Configuración y administración de una red mesh: MAYA

Realizado por: David Manzano González
Dirigido por: Pietro Manzoni

UNIVERSIDAD
POLITECNICA
DE VALENCIA

11 de marzo de 2007

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Tecnología Wi-Fi

A pesar de que este proyecto está orientado a la redes *mesh*, si queremos comprender mejor su utilidad, debemos empezar hablando de la tecnología sobre la que se apoyan este tipo de redes para proveer su servicio, la tecnología *Wi-Fi*.

1.1.1. Ventajas de la tecnología Wi-Fi

Wi-Fi, abreviatura de *Wireless Fidelity*, es un conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11. Como hemos mencionado anteriormente, las redes *mesh* o redes malladas se basan en la tecnología *Wi-Fi*, por tanto, aprovecha las ventajas que proporciona las comunicaciones inalámbricas, que constituyen una forma versátil de construir una red.

Si bien el estado actual de la tecnología inalámbrica no ofrece las altas velocidades de transferencia de las redes cableadas, ni su seguridad ni fiabilidad, la flexibilidad que proporciona justifica su uso e implementación. Resulta extremadamente cómodo la creación de una red que no esta sujeta a las limitaciones de movimiento y que puede estar en marcha en sólo unos minutos después de encender las estaciones de trabajo, además de proporcionarnos acceso a Internet de forma inmediata desde sitios hace años impensables, como parques, playas o cualquier lugar situado en el exterior de edificios o, incluso, desde construcciones históricas donde no podríamos realizar una instalación cableada.

Volviendo al aspecto de la velocidad en comunicaciones inalámbricas, es evidente que es una de las principales desventajas de las comunicaciones inalámbricas respecto a las cableadas, puesto que el límite de velocidad en *Wi-Fi* alcanza los 54 Mbps mientras que en tecnologías para redes cableadas llegan a alcanzar 10Gps. Si bien, esta reconocido oficialmente que nunca se alcanzan los 54Mbps puesto que es una medida teórica, en realidad la velocidad máxima alcanzable es de 27Mbps y eso en el caso de que la distancia sea de pocos metros entre los nodos (de hecho, para unos 60 metros la velocidad máxima se sitúa alrededor

de 5 Mbps). Además, el aire, medio de transmisión de las comunicaciones inalámbricas, es mucho más vulnerable que el cable, aun así los nuevas sistemas de cifrado para la tecnología 802.11, WPA y WPA2, usados de forma correcta deben proporcionarnos una comunicación totalmente segura.

Hay que tener en cuenta que la tecnología *Wi-Fi* lleva una gran desventaja de tiempo, y con la progresión y el auge en el que se encuentra actualmente, se espera un crecimiento imparable. Cada vez se fabrican mayor número de dispositivos que hace uso de este tipo de comunicaciones debido a la demanda de los usuarios, que prefieren la comodidad por encima de todo. Claros ejemplos de ello son PDAs, cámaras, teléfonos móviles y toda la nueva generación de vídeoconsolas (*Nintendo DS*, *Sony PSP*, *XBOX 360*, *Wii* o *Playstation 3*).

Esta demanda provoca que se realicen constantes mejoras y nuevos estándares para *Wi-Fi*, de hecho, un nuevo grupo del IEEE ya esta investigando formas de aumentar la velocidad en rangos desde los 108 Mbps hasta los 300 Mbps que formarán parte del IEEE 802.11n, así como protocolos de calidad de servicio (cómo WMM), que permitirán usar la tecnología VOIP (*Voice Over Ip Protocol*) y mecanismo de ahorro de energía entre otros que se establecerán el IEEE 802.11e.

Podemos decir que, debido a la utilidad y las ventajas de esta tecnología, basarnos en ella para desarrollar un producto o una nueva tecnología nos va a proporcionar un punto extra de partida de cara conseguir un buen resultado. Posteriormente, cuando nos adentremos en el campo de las redes *mesh*, veremos que muchas de las ventajas que proporciona son las de emplear tecnología inalámbrica. Por tanto, las redes *mesh* constituyen igualmente un sistema de comunicación de gran futuro y crecimiento, como demuestran las grandes inversiones ecónomicas que estan realizando empresas de la talla de *Microsoft* o *Google* en investigación y desarrollo de este tipo de sistemas.

1.1.2. Proyectos y comunidades Wi-Fi

Las comunidades *Wi-Fi*, tienen como objetivo crear grandes zonas de acceso a la red, sin cables y de manera totalmente gratuita. Su punto de partida es la emisión de señal de Internet desde routers inalámbricos instalados en los domicilios de los usuarios. La teoría en que se basan estas redes es en la de unir muchos focos emisores cercanos entre sí, con el objetivo de crear una amplia área inalámbrica.

El máximo exponente en este aspecto es FON[1], considerada la mayor comunidad Wi-Fi del mundo. La original idea de este grupo fue la de crear una comunidad de usuarios, de modo que al ser miembro de esta comunidad podamos utilizar la conexión de otros miembros para acceder a Internet a cambio de compartir con ellos nuestra conexión. Esto nos permite el acceso a la red gratuito desde cualquier punto del mundo que exista un usuario de esta comunidad. Además, cuenta con un localizador de puntos de acceso FON a nivel mundial, basado en el *Google Maps* (inversor en este proyecto) con lo que nos permite conocer los miembros que se hayan en nuestra ciudad, nuestro barrio e incluso nuestra calle.

Todo este sistema se caracteriza por la sencilla instalación, tan sólo hay que contar con un router FON que nos permitirá generar dos señales de red inalámbrica, una privada, que estará cifrada y ofrecerá privacidad absoluta que será la utilizada por el usuario y otra pública que será utilizada por el resto de usuarios. Estos usuarios serán usuarios registrados y deberán usar contraseña y password para acceder. Además permite controlar el ancho de banda para el usuario propietario y para el resto, con lo que es muy cómodo para éste y le ofrece total seguridad. Cabe mencionar que la hegemonía de FON parece que va a ser destronada por la francesa FREE[2] que van a proporcionar 300.000 puntos de acceso a sus clientes para formar la comunidad *Wi-Fi* más grande del mundo, aunque se espera una fuerte reacción de FON y sus inversores más importantes como el mencionado *Google* o *Skype*.

La idea de FON partió de expandir lo que se como *muniFi*, término con el se conocen las redes municipales *Wi-Fi*. Estas redes, financiadas por el estado o por proveedores de contenidos, se encargan de proporcionar cobertura en una ciudad o parte de ella a los dispositivos móviles para el acceso a Internet. Su crecimiento es imparable, cada vez son más las ciudades a nivel mundial que están incorporando este sistema; San Francisco, Nueva Orleans, Nueva York o Dublín son sólo un pequeño ejemplo de las más conocidas.

Hay que mencionar que actualmente no existe una ciudad cuya red *Wi-Fi* cubra todo su territorio, por motivos económicos sólo se suelen cubrir las zonas con mayor número de habitantes, como el centro de las ciudades o instalaciones específicas como aeropuertos. Sin embargo, en un breve espacio de tiempo esto cambiará radicalmente, Manchester ya anunciado que creará una red inalámbrica municipal cubriendo una superficie de unos 1.000 Km^2, es decir, todo el territorio de la ciudad. En España, ya se han realizado pruebas en diversas ciudades como San Sebastián, y se planea su instalación en ciudades como Córdoba, dónde se está usando como reclamo electoral.

1.1.3. WiMAX, ¿El sustituto de Wi-Fi?

Muchas son las expectativas que han surgido entorno a esta nueva tecnología inalámbrica que amenaza, incluso, con sustituir al *Wi-Fi*. WiMAX (IEEE 802.16) es un estándar de transmisión inalámbrica de datos proporcionando accesos concurrentes en áreas de hasta 48 kilómetros de radio y a velocidades de hasta 70 Mbps utilizando tecnología que no requiere visión directa NLOS.

Con esta descripción ya nos podemos hacer una idea de cuáles son las ventajas de WiMAX frente a Wi-Fi; mayor alcance y mayor velocidad (recordemos que la cobertura Wi-Fi es de unos 100 metros y la velocidad de 54Mbps), lo que también supone un incremento de usuarios beneficiados por una misma conexión.

A través de WiMAX es posible ahorrar cientos y quizás miles de kilómetros de cables, además del personal encargado de instalar los cables (la misma ventaja que en Wi-Fi pero mucho más acentuada por el gran incremento de la distancia). Las antenas de distribución de WiMAX son baratas ya que cuestan entre 15 y 20 euros, en comparación con una antena de telefonía celular, que llega hasta los

80 euros. Por lo tanto, WiMAX, es barato para el distribuidor y para el cliente.

Ahora bien, no tenemos que ver todo esto como una lucha entre WiMAX y *Wi-Fi* por imponerse el uno sobre el otro. Puesto que, precisamente, puede que la mejor solución sea la combinación de las dos tecnologías, dependiendo en cada caso del tiempo de implementación, la posición geográfica y la aplicación de red (tanto en datos, VoIP y vídeo). Una idea interesante es utilizar WiMAX para conectar diversas zonas que usan *Wi-Fi*, o conectar zonas *Wi-Fi* con el ISP (*Internet Service Provider*) como podemos ver en la figura 1.1. Otro ejemplo, dadas las características de cada tecnología, sería facilitar acceso *Wi-Fi* en núcleos urbanos donde muchos de los dispositivos actuales (como teléfonos móviles o PDAs) incorporan esta tecnología y usar WiMAX para dar acceso a zonas suburbanas o rurales, que por su ubicación lejana y dispersa, no sería económico cubrirla con una red *Wi-Fi*.

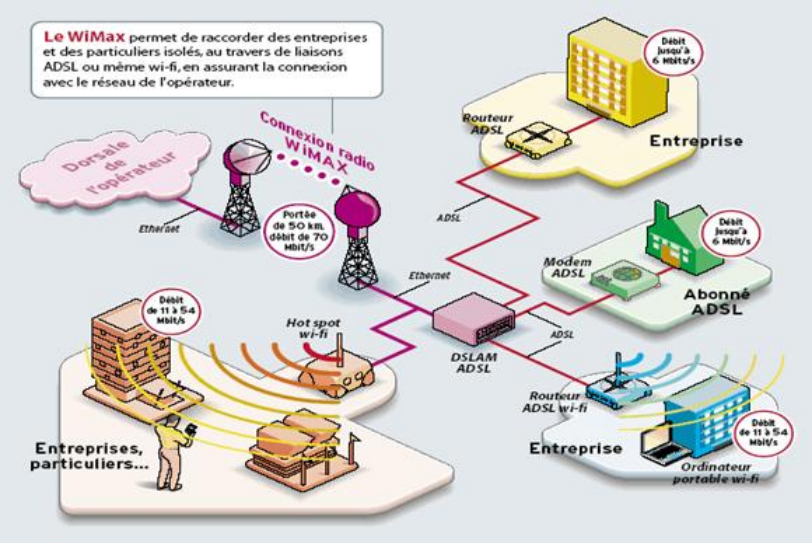

Figura 1.1: Ejemplo de uso de WiMAX

1.2. Redes mesh

1.2.1. ¿Qué es una red mesh?

Los inicios de las redes mesh o redes acopladas son, como no, militares. Inicialmente se usaron para comunicarse con aquellas unidades de militares que aún estando lejos de las zonas de cobertura de sus mandos estaban lo suficientemente cerca entre sí como para formar una cadena a través de la cual se pudiese ir pasando los mensajes hasta llegar a su destino (los mandos).

Las redes mesh, para definirlas de una forma sencilla, son aquellas redes en las que se mezclan las dos topologías de las redes inalámbricas. Básicamente son redes con topología de infraestructura, pero que permiten unirse a la red a dispositivos que a pesar de estar fuera del rango de cobertura de los puntos de acceso están dentro del rango de cobertura de algún nodo móvil que directamente o indirectamente está dentro del rango de cobertura del punto de acceso (topología sin infraestructura o *ad-hoc*).

También permiten que los nodos móviles se comuniquen independientemente del punto de acceso entre sí. Actúan como repetidores para transmitir la información de los nodos alcanzables hasta los extremos que quieren comunicarse y que están demasiado lejos para alcanzarse, dando como resultado una red que permite abarcar grandes distancias. Además son un tipo de red muy fiable, ya que cada nodo esta conectado a varios nodos. Si uno de ellos falla o abandona la red, sus vecinos simplemente buscarán otra ruta. Para ampliar la capacidad y la fiabilidad de la red basta con añadir más nodos.

Para que todo esto sea posible es necesario el contar con un protocolo de enrutamiento que permita transmitir la información hasta su destino con el mínimo número de saltos o con un número que aún no siendo el mínimo sea suficientemente bueno. Antiguamente las redes *mesh* no se usaban porque el cableado necesario para establecer la conexión entre todos los nodos era imposible de instalar y de mantener. Hoy en día con la aparición de las redes *wireless* este problema desaparece y nos permite disfrutar de sus grandes posibilidades y beneficios, entre los más destacados:

- Acceso a Internet gratuito para varios usuarios.
- Compatir entre los usuarios archivos, impresoras y otros dispositivos.
- Coste de insfraestructura bajo.

Por la proliferación y utilidad de este tipo de redes, así como de la tecnología en la que se basa, se ha considerado muy útil la implementación de un software que permita administrarlas de una forma rápida y cómoda desde un cualquier punto de la red.

1.2.2. Instalación y configuración de una red mesh

Los elementos que un usuario necesita para crear una red mesh son únicamente puntos de acceso (dispositivos de red *wireless*) funcionando en modo *ad-hoc* y al menos uno haciendolo también en modo infraestructura para proporcionar acceso a Internet a todos los usuarios de la red.

Puesto que los conceptos modo infraestructura y modo *ad-hoc* van a aparecer repetidas veces en este documento es importante que quede claro su significado. En el modo infraestructura los puntos de acceso funcionan de forma equivalente a los *hubs* o concentradores, permitiendo que varios clientes *wireless* se comuniquen entre sí. A menudo se utilizan varios puntos de acceso para cubrir un

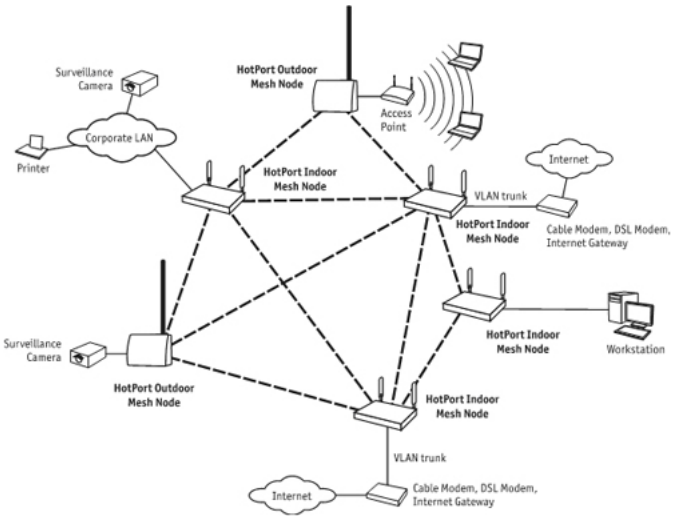

Figura 1.2: Ejemplo de una red mesh

área determinada como una casa, una oficina u otro tipo de localización delimitada. Los puntos de acceso poseen típicamente varias conexiones de red: la tarjeta *wireless* y una o más tarjetas Ethernet que se utilizan para comunicarse con el resto de la red. En el modo *ad-hoc* no existen conexiones establecidas entre los distintos puntos de acceso, de forma que cuando un dispositivo se quiere comunicar con otro que esta fuera de su rango de alcance envía el mensaje a los dispositivos alcanzables que actuarán como repetidores, reenviando el mensaje hasta llegar al dispositivo destino. Las configuraciones *ad-hoc* son comunicaciones punto a punto. Aquí entra en juego la función del protocolo de encaminamiento, que se encargará de encontrar un camino desde el dispositivo emisor al dispositivo receptor sin saturar la red. Para que los dispositivos de una red funcionen en modo *ad-hoc* es necesario configurarlos para que operen en ese modo, accediendo, por ejemplo, a la interfaz web del punto de acceso y también es necesario que se encuentren en el mismo canal y ESSID. En la figura 1.3 podemos ver un ejemplo de cada uno de los modos de funcionamiento.

Cada fabricante proporciona su *firmware* para administrar el punto de acceso, sin embargo, este *firmware* es, en la gran mayoría de los casos, muy poco flexible. Actualmente encontramos varias soluciones para proporcionar una mayor libertad de configuración a estos dispositivos y que permitirá a usuarios un poco más avanzados o familiarizados con el entorno Linux sacarle el máximo partido al punto de acceso. Hablamos de software como OpenWRT[3], actualmente el más utilizado dentro de este grupo y el que se ha utilizado en la realización de este proyecto.

OpenWRT es una distribución de Linux para sistemas empotrados. En lugar del *firmware* estático proporcionado por el proveedor, este software nos proporciona un sistema totalmente configurable. Esto libera al usuario de las

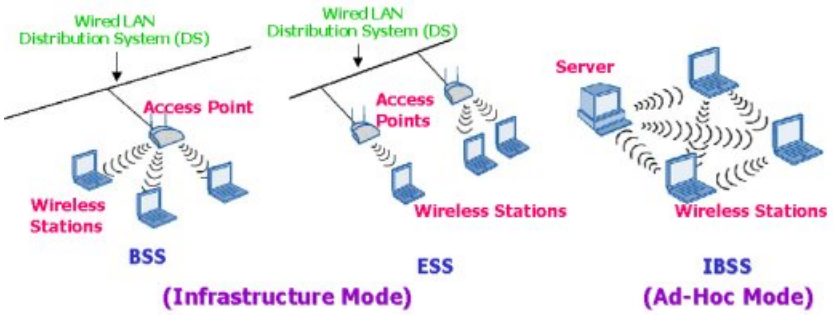

Figura 1.3: Modo Ad-hoc y modo infraestructura

aplicaciones y configuraciones suministradas por el vendedor y permite personalizar el dispositivo mediante el uso de extensiones (paquetes) que nos permitirá dotar de muy variada funcionalidad a nuestro punto de acceso. OpenWRT ofrece una enorme variedad de herramientas para ser instaladas (algunas de ellas reducidas para adaptarlas al tamaño de la memoria de un *router*), por ejemplo, servidor SNMP para obtener información y estadísticas del dispositivo, *tcpdump* para examinar los paquetes que circulan en la red o *iptables* el conocido firewall de Linux entre otros, además de infinidad de librerías como las de *OpenSSL* y muchas otras utilidades. Todo ello sin contar que OpenWRT ofrece un *framework* para construir un aplicación sin tener que construir un *firmware* por completo a su alrededor, esto proporciona para los usuarios una personalización absoluta y muchas formas de usar su dispositivo.

Así pues el tener un Linux en miniatura en el router nos proporciona todas las ventajas de este sistema operativo en nuestro dispositivo (salvando la diferencia de espacio de almacenamiento) y nos permite realizar e instalar nuestras propias aplicaciones, algo imposible con el *firmware* de fábrica. Hay que mencionar también, que en el entorno de desarrollo de aplicaciones que ofrece OpenWRT encontramos numerosas utilidades como el compilador cruzado, para la compilación de código (como C o C++) desde nuestra arquitectura (probablemente i386) a la arquitectura del dispositivo (normalmente MIPS).

Además, OpenWRT sigue contando con interfaz web para la configuración de los parámetros más importantes, por tanto, los usuarios no iniciados no tendrán ningún problema de adaptación a esta utilidad. En cambio, usuarios un poco más avanzados podrán acceder por SSH al sistema Linux y cambiar los mismos parámetros por línea de comandos o realizar otras acciones que serán básicas para nuestra red mesh, como es la instalación del protocolo de encaminamiento.

Como se acaba de mencionar, en una red con dispositivos funcionando en modo ad-hoc es necesaria la presencia de un protocolo de encaminamiento, que proporcione un camino relativamente bueno desde un origen a un destino. Debido a la creciente popularidad de OpenWRT, hay varios protocolos que son ofrecidos con la garantía de haber sido probados en dispositivos con este sistema, como es el caso de la implementación de DSR de la universidad de Upp-

sala, DSR-UU, además de otros que también están desarrollando[4]. Hay otros que se encuentran en el directorio de paquetes ofrecido por el propio OpenWRT para ser instalados directamente como el OLSR, y otros muchos que podemos compilar para el dispositivo (como el AODV o DYMO) y comparar cual se adapta mejor a las necesidades de nuestro sistema. Algo importante a tener en cuenta es que los usuarios que deseen forma parte de esta red, tendrán que ejecutar en su máquina el mismo protocolo de encaminamiento que estén usando los protocolos (y quizá la misma versión) para que el sistema funcione.

Con esto, ya tendríamos configurada la parte ad-hoc de la red mesh, un grupo de puntos de acceso distribuidos físicamente para ofrecer cobertura a usuarios que se deseen conectar a la red. Ahora bien, aún falta un paso básico para la creación de la red mesh y no es otro que proporcionar acceso a Internet, para ello, al menos un punto de acceso que forma la red *ad-hoc* debe encaminar los paquetes de la red *ad-hoc* y viceversa. En el caso de salida de paquetes de la red hacía internet deberá desechar las cabeceras del protocolo de encaminamiento para que la información viaje por internet, y en el caso inverso deberá añadirlos, para que la información llegue al nodo de la red *ad-hoc*.

1.2.3. Productos en el mercado para la construcción de una red mesh

A la hora de la adquisición de hardware para la construcción de una red de este tipo, el usuario se encontrará con un gran abanico de posibilidades en cuanto a productos y complementos. Sin duda, la elección de los puntos de acceso (la base de la infraestructura de la red) constituirá una fase muy importante en la construcción de la red.

Empresas como Broadcom, PRISM (Intersil) o RealTek son los principales fabricantes de *chipsets* que utilizan los diferentes puntos de acceso del mercado. Son muchos los fabricantes y modelos de estos dispositivos, en la tabla 1.1 podemos observar las carácteristicas de los más importantes dentro de los que han demostrado total compatibilidad con OpenWRT.

Sin embargo, no hemos mencionado el que es el más utilizado y recomendado (por razones que especificaremos a continuación) por expertos y usuarios, el WRT54G fabricado por *Linksys*, que permite interconectar varios ordenadores mediante enlaces Ehernet 802.3 y 802.11g inalámbricas. El modelo WRT54GS es prácticamente idéntico, excepto por el aumento de memoria RAM y la incorporación de la tecnología *SpeedBoost*. Este router es único entre los dispositivos de consumo doméstico, debido a que los desarrolladores de *Linksys* tuvieron que liberar el código fuente del *firmware* del router para cumplir con las obligaciones de la GNU GPL. Este hecho permite a los entusiastas de la programación modificar el firmware para añadir o cambiar funcionalidades del dispositivo. Existen varios proyectos de desarrollo que proveen versiones mejoradas del firmware para el WRT54G, como el ya mencionado antes; OpenWRT y otros como: *Batbox* o *HyperWRT*. Y ha servido de base para el desarrollo de numerosas comunidades *wireless*, como la mencionada FON.

Fabricante	Modelo	Plataforma y frecuencia	Flash	RAM	Wireless NIC
4G Systems	Cube MTX-1	AMD ALCH. 1500@400MHz	32MB	64MB	Atheros Mini-PCI
ALLNET	ALL0277	Broadcom 4710@125MHz	4MB	16MB	N/A
Asus	WL-500g	Broadcom 4704@266MHz	8MB	32Mb	Broadcom 4321
Belkin	F5D7230-4	Broadcom 4710@125MHz	4MB	16MB	Broadcom (mini-PCI)
Buffalo	WZR-RS-G54	Broadcom 4704@266MHz	8MB	64MB	Broadcom (mini-PCI)
Dell	TrueMobile 2300	Broadcom 4710@125MHz	4MB	16MB	Broadcom (mini-PCI)
Microsoft	MN-700	Broadcom 4710@125MHz	4MB	16MB	Broadcom (mini-PCI)
Motorola	WR850GP	Broadcom 4712@200MHz	4MB	16MB	Broadcom (integrated)
Siemens	SE505	Broadcom 4710@125MHz	4MB	16MB	Broadcom (mini-PCI)
US ROBOTICS	USR5430	Broadcom 4712@200MHz	2MB	8MB	Broadcom (integrated)
Viewsonic	WR100	Broadcom 4712@200MHz	4MB	8MB	Broadcom (integrated)

Cuadro 1.1: Puntos de acceso más destacados

El WRT54G original estaba equipado con una CPU MIPS a 125 MHz con 16 MB de memoria RAM y 4 MB de memoria *flash* para almacenar el *firmware*. En revisiones posteriores, se aumentó la velocidad de la CPU a 200 MHz y se doblaron tanto la memoria RAM como la flash a 32 y 8 MB, respectivamente. Todos los modelos vienen con un *switch* de 5 puertos (el puerto para Internet está en el mismo switch pero en una VLAN diferente) y con un chipset inalámbrico de *Broadcom*. Asimismo, dispone de dos antenas externas conectadas a través de conectores de polaridad inversa TNC. Hay que mencionar que la última versión del WRT54G (la versión 5) no aceptará *firmware* de terceros dado que el sistema ya no corre sobre Linux, desde hace poco tiene *firmware* de terceros como DD-WRT.

Figura 1.4: Linksys WRT54G

En cuanto a lo que software se refiere, además del mencionado OpenWRT, hay otros grupos que desarrollan software gratuito relacionado con las redes *mesh*. Un ejemplo de ello es CUWiN [5](the *Champaign-Urbana Community Wireless Network*), que se encarga de desarrollar *software* de código y arquitectura abierta para redes *mesh* a través de la marca *CUWiNware*. CUWiN esta considerado como el líder a nivel mundial de software para redes *mesh* dinámicas. Durante más de seis años se han encargado de desarrollar protocolos para hacer a las redes mesh más eficientes y escalables que cual otra tecnología que conozcamos. Actualmente, cuenta con más de 80 desarrolladores en todo el mundo que colaboran en diversos aspectos de la investigación y desarrollo.

Otro *software* interesante puede ser el que proporciona *Wifidog*[6]. Se trata de un proyecto que proporciona una solución empotrable y completa para el uso de un *captive portal* dentro de una comunidad *wireless* o para personas que quieren compartir su punto de acceso y desean evitar que se produzcan abusos de su conexión a internet.

Wifidog esta diseñado para tener la opción de tener un control de centralizado del acceso, reparto del ancho de banda de cada cuenta, lista de nodos activos e información especifica de cada punto de acceso y cada cliente. No esta basado en *javascript* por lo que funciona en cualquier plataforma con un navegador, incluso en PDAs o teléfonos móviles. Esta desarrollado en C para que sea sencilla la instalación en sistemas empotrados (ha sido diseñado precisamente para el *Linksys* WRT54G usando OpenWRT, pero funciona en cualquier plataforma de

Linux reciente), además de estar adecuado a la capacidad de almacenamiento de estos dispositivos (la instalación máxima apenas ocupa 30 KB).

El *captive portal Wifidog* se compone de un *gateway* y de un servidor de autenticación; el servidor de autenticación esta elaborado en PHP con base de datos en PostgreSQL y permitirá autenticar clientes en un entorno *captive portal*. Desde aquí el administrador podrá manejar las cuentas de usuario, estadísticas y podrá revisar las características específicas de cada usuario mediante los *logs* almacenados. El *gateway* será el encargado de conectarse al servidor de autenticación para comprobar si debe aceptar o denegar el acceso a un determinado usuario.

Capítulo 2

Objetivos

El objetivo de este proyecto es, básicamente, la creación de una herramienta *software* que permita al administrador de la red *mesh* configurar y administrar los dispositivos que forman su infraestructura, formada por puntos de acceso. Éstos contarán con un *firmware* altamente configurable y personalizable y la idea principal es la de proporcionar una forma cómoda y sencilla de poder realizar cambios en los parámetros de la red sin tener que acceder a cada dispositivo uno por uno.

Por consiguiente, vamos a basarnos en ése *firmware* de código abierto, con el objetivo de añadir la funcionalidad necesaria para, aprovechandonos de sus características, proporcionar soporte y adaptación a las necesidades de una red mesh.

En capítulos posteriores explicaremos la funcionalidad completa de la herramienta, a la que conoceremos con el nombre de MAYA, que contará con una una interfaz muy sencilla para el usuario. La elaboración de esta utilidad también implicará la instalación del *software,* tanto en el punto de control[1], como en los puntos de acceso que forman parte de la red, en el apéndice explicaremos los pasos de para la puesta a punto de la herramienta, así como de la instalación del *software* necesario con el que debe contar nuestro sistema.

Así pues, creemos que esta sencilla herramienta puede ser de gran utilidad, no sólo por la funcionalidad de la que ya dispone, si no también por la facilidad de ampliarla. En el capítulo 7 hablaremos de las posibles mejoras y del trabajo futuro.

[1] Este será el término con el que nos referiremos al computador desde el cual podremos controlar los dispositivos de la red.

Capítulo 3

Arquitectura

3.1. Infraestructura del sistema

La infraestrutura necesaria para el correcto funcionamiento del sistema consiste en un equipo, dónde se ubicara la aplicación central, con sistema operativo Linux y con una tarjeta *wireless*. Este equipo deberá formar parte de una red *mesh,* esto no implica que deba estar permanentemente dentro de la red, pero deberá estarlo a la hora de administrar los dispositivos que forman parte de ella. Un mismo equipo, por ejemplo un portátil, puede administrar todas las redes que se desee, mientras los dispositivos de esa red estén configurados para ello (y tengan instalado el *software* que proporcionamos) y el equipo pertenezca en ese momento a dicha red.

El otro componente básico para nuestra insfraestructura son, obviamente, los puntos de acceso. Todos ellos deberán estar funcionando en modo *ad-hoc* y correctamente configurados.

En resumen, la aplicación en sí se instalará en el punto de control y en los puntos de acceso será necesario el uso de un software muy reducido para comunicarse con el punto de control, además de una serie de configuraciones que se explicarán con todo detalle en el apéndice del documento.

En la figura 3.1 podemos observar la arquitectura global del sistema.

3.2. Lenguajes Utilizados

Los lenguajes de programación empleados en este proyecto han sido variados en función de las necesidades y de las limitaciones que implica programar para dispositivos como los puntos de acceso que cuentan con un espacio de almacenamiento mucho menor y un número de herramientas más reducido que un computador normal. Acto seguido enumeramos los lenguajes que hemos utilizado y para qué los hemos empleado.

1. HTML: Lenguaje básico de creación de creación de una página web. Se ha

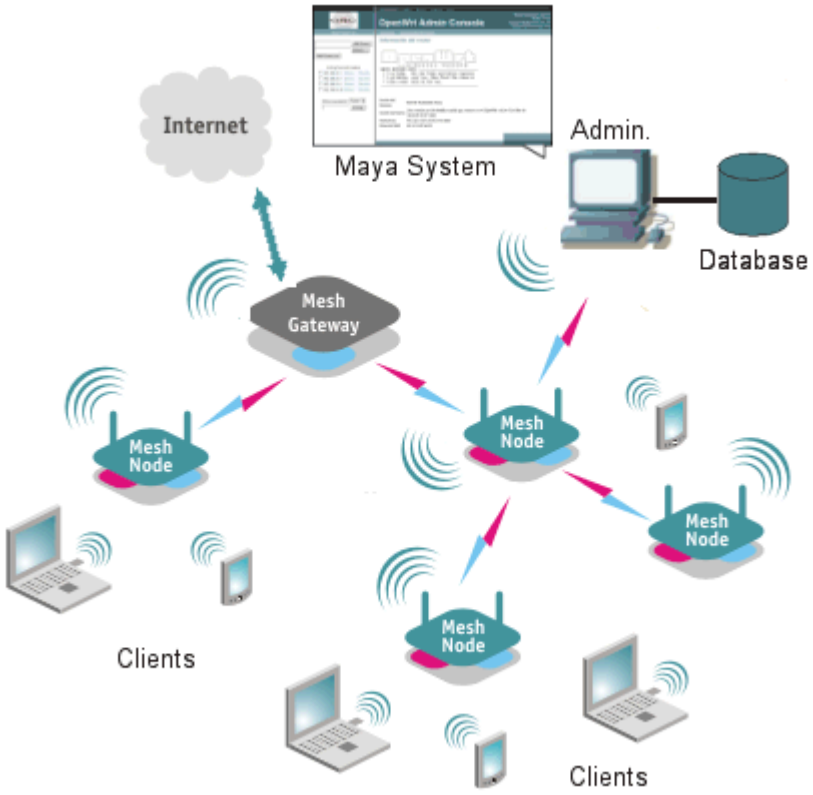

Figura 3.1: Arquitectura del sistema Maya

empleado para el aspecto básico de la aplicación (desarrollada en interfaz web) y que nos permitirá estrutucturarla.

2. CSS: Las hojas de estilo en cascada son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML. Esto nos facilitará el trabajo a la hora de crear la apariencia de la aplicación.

3. PHP: Es un lenguaje de programación interpretado se utiliza entre otras cosas para la programación de páginas web activas, y se destaca por su capacidad de mezclarse con el código HTML. Casi la totalidad de la aplicación principal ha sido escrita en este lenguaje, incluidas funcionalidades como el acceso a la base de datos, dónde se almacenará la información relativa a los dispositivos que forman una red.

4. Javascript: Es un lenguaje interpretado orientado a las páginas web, con una sintaxis semejante a la del lenguaje Java. Ha sido necesaria la uti-

lización de Javascript para poder solucionar diversos problemas en los que el uso de PHP únicamente, no nos permitía solucionarlo.

5. SQL: Es un lenguaje declarativo de acceso a bases de datos relacionales. Se encargará de mantener la información sobre los puntos de acceso en la base de datos, dónde se realizarán consultas y actualizaciones desde el código PHP.

6. C: Uno de los lenguajes más comunes de programación. Se ha utilizado para desarrollar pequeñas aplicaciones para los puntos de acceso dadas las características de ambos; el hecho de no ser un lenguaje de alto nivel, libera de bastante carga al dispositivo.

7. Shell Script: son programas escritos con comandos UNIX. Todas las opciones de cambio de parámetros en los dispositivos que ofrece la herramienta estan implementadas en Shell Script, asi como otros detalles que explicaremos más adelante.

Capítulo 4

Implementación

En este capítulo se pretende abordar como se han implentado todas las opciones de la herramienta. En este aspecto, el objetivo del sistema Maya consiste en crear un sistema de comunicación entre los equipo se va a encargar de administrar la red y los puntos de acceso que la forman, todo ello de una forma que garantice unas condiciones mínimas de seguridad. Todo ello lo vamos a explicar en esta sección, que se encuentra dividida en los siguientes partes:

- Explicación de un concepto fundamental en este sistema: la deshabilitación de un punto de acceso, esto aportará una funcionalidad importante, aunque también añadirá dificultades, que deberemos resolver, al correcto funcionamiento de la aplicación.
- El sistema de establecimiento de conexión entre el punto de control y los puntos de acceso, que será muy diferente en función de si se trata de un dispositivo habilitado o deshabilitado.
- Se tratará otro aspecto muy importante en la aplicación como es el cambio de parámetros, tanto a un sólo dispositivo como a todos los que forman una red de forma global. En este último caso habrá que tener muy en cuenta el orden de la realización de los cambios.
- Se tratará el aspecto de la seguridad, algo necesario por otra parte, ya que no deseamos que cualquier usuario con malas intenciones pueda controlar la configuración de los *routers* de la red *mesh*, que por otro lado supone un escenario bastante propicio para este tipo de usuarios.
- En último lugar hablaremos del manejo y de la lista de *routers* configurables por nuestra aplicación y las operaciones que ello traerá consigo.

4.1. Habilitar/Deshabilitar un punto de acceso

Esta es una de las ideas más importantes de la aplicación y la que abarca la mayoría de su complejidad. Dadas las características y la funcionalidad que

queríamos otorgar a nuestra herramienta, surgió la necesidad de crear estos dos conceptos que van a ser determinantes en todo nuestro desarrollo. Inicialmente podemos entender estos dos conceptos tal y como implica su significado, es decir, tener activo un punto de acceso o tenerlo desactivado. La utilidad de desactivar un dispositivo puede ser muy variada; desde desactivar un nodo que no esta siendo usado con el objetivo de reducir la carga de mensajes generados por el protocolo de encanamiento hasta desactivarlo con el fin de evitar que se unan a la red un determinado grupo de usuarios que accede a ella mediante ese dispositivo en concreto. A continuación vamos a explicar en profundidad que implican estos estados y como se han implementado.

4.1.1. Concepto

En primera instancia, hay que aclarar que el estado normal de un punto de acceso es habilitado. Este concepto se refiere a que no se restringe ningún tipo de tráfico y su funcionamiento es el que se espera de un dispositivo de estas características.

Así pues, dónde verdaderamente reside la complejidad de esta funcionalidad es en el concepto de deshabilitar un dispositivo . Para ello vamos a señalar cuáles van a ser las características de un punto de acceso desactivado:

- No recibirá ni enviará ningún otro tipo de tráfico
- No recibirá ni enviará el tráfico del protocolo de encaminamiento
- Sólo recibirá o enviará paquetes por un determinado puerto UDP

Aunque mencionamos que no recibe ningún tipo de tráfico, hemos querido destacar el hecho de que tampoco va a recibir ni enviar mensajes del protocolo de encaminamiento, un hecho que como vamos a comprobar es muy relevante y será el origen de la existencia de la tercera característica.

Si nos planteamos la desactivación de un router en función de la características hemos enumerado, el primer objetivo será aislarlo de cualquier tipo de tráfico, para ello, en este proyecto hemos optado por la que creemos más sencilla y robusta; *iptables*. Pese a que se va a explicar con bastante detalle si el lector quiere más información acerca de esta utilidad le aconsejamos que consulte un documento que hable ampliamente sobre ella[7]. Mediante este conocido *firewall* de los sistemas Linux podemos, con tan sólo tres instrucciónes; que no reciba, envíe ni encamine tráfico. Estas instrucciones son:

- *iptables -I INPUT -j DROP*
- *iptables -I OUTPUT -j DROP*
- *iptables -I FORWARD -j DROP*

El funcionamiento de *Iptables* se basa en tres cadenas de reglas: INPUT, OUTPUT y FORWARD que se refieren a los paquetes que van dirigidos a ese dispositivo, los que envía él mismo y los que llegan a él con otro destino y debe

encaminar, por eso debemos usar instrucciones individualizadas para cada caso. Así pues con las tres instrucciones anteriores hemos configurado el dispositivo para que rechace todos los paquetes en cualquiera de los casos.

Este precisamente es el funcionamiento cuando utilizamos la opción desactivar router desde la aplicación. Para ello accedemos mediante SSH desde el código PHP y ejecutamos las intrucciones. Sin embargo, esto produce un problema; obviamente, ejecutando esas instrucciones, el punto de acceso queda totalmente aislado del tráfico, así pues, no tendríamos ninguna forma de volverlo a activar a no ser de llegar hasta la ubicación física del router y reiniciarlo manualmente. Por supuesto esta solución no es en absoluto válida ya que contradeciría la fílosofía de este proyecto respecto a la creación de una herramienta cómoda y centralizada para el administrador.

Por tanto, este "aislamiento" del *router* no puede ser total, por ello hemos optado por crear un servidor que estará esperando recibir mensajes en un puerto UDP (el sistema Maya usa por defecto el puerto 5022). Para ello, deberemos añadir los siguiente comandos para actualizar *iptables:*

- *iptables -I INPUT -p UDP –destination-port 5022 -j ACCEPT*
- *iptables -I OUTPUT -p UDP –destination-port 5022 -j ACCEPT*

El motivo de no añadir la misma regla para el conjunto de los paquetes a encaminar es que, las intrucciones del punto de control llegarán a su destino a través de mensajes *broadcast* (sección 4.2), no del protocolo de encaminamiento, por tanto es innecesaria. Así pues, estas dos instrucciones unidas a las otras tres anteriores, serían las que ejecuta un dispositivo de nuestra red cuando le envíamos una orden de desactivación.

Es importante destacar el orden en que se ejecutan las instrucciones, especialmente cuando ejecutamos las instrucciones de desactivación vía SSH. Por cada *chain* (*input, output y forward*), *iptables* cuenta con una lista de reglas, de forma que cada paquete que recibe buscará en la *chain* correspondiente, una regla que coincida con su descripción siguiendo un orden descendente. El parámetro *-I* que usamos en nuestras instrucciones indica que colocaremos la regla al principio de la lista, es decir, será la más prioritaria, por tanto si la primera instrucción que ejecutamos es la rechazar todo el tráfico ya no podríamos enviar la instrucción de escuchar en el puerto UDP y si lo hiciéramos en el orden inverso, la regla de rechazar todo el tráfico quedaría la primera de la lista con lo que ninguna de las dos opciones cumpliría nuestro objetivo.

La solución para ejecutar las intrucciones correctamente es: en primer lugar las que permiten recibir y envíar tráfico UDP por el el puerto 5022 y a continuación las instrucciones de rechazar todo el tráfico con una pequeña modificación; añadir un "2" entre el nombre de la *chain* y la cadena *-j*. De esta forma le indicaremos a *Iptables* que esa regla irá en la posición número 2 de la lista.

Ahora bien, llegado a este punto aparece un problema; en el momento que deseemos volver a activar nuestro router desde el punto de control, ¿cómo podremos llegar hasta ese router si no recibe tráfico del protocolo de encaminamiento?. El problema que se plantea no es trivial, ya que el hecho de no recibir ni

enviar tráfico del protocolo implica que éste desconozca de la existencia de ese nodo en la red, por tanto no podremos localizarlo. Además, estando desactivado, podemos desear cambiar un parametro (sección 4.3) a este nodo, todo ello hará necesario la creación de un nuevo sistema que permita el envío de mensajes de configuración desde el punto de control al nodo o nodos desactivados. La solución que hemos elegido son los sockets *broadcast* y detallaremos todas sus características en el siguiente apartado.

4.2. Sockets Broadcast

Los sockets son un método relativamente sencillo para el intercambio de información entre procesos situados en máquinas diferentes, por tanto, los hemos considerado muy apropiados para nuestra aplicación. El lenguaje que hemos utilizado para su implementación es C (actualmente OpenWRT solamente incluye en su *kit* de desarrollo compiladores cruzados para C y C++). Un *socket* queda definido por tres elementos:

- Dirección IP
- Protocolo
- Número de puerto

Por tanto, debemos escoger la opción adecuada de cada elemento para nuestra aplicación. El protocolo que vamos a emplear es UDP como hemos especificado anteriormente, esto en principio supone un servicio no orientado a conexión y que no permite asegurar que los mensajes que enviemos lleguen a su destino.

A estas alturas el lector es posible que se pregunte por qué no emplear *sockets* que implementen el protocolo TCP; sencillamente, no se puede contemplar el uso de este protocolo para nuestra aplicación, la razón la podemos observar claramente en la figura 4.1:

Figura 4.1: Necesidad de sockets broadcast

Como podemos observar la única forma de acceder al router 2 de nuestra red *mesh* es pasar por el router 1 (recordemos que las líneas simulan la visibilidad

entre nodos), el cual está actualmente desactivado. Por tanto sería imposible realizar un establecimiento de la conexión, propia del protocolo TCP. La única solución por tanto es el envío de paquetes UDP que vayan pasando de nodo en nodo hasta llegar a su destino.

A pesar de usar *sockets* sobre UDP seguimos teniendo un problema, que no es otro que localizar el punto de acceso desactivado, al que deseamos enviar un mensaje, sin ayuda del protocolo de encaminamiento. La solución a esta dificultad no es otra que los *sockets broadcast.*

Cuando nosotros creamos un *socket,* éste va asociado a nuestra dirección IP y, por consiguiente, recibiremos todos los paquetes que tengan como destino dicha dirección, usando el protocolo que hemos elegido y el mismo puerto donde se escuchan las peticiones. Cuando otorgamos a un *socket* la propiedad de *broadcast* nos permite recoger los paquetes que vayan destinados a, no sólo nuestra dirección IP, si no también a los dirigidos a la dirección *broadcast* de nuestra red. De la misma forma cuando creamos un *socket* para enviar un mensaje, si lo configuramos con esta propiedad será recibido por todos los nodos que estén alcanzables por él y tengan la opción *broadcast* también activada.

4.2.1. Control de inundación

Así pues, la idea que surge para solucionar el problema de alcanzar un nodo con una determinada dirección IP sin hacer uso protocolo de encaminamiento consiste en enviar los mensajes a las dirección de *broadcast* de la red. Cada punto de acceso que reciba el mensaje deberá comprobar si el mensaje es para él[1], y en caso negativo reenviarlo a la dirección de *broadcast* de la red y así sucesivamente, de esta forma nos aseguramos que el mensaje pasará por todos sus nodos de la red y por tanto también por nuestro destino.

Evidentemente, este método necesita de un cierto control, de lo contrario no dejaría nunca de generarse mensajes de *broadcast* hasta el punto de saturar la red. Un ejemplo sencillo de este hecho se puede observar en la figura 4.2. La red *ad-hoc* que se aprecia en la imagen está formada por tres nodos: A,B y C. El punto de control envía un mensaje broadcast con el objetivo de alcanzar en dispositivo B (que se supone deshabilitado o no alcanzable). En un primer paso, el mensaje es recibido por los nodos A y C, que al no ser los destinatarios del mensaje lo reenviarán. El mensaje reenvíado por A lo recibirá por un lado el punto de control, que al no tener proceso servidor (punto 4.2.3.1), no reenviará el mensaje. Por otro lado llegará al nodo B que es el destinatario, con lo que ejecutará las instrucciones pertinentes y no volverá a envíar el paquete, sin embargo, también le llegará al nodo C que ya lo había recibido antes. Del mismo modo, el mensaje que recibió C desde el punto de control, al reenviarlo será recibido por el nodo A. En ambos casos ninguno de los dos es el destinatario del mensaje, con lo cual volverían a enviarlo, entrando en un bucle infinito. Este es el ejemplo más sencillo en cuanto a número de nodos se refiere, y concretamente en

[1] Esta información se incluirá en el paquete que se envía, todo ello se explicará en el punto 4.4.3

este escenario el número de mensajes generados no saturaría la red, sin embargo, a medida que el número de nodos creciese el aumento de mensajes aumentariá exponencialmente, colapsando totalmente las comunicaciones de la red.

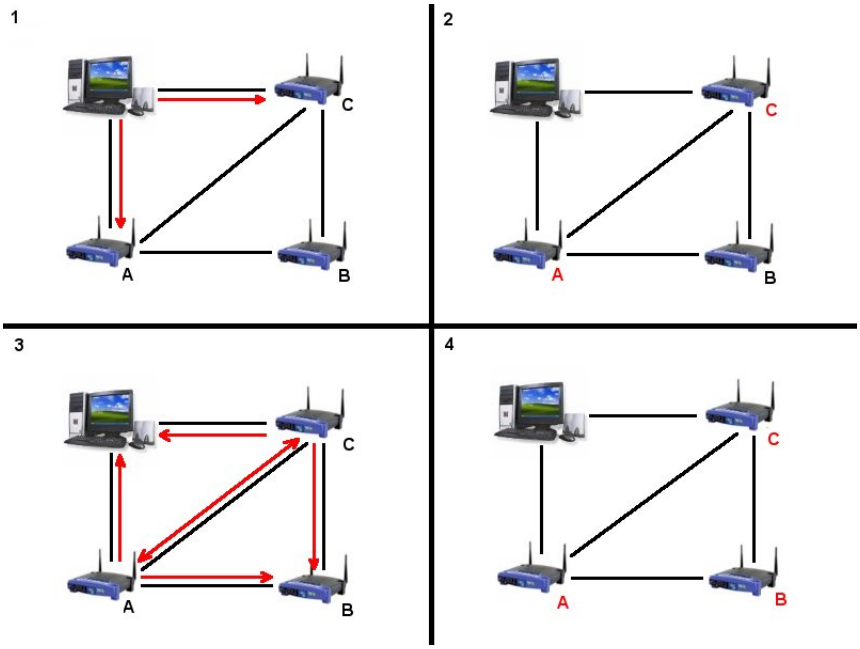

Figura 4.2: Saturación de la red

Para evitar que se produzca la situación que acabamos de comentar la solución es asegurarse de que cada nodo retransmite una sola vez el mismo paquete. Para ello se ha optado por una solución muy común en la implementación de protocolos de encaminamiento; consiste en mantener una tabla en cada *router* que permita controlar si ese paquete ya pasó por ese nodo. A esta tabla la llamaremos a partir de ahora tabla de control y pasamos a detallar su funcionamiento a continuación.

4.2.2. Tabla de control

Cada tabla de control mantenida en un punto de acceso contará con dos columnas: dirección IP origen y número de secuencia. El funcionamiento es muy sencillo:

1. Cuando cualquier dispositivos de la red envía un mensaje a una dirección *broadcast* incluye dentro del paquete que envía existe un número de secuencia, que representará el número de mensaje enviado por ese nodo de la red.

2. Cada *router* que tenga visión directa[2] recibirá el paquete y obtendrá la dirección IP del emisor del mismo. A continuación buscará esa dirección en su tabla.

3. Si no encuentra una entrada para esa dirección en la tabla, añade la dirección asi como el número de secuencia que lleva el paquete.

4. Si encuentra la dirección en la tabla:

   a) Si el número de secuencia asociado esa dirección es menor del que se incluye en el paquete, retransmitimos el paquete y actualizamos el campo del número de secuencia con el del paquete recibido.

   b) Si el número de secuencia asociado a esa dirección es igual o mayor del que posee el paquete, descartamos el paquete puesto que ya ha sido retransmitido por ese nodo.

4.2.3. Procesos

A continuación vamos a detallar los procesos que ha sido necesario implementar para la integración de los *sockets broadcast* en el sistema.

Figura 4.3: Procesos del sistema

4.2.3.1. Proceso servidor

Cada punto de acceso de la red *mesh* contará con un proceso servidor, que se iniciará automáticamente al arrancar el dispositivo. Este proceso se encargará de recibir mensajes, interpretar los campos para realizar los cambios correspondientes en el dispositivo y también se encargará de mantener la tabla de control comentada en el punto anterior. Cuando el proceso servidor necesite enviar

[2]Utilizamos este término para referirnos a los dispositivos que se encuentran dentro de rango de alcanzabilidad del dispositivo emisor.

un mensaje, bien un paquete que no es para él y debe reenviar o bien un reconocimiento, invocará al proceso cliente con los argumentos necesarios para la construcción del paquete y su envío.

4.2.3.2. Proceso cliente

Este proceso se hallará también en todos los puntos de acceso que forman la red y será invocado por el proceso servidor, usando los argumentos con los que fue invocado para rellenar los campos del mensaje que enviará. Una vez enviado el paquete, el proceso cliente termina su ejecución.

4.2.3.3. Proceso cliente CP

Este es el proceso que se va a ejecutar en el punto de control, es bastante parecido al proceso cliente, pero posee características añadidas destacables. Antes de pasar explicar la diferencia respecto al proceso cliente de los puntos de acceso hay que tener en cuenta una característica importante que no hemos comentado. Como hemos mencionado anteriormente, estamos empleando *sockets* que implementan en protocolo UDP, puesto que, además no teníamos elección. Pero esto implica que no podremos estar seguros de que nuestras instrucciones hayan llegado al destino, algo que como veremos luego es muy importante, ya que el administrador puede realizar un cambio de parámetro simultáneo en varios routers y debe saber cuáles han recibido el mensaje y cuáles no para reintentar el cambio o tomar las medidas oportunas. Esto nos ha llevado a crear un sistema de reconocimientos, de modo que cuando un proceso servidor recibe un mensaje que está destinado para él, enviará un reconocimiento al punto de control que lo envió.

Así pues, la diferencia básica de este proceso cliente respecto al anterior es que una vez envía el mensaje queda a la espera de recibir el reconocimiento. Realiza hasta tres reintentos (parámetro configurable por el usuario) en caso de no recibir ningún reconocimiento y pasado el número de reintentos o al recibir un reconocimiento, el proceso informa a la aplicación (código PHP) si la operación ha tenido éxito o no y termina su ejecución.

4.3. Cambio de parámetros

Esta es, sin duda, una de las funcionalidades más importantes de la aplicación. Por un lado nos va a permitir cambiar una serie de parámetros del punto de acceso sin tener que acceder a la página *web* del dispositivo para buscar el parámetro, cambiarlo y actualizar los cambios. Pero además, lo que nos va a permitir es cambiar un mismo parámetro en diversos dispositivos sin importar que estén habilitados o deshabilitados.

El método por defecto para efectuar los cambios es mediante conexión SSH para los dispositivos activados y mediante mensajes UDP para los desactivados aunque enseguida comprobaremos que no siempre es así.

4.3.1. Parámetros globales

De entre todos los parámetros que actualmente la aplicación ofrece hay algunos que tienen una restricción, y ésta es que que el cambio debe aplicarse a todos los dispositivos de la red, a estos parámetros se les denomina parámetros globales. Esta restricción reside en la naturaleza de algunos parámetros, por ejemplo el ESSID o identificador de la red. Si cambiaramos este parámetro en unos puntos de acceso y en otros no, la red quedaría divida, además de que perderíamos la posibilidad de administrar aquellos que han quedado en la red con el ESSID antigüo. Repetimos que el cambio se realizará en todos los dispositivos de la red, esto es el punto de control y los puntos de acceso, incluidos los desactivados. La razón de estos últimos es obvia: si en futuro los queremos volver a activar deben pertenecer a nuestra red (tomando el ejemplo del ESSID).

Cómo se explica en la parte de interfaz de usuario (capítulo 5), cuando seleccionamos un parámetro de este tipo, la aplicación, obliga al usuario a seleccionar todos los dispositivos de la lista. Destacamos este hecho porque es posible que en la lista de nodos configurables no se encuentren la totalidad de los nodos que componen la red, en ese caso "perderíamos" en el cambio a los dispositivos no añadidos.

4.3.2. Parámetros configurables

Los parámetros que actualmente la aplicación ofrece para ser modificados son los que hemos considerado más útiles, sin embargo, añadir más parámetros a la oferta actual es relativamente sencillo y será explicado en el apéndice B. Los parámetros disponibles son:

- Nombre: El nombre del dispositivo.
- ESSID: Parámetro global que identifica la red.
- Canal: Parámetro global que indica el rango de frecuencias en el que transmitirán los *routers*.
- WEP: Parámetro global de cifrado para dispositivos en modo *ad-hoc*.
- Password: Este parámetro sólo podrá ser cambiado a *routers* activados por motivos de seguridad.

Dependiendo del tipo de parámetro que deseemos cambiar, deberemos realizar una acción diferente para actualizar los cambios en el dispositivo. En nuestro sistema podemos encontrar dos casos distintos:

- El primer caso es el más simple, y deberá ser realizado para cualquier parámetro. Cuando se haya modificado el parámetro correspondiente, se deberá guardar su nuevo valor en la memoria no volátil del *router* mediante el comando *"nvram commit"*. Un ejemplo de este caso sería por ejemplo el nombre identificador del dispositivo.

Canal	Frecuencia(MHz)	FCC	IC	ETSI	España	Francia	MKK
1	2412	X	X	X	X	X	X
2	2417	X	X	X	X	X	X
3	2422	X	X	X	X	X	X
4	2427	X	X	X	X	X	X
5	2432	X	X	X	X	X	X
6	2437	X	X	X	X	X	X
7	2442	X	X	X	X	X	X
8	2447	X	X	X	X	X	X
9	2452	X	X	X	X	X	X
10	2457	X	X	X	X	X	X
11	2462	X	X	X	X	X	X
12	2467			X		X	X
13	2472			X		X	X
14	2477						X

Cuadro 4.1: Tabla de frecuencias *Wireless*

- El segundo caso se aplica a todos los parámetros que modifican algún aspecto de nuestra configuración de red inalámbrica. Dentro de este grupo se hallarían el canal, ESSID o clave WEP. En su caso, debemos ejecutar despúes de haber guardado los nuevos valores en la NVRAM, *"/sbin/wifi"* para reiniciar la interfaz Wi-Fi del dispositivo con los nuevos valores.

4.3.2.1. Canal

Un parámetro "especial", en cuanto a la forma de ser actualizado, en nuestro sistema es el canal. El canal representa a la frecuencia que va a trabajar el dispositivo para comunicarse con el resto. En la tabla 4.1 podemos observar los canales regulados actualmente los diferentes territorios.

El problema radica en que es más prioritario el ESSID que el canal a la hora de establecer una red inalámbrica. Esto quiere decir que el dispositivo que crea la red la configura con su ESSID y su canal; si acontinuación otro dispositivo deseara unirse a la red, aunque actualmente tenga otra frecuencia de canal configurada, al unirse a la red anterior se le asignará el valor de canal correspondiente a la red por razones obvias. Esto también implica que mientras forme parte de esa red no podremos hacerle cambiar de canal.

De esta forma, en nuestro sistema, hemos utilizado un método diferente para cambiar este parámetro, basándonos en la idea de que el primer nodo que forma la red es el que impone el canal. Por tanto, el método que seguimos para realizar el cambio es:

1. Modificar en los dispositivos el valor del canal
2. Actualizar los valores en la NVRAM y reiniciar el dispositivo

3. Cambiar el canal al punto de control

De esta forma, al estar, todos los dispositivos en proceso de reinicio, el punto de control que se haya provisionalmente solo en la red, podrá cambiar la fracuencia asociada al dicha red y conforme se incorporen los dispostivos lo harán la nueva frecuencia ya establecida.

4.3.3. Realización de los cambios

Cuando queremos realizar la modificación de un parámetro en un sólo dispositivo, el procedimiento es sencillo: sólo necesitamos saber si está habilitado o deshabilitado y proceder al cambio usando una conexión SSH o bien por un mensaje UDP. Ahora bien, cuando queremos realizar un cambio a varios nodos de la red y muy especialmente cuando queremos cambiar un parámetro global el procedimiento es más complicado. Para comprenderlo mejor supongamos un escenario como el que aparece en la figura 4.4 dónde la lineas lineas entre los puntos de acceso representan la alcanzabilidad entre ellos.

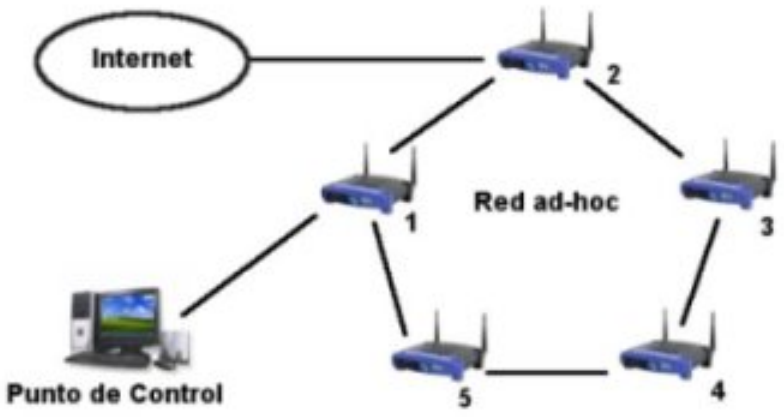

Figura 4.4: Escenario ejemplo red mesh

Como podemos observar el punto de control se une a la parte *ad-hoc* de la red *mesh* a través del punto de acceso 1 y el punto de acceso 2 es que el proporciona salida a internet.

Cuando desde el punto de control intentemos enviar datos a un nodo de la red, el protocolo de encaminamiento nos ofrecerá un camino, en teoría el mejor posible. Así pues, supongamos que queremos establecer una conexión SSH con el punto de acceso número 4, posiblemente el protocolo de encaminamiento dirija los datos desde el 1 hasta el 5 y del 5 hasta 4. Ahora supongamos que desactivamos el nodo 5, lo que supondrá a efectos de encaminamiento que se encuentra fuera de la red. En este caso, el protocolo intentará encontrar un camino alternativo para llegar a ese destino, que en nuestro caso existe y estaría formado

por los nodos 1,3 y, finalmente, el 4. Por último imaginemos que desactivamos el nodo 3.

Llegado a este punto, nos encontramos con un dispositivo (el 4) que aún estando activado, no podemos acceder a él por una conexión SSH, habitual en este caso, para cambiar los parámetros debido a que no tenemos un camino para llegar hasta él. Este caso supone una excepción en la cual debemos proceder, a la hora de enviar mensajes al dispositivo, como si se tratará de uno desactivado, es decir, mediante mensajes UDP.

Es posible que el lector se pregunte cómo se ha podido llegar a una situación o que utilidad tiene tener un *router* activo rodeado de inactivos. En ejemplo de la figura 4.4, evidentemente, no tiene ninguna a parte de proporcionar robustez a la hora de permitir al usuario de configurar la red libremente. Quizá su intención fuera la de desactivar también el nodo 4, pero no siguió el orden más adecuado. Sin embargo, vamos a suponer otro escenario, como el de la figura 4.5 en el que se puede observar un posible caso en el que nos interese llegar a esa situación.

Figura 4.5: Ejemplo real de una red mesh

Como vemos en la imagen, tenemos una red mesh formada por 8 puntos de acceso que componen la parte *ad-hoc* y el nodo 1 proporciona acceso a internet. Supongamos que la red pertenence una empresa y que cada uno de las dos ramas que salen del nodo 1 pertenece a un edificio. El nodo 1 proporciona acceso a internet al nodo 4 y 5 a través del 3 y al 6,7 y 8 a través del 2.

Puede resultar interesante, por motivos de seguridad, optimización del ancho de banda o necesidades temporales, desactivar el nodo 2 (o el nodo 3) de forma que sólo se permita el intercambio de información entre dispositivos de una misma área y que además sólo una de ellas tenga acceso a internet. Este es un caso caso realmente útil de *routers* activados detrás de *routers* desactivados. Además se aprecian otras posibilidades interesantes como desactivar el nodo 1, esto permitiría a todos los clientes de la red *mesh* de la empresa intercambiar información entre ellos, sin embargo no se permitirá la salida ni la entrada de

tráfico desde/hacia internet.

4.3.4. Orden de realización de cambios

Una vez hemos observado los posibles casos que se pueden dar a la hora de realizar un cambio de parámetro a más de un dispositivo vamos a detallar el orden que hay que seguir para que se lleven a cabo correctamente. Efectivamente, el orden de cambio de los parámetros es vital para el correcto funcionamiento de la aplicación. Un ejemplo sencillo, sería el de la figura 4.5, si quisieramos cambiar el identificador de la red (ESSID) a los nodos de nuestra red y comenzasemos por cambiar ese parámetro en los nodos 2 y 3, que son los más cercanos, cuando intentásemos enviar un mensaje a otro nodo nos encontraríamos que no tenemos que ningún nodo de nuestra red (con nuestro identificador) que nos permita alcanzar el objetivo, de hecho desde la ubicación física que se encuentra el punto de control, no tendría la posibilidad de comunicarse con nadie.

Esto nos da una pista sobre la solución; deberemos realizar los cambios desde los nodos más lejanos a los más cercanos, y por último cambiar el parámetro en el propio punto de control. Sólo nos quedaría saber cómo puede la aplicación conocer que *routers* se encuentran más alejados desde el punto de control en el que se está ejecutando.

Esta "distancia" la mediremos mendiante el TTL[3] que obtenemos del comando *ping*, de esta forma podremos obtener la distancia en saltos hasta el dispositivo y por tanto comenzar a realizar los cambios por el de menor TTL. Por supuesto habrá que almacenar esta información en la base de datos y mantenerla actualizada, ello se explicará en el punto siguiente.

4.3.5. Procedimiento general

De esta forma el procedimiento general de cambio de parámetros para varios routers queda resumido en los siguientes pasos:

1. Al realizar la operación de añadir un punto de acceso se obtendrá su TTL, mediante el comando *ping*, este valor será almacenado en la base de datos asociado a la dirección IP del dispositivo. En caso de no recibir respuesta al mensaje enviado se le asignará un valor de TTL=0. Se deduce que tendremos dos casos en los que se produzca este hecho: en caso de ser un dispositivo deshabilitado o un dispositivo habilitado pero no alcanzable al tener otros desactivados en su camino.

2. Ante la operación de realizar un cambio en varios dispositivos se comprobarán sus direcciones IP en la base de datos y se generaran dos listas: puntos de acceso con TTL=0 y puntos de acceso TTL<>0.

3. En primer lugar se realizarán los cambios en los *routers* TTL=0, esto implicará los dos casos mencionados en el punto 1, mediante en envío de

[3]Time To Live: nos indica el número de "saltos" que puede dar el mensaje por la red antes de ser descartado.

mensajes UDP. Por tanto no habrá que seguir ningún orden especial a la hora de llevar a cabo las modificaciones.

4. A continuación se realizará el procedimiento equivalente para los routers activados y alcanzables, a los que tendremos asociados un TTL<>0. Para ello ordenaremos la lista de dispositivos generada en el punto 2 de menor a mayor TTL, y efectuaremos los cambios en ese orden.

Para que este procedimiento funcione de forma correcta, un requisito indespensable es que los TTL de los dispositivos se mantengan actualizados en la base de datos, por este hecho, se actualizarán todos los TTL's cuando se produzca una habilitación/deshabilitación de un punto de acceso, con el objetivo de descubrir que nuevos caminos se han abierto/cerrado entre el punto de control y cada uno de los nodos de la red.

4.4. Seguridad

Hoy en día, este es un aspecto muy importante en el cualquier aplicación distribuida. En nuestro sistema esta característica va a ser importante, sobre todo, en asegurarnos que los mensajes los está enviando realmente el punto de control.

Por un lado, en lo que a los puntos de acceso habilitados se refiere, los cambios se realizan mediante conexión SSH lo que implica seguridad a la hora de enviar datos entre un dispositivo y otro. Sin embargo, el problema principal reside en los mensajes UDP que envía el punto de control. Estos mensajes, en caso de viajar sin cifrar por la red, pueden ser interceptados por un usuario mal intencionado que se encuentre en ese momento formando parte de la red *ad-hoc*. Evidentemente, tendremos que hacer uso de algún mecanismo que garantice a los puntos de acceso que el mensaje ha sido realmente enviado por el punto de control y lo conseguiremos mendiante los sistemas de cifrado de clave pública.

4.4.1. Cifrado de clave pública

El funcionamiento de este mecanismo de cifrado consiste en la generación de un par de claves por parte de cada dispositivo: una clave pública y otra privada. La clave privada se mantendrá en secreto y sólo el propio dispositivo la conocerá, mientras que la clave pública puede ser distribuida libremente. Estas claves están matemáticamente relacionadas, pero el coste de averiguar la clave privada a partir de la pública es computacionalmente intratable. Un mensaje cifrado con una clave pública solamente puede ser recuperado con la correspondiente clave privada.

Dentro de los sistemas cifrado de clave pública podemos encontrar dos grupos bien diferenciados:

- Cifrado basado en clave pública: Este tipo es el más común y consiste en cifrar un mensaje con la clave pública del usuario destino de forma que

sólo él, con su clave privada pueda leer el contenido del mensaje. Este método se usa para garantizar la confidencialidad de la información que se está enviando.

- Firma digital: En este caso el mensaje es cifrado con la clave privada del emisor de forma que, el resto de usuarios, al tener acceso a su clave pública, pueden leer el mensaje, incluso usuarios con malas intenciones. Ahora bien, esto nos garantiza auntenticidad pues nos aseguramos que sólo el poseedor de esa clave privada ha podido ser el autor del mensaje y que además este no ha podido ser modificado.

A la vista de las propiedades de ambos métodos, el que resulta más adecuado para nuestra aplicación es el de firma digital, ya que no es tan importante el hecho de que puedan interceptar nuestros mensajes como el hecho de asegurarnos que realmente los está enviando un punto de control.

4.4.1.1. RSA

Hay diversos algoritmos para la implementación del método de firma digital, nosotros hemos optado por el algoritmo RSA[8] debido a que las librerías que emplea se incluyen en el propio OpenSSL y son sencillas de manejar usando código C.

Las características más importantes de este método de cifrado son:

- El sistema criptográfico con clave pública RSA es un algoritmo asimétrico cifrador de bloques, que utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario.
- Los mensajes enviados usando el algoritmo RSA se representan mediante números y el funcionamiento se basa en el producto de dos números primos grandes (mayores que 10100) elegidos al azar para conformar la clave de descifrado
- Emplea expresiones exponenciales en aritmética modular
- La seguridad de este algoritmo radica en que no hay maneras rápidas conocidas de factorizar un número grande en sus factores primos utilizando computadoras tradicionales, aunque la computación cuántica podría proveer una solución a este problema de factorización

4.4.2. HIT

Quizá este término le resulte familiar al lector de este documento. HIT (*Host Identity Tag*) es un término que utiliza HIP (*Host Identity Protocol*) [9] para identificar de forma unívoca a un equipo. Nuestro concepto de HIT no se ajusta al estándar pues no nos es necesario para conseguir nuestro objetivo; simplemente aplicaremos el algoritmo SHA-1 sobre la clave pública del equipo para normalizar su tamaño a 20 *bytes*.

Por motivos de comodidad y para evitar posibles fallos inesperados, cabe decir que el HIT está generado en formato hexadecimal con el objetivo de evitar que se generen cadenas con caracteres como '\n' o '\0' que puedan dar problemas a la hora de ser tratadas por la aplicación o al ser insertadas en la base de datos.

De esta forma cuando el punto de control quiera enviar un mensaje a un determinado nodo de la red, no la enviará a la dirección IP de dicho nodo si no a su HIT. El objetivo principal es el de proveer transparencia al usuario de cara a un punto de acceso pueda ser reiniciado inesperadamente (por ejemplo un corte en el suministro eléctrico) y le sea asignada una dirección IP distinta.

4.4.3. Formato del mensaje

Tras observar todas las necesidades que presentaba nuestra aplicación pasamos a especificar el formato de los mensajes que el punto de control enviará con el fin enviar instrucciones a los dispositivos de la red que a su vez tiene el mismo formato que los mensajes de reconocimiento enviados por el destino.

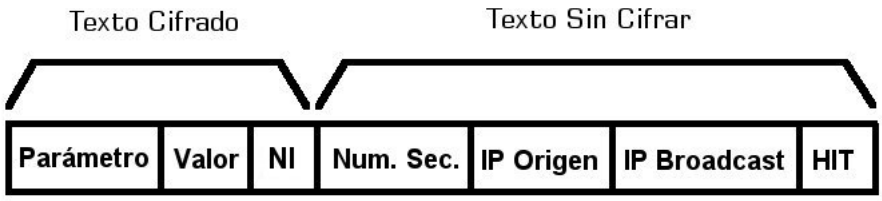

Figura 4.6: Formato del mensaje

En primer lugar vamos a explicar uno por uno cada campo del mensaje:

- Parámetro: Representa el parámetro que vamos a cambiar (ESSID, Canal, ...), en el caso que sea un mensaje para desactivar o activar un *router* este campo contendrá el valor *disable*. Cuando el mensaje llegue al destino, éste usará el valor del campo para ejecutar un *script* u otro.
- Valor: Este será el nuevo valor para el parámetro que hemos indicado en el campo anterior, en caso que sea un mensaje de activación/desactivación este campo contendrá el valor no y *yes* repectivamente.
- ID: Representa el número de instrucción, es decir, un identificador único generado por el punto de control para cada mensaje de configuración que envía. En la base de datos se almacenará una tabla con un histórico de todas las instrucciones enviadas por el punto de control cuya clave principal será dicho número.

- Num. Sec: Este campo representa el número de mensaje que esta enviando bien el punto de control, bien un determinado punto de acceso. Este valor comenzará tomando el valor cero y se irá incrementando en uno por cada mensaje que envíe. Aunque este número será siempre creciente, no tendrán porque ser número correlativos, puesto que en caso de realizar algún reintento por algún mensaje que no haya llegado a su destino incrementará su valor, por tanto, empleará varios números de secuencia para una misma instrucción.

- IP Origen: Como su nombre indica, contendrá la dirección IP que envía el mensaje. Esta información no la podemos obtener de las cabeceras puesto que la dirección IP origen irá cambiando conforme la reenvíe cada *router* que se encuentre en el camino entre el emisor y el receptor del mismo, y además es necesario conocerla en el caso los dispositivos empotrados para conocer la dirección a la cual enviar el reconocimiento y por el punto de control para distinguir quién le envía ese reconocimiento.

- IP Broadcast: Esta es la dirección a la que realmente se envían todos los mensajes. Cuando nosotros envíamos un mensaje a través de un *socket*, debemos especificar la dirección a la cual queremos enviarlo. Como a nuestros *sockets* les hemos añadido la opción de enviar tráfico *broadcast* podemos usar una dirección de este tipo. Quizá este parámetro no es necesario que viaje en todos los paquetes, ya que podría ser un parámetro configurable en cada proceso servidor de los dispositivos, pero el realizar un cambio en la dirección de red, del tipo de dirección IP o de la máscara, supondría la recompilación de los ficheros fuente de los procesos y su posterior actualización en los puntos de acceso sería algo costoso.

- HIT: Su significado fue explicado en el apartado anterior y realizaría la función de IP destino aportando las ventajas que proporciona esta nueva idea de identificador.

Como podemos observar en la figura 4.6, posiblemente lo que primero llame la atención al lector es el envio de un mensaje con una parte cifrada y otra sin cifrar, esto, por supuesto, tiene su justificación.

Vamos a suponer que la totalidad del mensaje viajara cifrada, cada *punto de acceso* que recibiera un mensaje debería desencriptarlo y comprobar si es o no para él, en caso negativo debería volver a cifrarlo y reenviarlo lo que supondría un pequeño coste extra. Sin embargo, ese no es el problema principal, la dificultad se halla en que el paquete debe ir cifrado con la clave privada del punto de control para asegurarnos que es él quién lo envía y sólo él conoce su clave privada. Así pues el HIT (que identifica el destinatario del mensaje) debe viajar fuera del texto cifrado.

Una vez que se hace necesario dividir el mensaje en estas dos partes, nos planteamos dónde conviene que viaje cada uno de los parámetros y optamos por que los parámetros que vayan a ser usado tanto por el destino como por todos los nodos intermedios viajen en la parte no cifrada para mejorar la eficiencia. Estos

parámetros son la dirección IP origen y el número de secuencia que nos servirán para mantener la tabla que controla el envío masivo de mensaje *broadcast* y la dirección IP de *broadcast* de la red que será el destino del paquete en cada salto de éste por la red.

Por último mantenemos en la parte cifrada los parámetros que realmente queremos proteger de ser modificados y que pueden dañar nuestro sistema como es el parametro que queremos cambiar, su nuevo valor y el ID.

Aunque lo hemos mencionado en varias ocasiones, no hemos hablado todavía de los reconocimientos. En la versión actual del proyecto, el formato del mensaje de reconocimiento es exactamente igual que el mensaje de control de la figura 4.4.3. El punto de acceso, deberá cambiar la ip origen, que ahora será la suya, el HIT que será el del punto de control y establecer el número de secuencia que le corresponda, por lo demás, los campos del paquete, seguirán teniendo los mismos valores. El enviar de nuevo un mensaje con tantos campos para un reconocimiento parece no ser del todo eficiente, sin embargo, ninguno de los campos del texto sin cifrar puede ser suprimido. La única posibilidad podría ser suprimir el parámetro y el valor de la parte cifrada, puesto que con el indentificador tendríamos suficiente, pero las funciones de cifrado de RSA devuelven el mensaje cifrado en 256 *bytes*, independientemente del tamaño del texto a cifrar con lo que no ganaríamos nada.

4.5. Funcionamiento general

Este apartado lo que pretende es detallar todos los pasos internos que realiza la aplicación, desde el punto de vista de la seguridad y el HIT, cuando el usuario pretende realizar un cambio de un parámetro.

1. Un usario selecciona de la lista de parámetros el que desea cambiar y marca en qué router quiere que se realice ese cambio, lo primero que hace la aplicación es consultar en su base de datos el HIT asociado a la IP del router seleccionado.

2. Una vez obtenido el HIT, se invocará al proceso clienteCP (apartado 4.2.3.3) se construirá el paquete como se muestra en el apartado 4.4.3, con una parte en texto plano y otra parte cifrada con la clave privada del punto de control. Este mismo proceso esperará a recibir un mensaje de reconocimiento, en caso de no recibirlo en un tiempo de 1 segundo, realizará hasta 3 reintentos[4] antes de desistir.

3. El proceso servidor (apartado 4.2.3.1) del punto de acceso recibirá el mensaje, y procederá a descifrarlo usando la clave pública del punto de control, almacenada en el archivo "CPpub.key". Una vez descrifado, ejecutará las intrucciones necesarias para realizar los cambios.

[4]Tanto el tiempo de espera como el número de reintentos son parámetros configurables.

4. A continuación el proceso servidor invocará al cliente (apartado 4.2.3.2) con lo valores de campo necesario, entre ellos el HIT del punto de control que se encuentra en el archivo "CPHIT" y la parte cifrada con la clave privada del punto de acceso.

5. Cuando llegue el mensaje al punto de control, éste abrirá el archivo que lleva el mismo nombre del HIT del punto de acceso al que envío el mensaje y que obtuvo en el punto 1, en ese fichero se encuentra la clave pública con el que deberá descifrar el mensaje y verificar que es el reconocimiento que esperaba.

El caso para un cambio de parámetro a varios dispositivos es repetir todos los pasos para cada uno de ellos, además haría falta un paso inicial, que sería el ya mencionado de ordenar los puntos de acceso por sus TTL's y en función de si están habilitados o deshabilitados como se explica en el apartado 4.3.4.

4.6. Manejo de la lista de dispositivos

La lista de routers que disponemos en la aplicación se relaciona directamente con la base de datos del punto de control. En ella, se almacenará toda la información necesaria sobre los routers. Actualmente, está base de datos cuenta con tan sólo cinco campos como se puede observar en el cuadro 4.2.

Campo	Tipo	Significado
IP	VARCHAR(15)	Dirección IP del dispositivo
Active	INT(1)	Indica el estado del dispositivo
Password	VARCHAR(25)	El password de ese dispositivo en concreto
TTL	INT(2)	Número de saltos hasta el dispositivo
HIT	VARCHAR(40)	El identificado único de ese dispositivo

Cuadro 4.2: Representación routers en la base de datos

En principio esta tabla estará vacía, y se irá rellenando tal a medida que insertemos los *routers*.

Debe quedar claro que lo que la lista representa no son todos los nodos que forman una determinada red *mesh,* si no que recoge todos los dispositivos que el usuario a añadido y que por tánto podrá configurar. Cada *router* de la lista tendrá un *link* a la su página de configuración el en dispositivo, de modo que al pulsar el enlace se abrirá en el frame derecho del navegador. Todo se podrá ver de forma más clara en el apartado de interfaz de la herramienta (capítulo 5).

4.6.1. Añadir punto de acceso

La primera operación que podemos realizar sobre esta lista consiste en añadir un elemento, en este caso un *router* identificado por su dirección IP, a la lista de dispositivos manejados, aunque en nuestro caso va a implicar muchas más cosas.

A continuación vamos a detellar todo el proceso que lleva a cabo la aplicación para añadir un dispositivo:

1. En primer lugar, a partir de la dirección IP proporcionada por el usuario se comprobará que el *router* está habilitado y es alcanzable. Para ello, se empleará el comando *ping* que además nos proporcionará el TTL, dato importante a la hora de realizar el cambio de parámetros explicado en el punto 4.3.4 y que se guardará en la base de datos.

   a) En caso de que no se obtenga respuesta al enviar los mensajes ICMP, se mostrará un mensaje al usuario indicando que el punto de acceso no responde o no existe un dispositivo con la dirección IP indicada.

   b) En caso de recibir respuesta, continuaremos en el paso 2.

2. En este punto se producirá el intercambio de claves públicas y HIT's por parte de los dos dispositivos en cuestión. Mediante una conexión segura como es SSH (para ello se usará el password por defecto definido en el punto B.2.1), depositaremos nuestro HIT y nuestra clave pública (ambos generados previamente) en el directorio */MAYA/server/* bajo los nombres de CPHIT y CPpub.key respectivamente. Del mismo modo obtendremos del mismo directorio la clave pública y el HIT, que contiene el identificador del punto de acceso y además de guardarlo en su información asociada en la base de datos, servirá de nombre de archivo para la clave pública del mismo, de modo que podamos identificar de que nodo es cada clave.

3. Así pues ya tenemos toda la información referente al punto de acceso que hemos añadido, por tanto hacemos una inserción en la tabla de la base de datos.

4.6.2. Añadir lista de puntos de acceso

Esta opción proporciona al administrador la posibilidad de añadir una lista de direcciones IP desde un fichero. Esto nos puede permitir manejar varias redes *mesh* de forma sencilla, puesto que podemos disponer de un archivo de direcciones para cada red y cargar el fichero correspondiente cuando vayamos a administrarla. Puesto que ésa es la idea original, cuando añadimos una lista de direcciones, todas las que teniamos en la lista, que se encontraban en la base de datos, se borrarán.

Del mismo modo que en el opción anterior, se realizará la comprobación de la actividad del dispositivo mediante el comando *ping* y se producirá el intercambio de ficheros, es decir, el mismo proceso del punto anterior se repetirá tantas veces como direcciones IP aparezcan en el archivo. Es probable que unos dispositivos estén activos y otros no, por ello, al final de la inserción, la aplicación mostrará qué dispositivos (si es que los hay) no han podido ser añadidos a la lista.

Campo	Tipo	Significado
ID	INT(8)	Identifica el número de instrucción envíado
parameter	VARCHAR(25)	El parámetro que queremos modificar
value	VARCHAR(30)	El valor del parámetro
HIT	VARCHAR(40)	HIT del dispositivo destino
ACK	INT(1)	Indica si se ha recibido reconocimiento o no

Cuadro 4.3: Tabla de *log* en la base de datos

4.6.3. Eliminar punto de acceso

Esta funcionalidad es trivial dado que no implica ningún tipo de operación asociado a parte del correspondiente borrado de la línea de la base de datos.

4.7. *Log*

Todos los mensajes de configuración que envíe el punto de control quedará registrados a modo de *log* en la base de datos, además quedará indicado si el mensaje fue envíado con éxito o sin embargo no se recibio del reconocimiento. En la figura 4.3 se muestran los campos asociados a esta tabla en la base de datos y la información que representan.

Por tanto, se realizará una inserción en la base de datos por cada mensaje UDP enviado o cada conexión SSH establecida con el dispositivo realizar algún cambio de parámetros, actualizando posteriormente el campo ACK en función de si recibimos confirmación por parte del dispositivo o no.

Capítulo 5

Interfaz de la herramienta. Ejemplos de uso

En este capítulo vamos a explicar y mostrar imágenes de las diferentes funcionalidades de la herramienta con el objetivo de familiarizarnos con su interfaz. Además servirá como una pequeña guía para que el usuarios aprendan su funcionamiento.

5.1. Interfaz inicial

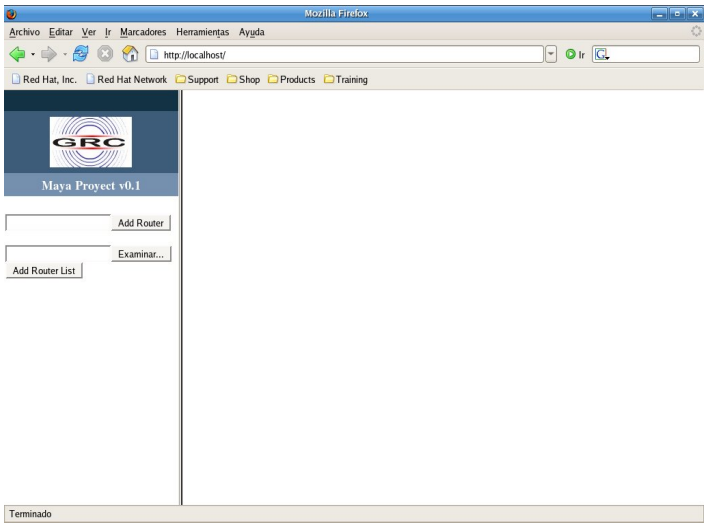

Figura 5.1: Interfaz inicial

Lo que podemos es observar en la imagen 5.1 es la interfaz de la aplicación,

con la que se encontrará el administrador del sistema cuando acceda en el explorador al directorio donde guarda el *index.html* de la aplicación.

En lo que primero que nos fijamos es que la página mostrada se encuentra dividida en dos *frames*, con el objetivo de integrar nuestra aplicación con la interfaz de configuración del propio dispositivo. Esto lo hace mucho más amigable al usuario y siempre le proporciona la posibilidad de realizar los cambios de la forma tradicional, accediendo a la página de configuración del dispositivo, o con las posibilidades que ofrece nuestra herramienta. Como vemos, incluso los colores que se han utilizado han sido elegidos cuidadosamente para que no se aprecie apenas diferencia entre las dos interfaces.

Recordemos que está adaptada para la versión de OpenWRT WhiteRussian RC6 y que uno de los principios básicos de la aplicación es proporcionar una funcionalidad añadida adaptada para redes *mesh*, de la que no dispondríamos con OpenWRT únicamente.

Se aprecia que en el *frame* de la izquierda, dónde se ubica Maya, inicialmente no aparece ningún dispositivo en la lista, por ello, sólo se muestran las opciones de añadir un dispositivos o una lista de ellos.

5.2. Añadiendo dispositivos

Por tanto las dos opciones son añadir de forma indivual o leer desde archivo las direcciones IP. En caso de optar por la primera opción, introduciremos la dirección en el primer cuadro de texto y pulsaremos *Add Router* (Imagen 5.2). Este proceso comportará un intercambio de ficheros (clave pública, HIT, etc...) explicado en el capítulo anterior, por lo que el usuario apreciará que la operación no es instántanea. En caso de que no se localice el dispositivo, la herramienta no nos dejará añadirlo a lista y nos mostrará un mensaje indicando que no ha sido posible establecer una comunicación con dicha dirección IP (Imagen 5.3).

Si por el contrario, decidimos insertar un conjunto de dispositivos leyendo direcciones desde un fichero, debemos de tener en cuenta que se repetirá el proceso anterior por tantos elementos como haya en la lista, por lo que, el retardo de respuesta de la interfaz será proporcional. Para ello deberemos indicar la ruta dónde se encuentra el archivo y pulsar *Add Router List*, como se muestra en la imagen 5.4. Del mismo que en el caso anterior, obtendremos como resultado todo el conjunto de dispositivos añadidos satisfactoriamente en la lista de dispositivos actuales y mostraremos un mensaje informativo con las direcciones IP (Imagen 5.5).

5.3. Interfaz completa

Una vez hemos añadido varios dispositivos al sistema podremos tener una situación como la de la imagen 5.6. En ella podemos observar la lista de todos los añadidos hasta el momento, como se puede apreciar cada dirección IP de la lista posee un *link* a la página de configuración del dispositivo que se desplegará

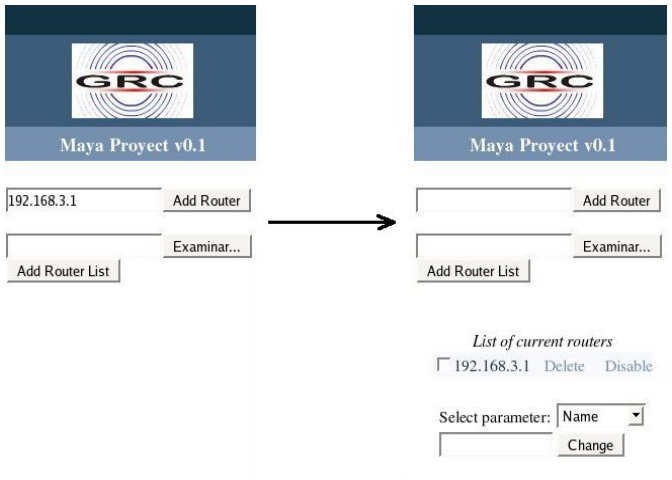

Figura 5.2: Dispositivo añadido satisfactoriamente

en el *frame* de la derecha. También cada *router* añadido posee un botón para ser activado o desactivado, en la imagen todos se encuentran activados por ello sólo tienen la opción de desactivarse. Por último, ahora que hay elementos en la lista, se muestra el cuadro de cambio de parametros que mostraremos más adelante.

5.4. Eliminando dispositivos

Para eliminar de la lista cualquier *router* que deseemos, tan sólo deberemos pulsar el bóton de *Delete* en la línea correspondiente, como se ve en la figura 5.7.

5.5. Cambio de parámetros

Hay dos tipo de parámetros, como se explico en la sección 4.3 del documento, los globales y los no globales. En primer lugar, vamos a centrarnos en los segundos; en este tipo de parámetros podremos seleccionar tantos dispositivos como el usuario desee, en el ejemplo de la imagen 6.7 se ha procedido al cambio de *password* en dos de los cuatro *routers* del sistema.

5.5.1. Restricciones

En este punto nos encontramos con los parámetros globales. Este tipo de parámetros tiene la restricción de tener que ser aplicado a todos los dispositivos que tenemos en la lista, de esta forma no se va a permitir al usuario que quite la selección de un *checkbox*. Esto sobre todo, se realiza como medida preventiva,

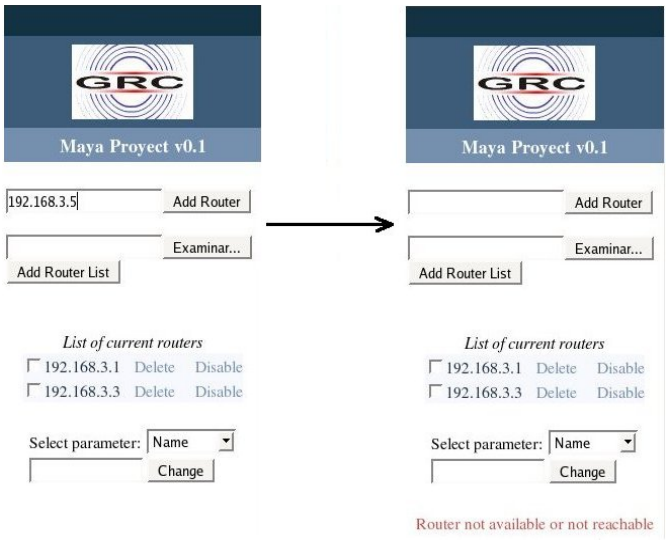

Figura 5.3: Error al añadir dispositivo

para evitar que el usuario cometa el error de dejar un dispositivo fuera de selección y por comodidad. En caso de que el usuario desee excluir de la operación a algún dispositivo puede hacerlo eliminándolo de la lista, de cualquier modo se mostrará un mensaje informando al usuario.

En la figura 5.9 se puede observar un ejemplo del mensaje que se le muestra al usuario y la deshabilitación de los *checkboxes* cuando seleccionamos un parámetro global como es el ESSID.

Del mismo modo también se mostrará mensajes informativos al usuario cuando introduzca valores erróneos para algún parámetro. Por ejemplo, a la hora de elegir un nuevo canal, sólo se le permitirá selecciona un valor entre 1 y 13 o a la hora de introducir una clave WEP, deberá respetar la longitud necesaria de éstas (de 10 o 26 caracteres hexadecimales) asi como utilizar únicamente símbolos del lenguaje hexadecimal (al igual que exige la interfaz de OpenWRT).

5.5.2. Retroalimentación

Este sistema sirve para informar al usuario, de modo que cuando se realice un cambio de en la configuración de uno o varios dispositivos el éxito o fracaso del cambio en cada uno ellos quede reflejado en la interfaz. Para ello se utilizarán códigos de colores; se mostará la línea correspondiente a un dispositivo de color verde cuando los cambios se han realizado correctamente, de color rojo cuando ocurra el caso contrario y, por último, si la línea aparece en gris indica que no se realizaron cambios en ese *router*. En la figura 5.10 se muestra un caso de este tipo, que representaría el estado siguiente de la aplicación tras la imagen 6.7. Como podemos observar, nos muestra que el cambio se ha realizado correcta-

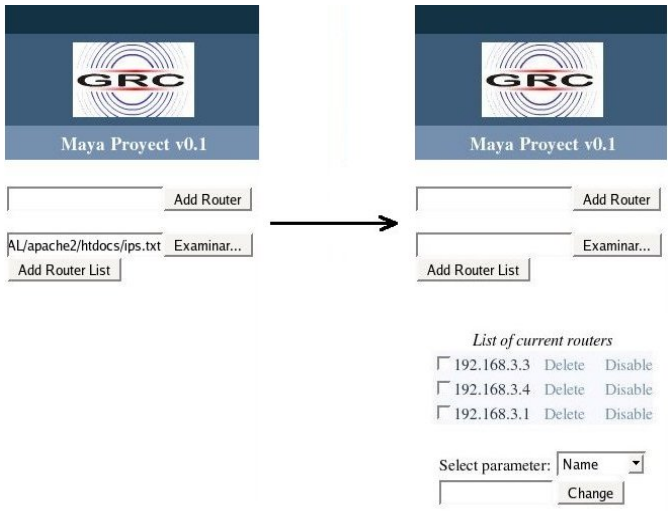

Figura 5.4: Añadir varios dispositivos

mente en el dispositivo con dirección IP 192.168.3.1 y no se ha podido realizar en el que tiene como dirección 192.168.3.7.

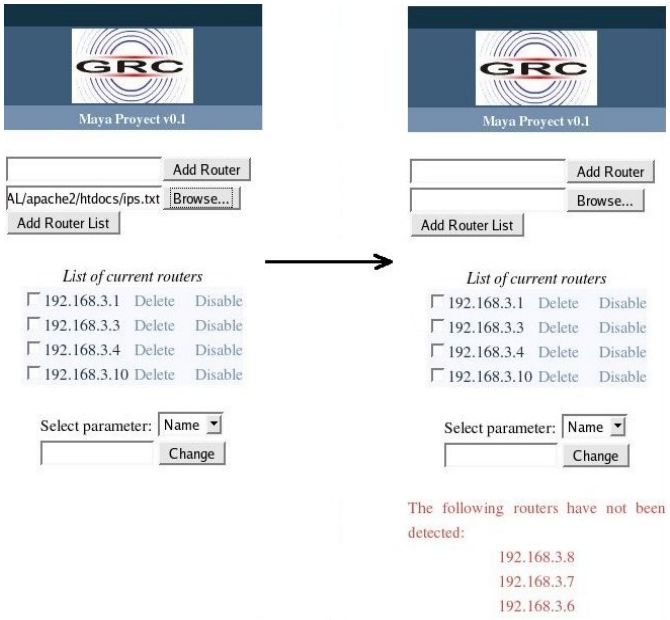

Figura 5.5: Error al añadir varios dispositivos

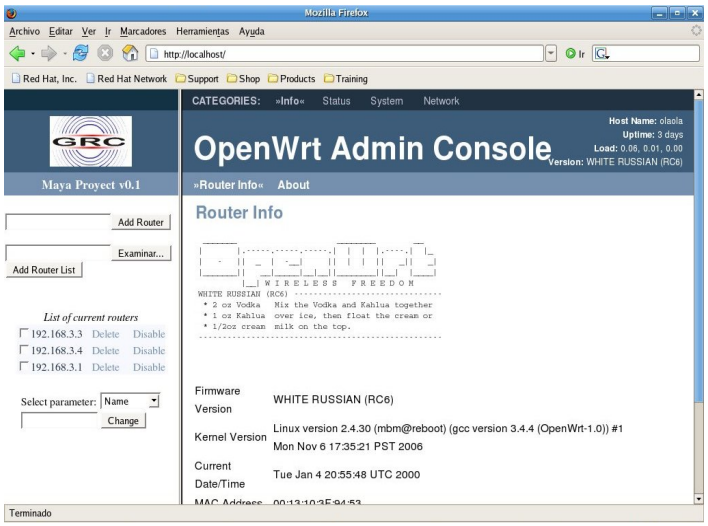

Figura 5.6: Interfaz completa

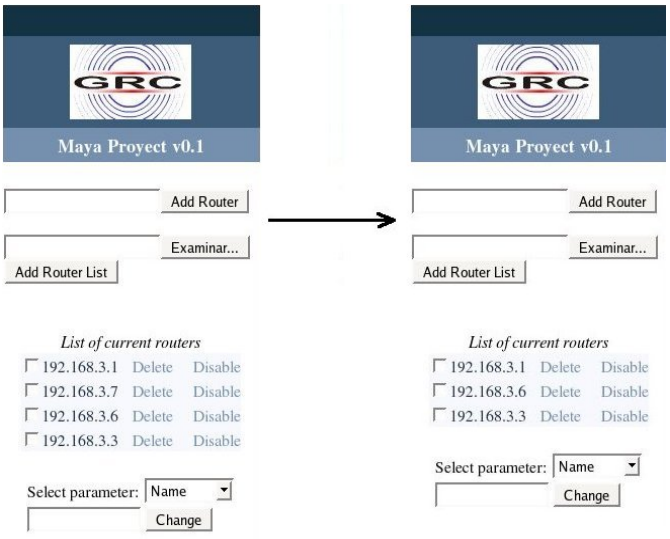

Figura 5.7: Borrar un dispositivo

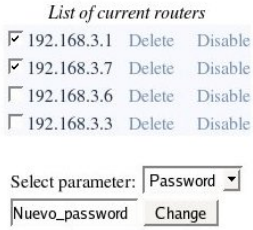

Figura 5.8: Cambio de parámetros

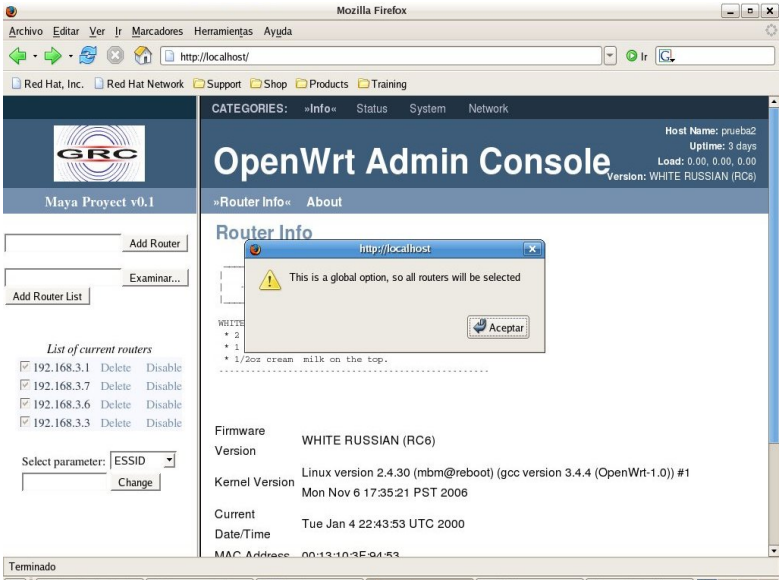

Figura 5.9: Parámetro Global

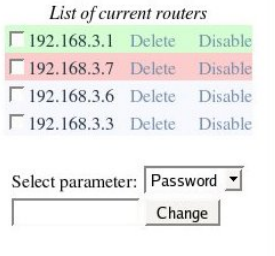

Figura 5.10: Retroalimentación

Capítulo 6

Experimentación

6.1. Dispositivos utilizados en las pruebas

Para las pruebas que hemos realizado y que vamos a presentar a continuación, ha sido necesario el uso de diversos dispositivos. Como nodos de la red *ad-hoc* hemos utilizado los ya mencionados puntos de acceso de *Linksys*, cuyas características ya fueron detalladas en el punto 1.2.3.

En cuanto a dispositivo que realiza las funciones de punto de control hemos utilizado un computador de sobremesa, con procesador AMD Athlon(TM) XP 2200+ y 512 MB de RAM. Aunque, verdaderamente, el componente de más relevancia es sin duda la tarjeta inalámbrica empleada, que en nuestro caso se trata de la Conceptronic 54Mbps USB. También es importante el sistema operativo empleado y de la versión del *kernel* para aportar mayor compatibilidad a las extensiones *wireless.* Nuestras pruebas se han llevado a cabo sobre la distribución de Linux Fedora Core 5 usando las *wireless extensions v19* así como las *wireless tools v28,* en ambos casos la versión más reciente hasta la fecha de publicación de este documento.

También hay que citar que se han realizado pruebas con diversas tarjetas inalámbricas (cómo la propia *Lnksys WRT54G*) sobre el sistema operativo mencionado líneas más arriba, obteniendo dificultades a la hora de realizar algún cambio de parámetro sobre ellas, como el canal o la clave de cifrado WEP.

6.2. Simulación alcanzabilidad entre nodos

Antes de comenzar a realizar pruebas y analizar las características temporales y de rendimiento de nuestro sistema, vamos a explicar el método utilizado para la simulación de los escenarios para las diferentes pruebas.

Como todos sabemos, la alcanzabilidad de la señal de un router tiene un radio limitado, éste se situa alrededor de los 100 metros. Puesto que las simulaciónes han sido realizadas en un laboratorio y no en un entorno real, tendremos que emplear un método para simular que la señal de un *router* no alcanza a otro,

y poder, de esta forma, simular un escenario con diversas topologías. Además, también es necesario para comprobar el correcto funcionamiento del protocolo de encamiento, pues si todos los nodos fueran alcanzables por todos, no habría ningún camino que buscar.

Nos basaremos de nuevo en *Iptables* para simular esta condición. La idea consiste en bloquear todas las tramas que lleguen con la dirección MAC del dispositivo con el que cuál queremos simular que no hay visión directa. Para aclarar este concepto vamos a ver un sencillo ejemplo:

Figura 6.1: Escenario Iptables

Como se puede apreciar en la figura 6.1, hay visión directa entre los *routers* 1 - 2, 2 - 3 y 3 - 4.

Para entender bien el procedimiento a seguir, tenemos que imaginarnos un escenario formado por esos mismos dispositivos en el que existen líneas de conectividad desde cada uno a todos los demás, y que para llegar a nuestro escenario objetivo, que es el de la figura 6.1, debemos ir eliminándolas una a una. Cada línea de alcanzabilidad que queramos suprimir va a suponer la ejecución de un comando *Iptables* en cada uno de los dos dispositivos implicados.

El comando a ejecutar en cada dispositivo será "*Iptables -I INPUT -m mac –mac-source mac_del_otro_dispositivo -j DROP*", lo que significa que todos los paquetes cuya dirección MAC sea la que hemos introducido que vayan destinados al dispositivo que ejecuta la instrucción serán rechazados.

De este modo las intrucciones que deberíamos ejecutar en cada dispositivo para crear nuestro escenario serán:

- Punto de acceso 1 (visibilidad con 2, no visibilidad con 3 y 4)
  - *Iptables -I INPUT -m mac –mac-source CC:CC:CC:CC:CC:CC -j DROP*
  - *Iptables -I INPUT -m mac –mac-source DD:DD:DD:DD:DD:DD -j DROP*
- Punto de acceso 2 (visibilidad con 1 y 3, no visibilidad con 4)
  - *Iptables -I INPUT -m mac –mac-source DD:DD:DD:DD:DD:DD -j DROP*
- Punto de acceso 3 (visibilidad con 2 y 4, no visibilidad con 1)

- *Iptables -I INPUT -m mac –mac-source AA:AA:AA:AA:AA:AA -j DROP*

- Punto de acceso 4 (visibilidad con 3, no visibilidad con 1 y 2)
  - *Iptables -I INPUT -m mac –mac-source AA:AA:AA:AA:AA:AA -j DROP*
  - *Iptables -I INPUT -m mac –mac-source BB:BB:BB:BB:BB:BB -j DROP*

Para llevar a cabo este método, hemos tenido que instalar el paquete *iptables-mod-extra* que se encuentra disponible en la lista de software disponible para OpenWRT (accediendo a la interfaz de configuración web, *system*, *installed software* y *update packages*). Esto nos añadirá mucha más funcionalidad a *Iptables* entre ellas las funciones del módulo *mac* que usamos en las instrucciones antes mencionadas (*-m mac*). Antes de ejecutarlas, tendremos que cargar dicho modulo en memoria, para ello: *"insmod /lib/modules/2.4.30/ipt_mac.o"*.

6.3. Resultados de las pruebas

A continuación, vamos a mostrar los resultados de tiempo obtenidos para las diferentes acciones que puede llevar a cabo la herramienta dentro de la red *mesh*. Además obtendremos y valoraremos previamente los tiempos del protocolo de encaminamiento que necesitaremos para realizar dichas operaciones dentro de la red. Todos los tiempos están tomados respecto al número de saltos (*Hops*) que hay en el camino entre el punto de control que envía los mensajes y el nodo destino de la red.

6.3.1. Tiempos del protocolo de encaminamiento

En primer lugar vamos a medir el retardo del envío de un mensaje ICMP, algo que se va a producir con relativa frecuencia en nuestra aplicación, para comprobar si un nodo está habilitado, actualización de TTL y otras funciones ya comentadas anteriormente.

Dado el pequeño tamaño y el propósito de los mensajes ICMP generados por el *ping,* nos servirá para obtener un medida estimatoria del retardo del protocolo de encaminamiento a la hora de encontrar una ruta adecuada. Como se puede observar en la figura 6.2 el tiempo es realmente bajo comparado con las operaciones que evaluaremos más adelante.

En ésta segunda gráfica (6.3), se han obtenido los tiempos del protocolo a la hora de encontrar una nueva ruta, esto es, la ruta entre el punto de control y un dispositivo que acaba de arrancar el protocolo de encaminamiento en ese instante. Como se observa, el tiempo para un dispositivo que se encuentra a tan sólo un salto de distancia el tiempo es de casi 16 segundos y conforme aumenta el número de saltos aumenta muy ligeramente.

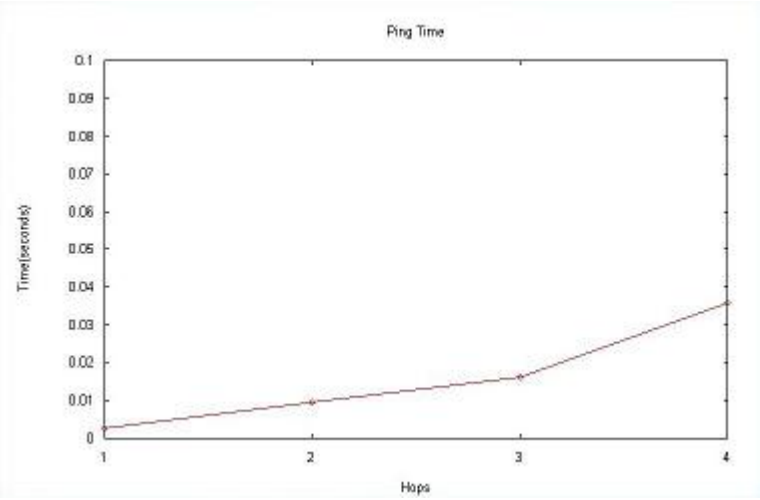

Figura 6.2: Retardo de un mensaje ICMP

Esto es debido al parámetro *wait on reboot* del AODV-UU, inicialmente configurado a 15 segundos, esto significa que esperará dicho tiempo para aplicar de forma coherente toda la información recibida por los nodos de alrededor y por tanto esto implica que éste parametro marca una cota mínima. Por otro lado decir que este parámetro es configurable por el usuario, para ello deberemos modificar el parámetro *active_route_timeout_hello* de fichero *params.h* antes de realizar la compilación del protocolo de encaminamiento.

6.3.2. Tiempos sin tráfico en la red

Las gráficas de las diferentes operaciones que se analizan a continuación, han sido realizadas tomando tiempos en una red en la que no había ningún tipo de tráfico que no fuera el generado por el propio protocolo de encaminamiento.

6.3.2.1. Cambio de parámetros en dispositivos activados

En este apartado vamos a obtener el tiempo necesario para poder cambiar un parámetro en un dispositivo. En este primer ejemplo todos los routers del sistema se encuentran habilitados, por tanto implica que el canvio de parámetros se realiza mediante una conexión SSH.

En la figura 6.4 se pueden observar los resultados. La gráfica representa el tiempo necesario para cambiar un parámetro en un dispositivo que se encuentra a un número de saltos determinado del punto de control. En el caso de un sólo salto tenemos que el tiempo es aproximadamente de un 1 segundo, que supone el tiempo en establecer la conexión SSH con el dispositivo, ejecutar las instrucciones necesarias y cerrar la conexión. Como se puede apreciar el aumento del coste temporal en función de los saltos evoluciona de una forma

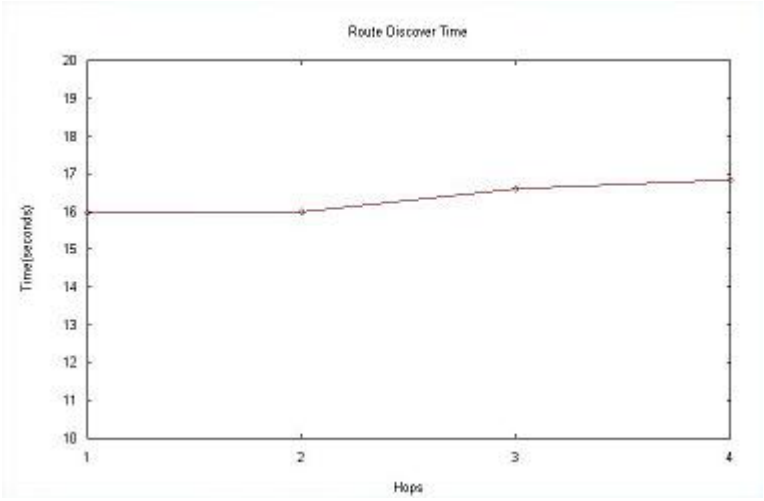

Figura 6.3: Tiempo de descubrimiento de una ruta

practicamente lineal con muy poca pendiente, de forma que para cuatro saltos el retardo no alcanza los 1,5 segundos. Tiempos más que aceptables si tenemos en cuenta que el número de saltos máximos de una red *mesh* debería crecer mucho más de los valores representados en la gráfica.

6.3.2.2. Cambio de parámetro en dispositivos desactivados

En este apartado vamos a medir el tiempo necesario para realizar un cambio de parámetros en dispositivos deshabilitados, esto implica que la instrucción llegará a su destino mediantes mensajes UDP de *broadcast*. En este grupo también se incluiran las opciones de habilitar/deshabilitar un dispositivo, puesto que esta opción también esta implementada en el sistema de mensajes UDP.

En la imagen 6.5 se aprecian los tiempo obtenidos tomados desde el momento que envíamos el mensaje hasta que recibimos el reconocimiento. Si los comparamos con los retardos de la conexión SSH necesaria para cambiar los parámetros en el apartado anterior (dispositivos habilitados) podemos observar que el tiempo que obtenemos es mucho menor. La razón es obvia, la tarea de crear una conexión SSH es mucho más costoso que el envío de un mensaje con un servicio no orientado a conexión como es UDP.

En cambio, a medida que aumenta el número de saltos entre el punto de control y el dispositivo destino, el coste temporal crece más rápidamente que en el caso de la conexión SSH. Esto es debido a que en este caso el mensaje es recibido, procesado y reenviado por todos los nodos del camino, lo que hace que se añada un coste extra. Sin embargo, como podemos comprobar en la gráfica el tiempo obtenido para un número de cuatro saltos apenas alcanza los 0,4 segundos, un tiempo más que correcto.

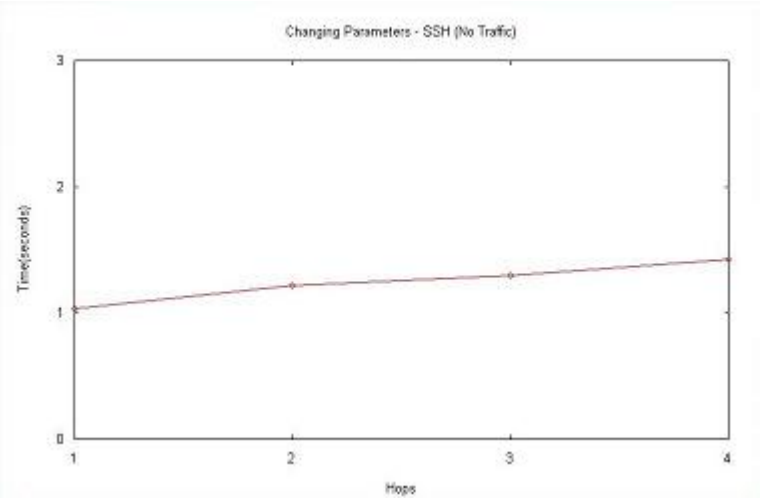

Figura 6.4: Cambio de parámetros en nodos activados sin táfico

6.3.2.3. Intercambio de claves y HITs

A continuación vamos a analizar el retardo de la operación de insertar un dispositivo nuevo en la lista asociada a la base de datos. Recordemos que esta operación supone el intercambio via SSH de las claves públicas y los HITs de cada dispositivo.

En la figura 6.6 se pueden observar que el tiempo medio para un nodo situado al menor número de saltos posible es de 3 segundos. Si comparamos estos resultados con la operación de cambio de parámetros, también realizada por SSH, podemos observar que el tiempo base (para un sólo *hop*) crece bastante debido precisamente al intercambio de varios ficheros entre las dos fuentes y no la simple ejecución de varias instrucciones en el dispositivo remoto.

6.3.3. Tiempos con tráfico TCP en la red

Para hacer una simulación más acorde con la realidad, en la que el administrador de una red realizará cambios en la configuración de ésta mientras los usuarios transmiten datos, hemos vuelto a realizar las tomas de los tiempos del mismo modo que en el ejemplo anterior al mismo tiempo que se establecen diversos flujos de datos TCP dentro de la red. El tráfico TCP ofrece un servicio orientado a conexión y tiende a utilizar todo el ancho de banda de la conexión establecida en los dispositivos por lo que provocará el retraso y la pérdida, en algunos casos, de los mensajes UDP.

Para la realización de estas pruebas se ha utilizado una utilidad implementada por Jorge Hortelano, responsable del proyecto CASTADIVA[10], y que pertenece al mismo grupo de investigación en dónde ha sido desarrollado el propio MAYA. Esta utilidad, que consta de dos procesos (cliente y servidor), nos

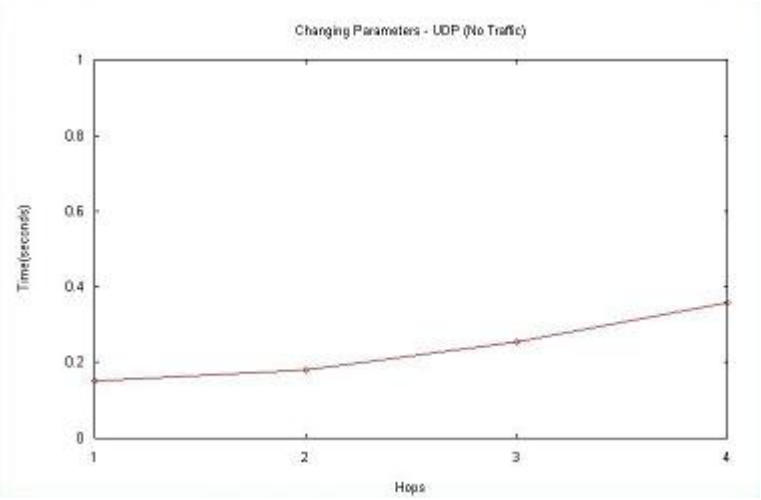

Figura 6.5: Cambio de parámetro en nodos desactivados sin tráfico

permitirá generar tráfico TCP entre dos dispositivos cualesquiera de la red.

Se han realizado mediciones con diversos número de flujos TCP entre en el punto de control y el dispositivo dónde se realizará el cambio, concretamente 1,5,10 y 20, dónde también se ha variado el número de saltos como en el punto anterior.

6.3.3.1. Cambio de parámetros en dispositivos activados

En la figura 6.7 podemos observar el aumento del coste temporal, al aumentar el número de saltos y de flujos de tráfico TCP dentro de la red.

Como es de esperar los tiempos aumentan ligeramente en función del número de flujos TCP y el número de saltos, sin embargo, como se aprecia en la figura 6.7 los tiempos se adecúan perfectamente a las necesidades del sistema, no superando los 4,5 segundos en el peor de los casos.

6.3.3.2. Cambio de parámetros en dispositivos desactivados

En la siguiente gráfica (6.8) observamos los tiempos de cambio de parámetros mediante mensajes UDP. Al igual que en el caso sin tráfico, el tiempo que se obtiene es la suma del envío del mensaje y de la recepción del reconocimiento.

Si lo comparamos con el canvio de parámetros usando SSH, vemos que el tiempo es mucho menor, sin embargo, como es de esperar, se producirán un mayor un número de intentos fallidos. De hecho, estos reintentos deben ser tomados en cuenta a la hora de evaluar los costes. Las pruebas fueron realizadas con el parámetro $WAIT\_TIME$=5, esto es el tiempo que espera el punto de control a reenviar el mensaje en caso de no haber recibido el reconocimiento. Por tanto, si el reconocimiento del mensaje llega al segundo intento realizado

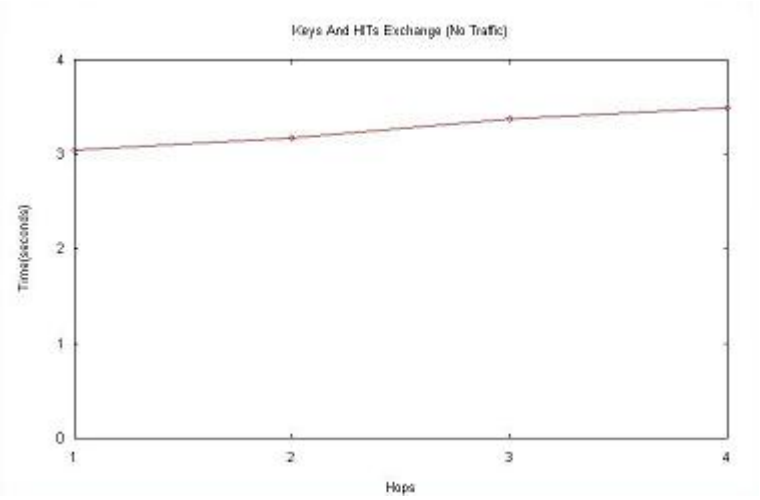

Figura 6.6: Intercambio de claves y HIT sin tráfico

por el punto de control el retardo que obtendremos será igual a 5 más el retardo real del mensaje.

Podriamos haber optado por la opción de eliminar los tiempos de espera entre mensaje y mensaje y de esta forma calcular el tiempo que necesitan para recorrer el camino hasta el nodo destino únicamente, es decir, considerando que todos los mensajes UDP envíados siempre llegan a su destino, sin embargo, no nos parecía lo más adecuado, puesto que no reflejaría el tiempo de espera por parte del usuario del sistema y lo que pretendemos es obtener los tiempos más reales posibles. Por tanto hemos optado por mostrar en una segunda gráfica (6.8) la tasa de mensajes UDP enviados con éxito. Este estudio nos permitirá conocer para un número de saltos y un número de flujos TCP en la red el porcentaje aproximado de éxito. Como es lógico y se puede apreciar en la gráfica a medida que aumenta el número de saltos y el número de transmisiones TCP se producen más pérdidas de paquete.

6.3.3.3. Intercambio de claves y HITs

La última parte que nos queda por analizar es el intercambio de ficheros iniciales entre el punto de control y el punto de acceso. Si lo comparemos con el intercambio de parámetros en dispositivos activados, también implementada mediante conexión SSH, observamos unos tiempos sensiblemente superiores debido a que no se trata de la ejecución de una instrucción en el terminal remoto, si no de la transmisión de varios ficheros entre los dos dispositivos.

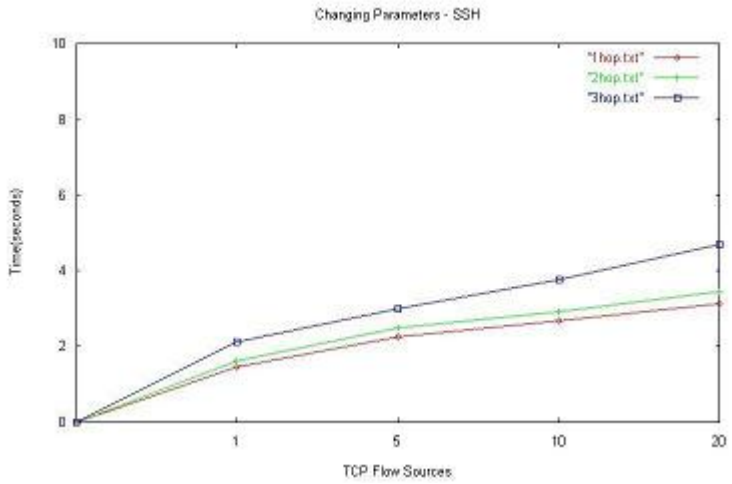

Figura 6.7: Cambio de parámetros en nodos activados con tráfico

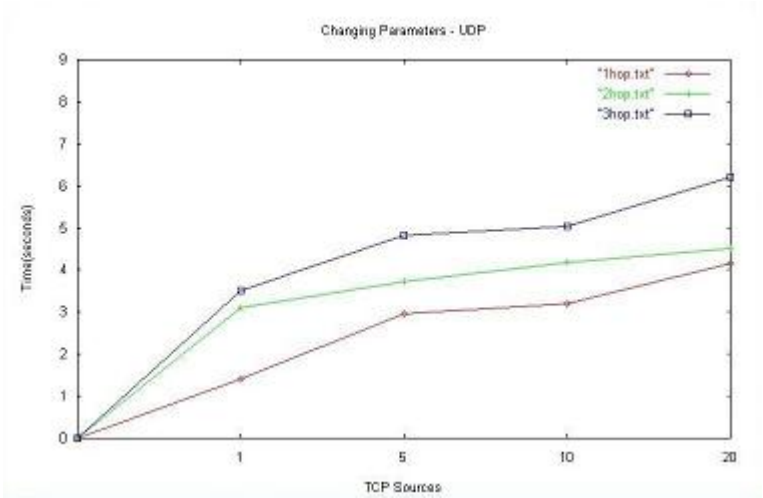

Figura 6.8: Cambio de parámetros en nodos desactivados con tráfico

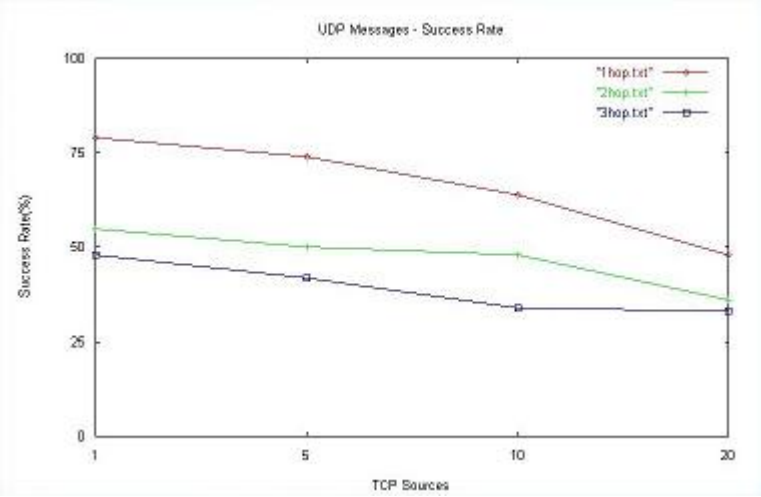

Figura 6.9: Tasa de mensajes UDP enviados con éxito

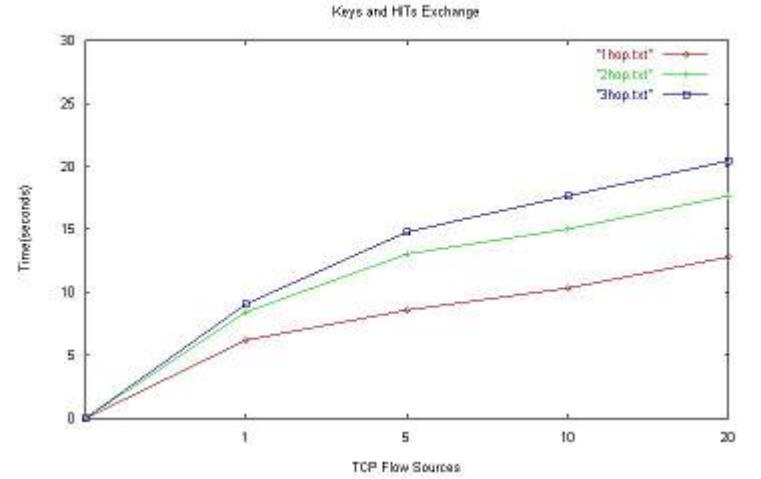

Figura 6.10: Intercambio de claves y HIT con tráfico

Capítulo 7

Trabajo Futuro

En este capítulo vamos a presentar algunas líneas de mejora de la aplicación, que pueden contribuir aportando nueva funcionalidad o mejorando la que ya se dispone. Hay que apuntar que, debido a la naturaleza de la aplicación, se pueden dar muchas situaciones diferentes en cuanto a ubicación del punto de control y los nodos de la red. En algunos casos tendremos un computador formando parte permanentemente de la red mesh, en otros casos tendremos un ordenador (posiblemente un portátil) que se encargará de configurar varias redes. Y, en cuanto a lo que los nodos de la red se refiere es posible que siempre los queramos mantener habilitados y sólo usemos la parte de configuración de parámetros de la herramienta o al revés.

Dependiendo de la funcionalidad final que le queramos dar a la herramienta nos pareceran más interesante una mejora u otra de modo que se adapte a nuestras necesidades, a continuación vamos a enumerar algunas importantes.

7.1. Función *gateway* de un punto de acceso

La función *gateway* de un nodo es lo que va a proporcionar a la parte *ad-hoc* de la red *mesh* aceso a internet, algo esencial para la finalidad de este tipo de redes (figura 3.1). Hay que destacar que el protocolo de encaminamiento AODV-UU, empleado en este proyecto, proporciona esta posibilidad en su última versión publicada hasta la fecha (0.9.3).

El procedimiento básico del nodo que encamine mensajes de la red *mesh* a internet, será la de eliminar la información adicional que añade el protocolo de encaminamiento de forma que el formato del paquete sea correcto para llegar a su destino. Y deberá realizar el proceso inverso para encaminar la información que llega desde internet y tiene como un destino un nodo que forma parte de la red *mesh*.

7.2. Detección automática de nodos mediante *fping*

En Maya, la forma actual de añadir dispositivos a la lista es manual, en el sentido que el administrador de la red debe recordar las direcciones IP de los nodos, aunque las puede guardar en un fichero y luego añadirlas mediantes la opción de insertar lista de puntos de acceso (4.6.2).

Sin embargo, una opción muy cómoda para el administrado sería añadir todos lo nodos que en ese momento formen parte de la red de forma automática. Una posible solución sería *fping*[11], que además se encuentra en la lista de paquetes para instalar en OpenWRT Esta utilidad nos permite comprobar, al igual que *ping*, si un *host* de la red está activo mediante mensajes ICMP, pero además permite averiguar que direcciones IP de una red están activadas dentro de todo el rango de direcciones de esa red, lo que resultaría bastante útil en nuestra aplicación.

Aunque, como vamos a ver, este método tiene algunos problemas. En primer lugar basa su implementación en el comando *ping*, de forma que si queremos comprobar todas las direcciones IP de una red, se va a generar una gran tráfico en ese momento determinado, y si, además, nos encontramos con una topología de red, con muchos saltos y enlaces entre nodos puede resultar tremendamente costoso. De cualquier forma habría que realizar pruebas para estudiar la fiabilidad y rendimiento de esta herramienta en un entorno como el de una red *mesh* para el cual no fue diseñado.

Otra desventaja es que todos los dispositivos de la red deben estar activados, de otra forma no respondería a los mensajes ICMP.

7.3. Detección automática de nodos mediante mensaje de reconocimiento

Esta es una posible solución que hemos estudiado para realizar la detección automática de los nodos de una red y que al ser una solución orientada a nuestro problema en particular resultaría una alternativa interesante a la comentada de *fping* en el punto anterior.

El objetivo básico de esta idea se basa en solucionar los puntos en contra de la solución anterior. Para ello, cuando el punto de control se incorpore a la red o simplemente quiera conocer las direcciones de los nodos que hay actualmente podría envíar un mensaje, similar al especificado en el punto 4.4.3, pero con algún valor especial en un campo determinado para indicar que se trata de un mensaje de reconocimiento. Uno de los valores especiales en el campo sería el de dirección destino, que en este caso coincidiría con la de *broadcast* puesto que ya no buscamos un destinatario concreto y las tablas de control impedirán que la red se sature.

Después del envío del mensaje, los puntos de acceso que lo hayan recibido, enviarán otro al punto de control para informarle de su existencia, pudiendole comunicar también si en ese momento se encuentra habilitado o deshabilitado,

así como otra información importante para que el punto de control lo anote en su base de datos.

De esta forma, evitaríamos por un lado, la saturación de la red y permitiríamos la detección tanto de nodos habilitados como deshabilitados.

7.4. Programación de los mensajes de configuración

Una funcionalidad que creemos muy útil sería la programación horaria de envio de mensajes de configuración, es decir, poder indicar al sistema que todos los días a una determinada hora queremos deshabilitar uno o varios dispositivos. De esta forma, podríamos conseguir, si nuestra red está implantanda en una ubicación dónde queremos proporcionar conectividad entre los miembros de la red o entre estos e internet durante un determinado horario podríamos llevarlo a cabo. O realizar cambios periódicos de clave de encriptación WEP o cambio de password semanales a los dispositivos, todo ello liberaría aún más de faena al administrador.

Capítulo 8

Conclusiones

En este documento se ha presentado la arquitectura y funcionamiento del sistema Maya. Una herramienta sencilla pero útil que proporcionará al administrador de una red *mesh* una forma sencilla de controlar y configurar los dispositivos que la forman.

A la hora de construir esta utilidad ha primado, por encima todo, la facilidad de uso y la interfaz sencilla de cara al usuario, de forma que rápidamente se familiarice con la herramienta y se adapte a su modo de empleo. Para ello, se ha integrado su diseño con OpenWRT, el *firmware* personalizable más extendido actualmente para dispositivos empotrados. Esto nos permite por un lado tener la interfaz *web* para la configuración individual de un *router* y por otro lado la funcionalidad que ofrece nuestra herramienta orientada a proporcionar facilidades para configurar una red mallada formada por varios dispositivos.

Como hemos observado en el apartado de experimentación, se obtenido que los tiempos y el rendimiento de la herramienta se adaptan a las necesidades de una red *mesh*, dónde hay que tener en cuenta que los cambios en su configuración no van a ser muy frecuentes.

Hay que tener en cuenta que hoy en día se esta produciendo un incremento vertiginoso en el uso de la tecnología inalámbrica en general y de este tipo de redes en particular, por lo que pronto se hará necesarias herramientas que faciliten su implantación y administración y que se vayan adaptando las nuevas necesidades que surjan. En ese proyecto hemos pretendido la creación de un sistema de configuración que cubra las necesidades principales de una red de este tipo. Además su fácil ampliabilidad lo puede convertir en posteriores revisiones en una herramienta más completa y desarrollada que no sólo cubra requisitos generales si no que pueda especializarse para casos de uso concretos, puesto que cada vez será mayor el número de usuarios que use este tipo de redes. Esto es debido, sobre todo, a lo barata que resulta la instalación y mantenimiento de este tipo de redes y a las ventajas de comodidad que ofrece.

Como acabamos de comentar, cada vez el uso de las redes malladas se va especializar, de hecho ya se usa actualmente en vigilancia usando cámaras de seguridad inalámbricas, por ejemplo, y otro tipo de redes provisionales que, bien

por ser temporales o bien por la dificultad de establecer cableado en una zona determinada, resulta más adecuado o ecónomico la instalación de una red sin cables.

Bibliografía

[1] FON. Wi-fi community. http://www.fon.com.

[2] FREE. Proveedor de servicios. http://www.free.fr.

[3] OpenWRT. Linux distribution for embedded systems. http://openwrt.org/.

[4] Core Group. Implementation portal. http://core.it.uu.se/core/index.php.

[5] CuWIN. Community wireless solutions. http://www.cuwireless.net/.

[6] Wifidog. A captive portal suite. http://dev.wifidog.org/.

[7] Pello Xabier Altadill Izura. Iptables, manual práctico. http://www.pello.info/filez/firewall/iptables.html.

[8] Wikipedia. Rsa algorythm overview. http://en.wikipedia.org/wiki/rsa.

[9] R. Moskowitz and P. Nikander. *Host Identity Protocol (HIP) Architecture*. ICSA Labs (Cybertrust) and Nomadic Lab (Ericsson), May 2006.

[10] Jorge Hortelano Otero. Grupo de investigación de redes de computadores. http://www.grc.upv.es/6software.html.

[11] Thomas Dzubin. Fping. a program to ping hosts in parallel. http://www.fping.com/.

[12] How to build packages using OpenWRT's SDK. http://wiki.openwrt.org/buildingpackageshowto.

Apéndice A

Instalación y configuración de Maya

A.1. Requisitos

Para un correcto funcionamiento del sistema se deben tener instalados una serie de componentes tanto en punto de control como en los puntos de acceso. En las siguientes subsecciones se explicará detenidamente la instalación de cada una de ellas.

A.1.1. Punto de control

1. APACHE: versión 2.0 o superior. Además deberá estar configurado para funcionar con OpenSSL. En el punto A.3.1.1 se detalla los pasos a seguir para ello.

2. MySQL: versión 5.0 o superior. Se recomienda así mismo disponer de una herramienta como MySQL Query Browser que facilita la realización de consultas y visualización de tablas de la base de datos.

3. PHP: versión 5.0 o superior. Debe estar configurado correctamente para funcionar con Apache y MySQL. Al igual que en el caso anterior, se explicará paso a paso su instalación.

4. OpenSSL: versión 0.9.8d. Es un requisito necesario por PHP ya que va a emplear funciones de acceso por SSH (libssh2) y necesita disponer de estas librerías.

5. AODV-UU: Modificación del AODV de la Universidad de Uppsala, versión 0.93 o superior. El protocolo de encaminamiento necesario para poder formar parte de la red *ad-hoc*.

6. OpenWRT SDK[1]: *Software Developement Kit* de OpenWRT[12], nos servirá para desarrollar nuestras propias aplicaciones para OpenWRT. Incluye numerosas utilidades como el compilador cruzado que permitirá compilar código para una máquina (como el punto acceso) que usa una arquitectura diferente al equipo dónde realizamos la compilación.

A.1.2. Puntos de acceso

1. OpenWRT: versión WhiteRussian RC6. La interfaz de la aplicación ha sido desarrollado sobre dicha versión (que es la más reciente a la finalización de este proyecto), por tanto se recomienda altamente su uso, ya que la interfaz gráfica varía ligeramente de una a otra.

2. AODV-UU: El protocolo de encaminamiento necesario para poder formar parte de la red *ad-hoc*, este será un componente que funcionará tanto en el punto de control como en los puntos de acceso de la red.

3. Libgcc: paquete disponible para instalación con el OpenWRT. Esta librería es necesaria, ya que vamos a utilizar código compilado en C en los puntos de acceso.

4. Libopenssl[2]: librería necesaria para el sistema de cifrado de clave pública por el método RSA.

A.2. Copia e instalación de los ficheros del programa

Suponiendo que tenemos todos los componentes que hemos enumerado ya se encuentren correctamente instalados en nuestro sistema (en caso de no ser así, en el punto A.3 se explica detenidamente la instalación de todos ellos), vamos a detallar los pasos a seguir para la instalación y configuración de nuestra aplicación.

Debemos señalar que toda la configuración de los puntos de acceso que vamos a explicar en el punto A.2.1, están preparados para funcionar sobre el modelo *Linksys WRT54G*. Esto implica que los procesos están ya compilados para arquitectura MIPS, la interfaz inalámbrica es la *eth1* y otras características, por tanto, recomendamos al usuario que vaya a realizar la instalación en otro dispositivo leer el punto A.2.3, dónde se detallará los aspectos que debe tener en cuenta.

[1] Este no es un requisito indispensable, ya que en este proyecto facilitamos la versión compilada del protocolo tanto para i386 como para MIPS, pero en caso de que el usuario desee compilar otra versión, otro protocolo, o cualquier otra utilidad siempre le será de gran ayuda este *software* de desarrollo.

[2] Actualmente la versión que proporciona OpenWRT de esta librería es la 0.9.8d (la más reciente), es muy posible que necesitemos la misma versión de OpenSSL en el equipo que realiza las funciones de punto de control.

A.2.1. Puntos de acceso

1. Debemos copiar la carpeta *MAYA* a cada uno de los puntos de acceso de la red *mesh*. Es importante que esta carpeta se encuentre en el directorio raíz de OpenWRT. Para ello podemos utlizar SSH y realizar la copia mediante el comando: "*scp -r path_a_la_carpeta_MAYA ip_del_dispositivo*".

2. A continuación debemos acceder al sistema operativo del router mediante SSH, ejecutando simplemente: "*ssh ip_del_dispositivo*"[3]. Llegado a este punto, si es la primera vez que accedemos al punto de acceso después de instalar OpenWRT, deberemos configurar el password vía *web* en la página de configuración del *router* (*http://ip_del_dispositivo*), dónde se nos informará que es la primera vez que accedemos al sistema y debemos introducir una clave nueva. Una vez realizado este paso, no deberíamos tener problema para acceder por SSH.

3. El siguiente paso es acceder a la subcarpeta *init_scripts*, dónde ejecutaremos el *script install.sh* de la forma: "*sh install.sh*". La función de este pequeño *script* es configurar el punto de acceso para ejecutar varios procesos cada vez que arranque. Estos procesos son: arrancar el servidor UDP para la recepción de mensajes, así como el protocolo de encaminamiento, y ,en caso de que sea la primera que se inicia el dispositivo, generar su clave pública y privada. Para ello añade en el directorio */etc/init.d* (dónde se encuentran los procesos que se ejecutarán al arrancar el sistema) enlaces simbólicos a los *scripts* que realizan las funciones anteriormente comentadas. Después de realizar esta operación el *router* se reiniciará automáticamente y la instalación estara completada.

A.2.2. Punto de control

1. Para empezar, debemos copiar los archivos que se encuentran en la carpeta *Maya_web* en el directorio donde tengamos configurado nuestro servidor *Apache* para mostrar las páginas. Para que no haya problemas a la hora de generar los archivos necesarios para el correcto funcionamiento de la aplicación, debemos asegurarnos que dicho directorio cuenta con todos los permisos de lectura escritura, al menos para el usuario que crea *Apache* (*daemon*) que será el que ejecute los comandos de nuestra aplicación.

2. Una vez hayamos otorgado los permisos, deberemos acceder a la carpeta *Maya_web*, ya correctamente ubicada, y accederemos al fichero *params.inc*, dónde deberemos configurar algunos parámetros:

   - w_interface: el nombre de nuestra interfaz inalámbrica (por defecto *wlan0*).

[3] OpenWRT, sólo cuenta con el usuario *root*, por tanto debemos acceder desde nuestro sistema siendo superusuario.

- default_password: el *password* por defecto que utilizará la aplicación para acceder vía SSH a los dispositivos.

A continuación, en el fichero *dbopen.inc*, se establecen el nombre y la contraseña para el acceso a la base de datos, debemos asegurarnos que usaremos en el punto 4 los mismos valores que aquí indiquemos, para que se pueda producir la conexión con la base de datos.

Por último ejecutaremos el *script rsa.sh* que se halla en la misma carpeta ejecutando para ello "*sh rsa.sh*". Esto nos permitirá generar las claves pública y privada y el HIT del punto de control.

3. Así pues, también deberemos dar privilegios al usuario *daemon* para que pueda ejecutar comandos de configuración de nuestra tarjeta *wireless*. Para ello deberemos debemos añadir la línea siguiente en el archivo de sistema *sudoers* (podemos acceder directamente a editar este archivo usando el comando de consola *visudo*) "*daemon ALL=NOPASSWD:/LOCAL/sbin/iwconfig*".

4. Cómo ultimo paso deberemos cargar el esquema de la base de datos Maya. Para conseguirlo, deberemos generar la base de datos ejecutando "*mysqladmin create maya -p*" y cargaremos el esquema que se proporciona en la carpeta Database: *"mysql maya <maya.sql -p*". Ahora tan sólo nos resta, crear el usuario Maya, ya que las peticiones a la base de datos se realizarán con este nombre. En primer lugar, entraremos en *mysql* como superusuario ejecutando "*mysql -u root -p*" y a continuación crearemos el usuario y le otorgaremos permisos con la instrucción *"grant all on maya.* to maya@localhost identified by password_del_punto_2*".

A.2.3. Compilación y adaptación a otras plataformas

En este apartado vamos a explicar las características a tener en cuenta cuando queramos compilar los ficheros de la aplicación a otra platarforma.

En la carpeta *source*, encontraremos los ficheros fuente de los procesos que utiliza nuestra aplicación, por un lado los ficheros de los *sockets udp* (*client, server y clientCP*) y por otro el protocolo de encaminamiento, *AODV-UU* (que se explicará en puntos posteriores). Además, para cada uno de ellos ya encontramos versiones compiladas para MIPS y X86. No obstante, es posible que el usuario quiera recompilarlos, bien para esa plataforma o bien para otra, especialmente si se está usando un modelo de punto de acceso diferente.

Para realizar la compilación a otra plataforma es requisito indipensable disponer de un compilador para ésa arquitectura objetivo y, cómo se verá más adelante, disponer de las librerías OpenSSL, ya que se hacen uso de funciones que se encuentran definidas en ellas. También será necesario acceder al fichero *sockets.h*, dónde deberemos indicar cuál es la interfaz inalámbrica en nuestro dispositivo en la función *get_wlan_ip* antes de realizar la compilación.

Con todo ello, ya estaría todo a punto para realizar la compilación ejecutando: "*compilador_cruzado_C -lssl archivo_fuente -o archivo_destino*". Recorde-

mos que la opción -lssl nos permitirá utilizar todas las funciones incluidas relacionadas con el sistema de claves.

A.3. Instalación de los programas necesarios

A.3.1. Punto de control

A.3.1.1. Apache

A continuación vamos a mostrar los pasos básicos para la instalación del servidor Apache en nuestro computador. No se aconseja el uso de paquetes que proporcionan las distribuciones ya que no pueden ser configurados con las opciones necesarias antes de ser instalado.

1. En primer lugar nos dirigiremos a la página oficial de esta aplicación *http://httpd.apache.org/* y descargaremos la versión que deseemos. El sistema Maya ha sido probado con la versión 2.2.2 de Apache.

2. Una vez se ha desargado el arhivo, accederemos al directorio donde lo hemos descargado y lo descomprimiremos usando: "*tar zxvf httpd-2_0_NN.tar.gz*".

3. Accederemos a la carpeta generada: "*cd httpd-2_0_NN.tar.gz*", y configuraremos Apache con las siguientes parámetros: "*./configure -prefix=/Apache -enable-so -enable-deflate -enable-ssl*".

4. Por último ejecutar en línea de comandos: "*make && make install*" para completar la instalación del servidor.

Si todo ha funcionado correctamente el servidor debe estar listo para funcionar en nuestro computador. Sin embargo, antes de ponerlo en funcionamiento debermos introducir unos cambios necesarios en su archivo de configuración que se encuentra en "*directorio_apache/conf/httpd.conf*", a continuación los detallamos:

1. En la sección *Dynamic Shared Object Support* debemos añadir la línea "*LoadModule php5_module modules/libphp5.so*", esto nos le permitirá reconocer el código PHP, algo esencial en nuestra aplicación.

2. Buscar la cadena *Addtype* y añadir al final de los tipos insertados que le siguen, los documentos PHP de la forma: "*AddType application/s-httpd-php .php .phtml*" y "*Addtype application/x-httpd-php-source .phps*"[4]. Esto permitirá a Apache reconocer los documentos a partir de la extensión del archivo.

[4]Este proyecto no cuenta con archivos .phtml ni .phps, pero es posible que sean usados en posibles ampliaciones, por lo que puede ser cómodo tener escrita por adelantado esta configuración.

3. Buscar el la cadena *Documentroot* y colocar aquí el directorio donde vayamos a guardar los ficheros de nuestra página web. A continuación habra que modificar más abajo la línea "*<Directory "/www/">*" sustituyendo *www* por el directorio que hayamos indicado en *Documentroot.*

4. La página *index.php* se encuentra entre las páginas por defecto que carga el servidor, por tanto debemos añadirlas. Para ello, localizaremos la cadena *DirectoryIndex* de forma que queda igual que: "*DirectoryIndex index.html index.html.var index.php index.htm*". Hay que destacar que el orden influye en el orden de prioridad del servidor a la hora de qué páginas buscar.

5. Asegurarse que el usuario del servidor, es el usuario apache (*daemon*). Si este usuario no existe en el sistema, añadirlo a los usuarios Linux.

6. Ya podemos arrancar nuestro servidor con el comando: "*directorio_ apache/bin/apachectl*".

A.3.1.2. MySQL

Para instalar MySQL, deberemos dirigirnos a su página oficial *http://www.mysql.com/* y dscargar los paquetes de servidor y cliente correspondientes para nuestra distribución de Linux. Los pasos a seguir en la instalación dependen de la distribución por tanto aconsejamos seguir las instrucciones de instalación que se ofrecen en la propia página. También aconsejamos la descarga de dos aplicaciones que no facilitarán la tarea a la hora de administrar nuestra base de datos que son: MySQl Query Browser y MySQL Administrator.

Una vez instalado el servidor, conviene añadir un password al administrador por razones de seguridad. Para ello, como *root* escribir: "*mysqladmin password cualquier_password*". Ahora lo que debemos hacer es introducir los esquemas en la base de datos que usará el sistema Maya.

A.3.1.3. PHP

Para la instalación de PHP y su compatibilidad con Apache y MySQL debemos seguir los siguientes pasos:

1. Accederemos vía web a la página *http://www.php.net/downloads.php* donde descargaremos la última versión de PHP. El sistema Maya ha sido probado con la versión 5.2.0.

2. Una vez se ha descargado el archivo, accederemos al directorio donde lo hemos descargado y lo descomprimiremos usando: "*tar zxvf php-5.N.N.tar.gz*".

3. Accederemos a la carpeta generada: "*cd php-5.N.N.tar.gz*", y configuraremos Apache con las siguientes parámetros: "*./configure –with-apxs2=/Apache/bin/apxs –with-mysql*".

4. Una vez configurado, procederemos a la instalación, usando: "*make && make install*".

5. A continuación debemos configurar el fichero *php.ini*, lo más cómodo es trabajar con el archivo de ejemplo que trae el propio paquete, para esto: "*cp php.ini-dist /usr/local/lib/php.ini*".

6. Deberemos cambiar el parámetro *doc_root* que se halla en el archivo de configuración, y escribir la ruta en la cual se guardan los sitios web a mostrar.

A.3.1.4. OpenSSL y Libssh2

Vamos a explicar el procedimiento a seguir para instalar estas librerías, necesarias para el uso de funciones SSH desde código PHP y para generar las claves pública y privada. OpenSSL suele estar disponible para todas las distribuciones de Linux como paquete instalable por el gestor de software de la distribución, sin embargo, explicaremos paso a paso como descargarlo desde la *web* ya que servirá para todos los usuarios.

1. Accederemos a página *http://www.openssl.org* de dónde descargaremos la versión correspondiente (la misma que dispongamos para descargar en el gestor de paquetes de OpenWRT, ello lo podemos comprobar en el punto A.3.2.3) en la sección *source*.

2. Descomprimiremos el fichero accediendo al directorio donde se encuentra y ejecutando "*tar zxvf openssl-X.X.X.tar.gz*".

3. Por último, situados dentro de la carpeta que hemos extraído teclearemos "*./configure && make && make install*".

4. Repetiremos exactamente los 3 primeros pasos, para el archivo *libssh2-X.X.tar.gz* que lo podemos obtener en la *http://sourceforge.net/projects/libssh2/*.

5. A continuación ejecutaremos el instalador *pear* para PCL/ssh2: "*pear install ssh2*", esto nos generará el archivo *ssh2.so*.

6. Copiaremos el archivo anterior al directorio especificado en nuestro fichero *php.ini* detrás de la etiqueta *extensions_dir*.

7. Por último añadiredemos al nuestro archivo *php.ini* la línea *extension=ssh2.o* detrás de la etiqueta *dynamic extensions* del fichero, gardaremos los cambios y reiniciaremos el servidor web (en nuestro caso *Apache*) en caso de que lo tengamos funcionando.

A.3.1.5. OpenWRT SDK

Recomendamos la instalación de esta utilidad para todo tipo de usuario ya que a parte de proporcionar muchas ventajas a los más expertos como la creación de sus propios paquetes de distribución o la creación de su propia versión de OpenWRT, tiene herramientas básicas como el compilador cruzado que podemos necesitar para recompilar código del programa a los que hemos modificado los parámetros. Los pasos a seguir son:

1. Descargar de *http://downloads.openwrt.org/whiterussian/newest/* el archivo *OpenWrt-SDK-Linux-i686-1.tar.bz2*.

2. Para finalizar, descomprimiremos el fichero (no hay que instalar nada) con el comando "*tar zxvf OpenWrt-SDK-Linux-i686-1.tar.bz2*".

A.3.1.6. AODV-UU

A continuación se detallan los pasos para la instalación del protocolo AODV-UU en nuestro ordenador:

1. Nos dirigimos a la página *http://core.it.uu.se/core/index.php/AODV-UU* dónde descargaremos la versión de AODV-UU más reciente. Maya ha sido probado con la versión 0.9.3.

2. Ejecutaremos el comando "*apt-get install kernel-source-2.6.8 kernel-headers-2.6-386*", esto nos permitirá instalar las cabeceras y fuentes del *kernel* 2.6 de Linux .

3. A continuación, descomprimiremos el archivo que acabamos descargar, para ello nos ubicaremos en el directorio dónde hemos guardado el fichero y ejecutaremos: "*tar zxvf aodv-uu-X.X.X.tar.gz*".

4. A continuación procederemos a la compilación del protocolo de encaminamiento para ello accederemos a la carpeta que se ha creado al descomprimir el archivo: "*cd aodv-uu-X.X.X*" y ejecutaremos el comando "*make && make install*"

5. Como resultado de la compilación obtendremos un archivo llamado *aodvd* y un módulo llamado *kaodv.o* o *kaodv.ko* dependiendo de la versión del kernel. El primero es el archivo ejecutable y se encargará de cargar el módulo en memoria. Por tanto para ejecutar el protocolo tan solo tenemos que ejecutar el comando "*aodvd*", automáticamente lo asociará a la interfaz dónde se encuentra nuestra tarjeta de red inalámbrica. En caso de que no asocie el protocolo a la interfaz que deseemos, podemos especificarselo usando "*aodvd -i nombre_interfaz*". Podemos también especificar otro tipo de parámetros, ejecutando "*aodvd –help*" la aplicación nos mostrará todas las opciones configurables.

A.3.2. Puntos de acceso

A.3.2.1. OpenWRT

Es altamente recomendado que instalemos la versión WhiteRussian RC6 de OpenWRT, ya que otra versión podría no ser totalmente compatible con nuestra aplicación. Para ello:

1. Accedemos a la *web http://downloads.openwrt.org/whiterussian/rc6/bin/* y descargamos el fichero binario *openwrt-wrt54g-squashfs.bin* que como su

nombre indica es para el router empleado por nosotros. En la misma página hay versiones binarias para otros modelos de *router*, por tanto elegiremos el que incluya el nombre de nuestro modelo.

2. A continuación debemos acceder via web a la página del *router* donde deberemos dirigirnos al apartado de *"system upgrade"* (el nombre puede cambiar dependiendo del *firmware* del punto de acceso) y escribir la ruta hasta el archivo binario que acabamos de descargar.

3. Por último para configurar los dispositivos con el objetivo de que trabajen en modo *ad-hoc* dentro de la misma red. Para ello, debemos dirigirnos a la sección *Network* dentro de la página web de configuración e indicar una dirección IP para el dispositivo ya que por defecto aparece la dirección 192.168.1.1 y si no la cambiamos nos encontraremos con más de un dispositivo con la misma dirección. También habrá que acceder a la sección *Wireless* y asegurarnos que está funcionando en modo *ad-hoc* así como asignarle el ESSID y el canal que deseemos para configurar nuestra red.

Este es el procedimiento más sencillo, aconsejado para la mayoria de usuarios. Para usuarios expertos que compilan su propio firmware con en el entorno de OpenWRT se recomienda el uso de un cliente TFTP para su instalación, en la dirección *http://wiki.openwrt.org/OpenWrtDocs/Installing* se explica el procedimiento a seguir.

A.3.2.2. AODV-UU

Si hemos copiado los archivo necesarios al punto de acceso como se indica en el punto A.2.1, contamos con una versión compilada del protocolo AODV-UU v0.9.3 para arquitectura MIPS. Por tanto, solo realizaremos la instalación que aquí se detalla en caso de que queramos usar otra versión de este protocolo o para otra arquitectura.

Para instalar el protocolo AODV-UU en los dispositivos debemos realizar los siguientes pasos:

1. El primer paso es averigüar la arquitectura de nuestro procesador en el dispositivo, muy probablemente sea MIPS (como en nuestro caso el Linksys WRT54G) por lo que necesitaremos hacer uso del compilador cruzado. Para ello debemos tener la ruta al directorio donde se encuentra el compilador cruzado en la variable PATH de nuestro sistema. El ejecutable del compilador se halla en la carpeta *staging_dir_mipsel* dentro de la carpeta de OpenWRT SDK que hemos generado en el punto A.3.1.5. Para arñadirlo al PATH ejecutaremos "*PATH=$PATH:/camino_hasta_SDK/OpenWrt-SDK-Linux-i686-1/staging_dir_mipsel*".

2. Realizaremos la compilación para mips, accediendo a la carpeta de AODV-UU (punto A.3.1.6) y tecleando: "*make mips*".

3. Ya hemos realizado la compilación, sólo nos falta copiar los archivos al dispositivo, para ello ejecutaremos el comando: "*scp aodvd kaodv.* ip.del.dispositivo.destino:/MAYA/aodvuu/*

4. Reiniciaremos el dispositivo, usando el comando "*reboot*".

Si hemos ejecutado el script *install.sh* (indicado en el punto A.3.1.6) y hemos reiniciado el dispositivo, el protocolo se cargará en memoria nada más arranque éste, por tanto, al ejecutar el comando *scp* el sistema nos indicará que dichos archivos se encuentran en uso. La solución será acceder al *router* via SSH: "*ssh ip.del.dispositivo.destino*", ejecutar el comando "*ps -e*" obteniendo como resultado la lista de procesos activos y localizar la ID del proceso *aodvd*. Por último, teclearemos la instrucción "*kill numero_de_proceso_de_aodvd*" y repetiremos el paso 3 y 4.

A.3.2.3. Libgcc y Libopenssl

Para la instalación de estos paquetes, es necesario que el *router* tenga acceso a internet. Para ello, es necesario que un dispositivo proporcione conectividad entre la red inalámbrica *ad-hoc* e internet de forma que pueda acceder a la información acerca de los paquetes acualmente disponibles.

Una vez nos hemos asegurado de esto, accederemos a la interfaz de configuración web de nuestro punto de acceso de la forma *http://ip.del.dispositivo.* accederemos a la sección *system* y dentro de ella a *installed software*. Acto seguido, pulsaremos en la opción *update packages lists* esto nos actualizará todos los paquetes disponibles para OpenWRT. Deberemos instalar los paquetes libgcc, libopenssl y openssl-utils. Para realizar esto, habrá que localizarlos en la lista y pulsar su correspondiente botón *install*.

Apéndice B

Manual del programador

B.1. Introducción

En este capítulo vamos a dar un repaso general a toda la estructura que compone la aplicación de forma que el usuario se familiarice con ella y sobre todo que sea capaz de orientarse a la hora de realizar modificaciones o ampliaciones en la herramienta.

B.2. Estructura de Maya

La herramienta se encuentra dividida en dos partes bien diferenciadas. Por un lado tenemos la parte del punto de control, donde se ubica la herramienta principal, con la interfaz al usuario y la base de datos y por otra parte encontramos la parte que se ubica en los dispositivos; scripts, procesos (cliente y servidor), etc.

B.2.1. Estructura de la herramienta principal

Como acabamos de mencionar la herramienta principal se instalará en el computador que haga las funciones de punto de control. Por un lado tendremos la interfaz web, implementada en PHP, con todos los ficheros necesarios contenidos en la carpeta *Maya_web*.

El fichero principal que carga la aplicación se llama *index.html* y a continuación vamos a detallar el resto de archivos se encuentran en la carpeta comenzando por los ficheros que manejan las distintas operaciones de la aplicación:

- *activateIP.php*: donde está implementada la funcionalidad de habilitar y deshabilitar un dispositivo.
- *changeparameters.php*: en este fichero se implementa toda la funcionalidad de cambio de parámetros excepto la habilitar/deshabilitar dispositivo.

- *deleteIP.php*: maneja la operación de borrado de un dispositivo de la lista.
- *insertIP.php*: se encarga de la inserción de un elemento de en la lista.
- *insertlist.php*: para la inserción de varios dispositivos cargados desde un fichero.
- *index.php*: el archivo principal de la aplicación.

También encontraremos varios ficheros con extensión *inc* que seran usados por los archivos que acabamos de mencionar con el objetivo de reutilizar código y hacerlo más sencillo de cara al usuario. Estos ficheros son:

- *array_ operations.inc*: se incluyen diversas funciones para manejar arrays.
- *dbopen.inc* y *dbclose.inc*: operaciones de abrir y cerrar la base de datos.
- *db_ operations.inc*: diversas funciones de consulta a la base de datos.
- *misc.inc*: funciones de diversa utilidad.
- *params.inc*: en este fichero encontraremos los parámetros que el usuario debe configurar para adaptar la aplicación a su computador.

Por último, hay una serie de ficheros que no foman parte de la interfaz web pero que son básicos en el correcto funcionamiento de esta utilidad, como son:

- *clientCP*: el proceso que se encargará de enviar los mensajes UDP para el cambio de parámetros en los dispositivos, es invocado desde el código PHP.
- *rsa.sh*: es el *script* que va a generar el par de claves pública y privada y el HIT del punto de control.
- *num_seq:* este fichero almacenará el último número de secuencia usado en los paquetes que se envía desde el punto de control.

En cuanto a la base de datos (de nombre *Maya*), podemos encontrar dos tablas que nos permiten almacenar toda la información correspondiente para cubrir las necesidades de la herramienta. La estructura y los campos de ambas tablas ya se describió anteriormente, concretamente en los puntos 4.6 y 4.7.

B.2.2. Estructura de los ficheros de los dispositivos remotos

Los archivos que deben ser copiados en los puntos de acceso que deseemos incluir en el sistema se encuentran dentro de la carpeta *MAYA*, dónde, a su vez, podemos encontrar cuatro subcarpetas:

- *cparam*: en este carpeta podemos encontrar los *scripts* implementados en comandos *shell* de Linux , que se encargan de realizar los cambios en la configuración. Hay un archivo por cada uno de los parámetros a cambiar.
- *init_scripts*: aquí se encuentran los *scripts* que ejecutará el punto de acceso cuando sea arrancando, también se encuentra el fichero *install.sh* que se encarga de configurar el dispositivo para que ejecute los scripts incluidos en este carpeta cuando éste arranque.
- *server*: en este directorio se hallan los ficheros relacionados con el aspecto de seguridad (claves pública y privada, HIT, etc...) y los procesos servidor y cliente para el sistema de mensajes UDP.
- *aodvuu*: aquí encontraremos los dos archivos necesarios para el funcionamiento del protocolo de encaminamiento en nuestro dispositivo, el ejecutable (*aodvd*) y el módulo que se carga en memoria (*kaodv.o*).

B.2.3. Archivos fuente

Además de los ficheros del punto de control que se situan en la carpeta *Maya_web* y de los de los puntos de acceso que se hallan en *MAYA,* existe otra carpeta en el directorio del proyecto llamada *sources*. Donde encontraremos los fuentes de AODV-UU y de los procesos escritos en lenguaje C para el envío de mensajes UDP. También encontraremos una versión ya compilada, tanto del protocolo de encaminamiento como de los archivos .c tanto para arquitectura x86 como para MIPS.

B.3. Ampliación de la herramienta

En esta sección explicaremos los pasos fundamentales a realizar para realizar alguna modificación en la herramienta, por ello se recomienda su lectura a todo usuario que desee implementar alguna nueva funcionalidad o modificar las ya existentes.

B.3.1. Añadir un nuevo parámetro

Quizá esta sea la acción más común que se pueda realizar sobre la aplicación. Añadir un parámetro a la oferta de los ya disponibles para ser configurados es una operación sencilla aunque quizá algo costosa, debido en gran parte a la diferente naturaleza de los parámetros y que implica modificar los procesos servidor y cliente que implementan la funcionalidad de los mensajes UDP para soportar el nuevo parámetro.

Las acciones que deberemos llevar a cabo son las siguientes:

1. En la interfaz de usuario (*index.php*) deberemos, añadir al desplegable donde se muestra la lista de parámetros, el nombre del parámetro que deseamos añadir.

2. Además, deberemos conocer si queremos que nuestro parámetro sea global, en caso afirmativo deberemos invocar a la función *javascript* del mismo documento llamada *check_all_disabled()* desde el atributo *OnClick* del elemento que hemos añadido, que obligará al usuario a seleccionar todos los dispositivos de la lista. En caso de que no sea global deberemos invocar a la función *check_none()* para permitir que el usuario los seleccione o no a su gusto. Por último, en caso de que tengamos que controlar alguna restricción del valor del parámetro introducido, deberemos indicarlo en la función *validate_option(),* dónde crearemos una nueva entrada *if*, con el valor de nuestro parámetro para comprobar si está seleccionado e introduciremos las instrucciones necesarias para controlar las restricciones.

3. A continuación, abriremos el archivo *changeparameters.php*, dónde añadiremos en el *switch* de selección de parámetro, el nombre del que hemos elegido y deberemos indicar el valor para los siguientes parámetros: *$OpenWRT_param* (nombre del parámetro en el sistema OpenWRT), *$reboot_necesary* (deberemos asignarle el valor *yes* o *no* dependiendo de si es necesario reiniciar el dispositivo tras realizar el cambio) y *$global* (valor yes o no dependiendo de si es un parámetro global o no).

4. Para finalizar los cambios en el código PHP, no dirigiremos a la parte final del archivo mencionado en el paso anterior y, sólo en caso de que sea un parámetro global, deberemos indicar el comando necesario para cambiar ése parámetro en el punto de control mediante línea de comandos, los que ya aparecer implementados servirá de buen ejemplo para el lector.

5. El siguiente cambio lo deberemos realizar en el archivo *listener.c*, dónde deberemos declarar, de la misma forma que esta hecho para los demás parámetros, un array con la ruta dónde ubicaremos el script que se encargará de configurar el parámetro que hemos creado. Esto implicará la recompilación del archivo y su posterior actualización en todos los dispositivos del sistema.

6. En ese mismo archivo, deberemos añadir una sentencia *"strcmp(rec_param, "Nombre_de_nuevo_parámetro")*" en la parte dónde se compara el campo parámetro del mensaje UDP con todas las posibilidades existentes, dónde el nombre del nuevo parámetro es el mismo que le asignamos a la opción del desplegable en el punto 2.

7. Por último, sólo quedaría crear el archivo *.sh* que se encargue de la configuración del parámetro y ubicarlo en la ruta que hemos indicado en el punto 5. En este fichero, debemos incluir la instrucción para actualizar el parámetro en la NVRAM del dispositivo y realizar un *commit* sobre ella. En algunos casos es posible que tegamos que reiniciar la interfaz *Wi-Fi* (*"/sbin/wifi"*) o incluso el dispositivo (*"reboot"*).

B.3.2. Añadir un nuevo campo al mensaje UDP

La modificación de la estructura del paquete, puede ser otra modificación bastante probable en la herramienta debido a nuevas necesidades o ampliaciones del sistema, por ello vamos a enumerar los pasos a seguir:

1. En primer lugar modificaremos el código PHP, para ello nos dirigiremos al fichero *changeparameters.php*, en el primer bucle *while*, dónde ejecutamos la instrucción para enviar un mensaje UDP a los dispositivos, deberemos añadir un nuevo parámetro para el proceso *clientCP*. Lo mismo debe hacerse en el fichero *activateIP.php*.

2. A continuación deberemos modificar los procesos, como observaremos, será importante en la posición que coloquemos el nuevo campo y si va a pertenecer a la parte cifrada o sin cifrar. En primer lugar, abriremos el fichero *client.c*. Si se trata de un campo que viajara en la parte cifrada del paquete, tendremos que añadirlo al string *data.message* de la forma que se hace con el resto de parámetros. En caso de ir en la parte sin cifrar, no tenemos que modificar nada, ya que el *string* que almacena todos los argumentos sin cifrar ya tiene espacio de sobra para albergarlos. Sin embargo, en ambos casos deberemos asegurarnos que el argumento número 7 sigue siendo la dirección de broadcast de la red, en caso contrario se debera modificar la línea "*he=gethostbyname(argv[6])) == NULL*", sustituyendo el 6 por el número de argumento dónde se encuentre la dirección broadcast menos 1.

3. En el archivo *clientCP.c*, habrá que realizar en primer luegar los mismos cambios que en el punto anterior. Posteriormente habrá que comprobar que el HIT sigue en la posición en *argv[7],* de lo contrario habrá que cambiar la línea "*pub_keyfile = fopen(argv[7], "r")) == NULL*". Por último habrá que modificar el *sscanf* y el *sprintf*, que usamos para leer y escribir la información respectivamente, con los nuevos argumentos.

4. Por último en el *server.c* deberemos, al igual que en *clienCP.c*, modificar el número de parámetros en el *sscanf* cuando recibimos un mensaje y en el *sprintf*, con el cual invocamos al proceso cliente.