooth* tiene una única dirección de dispositivo de 48 bits asignada, la cual está dividida en tres campos: campo *LAP*, campo *UAP* y campo *NAP*. Los campos *LAP y UAP* forman la parte significativa del código de acceso. En la Figura A.8 se puede observar el formato de la dirección para un dispositivo *Bluetooth.* La dirección del dispositivo es conocida públicamente y puede ser obtenida a través de una rutina *inquiry.*

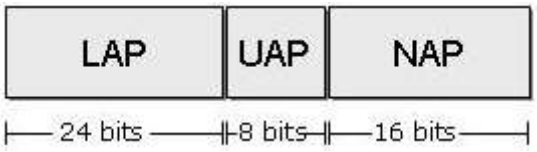

Figura A.8 Formato de la dirección en un dispositivo Bluetooth

A. 1.1.7 Seguridad en Bluetooth

Con el fin de brindar protección y confidencialidad a la información, el sistema debe ofrecer medidas de seguridad en las dos capas, la de aplicación y de enlace. Todas las unidades *Bluetooth* tienen implementadas las mismas rutinas de autenticación y encriptación. En la capa de enlace, estas rutinas constan de cuatro entidades diferentes: *una única dirección pública, dos llaves secretas y un número aleatorio* el cual es diferente para cada transacción. *Solamente es encriptada la carga útil.* El *código de acceso* y la cabecera de paquete nunca son *encriptados*.

Cada tipo de unidad *Bluetooth* tiene una rutina de *autenticación* común. El maestro genera un número aleatorio y lo envía al esclavo, el esclavo usa este número y su propia identidad para calcular el número de autenticación. Luego, este número es enviado al maestro quien hace el mismo cálculo. Si los dos números generados son iguales entonces la autenticación es concedida. La *encriptación,* frecuentemente está restringida por leyes de varios países. Para evitar estas limitaciones, en *Bluetooth* la llave de encriptación no es estática, ésta es deducida de la llave de autenticación cada vez que se activa la encriptación.

A.2 PROTOCOLO DE GESTIÓN DE ENLACE (LMP)

En el *protocolo de gestión de enlace*, *LMP,* se usan mensajes asociados con el establecimiento, seguridad y control. Los mensajes son enviados en la carga útil y no en los mensajes de datos de *L2CAP*. Los mensajes *LMP* son separados de los demás por medio de un valor reservado en uno de los campos de la cabecera de carga útil. Todos los mensajes *LMP* son extraídos e interpretados por la capa *LMP* del receptor, esto significa que ningún mensaje es enviado a capas superiores. Los mensajes *LMP* tienen mayor prioridad que los datos de usuario, esto significa que si la *gestión de enlace* necesita enviar un mensaje, éste no debe ser retrasado por otro tráfico. Solamente las retransmisiones de los paquetes del nivel de banda base pueden retrasar los mensajes *LMP.* Además, éstos no necesitan rutinas de reconocimiento ya que la capa banda base asegura un enlace confiable. El *protocolo de gestión enlace* soporta mensajes para:

- Autenticación
- Paridad
- Encriptación
- Temporización y sincronización
- Versión y características
- Switch para desempeño como maestro o esclavo dependiendo de si el dispositivo es quien inicia (maestro) o no (esclavo) el enlace con otro dispositivo.
- Petición de nombre
- Desconexión
- Modo *hold*: el maestro ordena al esclavo entrar en este estado para ahorro de potencia.
- Modo *sniff*: para envío de mensajes en timeslots específicos.
- Modo *park*: para que el esclavo permanezca inactivo pero sincronizado en la *piconet.*
- Enlaces *SCO*
- Control de paquetes *multi-slot*
- Supervisión de enlace

A.2.1 Establecimiento de conexión

Después del procedimiento *paging*, el maestro debe encuestar al esclavo enviando paquetes de sondeo. El otro lado recibe este mensaje y lo acepta o rechaza, si es aceptado, la comunicación incluyendo las capas superiores están disponibles tal como se muestra en la Figura A.9.

Figura A.9 Establecimiento de la conexión

A.3 INTERFAZ DEL CONTROLADOR DE HOST (HCI)

La *HCI* proporciona una interfaz de comando al *controlador banda base* y a la *gestión de enlace,* además de acceso al hardware y a los registros de control. Esta interfaz brinda un método estándar para acceder a los recursos de banda base *Bluetooth*.

A.3.1 Capas mas bajas del stack Bluetooth

A continuación se hace una breve descripción de las capas más bajas del *stack* de *software y hardware Bluetooth.* La Figura A.10 da una idea de estas capas.

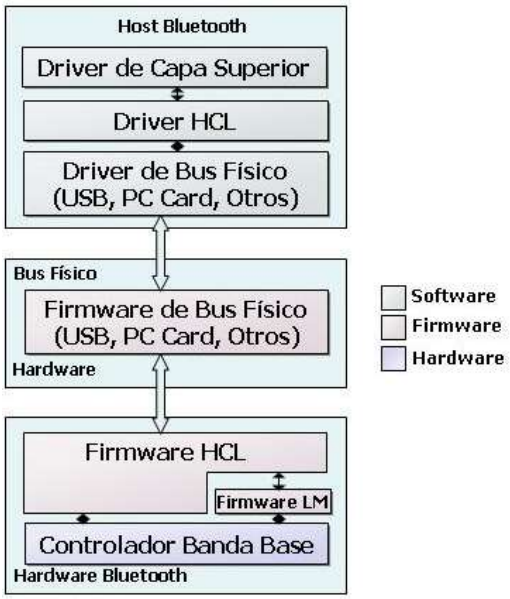

Figura A.10 Diagrama de las capas mas bajas en Bluetooth

El *firmware HCI* implementa los comandos *HCI* para el *hardware* a través del acceso de comandos banda base, comandos de la *gestión de enlace*, hardware de registros de estado, registros de control y registros de eventos. La Figura A.11 muestra el recorrido de un dato transferido de un dispositivo a otro. El *driver HCI* (en el *host***)** intercambia datos y comandos con el *firmware HCI (*en el hardware*)*. El *driver* de la capa de transporte, por ejemplo un bus físico, brinda a las dos capas *HCI* la posibilidad de intercambiar información.

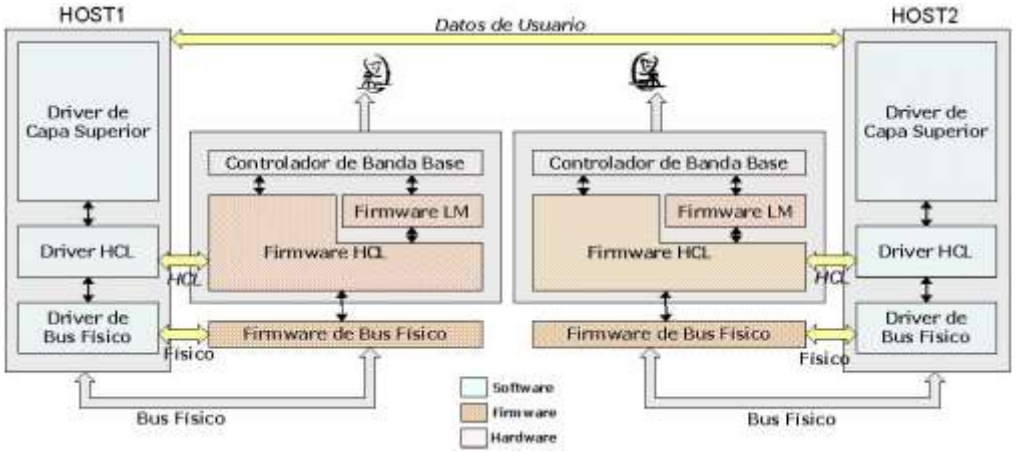

Figura A.11 Diagrama general end to end de las capas de software mas bajas

A.3.2 Posibles arquitecturas de bus físico

Los dispositivos *Bluetooth* tienen varias interfaces de bus físicas que pueden ser usadas para conectar el hardware. Estos buses pueden tener diferentes arquitecturas y parámetros. El *controlador de host* soporta tres arquitecturas de bus físico, *USB, UART y PC Card.* Todas ellas pueden manejar varios canales lógicos sobre el mismo canal físico simple (a través de *endpoints*), por lo tanto los canales de control, datos y voz no requieren alguna interfaz física adicional. El objetivo principal de la capa de transporte del *controlador de host* es la transparencia entre el driver del *controlador de host* y el *controlador de host.*

A.4 PROTOCOLO DE CONTROL Y ADAPTACIÓN DE ENLACE LÓGICO (L2CAP)

L2CAP se encuentra sobre el *protocolo de gestión de enlace (LMP)* y reside en la capa de enlace de datos. *L2CAP* permite a protocolos de niveles superiores y a aplicaciones la transmisión y recepción de paquetes de datos *L2CAP* de hasta 64 kilobytes, con capacidad de multiplexación de protocolo, operación de segmentación y reensamble, y abstracción de grupos. Para cumplir sus funciones, *L2CAP* espera que la *banda base* suministre paquetes de datos en *full duplex,* que realice el chequeo de integridad de los datos y que reenvíe los datos hasta que hayan sido reconocidos satisfactoriamente. Las capas superiores que se comunican con *L2CAP* son por ejemplo el *protocolo de descubrimiento de servicio (SDP), el RFCOMM y el control de telefonía (TCS)*.

A.4.1 Canales

L2CAP está basado en el concepto de canales. Se asocia un identificador de canal, *CID*, a cada uno de los *endpoints* de un canal *L2CAP.* Los *CIDs* están divididos en dos grupos, uno con identificadores reservados para funciones *L2CAP* y otro con identificadores libres para implementaciones particulares. Los canales de datos orientados a la conexión representan una conexión entre dos dispositivos, donde un *CID* identifica cada *endpoint* del canal. Los canales no orientados a la conexión limitan el flujo de datos a una sola dirección. La señalización de canal es un ejemplo de un canal reservado. Este canal es usado para crear y establecer canales de datos orientados a la conexión y para negociar cambios en las características de esos canales.

A.4.2 Operaciones entre Capas

Las implementaciones *L2CAP* deben transferir datos entre protocolos de capas superiores e inferiores. Cada implementación debe soportar un grupo de comandos de señalización, además,

debe ser capaz de aceptar ciertos tipos de eventos de capas inferiores y generar eventos para capas superiores. En la Figura A.12 se muestra esta arquitectura.

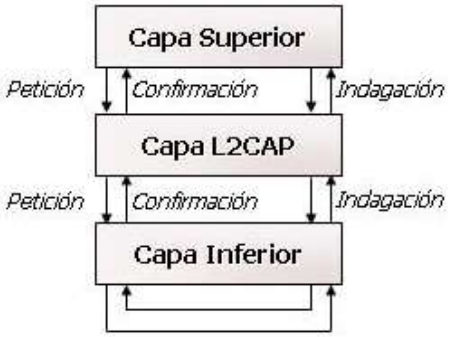

Figura A.12 Arquitectura L2CAP

A.4.3 Segmentación y Reensamblado

Los paquetes de datos definidos por el protocolo *banda base* están limitados en tamaño. Los paquetes *L2CAP* grandes deben ser segmentados en varios paquetes *banda base* más pequeños antes de transmitirse y luego deben ser enviados a la *gestión de enlace.* En el receptor los pequeños paquetes recibidos de la banda base son reensamblados en paquetes *L2CAP* más grandes. Varios paquetes *banda base* recibidos pueden ser reensamblados en un solo paquete *L2CAP* seguido de un simple chequeo de integridad. La segmentación y reensamblado, *SAR,* funcionalmente es absolutamente necesaria para soportar protocolos usando paquetes más grandes que los soportados por la *banda base*. La Figura A.13 muestra la segmentación *L2CAP.*

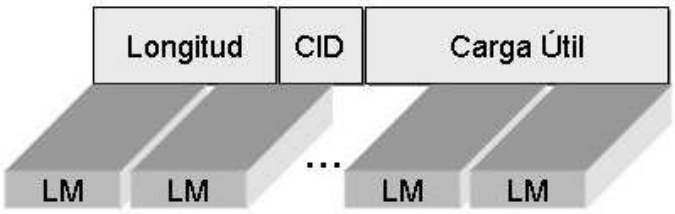

Figura A.13 Segmentación L2CAP

A.4.4 Eventos

Todos los mensajes y *timeouts* que entran en la capa *L2CAP*, son llamados eventos. Los eventos se encuentran divididos en cinco categorías:

Indicaciones y confirmaciones de capas inferiores, peticiones de señal y respuestas de capas *L2CAP*, datos de capas *L2CAP*, peticiones y respuestas de capas superiores, y eventos causados por expiraciones de tiempo.

A.4.5 Acciones

Todos los mensajes y *timeouts* enviados desde la capa *L2CAP* son llamados acciones (en el lado del receptor estas acciones son llamadas eventos). Las acciones se encuentran divididas en cinco categorías: peticiones y respuestas a capas inferiores, peticiones y respuestas a capas *L2CAP*, datos a capas *L2CAP*, indicaciones a capas superiores, y configuración de *timers.*

A.4.6 Formato del paquete de datos

L2CAP está basado en paquetes pero sigue un modelo de comunicación basado en canales. Un canal representa un flujo de datos entre entidades *L2CAP* en dispositivos remotos. Los canales pueden ser o no orientados a la conexión. Como se puede observar en la Figura A.14, los paquetes de canal orientado a la conexión están divididos en tres campos: *longitud de la información, identificador de canal, e información.*

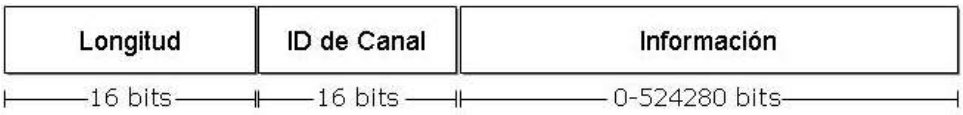

Figura A.14 Paquete L2CAP

Los paquetes de canal de datos no orientados a la conexión son iguales a los paquetes orientados a la conexión pero adicionalmente incluyen un campo con información multiplexada de protocolo y servicio.

A.4.7 Calidad de servicio (QoS)

La capa *L2CAP* transporta la información de calidad de servicio a través de los canales y brinda control de admisión para evitar que canales adicionales violen contratos de calidad de servicio existentes.

Algunos esclavos pueden requerir un alto rendimiento o una respuesta rápida. Antes de que un esclavo con grandes peticiones sea conectado a una *piconet,* el esclavo trata de obtener una garantía a sus demandas. Puede solicitar una determinada rata de transmisión, tamaño del buffer de tráfico, ancho de banda, tiempo de recuperación de datos, etc. Por lo tanto, antes de que el

maestro conecte a un nuevo esclavo o actualice la configuración de calidad, debe chequear si posee **timeslots** y otros recursos libres.

A.5 PROTOCOLO DE DESCUBRIMIENTO DE SERVICIO (SDP)

El *protocolo de descubrimiento de servicio*, *SDP,* brinda a las aplicaciones recursos para descubrir qué servicios están disponibles y determinar las características de dichos servicios.

A.5.1 Descripción General

Un servicio es una entidad que puede brindar información, ejecutar una acción o controlar un recurso a nombre de otra entidad. El *SDP* ofrece a los clientes la facilidad de averiguar sobre servicios que sean requeridos, basándose en la clase de servicio o propiedades específicas de estos servicios. Para hacer más fácil la búsqueda, el *SDP* la habilita sin un previo conocimiento de las características específicas de los servicios. Las unidades *Bluetooth* que usan el *SDP* pueden ser vistas como un servidor y un cliente. El servidor posee los servicios y el cliente es quien desea acceder a ellos. En el SDP esto es posible ya que el cliente envía una petición al servidor y el servidor responde con un mensaje. El *SDP* solamente soporta el descubrimiento del servicio, no la llamada del servicio.

A.5.2 Registros de servicio

Los registros de servicio contienen propiedades que describen un servicio determinado. Cada propiedad de un registro de servicio consta de dos partes, un identificador de propiedad y un valor de propiedad. El identificador de propiedad es un número único de 16 bits que distingue cada propiedad de servicio de otro dentro de un registro. El valor de propiedad es un campo de longitud variable que contiene la información.

A.5.3 El protocolo

El *protocolo de descubrimiento de servicio (SDP)* usa un modelo petición/respuesta. En la Figura 1-15 se muestra el procedimiento *SDP.*

Note que las flechas no continuas no forman parte de éste.

- *Petición de búsqueda de servicio:* se genera por el cliente para localizar registros de servicio que concuerden con un patrón de búsqueda dado como parámetro. Aquí el servidor examina los registros en su base de datos y responde con una *respuesta a búsqueda de servicio.*

- *Respuesta a búsqueda de servicio:* se genera por el servidor después de recibir una *petición de búsqueda de servicio* válida.
- *Petición de propiedad de servicio:* una vez el cliente ya ha recibido los servicios deseados, puede obtener mayor información de uno de ellos dando como parámetros el registro de servicio y una lista de propiedades deseadas.
- *Respuesta a propiedad de servicio:* El *SDP* genera una respuesta a una *petición de propiedad de servicio.* Ésta contiene una lista de propiedades del registro requerido.
- *Petición de búsqueda y propiedad de servicio:* se suministran un patrón de servicio con servicios deseados y una lista de propiedades deseadas que concuerden con la búsqueda.
- *Respuesta de búsqueda y propiedad de servicio:* como resultado se puede obtener una lista de servicios que concuerden con un patrón dado y las propiedades deseadas de estos servicios.

A.6 RFCOMM

El protocolo *RFCOMM* brinda emulación de puertos seriales sobre el protocolo *L2CAP*. La capa *RFCOMM* es una simple capa de transporte provista adicionalmente de emulación de circuitos de puerto serial *RS-232*. El protocolo *RFCOMM* soporta hasta 60 puertos emulados simultáneamente. Dos unidades *Bluetooth* que usen *RFCOMM* en su comunicación pueden abrir varios puertos seriales emulados, los cuales son multiplexados entre sí. La Figura A.15 muestra el esquema de emulación para varios puertos seriales.

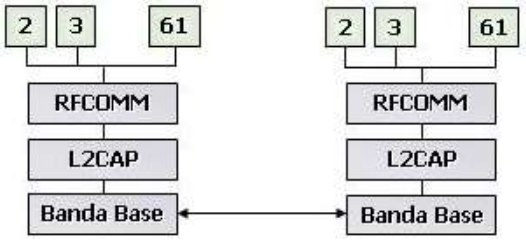

Figura A.15 Varios Puertos Seriales emulados mediante RFCOMM

Muchas aplicaciones hacen uso de puertos seriales. El *RFCOMM* está orientado a hacer más flexibles estos dispositivos, soportando fácil adaptación de comunicación *Bluetooth*. Un ejemplo de una aplicación de comunicación serial es el protocolo *punto-a-punto (PPP).* El RFCOMM tiene construido un esquema para emulación de *null modem* y usa a *L2CAP* para cumplir con el control de flujo requerido por alguna aplicación.