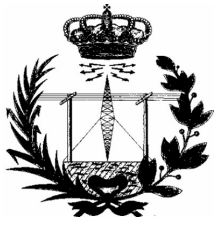
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Plataforma para asistencia a discapacitados: núcleo y aplicaciones



AUTOR: Juan Manuel Pérez Mañogil
DIRECTOR: Javier Vales Alonso
Septiembre / 2005

Autor	Juan Manuel Pérez Mañogil
E-mail del Autor	kwyjibo@ono.com
Director(es)	Javier Vales Alonso
E-mail del Director	jvales@upct.es
Codirector(es)	-
Título del PFC	Plataforma para asistencia a discapacitados: núcleo y aplicaciones
Descriptores	XScale, XML, Bluetooth, 802.11, discapacitados

Resumen

Uno de los problemas de los entornos urbanos en la sociedad actual, es su poca adaptación a los problemas de las personas discapacitadas, el entorno no está diseñado para ellas.

Con la idea de mejorar esta situación se ha desarrollado un sistema con la intención de que el entorno se adapte automáticamente al perfil del usuario. Este perfil, contenido en una ficha XML, es portado por un dispositivo móvil (teléfono o PDA). En el entorno del controlador, las fichas son capturadas, mediante interfaces inalámbricas, sin intervención del usuario. Después los perfiles son procesados y las actuaciones pertinentes son llevadas a cabo.

Se ha diseñado un prototipo a modo de semáforo, para ejemplificar los posibles usos de este sistema. En este prototipo, discapacitados con deficiencias visuales o dificultades de movimiento, al acercarse al dispositivo obtendrán un tiempo mayor para cruzar, y de señales auditivas, según su grado de minusvalía.

En esta parte del proyecto se ha diseñado un servidor sobre una plataforma hardware XScale SB-X255 con sistema operativo GNU Linux. La aplicación es capaz de capturar tanto las fichas portadas en dispositivos con Bluetooth o Wifi habilitado. Asismismo, se ha desarrollado la aplicación destinada a procesar las fichas, y a disparar las actuaciones adecuadas para la adaptación.

Titulación	Ingeniería Técnica de Telecomunicación, especialidad Telemática
Intensificación	-
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Septiembre – 2005

La consecución de este proyecto no habría sido posible sin muchas de las personas de las que estoy rodeado, aunque ellas ni sospechen que me ayudan con su sola presencia. Agradezco los esfuerzos de mi director de proyecto, Javier Vales, a mis compañeros, Félix Belzunce y Marco Antonio Esparza. También agradezco el apoyo de mi familia, a mi padre y a mi madre, y a mis hermanas. Y por último a mis amigos por mantenerse cerca, y a mis enemigos (si los tuviese) por mantenerse aún más cerca.

Juan Manuel Pérez Mañogil

Índice

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Índice de gráficos

Capítulo 1

Introducción

1.1 Planteamiento

En la sociedad actual, los discapacitados sufren una gran discriminación frente al resto de ciudadanos. A pesar de los esfuerzos desplegados en los últimos años, la mayoría de los accesos y aplicaciones urbanas siguen básicamente pensadas para el ciudadano medio, mientras que los discapacitados físicos se ven discriminados, y han de desplegar grandes esfuerzos para intentar desarrollar una vida normal en ciudades que no disponen de medios o éstos no son suficientes para atender sus necesidades.

Esto ocurre en particular con las personas que sufren discapacidad visual. En una cuestión tan elemental y vital como es el paso de peatones que cuentan con semáforo, se ven en la necesidad de tantear hasta poder encontrar el botón que active el color verde que les franquea el paso. Por otro lado, los discapacitados motrices, una vez que pulsan dicho botón, apenas disponen de tiempo suficiente para cruzar la vía.

Una buena solución a este problema puede ser la de conectar las aplicaciones ya existentes en los semáforos a los discapacitados, de manera que las puedan manejar sin esfuerzo alguno. Idealmente esta solución debería requerir la menor cantidad de interacción por parte del usuario, es decir, que el proceso de aclimatar el entorno urbano al usuario se llevase acabo sin la intervención de éste. Por lo tanto, debe ser el medio el que analice a sus usuarios para tomar las decisiones apropiadas en cada caso, es decir, que va a ser el entorno el que se adapte a una colectividad de usuarios.

Evidentemente, para que el entorno pueda analizar a los usuarios en un momento dado, éstos deben llevar alguna clase de identificador con información personal para la toma de decisión por parte de aquél. Como tenemos como requisito que la interacción entre el usuario y el entorno sea lo más automática posible, hemos de descartar remedios típicos como tarjetas de

identificación.

Actualmente, las tecnologías inalámbricas están en auge y no es difícil llegar a la conclusión de que esta respuesta puede ser la acertada. La mayoría de los usuarios disponen de algún tipo de dispositivo de comunicación inalámbrica, ya sea móvil, PDA u ordenador portátil. Aprovechando esta característica para solucionar el problema planteado, podríamos desarrollar una aplicación para dispositivos móviles que fuese el que interactuara con el medio mediante tecnologías inalámbrica actuales, principalmente Bluetooth para dispositivos móviles, Wi-fi para PDAs y ordenadores portátiles.

1.2 Tecnologías inalámbricas

Bluetooth y Wi-fi son tecnologías inalámbricas que actualmente llevan incorporado la mayoría de dispositivos móviles que están en el mercado. El bajo coste, los distintos niveles de seguridad que llevan incorporados, el gran ancho de banda, el bajo consumo así como las altas prestaciones que nos proporcionan estos dispositivos hacen que sean ideales para nuestro propósito.

El sistema que se propone ha sido desarrollado según estas dos tecnologías, no solo por todo lo anterior, sino también porque su modo de funcionamiento ofrece la posibilidad de que tan solo, a través del encendido del dispositivo móvil, el sistema empiece a funcionar sin necesidad alguna de actuación por parte del discapacitado, de aquí su gran ventaja.

Tanto Bluetooth como Wi-fi funcionan en las llamadas LANs por espectro ensanchado, construidas habitualmente de manera celular en las que celdas adyacentes utilizan distintas frecuencias para evitar interferencias. Los dispositivos suelen utilizar antenas omnidireccionales con una potencia máxima de transmisión de 1 Watio, aunque dicha configuración no es obligatoria. Utilizan la banda sin licencia de 2.4 GHz, es la llamada banda ISM (Industrial, Scientific and Medical).

Debido a que tanto Bluetooth como Wi-fi funcionan en la misma banda de frecuencia las pérdidas de transmisión se verán acentuadas, por ello se hacía necesario un protocolo orientado a conexión con el que conseguir minimizar las pérdidas de datos en transmisión y que nos aportara fiabilidad en la recepción de los datos. Por todo lo anterior y porque además son protocolos estándares totalmente probados y fiables se ha hecho uso de TCP/IP para Wi-fi y RFCOMM para Bluetooth.

1.3 Aplicaciones de las tecnologías inalámbricas en discapacitados

Nuestro objetivo es la realización de una aplicación para dar soporte a personas con alguna discapacidad de tipo locomotriz o visual, de tal forma que podamos facilitar algo tan común en el día a día como el mero hecho de cruzar una vía regulada por un semáforo.

Para la realización del proyecto contamos con un proceso servidor capaz de regular de forma inteligente el funcionamiento de un semáforo. El proceso servidor estará a la espera de la llegada de clientes dentro de la zona de cobertura y tras la conexión, estos dispositivos enviarán al servidor una ficha con unos determinados datos del cliente acerca de la discapacidad que tiene y en función de estas características el servidor tomará una decisión u otra.

Cabe la posibilidad de que varios discapacitados accedan simultáneamente al sistema, en este caso el servidor es capaz de tomar una decisión coherente al número y tipo de discapacitados que se encuentren en ese momento.

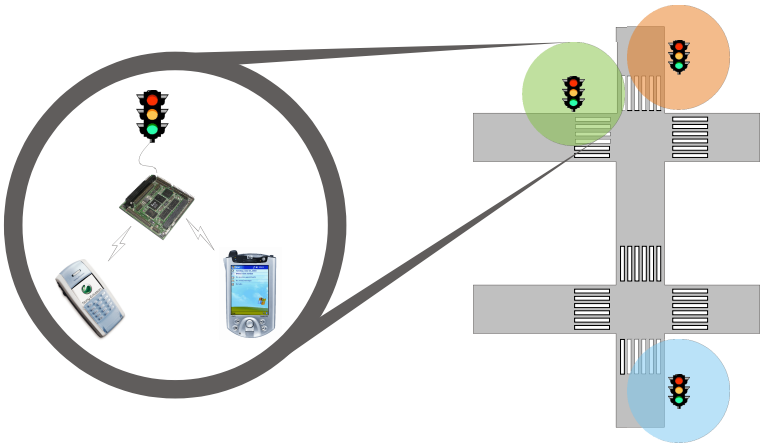


Figura 1.1 - Esquema lógico de la disposición del sistema

1.4 Arquitectura general del sistema

Se describirá el funcionamiento general del sistema a partir de la figura. En ella se pueden distinguir los siguientes procesos:

- Cliente: Es aquel que sin ningún tipo de ínteractuación por parte del usuario del dispositivo móvil, es capaz de forma automática de establecer una conexión con el proceso servidor para enviarle una ficha XML (en el que se indica el tipo de discapacidad del usuario) y

posteriormente detectar cuando ha salido de la zona de cobertura para volver a establecer conexión con un nuevo proceso servidor.

- Servidor: El proceso servidor es el encargado de recibir las fichas del proceso cliente, enviárselas al procesador XML y de forma parecida a como el cliente debe de detectar que ha salido de la zona de cobertura, el proceso servidor también debe de detectar cuando un cliente a abandonado la zona de cobertura que cubre.
- Procesador XML: Proceso que de forma local establece una comunicación con el proceso toma de decisión, indicándole a éste el tipo o tipos de discapacidad que ha extraído de la ficha que le ha pasado el proceso servidor.
- Proceso toma de decisión: El proceso toma de decisión, tras recibir el tipo de discapacidad del individuo o individuos que en un momento dado se encontraban dentro de la zona de cobertura, es capaz de tomar una buena decisión a la hora de controlar la placa que simula un semáforo con un zumbador.

Al final todos estos procesos deben de comunicarse con un determinado orden de manera que la placa funcione conforme se pretende. Las principales interacciones entre los procesos son:

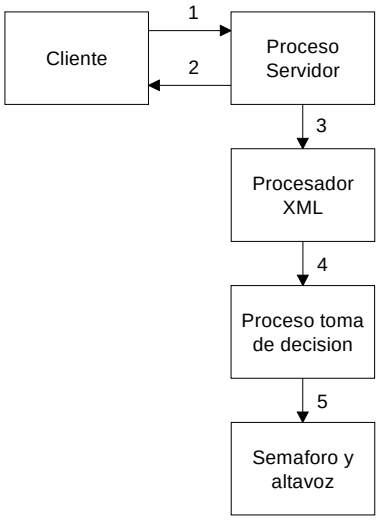


Figura 1.2 - Esquema básico del sistema

1. El cliente establece una conexión con el servidor y le envía su ficha XML.
2. El servidor indica al cliente si la ficha es válida.

3. Si la ficha es válida el servidor le pasa la ficha al procesador XML.
4. Proceso XML envía el tipo de discapacidad extraído de la ficha al proceso toma de decisión.
5. Proceso toma de decisión interactúa con la placa en función de los distintos tipos de discapacidades que podía haber en un determinado momento dentro de la zona de cobertura del servidor.

En el gráfico se muestra en detalle la interacción entre el cliente y el servidor:

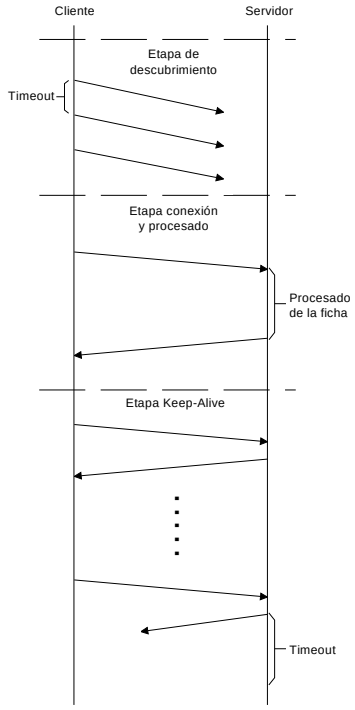


Figura 1.3 - Diagrama de flujo de la conexión

El cliente se mantiene en un estado de descubrimiento en el que intenta conectarse a un posible servidor cada cierto tiempo.

En el momento que se produce una conexión exitosa, se pasa a la siguiente etapa e, inmediatamente, envía su ficha XML, la cual es procesada por el servidor, recibiendo el cliente una respuesta sobre la validez de la ficha.

Si la ficha es correcta se procederá a entrar en una etapa para mantener viva la conexión. Cuando la comunicación con el servidor se corte, se entenderá que el cliente ha salido de cobertura, teniendo que volver al principio

del diagrama.

1.5 Objetivos del proyecto

El objetivo de este proyecto ha sido dotar al discapacitado de un sistema capaz de funcionar sin interacción alguna por parte del usurario, en el que el entorno se adapta a las necesidades de cada uno de los discapacitados que puedan acceder al sistema.

Capítulo 2

Trabajos Relacionados

En este capítulo presentamos algunos sistemas que tienen como objetivo el de hacer un poco mas fácil la vida de las personas discapacitadas. Presentamos los puntos más importantes de cada uno de ellos, destacando aspectos comunes y posibles mejoras.

2.1 Automation and Telematics for assisting people living at home

En este proyecto desarrollado por la Universidad de Helsinki se pretende usar la Telemática y la Automatización en beneficio de la personas mayores o con alguna minusvalía. Estos beneficios se llevan a cabo con la implantación en el domicilio de un conjunto de sistemas, intentado así dotar de completa autonomía a las personas discapacitadas en su hogar. Estos sistemas se conectan a un ordenador central encargado de recopilar información de todos los dispositivos y actuar según lo previsto.

Algunos de estos dispositivos son cámaras, sensores térmicos, detectores magnéticos y de movimiento, robot móvil etc. El sistema también ha sido desarrollado para ser controlado a través de un móvil o PDA e informar al usuario, a través de estos dispositivos, de los sucesos acontecidos dentro del hogar.

2.2 Accesibilidad para Discapacitados a través de Teléfonos y Servicios Móviles Adaptables

En este trabajo desarrollado en la Universidad de Deusto (Bilbao) se estudian los mecanismos que añaden accesibilidad a las personas discapacitadas en el uso de dispositivos móviles. Estos mecanismos son el uso de sintetizadores de voz para personas invidentes, incorporación de botones especiales con números de marcación automática (por ejemplo el de emergencias 112), aplicaciones para el uso de los dispositivos inalámbricos como controles remotos, reconocedores de cadenas de texto o códigos especiales por medios

de capturas fotográficas, para su posterior representación en audio por medio de los ya comentados sintetizadores de voz.

Dado que todos los sistemas desarrollados para mejorar la vida de las personas con alguna discapacidad cumplen un gran papel social es muy difícil encontrarles pegas, si bien el kit de la cuestión reside más en la posible interacción de los distintos sistemas para un funcionamiento compatible entre si. Es decir, sería ideal que tanto este sistema desarrollado por la Universidad de Helsinki que dota al usuario de total infraestructura dentro de su hogar, como el desarrollado por la Universidad de Deusto que dota a los dispositivos móviles de una mayor accesibilidad a personas discapacitadas, pudiesen interaccionar con el nuestro.

Un posible ejemplo de esta interacción lo podemos encontrar cuando la persona discapacitada llega a casa y tiene que buscar la llave que abre la puerta, pues bien este problema se podría solucionar con la colocación de un receptor en la puerta, capaz de comunicarse con el dispositivo inalámbrico, de la misma forma que en la elaboración de este proyecto se ha hecho para la comunicación con el semáforo. De esta forma la puerta se abriría automáticamente pudiendoselo comunicar al usuario por medio del sintetizador de voz.

Capítulo 3

Bluetooth

3.1 Bluetooth

Bluetooth es un estándar de comunicaciones que identifica un conjunto de protocolos, los cuales facilitan la comunicación inalámbrica entre diferentes dispositivos electrónicos. El origen de la palabra Bluetooth proviene del rey Vikingo, Harald Bluetooth (Harald Diente Azul), que gracias a su habilidad para la comunicación y el tratar con las personas consiguió unificar Dinamarca. Esto es lo que se pretende con Bluetooth, unificar los estándares y las tecnologías diferentes mediante un dispositivo universal para la interconexión de todo tipo de periféricos.

Es una tecnología reciente, de especificación abierta, que define una forma de comunicación inalámbrica, la cual posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia de corto alcance.

La especificación abierta hace posible que los fabricantes puedan, libremente, implementar sus propios protocolos de aplicación sobre los protocolos específicos de Bluetooth, permitiendo así el desarrollo de nuevas aplicaciones que fomentan el desarrollo de las capacidades tecnológicas de Bluetooth.

Los principales objetivos que se pretende conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre los equipos personales.

Bluetooth es ideal para el uso en dispositivos móviles ya que cumple

3 características básicas para poder ser usado en éstos: reducido consumo, pequeño tamaño y un bajo precio, lo que hace que actualmente la inmensa mayoría de móviles que se venden en el mercado ya lleven instalada esta tecnología.

3.2 Evolución

En 1994 Ericsson inició un estudio para investigar la viabilidad de una interfaz de radio, de bajo coste y escaso consumo, para la interconexión entre dispositivos móviles y otros accesorios con la intención de eliminar los cables.

El estudio partía de un largo proyecto que investigaba sobre unos multicomunicadores conectados a una red celular, hasta que se llegó a un enlace de radio de corto alcance, llamado *MC link*. Conforme ese proyecto se fue desarrollando se vio que ese tipo de enlace podría ser utilizado ampliamente en un gran número de aplicaciones, ya que tenía como principal ventaja que se basaba en un chip de radio relativamente económico.

A comienzos de 1997, según avanzaba el proyecto *MC link*, Ericsson fue despertando el interés de otros fabricantes de equipos móviles. En seguida se comprendió que para que el sistema tuviera éxito, un gran número de equipos deberían estar equipados con ésta tecnología.

Esto fue lo que originó a principios de 1998 la creación de un grupo de interés especial (SIG, *Special Interest Group)*, formado por 5 grandes fabricantes de la industria de las comunicaciones móviles: Ericsson, Nokia, IBM, Toshiba e Intel. La idea era abarcar una gran cantidad del mercado de las tecnologías, para lo que se conto con dos líderes del mercado de las telecomunicaciones, dos líderes del mercado de los PCs portátiles y un líder de la fabricación de chips.

El primer objetivo para los productos Bluetooth de primera generación eran los entornos de la gente de negocios que vieja frecuentemente. Por lo que se debería pensar en integrar el chip de radio Bluetooth en equipos como: PCs portátiles, teléfonos móviles, PDAs y auriculares. Esto originaba una serie de cuestiones previas que deberían solucionarse tales como:

- Un sistema capaz de operar en todo el mundo.
- Un pequeño consumo de energía, ya que se debía integrar en equipos alimentados por baterías.

- La conexión deberá soportar voz y datos, y por la tanto aplicaciones multimedia.

La versión actual en la que se encuentra bluetooth es la 1.1(posterior a la 1.0 y 1.0b). Teniendo en cuenta que ésta tecnología es muy reciente, no es de extrañar que surjan nuevas versiones en los próximos años, conforme aparezcan nuevos escenarios de uso de los equipos y se vayan incorporando al mercado nuevos productos.

3.3 Topología

Para poder establecer una comunicación entre dispositivos Bluetooth, uno de ellos debe establecer el papel de maestro y otro el de esclavo. El maestro es el encargado de la sincronización de la comunicación. Todos los esclavos conectados al maestro saltarán a la frecuencia establecida por el maestro. El maestro se encargará de controlar cuando un esclavo puede transmitir, reservando ranuras temporales para éste. El papel de maestro es, generalmente, de aquel que comienza la comunicación, aunque este papel puede cambiarse más tarde.

Un maestro puede establecer comunicación con hasta 7 esclavos a la vez. Cabe destacar que el papel de maestro o esclavo solamente tiene importancia a la hora de realizar el sincronismo a bajo nivel.

3.3.1 Piconet

Si un equipo se encuentra dentro del radio de cobertura de otro, estos pueden establecer una conexión. En principio sólo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace. Los participantes en la piconet pueden intercambiar los papeles si un esclavo quisiera asumir el papel de maestro. Sin embargo sólo puede haber un maestro en la piconet al mismo tiempo

Cada unidad de la piconet utiliza su identidad maestra y el reloj nativo para seguir en el canal de salto. Cuando se establece la conexión, se añade un ajuste de reloj a la propia frecuencia de reloj nativa de la unidad, para poder sincronizarse con el reloj nativo del maestro.

Las unidades maestras controlan el tráfico del canal. A un esclavo sólo se le permite enviar un slot a un maestro cuando éste se ha dirigido por su dirección MAC (control de acceso al medio) en el procedimiento de slot maestro-esclavo.

3.3.2 Scatternet

Los equipos que comparten un mismo canal sólo pueden utilizar una parte de la capacidad de éste. Aunque los canales tienen un ancho de banda de un 1Mhz, cuantos más usuarios se incorporan a la piconet, disminuye la capacidad hasta unos 10 kbit/s más o menos. Teniendo en cuenta que el ancho de banda medio disponible es de unos 80 MHz éste no puede ser utilizado eficazmente, cuando cada unidad ocupa una parte del mismo canal de salto de 1Mhz. Para poder solucionar éste problema se adoptó una solución de la que nace el concepto de scatternet.

Para crear una scatternet un dispositivo de una de las dos piconets tiene que pertenecer a dos de éstas. Esto se puede hacer de dor formas: ser esclavo de las dos piconets o ser esclavo de una piconet y maestro de la otra. Las unidades que se encuentran en el mismo radio de cobertura pueden establecer potencialmente comunicaciones entre ellas. Sin embargo, sólo aquellas unidades que realmente quieran intercambiar información comparten un mismo canal creando la piconet.

Éste hecho permite que se creen varias piconets en áreas de cobertura superpuestas. A un grupo de piconets se le llama scatternet. En un conjunto de varias piconets, éstas seleccionan diferentes saltos de frecuencia y están controladas por diferentes maestros, por lo que si un mismo canal de salto es compartido temporalmente por piconets independientes, los paquetes de datos podrán ser distinguidos por el código de acceso que les precede, que es único en cada piconet.

Una unidad al incorporarse a una nueva piconet debe modificar el offset (ajuste interno) de su reloj para minimizar la deriva entre su reloj nativo y el del maestro, por lo que gracias a éste sistema se puede participar en varias piconets realizando cada vez los ajustes correspondientes una vez conocidos los diferentes parámetros de la piconet. Cuando una unidad abandona una piconet, el esclavo informa al maestro actual que ésta no estará disponible por un determinado periodo, que será en el que estará activa en otra piconet. Durante su ausencia, el tráfico en la piconet entre el maestro y otros esclavos continua igualmente.

De la misma manera que un esclavo puede cambiar de una piconet a otra, un maestro también lo puede hacer, con la diferencia de que el tráfico de la piconet se suspende hasta la vuelta del maestro. El maestro que entra en una

nueva piconet, en principio, lo hace como esclavo, a no ser que posteriormente éste solicite actuar como maestro.

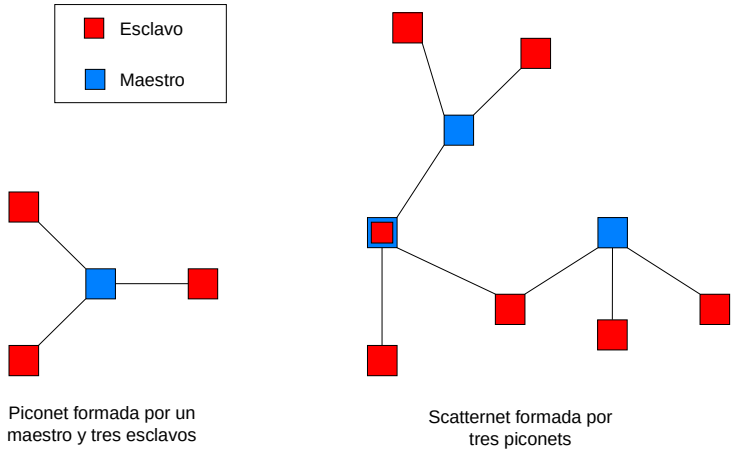


Figura 3.1 - Ejemplo de maestros y esclavos en una piconet y en una scatternet

3.4 Aspectos generales

Para poder operar era necesaria una banda de frecuencias abierta, es decir, sin necesidad de licencia independientemente del lugar del planeta donde nos encontremos, para transmitir hasta una distancia de 100 metros a velocidades de transferencia del orden de los 721 Kbps. Solo la banda ISM (médico-científica internacional) a 2,45 GHz cumple estos requisitos, con rangos que van de los 2400 MHz a los 2500 MHz, y sólo con algunas restricciones en países como Francia y Japón.

Un esquema de saltos de frecuencia aleatorios permite a los dispositivos comunicarse incluso en áreas donde existe gran interferencia electromagnética. Bluetooth suministra también mecanismos de encriptación y autentificación, para controlar la conexión y evitar que cualquier dispositivo, no autorizado, pueda acceder a los datos o modificarlos. El manejo de la clave se hace a nivel de aplicación.

Bluetooth se ha diseñado para operar en una ambiente multi-usuario. Los dispositivos pueden habilitarse para comunicarse entre sí e intercambiar datos de una forma transparente al usuario. Aunque cada enlace es codificado y protegido contra interferencias y pérdidas de enlace, Bluetooth puede considerarse como una red inalámbrica segura. Aun así también es posible es uso de algunas técnicas a nivel de aplicación para poder aumentarla.

3.4.1 Salto en frecuencia

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar múltiples interferencias que pudieran producirse. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia.

En los sistemas de radio Bluetooth se suele utilizar el método se salto en frecuencia debido a que ésta tecnología puede ser integrada en equipos de baja potencia y bajo coste. Éste sistema divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto, de manera pseudo-aleatoria.

Con esto se consigue que el ancho de banda instantáneo sea muy pequeño y también una propagación efectiva sobre el total de ancho de banda. El canal de radio de Bluetooth utiliza espectro ensanchado y salto en frecuencia (FHSS, *Frecuency Hopping Spread Spectrum*), por lo que puede cambiar de estado hasta 1600 veces por segundo sobre 79 frecuencias distintas separadas 1 MHz, empezando en 2,402 GHz y terminando en 2,480 GHz. Para cumplir las normas de banda en cada país, se deja una banda de guardia en los límites superior e inferior de la banda. Otro aspecto a destacar es la pequeña longitud de los paquetes, con lo que si otro dispositivo intentara transmitir a la misma frecuencia la probabilidad de colisión también seria baja debido a este motivo. En la siguiente figura tenemos un ejemplo de dos dispositivos que están intentando transmitir en la misma frecuencia.

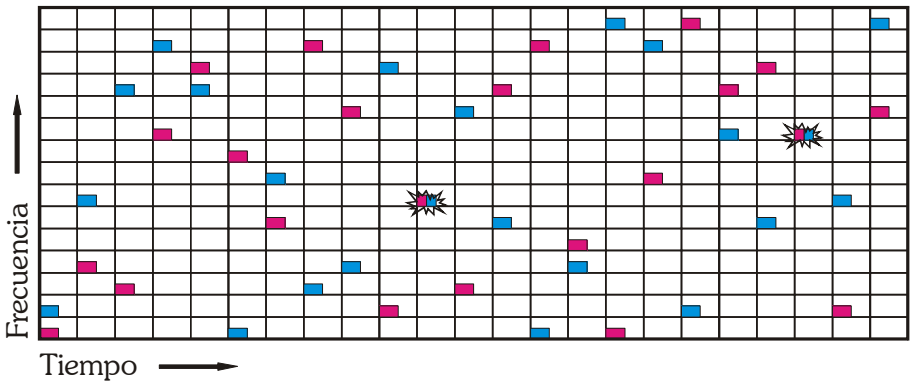


Figura 3.2 - Ejemplo de la técnica de salto en frecuencia

Este caso se refiere a una situación donde hay dos o mas piconets activas transmitiendo a una determinada frecuencia o un dispositivo no

Bluetooth usando la misma frecuencia que uno que si lo sea transmitiendo a la vez.

3.4.2 Definición del canal

El canal queda dividido en intervalos de 625 µs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Esto da lugar a una frecuencia de salto de 1600 veces por segundo, en la que un paquete de datos ocupa un slot para la emisión y otro para la recepción y que pueden ser usados alternativamente, dando lugar a un esquema de división dúplex en el tiempo(TDD, *Time Division Duplex*).

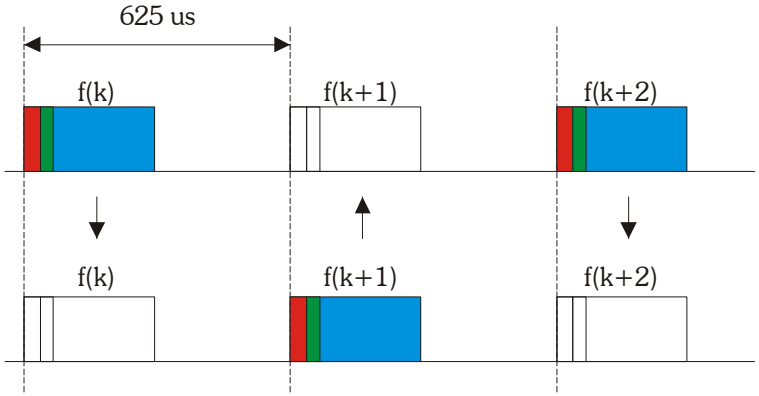


Figura 3.3 - Transmisión por división en el tiempo

3.4.3 Enlaces

Existen dos tipos de enlaces que permiten soportar incluso aplicaciones multimedia:

- Enlace se sincronización de conexión orientada (SCO, Sinchronous Connection Oriented). Los enlaces SCO soportan conexiones asimétricas, punto a punto, usadas normalmente en conexiones de voz. El enlace es punto a punto entre el esclavo y el maestro de la piconet. Pudiendo tener éste hasta tres enlaces full-duplex simultáneos de. La transmisión de voz se realiza sin ningún mecanismo de protección, y los paquetes no se retransmiten nunca.
- Enlace asíncrono de baja conexión (ACL, Assinchronous Connectionless). Los enlaces ACL soportan conmutaciones punto-multipunto (enlace entre el maestro y varis esclavos) típicamente usadas en la transmisión de datos. El maestro puede establecer conexión con cualquier esclavo de la piconet, aunque éste tenga un enlace SCO ya establecido. Permite la retransmisión de paquetes y

utiliza los slots del canal que no están siendo utilizados por enlaces SCO.

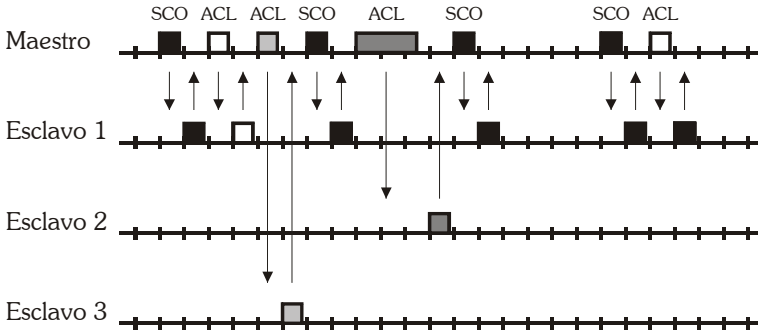


Figura 3.4 - Ejemplo de enlaces SCO y ACL simultaneos

3.4.4 Definición de paquete

La información que se intercambia entre dos unidades Bluetooth se realiza mediante un conjunto de slots que forman un paquete de datos. En una primera clasificación, hay tres tipos de paquetes, los que ocupan 1,3 ó 5 ranuras temporales (slots). Además de las ranuras temporales, los paquetes también se diferencian por los distintos modos de corrección de errores, que son DM (*Data Medium Speed*) o DH (*Data High Speed*). También existen paquetes DV (*Data Voice*), como combinación de datos SCO y ACL en un mismo paquete en una ranura. Por lo que existen nueve tipos de paquetes.

Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54 bits. Éste contiene importante información de control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabeza. Finalmente, la parte que contiene la información, de longitud variable según sea el tipo de paquete.

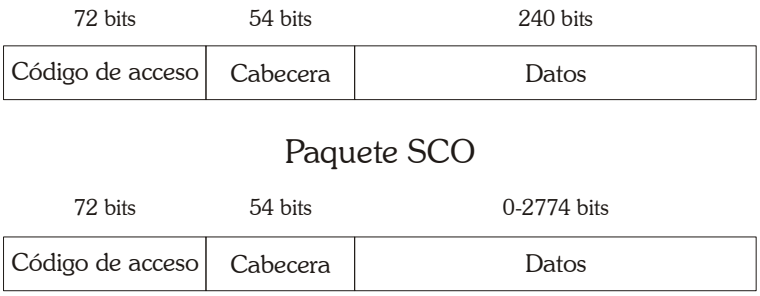


Figura 3.5 - Formato de los paquetes SCO y ACL

3.4.5 Establecimiento de la conexión

Un dispositivo Bluetooth, al principio, si no conoce nada de los dispositivos del entorno, realizará un proceso de búsqueda, y a continuación el proceso de paginación.

- Proceso de búsqueda: Consiste en descubrir los dispositivos que están en su área de cobertura y determina las direcciones y los relojes de los aparatos. El dispositivo fuente envía los paquetes de la búsqueda y espera a recibir respuesta. En el otro lado, el destino, recibe los paquetes de búsqueda siempre que esté en estado de análisis de la búsqueda, y pasará al estado de respuesta, mandando los paquetes de respuesta de la búsqueda a la fuente.
- Proceso de paginación: Consiste en establecer conexión con el otro dispositivo. Únicamente se necesita la dirección del dispositivo Bluetooth con el que se quiere crear la conexión. En el primer paso, el dispositivo fuente pagina el dispositivo destino, en el otro lado, el destino recibe la paginación, entonces, éste manda una contestación a la fuente. Después, la fuente manda un paquete FHS (Frecuency Hop Synchronisation) al destino y éste manda una segunda contestación a la fuente. Por último, la fuente y el destino intercambian características del canal.

3.4.6 Modos de conexión

Un dispositivo Bluetooth en estado de conexión pueden entrar en los modos de ahorro de energía en los cuales la actividad del dispositivo es menor. Existen tres tipos de modos:

- Modo Sniff: En el modo sniff, un dispositivo esclavo escucha a la

piconet a una tasa reducida, lo que reduce su ciclo de trabajo, El intervalo sniff es programable y depende de la aplicación. Tiene el mayor ciclo de vida de los tres modos de ahorro de energía.

- Modo Hold: La unidad maestra puede poner unidades esclavas en modo hold, donde únicamente está funcionando un contador interno. Las unidades esclavas también pueden demandar ser puestas en modo hold. La transferencia de datos vuelve a comenzar de forma instantánea cuando las unidades abandonan este modo.
- Modo Park: En este modo, un dispositivo se encuentra sincronizado a la piconet pero no participa en el tráfico. Los dispositivos en el estado park han abandonado sus direcciones MAC y ocasionalmente escuchan el tráfico del maestro para volver a sincronizarse y comprobar los mensajes broadcast. Tiene el ciclo de trabajo más corto de los tres modos de ahorro de energía.

3.5 Perfiles

Un perfil es un conjunto de características funcionales para cada aplicación que Bluetooth soporta. El concepto de perfil se utiliza para segurar la interoperabilidad entre varias unidades Bluetooth que cumplan los mismos perfiles. Cada dispositivo Bluetooth tiene al menos un perfil, es decir, una aplicación para la cual se puede utilizar el dispositivo. Cuando dos equipos quieran comunicarse entre sí, deben tener un perfil compartido.

El concepto de perfil surge debido a que en pequeños dispositivos el implementar toda la pila de protocolos de Bluetooth es algo innecesario, pues éste solo va a hacer uso de servicios muy específicos (ejemplo un ratón inalámbrico). Los perfiles son dinámicos, en el sentido de que cuando aparecen nuevas aplicaciones no se tiene más que añadir un nuevo perfil a la especificación de Bluetooth.

La posibilidad de definir perfiles en Bluetooth marca una diferencia con otras tecnologías inalámbricas que solamente se centran en capas física y enlace.

Los perfiles se clasifican dentro de modelos de uso. Dentro de cada modelo existen uno o más perfiles que definen las diferentes capas del protocolo bluetooth. Hay muchos modelos de uso, y a medida que pasa el tiempo se diseñan nuevos modelos, por lo que es muy difícil enumerarlos todos.

La especificación inicial de Bluetooth incluía 13 perfiles. Para garantizar la interoperatibilidad en algunas áreas de aplicaciones, el SIG añadió 12 nuevos perfiles.

En la siguiente figura observamos los perfiles de Bluetooth mas utilizados, centrandonos en aquellos mas genéricos:

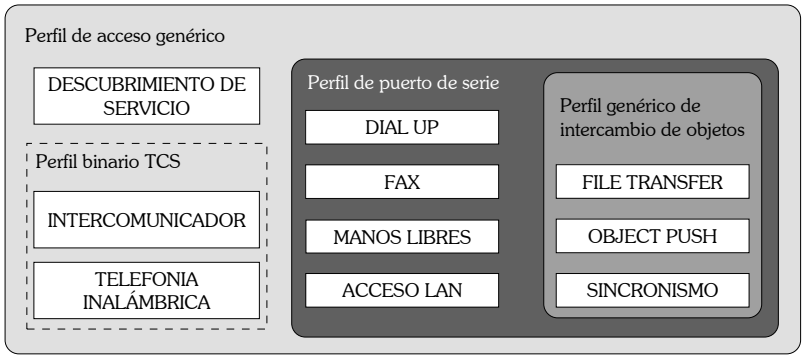


Figura 3.6 - Perfiles de Bluetooth

- Perfil de Acceso Genérico (GAP)
  Este perfil es el más importante de todos, ya que el resto de perfiles depende de el. Define los procedimientos generales para el descubrimiento y establecimiento de conexión entre dispositivos Bluetooth. El GAP asegura que cualquier dispositivo Bluetooth, sin importar el fabricante, pueda intercambiar información con otro cualquiera, para descubrir que tipo de aplicaciones soportan.
- Perfil de Puerto Serie (SPP)
  Este perfil especifica los requisitos necesarios para establecer una conexión emulada de cable serie usando RFCOMM entre dos dispositivos similares. Este perfil solamente requiere soporte para paquetes de un slot. Esto significa que pueden ser usadas ráfagas de datos de hasta 128 Kbps. siendo el soporte para ráfagas más altas opcional. RFCOMM es usado para transportar los datos de usuario, señales de control de módem y comandos de configuración. El perfil de puerto serie es dependiente del GAP.
- Perfil de Descubrimiento de Servicio (SDAP)
  Este perfil concreta los protocolos y procedimientos para una aplicación en un dispositivo Bluetooth donde se desea descubrir y recuperar información relaciona con servicios localizados en otros dispositivos. También depende del GAP.
- Perfil genérico de Intercambios de Objetos (GOEP)
  Este perfil define protocolos y procedimientos usados por aplicaciones

para ofrecer características de intercambio de objetos. Los usos pueden ser, por ejemplo, sincronización, transferencia de archivos o modelo Object Push. Los dispositivos más comunes que usan este modelo son agendas electrónicas, PDAs, teléfonos móviles y teléfonos móviles. El GOEP depende del perfil de puerto serie.

3.6 Pila de protocolos

La pila de protocolos Bluetooth tiene como objetivo que dos dispositivos con la misma pila puedan permitir que las aplicaciones funcionen correctamente entre sí. Cada pila de protocolos usa un enlace de datos y una capa física común.

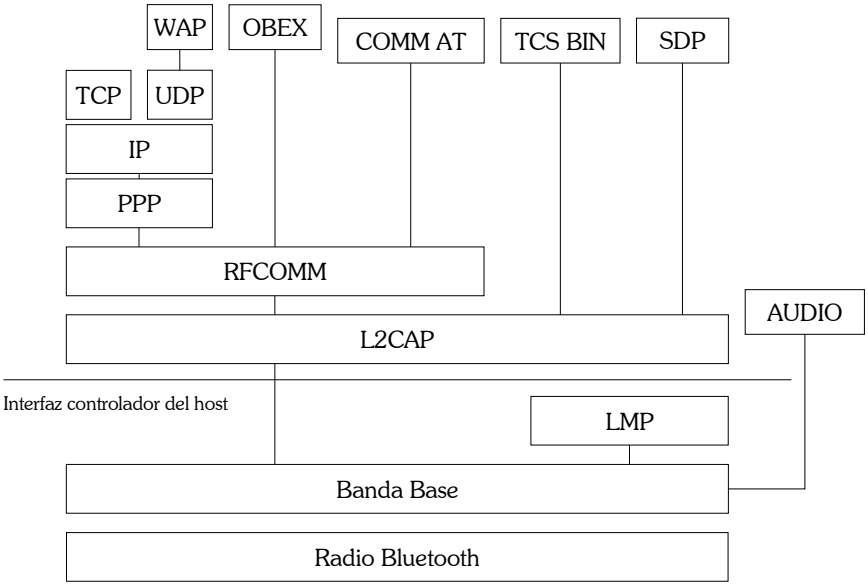


Figura 3.7 - Pila de protocolos Bluetooth

Los protocolos por debajo de la interfaz del controlador del host (HCI) son los protocolos que necesariamente debe contener un dispositivo Bluetooth ya que componen el núcleo de Bluetooth.

El resto de protocolos, no pertenecientes al núcleo de Bluetooth, sólo serán necesarios si lo requiere la aplicación concreta. La mayor o menor complejidad de la pila de protocolos para una aplicación dada vendrá determinada por los requerimientos de ésta. A continuación se explican se explican las características principales de capa del protocolo:

3.6.1 Arquitectura Hardware

Las capas o protocolos inferiores que se encuentran por debajo de HCI, son implementados por el propio hardware del dispositivo Bluetooth. Estas capas dan una interfaz simple para crear comunicaciones de voz y datos entre dos dispositivos.

- Capa de radio
  Es la capa más baja definida en la especificación de Bluetooth. Especifica los requisitos del aparato transceptor. Maneja y controla las señales de radiofrecuencia (RF, *Radio Feecuency*). Un dispositivo Bluetooth se clasifica dentro de 3 tipos de potencia:
  - Nivel de potencia 1: para dispositivos de gran cobertura, 100 metros, con una potencia máxima de 20 dBm.
  - Nivel de potencia 2: para dispositivos de cobertura media, 10 metros, con una potencia máxima de 4 dBm.
  - Nivel de potencia 3: para dispositivos de bajo alcance, 10 centímetros, con una potencia máxima de entorno a 0 dBm.
- Banda base
  Esta capa se encarga de la corrección de errores, seguridad en Bluetooth y selección de salto. También trata los paquetes, hace la paginación y aplica un esquema de división duplex en el tiempo (TDD, *Time Division duplex*), para así poder, alternativamente, ser transmisor y receptor. El maestro empieza su transmisión en los slots pares y el esclavo en los slots impares, para así poder llevar a cabo la transmisión.
- LMP
  Es el protocolo encargado de administrar los recursos preestablecidos por la capa banda base, realiza transición entre modos de potencia, administra los enlaces y realiza funciones de seguridad. Principalmente, el Protocolo de administración del enlace, se encarga de descubrir otros administradores remotos y comunicarse con ellos.

3.6.2 Arquitectura Software

Ahora explicaremos las capas de las pilas Bluetooth que no pertenecen a la parte Hardware. Estas capas solamente son utilizadas por un dispositivo si las necesita para desarrollar la aplicación para la que ha sido desarrollado.

- L2CAP
  Es uno de las capas necesarias para los protocolos de niveles superiores, ya que ofrece servicios de datos a estos protocolos. L2CAP

puede ser visto como un multiplexor, ya que se encarga de coger los datos de las capas superiores y de las aplicaciones y enviarlas, gracias a la interfaz HCI, a las capas bajas de la pila. También se encarga de la segmentación y reensamblado de los paquetes que permite a las capas superiores transmitir paquetes mayores que los que aceptan las capas bajas de la pila. Esta capa, sólo puede transmitir datos, nunca audio.

- SDP
  El protocolo de descubrimiento de servicios sirve para ofrecer u obtener información sabre otros dispositivos Bluetooth, con vistas a comunicarse con ellos y utilizar los servicios que ofrecen. El protocolo SDP se utiliza siempre y su objetivo principal es descubrir que servicios ofrecen otros aparatos Bluetooth.
- RFCOMM
  El protocolo de radio frecuencia orientado a emular puertos serie COM en un PC (RFCOMM, *Radio Frecuency oriented of the serial COM port on a PC*) proporciona la emulación de puertos sucesivos sobre el protocolo L2CAP. RFCOMM es un protocolo de transporte que modula una comunicación a través de l puerto RS-232 sobre un canal L2CAP. Además, RFCOMM proporciona mecanismos para el control del flujo.

3.6.3 Protocolos adaptados

Ante la necesidad de utilizar otros protocolos ya existentes , como IP, se han adptado éstos y es posible tener aplicaciones Bluetooth utilizando TCP/IP (*Transmision Control Protocol / Internet Protocol*) o PPP (*Point-to-point*)

3.7 Seguridad en Bluetooth

Aunque Bluetooth ha sido diseñado principalmente para conexiones de corto alcance entre dispositivos personales o periféricos, se han incluido algunos mecanismos de seguridad para prevenir un uso no autorizado. En el establecimiento de la conexión se lleva a cabo un proceso de autenticación para verificar las identidades de las unidades involucradas en la comunicación.

Para esta seguridad básica se necesita una dirección publica que sea única para cada dispositivo (su dirección), dos claves secretas (clave de autenticación y de encriptación) y un generador de números aleatorios. La dirección se puede obtener a través de un procedimiento de consulta. La clave

privada se deriva durante la inicialización y no es relevada posteriormente. El número aleatorio se genera en un proceso pseudo-aleatorio en cada dispositivo.

Bluetooth proporciona un número limitado de elementos de seguridad. Si se quieren instalar otros procedimientos más avanzados de tendrán que aplicar sobre las capas superiores.

Capítulo 3 - Bluetooth

Capítulo 4

Wi-Fi

4.1 Introducción a Wireless Lan

En los últimos años se ha producido un crecimiento espectacular en lo referente al desarrollo y aceptación de las comunicaciones móviles y en concreto de las redes de área local (Wireless LANs). La función principal de este tipo de redes es la proporcionar conectividad y acceso a las tradicionales redes cableadas (Ethernet, Token Ring...), como si de una extensión de éstas últimas se tratara, pero con la flexibilidad y movilidad que ofrecen las comunicaciones inalámbricas. El momento decisivo para la consolidación de estos sistemas fue la conclusión del estándar IEEE 802.11.

Las redes WLAN no pretenden sustituir a las tradicionales redes cableadas, sino más bien complementarlas. En este sentido el objetivo fundamental de las redes WLAN es el de proporcionar las facilidades no disponibles en los sistemas cableados y formar una red total donde coexistan los dos tipos de sistemas.

4.2 Definición de red de área local inalámbrica

Una red de área local inalámbrica puede definirse como a una red de alcance local que tiene como medio de transmisión el aire. Por red de área local entendemos una red que cubre un entorno geográfico limitado, con una velocidad de transferencia de datos relativamente alta (mayor o igual a 1 Mbps tal y como especifica el IEEE), con baja tasa de errores y administrada de forma privada. Por red inalámbrica entendemos una red que utiliza ondas electromagnéticas como medio de transmisión de la información que viaja a través del canal inalámbrico enlazando los diferentes equipos o terminales móviles asociados a la red.

En las redes tradicionales cableadas esta información viaja fundamentalmente a través de cables coaxiales, pares trenzados o fibra óptica.

Capitulo 4 - Wi-Fi

Una red de área local inalámbrica, también llamada wireless LAN (WLAN), es un sistema flexible de comunicaciones que puede implementarse como una extensión o directamente como una alternativa a una red cableada. Este tipo de redes utiliza tecnología de radiofrecuencia minimizando así la necesidad de conexiones cableadas. Este hecho proporciona al usuario una gran movilidad sin perder conectividad.

El atractivo fundamental de este tipo de redes es la facilidad de instalación y el ahorro que supone la supresión del medio de transmisión cableado. Aún así, debido a que sus prestaciones son menores en lo referente a la velocidad de transmisión que se sitúa entre los 2 y los 54 Mbps frente a los 10 y hasta los 100 Mbps ofrecidos por una red convencional, las redes inalámbricas son la alternativa ideal para hacer llegar una red tradicional a lugares donde el cableado no lo permite, y en general las WLAN se utilizarán como un complemento de las redes fijas.

4.3 Aplicaciones de los sistemas WLAN

Las aplicaciones más típicas de las redes de área local que podemos encontrar actualmente son las siguientes:

- Implementación de redes de área local en edificios históricos, de difícil acceso y en general en entornos donde la solución cableada es inviable.
- Posibilidad de reconfiguración de la topología de la red sin añadir costes adicionales. Esta solución es muy típica en entornos cambiantes que necesitan una estructura de red flexible que se adapte a estos cambios.
- Redes locales para situaciones de emergencia o congestión de la red cableada. Estas redes permiten el acceso a la información mientras el usuario se encuentra en movimiento. Habitualmente esta solución es requerida en hospitales, fábricas, almacenes...
- Generación de grupos de trabajo eventuales y reuniones ad-hoc. En estos casos no merece la pena instalar una red cableada. Con la solución inalámbrica es viable implementar una red de área local aunque sea para un plazo corto de tiempo.
- En ambientes industriales con severas condiciones ambientales este tipo de redes sirve para interconectar diferentes dispositivos y máquinas.
- Interconexión de redes de área local que se encuentran en lugares

físicos distintos. Por ejemplo, se puede utilizar una red de área local inalámbrica para interconectar dos o más redes de área local cableadas situadas en dos edificios distintos.

4.4 IEEE 802.11

En este estándar se encuentran las especificaciones de las dos capas mas bajas de la capa OSI que hay que tener en cuenta a la hora de implementar una red de área local inalámbrica.

La norma 802.11 ha sufrido diferentes extensiones sobre la norma para obtener modificaciones y mejoras. De esta manera, tenemos las siguientes especificaciones:

- 802.11 Especificación para 1-2 Mbps en la banda de los 2.4 GHz, usando salto de frecuencias (FHSS) o secuencia directa (DSSS).
- 802.11b Extensión de 802.11 para proporcionar 11Mbps usando DSSS.
- Wi-Fi (Wireless Fidelity) Promulgado por el WECA para certificar productos 802.11b capaces de inter operar con los de otros fabricantes.
- 802.11a Extensión de 802.11 para proporcionar 54Mbps usando OFDM.
- 802.11g Extensión de 802.11 para proporcionar 20-54Mbps usando DSSS y OFDM. Es compatible hacia atrás con 802.11b. Tiene mayor alcance y menor consumo de potencia que 802.11a.

4.4.1 Arquitectura lógica

El grado de complejidad de una red de área local inalámbrica es variable, dependiendo de las necesidades a cubrir y en función de los requerimientos del sistema que queramos implementar podemos utilizar dos tipos de configuración: infrastructure o peer-to-peer (Ad-Hoc) mode.

4.4.1.1 Modo de configuración Infraestructura

El portátil o dispositivo inteligente, denominado "estación" en el ámbito de las redes LAN inalámbricas, primero debe identificar los puntos de acceso y las redes disponibles. Este proceso se lleva a cabo mediante el control de las tramas de señalización procedentes de los puntos de acceso que se anuncian a sí mismos o mediante el sondeo activo de una red específica con tramas de sondeo.

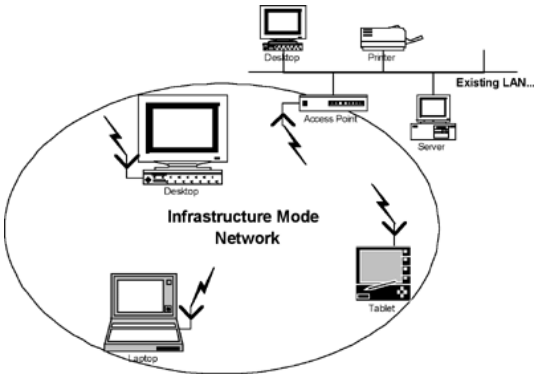


Figura 4.1 - Topología del modo de funcionamiento Infraestructura

La estación elige una red entre las que están disponibles e inicia un proceso de autenticación con el punto de acceso. Una vez que el punto de acceso y la estación se han verificado mutuamente, comienza el proceso de asociación.

La asociación permite que el punto de acceso y la estación intercambien información y datos de capacidad. El punto de acceso puede utilizar esta información y compartirla con otros puntos de acceso de la red para diseminar la información de la ubicación actual de la estación en la red. La estación sólo puede transmitir o recibir tramas en la red después de que haya finalizado la asociación.

En la modalidad de infraestructura, todo el tráfico de red procedente de las estaciones inalámbricas pasa por un punto de acceso para poder llegar a su destino en la red LAN con cable o inalámbrica.

El acceso a la red se administra mediante un protocolo que detecta las portadoras y evita las colisiones. Las estaciones se mantienen a la escucha de las transmisiones de datos durante un período de tiempo especificado antes de intentar transmitir (ésta es la parte del protocolo que detecta las portadoras). Antes de transmitir, la estación debe esperar durante un período de tiempo específico después de que la red está despejada. Esta demora, junto con la transmisión por parte de la estación receptora de una confirmación de recepción correcta, representan la parte del protocolo que evita las colisiones. Observe que, en la modalidad de infraestructura, el emisor o el receptor es siempre el punto de acceso.

Dado que es posible que algunas estaciones no se escuchen

mutuamente, aunque ambas estén dentro del alcance del punto de acceso, se toman medidas especiales para evitar las colisiones. Entre ellas, se incluye una clase de intercambio de reserva que puede tener lugar antes de transmitir un paquete mediante un intercambio de tramas "petición para emitir" y "listo para emitir", y un vector de asignación de red que se mantiene en cada estación de la red. Incluso aunque una estación no pueda oír la transmisión de la otra estación, oirá la transmisión de "listo para emitir" desde el punto de acceso y puede evitar transmitir durante ese intervalo.

El proceso de movilidad de un punto de acceso a otro no está completamente definido en el estándar. Sin embargo, la señalización y el sondeo que se utilizan para buscar puntos de acceso y un proceso de reasociación que permite a la estación asociarse a un punto de acceso diferente, junto con protocolos específicos de otros fabricantes entre puntos de acceso, proporcionan una transición fluida.

La sincronización entre las estaciones de la red se controla mediante las tramas de señalización periódicas enviadas por el punto de acceso. Estas tramas contienen el valor de reloj del punto de acceso en el momento de la transmisión, por lo que sirve para comprobar la evolución en la estación receptora. La sincronización es necesaria por varias razones relacionadas con los protocolos y esquemas de modulación de las conexiones inalámbricas.

4.4.1.2 Modo de configuración Ad-hoc

En una topología ad hoc, los propios dispositivos inalámbricos crean la red LAN y no existe ningún controlador central ni puntos de acceso. Cada dispositivo se comunica directamente con los demás dispositivos de la red, en lugar de pasar por un controlador central. Esta topología es práctica en lugares en los que pueden reunirse pequeños grupos de equipos que no necesitan acceso a otra red. Ejemplos de entornos en los que podrían utilizarse redes inalámbricas ad hoc serían un domicilio sin red con cable o una sala de conferencias donde los equipos se reúnen con regularidad para intercambiar ideas.

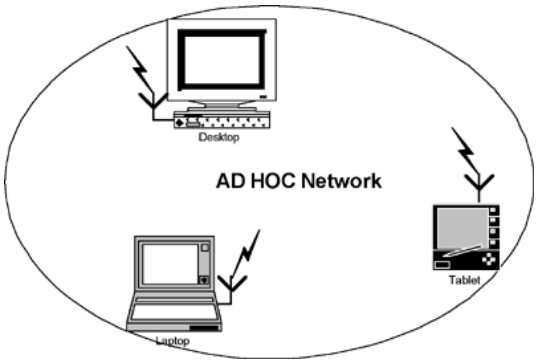


Figura 4.2 - Topología del modo de funcionamiento Ad-Hoc

Las redes 802.11 pueden extenderse arbitrariamente, para ello se enlazan varios BSS, con lo que se forma lo que se llama área de servicio extendida (Extended Service Set, ESS). Las BSS se encadenan mediante una red de transporte. Esta red de transporte se denomina sistema de distribución (Distribution System, DS). La especificación no señala una tecnología en particular para realizar esta función, sólo requiere que proporcione una serie de servicios.

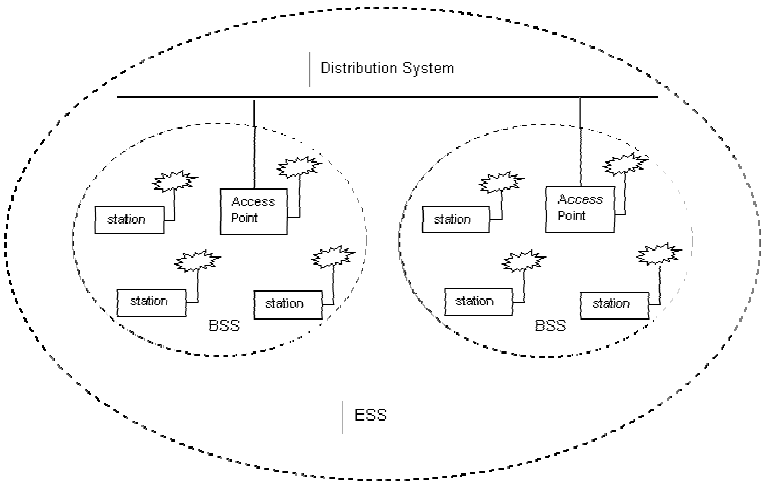


Figura 4.3 - Extended Service Set (ESS)

La ESS se forma por la unión de varias BSS mediante el sistema de distribución. Las estaciones dentro de una misma ESS pueden comunicarse entre todas ellas, independientemente de si pertenecen a la misma BSS o a diferentes, o incluso si se están moviendo entre ellas. Una estación puede moverse de una BSS a otra, con lo que se asocia a un AP distinto, el sistema de

distribución debe gestionar estas asociaciones, debe permitir que los AP comuniquen al resto de los AP qué estaciones están asociadas a ellos. Es necesario un método para realizar estas notificaciones. No hay, de momento, un método estandarizado, aunque el grupo 802.11 trabaja en ello.

4.4.2 Límites de movilidad

IEEE 802.11 soporta la movilidad de las estaciones garantizando el mantenimiento de la conexión en las transiciones entre BSS. Las estaciones monitorizan continuamente la calidad de la señal de todos los AP que han sido asignados a cubrir una ESS. Cuando una estación detecta una señal mejor de un AP distinto al que está asociado, termina su asociación con el primero y se asocia con el segundo. El DS permitirá que la estación reciba las tramas a través del nuevo AP. Se mantiene la conexión de nivel de enlace en todo momento. Sin embargo, no garantiza el mantenimiento de las conexiones entre ESS.

4.4.3 Servicios de red

Una forma de definir una tecnología es declarar los servicios que debe proporcionar y permitir que los vendedores lo implementen de la manera que consideren adecuada. IEEE 802.11 define nueve servicios que deben proporcionar las redes inalámbricas, de tal manera que la funcionalidad que proporcionen sea equivalente a la de una LAN de cable.

Los servicios se describen a continuación:

- Distribución. Lo usan las estaciones para transmitir tramas en redes de infraestructura. Una vez que la trama ha sido aceptada por el AP, usa el sistema de distribución para entregar la trama a la estación destino.
- Integración. Se encarga de la función de pasarela con otros sistemas IEEE802.x. En concreto, define el componente portal que se encargará de aspectos necesarios como redireccionamiento.
- Asociación. Servicio necesario para que una estación pueda adherirse al modo infraestructura y utilizar sus servicios.
- Reasociación. Permite que una asociación se transfiera de un AP a otro. La reasociación la inicia una estación cuando las condiciones de la señal indican que asociarse con otro AP sería beneficioso. Cuando la reasociación termina, el DS actualiza sus registros de localización.
- Disociación. Para terminar una asociación. Es una notificación y por tanto no puede ser rechazada. Las estaciones deberían disociarse al abandonar la red.
- Autenticación y Deautentificación. La autenticación se usa para

establecer la identidad una estación. El estándar soporta diferentes métodos de autenticación, desde la autenticación de sistema abierto (método por defecto, pero inseguro) hasta sistemas de clave pública, aunque no obliga a usar ninguno en particular. Este servicio se usa obligatoriamente antes de establecer la asociación. El estándar especifica el servicio complementario, es decir, la finalización de una relación autenticada.

- Privacidad. Este servicio utilizará WEP para el encriptado de los datos en el medio.
- Entrega de tramas (MSDU) entre STAs. Es el servicio básico que implementa toda estación: la transferencia de datos al destino.

4.4.4 Nivel de acceso al medio MAC

Los diferentes métodos de acceso de IEEE802 están diseñados según el modelo OSI y se encuentran ubicados en el nivel físico y en la parte inferior del nivel de enlace o subnivel MAC.

Además, la capa de gestión MAC de 802.11 cubre tres áreas funcionales: control de acceso, entrega fiable y seguridad.

4.4.4.1 Descripción Funcional MAC

La arquitectura MAC del estándar 802.11 se compone de dos funcionalidades básicas que controlan el acceso al medio: la función de coordinación puntual (PCF) y la función de coordinación distribuida.

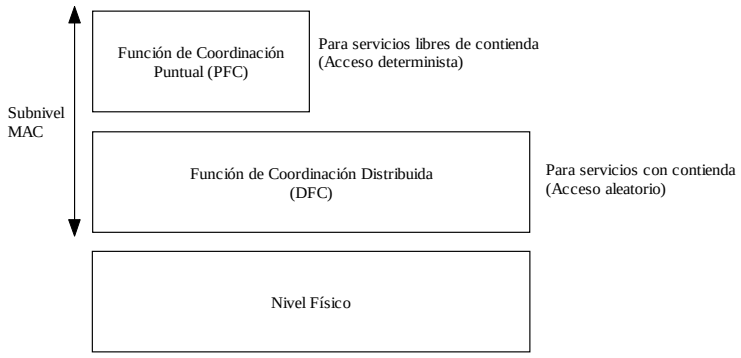


Figura 4.4 - Descripción funcional Mac

4.4.4.1.1 DFC Función de Coordinación Distribuida

La función de coordinación es la funcionalidad que determina, dentro de un conjunto básico de servicios (BSS), cuándo una estación puede transmitir y/o recibir unidades de datos de protocolo a nivel MAC a través del medio inalámbrico. En el nivel inferior del subnivel MAC se encuentra la función de coordinación distribuida y su funcionamiento se basa en técnicas de acceso aleatorias de contienda por el medio.

Las características de DFC las podemos resumir en estos puntos:

- Utiliza MACA (CSMA/CA con RTS/CTS) como protocolo de acceso al medio
- Necesario reconocimientos ACKs, provocando retransmisiones si no se recibe
- Usa campo Duration/ID que contiene el tiempo de reserva para transmisión y ACK. Esto quiere decir que todos los nodos conocerán al escuchar cuando el canal volverá a quedar libre
- Implementa fragmentación de datos
- Concede prioridad a tramas mediante el espaciado entre tramas (IFS)
- Soporta Broadcast y Multicast sin ACKs

4.4.4.1.2 PFC Función de Coordinación Puntual

Por encima de la funcionalidad DCF se sitúa la función de coordinación puntual, PCF, asociada a las transmisiones libres de contienda que utilizan técnicas de acceso deterministas. El estándar IEEE 802.11, en concreto, define una técnica de interrogación circular desde el punto de acceso para este nivel. Esta funcionalidad está pensada para servicios de tipo síncrono que no toleran retardos aleatorios en el acceso al medio.

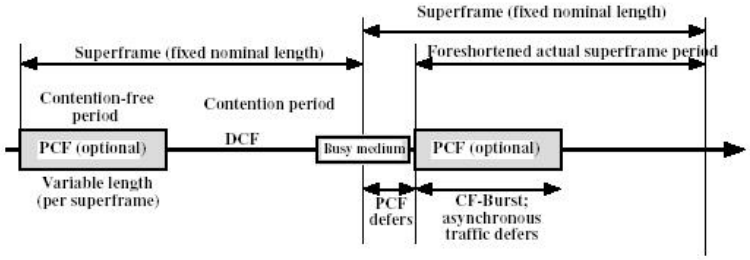


Figura 4.5 - Intervalos de Supertrama

Estos dos métodos de acceso pueden operar conjuntamente dentro de una misma celda o conjunto básico de servicios dentro de una estructura llamada supertrama. Al comienzo de la supertrama el AP sondea las estaciones (aquellas configuradas para usar PCF) siguiendo un esquema round-robin. Les

envía los datos dirigidos a ellas y pregunta si tienen algo que transmitir. Para asegurar la captura del medio, el AP establece la duración del intervalo sin contienda mediante el NAV, (Network Allocation Vector) que mantendrá una predicción de cuando el medio quedará liberado. El resto del tiempo se dedica a tráfico asíncrono. El intervalo de supertrama tiene una duración nominal fija. Al final de este intervalo, el AP lucha por el medio usando del PIFS, Point Coordination Function IFS, que es un tipo de espaciado entre tramas o IFS. De duración media. Es usado por el PCF para enviar tráfico sin contienda previa.. Si el medio está ocupado, el AP espera antes de empezar el periodo sin contienda, que estará acortado en el siguiente intervalo de supertrama, por tanto.

El nodo utilizará una trama para la configuración de la supertrama, llamada Beacon, donde establecerá una CFRate o tasa de periodos de contienda. Pese a que el periodo de contienda se puede retrasar por estar el medio ocupado, la tasa se mantendrá en el siguiente periodo con medio libre.

4.4.4.2 Protocolo de Acceso al medio CSMA/CA

El algoritmo básico de acceso a este nivel es muy similar al implementado en el estándar IEEE 802.3 y es el llamado CSMA/CA (Carrier Sense Multiple Access / Collision Advoidance). Este algoritmo funciona tal y como describimos a continuación:

El mecanismo de CSMA/CA se basa en un conjunto de retardos que permiten un cierto esquema de prioridades. Se define un espacio entre tramas (Interframe Spacing, IFS), que es un determinado tiempo de retardo. En realidad, hay tres tiempos distintos, pero el algoritmo se explica mejor ignorando este detalle.

Las reglas del algoritmo son las siguientes:

1. Una estación que quiere transmitir comprueba el medio. Esta comprobación es virtual y física. Es decir, mientras el NAV no sea cero, el medio estará ocupado. Cuando el NAV es cero se comprueba físicamente el medio. Si el medio está libre, la estación espera un tiempo igual al IFS. Si al finalizar este intervalo el medio continúa libre, transmite.
2. Si el medio está ocupado (porque estaba ocupado o porque se ocupó durante el IFS) la estación abandona la transmisión y continúa

monitorizando hasta que la transmisión en curso acabe.

3. Cuando la transmisión acaba, la estación espera otro IFS. Si el medio permanece libre ese último intervalo, la estación espera un tiempo aleatorio (usa el algoritmo de retroceso exponencial binario, como en Ethernet) y de nuevo comprueba el medio. Si sigue libre, la estación puede transmitir. Si durante el tiempo de espera (backoff) el medio se ocupa, el contador se para y continúa al volver a desocuparse el medio.

Para asegurar la estabilidad del algoritmo, se utiliza exactamente el mismo algoritmo de espera que Ethernet: el algoritmo de retroceso exponencial binario. El tiempo se ranura, con un tiempo de slot que depende de la velocidad de transmisión; se elige un entero aleatorio y en función de él se transmite en un slot u otro. Cada vez que se reintenta la transmisión, el intervalo de slots aumenta.

En CSMA/CD el reintento se produce cada vez que se detecta una colisión. Con un medio inalámbrico no se puede detectar la colisión, por tanto, se utiliza el mecanismo de reconocimiento para detectar errores. Como se comentó previamente, la transmisión de una trama y su reconocimiento es una operación atómica. Si el transmisor no recibe un ACK, incrementa el contador de reintentos. Esto implica que el intervalo del algoritmo de retroceso se extenderá en el siguiente intento. Al llegar a cierto número de reintentos, se descarta la trama.

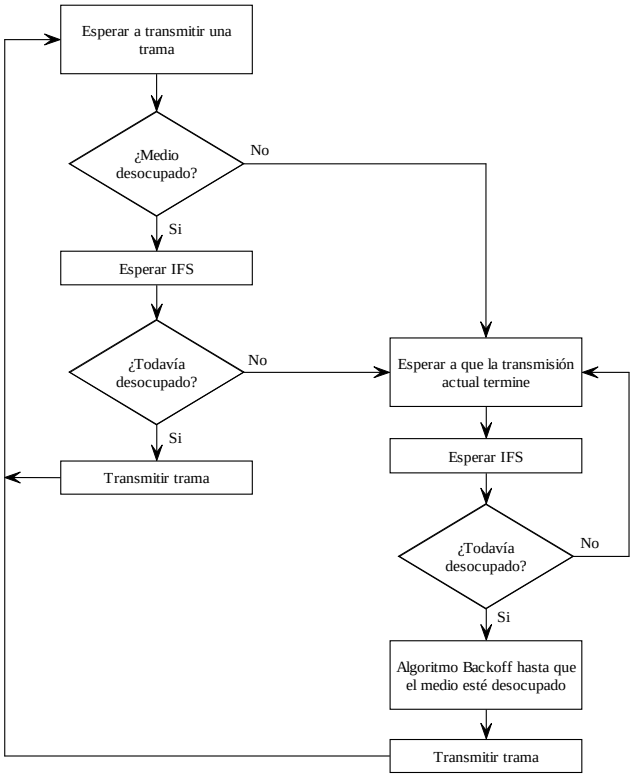


Figura 4.6 - Algoritmo de acceso al medio

Sin embargo, CSMA/CA en un entorno inalámbrico y celular presenta una serie de problemas que intentaremos resolver con alguna modificación. Los dos principales problemas que podemos detectar son:

- Nodos ocultos. Una estación cree que el canal está libre, pero en realidad está ocupado por otro nodo que no oye
- Nodos expuestos. Una estación cree que el canal está ocupado, pero en realidad está libre pues el nodo al que oye no le interferiría para transmitir a otro destino.

La solución que propone 802.11 es MACA o MultiAccess Collision Avoidance. Según este protocolo, antes de transmitir el emisor envía una trama RTS (Request to Send),

4.4.4.3 Espaciado entre tramas IFS

El tiempo de intervalo entre tramas se llama IFS. Durante este periodo

mínimo, una estación STA estará escuchando el medio antes de transmitir. Se definen cuatro espaciados para dar prioridad de acceso al medio inalámbrico. Veámoslos de más cortos a más largos

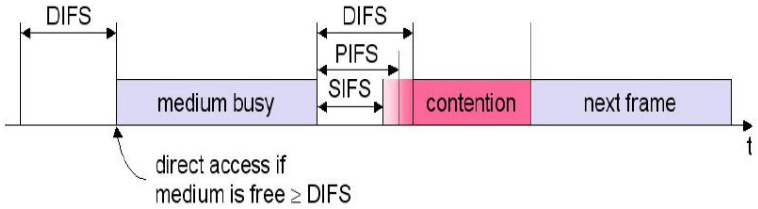


Figura 4.7 - Espaciado entre tramas IFS

- SIFS (Short IFS). Este es el periodo más corto. Se utiliza fundamentalmente para transmitir los reconocimientos. También es utilizado para transmitir cada uno de los fragmentos de una trama. Por último, es usado por el PC o Point Control para enviar testigo a estaciones que quieran transmitir datos síncronos
- PIFS (Point Coordination Function IFS). Es utilizado por los PCF para ganar prioridad de acceso en los periodos libres de contienda. Lo utiliza el PCF para ganar la contienda normal, que se produce al esperar DIFS.
- DIFS (Distributed Coordination Function IFS). El IFS de mayor longitud. El tiempo mínimo que tiene que estar el canal libre para el acceso por contienda.
- EIFS (Extended IFS). Controla la espera en los casos en los que se detecta la llegada de una trama errónea. Espera un tiempo suficiente para que le vuelvan a enviar la trama u otra solución.

4.4.4.4 Conocimiento del medio

Las estaciones tienen un conocimiento específico de cuando la estación, que en estos momentos tiene el control del medio porque está transmitiendo o recibiendo, va a finalizar su periodo de reserva del canal.

Esto se hace a través de una variable llamada NAV (Network Allocation Vector) que mantendrá una predicción de cuando el medio quedará liberado.

Tanto al enviar un RTS como al recibir un CTS, se envía el campo Duration/ID con el valor reservado para la transmisión y el subsiguiente reconocimiento. Las estaciones que estén a la escucha modificarán su NAV según el valor de este campo Duration/ID. En realidad, hay una serie de normas

para modificar el NAV, una de ellas es que el NAV siempre se situará al valor más alto de entre los que se disponga.

4.4.4.5 Entrega Fiable de datos

Para minimizar los problemas del medio inalámbrico se incorporan dos mecanismos. Por una parte, para evitar el problema de la pérdida de tramas en el canal, se incorpora un mecanismo de reconocimiento positivo. Todas las tramas transmitidas deben ser reconocidas (mediante una trama ACK). Este intercambio se trata como una operación atómica y no puede ser interrumpido por la transmisión de otra estación.

Por otra parte, para solucionar el problema de los nodos ocultos se utiliza un mecanismo de RTS/CTS. Las tramas RTS/CTS se transmiten antes de la trama de datos. Las estaciones que escuchan estas tramas ceden el medio durante un tiempo determinado. Además, la trama de datos debe ser reconocida, por tanto, la transmisión se produce en 4 pasos. Esto implica que se consume una cantidad considerable de capacidad, además de la latencia añadida.

Este mecanismo se usa sólo en entornos de alta ocupación. El mecanismo se puede deshabilitar y además se puede controlar un umbral de RTS (RTS threshold). Sólo a las tramas cuyo tamaño sea mayor que el umbral se les aplica el mecanismo de RTS/CTS.

4.4.4.6 Fragmentación y reensamblado

A diferencia de Ethernet y de las tecnologías de nivel de enlace de datos, el MAC de 802.11 sí que fragmenta y reensambla paquetes. El canal inalámbrico tiene un nivel muy alto de interferencias y es preferible enviar paquetes cortos para evitar la retransmisión de tramas largas dañadas.

La fragmentación se produce cuando el tamaño de un paquete excede un umbral de fragmentación. Las tramas enviadas tienen el mismo número de secuencia y un número ascendente de fragmentos para permitir el reensamblado. Todos los paquetes que componen una trama se envían en una ráfaga de fragmentación, es decir, utilizando el SIFS después de cada ACK del receptor para cada fragmento. Además se utiliza el NAV para asegurar el control del medio.

El umbral de fragmentación suele coincidir con el umbral de RTS, es decir, que los paquetes fragmentados usan el mecanismo RTS/CTS. En definitiva, una estación captura el medio durante todo el tiempo necesario para enviar todos los fragmentos.

4.4.5 Formato de la trama

Para cubrir los requerimientos del medio inalámbrico, así como los servicios ofrecidos por 802.11 se tiene que aumentar la complejidad del formato de trama 802.11. El formato de trama se muestra en la siguiente figura El preámbulo lo incorpora la capa de nivel físico.

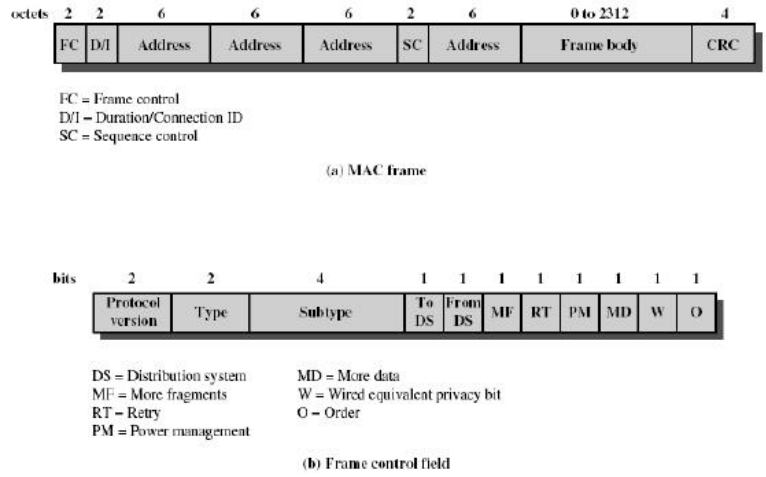


Figura 4.8 - Formato de la trama 802.11

La característica más notable es que se puede usar hasta 4 campos de direcciones. No todas las tramas usan los 4 campos de direcciones y los valores asignados al campo varían en función del tipo de trama. Los campos que incluye son los siguientes:

- Frame Control. Se utiliza para identificar el tipo de trama. Es necesario porque para ofrecer todos los servicios (asociación, disociación, etc.) hay que utilizar diferentes tipos de trama.
- Type y subtype. Indican el tipo de trama usado. La figura 8, muestra todos los tipos posibles.
- ToDS y FromDS. Indican si una trama está destinada al sistema de distribución. La interpretación de los campos de direcciones depende de estos bits.
- More fragments. Es necesario para indicar si quedan más fragmentos.
- Retry. Si la trama es una retransmisión, para evitar duplicados en el receptor.

- Power Management. Este bit indica si el origen pasará a modo de ahorro de potencia después de este intercambio.
- More Data. Indica si quedan más datos para transmitir.
- WEP. Indica si la trama ha sido procesada con el algoritmo WEP.
- Order. Indica a la estación que las tramas deben ser procesadas en orden.
- Duration/ID. Se utiliza, en función del tipo de trama, para establecer un valor de NAV o para indicar el identificador de asociación, es decir, la BSS a la que pertenece la estación.
- Campos de direcciones. Las direcciones siguen el formato de 48 bits de IEEE 802. Los campos de dirección están numerados porque se usan diferentes campos para diferentes propósitos en función del tipo de trama.

La regla general sería que la Dirección 1 se usa para el receptor, la Dirección 2 para el transmisor y la Dirección 3 para filtrados del receptor. La mayoría de las tramas de datos usan 3 campos, para destino, origen e ID de BSS respectivamente. El número y posición de los campos en una trama de datos depende de cómo la trama viaja en relación con el sistema de distribución.

- Sequence Control. Este campo se usa para los números de secuencia, necesarios al usar reconocimiento, y para numerar los fragmentos, necesarios al utilizar fragmentación.
- Cuerpo. Es el campo de datos. El tamaño máximo es de 2304 octetos.
- Frame Check Sequence. Es el campo de control de errores. Consiste en un CRC de 32 bits.

Hay 3 tipos de tramas, que incluyen diferentes subtipos:

- Tramas de control. Ayudan en la entrega fiable de tramas de datos.
- Tramas de datos. Hay hasta 8 subtipos, organizados en dos grupos.
- Tramas de gestión. Gestionan la comunicación entre AP y estaciones.

4.4.6 Unión a la red

Cuando una estación entra en funcionamiento, debe determinar si hay otra estación o un punto de acceso presente, para asociarse con ellos. La estación realiza un proceso de descubrimiento antes de empezar con la asociación. Hay dos formas de realizar el descubrimiento:

- Rastreo pasivo (passive scanning). Para cada canal, la estación escucha durante un determinado tiempo. Cada AP emite señales de faro que incluyen el identificador de la BSS. Una vez detectada la BSS con el identificador deseado se inicia el proceso de autenticación y asociación.
- Rastreo activo. La estación envía tramas sonda indicando el identificador de la red a la que quiere unirse. Entonces espera una trama de respuesta de la red a la que quiere unirse. Una estación también puede enviar una sonda con el identificar de red broadcast, que hará que todas las redes en la vecindad respondan. Responden los AP y, en el caso de redes ad-hoc, la última estación que generó una trama de faro.

4.4.7 Nivel físico

IEEE 802.11 define varios niveles físicos, en diversas especificaciones. Sus características resumidas son las siguientes:

- 802.11. Define tres niveles físicos.
  - DSSS a 2.4 GHz con velocidades de 1 y 2 Mbps.
  - FHSS a 2.4 GHz con velocidades de 1 y 2 Mbps.
  - Infrarrojos entre 850 y 950 nm con velocidades de 1 y 2 Mbps.
- 802.11b. Es el más usado. DSSS a 2.4 GHz con velocidad de 5.5 y 11 Mbps.
- 802.11g. El más reciente. DSSS a 2.4 GHz con velocidad de 54 Mbps.
- 802.11a. Modulación OFDM a 5 GHz con velocidad de 54 Mbps.

En la arquitectura del sistema la capa física se divide en dos subcapas: la Physical Layer Convergence Procedure (PLCP) y la Physical Medium Dependent (PMD) cuyas misiones son la detección de portadora, la transmisión y la recepción. La primera se encarga de añadir el preámbulo y la cabecera física. La segunda se encarga de transmitir y recibir los bits por la antena.

Para asegurar la compatibilidad entre versiones, la cabecera de nivel físico indica la velocidad a la que se transmitirán los datos. Entonces, todas las cabeceras (y preámbulo) se transmiten a 1 Mbps, independientemente de la velocidad a la que se transmitan luego los datos, es decir, la trama MAC.

La mayoría de los productos, si detectan un nivel alto de interferencias, reducen la velocidad de transmisión. Nivel físico 802.11 y 802.11b por espectro ensanchado Puesto que es el nivel físico más usado se describirán brevemente sus características. El nivel físico de 802.11b está basado

en el nivel físico de espectro ensanchado de 802.11.

Para 802.11, la secuencia de ensanche es un código Barker de 11 chips {+1,- 1,+1,+1,-1,+1,-1,-1,-1}. Esta secuencia se suma modulo-2 con la señal de datos. En función de la velocidad de los datos se utiliza una modulación distinta: para 1 Mbps se utiliza DBPSK y para 2 Mbps se utiliza DQPSK. Hay que prestar atención al hecho de que todos los equipos y redes usan la misma secuencia de Barker. Por tanto, no se está utilizando CDMA para el acceso al medio, sino simplemente técnicas de espectro ensanchado para reducir interferencias. Para conseguir altas velocidades, 802.11b ya no usa la secuencia de Barker, sino una codificación bastante compleja, denominada Complementary Code Keying (CCK), que posteriormente se modula con DQPSK.

Capítulo 5

XML

El XML (eXtended MarkUp Language) está llamado a ser el nuevo estándar en la gran telaraña de la World Wide Web, de momento este lenguaje ya ha recibido el apoyo de algunas de las compañías informáticas más importantes como IBM o Microsoft.

5.1 Introducción

El XML es un descendiente del SGML (Standard Generalised Markup Language), pero está orientado, al igual que el HTML (HyperText MarkUp Language) , a aplicaciones basadas en la Web. El XML está llamado a ser el sucesor del que hoy por hoy es el lenguaje más utilizado en la creación de documentos web: el HTML. Frente a éste tiene algunas ventajas, especialmente a nivel definición de etiquetas.

Pero lo que más hace destacar a este lenguaje frente a su antecesor es su carácter abierto, ya que al contrario que en HTML, donde las etiquetas vienen predefinidas, XML permite definir nuestras propias etiquetas. El lenguaje XML permite trabajar de una forma más óptima con los datos, ya que aporta un componente más a los que el HTML ya aportaba, las entidades (en inglés entities) que permite al sistema saber el tipo de dato con el que está trabajando.

Esta nueva característica que aporta el XML, permitirá desarrollar un trabajo más efectivo a la hora de consultar una base de datos; esto a nivel intranet, ya que realmente para el entorno para el que ha sido pensado, Internet, permite, por ejemplo, mejorar de manera espectacular la búsquedas en dicho entorno.

Sobre todo en el campo del comercio electrónico, ya que permitirá que nuestro sistema reciba la información sobre, por ejemplo, los tejidos disponibles en el mercado, aunque cada fabricante use un software de base de datos completamente distinto. Así podremos consultar las últimas novedades por

ejemplo en música, recibiéndolas en nuestra base de datos personal sin tener que perder un tiempo precioso en pasar al formato debido la información que nos va llegando de cada tipo de sistema, y sin tener que instalar una aplicación para cada formato de fichero.

Son muchas las novedades que aporta este lenguaje frente a sus antecesores pero destacan las siguientes:

- Es un lenguaje diseñado de forma que sea más sencilla su programación a los desarrolladores.
- Al estar basado en el extendido lenguaje SGML, no se necesitan crear nuevas aplicaciones para utilizarlo, ya que éste posee un gran número de ellas.
- Todo en el XML ha sido pensado para el mejor aprovechamiento de la Red, es un lenguaje "maduro", que ha surgido fruto de la experiencia con HTML y SGML.
- Al contrario que HTML, en XML se pueden desarrollar nuevas etiquetas, definiéndolas previamente.
- En XML la interactividad entre programas, para que estos usen fragmentos de código unos de otros, es total.
- XML al igual que sus antecesores es un lenguaje "sin propietario", ya que se trata de un estándar internacional.
- Además permite usar distintos códigos de signos, logrando así abarcar todas las lenguas del planeta.

Antes de crear un fichero XML, debemos de crear un fichero DTD (Document Type Definition), en el cual se deben definir todas las etiquetas que se están utilizando. Después ese fichero XML si es correcto es validado, creándose así un fichero con extensión XML accesible, utilizando el estilo XSL, desde los navegadores, que soportan esta utilidad.

5.2 XML frente a SGML

El XML es ni más ni menos que una versión mejorada de SGML, como se ha dicho antes, al cual se han añadido nuevas características orientadas a hacer de éste un lenguaje óptimo para Internet. Al ser el XML un sucesor del SGML, éste puede utilizar todas las herramientas del anterior (que son bastantes) debido a su gran éxito, de modo que XML está empezando a ser conocido de forma mucho más rápida que otros lenguajes.

	HTML/DHTML	XML	SGML
Gramática	Fija y no ampliable	Extensible	Extensible
Estructura	Monolítica	Jerárquica	Jerárquica
Nº de marcas	Fijas	Sin límite	Sin límite
Complejidad	Baja	Mediana	Alta
Diseño de páginas	Fijado por tags. Etiquetas con atributos CSS en DHTML	CSS o XSL	DSSSL
Enlaces	Simples enlaces	Poderosos enlaces (XLL)	HyTime
Exportabilidad (formatos/aplicaciones)	No	Sí	Sí
Validación	Sin validación	Pueden validarse	Obligatorio DTD
Búsquedas	Simple y a veces resuelta por *scripts* o CGI	Potente búsqueda. Con capacidad para personalizarla	Son posibles potentes búsquedas.
Indización/Catalogación de páginas web	Sólo lo permite los atributos de la etiqueta <META>, e implementaciones como DC.	Una descripción abierta y personalizable con el RDF.	Algún proyecto como TEI, DLI, etc.

5.3 Estructura de un documento XML

Lo primero que debemos saber es que hay dos tipos de documentos XML: válidos y bien formados. Éste es uno de los aspectos más importantes de este lenguaje, así que hace falta entender bien la diferencia:

- Bien formados
  Son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas que después se van a explicar, sin estar sujetos a unos elementos fijados en un DTD (luego veremos lo que es un DTD). De hecho los documentos XML deben tener una estructura jerárquica muy estricta, de la que se hablará más tarde, y los documentos bien formados deben cumplirla.
- Válidos
  Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD: sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

En un DTD se lleva acabo la definición de los elementos que puede haber en el documento XML, y su relación entre ellos, sus atributos, posibles

valores, etc. De hecho DTD son las siglas de Document Type Definition, o Definición de Tipo de Documento. En definitiva es una especie de definición de la gramática del documento.

Es importante entender que cuando se procesa cualquier información formateada mediante XML, lo primero es comprobar si está bien formada, y luego, si incluye o referencia a un DTD, comprobar que sigue sus reglas gramaticales.

Hay pues diferencia entre los parsers que procesan documentos XML sin comprobar que siguen las reglas marcadas por un DTD (sólo comprueban que está bien formado), que se llaman parsers no validadores, y los que sí lo hacen, que son parsers validadores (comprueba que además de bien formado, se atiene a su DTD y es válido).

A continuación podemos ver un ejemplo de documento XML muy sencillo, que iremos estudiando para ver las características del lenguaje:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ficha>
<nombre>Angel</nombre>
<apellido>Barbero</apellido>
<direccion>c/Ulises, 36</direccion>
</ficha>
```

Lo primero que tenemos que observar es la primera línea. Con ella deben empezar todos los documentos XML, ya que es la que indica que lo que la sigue es XML. Aunque es opcional, es más que recomendable incluirla siempre. Puede tener varios atributos (los campos que van dentro de la declaración), algunos obligatorios y otros no:

- version: indica la versión de XML usada en el documento. La actual es la versión 1.0, con lo que no debe haber mucho problema. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía (ya veremos qué documentos externos puede haber).
- encoding: la forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.
- standalone: indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay porqué ponerlo, porque

luego se indica el DTD si se necesita.

5.3.1 Sintaxis de un documento XML

Antes de entrar en el estudio de las etiquetas, hay que resaltar algunos detalles importantes y a los que nos debemos acostumbrar:

- Los documentos XML son sensibles a mayúsculas, esto es, en ellos se diferencia las mayúsculas de las minúsculas. Por ello <FICHA> sería una etiqueta diferente a <ficha>.
- Además todos los espacios y retornos de carro se tienen en cuenta (dentro de las etiquetas, en los elementos).
- Hay algunos caracteres especiales reservados, que forman parte de la sintáxis de XML: <, >, &, " y '. En su lugar cuando queramos representarlos deberemos usar las entidades <, >, &, " y ' respectivamente. Más adelante hablaré de las entidades y lo que son, pero basta con saber ahora que si escribimos cualquiera de las secuencias anteriores equivaldrá a los correspondientes caracteres citados.
- Los valores de los atributos de todas las etiquetas deben ir siempre entrecomillados. Son válidas las dobles comillas (") y la comilla simple (').

Pasando al contenido en sí, vemos etiquetas que nos recuerdan a HTML, y que contienen los datos. Es importante diferenciar entre elementos y etiquetas: los elementos son las entidades en sí, lo que tiene contenido, mientras que las etiquetas sólo describen a los elementos. Un documento XML está compuesto por elementos, y en su sintáxis éstos se nombran mediante etiquetas.

Hay dos tipos de elementos: los vacíos y los no vacíos. Hay varias consideraciones importantes a tener en cuenta al respecto:

- Toda etiqueta no vacía debe tener una etiqueta de cerrado: <etiqueta> debe estar seguida de </etiqueta>. Esto se hace para evitar la aberración (en el buen sentido de la palabra) a la que habían llegado todos los navegadores HTML de permitir que las etiquetas no se cerraran, lo que deja los elementos sujetos a posibles errores de interpretación.
- Todos los elementos deben estar perfectamente anidados: no es válido poner:

```
<ficha><nombre>Angel</ficha></nombre>
```

y sí lo es sin embargo:

```
<ficha><nombre>Angel</nombre></ficha>
```

- Los elementos vacíos son aquellos que no tienen contenido dentro del documento. Un ejemplo en HTML son las imágenes. La sintáxis correcta para estos elementos implica que la etiqueta tenga siempre esta forma: <etiqueta/>.

Hasta aquí la sintáxis de XML resumida. Aunque la especificación entera es más prolija en cuanto a detalles sintácticos, codificaciones. Ahora quedan por ver otros aspectos, el más prioritario, los DTD.

5.4 DTD: Definición de Tipos de Documento

Como antes se comentó, los documentos XML pueden ser válidos o bien formados. En cuanto a los válidos, ya sabemos que su gramática está definida en los DTD.

Pues bien, los DTD no son más que definiciones de los elementos que puede incluir un documento XML, de la forma en que deben hacerlo (qué elementos van dentro de otros) y los atributos que se les puede dar. Normalmente la gramática de un lenguaje se define mediante notación EBNF; si alguno la conoce, se habrá dado cuenta de que es bastante engorrosa. Pues el DTD hace lo mismo pero de un modo más intuitivo.

Hay varios modos de referenciar un DTD en un documento XML:

- Incluir dentro del documento una referencia al documento DTD en forma de URI (Universal Resource Identifier, o identificador universal de recursos) y mediante la siguiente sintáxis:

```
<!DOCTYPE ficha SYSTEM "http://www.corel.es/ficha.dtd">
```

  En este caso la palabra SYSTEM indica que el DTD se obtendrá a partir de un elemento externo al documento e indicado por el URI que lo sigue, por supuesto entrecomillado.
- O bien incluir dentro del propio documento el DTD de este modo:

```
<?xml version="1.0"?>
<!DOCTYPE ficha [
          <!ELEMENT ficha (nombre+, apellido+,
          direccion+, foto?)>
          <!ELEMENT nombre (#PCDATA)>
          <!ATTLIST nombre sexo (masculino|femenino)
          #IMPLIED>
          <!ELEMENT apellido (#PCDATA)>
          <!ELEMENT direccion (#PCDATA)>
          <!ELEMENT foto EMPTY>
]>
<ficha>
<nombre>Angel</nombre>
<apellido>Barbero</apellido>
<direccion>c/Ulises, 36</direccion>
</ficha>
```

La forma de incluir el DTD directamente como en este ejemplo pasa por añadir a la declaración <!DOCTYPE y después del nombre del nombre del tipo de documento, en vez de la URI del DTD, el propio DTD entre los símbolos '[' y ']'. Todo lo que hay entre ellos será considerado parte del DTD.

En cuanto a la definición de los elementos, es bastante intuitiva: después de la cláusula <!ELEMENT se incluye el nombre del elemento (el que luego se indicara en la etiqueta), y después diferentes cosas en función del elemento:

- entre paréntesis, si el elemento es no vacío, se indica el contenido que puede tener el elemento: la lista de elementos hijos o que descienden de él si los tiene, separados por comas; o el tipo de contenido, normalmente #PCDATA, que indica datos de tipo texto, que son los más habituales.
- si es un elemento vacío, se indica con la palabra EMPTY.

Ejemplos de cada caso se pueden ver en el DTD de muestra.

A la hora de indicar los elementos descendientes (los que están entre paréntesis) vemos que van seguidos de unos caracteres especiales: '+', '*', '?' y '|'. Sirven para indicar qué tipo de uso se permite hacer de esos elementos dentro del documento:

- + : uso obligatorio y múltiple; permite uno o más elementos de ese tipo dentro del elemento padre, pero como mínimo uno.
- * : opcional y múltiple; puede no haber ninguna ocurrencia, una o varias.
- ? : opcional pero singular; puede no haber ninguno o como mucho uno.
- | : equivale a un OR, es decir, da la opción de usar un elemento de

entre los que forman la expresión, y solo uno.

De este modo, si por ejemplo encontramos en un DTD la siguiente declaración:

```
<!ELEMENT ficha (nombre+, apellido+, direccion*, foto?,
telefono*|fax*)>
```

sabremos del elemento ficha que puede contener los siguientes elementos: un nombre y un apellido como mínimo, pero puede tener más de uno de cada; opcionalmente puede incluirse una o varias direcciones, pero no es obligatorio; opcionalmente también se puede incluir una única foto; y por fin, pueden incluirse, aunque no es obligatorio en ninguno de los dos casos, uno o más teléfonos o uno o más números de fax.

Como ya se comentó un documento XML presenta una jerarquía muy determinada, definida en el DTD si es un documento válido, pero siempre inherente al documento en cualquier caso (siempre se puede inferir esa estructura a partir del documento sin necesidad de tener un DTD en el que basarse), con lo que se puede representar como un árbol de elementos. Existe un elemento raíz, que siempre debe ser único (sea nuestro documento válido o sólo bien formado) y que se llamará como el nombre que se ponga en la definición del <!DOCTYPE si está asociado a un DTD o cualquiera que se desee en caso contrario. Y de él descienden las ramas de sus respectivos elementos descendientes o hijos. De este modo, la representación en forma de árbol de nuestro documento XML de ejemplo sería:

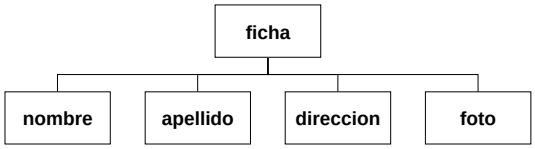


Figura 5.1 - Esquema de ejemplo de ficha XML

Vemos que es un documento muy sencillo, con una profundidad de 2 niveles nada más: el elemento raíz ficha, y sus hijos nombre, apellido, direccion, foto. Es obvio que cuanta más profundidad, mayor tiempo se tarda en procesar el árbol, pero la dificultad siempre será la misma gracias a que se usan como en todas las estructuras de árbol algoritmos recursivos para tratar los elementos.

El DTD, por ser precisamente la definición de esa jerarquía, describe precisamente la forma de ese árbol. La diferencia (y la clave) está en que el DTD define la forma del árbol de elementos, y un documento XML válido puede basarse en ella para estructurarse, aunque no tienen que tener en él todos los elementos, si el DTD no te obliga a ello. Un documento XML bien formado sólo tendrá que tener una estructura jerarquizada, pero sin tener que ajustarse a ningún DTD concreto.

Para la definición de los atributos, se usa la declaración <!ATTLIST, seguida de:

- El nombre de elemento del que estamos declarando los atributos;
- El nombre del atributo;
- Los posibles valores del atributo, entre paréntesis y separados por el carácter |, que al igual que para los elementos, significa que el atributo puede tener uno y sólo uno de los valores incluidos entre paréntesis. O bien, si no hay valores de finidos, se escribe CDATA para indicar que puede ser cualquier valor (alfanumérico, vamos). También podemos indicar con la declaración ID que el valor alfanumérico que se le de, será único en el documento, y se podrá referenciar ese elemento a través de ese atributo y valor;
- De forma opcional y entrecomillado, un valor por defecto del atributo si no se incluye otro en la declaración;
- Por último, si es obligatorio cada vez que se usa el elemento en cuestión declarar este atributo, es necesario declararlo con la clausula #REQUIRED; si no lo es, se debe poner #IMPLIED, o bien #FIXED si el valor de dicho atributo se debe mantener fijo a lo largo de todo el documento para todos los elementos del mismo tipo (notar que no es lo mismo esto a lo que significaba ID).

5.5 Las Entidades o Entities

Mediante estos elementos especiales es posible dotar de modularidad a nuestros documentos XML. Se pueden definir, del mismo modo que los propios DTDs de los que ya hemos hablado, dentro del mismo documento XML o en DTDs externos.

Las primeras entidades que hemos conocido son los carácteres especiales &, ", ', < y >, que vimos que debíamos escribir mediante las declaraciones: &amp, ", ', < y >. Es decir, que cuando queramos refe renciar alguna entidad definida dentro de nuestro documento o en otro documento externo, deberemos usar la sintáxis: &nombre;.

Pero por supuesto las entidades no solo sirven para incluir caracteres especiales no ASCII. También las podemos usar para incluir cualquier documento u objeto externo a nuestro propio documento.

Por ejemplo, y como uso más simple, podemos crear en un DTD o en el documento XML una entidad que referencie un nombre largo:

```
<!ENTITY DAT "Delegación de Alumnos de Teleco">
```

Y de este modo, cada vez que queramos que en nuestro documento aparezca el nombre "Delegación de Alumnos de Teleco", bastará con escribir &DAT;.

El texto referenciado mediante la entidad puede ser de cualquier tamaño y contenido. Si queremos incluir una porción de otro documento muy largo que hemos guradado aparte, de tipo XML o cualquier otro, podemos hacerlo por lo tanto de este modo:

```
<!ENTITY midoc SYSTEM  "http://www.abc.com/midoc.xml">
```

Del mismo modo que usábamos con los DTDs externos, indicamos al procesador con SYSTEM que la referencia es externa y que lo que sigue es una URI estándar, y es lo que queda entrecomillado a continuación.

Hay que resaltar que esto se aplica dentro de documentos XML, pero no de DTDs. Para incluir entidades en DTDs debemos usar el carácter %:

```
<!ENTITY % uno "(uno | dos | tres)">
```

Y para expandirlo en un DTD tendremos que escribir:

```
<!ELEMENT numero (%uno;) #IMPLIED>
```

Capítulo 6

Infraestructura del sistema

6.1 Descripción general

Como se ha visto en la introducción, el planteamiento llevado a cabo se hacen ciertas suposiciones de capacidades hardware de cada estación base (Bluetooth, Wi-Fi, etc...), además de dejar entrever la idoneidad de un tamaño reducido del hardware dedicado a alojar los procesos mencionados.

Por lo tanto, en el desarrollo del proyecto se debía escoger una plataforma de implementación que cumpliese con los requisitos de interoperabilidad con el hardware con el que se iba a trabajar, mayormente dispositivos Bluetooth con conexión USB y tarjetas Wi-Fi 802.11, además de ofrecer un buen ejemplo de integración, en cuanto a tamaño se refiere, y de capacidades de portabilidad de los programas.

En estos términos, podemos encontrar un buen número de soluciones en el mercado, y como ejemplo de todas las características buscadas citadas anteriormente, se escogió la solución diseñada por la empresa Compulab, empresa especializada en soluciones computacionales de tamaño reducido.

El producto escogido es el modelo SB-X255, que viene provisto, entre otras cualidades, de un procesador Intel XScale 255 a 400 Mhz, 64 Mb de memoria RAM, 128 Mb de memora no volátil, puertos USB, ranuras PCMCIA, etc... Adicionalmente, la empresa provee gratuitamente una distribución de Linux apta para sus productos, lo que complementa de una manera ideal el entorno de desarrollo.

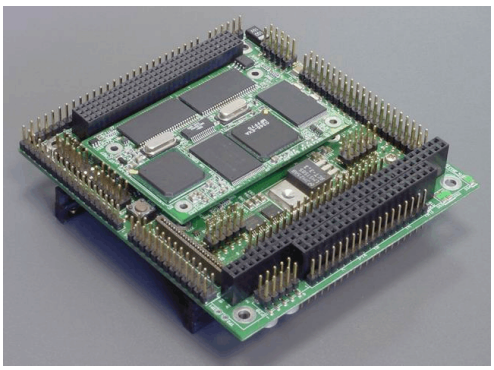


Figura 6.1 - SB-X255

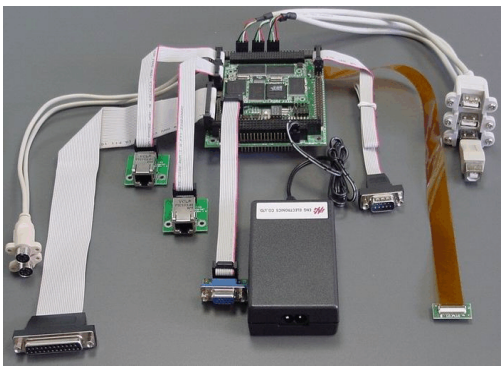


Figura 6.2 - SB-X255 con todos sus conectores

Al trabajar con Linux, nos aseguramos un soporte de la comunidad que trabaja en el desarrollo de este sistema operativo y con ello gran cantidad de documentación y soporte. Por otro lado, la elección de este hardware ha obligado ha tomar decisiones sobre otros temas. Por ejemplo, nos llevo a descartar el lenguaje de programación Java, por no existir una maquina virtual disponible para este sistema.

Sobre esta plataforma, y con vistas a ofrecer un ejemplo de implementación atractivo, no solo por tamaño, si no por coste de desarrollo y adaptación, se ha efectuado las pruebas con los programas desarrollados.

6.2 Características del sistema SB-X255

El modelo SB-X255 es una computadora estándar bajo las especificaciones PC/104+ y que se presenta en una sola placa. Usa un modulo CM-X255 para implementar la mayoría de las funciones provistas, además de ofrecer otras funcionalidades importantes a través de la placa en la que va montada. Ademas, todas las capacidades provistas por el SB-X255 son personalizables a demanda de los objetivos de precio o rendimiento exigidos por el usuario.

El SB-X255 incluye una serie de conectores de expansión, abriendole las puertas a un inmenso rango de periféricos estándar. Además, el SB-X255 incluye un controlador PCMCIA y uno o dos slots. Se le puede insertar y asegurar una tarjeta PCMCIA sin medios mecánicos adicionales.

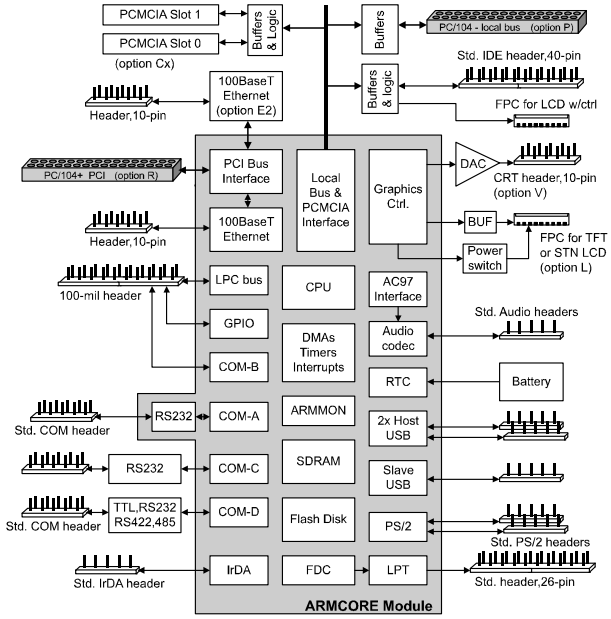


Figura 6.3 - Diagrama del SB-X255

- Computador PC/104+ en placa única con modulo CM-X255
- Procesador Intel XScale PXA255, hasta 400 Mhz
- 16-64 Mb SDRAM
- 1-128 Mb Flash Disk
- PCI y bus local de expansión en formato PC/104+
- Puertos COM 1 a 4 con opción de controlador RS232 / RS485 / RS422 / TTL
- Interfaces IrDA, LPT, GPIO y PS/2 para ratón y teclado
- Puertos USB maestros y esclavos
- Interfaces para discos duros y disqueteras estándar
- Tarjeta de sonido (opcional)
- Interface para pantalla táctil (opcional)
- Controlador de gráficos VGA
- Conectores para VGA y LCD (opcional)
- Uno o dos puertos Ethernet 10/100BaseT
- Uno o dos slots PCMCIA
- Reloj interno con batería de litio (opcional)
- Tamaño reducido: 96 mm x 91 mm

6.3 Puesta en marcha del sistema

En primer lugar deberemos conectar el cable de alimentación a la placa, teniendo cuidado de no equivocarse de emplazamiento, lo que podría ser

perjudicial, ya que podría dañar la placa. Para mostrar este paso lo mas claro posible, se incluye la siguiente figura, en la que podemos ver la posición exacta del conector.

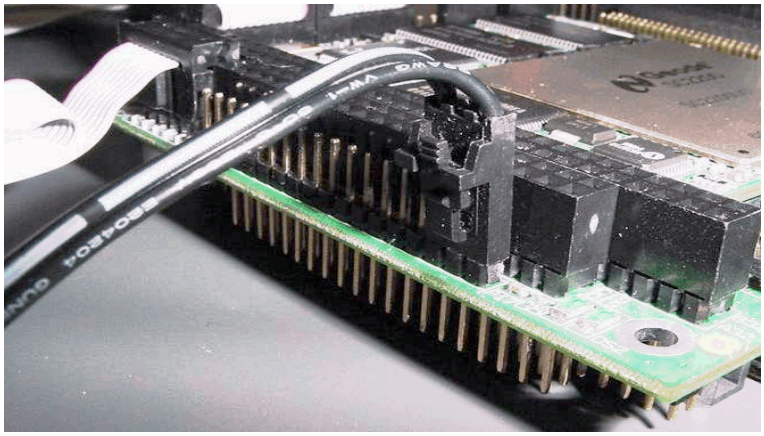


Figura 6.4 - Detalle de conexión de alimentación

El siguiente paso indispensable sera conectar uno de los conectores de puerto serie al conector COM1, que en la placa esta etiquetado como P20 y también como COM1.

A través de este puerto haremos las primeras configuraciones. En la siguiente imagen, se detallan estas conexiones, así como las conexiones de un puerto Ethernet y un conector USB, que harán falta más adelante.

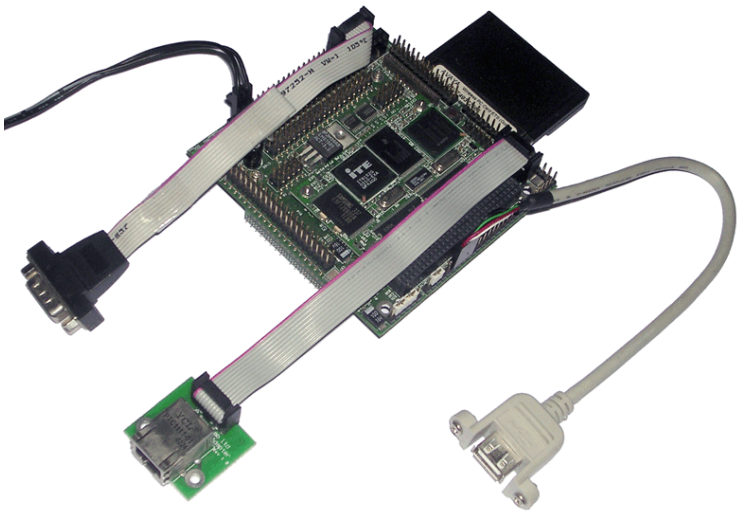


Figura 6.5 - Detalle de conectores usados

Para comunicarnos con el SB-X255 a través del puerto serie, deberemos usar el cable provisto para conectarlo al ordenador de desarrollo. Utilizaremos una aplicación de emulación de terminal a nuestro gusto, HyperTerminal en Windows, Minicom o Xterm en Linux, por ejemplo

Configuraremos el terminal con las siguientes Características:

- 38400 bps
- 8 bit por trama
- 1 bit the parada
- Sin paridad
- Sin control de flujo

Con estos parámetros configurados, podemos encender el SB-X255. Si es la primera vez que vamos a trabajar con el SB-X255, puesto que no tendrá nada cargado en la memoria, nos aparecerá directamente la utilidad ARMonitor, equivalente a una BIOS en un ordenador normal, pero controlada por comandos. Si no es la primera vez que se trabaja con el, o sabemos que tiene algo cargado en memoria, entonces deberemos pulsar la combinación CTRL+C para entrar en la utilidad ARMonitor, que se muestra en la siguiente figura:

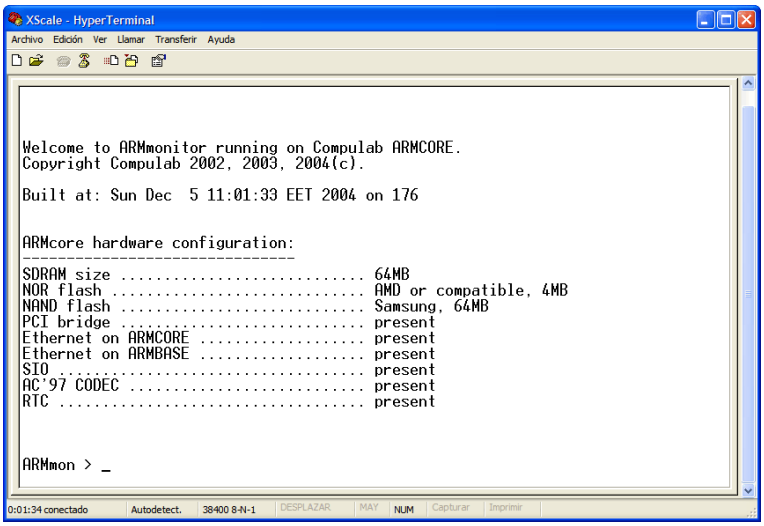


Figura 6.6 - ARMonitor desde HyperTerminal

6.4 Instalación del kernel y la imagen del sistema de ficheros

Ahora vamos a proceder a explicar como instalar estos dos componentes, el kernel y el sistema de ficheros de la distribución Linux. Para realizar esta tarea nos encontramos con dos posibilidades:

- Instalación por puerto de serie
  Este método es el más sencillo, pero también el más lento. Para instalar el kernel, que es un archivo de entorno a 1 Mb es apropiado, pero para descargar la imagen del sistema de ficheros, que es en torno a 64 Mb es excesivamente lento.
- Instalación por Ethernet
  Este método es algo más complicado, ya que requiere configurar la interfaz Ethernet desde ARMonitor y poner a punto un servidor TFTP, pero por otro lado, es el más rápido, y si vamos a hacer muchas pruebas descargando diversas versiones del kernel o actualizamos a menudo el sistema de ficheros, es lo mas recomendable.

6.4.1 Instalación por Ethernet

En primer lugar deberemos asegurarnos de que tenemos un servidor TFTP corriendo en un ordenador, y que en el directorio raíz de este servidor, están los archivos con el kernel y la imagen del sistema de ficheros. A partir de ahora se referirá al archivo del kernel por zImage, y al archivo imagen por disk.dat.

En cuanto al SB-X255, deberemos tener conectado el interfaz ETH0 a la red. Este conector viene etiquetado en la placa como ETH0, por lo que no será un problema identificarlo.

Recuerde que si ha instalado anteriormente un kernel, es posible que tenga que pulsar CTRL+C para acceder a la utilidad ARMonitor.

Una vez dentro de esta deberemos configurar la interfaz ETH0, esto lo haremos mediante los siguientes comandos:

```
> setip eth0
> setip ip 192.168.0.100
> setip mask 255.255.255.0
```

Tome los valores anteriores como mero ejemplo. Con el interfaz Ethernet correctamente configurado, procederemos a la descarga del kernel:

```
> download kernel tftp zImage 192.168.0.1
```

Substituya la dirección IP para coincidir con el equipo que dispone del servidor TFTP con los archivos. A continuación se procederá a programar la

memoria flash NOR, que es donde se aloja el kernel:

```
> flash kernel
```

Tenga precaución de no interrumpir el proceso, o se verá obligado a repetirlo. Ahora resta pasar el sistema de ficheros, que se hará en la memoria flash NAND. Antes de nada, formatearemos esta memoria:

```
> nand flash format
```

Se le pedirá confirmación. Con esta operación se destruye toda la información que pudiese haber en la memoria NAND, por lo que asegurese de que no cometer un error. Como siguiente paso, programamos la memoria flash NAND, con el sistema de ficheros:

```
> nand write disk.dat 192.168.0.1
```

Con las dos memorias programadas, vamos a establecer como arranque predeterminado el kernel recién descargado, y salvaremos la configuración:

```
> setboot os
> save
```

Se pedirá confirmación para salvar la configuración actual. Ahora podemos arrancar el sistema operativo recién descargado con el comando:

```
> bootos
```

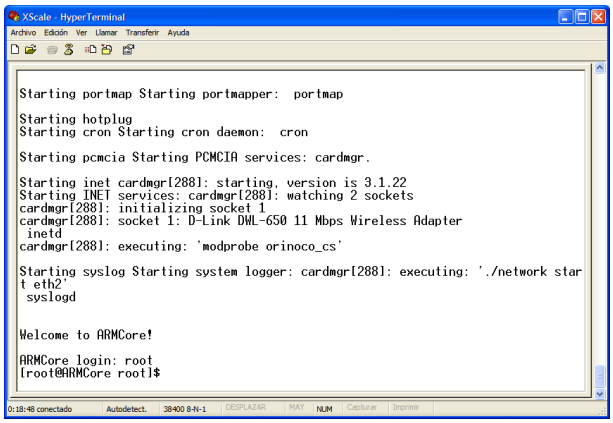


Figura 6.7 - SB-X255 con Linux una vez arrancado

6.4.2 Instalación por puerto serie

En esta modalidad de instalación será necesario el uso de una utilidad para poder controlar el SB-X255 desde el puerto de serie, cuando esta en modo "Manufacturating Link". Esta utilidad puede ser descargada desde la pagina de Compulab.

Arrancaremos el SB-X255 con el terminal configurado como en el caso anterior. Formatearemos la memoria NAND flash:

```
> nand flash format
```

Seguidamente iniciaremos el modo "Manufacturating Link":

```
> mlink start
```

Una vez hemos ejecutado este comando, cerraremos el programa terminal. Por comodidad se recomiendo colocar todos los ficheros relacionados, el kernel, el archivo de imagen y el programa descargado desde la pagina de Compulab (host-arm.exe, nand.bat y kernel.bat).

Para programar el kernel, ejecutaremos:

```
> kernel.bat COMx
```

Donde x indica el puerto COM que se usa. Este proceso dura unos 4 minutos. Para programar la distribución (el archivo de imagen), ejecutaremos:

```
> nand.bat COMx
```

Donde nuevamente, la x indica el puerto usado. Este proceso dura unas 4 horas.Una vez programado, resetearemos el SB-X255, y volveremos a entrar en ARMonitor, desde el emulador de terminal, para finalmente ejecutar los siguientes comandos:

```
> setboot os
> save
> bootos
```

Con este ultimo comando arrancaremos el sistema recien programado.

6.5 Trabajando con el sistema de archivos en modo local

El archivo disk.dat, que contiene una imagen del sistema de ficheros para el SB-X255, puede ser montado en el PC de desarrollo para poder acceder directamente a los ficheros. Esto lo conseguimos con el siguiente comando:

```
# mount disk.dat /mnt/image -o loop
```

De esta manera, podemos acceder a los ficheros, e instalar los programas directamente en el archivo de imagen (en el siguiente capitulo, se hace mención al argumento --prefix, que bien puede utilizarse para establecer el path the instalación al punto de montaje).

Adicionalmente, trabajar de este modo, nos puede ahorrar muchos quebraderos de cabeza, al tener disponibles en el punto de montaje muchas de las librerías nativas y cabeceras que hacen falta a la hora de compilar. Recuerde que para desmontar, únicamente bastará con ejecutar el comando:

```
# umount /mnt/image
```

6.6 Detalles adicionales

La configuración por defecto que ofrece la distribución de Linux, tiene establecido el usuario root sin contraseña, para facilitar la etapa de desarrollo, tanto desde login local, como desde login remoto mediante telnet.

Usuario	root
Contraseña	

Además, existe un usuario para el servidor FTP con las siguientes señas:

Usuario	ftp
Contraseña	11111

6.7 Obtención de los ficheros necesarios.

Desde la página de Compulab, en la sección de desarrollador (Developer), se puede tener acceso a los archivos mencionados arriba, el kernel

Capitulo 6 - Infraestructura del sistema y el archivo imagen del sistema de ficheros.

Capítulo 7

Herramientas usadas en el desarrollo

Para el desarrollo de la aplicación servidor y procesador XML, se ha hecho uso de las siguientes herramientas:

- Librerías BlueZ
- Librería XML2

Por otro lado, otras herramientas usadas, han sido fruto de las elecciones tomadas sobre la plataforma sobre la que se iban a hacer las pruebas, y por lo tanto, no son estrictamente necesarias en todos los casos, y solo dependen de la elección de plataforma que se haga.

7.1 Librerías BlueZ

BlueZ es el proyecto oficial, continuamente en desarrollo, para la implementación de las especificaciones del estándar inalámbrico Bluetooth para sistemas operativos Linux. BlueZ proporciona acceso a las capas y protocolos establecidos en Bluetooth.

Esta implementación no solo proporciona un driver al kernel para poder hacer uso de dispositivos provistos de tecnología Bluetooth, si no que también proporciona drivers para las distintas formas de conectar un dispositivo Bluetooth al ordenador, ya sea por USB, PCMCIA o puerto serie; además de una serie de utilidades para la configuración de los dispositivos, configuración de los servicios propios de la tecnología Bluetooth y otras utilidades.

Por otro lado, BlueZ, agrega una nueva implementación de sockets, con dominio Bluetooth (AF_BLUETOOTH), para que el programador pueda tener acceso a estos dispositivos como un dispositivo de comunicaciones más, manteniendo la estructura de sockets habitual en C para otros dominios como AF_INET o AF_UNIX.

7.1.1 Descripción de componentes BlueZ

En un principio, dentro del proyecto BlueZ, uno de los paquetes esenciales para su uso era un parche para el kernel de Linux, que añadía el la capa de control de dispositivos Bluetooth y el soporte para los protocolos asociados a esta. Afortunadamente, a partir de la versión del kernel 2.4.18 (aunque con algunas limitaciones, que se solventan a partir de las 2.4.26), este soporte, estos módulos, ya viene incluido en el propio código del kernel. Por lo tanto si nuestro kernel es superior al 2.4.18 o de la serie 2.6, podemos olvidarnos de este paquete.

Con el soporte en el kernel tendremos gran parte del trabajo hecho para poder empezar a usar dispositivos Bluetooth en sistemas Linux. Únicamente resta asegurarse de la existencia de los siguientes paquetes en el sistema:

```
bluez-libs-x.xx.tar.gz
bluez-utils-x.xx.tar.gz
```

Donde x.xx indica un numero de versión, que no indico aquí ya que se suele actualizar muy a menudo. En una distribución moderna, estos paquetes suelen venir incluidos, pero en cualquier caso, puede obtenerse la ultima versión en la pagina oficial de BlueZ.

Cada uno de estos paquetes nos va a proporcionar las siguientes utilidades:

- bluez-libs-x.xx.tar.gz
  En este paquete nos encontraremos con una serie de cabeceras y de librerías para poder hacer accesibles desde código C las distintas capacidades BlueZ.
- bluez-utils-x.xx.tar.gz
  En este paquete nos vamos a encontrar con una serie de utilidades para poder configurar todo tipo de aspectos relacionados con nuestro dispositivo Bluetooth.

7.1.2 Compilación en instalación

Una vez descargados los paquetes, procederemos a descomprimirlos:

```
# tar -xvzf bluez-libs-x.xx.tar.gz
# tar -xvzf bluez-utils-x.xx.tar.gz
```

Esto nos creara dos directorios llamados:

```
bluez-libs-x.xx
bluez-utils-x.xx
```

En primer lugar procederemos a compilar el paquete de librerías, que son necesarias para poder compilar el otro paquete. Esto lo haremos con los siguientes comandos, dentro del directorio recién creado:

```
bluez-libs-x.xx# ./configure
bluez-libs-x.xx# make
bluez-libs-x.xx# make install
```

Con estos comandos, suele ser suficiente en casi cualquier equipo para obtener una compilación exitosa. Si se encuentra algún problema consulte el archivo de ayuda INSTALL y README que encontrara en el mismo directorio, donde viene información básica sobre los requisitos del paquete. Si esta intentando compilar este paquete con un compilador cruzado, se encontrara con ciertos problemas, ya que no basta con estos comandos. Sobre compilación cruzada se habla en el siguiente apartado.

Con el paquete de utilidades procederemos de una manera similar, dentro del directorio donde lo hemos descomprimido introduciremos los siguientes comandos:

```
bluez-utils-x.xx# ./configure
bluez-utils-x.xx# make
bluez-utils-x.xx# make install
```

Nuevamente, si encuentra algún problema, refierase al archivo INSTALL o README, o bien al siguiente apartado para compilación cruzada.

7.1.3 Compilación cruzada

El proceso a aseguir es básicamente el mismo, excepto que ahora hemos de darle información adicional acerca del proceso de compilación cruzada. Generalmente, en los archivos README e INSTALL podremos encontrar información especifica para el caso en el que estemos.

Para el caso que nos ocupa, estableceremos en primer lugar una serie de variables de entorno para definir que programas se usaran como compilador,

linkador, ensamblador, etc...

```
export AR="/usr/local/arm/2.95.3/bin/arm-linux-ar"
export CC="/usr/local/arm/2.95.3/bin/arm-linux-gcc"
export AS="/usr/local/arm/2.95.3/bin/arm-linux-as"
export LD="/usr/local/arm/2.95.3/bin/arm-linux-ld"
```

Las rutas indicadas son solo a modo de ejemplo y deberán reflejar la situación real de los ejecutables a usar.

Una vez hecho esto deberemos, comenzaremos como antes por el paquete de librerías (recordemos que es necesario para el paquete de utilidades). Así que desde el directorio donde se encuentra el código de las librerías BlueZ ejecutaremos:

```
bluez-libs-x.xx# ./configure --host=arm-linux --
prefix=/mnt/image
```

Notese que ahora le estamos pasando información adicional, como es la maquina en la que se va a compilar, con el argumento host, (en este caso un equipo Linux con procesador x86), y ademas el comando prefix. Este ultimo sirve simplemente para indicar donde se dejaran los archivos compilador cuando hagamos make install, por lo que es prescindible si vamos a instalar los archivos manualmente. Para terminar ejecutaremos:

```
bluez-libs-x.xx# make
bluez-libs-x.xx# make install
```

Tenga presente, que si esta haciendo una compilación cruzada, y ejecuta make install, sin haber establecido el argumento --prefix en la etapa de configuración a algún directorio, se tomara como destino de instalación por defecto la raíz del sistema, por lo que se instalaran archivos de otra arquitectura en sus directorios habituales, sustituyendo versiones que ya tuviese. Por lo tanto es mas que recomendable, si no obligatorio, que no ejecute make install, si no ha establecido el argumento --prefix en la etapa de configuración.

Una vez listo, pasamos a compilar el paquete de utilidades, cuyo procedimiento es básicamente el mismo:

```
bluez-utils-x.xx# ./configure --host=arm-linux --
prefix=/mnt/image --with-bluez=/mnt/image
```

Aquí se le pasa un nuevo argumento, with-bluez, que indica donde están las librerías BlueZ para esa arquitectura. Si anteriormente, a la hora de configurar el paquete de librerías, indicamos como lugar de instalación /mnt/image, ahora le indicaremos que es ahí donde debe buscar las librerías y cabeceras. Únicamente restaría:

```
bluez-utils-x.xx# make
bluez-utils-x.xx# make install
```

Vuelvo a recordar, que es desaconsejable ejecutar make install, si no se ha establecido el argumento --prefix en la etapa de configuración.

7.1.4 Problemas con la compilación cruzada

En el apartado anterior se ha explicado el procedimiento a seguir para su compilación e instalación, pero se ha supuesto que no se iba a presentar ningún problema. A continuación, se enumeran una serie de problemas comunes y la manera de solucionarlos.

- No se encuentran las librerías o las cabeceras
  Este problema suele venir a consecuencia de un uso inapropiado del argumento --with-bluez. Este argumento debe indicar un directorio con dos subdirectorios, lib e include, en los que se encontraran librerías y cabeceras respectivamente. Tras ejecutar el comando make, las librerías están en un subdirectorio oculto dentro del subdirectorio src, mientras que las cabeceras están en el directorio include. Se recomienda encarecidamente indicar un lugar de instalación con --prefix.

- No se ha definido PATH_MAX
  Parece ser que existe un error, y es necesario incluir la cabecera limits.h en un par de archivos. Para corregirlo incluiremos la siguiente linea en los archivos tools/hciconfig.c y hidd/sdp.c:

  ```
  #include <linux/limits.h>
  ```

- No se ha definido __ctype_b_loc
  Este error se debe a ciertos problemas con la librería libc. Se recomienda que para compilar las utilidades, y solventar este problema, haya instalado las librerías en la imagen del sistema de archivos montada localmente, como se explico en el Capitulo 6.

7.2 Librería XML2

La librería XML2 es un proyecto desarrollado, en principio, para el proyecto Gnome para Linux, pero que puede ser usado independientemente de este. XML es un lenguaje de marcas para describir otros lenguajes de marcas (como por ejemplo HTML), y la librería XML2 nos proporciona una serie de utilidades a la hora de trabajar con archivos descritos en XML.

Entre estas utilidades, nos encontramos con la posibilidad de leer archivos en XML de manera que la información se desarrolla como un árbol en memoria, de manera que podremos acceder a los campos que nos interesan, o también nos encontramos ante la posibilidad de conocer la coerción de un archivo a través de su DTD.

7.2.1 Compilación e instalación

En primer lugar nos hará falta bajar, si es que no disponemos de las librerías XML2 en nuestro ordenador, el paquete. Esto lo podemos hacer desde la pagina de oficial XMLsoft, en la sección de descargas. De ahí podremos bajar el código fuente y también versiones compiladas para ciertas distribuciones. Aquí se explicará como compilar dichas fuentes, que es el método para obtener las librerías mas genérico posible.

Una vez obtenida la descarga procederemos a descomprimir el archivo:

```
# tar -xvzf libxml2-sources-x.x.xx.tar.gz
```

Con esto obtendremos un directorio llamado libxml2-x.x.xx. Entraremos en el directorio y ejecutaremos los pasos típicos.

```
libxml2-x.x.xx# ./configure
libxml2-x.x.xx# make
libxml2-x.x.xx# make install
```

En principio esto debería funcionar a la primera. Si encuentra algún error recurra en primer lugar a los archivos README e INSTALL, que le proporcionarán ayuda específica sobre problemas y requisitos.

7.2.2 Compilación cruzada

Al igual que con las librerías BlueZ, los pasos a seguir son los mismos, excepto por algunos pequeños detalles. Acuerdese de establecer las variables de entorno para los compiladores, linkadores, etc...

```
export AR="/usr/local/arm/2.95.3/bin/arm-linux-ar"
export CC="/usr/local/arm/2.95.3/bin/arm-linux-gcc"
export AS="/usr/local/arm/2.95.3/bin/arm-linux-as"
export LD="/usr/local/arm/2.95.3/bin/arm-linux-ld"
```

Puede guardar estos comandos en un archivo, y usarlo como script, pero tenga en cuenta que para ejecutarlo deberá usarlo de esta manera:

```
#. setcrossgcc
```

En vez de:

```
#./setcrossgcc
```

La diferencia estriba que en el segundo caso, para ejecutar el script se ejecuta otro interprete de comandos, con lo que las variables de entorno que se quieren establecer, lo hacen en el segundo interprete, no en el actual, que es lo que queremos y que conseguimos usando un espacio en vez de /. Recuerde que para ejecutar el script deberá darle permisos de ejecución.

```
#chmod +x setcrossgcc
```

Una vez establecidas las variables de entorno podemos pasar a configurar y compilar las fuentes.

```
libxml2-x.x.xx# ./configure --host=arm-linux --
prefix=/mnt/image
```

Recuerde, el argumento --prefix no es estrictamente necesario. A continuación solo nos queda ejecutar:

```
libxml2-x.x.xx# make
libxml2-x.x.xx# make install
```

Insisto en recordar en que no se ejecute make install, si no se ha establecido el argumento --prefix en la etapa de configuración. Recuerde que esto podría causarle algunos problemas con el equipo que este trabajando.

7.2.4 Problemas con la compilación cruzada

- No se ha definido PATH_MAX
  Parece ser que existe un error, y es necesario incluir la cabecera limits.h en un par de archivos. Para corregirlo substituiremos la linea:

```
#include <limits.h>
```

  por la linea:

```
#include <linux/limits.h>
```

  en el archivo testModule.c

7.3 Ejemplo de potabilidad sobre sistema XScale

En la parte practica del proyecto, la implementación del proceso servidor y del procesador XML se ha hecho sobre un sistema con procesador XScale, como ejemplo de la gran potabilidad y de las distintas opciones de implementación del código.

En virtud al sistema operativo escogido, Linux, cualquier sistema que soporte este sistema, independientemente de su arquitectura interna, y con poca o ninguna modificación del código original, podría ejecutar ambos programas.

7.4 Otras herramientas para la implementación sobre sistema XScale

Al optar por un sistema basado en un procesador de la gama XScale de Intel, ha sido necesario el uso de diversas herramientas para poder compilar el código y obtener un ejecutable nativo para esta arquitectura.

7.4.1 Compilador cruzado GCC para procesadores ARM/XScale

Un compilador cruzados tiene el mismo fin que cualquier otro compilador de código fuente, y es generar código nativo a una arquitectura. Habitualmente, un compilador que genera código para una arquitectura determinada, también se ejecuta en esa misma arquitectura. En cambio, en un compilador cruzado esto no es así, ya que la arquitectura para la que se compila no es la misma que la arquitectura en la que se ejecuta el compilador.

La existencia de compiladores cruzados esta marcada por la imposibilidad o dificultad de contar de un compilador nativo para la arquitectura de destino, debido a que el entorno de desarrollo que podríamos desplegar entorno a esta arquitectura sería bastante limitado. Por esto razón se opta por usar como entorno de desarrollo uno que nos proporcione una serie de facilidades.

Poniendo como ejemplo nuestro caso, usamos como entorno de desarrollo un PC normal, es decir arquitectura x86, mientras que estábamos generando código para un procesador XScale.

Por regla general, podremos obtener un compilador cruzado del fabricante del sistema para el que estamos programando. En nuestro caso, Compulab proporciona en su pagina, en el apartado de Desarrollo (Developer) para CM-X255 y SB-X255 dos versiones de gcc para arm, la 2.95.3 y la 3.3.2.

7.4.1.1 Instalación

La instalación de un compilador cruzado es sencilla, solo hace falta descomprimirla en un directorio de nuestro agrado, aunque también se recomienda descomprimirla en un lugar típico, como es en /usr/local.

7.4.2 Wireless Tools para Linux

Este conjunto de herramientas está destinada a la configuración y mantenimiento de dispositivos 802.11 conectados a un computador, como por ejemplo, establecer el modo de operación, velocidad, etc...

En la distribución de Linux que Compulab suministra no incluyen estas herramientas, por lo que se procedió a descargar el código fuente, y compilarlo, mediante el compilador cruzado, para su uso en el sistema sobre el que se ha desarrollado el proyecto.

Como se ha dicho, su función es administrar dispositivos 802.11, por lo que si no se implementa esta parte, no son estrictamente necesarias. Por otro lado, en las distribuciones Linux para arquitecturas x86 suelen venir incluidas por defecto.

Para obtener las fuentes de esta utilidades, las descargaremos de su pagina, en la sección de descargas.

7.4.2.1 Compilación e instalación

Simplemente bastara con ejecutar los siguientes comandos. Estas fuentes no necesitan configuración, o mejor dicho, su configuración se hace manualmente.

```
wireless_tools.xx# make
wireless_tools.xx# make install
```

7.4.2.2 Compilación cruzada

El método es exactamente el mismo que antes, simplemente que ahora tendremos que editar nosotros el archivo Makefile, ya que no disponemos de un script para configurarlo automáticamente.

Abriremos el archivo Makefile, y buscaremos las siguientes lineas:

```
PREFIX = /usr/local

CC = gcc

AR = ar
```

Y las substituiremos por estas:

```
PREFIX = /mnt/image

CC = /usr/local/arm/2.95.3/bin/arm-linux-gcc

AR = /usr/local/arm/2.95.3/bin/arm-linux-ar
```

Recuerde que estos valores son simplemente de ejemplo, y deberá editarlos en función de donde se encuentren esos programas en su ordenador.

Además, la compilación necesita del archivo version.h, que suele encontrarse en las cabeceras del kernel. Puede indicar la localización de este archivo, añadiendo su ruta, excepto el ultimo nivel de directorio, es decir, si el archivo se encuentra en /usr/linux-2.4.26/include/linux. Únicamente indicaremos /usr/linux-2.4.26/include, puesto que las fuentes ya hacen referencia al directorio linux. De este modo nos quedaría en el archivo Makefile:

```
CFLAGS=-Os -W -Wall -Wstrict-prototypes \
-Wmissing-prototypes -Wshadow -Wpointer-arith \
-Wcast-qual -Winline -I. -I/usr/linux-2.4.26/include
```

Alternativamente, podemos crear un directorio en el directorio donde tenemos las fuentes, llamado linux, y allí dentro copiar el archivo version.h, en cuyo caso, no necesitaríamos variar la linea CFLAGS.

Una vez hecho esto, basta ejecutar en la linea de comando:

```
wireless_tools.xx# make
wireless_tools.xx# make install
```

Capitulo 7 - Herramientas usadas en el desarrollo

Capítulo 8

Arquitectura e implementación

A continuación se va a dar una descripción de las tareas llevadas a cabo por los procesos que se han programado.

8.1 Arquitectura general del sistema

La implementación real que subyace bajo este proceso incluye en realidad dos procesos esenciales, lanzados por un proceso inicial, que denominaremos el "Lanzador de Servidores". La misión de este, como indica su nombre, es la de crear un hilo de ejecución por cada uno de los servidores que se vayan a lanzar. El motivo de esto se debe a que los sockets Bluetooth y TCP son de diferente familia, y no pueden compartirse, por lo que es necesario declarar un socket diferente para escuchar en distintos dispositivos. De este modo tenemos que el "Lanzador de Servidores", lanza dos hilos, (1a) y (1b), uno para un servidor Bluetooth, y otro para un servidor TCP.

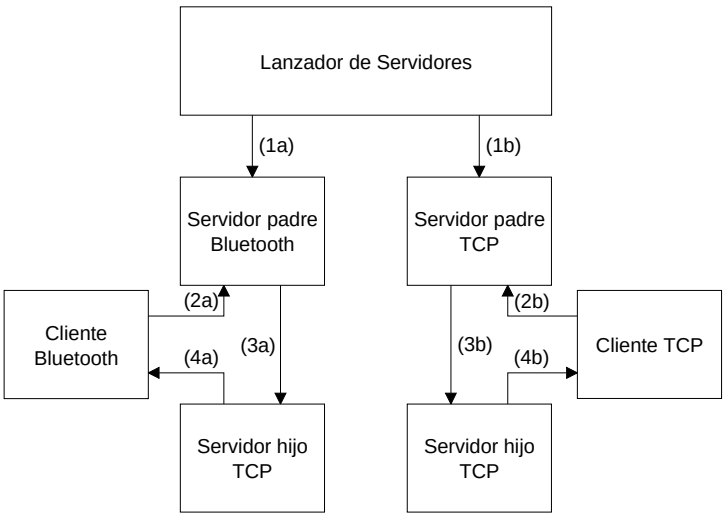


Figura 8.1 - Diagrama básico del proceso servidor

A su vez, cada uno de los servidores lanzados, son servidores que no se van a encargar de ofrecer servicio directamente a los clientes, es decir, se encargan únicamente de aceptar nuevas conexiones, (2a) y (2b). Para servir a los clientes, el proceso padre creara procesos hijos, (3a) y (3b), para tal fin. De esta manera, el proceso padre puede crear varios procesos hijos que pueden servir, (4a) y (4b), a varios clientes (limitados por la capacidad del sistema), aumentando el rendimiento.

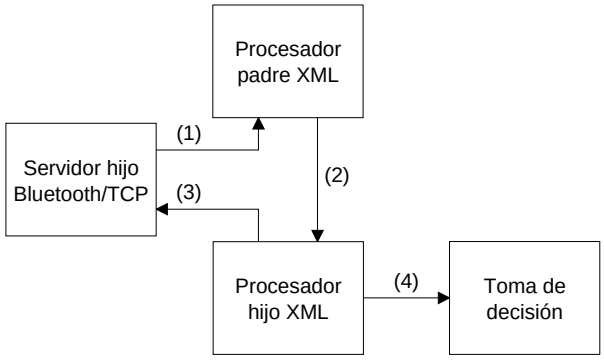


Figura 8.2 - Diagrama básico del procesador XML

Por otro lado, el procesador XML, es un proceso que esta ejecutandose independientemente, en espera de recibir peticiones de servicio. Al igual que los servidores, el procesador XML padre esta en espera de la recepción de una petición de servicio, (1). Cuando esta se produce se crea un proceso hijo, (2), para servir la petición del cliente (3). La información extraída es enviada, (4) al proceso toma de decisión.

8.2 Proceso servidor

Como se ha visto en la descripción general, lo que hasta ahora habíamos llamado como proceso servidor, realmente esta compuesto de varios hilos y procesos padre/hijo. A continuación se va a dar una descripción detallada de los componentes que componen este proceso y el desarrollo que se lleva a cabo cuando se ejecuta.

8.2.1 Lanzador de servidores.

Es el proceso que se encarga de lanzar los distintos servidores, y por lo tanto es el primer proceso en arrancar. Se encarga únicamente de arrancar los servidores y monitorizar su cierre.

8.2.2 Servidores de clientes

Son los procesos encargados de dar servicio a los clientes que intente conectarse. En la implementación practica del proyecto se han desarrollado dos servidores, uno para clientes con dispositivos Bluetooth, y otro para clientes TCP, como por ejemplo los clientes con dispositivos 802.11.

8.2.2.1 Servidor para clientes TCP

Este proceso se encarga de dar servicio únicamente a los clientes que intente conectarse a través de un socket TCP. El proceso servidor padre esta monitorizando la llegada de nuevos clientes, y para cada nuevo cliente que llega, el proceso padre crea un proceso hijo para dar servicio al cliente.

Por otro lado, el proceso hijo, tiene como tarea la recepción de la información que el cliente le manda. Esta información consiste en una ficha, estructurada en XML que contiene detalles sobre la discapacidad del cliente. El cliente primero envía el tamaño de dicha ficha, y luego la ficha en sí misma.

Cuando el proceso hijo ha recibido la ficha, esta es guardada en el una unidad de almacenamiento, para su procesamiento por el procesador XML, y se le comunica a este proceso la ubicación de la ficha, para su procesado. El procesador XML da una respuesta sobre la validez de la ficha, y si no fuese correcta, se le comunicaría al cliente, y se cortaría la comunicación.

En caso de que la ficha sea correcta, la comunicación se mantiene viva, de manera que se puede detectar cuando el cliente se sale de la cobertura.

8.2.2.2 Servidor para clientes Bluetooth

El funcionamiento no difiere de su análogo para TCP, únicamente ha diferencias en el modo de implementarlo.

8.3 Procesador XML

Este proceso es el encargado de extraer la información de la ficha suministrada por el cliente. Como se comento antes, es un proceso totalmente independiente, con vistas a que cualquier tipo de servidor pueda acceder a sus servicios. En la implementación práctica solo se comunica con un servidor TCP y otro Bluetooth, pero podría darse el caso de querer añadir un tercer servidor de otra índole, y mediante este modo, no seria necesario ningún cambio.

El procesador XML esta implementado de manera que se comporta como un servidor habitual, es decir, esta a la espera de conexiones para recibir referencias a ficheros que procesar y de los que extraer información. Como es necesario que hiciese frente a un elevado numero de peticiones desde los servidores hijo, al ser estos creados por cada cliente, el procesador XML también tiene una estructura padre/hijo, para poder servir a mas de un cliente a la vez.

Una vez recibida la referencia al fichero XML a procesar, este se procesa, y se responde al cliente si la ficha es correcta o no con respecto a su DTD. En caso de ser correcta, la información pertinente se entrega al proceso "Toma de decisión".

Capítulo 9

Manual de uso

9.1 Compilación del proceso servidor

Para comodidad de uso, se han configurado todas las opciones posibles de compilación en un fichero Makefile. En este fichero se han definido varios objetivos (targets) de compilación con las posibles combinaciones. Para el proceso servidor existen varios objetivos, a parte de seleccionar la arquitectura de destino. Para el procesador XML solo existe un objetivo posible, a parte de poder seleccionar la arquitectura destino.

Como usted sabrá de sobra, para compilar uno de estos objetivos basta con ejecutar desde la linea de comandos:

```
# make target
```

Los posibles objetivos de compilación para el proceso servidor se muestran en la siguiente tabla, junto con una descripción.

Objetivo	Descripción
tcp_server	Solo se atienden peticiones TCP
bluez_server	Solo se atienden peticiones Bluetooth
bluez_tcp_server	Se atienden peticiones TCP y Bluetooth

9.1.1 Compilación para otras arquitecturas

El archivo Makefile provisto con el código fuente esta configurado para poder hacer uso de un compilador cruzado para obtener un ejecutable apto para la arquitectura que deseemos. Los objetivos para otra arquitectura son los mismos vistos anteriormente.

Para la compilación de los objetivos del proceso servidor para otra arquitectura, simplemente se añadirá la palabra clave arm a la linea de comandos:

```
# make target arm
```

9.2 Compilación del procesador XML

Para el procesador XML, existen dos objetivos, por lo que bastará con ejecutar:

```
# make xmlparser
```

9.2.1 Compilación para otras arquitecturas

Para el procesador XML, añadiremos, igual que para el caso del servidor, la palabra clave arm, para obtener un binario para otra arquitectura:

```
#make xmlparser arm
```

9.3 Configuración de la compilación de otras arquitecturas

Los archivos Makefile provistos con el código fuente están configurados de manera genérica para usar unos ejecutables determinados en ciertas localizaciones. Puesto que esto ha sido fruto de un Configuración particular, puede que esta no coincida con la suya.

Para el correcto funcionamiento de la compilación cruzada, deberá editar las siguientes lineas del fichero Makefile, de manera que refleje su Configuración particular:

Variable	Descripción
CFLAGS	Aquí se le pasaran la localización de librerías y cabeceras de la arquitectura destino
AR	Localización del programa ar
CC	Localización del programa cc (compilador)
AS	Localización del programa as (ensamblador)

LD	Localización del programa ld (linkador)

Los valores a los que se inicializan estas variables en el desarrollo del proyecto son los siguientes:

```
CFLAGS=-I/mnt/image/include -L/mnt/image/lib/
AR=/usr/local/arm/2.95.3/bin/arm-linux-ar
CC=/usr/local/arm/2.95.3/bin/arm-linux-gcc
AS=/usr/local/arm/2.95.3/bin/arm-linux-as
LD=/usr/local/arm/2.95.3/bin/arm-linux-ld
```

9.2 Ejecución

A continuación se van a detallar las opciones de ejecución de los programas desarrollados.

9.2.1 Ejecución del proceso servidor

La ejecución es sencilla, basta ejecutar el comando:

```
# ./server <puerto>
```

Donde <puerto> indica el puerto a usar en el sistema. Este sera el puerto TCP usado, mientras que para Bluetooth, se usara como canal, el numero indicado en <puerto> dividido entre mil, de manera que si indicamos el puerto 5000 se usara el canal 5.

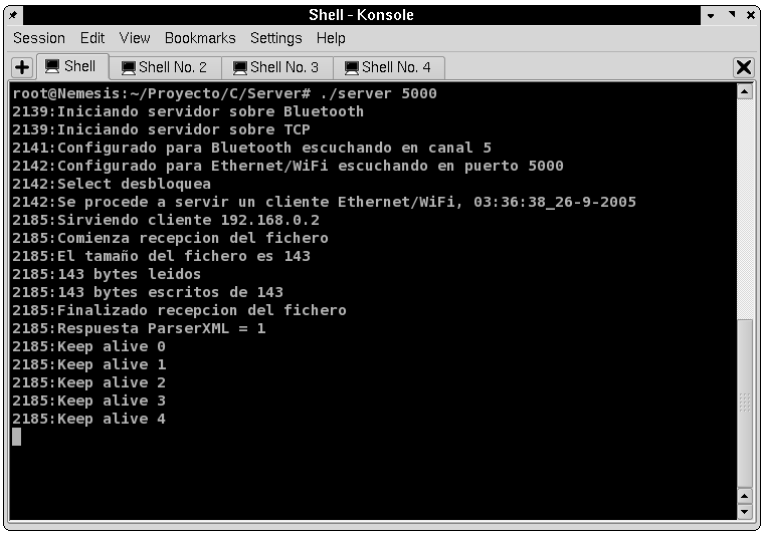


Figura 9.1 - Pantalla de ejecución típica del servidor

Antes de arrancar el servidor, hemos de asegurarnos de que se cumplen algunos requisitos:

- Si vamos a servir clientes TCP/IP, deberemos tener al menos un interfaz IP debidamente configurado.
- Si vamos a servir clientes Bluetooth, deberemos tener al menos un interfaz Bluetooth debidamente configurado.
- Puesto que el proceso servidor se comunica con el procesador XML mediante el socket local /tmp/xmlparser, si este no ha sido arrancado, ninguna ficha podrá ser validada, por lo que se supondrán todas erróneas. Asegurese de que arranca el procesador XML.

9.2.1.1 Configuración del interfaz IP

Bastará con que existe un interfaz IP configurado con una IP válida. En el caso de existir varias interfaces validas, el proceso servidor escucha todas ellas.

9.2.1.2 Configuración del interfaz Bluetooth

En primer lugar nos aseguraremos de que el interfaz esta en estado "up" (en el ejemplo se supone el interfaz 0):

```
# hciconfig hci0 up
```

A continuación es necesario que carguemos el módulo del protocolo sco y el módulo del protocolo rfcomm:

```
# modprobe rfcomm
# modprobe sco
```

Seguidamente estableceremos una relación entre el nodo /dev/rfcomm0 (por ejemplo) con todas las conexiones entrantes por el canal 5:

```
# rfcomm bind /dev/rfcomm0 all 5
```

Ahora cargamos el demonio para el protocolo de descubrimiento de servicios, y registramos en el canal 5, el servicio de puerto serie:

```
# sdpd
# sdptool add --channel=5 SP
```

9.2.2 Ejecución del procesador XML

Bastará con ejecutar:

```
# ./xmlparser
```

No es necesario pasarle ninguna información como argumento, aunque si se le pasa una ficha XML, podremos usarlo como validador de fichas.

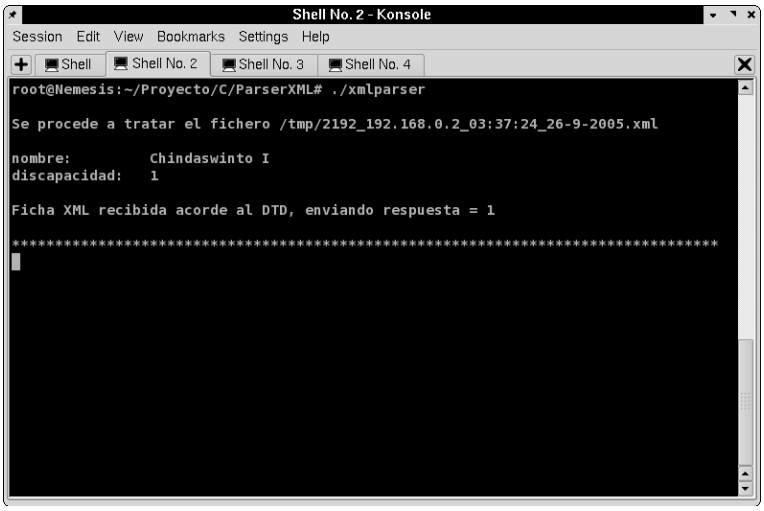


Figura 9.2 - Pantalla de ejecución típica del procesador XML

El procesador XML se comunica con el proceso que gestiona el periférico conectado en el puerto paralelo a través del socket local /tmp/ppserv. Si este proceso no esta ejecutandose, las fichas no serán validadas.

Capitulo 9 - Manual de uso

Capítulo 10

Pruebas

10.1 Equipos utilizados para las pruebas

- Dongle bluetooth
- PDA iPAQ 5550
- Sony Ericsson P800
- Tarjeta Wifi
- Mini-ordenador SB-X255 de Compulab
- PC compatible con Linux

10.2 Configuración del sistema

Se conectaron y situaron los equipos tal y como se muestra en la figura. Se puede observar que la antena wireless se coloco en posición de máxima directividad en relación a la posición de los dispositivos inalámbricos. Por otro lado, se utilizo un pasillo del laboratorio a modo de paso cebra.

Figura 10.1 - Equipo de pruebas A

10.3 Pruebas

Para las pruebas con respecto a los tiempos se comprobó que una deficiencia locomotriz de grado uno podría tener un factor multiplicativo de cuatro en cuanto al tiempo en relación a la temporización normal del sistema. Por otro lado, una deficiencia de grado dos podría tener un factor multiplicativo de aproximadamente tres, mientras que una de grado tres un factor multiplicativo de dos.

Figura 10.2 - PDA usada en las pruebas

Se comprobó que como era de esperar el alcance de los Bluetooth era de 10 metros aproximadamente (a esta distancia salían de la zona de cobertura) mientras que el wifi alcanzaba unos 30 metros. La disminución de las distancias se debe a las reflexiones y difracciones que se producían en el entorno de pruebas, al no utilizar como medio de transmisión el espacio libre.

El sistema también fue sometido a una prueba con varios usuarios, donde se comprobó que éste respondía correctamente incluso con una gran demanda de peticiones, haciendo prevalecer de esta forma aquella el caso más restrictivo.

Figura 10.3 - Móvil usado en las pruebas

El sistema no presentó ningún problema por estar en funcionamiento con dos tecnologías distintas como son Bluetooth y Wifi, que son dos tecnologías que funcionan a la misma frecuencia de 2.4 Ghz. Por otro lado el sistema en Bluetooth no estaba dotado de encriptación, quedando este último para proyectos futuros. En wifi se encriptó por WEP, pero sería mejor una técnica de encriptado como SSL, ya que de esta forma nos ahorraríamos muchos problemas si el sistema se llegase a desarrollar en un entorno real.

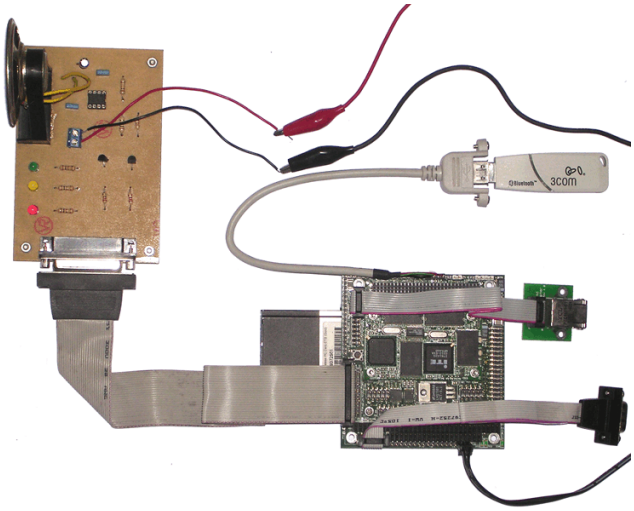


Figura 10.4 - Equipo de pruebas B

Capitulo 10 - Pruebas

Un problema que surgió y que queda para un desarrollo posterior de este proyecto fue el largo alcance de los dispositivos. Como consecuencia de ésto, el sistema se activa aunque el discapacitado esté a varios a metros del semáforo. Una solución sería el diseño de una antena que se adecuada para nuestro proposito tal y cómo se explica en líneas futuras

Conclusión y líneas futuras

Este proyecto ha consistido en la elaboración de un sistema capaz de regular el paso de una vía regida por un semáforo. El sistema ha sido desarrollado con la intención de ser utilizados para el beneficio de personas que sufren algún tipo de discapacidad, y a las cuales el mero hecho de cruzar una vía o tantear donde está el botón que hace cambiar el estado del semáforo a verde les supone un gran esfuerzo.

Ya que el sistema está destinado a este tipo de personas unos de los principales esfuerzos se ha centrado en conseguir que el sistema sea los más automático posible, para liberar así al usuario de la tediosa tarea de interactuar con el dispositivo móvil.

Este sistema ha consistido en el desarrollo de una aplicación cliente-servidor. La parte del cliente se desarrolló para dos tipos de dispositivos inalámbricos distintos (móvil y PDA) usando también dos tecnologías inalámbricas distintas para cada uno (Bluetooth y Wi-fi). La parte del servidor se desarrollo pará un mini ordenador, el cual por su pequeño tamaño seria fácilmente utilizable para su funcionamiento en semáforos. Éste también está adaptado para servir a clientes por medio de las dos tecnologías anteriormente comentadas.

Lineas futuras

El presente proyecto deja puertas abiertas a futuras investigaciones, debido a la continua y rápida evolución que experimentan todos los tipos de redes inalámbricas. Algunas posibles investigaciones son:

- Añadir seguridad a las comunicaciones inalámbricas.
- Posibilidad de modificar la potencia de transmisión de las antenas, para así poder adaptarlas al entorno según las necesidades pertinentes.