Universitat Autònoma
de Barcelona

DESARROLLO E IMPLANTACIÓN DE UN ENTORNO WIFI ABIERTO

Memoria del proyecto de
Ingeniería Técnica en
Informática de Sistemas

Realizado por
José Carlos Mut Rojas
y dirigido por
David Megías Jiménez

Escuela Universitaria de Informática
Sabadell, junio de 2009

El firmante, David Megías Jiménez,

profesor de la Escuela Universitaria de Informática de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria
ha sido realizado bajo su dirección
por José Carlos Mut Rojas

Y para que conste firma la presente.
Sabadell, junio de 2009

Firmado: David Megías Jiménez

Resumen

Un hecho que no se puede negar es que vivimos en una sociedad donde la actualidad informativa y la comunicación en tiempo real es una prioridad y en algunos casos casi una necesidad, con la aparición de Internet se extendió la posibilidad de ese tipo de comunicación a la mayoría de la población y ha pasado a ser un evento tan cotidiano como tomar un café.

La sociedad cambia y exige nuevas necesidades, por lo tanto, también tienen que evolucionar las redes para sufragar estas nuevas demandas. Cada vez más se necesita estar conectado aunque tu ubicación cambie o donde las infraestructuras existentes no son suficientemente buenas. De este hecho surgen nuevas topologías de red donde se dejan atrás la dependencia de los puntos de acceso para dar paso a una red muy adaptativa tanto al escenario como a los usuarios que junto con los protocolos de encaminamiento dinámicos de nueva generación integrados en *OpenWRT* hacen una red mallada más versátil y estable ya que cada nodo se conecta con todos los nodos que tiene a su alrededor y existe más de una ruta para llegar al destino.

Este proyecto pretende abarcar este tema tan actual junto con la colaboración de la red *guifi.net*, la mayor red de telecomunicaciones inalámbricas libre, abierta y neutral del mundo que se construye a partir de un acuerdo de interconexión entre iguales.

Citada la motivación y contexto que implica este proyecto los objetivos principales que se persiguen son:

- Personalización de un *firmware* con *OpenWRT* para un tipo de nodo inalámbrico con una serie de servicios y características determinadas integradas en el propio sistema.

- Integración del sistema a la red *mesh* de *guifi.net*

- Sistematización y documentación de todo el proceso para que sea extensible a otras personas con las mismas inquietudes.

Tabla de Contenidos

1. Introducción

1.1. Presentación

En este proyecto se desarrolla un entorno con conectividad inalámbrica basada en *Wi-Fi*, sistema de envío y recepción de datos en redes computacionales que utiliza ondas hertzianas para la comunicación y esta acotada por la *Wi-Fi Alliance* bajo el estándar 802.11.

Una de las grandes ventajas de este entorno es la utilización de una topología de red especialmente diseñada para entornos móviles o donde no es posible un sistema de infraestructura estable. Dicha topología denominada *mesh* o red mallada se sustenta en una serie de protocolos de encaminamiento dinámicos especialmente diseñados que harán una red muy versátil y estable. Para conseguir esto se utilizan nodos, elementos de red que en este caso son tanto de acceso a la propia red como de conmutación que permitirán recibir y encaminar las comunicaciones. A estos nodos se les suministra un *firmware*, soporte lógico de bajo nivel que interactúa con el soporte físico cuya interfaz controla la ejecución correcta de las instrucciones externas.

Este *firmware* es implementado bajo *OpenWRT*[5], distribución de *GNU/Linux* para dispositivos empotrados que junto a una serie de protocolos de encaminamiento y utilidades dotan al sistema con capacidades de conexión a una red *mesh* como la que tienen creada *guifi.net*[7].

Como se describe en esta memoria, este proyecto tiene un gran abanico de posibles funcionalidades y ampliaciones ya que al ser en el *firmware* donde están especificadas todas las características, el cliente puede moldear el sistema a su gusto para conseguir las expectativas deseadas para su uso tanto lucrativo como al contrario. En este aspecto la red a la que se da apoyo es una red de libre acceso.

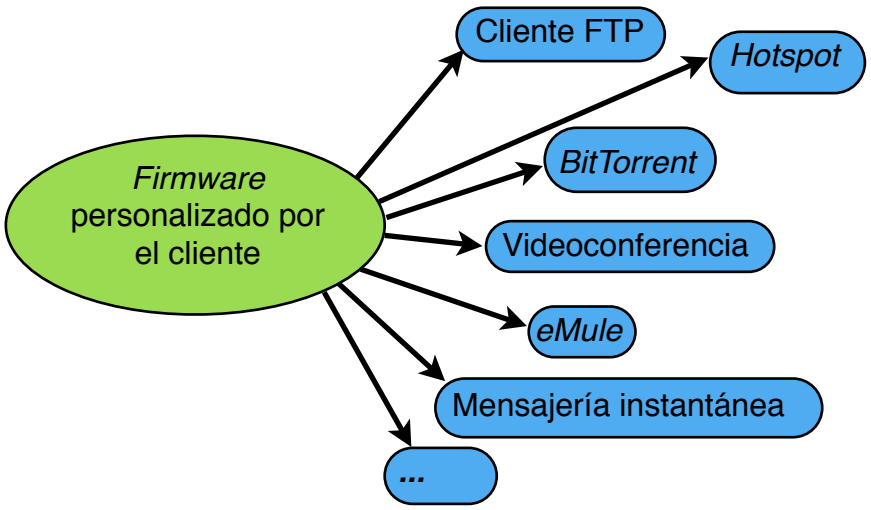


Figura 1.1: Posibles servicios

1.2. Objetivos

Los objetivos de este proyecto a un nivel estructural son generar el entorno inalámbrico de comunicaciones para que pueda ser integrado en la red de *guifi.net*. Este entorno al ser tan moldeable se puede construir dependiendo de las necesidades que se exijan, en nuestro caso se crea un entorno con conectividad a una red *mesh* con unas configuraciones y patrones de direccionamiento determinados por la red de *guifi.net*. A todo esto se le dota de utilidades de control, test y experimentación sobre redes para así crear una base donde trabajar en futuros proyectos

Este *firmware* se utilizará en nodos propiamente construidos y específicos para el sistema que se genera, que en conjunto, todo el sistema será capaz de implantarse en el sistema global de la red *mesh* de *guifi.net*.

Los propósitos descritos anteriormente hacen referencia directa al éxito del proyecto, pero inherentes a éstos se ocultan los referidos al proyectista y no por ello menos importantes, empiezan por la asimilación y aprendizaje por parte del proyectista de una serie de conocimientos primeramente sobre la puesta en funcionamiento de un proyecto real de informática así como su gestión, seguidamente de conocimientos aprendidos en diferentes asignaturas de la carrera para su aplicación real y obviamente otros conocimientos más técnicos y específicos sobre redes, comunicación inalámbrica, sistemas *UNIX* y telecomunicaciones que serán esenciales para la finalización del proyecto y adjudicarán al proyectista una serie de conocimientos muy valorados por él, que de otra forma hubiesen sido improbables de aprender. Finalmente conllevará el éxito del proyecto a la obtención del título de ingeniero técnico de sistemas al proyectista.

Como se describe anteriormente, la unión de todos estos factores genera un proyecto final de carrera muy interesante tanto por la parte del proyectista como también de los interesados en dicho proyecto que se publicará en *guifi.net* para que cualquier usuario pueda hacer uso de él y le sea una guía para poder construir su propio entorno inalámbrico abierto.

1.3. Organización de la memoria

Esta memoria consta de seis capítulos y tres anexos

En el primer lugar tenemos un breve resumen sobre la esencia del proyecto con sus implicaciones y seguidamente se detalla la tabla de contenidos de la memoria.

Seguidamente en el primer capítulo se realiza una introducción al proyecto y así iniciar al lector en el tema que se está tratando y darle una primera visión de los aspectos y factores que implican este proyecto.

En el segundo capítulo se analiza el sistema que existe actualmente para el mismo propósito y donde se encuentra un estudio de viabilidad del proyecto donde se analizan diferentes factores que le repercuten para evaluar la conveniencia de la realización del proyecto.

El capítulo tercero recoge el diseño del nuevo sistema tanto un análisis exhaustivo de requerimientos como también el diseño del *firmware* y la adecuación del equipo electrónico para su correcto funcionamiento.

En el cuarto y quinto capítulo se describen los pasos de implementación de *scripts* y archivos del sistema junto con la implantación, funcionamiento, pruebas, análisis y demás pasos necesarios.

En el último capítulo se presentan las conclusiones extraídas una vez realizado el proyecto.

En lo que respectivo a los anexos, el primer anexo pretende explicar brevemente los protocolos de encaminamiento que se utilizan en este proyecto, el segundo anexo se muestra el código del *script* de generación del *firmware y* el tercer anexo muestra los *scripts* utilizados para efectuar las pruebas de rendimiento de los protocolos de encaminamiento.

Finalmente, se describirán las fuentes de información utilizadas en la sección de bibliografía y por ultimo un agradecimiento a todas las personas que han colaborado en el proyecto.

2. Análisis del sistema actual

2.1. Introducción

La red de telecomunicaciones construida por *guifi.net* a lo largo del territorio catalán funciona actualmente con un modo de operación infraestructura en donde cada cliente de la red se ha de conectar forzosamente a un punto de acceso (*AP*) y este último se encarga de gestionar la recepción y envío de datos a través de la red. La topología de red mayoritaria es mixta centralizada, esto quiere decir que conviven topologías en estrella y árbol como se observa en la figura 2.1 junto con una infraestructura de supernodos que interconectan nubes de nodos para así crear y extender esta red. Esto hace necesario la utilización de puntos de acceso a la red y nodos troncales donde su dependencia a ellos ocasiona que si uno de estos falla, las conexiones de los clientes conectados a él se perderán como se observa en la figura 2.2

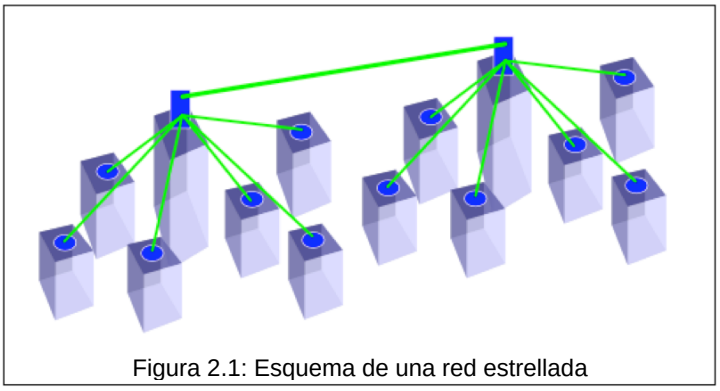


Figura 2.1: Esquema de una red estrellada

Otro de los problemas que surgen actualmente es en sitios donde la abundancia de clientes hace que el punto de acceso al que se conecten quede saturado. La solución actual es aumentar el número de puntos de acceso. Por todos estos problemas se buscan nuevos modos de operación como el *Ad-Hoc,* donde se actúa como en una red punto a punto entre un conjunto de clientes o nodos. Con esta nueva incorporación también surgen nuevas topologías de red como las redes malladas que no solo posibilitan solucionar los problemas actuales sino también abren nuevas vías de conexión a dispositivos móviles como portátiles o PDAs.

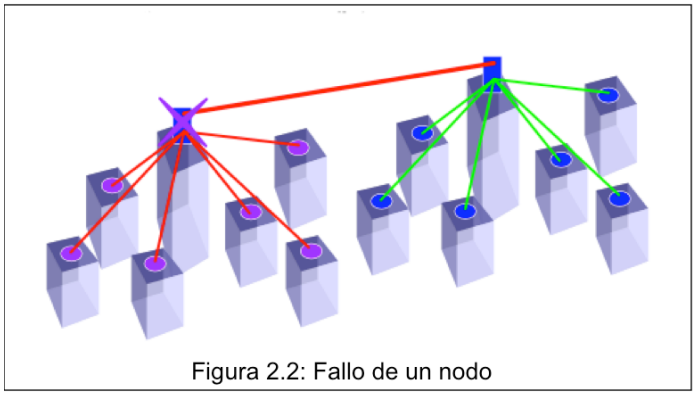


Figura 2.2: Fallo de un nodo

Las conexiones inalámbricas de la red de telecomunicaciones actual se sitúan en los puntos más altos posibles ya que se necesita una visión directa entre los dos puntos a conectar. Esto implica que la ubicación de todos los nodos se sitúa en terrados, tejados, campanarios o lugares estratégicamente visibles.

Hay que tener en cuenta también el sistema de direccionamiento que se utiliza actualmente para así asegurar la convivencia del nuevo sistema al actual sin problemas.

2.2. Estudio de Viabilidad

2.2.1. **Introducción**

El proyecto que se desea realizar es la implementación, en un *router* de una distribución de *GNU/Linux* para dispositivos empotrados llamada *OpenWRT*. Dicho sistema es utilizado en la red de telecomunicaciones abierta *guifi.net* debido a su inmenso potencial como *firmware* en los nodos y a su libre distribución.

Con este proyecto se pretende sistematizar el proceso de implementación de *OpenWRT* a un *router* para así llevarlo a cabo por los usuarios de *guifi.net* sin un conocimiento demasiado avanzado sobre este sistema. Las ventajas de esta distribución de *GNU/Linux* son muy amplias, pasando desde el coste nulo que requiere integrarlo, debido a que su distribución es libre, como también la posibilidad de personalización tan grande como para alcanzar los requisitos deseados, aun siendo más avanzados o extensos que los que el propio fabricante del *router* establezca para ese dispositivo. No sólo con esto también es una plataforma idónea para la experimentación de nuevos tipos de red y configuraciones.

El tipo de estructura de red al que se pretende agregar este nodo una vez en funcionamiento con *OpenWRT* es una red *mesh* o mallada. Este tipo de red es ideal para entornos con muchos cambios en los nodos ya que hereda las ventajas de una red tipo infraestructura y además las de tipo *ad-hoc*, siendo así una candidata perfecta en redes inalámbricas donde los otros dos tipos de estructura fracasarían.

El objetivo una vez expuesto esta pequeña explicación será implementar este *firmware* para un tipo de *router* el cual tendrá una serie de características y servicios donde tendrán que funcionar correctamente en una red inalámbrica *mesh* de estándar 802.11.

2.2.2.**Objeto**

2.2.2.1.**Estado del arte**

En la actualidad existen muchas empresas dedicadas al diseño y fabricación de *routers*, las cuales, crean su propio *firmware* para sus productos dotándoles de funciones y características especificas para cada modelo. Muchas de estas funciones están por debajo del rendimiento real del *hardware*, esto se debe a varios factores tales como comercializar con el mismo diseño varios modelos (*marketing*), alargar la vida útil del *hardware* o evitar el colapso del sistema. Debido a que la mayoría de estos dispositivos se comercializan con el sistema operativo del fabricante esto implica asumir las restricciones que este impone con sus correspondientes desventajas que se explicaran en este estudio.

Personalizar y optimizar el *firmware* del *router* es el paso que se quiere dar con la implantación de *OpenWRT* ya que en la actualidad el *firmware* que se utiliza en *guifi.net*, es en gran mayoría software propietario.

2.2.2.2.**Perfil del cliente/usuario**

El cliente de este proyecto es una comunidad de personas que se unen para crear una red de telecomunicaciones abierta, por este hecho el perfil del cliente es el mismo que el del usuario final, actualmente un usuario con un nivel medio/alto de conocimientos tanto informáticos (concretamente sobre *GNU/Linux*) como también sobre redes y telecomunicaciones.

Por este motivo se quiere sistematizar la implementación para abarcar a un número más extenso de usuarios con un nivel más bajo de conocimientos.

Estos usuarios son los que en definitiva extenderán la red hacia nuevas ubicaciones.

El abanico de usuarios o clientes potenciales de este proyecto es muy grande ya que permite a cualquier dueño de un *router* liberarse de las limitaciones implantadas por el fabricante y conseguir funcionalidades superiores. También se podría optar por utilizarlo como herramienta de captación de nuevos clientes de empresas (*Hotspot*) configurándolo para hacer publicidad sobre sus productos o darse a conocer en la red creada por estos dispositivos, los cuales se colocarían en puntos estratégicos para que conectasen usuarios de forma gratuita o en contra con ánimo de lucro.

Es cierto que a priori no existe una interfaz gráfica agradable como tienen otras aplicaciones de similares características, por este motivo se pretende guiar al usuario y si en caso de que este quiere extender, analizar o optimizar algún protocolo o servicio tenga el entorno y las herramientas necesarias para hacerlo sin ningún problema debido a la alta escalabilidad del sistema.

2.2.2.3.**Objetivos**

Disponer de un *firmware* para un *router* estándar el cual funcione perfectamente con las características y configuraciones de la red *guifi.net* tipo *mesh* así como también tener una serie de aplicaciones y protocolos operativos.

Estos objetivos son:

✳ Definición y especificación del *firmware*

✳ Clasificación y elección de paquetes integrados en el sistema

✳ Implementación del *firmware* en una imagen

✳ Integración de protocolos de encaminamiento dinámicos

- *BMX* (*B.A.T.M.A.N. Experimental*)
- *BATMAN* (*Better Approach To Mobile Ad-hoc Networking*)
- *OLSR* (Protocolo de Encaminamiento por Optimización del Estado del Enlace)

✳ *DHCP* (Protocolo de Configuración Dinámica de Anfitrión)

✳ Configuración de diversos servicios del sistema

✳ Herramientas de monitorización de red

✳ Herramientas de visualización de red

✳ Interfaz Web

✳ Sistematización y documentación de todo el proceso de implementación

En esencia lo que se quiere conseguir es ayudar a implantar este sistema a los nodos de la red de telecomunicaciones con pasos claros y guiados. Así un usuario que quiera llevar a cabo la implantación en su *router*, podrá saber en todo momento que hace y porqué lo hace.

2.2.2.4.**Fuentes de información**

En la época en que vivimos la información es instantánea, nos envuelve y también nos satura, por esto se pude encontrar una infinidad de documentos sobre *OpenWRT, scripts* para este sistema, archivos de configuración o modificaciones hechas por otras personas. Uno de los motivos de por qué hallaremos más información sobre *OpenWRT* en Internet es debido a que el sistema es de distribución libre con licencia GPL y muchas comunidades se han interesado y han comenzado a hacer su propio *firmware* para sus dispositivos y crear sus propias redes.

Por lo tanto, tener contacto con estas comunidades, como puede ser *guifi.net*, tiene que ser esencial para absorber la mayor cantidad de conocimientos sobre este tema que como ya se ha descrito anteriormente es experimental y lo que en un principio se intenta conseguir al final puede cambiar totalmente.

No sólo tendremos que documentarnos sobre este sistema sino también sobre el funcionamiento, configuración y mantenimiento que se da en el medio de comunicación sin cables en que nos encontramos, el estándar de comunicaciones inalámbricas *IEEE 802.11* así como sus variantes. Dado que es un estándar podremos encontrar mucha bibliografía sobre este tema. Por otro lado se analizará documentación sobre telecomunicaciones ya que su ubicación final será en una red de esta naturaleza.

Un factor a tener en cuenta para completar la documentación será la legislación que rige tanto la ley general de telecomunicaciones vigente sobre la auto-prestación de servicios, como el cuadro nacional de atribución de frecuencias del Ministerio de Ciencia y Tecnología para la distribución de nuestra señal y así cumplir con el marco legislativo vigente.

Finalmente, y aunque se ha hecho reseña anteriormente, la mayor fuente de información se obtendrá de los conocimientos de la comunidad *guifi.net* ya que están en una situación de investigación y desarrollo para la creación de su red *mesh* de libre acceso en la ciudad de Barcelona.

2.2.3.**Descripción del sistema**

2.2.3.1.**Descripción**

Crear un *firmware* para un *router* sobre *OpenWRT* con las funciones de conectividad inalámbrica especificada por *guifi.net* y que más adelante se explicarán.

Actualmente en el mercado existen multitud de fabricantes de estos dispositivos, pero nos centraremos en uno concretamente por cuestiones diversas tales como la relación calidad/precio sobre los demás, facilidad de implantación, ya que esta plataforma viene de serie sin sistema operativo, y su posibilidad de ampliación del equipo electrónico.

Figura 2.3 - Foto de una placa *Alix*.2d2

La plataforma elegida será *Alix* de *PC Engines*[1], concretamente el modelo *Alix. 2D2*. Dicha *routerboard* con unas características tales como un procesador de 500 MHz (*AMD Geode LX800*), 256 MB de memoria *DRAM*, dos interfaces de red *ethernet*, dos puertos *miniPCI* y dos puertos *USB* le hacen un candidato perfecto para nuestro proyecto. Unido a lo anterior se le instalarán 2 radios *802.11ABG* de 200mW con sus respectivas antenas y complementos que se describirán más adelante.

En la definición, especificación, elección de paquetes e implementación del *firmware* se tendrán en cuenta objetivos como el arranque del dispositivo correctamente, funciones de configuración de servicios como *SSH (Secure Shell)*, *Hotspot*, *DHCP*, conectarse a la red *mesh* de *guifi.net* y a su estructura mediante los protocolos de encaminamiento descritos anteriormente. Por la naturaleza de la red que se muestra en la figura 2.4, esta implementación de los tres algoritmos de encaminamiento tiene que ser muy estudiada para que funcionen correctamente y en caso de fallo de alguno, la sustitución de éste por otro algoritmo.

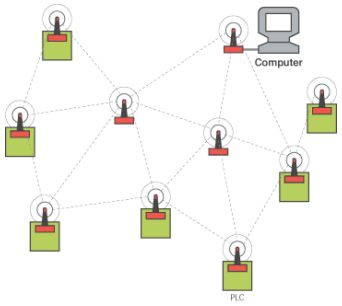


Figura 2.4 - Red *mesh* o mallada

Sobre la interfaz web del sistema operativo se utiliza siempre un servidor *HTTP* (Protocolo de Transferencia de Hipertexto) para visualizar de manera amigable y con la posibilidad de varios idiomas para la administración y configuración del sistema operativo. Para conseguir esto se utiliza *luci* que es una sintaxis de *scripts* (con el potencial añadido del lenguaje de programación de alto nivel *LUA*) para asignar o modificar las configuraciones de todos los servicios y hasta del propio sistema.

2.2.3.2.**Modelo de desarrollo**

Debido a que la naturaleza del desarrollo del proyecto es experimental y no se ha realizado nunca, se tendrá que ir haciendo análisis e ir construyendo el sistema para después analizar su resultado en la máquina así como el enlace entre otros nodos cercanos. Este factor nos hará seguir un modelo de desarrollo evolutivo donde iremos incrementando los diversos objetivos de menor a mayor dificultad y crear diversos sistemas, este modelo también afectará positivamente a la documentación de los pasos seguidos para su creación. También nos hace pensar que se utilizará el modelo de prototipos, donde se verá más claramente y con un mejor seguimiento el éxito de los objetivos, la viabilidad de estos junto con la documentación y explicación que corresponderán a cada punto.

Se han descrito dos modelos de desarrollo que se creen que serán los idóneos para el proyecto, la conclusión de esto es seguir una fusión de modelos entre el diseño evolutivo y el de prototipo para conseguir la simbiosis perfecta de desarrollo.

1 http://www.pcengines.ch/

2.2.3.3.**Requerimientos funcionales**

- Construcción del *firmware*
- Elección de paquetes
- Encapsulado en imagen
- Servicio de *DCHP*.
- Integración de protocolos de encaminamiento dinámicos de nueva generación
- Herramientas de monitorización y visualización de la red
- Interfaz web

2.2.3.4.**Requerimientos no funcionales**

- Concurrencia de clientes
- Ejecución en *GNU/Linux*
- Criterio de asignación de direcciones *IPv4*
- Componentes hardware del nodo
- Situación geográfica
- Normas de conexión
- Interfaces externas
- Salto ente nodos
- Mantenimiento del sistema
- Escalabilidad
- Funciones de los usuarios

2.2.3.5.**Recursos**

Antes de tratar este apartado se debe comentar que los recursos que se expondrán, en un futuro pueden ser sustituidos por componentes mejores sin que repercuta en el coste, debido a que los fabricantes sacan frecuentemente actualizaciones de sus dispositivos. Por este motivo, en el momento de su implementación, se deberá estudiar el mercado para elegir el candidato idóneo y así prevenir el riesgo de quedar obsoleto.

Equipo electrónico

- *Router* exterior
  - Placa Base *ALIX.2D2*
  - Transformador 18 v. 0,8 A (15 W)
  - 2 Radios 802.11AGB 200mW 108 Mb/s
  - 2 Conectores *Pigtail* 5 GHz. UFL-N de 30 cm
  - 2 Antenas dipolo 2.4/5 GHz. 5 dBi Connector N
  - Caja estanca de poliéster con tapa
  - 1 Tarjeta *CompactFlash* de 512MB o superior
- Grabador de tarjetas de memoria con soporte *CompactFlash*

Soporte lógico

- Sistema Operativo *GNU/Linux*
- Herramienta de compilación *GCC*
- *OpenWRT* en su última versión 8.09 *Kamikaze*
- Navegador web *Firefox* v.2 o similar

Recursos humanos

Al comienzo de la implantación se tendrá que poner en contacto con los miembros de la comunidad *guifi.net* para el alta de su nodo y asignación de *IP* pública así como el nivel de estructura que tendrá el nodo.

Cliente/usuario

- El usuario tendrá un ordenador para la generación del *firmware* y su utilización con los requisitos de soporte lógico anteriormente descrito.
- Se requerirá conexión a Internet para la descarga de *Kamikaze* 8.09 y paquetes adicionales.
- El usuario tendrá que estudiar la ubicación del dispositivo para la correcta visibilidad de la red.

2.2.3.6.**Evaluación de riesgos**

Las dificultades que se prevén en un futuro principalmente recaen en el proyectista ya que se ha de asimilar muchos conocimientos técnicos sobre redes y telecomunicaciones inalámbricas no adquiridos hasta la fecha y por lo tanto el factor de la curva de aprendizaje es el más importante si se quiere llegar a la finalización del proyecto en las fechas establecidas.

Por otro lado también se destaca que estamos hablando de un entorno experimental, es decir, este tipo de redes con esta tecnología nunca se ha realizado, por lo tanto surgen dos consecuencias, la primera es que no hay una documentación clara sobre lo realizado. Es cierto que en otras ubicaciones geográficas ya existen este tipo de redes pero su implementación es caótica y con la única máxima de “si funciona déjalo como está”.

La otra consecuencia viene dada porque no se tiene la completa seguridad que funcione la red correctamente en algún caso de uso o entorno y su tratamiento en estos casos puede ser de una dificultad extrema debido a la complejidad de la estructura de la red o a factores externos.

Otro riesgo que se plantea al proyectista se ha expuesto en el párrafo anterior sobre la experimentación de estas redes en otros puntos geográficos que trae otra dificultad añadida. La documentación que se puede obtener esta en otros idiomas generalmente en inglés pero una gran parte está en alemán ya que uno de los focos donde se ha experimentado con mayor fuerza y hay más experiencia de implantación es la ciudad de Berlín. Esto conlleva que no siempre se pueda entender correctamente las anotaciones y *scripts* estables que utilizan.

Finalmente, dejando de lado los grandes riesgos descritos anteriormente, además nos encontramos con las dificultades que recaen sobre la propia red o el dispositivo. El *router* será exterior y eso conlleva que las adversas situaciones climatológicas le afecten negativamente hasta incluso dejando de funcionar, aunque este riesgo sea mínimo por el aislamiento que ofrece la caja estanca de poliéster. Este factor también le afecta a la red ya que el medio en el que viaja es el exterior, asimismo la distancia entre los nodos, la calidad del enlace, los ruidos externos producidos por otros dispositivos, o una ubicación inapropiada afectará directamente a la calidad de la conexión a la red *mesh* y es un factor muy a tener en cuenta.

2.2.3.7.**Organización del proyecto**

Para la realización del proyecto se seguirán estas pautas para su correcta realización y planificación.

- Reunión con el tutor del proyecto para encontrar una buena armonía entre los objetivos que se incluirán en el proyecto y la capacidad de asimilación de estos.

- Reunión con la comunidad *guifi.net* para la iniciación a estas redes y los requisitos que se exigirán.

- Realización de un estudio de viabilidad del proyecto.

- Planificación de las etapas del proyecto así como las reuniones con los diferentes implicados en el proyecto.

- Ejecución de estas etapas teniendo en cuenta el factor experimental y novedoso del proyecto que podría implicar cambios tanto en las etapas como en la implementación.

- Reuniones continuas con no solo esta comunidad sino otras con redes parecidas para la adquisición de nuevos conocimientos, métodos de implementación, nuevos avances sobre estas redes y análisis de los prototipos que se construirán.

2.2.3.8.**Análisis de costes/beneficios**

Los costes para la realización no dejan de ser exclusivamente de compra de los componentes del *router* ya que los costes del software son nulos y con los pasos guiados como se darán la implementación será sencilla. El gran abanico de beneficios que aporta este proyecto pasa por todas las ventajas de una red de comunicaciones como son la posibilidad de intercambio de información, comunicación con otros usuarios, etc. Por la topología de red hace posible la conexión en áreas que otros tipos de topología fracasan por motivos tales como que la infraestructura no llega o es imposible realizarla.

Una red abierta tiene además las ventajas de que se puede saber cómo se ha creado, su estructura y en general, al no pertenecer a ningún propietario, cada uno la puede utilizar a su conveniencia y sin restricciones.

No hace falta decir que la conexión a estas redes es gratuita y sus recursos accesibles a todos.

2.2.3.9.**Presupuesto**

Concepto	Cantidad	Precio
ALIX.2C2 system board (LX800 / 256 MB / 2 LAN / 2 miniPCI / USB)	1	97,80 €
Power Supply 18 v. 0,8 A (15 W) ALIX 2C/3C	1	5,78 €
Compex WLM54SAG23 802.11AGB 200mW 108 Mb/s	2	33,30 €
Pigtail 5 GHz. UFL-N Jack Bulkhead 30 cm	2	4,70 €

Concepto	Cantidad	Precio
2.4/5 GHz. 5 dBi Dipole Antenna N Connector	2	12,20 €
Caja estanca externa de poliéster con tapa	1	35,12 €
Lector multitargeta SITECOM MD 014	1	12,95 €
CompactFlash 512MB	1	5,44 €
Total*		257,49 €
I.V.A. (16 %)		41,20 €
Total Presupuesto**		298,69 €

*Se tendría que incluir soporte para la colocación del mismo en la ubicación seleccionada con sus respectivos anclajes. Como depende de la ubicación no se tendrán en cuenta.

**Conceptos y precios extraídos de la tienda *on-line* especializada “Landashop”[2]

2.2.3.10.**Viabilidad legal**

Sobre la legislación actual que afecta directa o indirectamente a este proyecto hay dos vertientes definidas a continuación. La primera de ellas hace referencia y afecta al medio físico donde se propagan las ondas de estas redes: el exterior. El Ministerio de Industria, Turismo y Comercio, junto con la secretaria de estado de telecomunicaciones y para la sociedad de la información aplican al espectro radioeléctrico el CNAF, es decir, Cuadro Nacional de Atribución de Frecuencias recogido en la ley general de telecomunicaciones 32/2003 articulo 43, donde asignan a unas aplicaciones concretas un número de frecuencias determinado para así garantizar su buen uso sin interferencias ni obstáculos, haciendo previsión de nuevos usos futuros.

Este cuadro no es exclusivo del gobierno de España sino que en los demás países también existe siguiendo las directrices de los organismos con competencias en esta materia tales como la Unión Internacional de Telecomunicaciones (UIT), la Conferencia Europea de Administraciones Postales y de Telecomunicación (CEPT), la Unión Europea (UE) y el Instituto Europeo de Normas de Telecomunicación (ETSI).

En la Orden ITC/3391/2007, del 15 de noviembre de 2007 se aprueba el Cuadro Nacional de Atribución de Frecuencias que está actualmente en vigencia en el estado español. Éste recoge en la UN-85 la banda de frecuencias 2400-2483,5 MHz y en la UN-128 esta otra que va de 5150-5350 MHz y de 5470-5725 MHz designadas para aplicaciones de tipo redes inalámbricas donde será el rango que se utilizara para la emisión de nuestras ondas sin necesidad de licencias, ya que son de uso común.

2 http://www.landashop.com/

La segunda vertiente sobre la legalidad de este proyecto nos apunta otra vez a la ley general de telecomunicaciones 32/2003 articulo 23 sobre la prestación del servicio universal. Este artículo se refiere a la cobertura total del territorio nacional se asignara a operadores que serán los encargados de llevar a cabo. Dado que éstos operadores se rigen, como toda gran empresa, por beneficios, la cobertura y calidad de éstos a sus redes de telecomunicaciones es muy diferente en áreas urbanas que en áreas rurales. Debido a esto, este proyecto intenta ayudar a la sociedad que no vive en grandes urbes a llegar con calidad a redes de telecomunicaciones donde las infraestructuras son nulas o muy deficientes.

Finalmente, cabe destacar que la comunidad *guifi.net* se rige por un principio básico de auto-prestación de servicios, es decir, dar una serie de servicios a cambio de recibir otros. El concepto no es nuevo sino que se remonta a mucho tiempo en nuestra historia y en esencia es este factor clave para la expansión de las redes abiertas como *guifi.net*.

2.2.3.11.**Alternativas**

Crear una red de telecomunicaciones implica muchos recursos, tanto financieros como humanos, costes que solo lo pueden asumir grandes empresas del sector y ellas son las que tienen los derechos de explotación de éstas, pero ¿qué pasa cuando estas empresas no llegan donde un usuario vive, o el nivel de calidad/precio del enlace es ínfimo o simplemente quiere comunicarse con otra persona que vive cerca suyo? Esto nos elimina la alternativa de utilizar este tipo de redes, por lo tanto la única solución es crear una red propia.

Para crear la red necesitaremos infraestructura, pero como ciudadanos no podemos construir infraestructura en lugares públicos, por ejemplo pasar un cable de fibra óptica del domicilio de una persona a otro domicilio que está en la otra manzana de edificios. Por lo tanto no queda otro remedio que utilizar lo que legalmente puede utilizar un ciudadano, las frecuencias de radio citadas en el apartado anterior.

Finalmente, para la conexión a la propia red, podríamos utilizar *routers* con software propietario pero esto nos recortaría prestaciones, estaríamos atados a las especificaciones del fabricante y también tendría un coste más elevado ya que se tendría que pagar el software al contrario con lo que pasa con *OpenWRT*, que es de libre distribución y su personalización de requisitos es extremadamente alta. Por esta razón utilizaremos en nuestros *routers* el *firmware* que se creará en este proyecto.

2.2.4.**Planificación**

2.2.4.1.**Definición de las etapas**

Este proyecto se desarrolla con una simbiosis entre los modelos de desarrollo por prototipos y diseño evolutivo, por esta razón las etapas del proyecto serán largas en su desarrollo, en cada prototipo de *firmware* que se creará se aplicarán una serie de test y análisis para comprobar tanto el correcto funcionamiento del aparato como la

conectividad de éste hacia otros dispositivos semejantes y, si finalmente se ha conseguido un progreso, este se incluirá en el prototipo final.

Las etapas más importantes que en un principio se seguirán en el proyecto serán estas:

Nº	**Actividad**	**Duración (Horas)**
1	Estudio y documentación sobre redes inalámbricas	45
2	Reuniones con comunidades similares	20
3	Estudio de Viabilidad	10
4	Análisis de requisitos	10
5	Creación del *firmware*	40
6	Comprobación de errores	10
7	Creación de una red *mesh*	10
8	Conexión a la red *mesh*	10
9	Comprobación de conectividad	25
10	Documentación sobre el *firmware*	40
11	Implantación de interfaz web	10
12	Comprobación de errores en la interfaz	10
13	Implantación del sistema final a la red guifi.net	10
14	Memoria proyecto final de carrera	30
	Duración Total	**280**

*Precio por hora del proyectista	15 €
Precio Total (Presupuesto + Horas proyectista)	**4498,69 €**

* El precio de las horas de duración del proyecto las asume el proyectista por lo tanto el precio es nulo.

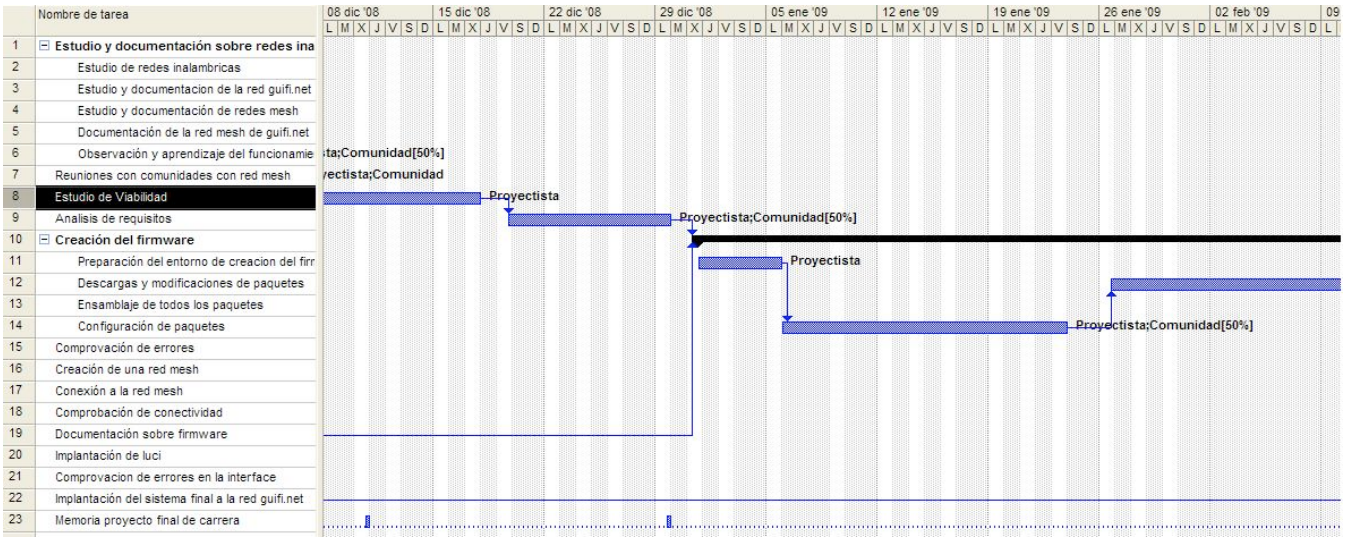
2.2.4.2.**Utilización de los recursos**

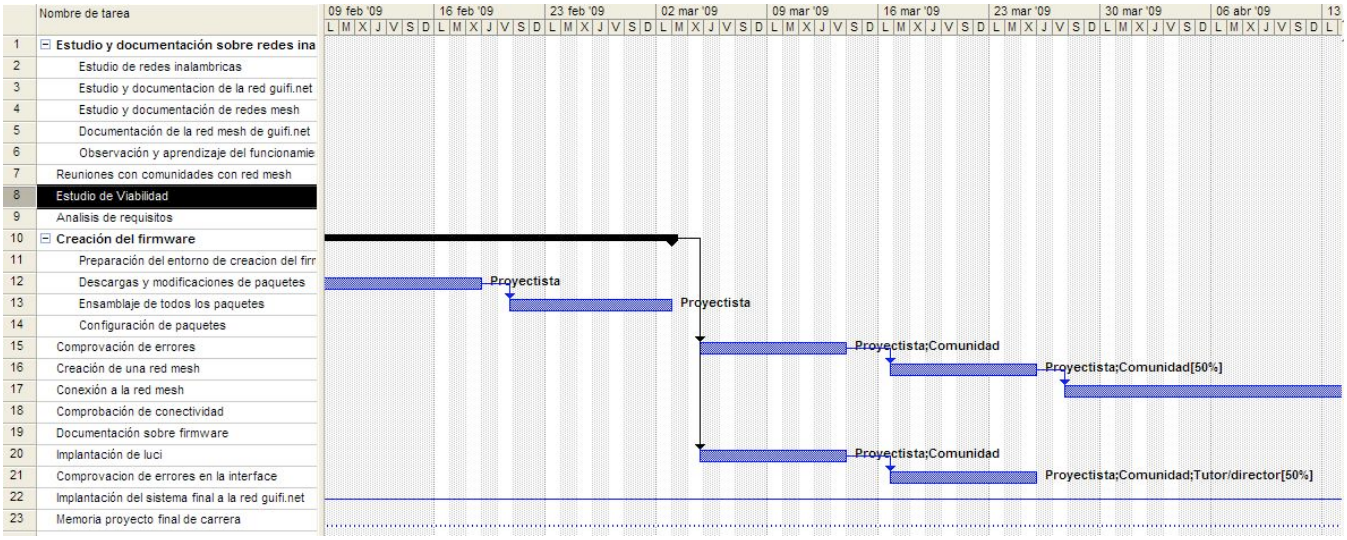
La disponibilidad de los recursos asignados al proyecto es como principal el propio proyectista junto con el tutor del proyecto que serán los que en definitiva finalizarán el proyecto y éste estará asignado a la totalidad de las tareas. Otro recurso

también disponible será el hardware que será asignado a la gran mayoría de las tareas de creación del *firmware* y análisis de éste que se realizarán intentando derivarlo al recurso software con virtualización de imágenes para que el impacto del nuevo *firmware* sea lo menor posible.

Finalmente tenemos un recurso indispensable que es el humano con las reuniones periódicas con las comunidades como *guifi.net,* donde a pesar del apoyo para la absorción de nuevos conocimientos, como la ayuda en las pruebas que se realizarán son vitales para la finalización con éxito del proyecto.

2.2.4.3.**Planificación con diagrama de Gantt**





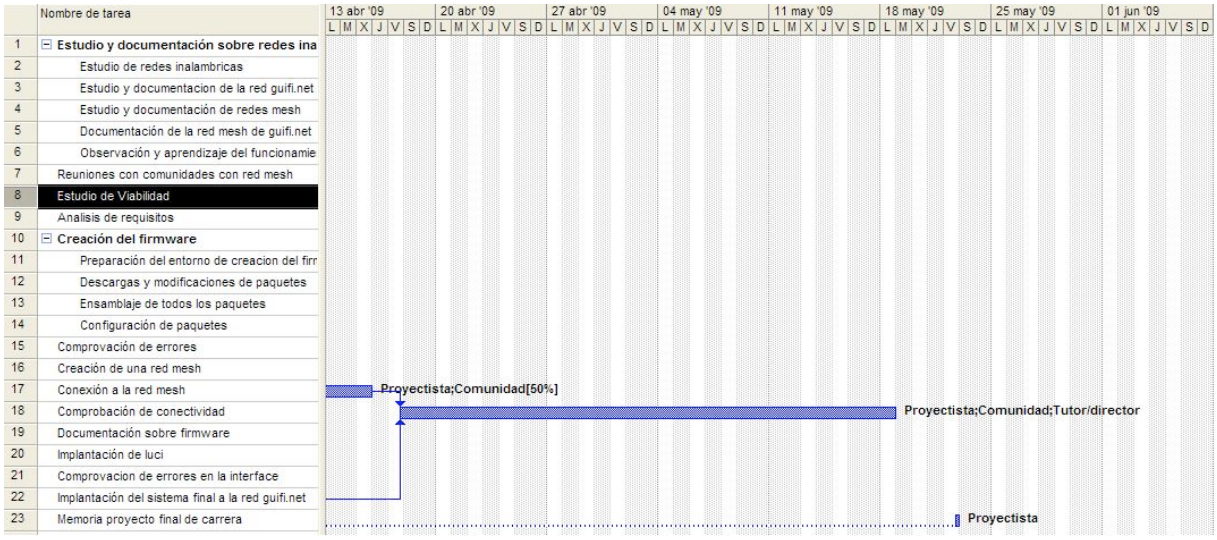


Figura 2.5 - Diagramas de Gantt

2.2.5.**Conclusión**

2.2.5.1.**Síntesis y evaluación**

Las redes *mesh* son una apuesta de futuro para garantizar la conexión en cualquier lugar. Esto asegurará un éxito cuando este tipo de redes estén implantadas establemente, por este motivo la realización de este proyecto puede ser muy ventajosa en un futuro con el valor añadido de un entorno inalámbrico nuevo para ser base sobre otros futuros proyectos sobre redes y telecomunicaciones. Así mismo, y una vez realizado el estudio correctamente, se puede afirmar que el proyecto será viable.

2.2.5.2.**Dedicación**

La dedicación que se ha de dar al proyecto ha de ser de interés continuo por las novedades y progresos que se consiguen en otras comunidades, así como su experiencia. Esto conllevara a reunirse en los lugares de la infraestructura de estas redes para estudiar y evaluar su funcionamiento. No sólo bastará con esto, sino que se tendrá que crear una red de pruebas con los dispositivos facilitados por la universidad para generar valoraciones y análisis que se llevarán a la realidad el día de la presentación del proyecto.

Finalmente cabe destacar que el trabajo en la universidad como en el lugar de trabajo será esencial para la creación del *firmware* aunque después éste mismo se pruebe en un entorno simulado o en la propia máquina.

2.2.5.3.**Contenido**

Este tema es, aparte de ser el preferido por el proyectista, de rabiosa actualidad en la sociedad de la información en que vivimos, donde la necesidad de conexión con otros colectivos o grupos de personas se ha vuelto casi imprescindible. Por este motivo, implementar nuevos tipos de red se ha vuelto casi una necesidad para el progreso de la sociedad, y con este proyecto se pretende abarcar un mayor abanico de personas dispuesta a expandir este tipo de redes.

2.2.5.4.**Adecuación a la carrera**

Por la naturaleza del tema proyectado, la gran mayoría de los conceptos estudiados en la ingeniería técnica informática de sistemas se quedan cortos debido a la alta especialización de éstos sobre el tema de redes inalámbricas. Es normal esta situación, la carrera técnica pretende asentar unas bases sólidas sobre la informática de sistemas y por lo tanto dar la posibilidad de especialización. Por este factor la carrera se adecua correctamente al tema proyectado.

3. Diseño del nuevo sistema

3.1. Análisis de requerimientos

En este apartado se tratará y analizará los requerimientos tanto del *firmware* como así del propio entorno que se requiere en un computador para el correcto funcionamiento y compilación de *kamikaze*. Previamente hay que tener en cuenta que dicho entorno es desarrollado y evolucionado con una frecuencia muy alta, por este motivo puede que sea necesario agregar paquetes o librerías adicionales y revisar constantemente los servicios del sistema así como sus *scripts* de inicialización y configuración para detectar posibles anomalías.

3.1.1. **Requerimientos previos**

Comenzaremos por este tipo de requisitos ya que han de solucionarse en primer lugar y han de ser exactamente como se exponen en esta memoria ya que el entorno para crear el *firmware* es complejo, pero sobre este tema ya se describirá en capítulos posteriores.

RP1: Máquina Compiladora

Una máquina con un hardware de características y rendimiento normales pero con un software muy concreto para obtener un entorno de compilación idóneo. Por este motivo se necesitará un sistema operativo *GNU/Linux* con una serie de paquetes y librerías que necesitara *kamikaze*.

Al tener tanta diversidad de distribuciones de este sistema operativo y cada una de ellas con unas características, paquetes y librerías diferentes instalados de serie en el sistema, elegiremos una distribución en concreto para construir el entorno de *kamikaze* y asegurarnos que dicho entorno con sus respectivos paquetes y librerías que enumerare más adelante no creen ningún conflicto o error con el compilador.

La distribución elegida es *Ubuntu 8.10 Intrepid Ibex*[3]. Hay que tener en cuenta que los paquetes que se enumeran podrán ser diferentes o no haga falta instalarlos en otras distribuciones, por este motivo se ha de comenzar por una instalación limpia del sistema y añadir dichos paquetes.

Paquetes:

- Subversion
- build-essential
- ncurses-term

3 http://www.ubuntu.com/

- libncurses5-dev
- zlib1g-dev
- gawk
- bison
- flex
- Autoconf

RP2: Grabador de tarjetas

Será necesario obtener un grabador de tarjetas de memoria *CompactFlash* para la copia de la imagen generada a la tarjeta que seguidamente se conectará a la placa del nodo ya que en esta memoria es donde se almacena el *firmware*.

RP3: Conexión a la red Internet

El computador que compile la imagen deberá tener acceso a Internet en el momento de la creación del *firmware* ya que en el proceso se ha de bajar el código fuente tanto de *kamikaze* como también de los paquetes seleccionados en la configuración del sistema. También será necesaria para posteriores actualizaciones de los códigos fuente o nuevos paquetes.

3.1.2. Requerimientos funcionales

En este tipo de requisitos están destinados hacia el objetivo de conseguir un *firmware* con capacidad de conexión a redes *mesh*.

RF1: Construcción del *firmware*

El nodo que se construirá viene sin ningún *firmware*, por este motivo la puesta en marcha del sistema no se podrá efectuar sin antes no haber generado dicho *firmware* con un mapa de particiones y sistema de archivos muy concreto que se describirá más adelante.

RF2: Elección de paquetes

Dicho requisito se utilizara para generar una imagen lo más limpia posible de paquetes innecesarios y así obtener una imagen lo mas minimalista posible para solo ofrecer los servicios deseados aunque después se quieran hacer ampliaciones.

RF3: Encapsulado en imagen

Este requisito es necesario para volcar el *firmware* al dispositivo, el encapsulado del *kernel* del sistema operativo junto con el sistema de archivos a una imagen que se deberá construir como imagen tipo *GRUB Linux x86*.

RF4: Protocolo de encaminamiento *BMX*

Este será el principal protocolo de encaminamiento del nodo, como ya se explicará correrán al mismo tiempo tres protocolos de los cuales *BMX* será el que se utilizará normalmente ya que la *IP* pública que se anunciará es de la interfaz de dicho protocolo.

RF5: Protocolo de encaminamiento *BATMAN*

Protocolo secundario que se implementará para que el sistema sea tolerante a errores y caídas de los demás protocolos. Otro de los motivos es que al utilizar una topología de red experimental no se tiene claro que protocolo es el mejor, así que lanzaremos varios para poder analizarlos o hacer pruebas de rendimiento.

RF6: Protocolo de encaminamiento *OLSR*

Ultimo de los protocolos que utilizará el nodo para el encaminamiento incluido por las razones citadas anteriormente y que es el protocolo más estándar y extendido de todos.

RF7: *DHCP*

Este servicio será necesario para la conexión a la red *mesh* de nodos itinerantes tales como portátiles, PDAs por este motivo se ha de incluir en la imagen aunque de su configuración ya se describirá más adelante.

RF8: Herramientas de monitorización de la red

Se ofrecerán sistemas para la monitorización, gestión, y visualización de la red y así ayudar a la comprobación de su correcto funcionamiento pero también para poder realizar nuevos análisis o experimentos.

RF9: Interfaz web

Para mejorar la interfaz del usuario se implementará una interfaz web intuitiva para la configuración de diversas variables del sistema.

3.1.3.**Requerimientos no funcionales**

Estos requisitos hacen referencia a las limitaciones del diseño, por esto, habrá que tomarlos en cuenta a la hora de construir el sistema para que este se integre perfectamente en el sistema global de la red *mesh*.

3.1.3.1.**Requerimientos de rendimiento**

RNF1: Concurrencia

El nodo que se construirá tendrá cuatro interfaces de red, dos inalámbricas (*ath0*, *ath1*) y otras dos *Ethernet* (*eth0*, *eth1*). Las interfaces *ath0* y *eth0* están destinada a la conexión entre nodos, las *ath1* y *eth1* serán las que se utilizarán para conectar clientes con el nodo en un radio cercano o por cable, por este motivo, dichas interfaces tendrán que aceptar más de un usuario conectado a la vez y garantizar los mismos servicios. Este planteamiento se podrá cambiar a las necesidades que existan.

RNF2: Arquitectura de ejecución *GNU/Linux*

El sistema del nodo se basa como ya se ha explicado en una distribución de *GNU/Linux* para dispositivos empotrados. Los *scripts* y paquetes que se utilizarán serán construidos para funcionar en dicha arquitectura, esto quiere decir que toda aplicación que esté desarrollada para esta arquitectura no tendría problemas para ejecutarse, también existirá en el propio sistema una herramienta informática de gestión de paquetes parecida al *APT* de *Debian* llamada *opkg* para una vez montado el sistema poder hacer ampliaciones.

3.1.3.2.**Requerimientos de diseño**

RNF3: Criterio de asignación de direcciones *IPv4*

La red *guifi.net* sigue un patrón de asignación de direcciones *IPv4* para definir en cualquier nodo de la red sea visible a los demás y no halla conflictos de solapamiento. Por este motivo se contribuirá a seguir dicho criterio para que sea gestionado correctamente aunque diseñado para utilizarlo en modo infraestructura. El sistema asignará a cada una de las cuatro interfaces físicas, tres interfaces virtuales para correr en cada una de ellas, un protocolo diferente de encaminamiento mas otra dedicada al servicio *DHCP*, esto da un total de dieciséis *IPs* mas una última que se anunciará como pública. Dicho criterio se detalla a continuación.

El rango de direcciones asignado a la red *guifi.net* es de la 10.0.0.0 hasta la 10.255.255.255, este rango es público y se le asigna una *IP* a cada nodo con la cual se conocerá este nodo en la red tal y como se describe en el párrafo siguiente. Como hay diversas interfaces de red en el nodo, se asignaran a cada interfaz una *IP* privada del rango 172.16.0.0 - 172.31.255.255 para así no malgastar *IPs* públicas.

El rango escogido para las *IPs* públicas depende de la ubicación de la red (nube) y es escogido por *guifi.net*, así entonces se ha escogido el rango 10.139.68.0/24 para albergar esta nube.

Así podríamos tener hasta 2^4 = 16 rangos (con los bits N), y multiplicando los 2^8 = 256 de la máscara (bits Z) da un total de 4096 nodos posibles.

El último octeto lo asigna *guifi.net* que hace de mediador para registrar, controlar y publicar la *IP* del nodo. Al comienzo este número lo calcula el propio sistema con el programa *meshipcalc* para dar conectividad desde el principio, pero este número es provisional ya que se debe registrar el nodo para evitar, por ejemplo, solapamientos.

Para las *IPs* privadas asignadas a cada una de las interfaces virtuales con el rango anteriormente descrito se sigue este criterio:

- Bits fijos del rango
- Protocolo: 00 OLSR / 01 BMX / 10 BATMAN
- Interfaz: 000 eth0 / 010 eth1 / 100 ath0 / 110 ath1
- Bits reservados sin utilización
- Número incremental de la web de *guifi.net*
- Nodo: Último octeto de la *IP* pública del nodo

Para el Protocolo *DHCP* se modifica ligeramente este criterio de la siguiente forma:

- Protocolo: 11 DHCP
- Interfaz: 00 eth0 / 01 ath1
- Clientes

Ejemplo de Direccionamiento

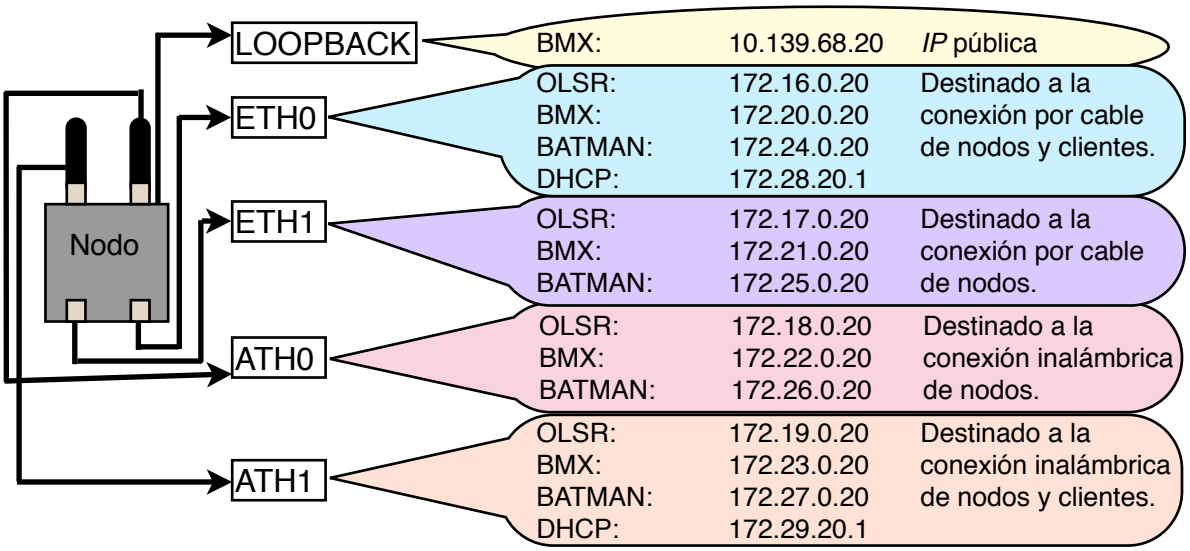


Figura 3.1: Ejemplo de direccionamiento

RNF4: Componentes físicos

El soporte físico disponible es una limitación del potencial del nodo, en el momento de la realización se podrá construir el nodo con otro soporte físico aumentando o disminuyendo el sistema, en el que se detalla en la memoria es suficiente para un nodo local pero si lo que se requiere es un supernodo troncal, se tendría que elegir otro tipo de soporte físico con mas interfaces o mas rendimiento.

RNF5: Situación geográfica

Este punto es vital para la conexión del dispositivo a la red, ya que el medio a tratar es inestable, la situación geográfica es un punto a estudiar antes de montar el nodo, no solamente por la topología de red sino por puntos como el tipo de antenas a utilizar, la visión libre de otros nodos o el algoritmo de encaminamiento que funcionará idóneamente.

RNF6: Normas de conexión

Hay que tener en cuenta que este tipo de redes se han diseñado para ser de libre acceso y compartir los recursos que ofrece cada usuario, por este motivo y también para que el espacio radioeléctrico no se desaproveche inútilmente, la red *guifi.net* sigue un código de buenas prácticas que ha de seguir un usuario que se conecte para poder consumir los recursos disponibles como también ofrecerlos.

3.1.3.3. Requerimientos sobre las interfaces externas

RNF7: Interfaz RS232

Esta interfaz de conexión al nodo mediante el puerto serie se utilizará para comprobaciones de funcionamiento o cuando el nodo este fallando y sea imposible la conexión con las interfaces.

RNF8: Interfaz SSH

Una vez conectados al nodo, el mejor modo de operar con el dispositivo es con la conexión *SSH*. Se utilizará para hacer operaciones con el nodo como saltos entre nodos, comprobaciones de conectividad, ampliaciones o mantenimiento, dicho este factor, en el nodo se tendrá que implementar un servicio de *SSH* para realizar estas funciones.

RNF9: Salto de nodos

Este requisito hay que tenerlo en cuenta cuando esté operativa la red y se de uso de ella, por su topología, es posible que en algún momento no sea posible la visibilidad de un nodo a otro por motivos diversos y sea necesario saltar a otro nodo disponible que sí que lo vea, esto en teoría no tendría que ser un problema ya que el algoritmo de encaminamiento se encargaría, pero como se está en un medio inestable, un entorno abierto, y el software no estable al 100%, se tendrá en cuenta.

3.1.3.4. Requerimientos de objetivos de diseño

RNF10: Mantenimiento del sistema

El sistema una vez construido y montado, no necesitará casi mantenimiento, es decir, la alimentación que necesita para funcionar se la suministraremos por el mismo cable de red a través del dispositivo *PoE (Power on Ethernet)* el cual le conectaremos la fuente de alimentación, después el cable de red que irá al nodo para alimentarlo y finalmente el cable de red que se conectará el usuario, con este método nos ahorramos una instalación más complicada y con un solo cable. Una vez hecha la instalación, y comprobación del funcionamiento no habrá que mantener el sistema con regularidad ya que él se encargará de hacer las gestiones correspondientes.

RNF11: Escalabilidad

El sistema descrito es muy ampliable tanto el soporte físico con la adquisición de nuevas placas, radios o antenas así como el software que se podrá implementarse añadiendo o eliminando funcionalidades respecto a los objetivos que se requieran.

RNF12: Definición de funciones de usuario

En este sistema habrá una serie de competencias que brevemente se definirán a continuación y se ampliarán en el capítulo de diseño y así asignar a cada usuario su función en el sistema. Estas funciones en ocasiones se podrán cruzar, ampliar o reducir dependiendo del grado de implicación del usuario.

Actores del sistema:

- Usuario Final:

Este actor es el destinatario final de toda la red, el sólo se tendrá que preocupar de conectar el cable de red que le llega a su ordenador y dispondrá de los recursos que se ofrecen en esta red así como también podrá ofrecer el sus propios recursos a compartir.

- Propietario del nodo:

El propietario del nodo es el encargado legal del nodo en donde este se encuentre, también se encargara de su instalación y mantenimiento del nodo, este actor la mayoría de veces ira unido con el actor usuario final o administrador de red.

- Administrador de red:

Este actor será el encargado de la gestión de la red, comprobando que los nodos estén correctamente configurados y haciendo pruebas de rendimiento para controlar la red, tendrá conocimientos necesarios para acceder a un nodo remotamente y configurar una serie de parámetros del sistema.

- Desarrollador de *firmware*:

Tendrá que ser experto en las áreas anteriores y con posibilidad de actuar con los demás perfiles, será el encargado de implementar el *firmware* de los nodos actualizando su sistema operativo para solucionar posibles *bugs* o mejoras de rendimiento, así como, añadir o eliminar servicios en función de las demandas de los usuarios.

3.2. Adecuación de los componentes

Debido a la naturaleza del proyecto, la conexión a otros nodos o a la red *guifi.net* no solo dependerá de la generación correcta del software, sino también del equipo electrónico. Por este motivo en este apartado se detallará las peculiaridades a tener en cuenta para el montaje del nodo correctamente.

Antes de empezar con la instalación un factor clave es la ubicación del nodo. Por lo tanto como ya se ha descrito anteriormente se hará un estudio de este factor.

Al necesitar una visión directa para la conexión entre los nodos se ha de buscar un lugar idóneo donde se visualicen los demás nodos que previamente, el propietario del nodo habrá de ser notificado para facilitar esta conexión, por ejemplo, orientando la antena hacia el lugar deseado. Por norma general, las ubicaciones idóneas se encontrarán en los tejados, terrados o lugares más altos de la vivienda.
Una vez hallado el lugar, hay que ponerse al corriente del propietario de esa ubicación, ya que como se ha dicho antes la mayoría son en terrados, terrazas o tejados que pertenecen a la comunidad de vecinos y por lo tanto se ha de pedir permiso para su instalación.

Dicho esto, se ha de instalar un mástil que sujete firmemente el nodo o si existen otros ya instalados aprovecharlos. También seguidamente se instalará un solo cable, de par trenzado *UTP* o *STP* de categoría 5 o superior según la especificación *568A Commercial Building Wiring Standard* de la *EIA/TIA,* que comenzará en el mástil y acabará en un *PoE* (*Power over Ethernet*) que alimentará el sistema para seguidamente conectar con la vivienda del usuario final.

Una vez realizadas todas las tareas anteriores, se procederá al montaje de los componentes electrónicos donde se seguirán una serie de pasos que se describen a continuación.

El primer paso consistirá en preparar la caja estanca que se ha adquirido, para ello antes se tendrá en cuenta que la caja deberá tener un índice de protección de como mínimo IP65 según el estándar *CEI 60529*.

Este estándar se utiliza en equipamiento eléctrico y/o electrónico para clasificar los diferentes grados de protección que resguardan al sistema contra la entrada de materiales extraños. Mediante la asignación de diferentes códigos numéricos, el grado de protección del equipamiento puede ser identificado de manera rápida y con facilidad como se visualiza en la figura 3.2.[10]

Figura 3.2: Protección IP

En las tablas siguientes se muestran los grados de protección:[10]

Primer dígito (IP X)

Nivel	Tamaño del Objeto	Resultado
0	-	Sin protección.
1	>50 mm	El elemento (esfera de 50 mm de diámetro) no debe llegar a entrar por completo.
2	>12.5 mm	El elemento (esfera de 12,5 mm de diámetro) no debe llegar a entrar por completo.
3	>2.5 mm	El elemento (esfera de 2,5 mm de diámetro) no debe llegar a entrar por completo.
4	>1 mm	El elemento (esfera de 1 mm de diámetro) no debe llegar a entrar por completo.
5	Protección contra polvo	La entrada de polvo no puede evitarse, pero el mismo no debe entrar en una cantidad tal que interfiera con el correcto funcionamiento del equipamiento.
6	Protección fuerte contra polvo	El polvo no debe entrar bajo ninguna circunstancia.

Segundo dígito (IPX)

Nivel	Protección	Resultado
0	Sin protección	El agua entrará en el equipamiento.
1	Goteo de agua	No debe entrar el agua cuando se la deja caer, desde 200 mm de altura respecto del equipo, durante 10 minutos.
2	Goteo de agua	No debe entrar el agua cuando se la deja caer, durante 10 minutos (a razón de 3-5 mm 3 por minuto). Dicha prueba se realizará cuatro veces a razón de una por cada giro de 15º tanto en sentido vertical como horizontal, partiendo cada vez de la posición normal de trabajo.
3	Agua nebulizada	No debe entrar el agua nebulizada en un ángulo de hasta 60º a derecha e izquierda de la vertical a un promedio de 10 litros por minuto y a una presión de 80-100kN/m2 durante un tiempo que no sea menor a 5 minutos.
4	Chorros de agua	No debe entrar el agua arrojada desde cualquier ángulo a un promedio de 10 litros por minuto y a una presión de 80-100kN/m2 durante un tiempo que no sea menor a 5 minutos.
5	Chorros de agua	No debe entrar el agua arrojada a chorro (desde cualquier ángulo) por medio de una boquilla de 6,3 mm de diámetro, a un promedio de 12,5 litros por minuto y a una presión de 30kN/m2 durante un tiempo que no sea menor a 3 minutos y a una distancia no menor de 3 metros.

Nivel	Protección	Resultado
6	Chorros muy potentes de agua	No debe entrar el agua arrojada a chorros (desde cualquier ángulo) por medio de una boquilla de 12,5 mm de diámetro, a un promedio de 100 litros por minuto y a una presión de 100kN/m2 durante no menos de 3 minutos y a una distancia que no sea menor de 3 metros.
7	Inmersión completa en agua	No debe entrar agua al objeto que debe soportar (sin filtración alguna) la inmersión completa a 1 metro durante 30 minutos.
8	Inmersión completa y continua en agua	No debe entrar agua al equipamiento eléctrico / electrónico que debe soportar (sin filtración alguna) la inmersión completa y continua a la profundidad y durante el tiempo que especifique el fabricante del producto con el acuerdo del cliente, pero siempre que resulten condiciones más severas que las especificadas para el valor 7.

Seguidamente y comprobado que es compatible con el mástil instalado anteriormente se procederá a hacer los agujeros que se necesiten para las conexiones con las antenas o el cable de red, esto varía dependiendo de la caja utilizada, las antenas a conectar, pero en nuestro caso se facilitaran tres agujeros, dos para las conexiones con las antenas y otro para la alimentación.

Seguidamente se instalará la *routerboard* a la caja como corresponda dependiendo del modelo de esta y a continuación se procederá a insertar las radios que irán unidas con un *pigtail* o latiguillo que se encarga de conectar la radio a la antena.

Por último se enroscarán las antenas, se introducirá la *CompactFlash* previamente escrita con el *firmware* generado y finalmente se conectará el cable de red que alimentará el sistema. Un ejemplo de instalación correcta es la figura 3.3.

Figura 3.3: Montaje de un nodo

Sobre las antenas a utilizar hay que remarcar que dependerá del caso habrá que instalar un tipo de antenas determinado, omnidireccionales, sectoriales o direccionales con sus características especificas y más o menos potentes según la conexión que desee hacer, para el caso que describimos al utilizar la interfaz *ath1* para emitir en 2,4GHz se utilizara una antena omnidireccional bipolar para la conexión con los usuarios y la otra interfaz *ath0* al emitir en 5GHz se utilizara para la conexión entre nodos con una antena direccional de polaridad vertical. No solo se puede utilizar esta configuración, dependiendo de las expectativas del nodo y al tener la posibilidad de varias interfaces se pueden instalar diferentes antenas para así construirlo como un supernodo.

Por último, una vez acoplado al mástil y comprobado el correcto funcionamiento de este como se explicará en una sección posterior, no se requerirá mucho mantenimiento más que comprobar aleatoriamente o cuando hay situaciones climáticas adversas el estado en que se encuentra.

3.3. Diseño del sistema

El *firmware* que se desea construir se basa en *OpenWRT* que como ya se ha explicado es una distribución de *GNU/Linux* para dispositivos empotrados. En esta distribución una de las primeras acciones que hay que hacer es configurar el sistema mediante un archivo de configuración que determinará las características del sistema.

Esta sección se dedicará a moldear el sistema a la medida para que la imagen generada sea "*plug and play*" es decir tenga todo lo necesario para funcionar correctamente al inicio. Estas son las características que el sistema cumplirá.

Se utilizará la versión más estable de *OpenWRT* llamada *kamikaze* en su última versión 8.09.

Como ya se ha dicho, el sistema se basará en *GNU/Linux x86* con *kernel* 2.6 ya que es el más nuevo.

La arquitectura que se deberá construir el sistema será para *PC Engines Alix*.

Al utilizar un dispositivo de almacenamiento con memoria flash en una tarjeta, esto hace que con el tiempo se degrade haciendo inservible el dispositivo debido a la continua lectura y escritura de datos. Por este motivo, el sistema de archivos que utilizaremos tendrá unas peculiaridades específicas para disminuir el número de lecturas o escrituras y así aumentar la vida útil de la tarjeta.

El mapa de particiones quedará de la siguiente forma, una primera partición con 12 MB en *ext2* para el *kernel*, utilizando este sistema de archivos por motivos de arquitectura y estabilidad. La otra partición de 48 MB se utilizará para la estructura de archivos donde se volverá a dividir en dos memorias diferentes, una para utilizarla como memoria de solo lectura (*ROM*) y la otra para lectura/escritura (*RAM*). Por este motivo, la memoria *ROM* estará en *squashFS*, sistema de archivos comprimido de solo

lectura para *GNU/Linux* especialmente diseñados para sistemas empotrados y la otra parte estará en *jffs2*, sistema de archivos con soporte para transacciones especializado en memoria flash.

Con el mapa de particiones anteriormente descrito se conseguirá un sistema cíclico reduciendo las lecturas/escrituras de archivos utilizando como base la partición en *squashFS* y con modificaciones posteriores de los archivos en la partición en *jffs2*. Esto repercutirá también en posibles copias de seguridad automáticas ya que siempre se podrá volver a un estado anterior del archivo almacenado en *ROM*.

Como se observa en la figura 3.4 el mapa de particiones quedará de la siguiente forma.

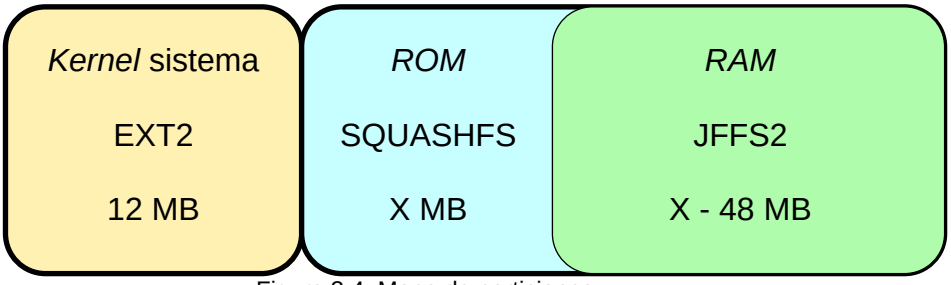


Figura 3.4: Mapa de particiones

Al utilizar el sistema una interfaz RS232 determinaremos la velocidad del puerto a 38400 bps.

El máximo número de *i-nodos* del sistema será de 6000 al no contener una estructura de archivos muy compleja, en el caso que se necesite una estructura más compleja o almacenaje de archivos como por ejemplo para dar servicios de *FTP, HTTP, BitTorrent*… se aumentará este número.

A partir de ahora describiremos los paquetes que incluirá la imagen clasificados por categorías:

- **Paquetes base del sistema**

En este apartado se encuentran los pilares del sistema operativo, se han elegido los necesarios para la arquitectura escogida y otros necesarios para ejecutar los servicios deseados.

✓ *base-files* (Este paquete contiene la base del sistema de archivos y *scripts* de *OpenWRT*)

✓ *busybox* (Paquete que combina versiones reducidas de muchas utilidades comunes de *UNIX* diseñado para sistemas con recursos reducidos)

✓ *dnsmasq* (Paquete diseñado para proporcionar un servicio de *DNS* y *DHCP* en redes de poca extensión)

✓ *dropbear* (Pequeño servidor/cliente de *SSH* diseñado para consumir pocos recursos, es particularmente útil para ser integrado en sistemas de tipo *UNIX*, tales como *routers* inalámbricos)

✓ *firewall* (Cortafuegos de *OpenWRT* basado en *UCI*)

✓ *hotplug2* (Sistema de administración de dispositivos dinámicos. Combina la flexibilidad de *udev* con el diseño de distribución de eventos *hotplug* en un entorno reducido)

✓ *iptables* (Herramienta de administración del cortafuegos *Netfilter* para *IPv4*)

✓ *ldconfig* (Biblioteca compartida de configuración de rutas)

✓ *ldd* (Utilidad para enlace de librerías)

✓ *libgcc* (Librería de soporte *GCC*)

✓ *mtd* (Utilidad para actualizar el *firmware* desde otros *firmware* o viejas versiones de *OpenWRT*)

✓ *uci* (Interfaz de Configuración Unificada)

✓ *uclibc* (Librería de C para sistemas empotrados)

✓ *udevtrigger* (Pequeña utilidad para gestionar eventos de dispositivos con conexión “en caliente”)

‣ Módulos del *kernel para el sistema de archivos*

Este apartado se especifican los sistemas de archivos que soportará el sistema, en este caso se añadirán los básicos.

✓ *kmod-fs-cifs* (Módulo del *kernel* para soportar el sistema *CIFS-SAMBA*)

✓ *kmod-fs-ext2* (Módulo del *kernel* para soportar el sistema de archivos *EXT2*)

✓ *kmod-fs-ext3* (Módulo del *kernel* para soportar el sistema de archivos *EXT3*)

✓ *kmod-fs-vfat* (Módulo del *kernel* para soportar el sistema de archivos *FAT*)

‣ Módulos del *kernel para Netfilter*

Se añade una extensión del cortafuegos para gestionar las conexiones.

✓ *kmod-ipt-nathelper* (Extensión de *Netfilter* para manipulación y seguimiento de conexiones así como también *NAT*. Los servicios incluidos con estos soportes son *FTP, TFTP, IRC*)

- **Módulos del *kernel para dispositivos de red***

Estos módulos son necesarios como controladores de los dispositivos de red integrados en el sistema.

✓ *kmod-natsemi* (Extensión del *kernel* para soportar la serie de semiconductores *DP8381x* destinados a interfaz *ethernet*)

✓ *kmod-ne2k-pci* (Extensión del *kernel* para el adaptador *ethernet NE2000*)

✓ *kmod-via-rhine* (Extensión del *kernel* para *chipsets ethernet VIA Rhine*)

- **Módulos del *kernel para soporte de red***

El paquete añadido es necesario para la correcta gestión de los protocolos de encaminamiento.

✓ *kmod-ipip* (Extensión del *kernel* para encapsulación *IP* a *IP*)

- **Otros módulos del *kernel***

La extensión elegida permitirá el funcionamiento de los tres *LEDs* de la placa.

✓ *kmod-leds-alix* (Extensión del *kernel* para soportar los *LED*s de la placa)

- **Módulos del *kernel para soporte USB***

En este apartado se agregan las funcionalidades que tendrán las dos conexiones USB, en este caso se eligen las exclusivas para soportar dispositivos de almacenamiento extraíble, pero cabe la posibilidad de agregar soporte a dispositivos tales como módems 3g, conversores USB a Serie, impresoras, y muchos otros periféricos con conexión USB.

✓ *kmod-usb-core* (Extensión del *kernel* para soportar *USB*)

✓ *kmod-usb-ohci* (Extensión del *kernel* con controladores *USB OHCI*)

✓ *kmod-usb-storage* (Extensión del *kernel* para soportar dispositivos de almacenamiento extraíble)

✓ *kmod-usb-uhci* (Extensión del *kernel* con controladores *USB UHCI*)

- **Módulos del *kernel con controladores de dispositivos inalámbricos***

El paquete elegido es necesario para el funcionamiento de las interfaces inalámbricas *aht0* y *ath1.*

✓ *kmod-madwifi* (Extensión del *kernel* con controladores *Atheros 802.11a/b/g*)

- **Paquetes de Administración**

En este apartado no se ha escogido ningún paquete al existir una interfaz propia integrada y personalizada al sistema, pero es posible instalar otras interfaces web como *XWRT* o *LUCI* (*Lua Unified Control Inteface*)

- **Paquetes relacionados con redes**

Este apartado es muy importante, se añaden muchos paquetes para la administración, gestión y monitorización de la red así como también los protocolos de encaminamiento.

✓ *batmand* (Demonio de *BATMAN* para encaminamiento de nivel 3)

✓ *fping* (Herramienta para gestionar varias peticiones de tipo *ping* en diferentes *hosts* en paralelo)

✓ *horst* (Herramienta para escanear y analizar redes inalámbricas de tipo 802.11 especialmente diseñado para modo *ad-hoc* y topología *mesh*)

✓ *ip* (Utilidad de control de encaminamiento)

✓ *iputils-ping* (Programa para enviar *ping* a través de redes *IPv4*)

✓ *iputils-tracepath* (Programa para descubrir la ruta de los paquetes en una red)

✓ *mtr* (Herramienta para el diagnostico de redes)

✓ *olsrd* (Demonio de *OLSR* para encaminamiento de nivel 3)

✓ *olsrd-luci* (Paquete para configuración de *OLSR* mediante *LUCI*)

✓ *olsrd-mod-arprefresh* (Extensión de *OLSR* para refresco de la cache *ARP*)

✓ *olsrd-mod-bmf* (Extensión de *OLSR* para envío básico de *multicast*)

✓ *olsrd-mod-dot-draw* (Extensión de *OLSR* para información de topología DOT)

✓ *olsrd-mod-dyn-gw* (Extensión de *OLSR* para puerta de enlace dinámica hacia INTERNET)

✓ *olsrd-mod-dyn-gw-plain* (Extensión de *OLSR* para puerta de enlace simple hacia INTERNET)

✓ *olsrd-mod-httpinfo* (Extensión de *OLSR* con pequeño servidor web de información)

✓ *olsrd-mod-nameservice* (Extensión de *OLSR* para resolver nombres de *hosts*)

✓ *olsrd-mod-secure* (Extensión de *OLSR* con firma digital para garantizar la seguridad del dominio)

✓ *olsrd-mod-txtinfo* (Extensión de *OLSR* con pequeño servidor web de información)

✓ *tc* (Utilidad de control del tráfico de la red)

✓ *tcdump* (Utilidad de red para el monitoreo y adquisición de datos)

✓ *wget* (Programa para la descarga de datos con soporte SSH)

- **Paquetes de *GSF-Guifi.net***

Los siguientes paquetes han sido generados por *graciasensefils*[8] y *guifi.net*, necesarios para los objetivos del proyecto.

✓ *meshipcalc* (Paquete para calcular la *IP* de un nodo en la red *mesh*)

✓ *bmx* (Última versión del demonio de *BATMAN* experimental para encaminamiento de nivel 3 con *plugins* para *mesh*)

- **Paquetes de Utilidades**

En el último apartado se añaden utilidades varias, para ampliar los programas del sistema.

✓ joe (Editor de textos)

Una vez añadidos todos los paquetes descritos, habrá que aplicar un parche a la compilación con los archivos y *scripts* necesarios para la configuración correcta de todas las variables del sistema. En este parche se incluirán todos los ficheros que se quieran agregar a la imagen. A continuación se explicarán los necesarios para la correcta inicialización de todos los servicios y demonios del sistema.

El parche sigue la misma estructura de archivos del sistema por lo tanto se deberán crear los directorios en donde se quiera insertar archivos en la imagen.

El esquema quedará así:

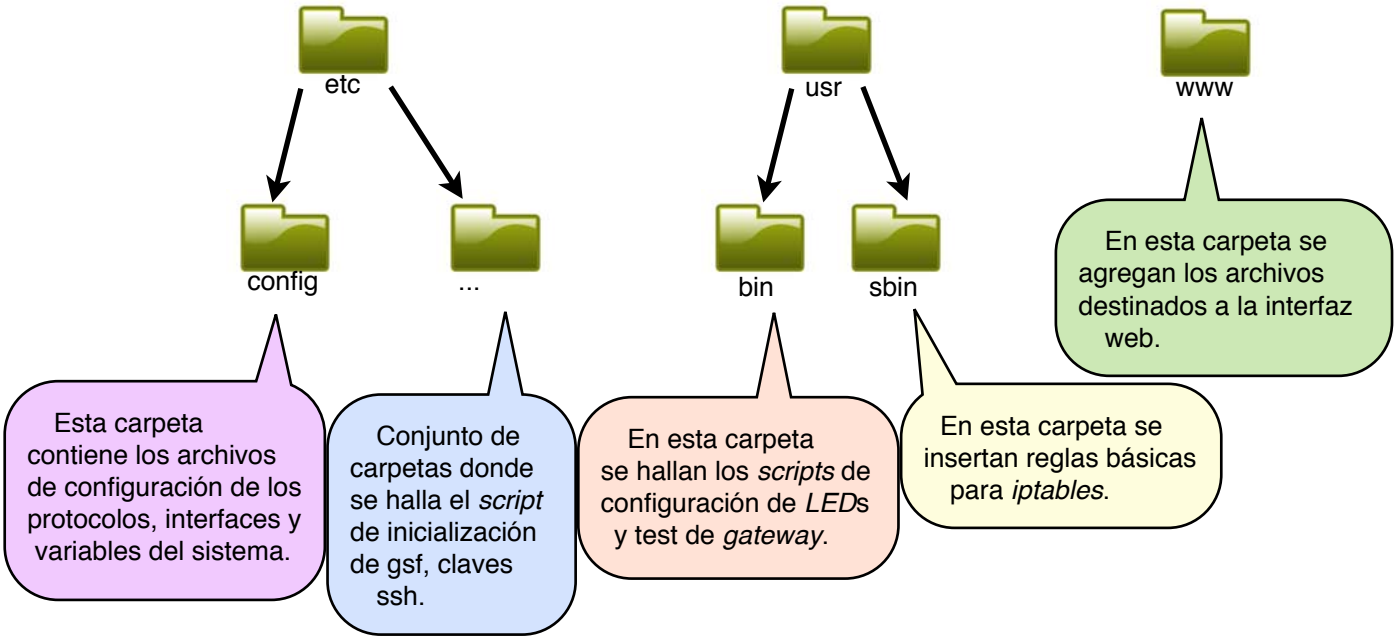


Figura 3.5: Esquema parche

Llegado a este punto el sistema podrá funcionar correctamente y solo quedará compilar el sistema junto con todos los paquetes añadidos, seguidamente aplicar el parche de archivos y finalmente encapsularlo todo en una imagen para su posterior escritura en dispositivo de almacenamiento extraíble de la *routerboard*.

De todo esto se encargará el propio sistema *kamikaze* y el *script* de generación del *firmware* que se desarrollará para sistematizar el proceso, uno de los objetivos del proyecto.

4. Desarrollo del nuevo sistema

4.1. Implementación

En este apartado se detallan los *scripts* más importantes del sistema, el primero sistematiza el proceso de generación del *firmware* en la máquina compiladora mientras que los siguientes inicializan todos los valores y servicios del sistema cuando se arranca el nodo.

En primer lugar, existe un directorio principal en el software del proyecto llamado '*mesh*' donde se haya todo lo necesario para conseguir con éxito la generación del firmware. En el contenido de este directorio se encuentra:

'*alix2c2*' carpeta donde se haya la configuración del sistema operativo que se ha detallado en anteriores apartados.

'*dl*' carpeta donde se descargarán todos los códigos fuente de los paquetes del sistema operativo, esta carpeta no es necesario crearla ya que el propio *script* se encarga de este cometido y de esta manera puede ser compartida por diversas compilaciones.

'*files*' carpeta donde se ubica el parche de archivos que se quieran incrustar en el *firmware.*

'GsfGuifi.net' carpeta donde se encuentran los paquetes para calcular la *IP* del nodo en la red *mesh* y el demonio del protocolo de encaminamiento *BMX*.

'*kamikaze*' carpeta principal donde se encuentra el código fuente de *OpenWRT*.

'firmware' script de generación guiada del *firmware*. Este se detalla a continuación.

Seguidamente explicamos el proceso de generación del *firmware*. Este proceso consta de una serie de pasos que se distinguen porque en el *script 'firmware'* al comienzo de cada uno se pregunta si se desea ejecutar dicho paso.

En el primer paso se descargan las fuentes de *OpenWRT*, estas fuentes se hallan en el repositorio oficial[6], debido a la diversidad de revisiones existen diferentes ramas en el repositorio, que es actualizado con nuevas revisiones muy frecuentemente y esto sugiere un problema con la compatibilidad del sistema al sufrir cambios que produzcan errores y dejen de funcionar servicios o los propios *scripts*. Para corregir este inconveniente se elige una rama muy estable que asegurará el correcto funcionamiento del sistema. Esta rama es:

```
svn://svn.openwrt.org/openwrt/tags/8.09/
```

Normalmente, se utiliza la carpeta '*kamizaze*' para descargar estas fuentes pero si se desea tener varias versiones o compilaciones es posible crear diferentes carpetas que compartan la carpeta '*dl*'. Con el *script* '*firmware*' esto es posible y se enlaza automáticamente.

El siguiente paso es actualizar e instalar todas las cabeceras de los paquetes de *OpenWRT*, este paso no es necesario pero es recomendado ya que al surgir actualizaciones con bastante frecuencia, se asegurará un sistema actualizado. La acción se ejecuta con estos comandos:

```
./script/feeds update -a
./script/feeds install -a
```

En el tercer paso se copia la configuración del sistema operativo tal y como se describe en los apartados anteriores, esta configuración está destinada a alcanzar los objetivos marcados en el proyecto, no obstante es posible modificarla para adecuarla a las necesidades del usuario. Esta configuración se encuentra en la carpeta '*alix2c2*' en el archivo '*.config*'. Por ello se ejecutará:

```
cp ../alix2c2/.config .config
```

Hay que remarcar que como se ha comentado antes sobre la frecuencia de actualización de las revisiones es posible que este archivo no sea compatible total o parcialmente. Para solucionar el inconveniente es recomendable siempre repasar la configuración para darla por válida y también por si se desea agregar o eliminar funcionalidades

Para visualizar la configuración se utiliza el comando:

```
make menuconfig
```

En el siguiente paso se está preparado para la compilación del sistema. El proceso de compilación es complejo ya que se utiliza la cross-compilación, este método se utiliza al existir multitud de arquitecturas que soportan este sistema operativo. El proceso se demora entre 20 minutos y 2 horas y es posible que surjan errores que se tendrán que analizar para ser reparados. El sistema no solo se encarga de descargar y compilar todo el sistema sino también de aplicar el parche de archivos y encapsular todo en una imagen. Para asegurar que se completa y visualiza todo el proceso se agregará el modificador '*V=99*' y el comando quedará de la siguiente forma:

```
make world V=99
```

Una vez acabado el proceso de compilación la imagen generada se halla en la carpeta '*bin*' que se encuentra en el directorio '*kamizake*' o como el usuario lo haya llamado. La imagen se llama '*openwrt-x86-squashfs.image*' si se cree necesario existe un archivo '*md5*' en la misma ubicación para comprobar la integridad de dicha imagen que será la que copiaremos a la *CompactFlash.* En el *script* '*firmware*' el proceso de *flash* es posible hacerlo automáticamente, ya que detectará la *CompactFlash* si no se

ha insertado conociendo el mapa de dispositivos que utiliza el sistema. Para el proceso de *flash* de la *CompactFlash* se utiliza el comando:

```
sudo dd if=openwrt-x86-squashfs.image of=/dev/sdc
```

Una vez finalizados estos pasos se llega al final del proceso de generación del *firmware*, este *script* que se muestra en anexo B, es totalmente automático, guiado y sencillo.

En este punto el sistema ya puede encenderse y esperar a su conexión automática con la red *mesh*. Este paso no es trivial, para conseguirlo, detrás existen una serie de *scripts* que intervienen al arrancar el sistema y se detallan a continuación.

En la carpeta '*init.d*' se ubican los *scripts* que facilitan el inicio y el cierre de demonios o programas como el *script* '*gsf*' que inicializa todos los parámetros de red desde la asignación de las *IPs* hasta los protocolos de encaminamiento. Este *script* se encarga de modificar parámetros tanto del propio sistema al darle nombre al sistema con la propia *IP* pública, coordenadas para geolocalización, e-mail del propietario, el tipo de hardware que utiliza el nodo. Todos estos parámetros y muchos otros se modifican a través de la interfaz *uci* que permite acceder a cualquier variable del sistema para manipularla. Para facilitar la actualización o modificación de algunos valores se ha creado un archivo de configuración del propio *script* ubicado en la carpeta '*config*' del directorio '*etc*' que a través de *uci* cogerá los valores y los integrará en el sistema.

Del cálculo de las *IPs* se encarga un paquete llamado '*meshipcalc*' que genera la *IP* pública a través de un *hash* de la dirección *MAC* de la interfaz *eth0* junto con otros valores como el rango de la red. Como ya se ha descrito anteriormente la *IP* asignada será temporal y se tendrá que registrar el nodo para conseguir un *IP* pública propia.

Otro *script* integrado es el '*gwtest*' es el encargado de comprobar si el nodo tiene acceso a Internet comprobando la conectividad mediante *pings* a diversas direcciones.

En la carpeta '*config*' se ubican los archivos de configuración del sistema, de las interfaces de red, de las interfaces inalámbricas y de los tres protocolos de encaminamiento. Cada archivo está definido por un estándar que se ha de seguir para el correcto funcionamiento del proceso.

5. Implantación del sistema

5.1. Generación del *firmware*

En este apartado se empieza el proceso de generación del *firmware* que posteriormente se insertará a la *routerboard*.

Previamente al proceso se recomienda tener construido el nodo, en secciones anteriores, se describe la adecuación de los componentes del nodo para el correcto funcionamiento, aunque no hace falta que este instalado en su ubicación prevista, esto es así ya que una vez generado el *firmware* se debería probar antes de ponerlo en funcionamiento en la red *mesh*. En el apartado siguiente se describirá como saber si el nodo funciona correctamente. Como se ha descrito anteriormente este proceso se ha sistematizado con un *script* que hace una serie de preguntas al usuario para guiarle a través del proceso. Este se encuentra dentro de la carpeta principal con el nombre de '*firmware*'.

El proceso guiado es el siguiente, como notación el proceso entero tiene una duración estimada de entre 30 minutos a 2 horas:

En primer lugar abrimos un terminal en la máquina compiladora y nos situamos en la carpeta principal '*mesh*'

```
cd mesh
```

Seguidamente ejecutamos el *script*:

```
./firmware
```

En un primer paso que preguntará es si se quiere descargar el código fuente de *OpenWRT* como se ve en la figura 5.1.

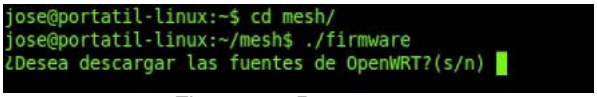


Figura 5.1: Paso 1

En caso afirmativo, se pedirá el nombre de la carpeta donde se hallará el código fuente y seguidamente empezará la descarga.

Figura 5.2: Paso 1

En caso negativo, se pedirá el nombre de la carpeta donde se encuentra el código fuente ya descargado.

Figura 5.3: Paso 1

En el segundo paso se pedirá actualizar e instalar todos los paquetes de *OpenWRT* como se ve en la figura 5.4.

Figura 5.4: Paso 2

En caso afirmativo, las cabeceras de los paquetes se actualizarán e instalarán.

En caso negativo, no se hará ninguna acción continuando con el siguiente paso

La siguiente pregunta pedirá si se desea cargar la configuración básica del sistema operativo, esta configuración está diseñada para el correcto funcionamiento de la red *mesh* con una serie de parámetros, paquetes y variables del sistema establecidos.

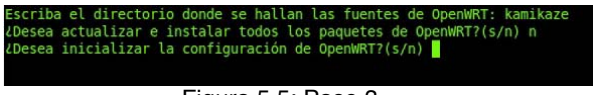


Figura 5.5: Paso 3

En caso que se quiera dejar la configuración anterior, modificada o no por el usuario no se deberá cargar esta configuración.

Continuando, nos preguntará si deseamos validar la configuración del sistema, esto es debido a que en posibles nuevas versiones el archivo puede variar haciendo que no funcione correctamente o simplemente se quiere modificar algún parámetro o paquete.

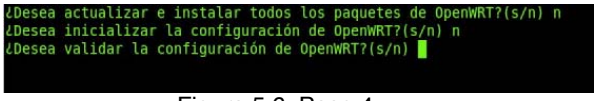


Figura 5.6: Paso 4

Una vez completados los pasos anteriores se está preparado para compilar el sistema, este proceso como se ha descrito en anteriores apartados es complejo al utilizar técnicas de cross-compilación y el tiempo de duración de la compilación completa es elevado. Se preguntará si se desea compilar, en caso afirmativo se notifica el comienzo y el final del proceso de compilación.

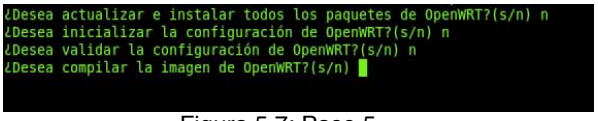


Figura 5.7: Paso 5

El último paso consiste en copiar la imagen a la tarjeta *CompactFlash* para posteriormente insertarla en la *routerboard*.

Este paso se puede hacer manualmente con las notaciones del apartado anterior de implementación o automáticamente, el proceso automático hay que asegurarse que no haya ningún dispositivo más que el lector de tarjetas insertado en el sistema para

evitar problemas. Una vez realizado dicha comprobación el sistema se esperará a que se inserte la *CompactFlash* como se ve en la figura 5.8

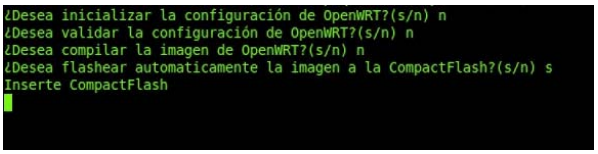


Figura 5.8: Paso 6

Una vez insertada, será detectada y empezará la copia de la imagen, en este proceso no se deberá sacar la tarjeta ni manipularla para evitar posibles fallos en la copia o desperfectos en la tarjeta.

Realizados todos los pasos correctamente se llegará al final de la generación del *firmware* con el mensaje de proceso finalizado y se estará en condiciones de insertar la tarjeta al nodo y encenderlo para comprobar su funcionamiento y integración en la red *mesh*.

5.2. Funcionamiento

El sistema está pensado para un funcionamiento muy simple, los propios enlaces a otros nodos de la red se hacen automáticamente con solo detectarse, y mediante la interfaz web se puede visualizar estos enlaces así como el estado de los protocolos de encaminamiento. Para comprobar que la imagen se ha creado y funciona correctamente se utilizará una interfaz inalámbrica para conectarse al nodo como cliente. Esta interfaz opera en modo *ad-hoc* a 2,4 GHz y junto al sistema *DHCP* nos dará una *IP*. Una vez conectados hay varias formas de acceder al nodo, la primera es abriendo un terminal a través de un túnel *ssh* en la que se identificará como '*root*' con contraseña '*guifi*'. Seguidamente se realizan una serie de comprobaciones escribiendo:

```
ip a
```

Este comando nos debería mostrar una salida como la de la figura 5.9

```
root@10_139_68_232:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet 10.139.68.232/32 brd 10.139.68.232 scope global lo:bmx
2: tunl0: <NOARP> mtu 1480 qdisc noop state DOWN
    link/ipip 0.0.0.0 brd 0.0.0.0
3: wifi0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 195
    link/ieee802.11 00:80:48:52:ff:9b brd ff:ff:ff:ff:ff:ff
4: wifi1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 195
    link/ieee802.11 00:80:48:52:ff:93 brd ff:ff:ff:ff:ff:ff
5: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 00:0d:b9:14:ae:ac brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.232/14 brd 172.19.255.255 scope global eth0
    inet 192.168.1.25/24 brd 192.168.1.255 scope global eth0:bck
    inet 172.20.0.232/14 brd 172.23.255.255 scope global eth0:bmx
    inet 172.28.232.1/28 brd 172.28.232.15 scope global eth0:dh
    inet 172.24.0.232/14 brd 172.27.255.255 scope global eth0:bat
6: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 00:0d:b9:14:ae:ad brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.232/14 brd 172.19.255.255 scope global eth1
    inet 172.21.0.232/14 brd 172.23.255.255 scope global eth1:bmx
    inet 172.25.0.232/14 brd 172.27.255.255 scope global eth1:bat
7: ath0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 00:80:48:52:ff:9b brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.232/14 brd 172.19.255.255 scope global ath0
    inet 172.22.0.232/14 brd 172.23.255.255 scope global ath0:bmx
    inet 172.26.0.232/14 brd 172.27.255.255 scope global ath0:bat
8: ath1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 00:80:48:52:ff:93 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.232/14 brd 172.19.255.255 scope global ath1
    inet 172.29.232.1/28 brd 172.29.232.15 scope global ath1:dhc
    inet 172.23.0.232/14 brd 172.23.255.255 scope global ath1:bmx
    inet 172.27.0.232/14 brd 172.27.255.255 scope global ath1:bat
root@10_139_68_232:~#
```

Figura 5.9: Interfaces

Aquí nos muestra todas las interfaces de red físicas y virtuales con sus correspondientes *IPs* que corresponden con el criterio de direccionamiento descrito en apartados anteriores, este es un ejemplo de nodo funcionando correctamente.

Otra verificación posible es ver si los parámetros de las interfaces inalámbricas están correctos, para este propósito se escribe el comando:

```
iwconfig
```

Que nos da una salida como la figura 5.10

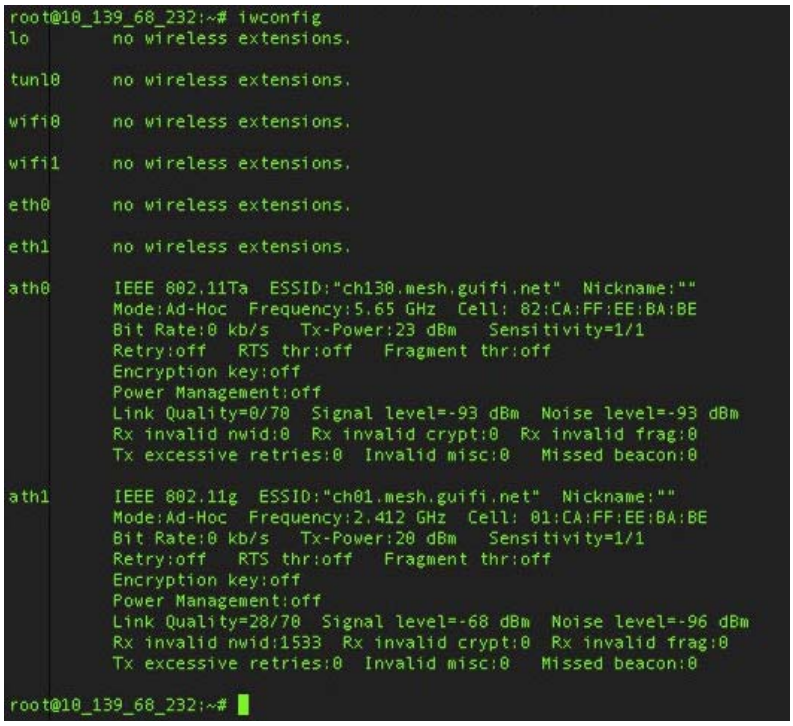


Figura 5.10: Interfaces inalámbricas

Se puede observar que una interfaz emite a 5GHz y la otra a 2,4 GHz así como también una serie de parámetros establecidos que se podrían modificar, para listar estos parámetros utilizaremos el comando:

```
iwlist interfaz parámetro
```

Con estos parámetros a listar:

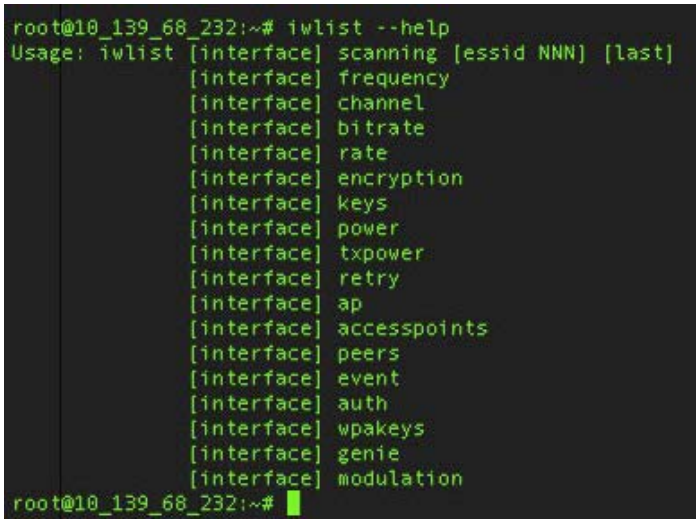


Figura 5.11: Parámetros

Para modificar estos parámetros inalámbricos se utiliza el comando:

```
iwconfig interfaz parámetro valor
```

Como se muestra en la figura 5.12

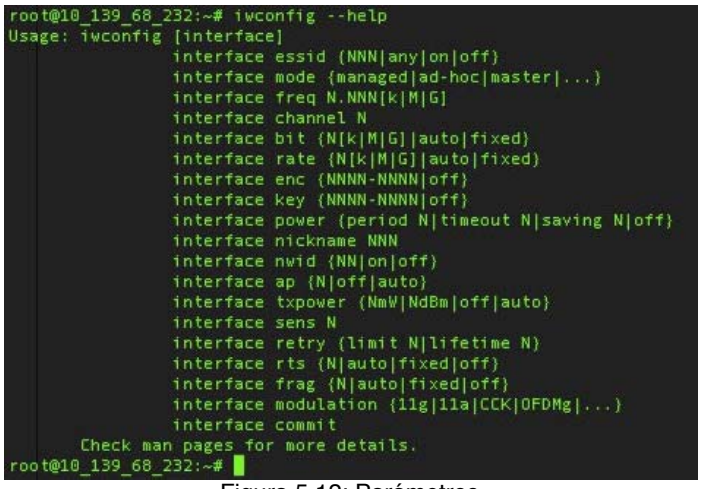


Figura 5.12: Parámetros

Estos valores están predefinidos para que funcione correctamente la red, por esto, no se deberían modificar a no ser que así se requiera.

Otro modo de acceder al nodo es a través de la interfaz web, para ello, se pondrá la *IP* del nodo en un navegador web y directamente accederemos al índice como la figura 5.13 donde se muestra la *IP* del nodo, un apartado de administración y los tres protocolos de encaminamiento donde justo arriba unas marcas que se explican en la figura 5.14.

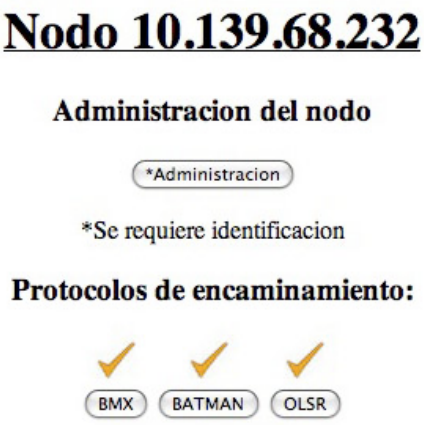


Figura 5.13: Interfaz web

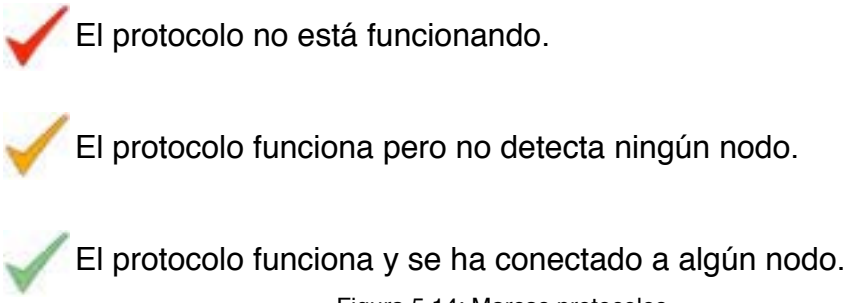


Figura 5.14: Marcas protocolos

La web se refresca automáticamente así que las marcas nos muestran el estado en tiempo real de los protocolos.

A continuación, si se accede a los botones de los protocolos nos mostrará también en tiempo real las tablas de encaminamiento con una serie de parámetros como los vecinos, calidad de los enlaces, topología y demás.

En el apartado de administración en primer lugar se identificará al usuario solamente permitiendo el acceso al 'root'. Una vez identificado accederemos como se muestra en la figura 5.15 a una página donde se administrará el nodo teniendo la opción de reiniciar el nodo, detener o arrancar cada uno de los protocolos de encaminamiento, mostrar los procesos que corren en el sistema o modificar parámetros de la red *mesh*.

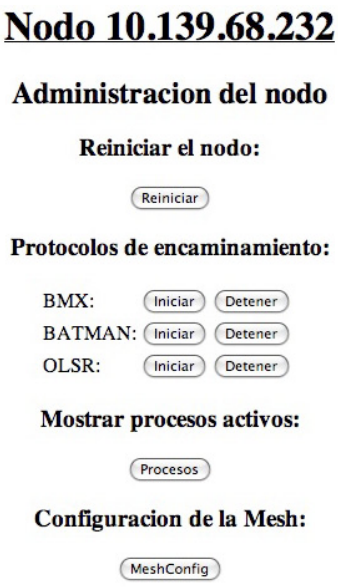


Figura 5.15: Administración

Una vez descrito, la conexión *ssh* y la interfaz web solo queda utilizar la red consumiendo o compartiendo recursos, servicios y conexión a Internet.

5.3. Pruebas y análisis

El escenario escogido para realizar las pruebas de rendimiento sobre los tres protocolos de encaminamiento ha sido el siguiente, un entorno al aire libre con dos nodos y visualización directa separados entre ellos 10, 20 y 30 metros con antenas omnidireccionales emitiendo a 5,650 GHz con polaridad vertical y una potencia de 199mv (23 DBs)

Pruebas de calidad de señal

Escenario 1:

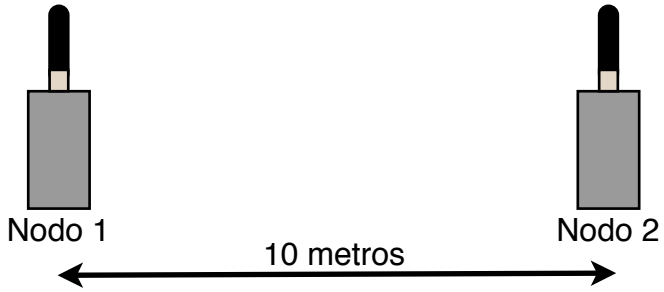


Figura 5.16: Escenario 1

Monitorización de la relación Señal/Ruido (SNR):

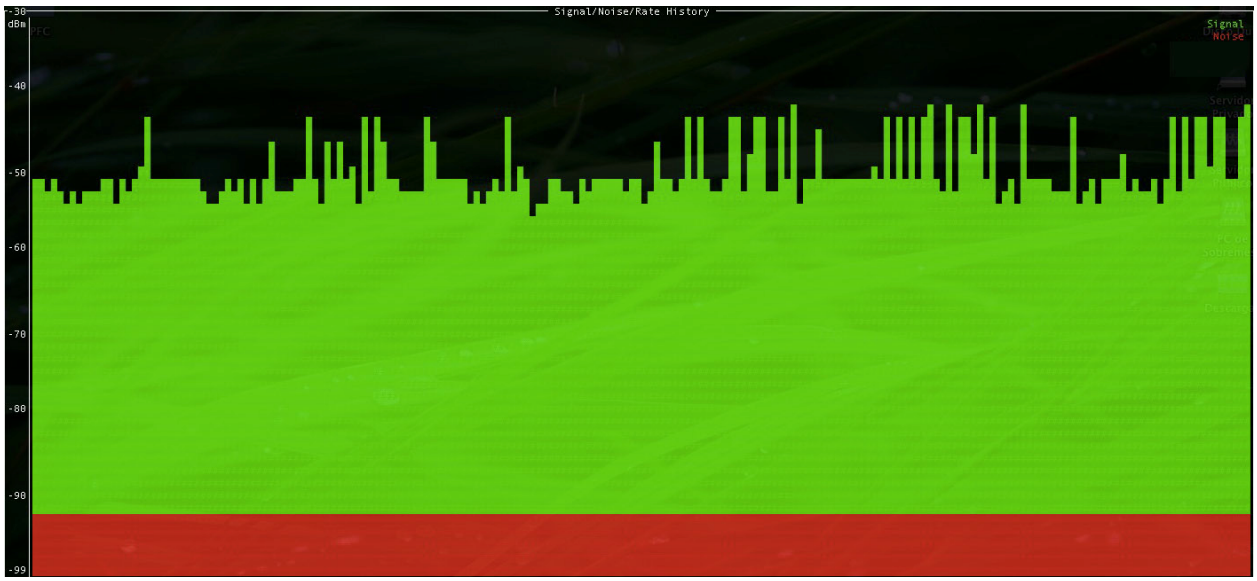


Figura 5.17: SNR 1

Escenario 2:

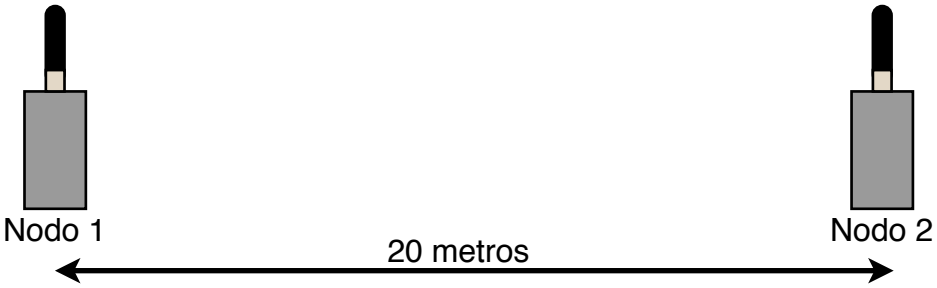


Figura 5.18: Escenario 2

Monitorización de la relación Señal/Ruido (SNR):

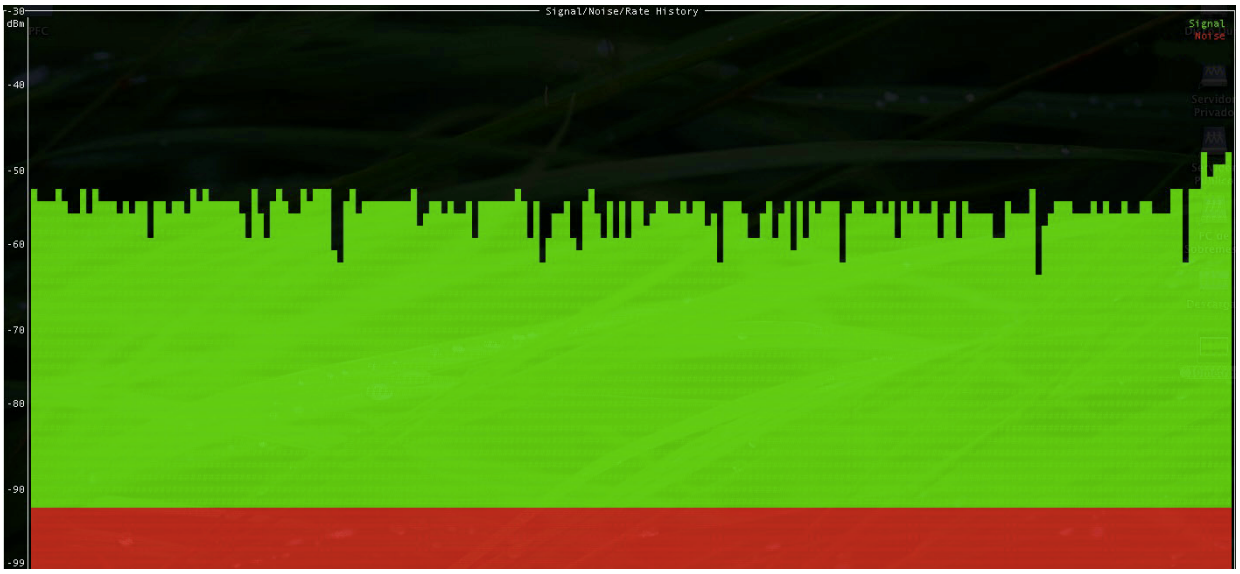


Figura 5.19: SNR 2

Escenario 3:

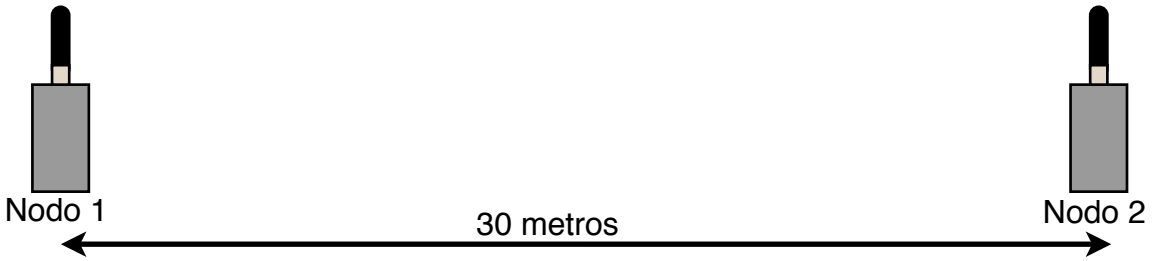


Figura 5.20: Escenario 3

Monitorización de la relación Señal/Ruido (SNR):

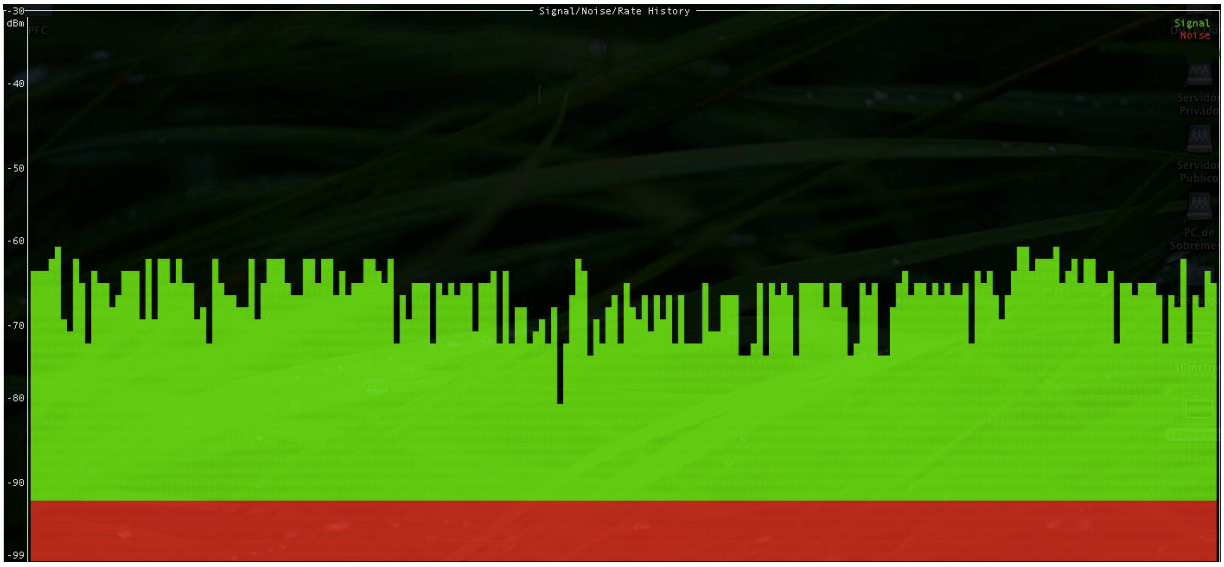


Figura 5.21: SNR 3

Como análisis de esta primera prueba se extrae que la calidad del señal en estos escenarios es muy buena, disminuyendo en unos 4 o 5 dBs por cada 10 metros, hay que comentar que se está en unos escenarios idóneos con visibilidad directa ya que si se encuentran obstáculos entre nodos la señal se va deteriorando al tener que cambiar de medio con otra serie de características físicas, por este motivo siempre se recomienda que haya visibilidad directa entre nodos.

Otro aspecto que afecta a este punto es el tipo de antenas, en este caso se han utilizado omnidireccionales como en la figura 5.22 y para largas distancias este tipo no servirían debido a que al emitir en 360 grados pierden potencia. El tipo elegido para conectar nodos es por lo tanto antenas direccionales que apuntan todo el potencial de la señal en una dirección consiguiendo que sea más efectiva.

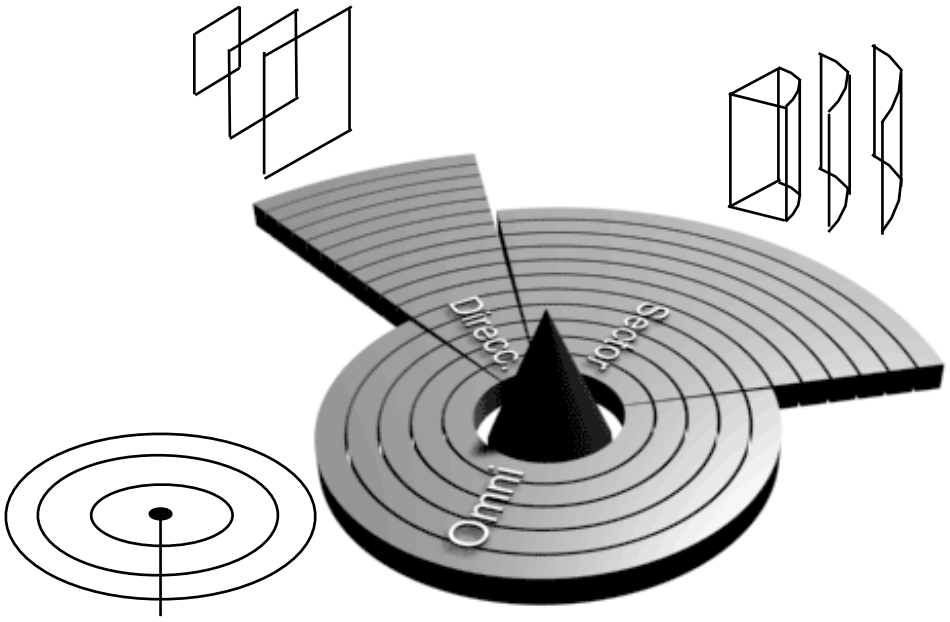


Figura 5.22: Antenas

Pruebas de descubrimiento de nodos

Se escoge el escenario 2 para realizar dichas pruebas. Los resultados se han tomado a partir de la captura en tiempo real de las tablas de encaminamiento de los protocolos:

☑ Hora de referencia 00:00 (minutos:segundos)

☑ Los nodos tardan de apagado a operativo 00:58

Prueba 1:

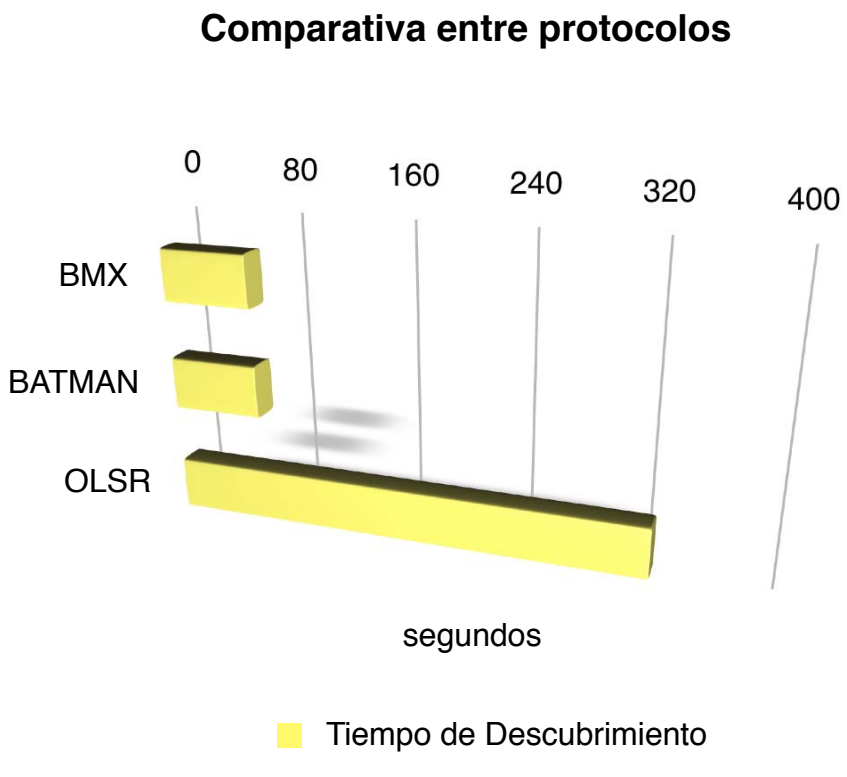
1. Se arranca el primer nodo hasta que este operativo.

2. Los datos son capturados desde el primer nodo.

3. Se arranca el segundo nodo tomando como tiempo de referencia las 00:00

4. Descubrimiento del segundo nodo por parte de los protocolos:

BMX:	01:01
BATMAN:	01:02
OLSR:	05:26



Prueba 2:

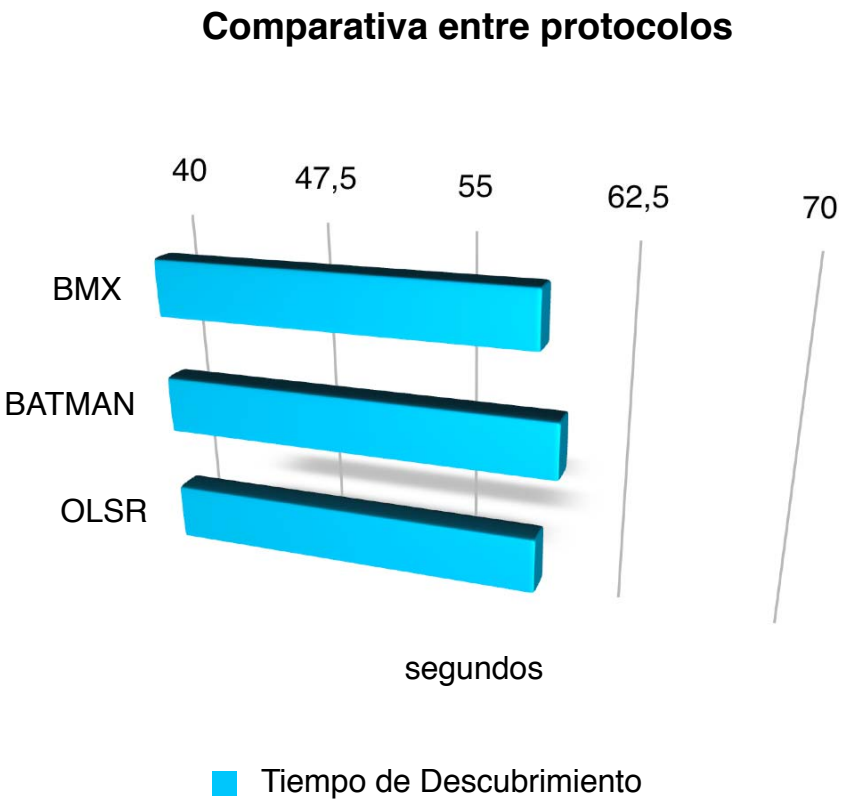
Se arranca el primer nodo hasta que este operativo

Se arranca el segundo nodo a las 00:00

Los datos son capturados desde el segundo nodo

Descubrimiento del primer nodo por parte de los protocolos:

BMX:	00:59
BATMAN:	01:00
OLSR:	00:59

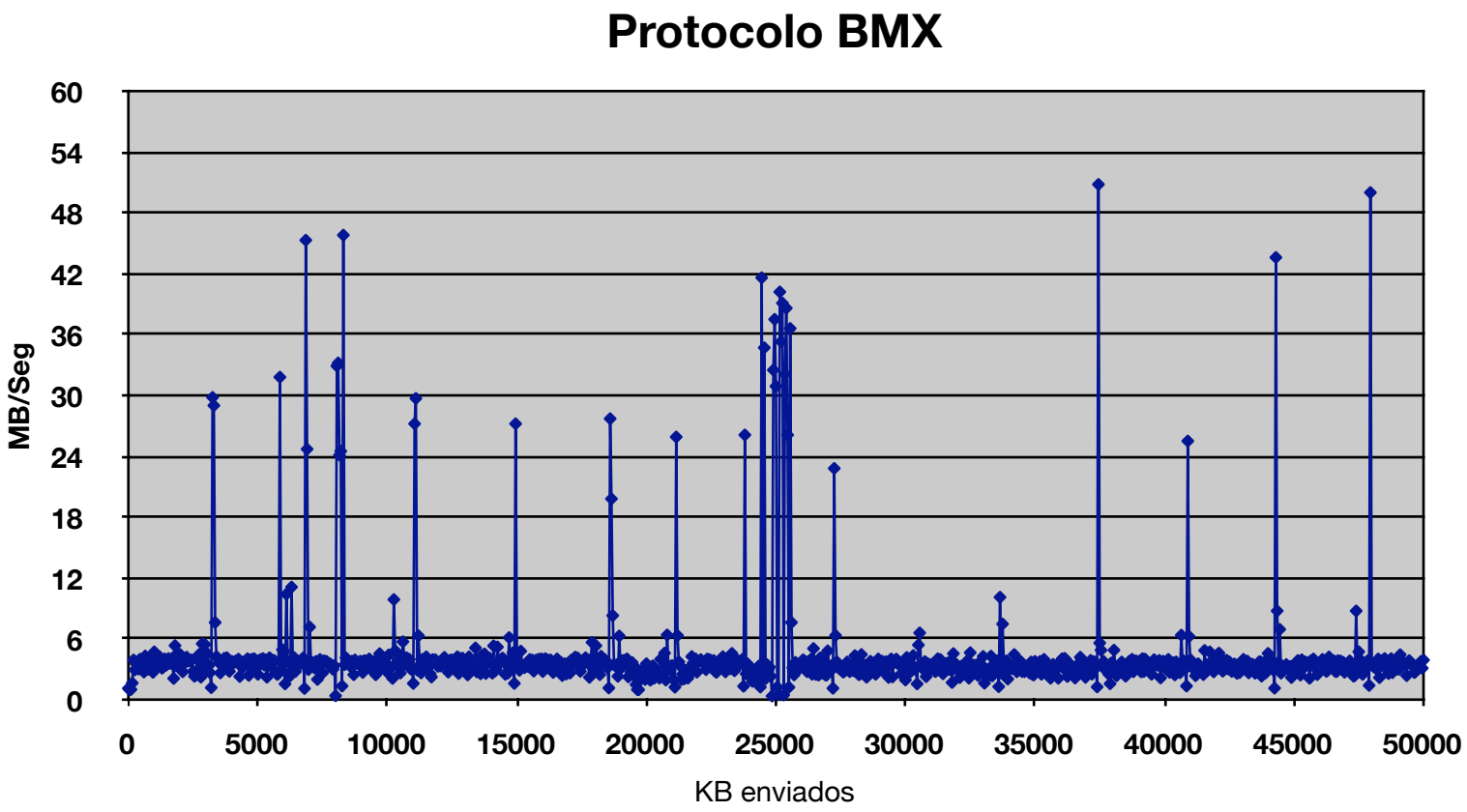


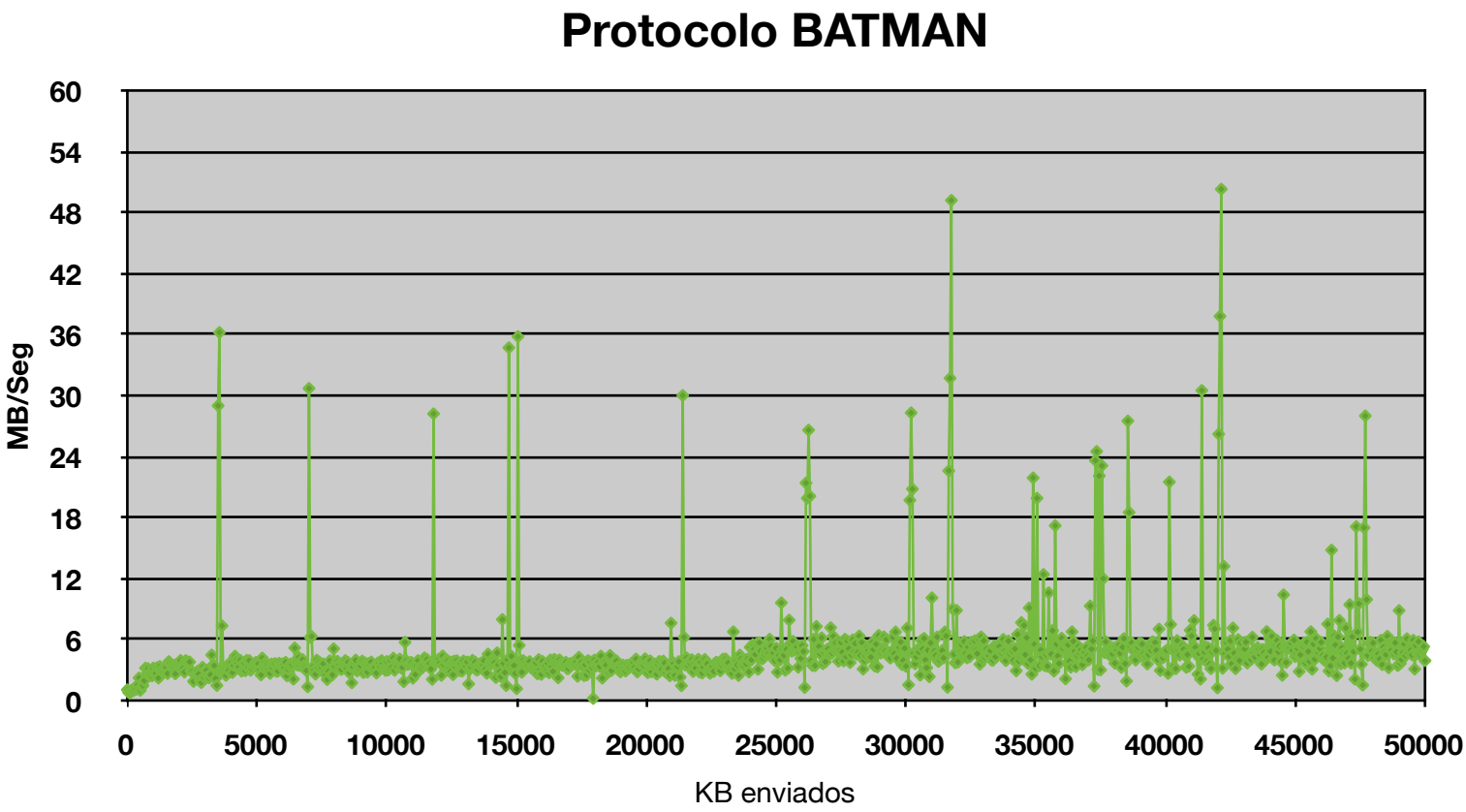
Como análisis de esta prueba se puede observar que de los tres protocolos analizados los mejores siempre son el *BMX* y *BATMAN* ya que *OLSR* es más lento. También hay que tener en cuenta que *OLSR* es un protocolo pro-activo, es decir que guarda toda la topología de los nodos con sus calidad de enlace y demás para trazar la ruta idónea, esto es un inconveniente si es una red con muchos nodos porque las tablas se vuelven intratables y se acaban saturando.

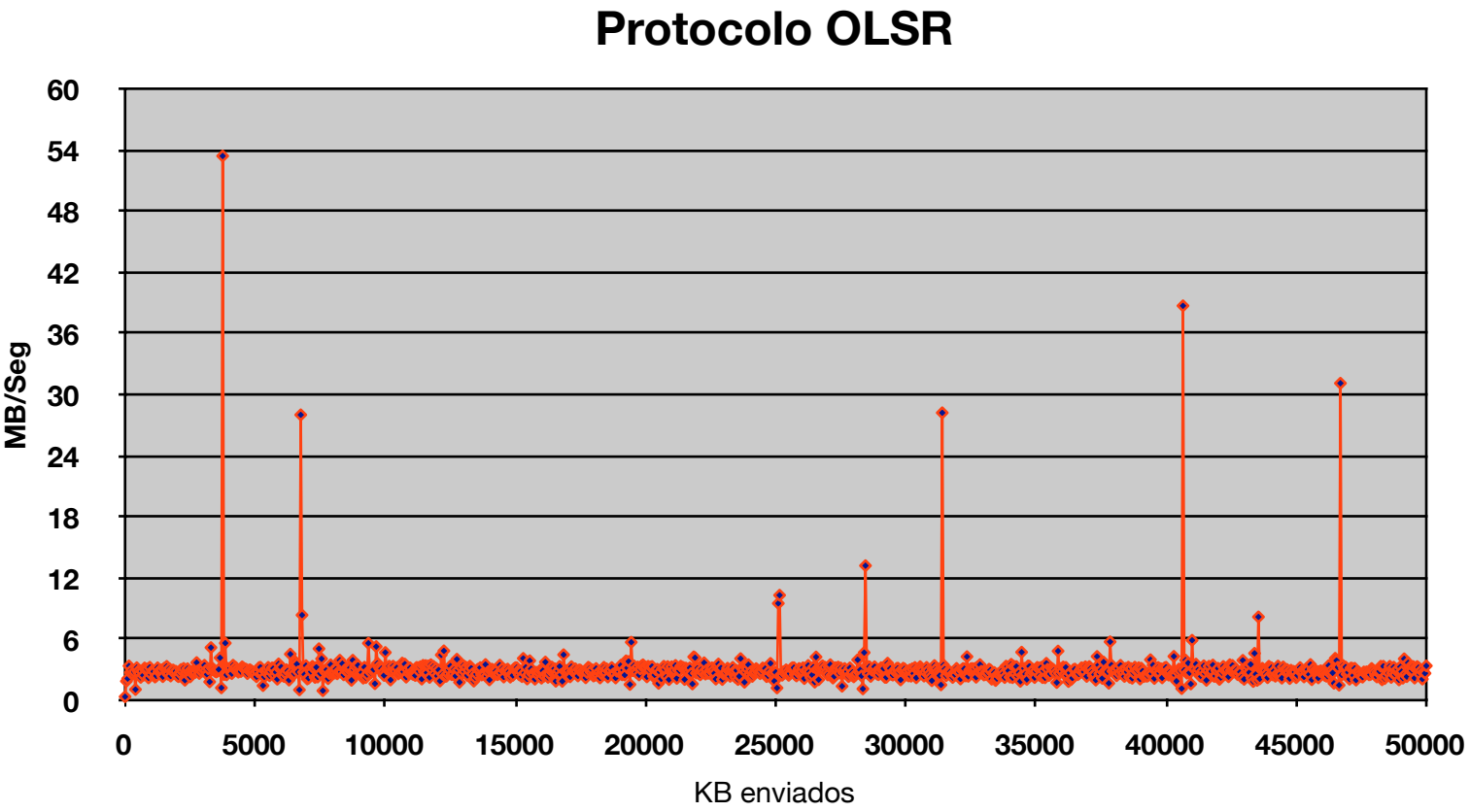
Como clasificación se puede decir que para redes *mesh* no es recomendable utilizar el protocolo *OLSR* siempre y cuando el nivel de nodos sea muy elevado y los protocolos *BMX* y *BATMAN* quedan muy cerca en las estadísticas y se debería de realizar más pruebas de rendimiento para poder afirmar cual de los dos es mejor.

Pruebas de velocidad de transferencia

Se escoge el escenario 2 para realizar dichas pruebas. Los resultados se han tomado a partir de la transferencia del nodo 2 al nodo 1 de un fichero binario de 50 MB mediante la utilidad *wget* capturando cada 50 KB la tasa de transferencia con estos resultados.




Protocolo BATMAN
MB/Seg
60
54
48
42
36
30
24
18
12
6
0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
KB enviados


Protocolo OLSR
MB/Seg
60
54
48
42
36
30
24
18
12
6
0
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
KB enviados

Como resultados de esta prueba se ven una serie de picos en todas las gráficas, esto se cree que es debido a una mala medición de la tasa de velocidad por parte del programa ya que en muchos casos es imposible llegar a esos valores.

En la primera gráfica vemos el protocolo *BMX* con una media de 5 MB/s aunque la línea es bastante gruesa, así que las variaciones de velocidad aunque muy pequeñas son frecuentes.

En la segunda gráfica del protocolo *BATMAN* al comienzo se puede observar como empieza de menor a mayor velocidad hasta conseguir estabilizarse y también se distingue un claro aumento de la velocidad a partir de la mitad del archivo haciendo bastante inestable la tasa.

Por último el protocolo *OLSR* se observa una línea bastante delgada comparando con las anteriores sin tantos picos y estable hasta el final.

Con estos resultados se observa que en primer lugar todos los protocolos tienen una media bastante parecida aunque el protocolo más estable es el *OLSR*.

Para la realización de todas estas pruebas se han creado por una serie de *scripts* que a continuación se explican brevemente y en el anexo C se mostrarán:

- *tabla_bmx*: *Script* que durante un minuto captura la tabla de encaminamiento del protocolo *BMX* cada segundo y lo guarda en un registro para ser posteriormente analizado.

- *tabla_batman*: *Script* que durante un minuto captura la tabla de encaminamiento del protocolo *BATMAN* cada segundo y lo guarda en un registro para ser posteriormente analizado.

- *tabla_olsr*: *Script* que durante un minuto captura la tabla de encaminamiento del protocolo *OLSR* cada segundo y lo guarda en un registro para ser posteriormente analizado.

- *trans_bmx*: *Script* que a través de la aplicación *wget* descarga un archivo binario de 50 MB de otro nodo por la interfaz del protocolo BMX y captura en un registro cada 50 KB la tasa de transferencia para posteriormente ser analizado.

- *trans_batman*: *Script* que a través de la aplicación *wget* descarga un archivo binario de 50 MB de otro nodo por la interfaz del protocolo BATMAN y captura en un registro cada 50 KB la tasa de transferencia para posteriormente ser analizado.

- *trans_olsr*: *Script* que a través de la aplicación *wget* descarga un archivo binario de 50 MB de otro nodo por la interfaz del protocolo OLSR y captura en un registro cada 50 KB la tasa de transferencia para posteriormente ser analizado.

6. Conclusiones

6.1. Conclusiones globales

Una vez finalizado el proyecto, se exponen las conclusiones que por un lado hacen referencia a los objetivos del proyecto y por otro a conclusiones personales del proyectista.

Se puede afirmar que el sistema generado se implanta perfectamente en una red *mesh*, esto implica que los objetivos descritos en los primeros apartados de la memoria se cumplen todos y cada uno de ellos. El sistema final cumple con las expectativas del proyecto y crea un entorno inalámbrico base con una topología experimental y unas características peculiares que hacen de él un sistema idóneo para experimentación y pruebas.

Como desventaja, hay que comentar que al utilizar esta topología con unos protocolos de encaminamiento no estandarizados y en proceso de desarrollo no se puede asegurar la efectividad al 100% de dicha red, aún pareciendo una gran desventaja a favor hay una gran multitud de recursos humanos implicados en estos proyectos para perfeccionar, desarrollar y ampliar estos protocolos y el propio sistema operativo *OpenWRT*. Los usuarios de estas redes en países como Alemania llegan a los 350.000, esto hace que el número de *feedbacks* (notificaciones de errores) sea muy elevado y empuje a una evolución de los protocolos y el sistema operativo enorme. Un hecho que corrobora la afirmación anterior es el elevadísimo número de actualizaciones que sufre el sistema operativo *OpenWRT* que como apunte el mes de junio de 2009 sufrió más de 350 *commits* (actualizaciones).

En verdad no ha habido ningún objetivo sin finalizar, pero en la interfaz web se deseaba implementar en *luci* ya que pretende ser un estándar, pero por motivos de tiempo no ha sido posible y por este motivo se ha implementado en *shell script*. Por último se comentó la idea de también integrar un sistema de *hotspot* pero se desechó al pensar el sistema como base para que después el usuario lo personalice a sus gustos y necesidades.

Las conclusiones del proyectista son muy favorables, le ha permitido adquirir un número muy elevado de conocimientos sobre redes inalámbricas y sistemas operativos *UNIX*, *GNU/Linux*. Así como también realizar un proyecto en informática colaborando con agrupaciones de desarrolladores y usuarios.

En lo que respecta a la planificación temporal, hay desviaciones en el tiempo establecido para cada tarea, las reuniones con las comunidades se han realizado un total de 30 horas, en creación del firmware se ha realizado en 50 horas, en la interfaz web se ha demorado 10 horas más al realizarla en *shell script* con un total de 20 horas y finalmente en la memoria final del proyecto se ha realizado en 40 horas. Esto nos da una desviación de 50 horas más de las preestablecidas con una duración total de 330 horas.

Finalmente, se puede decir que el proyecto finalizado con todos los objetivos es un proyecto innovador y actual con muchos usuarios potenciales y toca muchas de las asignaturas cursadas en la carrera de Ingeniería Técnica Informática de Sistemas como redes y ampliación de redes, sistemas operativos, grafos o microprocesadores, incrementando así el nivel de conocimientos del proyectista.

6.2. Ampliaciones

El entorno inalámbrico Wifi generado por el proyecto se ha diseñado para ser sólo un entorno base. Esto hace que las ampliaciones de este proyecto sean muchas y muy variadas. En lo respectivo al sistema operativo, al ser un sistema basado casi totalmente en *UNIX,* sería factible encarar líneas de proyectos sobre comunicación inalámbrica, sistemas empotrados, *shell scripting*,… En lo que respecta a redes una línea de proyectos muy interesante podría ser la comparación de protocolos de encaminamiento por medio de pruebas de rendimiento, implementación o similares a las efectuadas en el apartado de pruebas y análisis.

Al ser un sistema altamente personalizable se podrían implementar nuevos servicios para dar más valor al sistema o nuevos modelos de desarrollo de este tipo de redes.

Otro aspecto a ampliar seria la interfaz web que al no ser un objetivo primario y estar implementada en *shell script* no es atractiva visualmente y sólo se han integrado las funciones básicas y usuales.

Como se ha visto el abanico de posibles ampliaciones es extenso y con diferentes dificultades, hay líneas de proyectos futuros con menos dificultad como nuevos servicios o mejoras en la interfaz web así como de mayor dificultad como la comparación de protocolos de encaminamiento, ya que al ser un medio de transmisión inestable se requerirían unos análisis exhaustivos o también nuevos modelos de desarrollo de redes *mesh*.

Bibliografía

Publicaciones impresas

[1] BENEDI PALACIOS, JOSE ALBERTO (2004) Unix (guía práctica), Anaya Multimedia.

[2] SOBELL, MARK G. (2008) Manual práctico de Linux: comandos editores y programación shell, Anaya Multimedia.

Recursos electrónicos

[3] Amadeu Albós Raya, Amadeu; Sánchez Jiménez, Óscar David. (2004) Implantació, projectes i empreses de programari lliure. UOC.

[4] Otero García, Alberto. (2007) Proyecto de dirección de sistemas de información. UOC.

[5] OpenWRT - Wireless Freedom. [http://openwrt.org]
Último acceso: 30 de junio de 2009

[6] OpenWRT Sources. [https://svn.openwrt.org/browser]
Último acceso: 30 de junio de 2009

[7] Guifi.net. [http://guifi.net]
Último acceso: 30 de junio de 2009

[8] GraciaSenseFils. [http://graciasensefils.net/doku.php]
Último acceso: 30 de junio de 2009

[9] Com s'ha construit el mesh 2.0 [http://merry.biruji.org/gsf/mesh-gsf]
Último acceso: 30 de junio de 2009

[10] Wikipedia. [http://es.wikipedia.org]
Último acceso: 30 de junio de 2009

[A1] Tema 5: Encaminamiento. [http://www.it.uc3m.es/~prometeo/rsc/apuntes/encamina/encamina.html]
Último acceso: 30 de junio de 2009

[A2] RFC3626: OLSR. [http://tools.ietf.org/html/rfc3626]
Último acceso: 30 de junio de 2009

Las citas que comienzan por A_ pertenecen al anexo A

[A3] Como trabaja el algoritmo de *batman*. [http://www.lugro.org.ar/pipermail/lugro-mesh/2008-September/000572.html]
Último acceso: 30 de junio de 2009

Agradecimientos

En primer lugar el proyectista agradece al director del proyecto, David Megías Jiménez, por la ayuda que le ha dado en todo el proceso del proyecto, destacando las ideas y tiempo que ha dedicado en el proyectista.

A la universidad, por confiar en este proyecto e invertir en él recursos económicos como también al apoyo de los profesores y compañeros.

Otro pilar clave para finalizar el proyecto a sido Roger Baig Viñas de *guifi.net* que sin su ayuda de conocimientos sobre este tema y guía de la documentación hubiese sido imposible alcanzar los objetivos. Agustí Moll de graciasensefils también tiene parte de esta culpa al ofrecer sus conocimientos y tiempo al proyectista que tanto le han ayudado.

Seguidamente, a las personas de la fundación *guifi.net* junto con *graciasensefils* por dejar asistir al proyectista a reuniones y charlas que han ayudado a conocer a más personas interesada en el tema y comentar inquietudes sobre diversos modelos de desarrollo de este tipo de redes.

Finalmente, a mi familia y amigos por aguantar, ayudar, apoyar y motivar al proyectista aun cuando hubo visiones pesimistas o cuando el estrés se hacía más fuerte.

A todos y a todas, mil gracias.

José Carlos Mut Rojas,

Sabadell, junio de 2009.

Anexo A: Protocolos de encaminamiento

A continuación se clasifican los tres protocolos de encaminamiento utilizados en el proyecto. Esta clasificación se puede realizar en función de donde se decide encaminar o de la adaptabilidad del protocolo.

En función de donde se decide encaminar, existen:

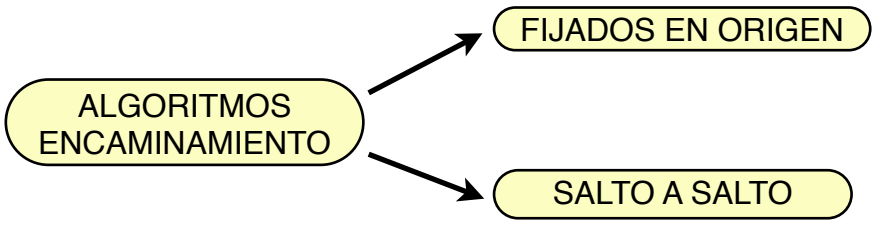


Figura A.1: Primera clasificación

➡ **Fijados en el origen:** [A1]Se calcula la ruta completa que seguirá para llegar al destino. Para ello, cada paquete lleva un campo que especifica su ruta (campo RI: *Routing Information*), y los nodos intermedios sólo se dedican a reenviar los paquetes por esas rutas ya especificadas. Son los nodos finales los que tienen las tablas de encaminamiento y no se hace necesaria la consulta o existencia de tablas de encaminamiento en los nodos intermedios.

➡ **Salto a salto:** [A1] No se calcula la ruta completa que seguirá, sino sabiendo donde está el destino, conocen sólo el siguiente salto a realizar.

En esta clasificación los protocolos de encaminamiento *BMX* y *BATMAN* están en el tipo salto a salto. El *OLSR* se podría clasificar como salto a salto pero con diferencias ya que en cada nodo existen tablas de encaminamiento de toda la red y en función de esas tablas calcula una ruta parcial hasta otro nodo que volverá a calcular la mejor ruta hacia el destino.

En función de la adaptabilidad, existen:

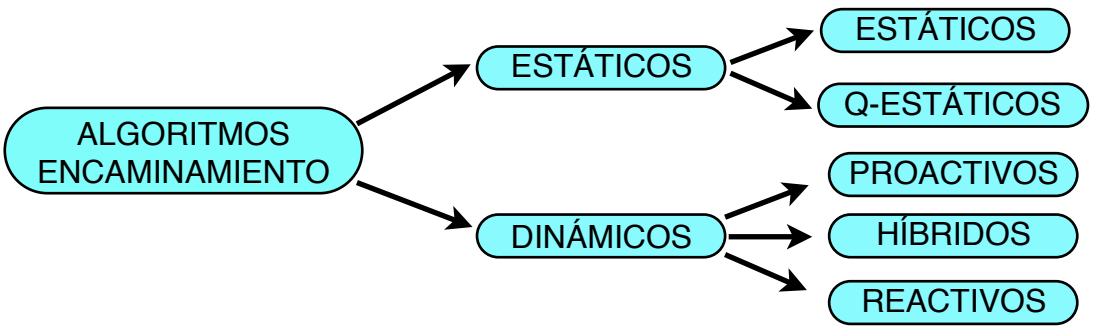


Figura A.2: Segunda clasificación

➡ **Estáticos:** Estos algoritmos no calculan rutas, sino que sus tablas de encaminamiento se generan al comienzo y no se vuelven a modificar siguiendo siempre las mismas rutas para llegar al destino.

➡ **Q-Estáticos:** Los algoritmos de este tipo son muy parecidos a los estáticos pero con la diferencia que existen rutas alternativas preestablecidas en las tablas de encaminamiento.

➡ **Proactivo:** Estos algoritmos se basan en tablas y calculan la ruta previamente para todos los destinos. Cuando se desea enviar se sigue la ruta preestablecida anteriormente. Ej. *OLSR, TBRPF, FRP, HSLS, MMRP, OSPF...*

➡ **Híbrido:** Este tipo de algoritmos no es usual ya que contienen las ventajas de los proactivos y los reactivos. Ej. *BMX, BATMAN...*

➡ **Reactivo:** En este tipo se calcula la ruta en demanda, es decir, una vez que se crea o llega un nuevo paquete. Ej. *AODV...*

El protocolo de encaminamiento *OLSR* utilizado en el proyecto es del tipo proactivo, esto le hace más lento que los otros dos utilizados como se observa en el apartado de pruebas, mientras que los protocolos *BMX* y *BATMAN* se clasifican como híbridos.

Finalmente se detallan las características de cada protocolo.

OLSR o Protocolo de Encaminamiento por Optimización del Estado del Enlace [A2]que se desarrolla para redes móviles *ad hoc*. Cada nodo selecciona un conjunto de nodos de su vecino como "enlaces multipunto" (MPR). En OLSR, sólo los nodos seleccionados como tal MPRs, son responsables de la transmisión de control de tráfico, destinados a la difusión en toda la red. Los MPRs garantizan un mecanismo de control de inundaciones de tráfico mediante la reducción del número de transmisiones necesarias.

Cada nodo tiene la información de toda la topología de la red y genera un grafo ponderado con las flechas de los enlaces. El peso de cada flecha se calcula con la métrica *ETX* (*Expected Transmission Count*) consiste en una técnica sencilla que favorece a los enlaces más confiables y de mayor capacidad. Esta se calcula de la siguiente forma:

Se cuentan los paquetes enviados, recibidos y perdidos para generar un porcentaje de calidad *LQ* (*Link Quality*) Si por ejemplo se han enviado 10 paquetes y se han perdido 4 entonces 4/10 = 0,4 o 40%. Otra variable similar *NLQ* (*Neighbor Link Quality*) o *ILQ* (*Inverse Link Quality*) calcula el enlace a la inversa. Una vez determinados estos parámetros se calcula la formula siguiente que nos dará una estimación de cuantas veces hay que enviar un paquete para que llegue al destino.

$$ETX = \frac{1}{(LQ \times NLQ)}$$

B.A.T.M.A.N. o *Better Approach To Mobile Ad-hoc Networking* es un protocolo de encaminamiento diseñado para redes *ad-hoc*. Tiene la peculiaridad de solo guardar la información del estado de los vecinos, es decir, si se desea enviar un paquete el protocolo calculará el mejor vecino para llegar al destino, pero una vez llegado a ese vecino, él es el encargado que el paquete siga la ruta y calculará el siguiente mejor vecino hasta llegar al destino. [A3]Cada nodo propaga su propia existencia en la malla simplemente inundando con mensajes de originador. (OGM) Cada OGM puede ser identificado unívocamente por un número de secuencia y la dirección IP del nodo que inicio el mensaje. Los mensaje OGM enviados a través de enlaces pobres o muy congestionados sufrirán retados y paquetes perdidos. Los OGM inundados vía enlaces buenos y descongestionados se propagaran mejor, más rápido y en forma más fiable.

Cada nodo selecciona uno de sus vecinos como el mejor vecino y pasarela hacia un cierto y especifico nodo. Este proceso de identificar el mejor vecino hacia un nodo distante se denomina *Node Ranking*. Al vecino seleccionado se lo refiere como el vecino mejor *"rankeado"*. Cada nodo mantiene un vecino mejor *rankeado* para cada otro nodo de la red. El algoritmo del *"nodo mejor rankeado"* simplemente selecciona al vecino vía el cual recibió y acepto el OGM mas reciente desde el nodo que inicio el OGM.

BMX o *B.A.T.M.A.N. Experimetal* es un protocolo que surgió de la misma rama de desarrollo que *BATMAN* separandose posteriormente. Este protocolo acepta plugins, maneja mejor los modos de direccionamiento y varía sustancialmente su algoritmo de toma de decisión del mejor vecino.

Anexo B: *Script* para generar el *firmware*

Código del *script* para generar el *firmware*.

```
#!/bin/sh

#Comprobamos si esta la carpeta de descargas global para sino crearla
DESCARGAS='dl'
if test -d $DESCARGAS
    then wait
    else mkdir $DESCARGAS
fi

#Descargamos el codigo fuente a la carpeta deseada
echo -n "¿Desea descargar las fuentes de OpenWRT?(s/n) "
read respuesta1
if [ $respuesta1 = 's' ] || [ $respuesta1 = 'S' ]
    then
            echo -n "Escriba el directorio donde desea descargar las
fuentes: "
            read directorio

            #Pre-requisitos para el entorno de compilacion
            TMP=tmp

            paquete1='subversion'
            paquete2='build-essential'
            paquete3='ncurses-term'
            paquete4='libncurses5-dev'
            paquete5='zlib1g-dev'
            paquete6='gawk'
            paquete7='bison'
            paquete8='flex'
            paquete9='autoconf'

            dpkg -l | grep $paquete1 | awk '{print $2}' > $TMP
            dpkg -l | grep $paquete2 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete3 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete4 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete5 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete6 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete7 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete8 | awk '{print $2}' >> $TMP
            dpkg -l | grep $paquete9 | awk '{print $2}' >> $TMP

            if grep -n $paquete1 $TMP
                  then
                        echo '  instalado'
```

```
else
      sudo apt-get install $paquete1
fi
if grep -n $paquete2 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete2
fi
if grep -n $paquete3 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete3
fi
if grep -n $paquete4 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete4
fi
if grep -n $paquete5 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete5
fi
if grep -n $paquete6 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete6
fi
if grep -n $paquete7 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete7
fi
if grep -n $paquete8 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete8
fi
if grep -n $paquete9 $TMP
      then
            echo '  instalado'
else
      sudo apt-get install $paquete9
```

```
            fi
            rm -f $TMP

            svn co svn://svn.openwrt.org/openwrt/tags/8.09/
$directorio/

            #Entramos al codigo fuente
            cd $directorio

            #Enlazamos simbolicamente el parche de archivos
            ln -s ../files

            #Enlazamos simbolicamente la carpeta de descargas global
            ln -s ../dl

            #Enlazamos simbolicamente los paquetes que se añadiran
            cd package
            ln -s ../../GsfGuifi.net/meshipcalc
            ln -s ../../GsfGuifi.net/bmxd bmx
            cd ..
else
    if [ $respuesta1 = 'n' ] || [ $respuesta1 = 'N' ]
            then
                  echo -n "Escriba el directorio donde se hallan las
fuentes de OpenWRT: "
                  read directorio
                  if test -d $directorio
                        then
                              #Entramos al codigo fuente
                              cd $directorio
                  else
                        echo 'ERROR: directorio no hallado'
                        exit
                  fi
    else
            echo 'ERROR: elija si o no'
            exit
    fi
fi

echo -n "¿Desea actualizar e instalar todos los paquetes de OpenWRT?
(s/n) "
read respuesta2
if [ $respuesta2 = 's' ] || [ $respuesta2 = 'S' ]
    then
            #Actualizamos e instalamos todos los paquetes de OpenWRT
            ./scripts/feeds update -a
            ./scripts/feeds install -a
else
    if [ $respuesta2 = 'n' ] || [ $respuesta2 = 'N' ]
```

```
            then
                  wait
    else
            echo 'ERROR: elija si o no'
            exit
    fi
fi

echo -n "¿Desea inicializar la configuración de OpenWRT?(s/n) "
read respuesta3
if [ $respuesta3 = 's' ] || [ $respuesta3 = 'S' ]
    then
            #Copiamos el archivo de configuración del sistema
             cp ../alix2c2/.config .config
else
    if [ $respuesta3 = 'n' ] || [ $respuesta3 = 'N' ]
            archivo_conf='.config'
            then
                  if test -e $archivo_conf
                        then
                              wait
                  else
                        #Copiamos el archivo de configuración del
sistema
                         cp ../alix2c2/.config .config
                  fi
    else
            echo 'ERROR: elija si o no'
            exit
    fi
fi

echo -n "¿Desea validar la configuración de OpenWRT?(s/n) "
read respuesta4
if [ $respuesta4 = 's' ] || [ $respuesta4 = 'S' ]
    then
            #Validamos la configuracion
            make menuconfig
else
    if [ $respuesta4 = 'n' ] || [ $respuesta4 = 'N' ]
            then
                  wait
    else
            echo 'ERROR: elija si o no'
            exit
    fi
fi

echo -n "¿Desea compilar la imagen de OpenWRT?(s/n) "
read respuesta5
```

```
if [ $respuesta5 = 's' ] || [ $respuesta5 = 'S' ]
    then
            #Compilamos
            echo 'Empezando la compilación'
            make world V=99
            echo 'Compilación finalizada'
else
    if [ $respuesta5 = 'n' ] || [ $respuesta5 = 'N' ]
            then
                  wait
    else
            echo 'ERROR: elija si o no'
            exit
    fi
fi

#Flasheado automatico
echo -n "¿Desea flashear automaticamente la imagen a la CompactFlash?
(s/n) "
read respuesta6
if [ $respuesta6 = 's' ] || [ $respuesta6 = 'S' ]
    then
            echo "¿Que nombre segun la tabla de particiones recibe la
CompactFlash?"
            echo "Escriba solo la ultima letra de /dev/sdX donde X sea
a, b, c, d..."
            echo "Por defecto si no escribe nada X sera c"
            echo "Si no lo tiene claro NO CONTINUE o tendra
consecuencias catrastroficas"
            echo -n "letra: "
            read letra
            if test -e $letra
                  then
                        dispositivo=/dev/sdc
            else
                  dispositivo=/dev/sd$letra
            fi
            montado='/media/disk'
            #Detectamos la CompactFlash
            if test -e $dispositivo && test -e $montado
                  then
                        echo 'CompactFlash detectada'
                        #Flasheamos la imagen a la CompactFlash
                        cd bin
                        sudo dd if=openwrt-x86-squashfs.image of=
$dispositivo
                        sync

                        #Visualizamos que el proceso ha finalizado
correctamente
```

```
                        echo ''
                        echo '¡¡PROCESO FINALIZADO!!, extraiga la
CompactFlash e insertela en el nodo'
                        echo ''
            else
                  echo 'Inserte CompactFlash'
                  while test ! -e $dispositivo
                  do
                        wait
                  done
                  while test ! -e $montado
                  do
                        wait
                  done
                  echo 'CompactFlash detectada'
                  #Flasheamos la imagen a la CompactFlash
                  cd bin
                  sudo dd if=openwrt-x86-squashfs.image of=
$dispositivo
                  sync

                  #Visualizamos que el proceso ha finalizado
correctamente
                  echo ''
                  echo '¡¡PROCESO FINALIZADO!!, extraiga la
CompactFlash e insertela en el nodo'
                  echo ''
            fi
else
    if [ $respuesta6 = 'n' ] || [ $respuesta6 = 'N' ]
            then
                  #Visualizamos que el proceso ha finalizado
correctamente
                  echo ''
                  echo '¡¡PROCESO FINALIZADO!!, grabe la imagen a la
CompactFlash'
                  echo ''
    else
            echo 'ERROR: elija si o no'
            exit
    fi

fi
```

Anexo C: *Scripts* de las pruebas realizadas

Código del *script* para capturar la tabla de encaminamiento de BMX.

```
#!/bin/sh

reg=/usr/local/reg_bmx

for i in $(seq 60)
do
    date >> $reg
    bmxd -cd 8 >> $reg
    sleep 1
done
```

Código del *script* para capturar la tabla de encaminamiento de BATMAN.

```
#!/bin/sh

reg=/usr/local/reg_batman

for i in $(seq 60)
do
    date >> $reg
    batmand -cd 1 -b >> $reg
    sleep 1
done
```

Código del *script* para capturar la tabla de encaminamiento de OLSR.

```
#!/bin/sh

reg=/usr/local/reg_olsr

for i in $(seq 60)
do
    date >> $reg
    echo "/neigh" | nc localhost 2006 >> $reg
    sleep 1
done
```

Código del *script* para la prueba de velocidad de transferencia con BMX.

```
#!/bin/sh

wget -o /usr/local/regtrans_bmx -O /dev/null http://172.22.0.XXX/cgi-bin/cgi-bin-dev-zero.bin
```

Código del *script* para la prueba de velocidad de transferencia con BATMAN.

```
#!/bin/sh

wget -o /usr/local/regtrans_batman -O /dev/null http://172.26.0.XXX/cgi-bin/cgi-bin-dev-zero.bin
```

Código del *script* para la prueba de velocidad de transferencia con OLSR.

```
#!/bin/sh

wget -o /usr/local/regtrans_olsr -O /dev/null http://172.18.0.XXX/cgi-bin/cgi-bin-dev-zero.bin
```