UNIVERSITAT OBERTA DE CATALUNYA
Máster Oficial en Software Libre

# Protocolos de encaminamiento en redes inalámbricas mesh: un estudio teórico y experimental

Albert Batiste Troyano
Tutor: Pablo Neira Ayuso, Universidad de Sevilla

Sevilla, junio de 2011

# Índice general

# Capítulo 1

# Introducción

Desde que se desarrollaron las primeras máquinas computadoras, una de las grandes líneas de investigación e innovación en el campo de la informática ha sido la de las redes de computadores, cómo interconectar equipos alejados en el espacio para que sus usuarios puedan comunicarse y compartir servicios e información.

Se ha avanzado tanto en esta materia, que hoy en día, el mundo que conocemos sería impensable sin la más extensa red de ordenadores que ha creado la humanidad, Internet. Desde las primeras redes de los 60, como ARPANET, hasta el Internet actual, la técnica ha recorrido un vertiginoso camino, en el que se han desarrollado infinidad de protocolos, dispositivos y estándares destinados a interconectar computadores.

En este trabajo estudiaremos, concretamente, los aspectos relacionados con el diseño, la arquitectura, la implementación, el despliegue y la evaluación experimental para redes inalámbricas mesh, centrándonos, sobre todo, en la forma en que los distintos equipos o nodos mesh intercambian datos y deciden qué camino debe seguir un mensaje para llegar del origen al destino. Es decir, analizaremos, de forma teórica y experimental, distintos protocolos de encaminamiento o enrutamiento usados en redes inalámbricas mesh.

Para desarrollar el trabajo será necesario hacer un estudio del arte que describa los protocolos más extendidos para este tipo de redes y, posteriormente, desplegar una pequeña red mesh experimental sobre la que se probarán y compararán dichos protocolos de enrutamiento. Finalmente, a partir de las pruebas realizadas, extraeremos una serie de resultados experimentales

que nos servirán para redactar las conclusiones.

Los protocolos elegidos para el estudio serán: DSR (*Dynamic Source Routing*), AODV (*Ad-hoc On-demand Distance Vector*), OLSR (*Optimized Link State Routing*), Babel, BATMAN (*Better Aproach To Mobile Ad-hoc Networking*) e IEEE[1] 802.11s. Los protocolos DSR y OLSR están incluidos en el estudio teórico, pero no en el experimental, ya que consideramos necesario describirlos por su importancia y por ser precursores de otros protocolos. Elegimos estos protocolos en concreto por ser los más ampliamente usados a la hora de desplegar redes mesh y porque existen implementaciones libres, lo que nos da la posibilidad de usarlos, estudiar en profundidad cómo están diseñados los algoritmos e incluso introducir alguna modificación, si fuera necesario, en el código.

Podríamos describir de forma breve (en 1.3.2 lo hacemos de manera más detallada) una red mesh como un conjunto de dispositivos conectados de manera inalámbrica formando una topología de malla (*mesh*, en inglés) en la que cada nodo puede establecer un enlace inalámbrico con uno o más vecinos. Dichos dispositivos no necesitan nodos especiales a los que conectarse para formar parte de la red. Dentro de esta red, los nodos deben ser capaces de reenviar paquetes de datos hacia otros nodos, distribuyendo la tarea de encaminar o enrutar la información entre todos los nodos de la red. La decisión sobre qué nodo debe enrutar un paquete determinado se hace de forma dinámica, según la conectividad de la red.

Esta definición se puede aplicar también a lo que se conoce como red ad-hoc. Pero a diferencia de esta, en una red mesh existen nodos que funcionan también como puntos de acceso, a los que se conectan otros dispositivos (clientes) para formar parte de la red. Finalmente, ciertos nodos mesh pueden actuar como pasarela o portal mediante el cual interconectar la red mesh a redes de otro tipo (una red cableada o Internet, por ejemplo).

El hardware necesario para implementar una red mesh es relativamente económico y accesible, ya que es el mismo que se utiliza para redes WLAN (*Wireless Local Area Network*) convencionales. Por otra parte, esta tecnología opera en una banda del espectro radioeléctrico que es de libre uso, sin licencia, por lo que cualquiera puede usar el medio sin necesidad de trámites o permisos.

Todos estos hechos hacen de las redes mesh soluciones ideales en escenarios donde es complicado desplegar una infraestructura cableada, debido a

[1] Institute of Electrical and Electronics Engineers

dificultades técnicas y/o económicas. Posibles escenarios son: un edificio en el que sus habitantes se interconectan de manera inalámbrica para formar una red en la que compartir información, servicios y, adicionalmente, una o varias conexiones a Internet, un evento puntual en el que queremos desplegar una red inalámbrica (una feria, un festival, un congreso), áreas rurales a las que no llega la infraestructura comercial necesaria para tener conexión ADSL o un conjunto de usuarios que decide crear una red comunitaria y abierta, que va creciendo cuantos más usuarios se añaden.

Consideramos este último ejemplo especialmente interesante. Desde hace unos años han surgido, alrededor de todo el mundo, redes comunitarias inalámbricas basadas en topologías mesh. Son redes sin ánimo de lucro, con un origen ciudadano, autogestionadas y autofinanciadas. En ellas sus usuarios comparten libremente todo tipo de recursos y servicios digitales (intercambio de archivos, servidores ftp, chats, correo, etc.). Pueden abarcar desde un edificio hasta áreas de varios cientos de kilómetros cuadrados.

Además, estas iniciativas suelen usar y desarrollar software libre para hacer funcionar sus redes, permitiendo que otras personas y comunidades aprovechen esos conocimientos para crear nuevas redes mesh comunitarias. Existen numerosas aplicaciones y protocolos con licencia libre para implementar redes mesh. Incluso el núcleo Linux, en sus últimas versiones, ya incluye funcionalidad mesh.

La comunidad del software libre y las redes mesh comunitarias tienen algunos rasgos comunes:

- Organizaciones de base, ciudadanas.
- Despliegue espontáneo.
- Uso de tecnología barata y estándar.
- Sin ánimo de lucro.
- Coordinación informal.

Estamos hablando, por lo tanto, de un nuevo campo en el mundo de las redes informáticas que está destinado a ser protagonista en los próximos años, de la mano del software libre y las iniciativas comunitarias.

## 1.1. Objetivos.

Veamos, de forma detallada, los objetivos que nos han llevado a realizar este trabajo:

- Ofrecer un informe del estado del arte de las implementaciones de protocolos de enrutado mesh. Concretamente, se describirán los siguientes protocolos: AODV, OLSR, Babel, B.A.T.M.A.N., IEEE 802.11s y DSR. Como ya hemos mencionado anteriormente, la razón principal para escoger estos protocolos es que son los más usados, los que ofrecen mejores resultados y además cuentan con implementaciones libres.
- Desplegar una red mesh para evaluar experimentalmente, y en distintos escenarios, diferentes implementaciones de los protocolos anteriormente citados (menos DSR y OLSR). Se recogerán datos objetivos para poder elaborar una comparación de su funcionamiento, entre los que destacamos:
  - Latencia, en base al RTT (round-trip time) que depende del número de saltos entre los nodos mesh.
  - Número máximo de sesiones por segundo.
  - Throughput.
  - Tiempo de convergencia de la red debido a la caida de un nodo.

## 1.2. Planificación.

El plazo temporal en el que se desarrollará el proyecto es de 5 meses, desde febrero hasta junio de 2011. Consta de dos partes diferenciadas. La primera es la parte teórica y de estudio del arte. En ella describiremos las características de las redes mesh y varios protocolos de enrutamiento. En la segunda, aplicaremos lo visto en la primera parte en un entorno real, desplegando una pequeña red mesh. Sobre esta red se realizará la parte central del proyecto, el estudio comparativo de distintos protocolos mesh, en los términos explicados en el punto anterior. Finalmente, a partir de los resultados obtenidos, extraeremos una serie de conclusiones que muestren el mejor o peor funcionamiento de cada protocolo en distintos escenarios y situaciones. Veamos en detalle la planificación temporal mediante una tabla de tareas.

**Tareas.**

1. Recopilación y consulta de información.
   - Objetivo: adquirir la documentación y conocimientos necesarios para dominar los conceptos básicos en cuanto a redes ad-hoc y mesh. Recopilar la información necesaria para redactar el estado del arte y la bibliografía.
   - Duración: 15 días (las actividades se realizarán de manera simultánea).
   - Actividades:
     - Búsqueda de información on-line. 15 días.
     - Consulta de bibliografía relacionada. 15 días.
     - Lectura. 15 días.
2. Redacción del estado del arte.
   - Objetivo: redactar un capítulo que describa los avances conseguidos en materia de redes informáticas ad-hoc, centrándonos en las redes mesh, y hacer una clasificación de los protocolos de enrutamiento más importantes para este tipo de redes. Poner algunos ejemplos de despliegues reales de redes mesh.
   - Duración: 15 días.
   - Actividades:
     - Recopilación de protocolos. 2 días.
     - Clasificación y elección de los protocolos a estudiar. 2 días.
     - Redacción. 11 días.
3. Búsqueda, elección y adquisición de materiales
   - Objetivo: Conocer qué hardware es el adecuado para las necesidades del proyecto, elegir, reciclar y adquirir componentes.
   - Duración: 15 días (las actividades se realizarán de manera simultánea).
   - Actividades:
     - Buscar información sobre hardware para redes mesh. 15 días.
     - Reciclar componentes. 15 días.
     - Adquirir componentes nuevos. 15 días.

4. Montaje, instalación y configuración de los nodos mesh

   - Objetivo: ensamblar, instalar software y configurar los nodos mesh. Buscar las localizaciones y montar los nodos.
   - Duración: 30 días.
   - Actividades:
     - Ensamblado y montaje de nodos. 5 días.
     - Instalación y configuración del software. 10 días.
     - Búsqueda de localizaciones. 5 días.
     - Instalación de los nodos en las localizaciones. 10 días.

5. Puesta en funcionamiento, resolución de incidencias

   - Objetivo: configurar de forma adecuada los nodos para conseguir una red mesh operativa.
   - Duración: 10 días (las actividades se realizarán de manera simultánea).
   - Actividades:
     - Puesta en marcha y configuración de cada nodo. 10 días.
     - Resolución de incidencias. 10 días.

6. Realización de pruebas

   - Objetivo: realización de las pruebas elegidas sobre la red para comparar los protocolos bajo estudio y extraer resultados y conclusiones para distintos escenarios.
   - Duración: 8 días.
   - Actividades:
     - Configuración de los escenarios de prueba. 2 días.
     - Generación de tráfico. 2 días.
     - Extracción de resultados. 4 días.

7. Redacción del estudio comparativo y las conclusiones

   - Objetivo: redacción de lo extraído en la tarea anterior y de las conclusiones sobre la eficiencia de cada protocolo en los distintos escenarios.
   - Duración: 10 días (actividades parcialmente solapadas).

- Actividades:
  - Clasificación y ordenación de los resultados obtenidos. 5 días.
  - Redacción y diseño de gráficos. 7 días.

| WBS | Nombre | Inicio | Fin | Duración |
|---|---|---|---|---|
| 1 | **Recopilación y consulta de información** | **feb 15** | **mar 7** | **15d** |
| 1.1 | Búsqueda de información on-line | feb 15 | mar 7 | 15d |
| 1.2 | Consulta de bibliografía relacionada | feb 15 | mar 7 | 15d |
| 1.3 | Lectura | feb 15 | mar 7 | 15d |
| 2 | **Redacción del estado del arte** | **feb 24** | **mar 16** | **15d** |
| 2.1 | Recopilación de protocolos | feb 24 | feb 25 | 2d |
| 2.2 | Clasificación y elección de los protocolos a estudiar | feb 28 | mar 1 | 2d |
| 2.3 | Redacción | mar 2 | mar 16 | 11d |
| 3 | **Búsqueda, elección y adquisición de materiales** | **mar 16** | **abr 5** | **15d** |
| 3.1 | Buscar información sobre hardware para redes mesh | mar 16 | abr 5 | 15d |
| 3.2 | Reciclar componentes | mar 16 | abr 5 | 15d |
| 3.3 | Adquirir componentes nuevos | mar 16 | abr 5 | 15d |
| 4 | **Montaje, instalación y configuración de los nodos mesh** | **abr 6** | **may 17** | **30d** |
| 4.1 | Ensamblado y montaje de nodos | abr 6 | abr 12 | 5d |
| 4.2 | Instalación y configuración del software | abr 13 | abr 26 | 10d |
| 4.3 | Búsqueda de localizaciones para los nodos | abr 27 | may 3 | 5d |
| 4.4 | Instalación de los nodos en las localizaciones | may 4 | may 17 | 10d |
| 5 | **Puesta en funcionamiento, resolución de incidencias** | **may 18** | **may 31** | **10d** |
| 5.1 | Puesta en marcha y configuración de cada nodo | may 18 | may 31 | 10d |
| 5.2 | Resolución de incidencias | may 18 | may 31 | 10d |
| 6 | **Realización de pruebas** | **jun 1** | **jun 10** | **8d** |
| 6.1 | Configuración de los escenarios de pruebas | jun 1 | jun 2 | 2d |
| 6.2 | Generación de tráfico | jun 3 | jun 6 | 2d |
| 6.3 | Extracción de resultados y conclusiones | jun 7 | jun 10 | 4d |
| 7 | **Redacción del estudio comparativo y las conclusiones** | **jun 7** | **jun 20** | **10d** |
| 7.1 | Clasificación y ordenación de los resultados obtenidos | jun 7 | jun 13 | 5d |
| 7.2 | Redacción de resultados, generación de gráficas | jun 10 | jun 20 | 7d |

## 1.3. Conceptos básicos.

### 1.3.1. WLAN/IEEE 802.11.

Una WLAN (*Wireless Local Area Network*) es una red inalámbrica de dispositivos que generalmente se rige por el estándar IEEE 802.11. 802.11 define los dos niveles inferiores del modelo OSI para redes de este tipo. Es una norma aprobada por el IEEE, asociación internacional dedicada a la investigación y estandarización científico-técnica. Además existe la marca Wi-Fi, de la Wi-Fi Alliance, organización empresarial que adopta, prueba y certifica que los dispositivos inalámbricos cumplen los estándares 802.11 relacionados con redes WLAN.

La primera versión de 802.11 (802.11-1997) fue aprobada en el año 1997. En 1999 se aprobaron 802.11a y 802.11b, punto de inflexión en el proceso de difusión de las redes inalámbricas para usos domésticos. En 2003 fue aprobado el estándar 802.11g, rápidamente adoptado por las empresas suministradoras de dispositivos inalámbricos y más rápido que 802.11b (11 Mbit/s frente a 54 Mbit/s). Estas dos normas operan en la frecuencia de 2.4GHz. Actualmente existe una nueva revisión, la 802.11n, con velocidades de 600 Mbit/s y que puede operar en 2.4 Ghz o en 5 Ghz. Las tres son compatibles y los equipos inalámbricos que podemos encontrar en el mercado en la actualidad implementan uno, dos o los tres estándares.

En una red 802.11 podemos encontrar los siguientes componentes, aunque, como veremos justo después, no todos los elementos son necesarios para desplegarla:

- Estación (*Station*, STA): dispositivo equipado con una interfaz 802.11 (PC, PDA, teléfono móvil).
- Punto de acceso (*Access Point*, AP): dispositivo que interconecta varias STA para formar una red. Muchas veces incorporan funcionalidades extra, que los convierten en *routers* (enrutadores), con capacidad para encaminar paquetes de datos y asignar direcciones IP (mediante el protocolo DHCP, *Dynamic Host Configuration Protocol*).
- Portal: dispositivo que interconecta una red 802.11 con otra de tipo 802. Generalmente, en una WLAN, encontraremos el AP y el portal en el mismo dispositivo físico.

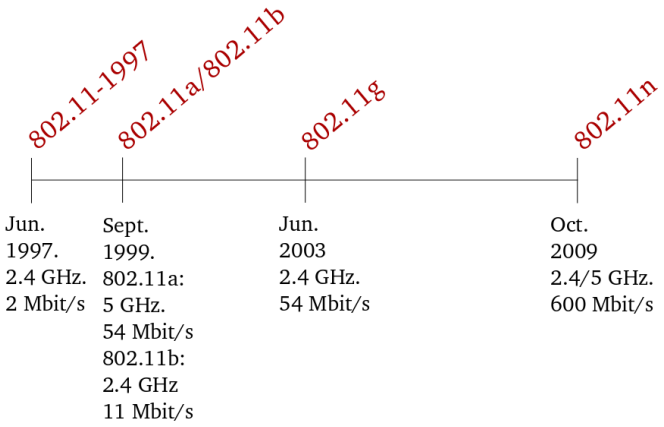


Figura 1.1: Línea temporal de las revisiones para IEEE 802.11.

- Sistema de distribución (*Distribution System*, DS): varios AP's pueden conectarse para formar una columna vertebral (*backbone*). El conjunto de AP's conectados conforman el sistema de distribución. Normalmente, el DS suele estar implementado sobre Ethernet (IEEE 802.3).

- BSS (*Basic Service Set*): el conjunto formado por un AP y sus STA's asociadas constituye un BSS. Un BSS se identifica por su BSSID (identificador de 6 bytes). Ver figura 1.2.

- ESS (*Extended Service Set*): el conjunto de WLAN's interconectadas, formado por cada BSS, sus respectivos AP y STA's y el sistema de distribución constituyen un ESS, que es visto como una sola red 802 para los niveles superiores del modelo OSI. Un ESS se identifica a través de su ESSID (identificador de hasta 32 carácteres ASCII). Dentro de un ESS las estaciones pueden realizar itinerancia o *roaming*, es decir, pueden moverse dentro del ESS y cambiar de AP según la calidad de la señal. Ver figura 1.3.

### Modo infraestructura.

Una red WLAN puede operar de dos maneras distintas, en modo infraestructura o en modo ad-hoc.

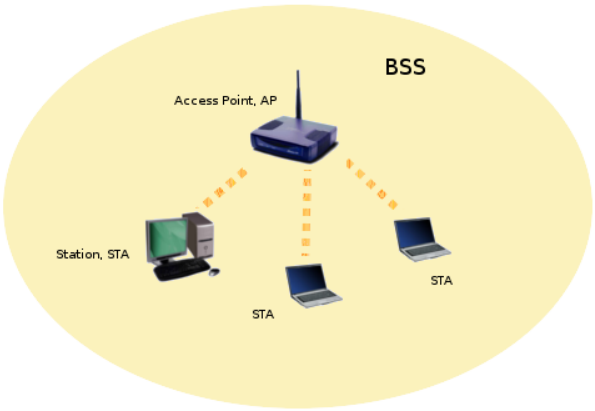


Figura 1.2: IEEE 802.11 Basic Service Set.

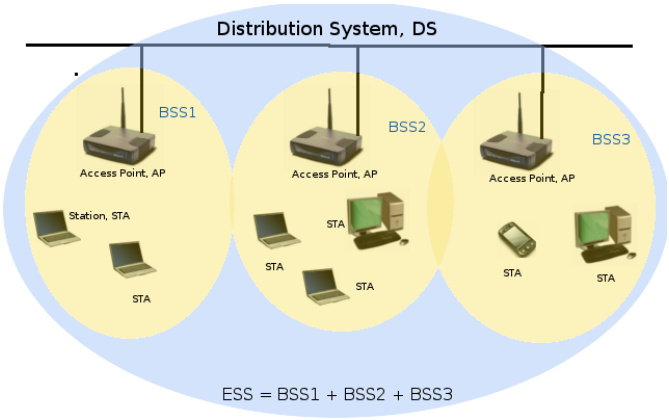


Figura 1.3: IEEE 802.11 Extended Service Set.

El modo infraestructura es el caso típico que hemos visto en el apartado anterior. Es decir, el BSS está conformado por un AP y distintas STA's que se conectan a él para formar la WLAN. Este es el tipo más común, el que encontramos en la mayoría de hogares y empresas con conexión ADSL, donde el AP funciona como portal hacia Internet.

### Modo ad-hoc.

En el modo ad-hoc, la WLAN está formada solamente por STA's. Es decir, no necesitamos un AP central que conforme el BSS, si no que las

estaciones se conectan entre ellas, punto a punto, para formar una WLAN. Cada STA actúa como emisor, receptor y transmisor de información, por lo que se hacen necesarios protocolos de enrutamiento específicos que permitan a los nodos reenviar paquetes.

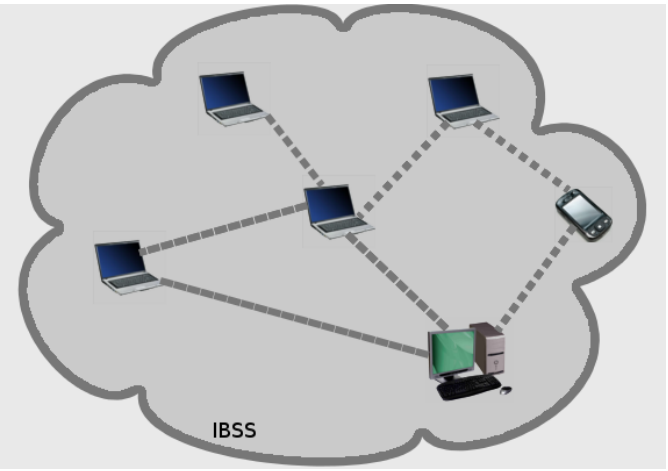


Figura 1.4: Red ad-hoc.

Un conjunto de estaciones que forman una red ad-hoc se denomina IBSS (*Independent Basic Service Set*), fig. 1.4.

### 1.3.2. Redes ad-hoc.

Como hemos dicho anteriormente, una red ad-hoc es aquella en la que los nodos se interconectan directamente, de forma inalámbrica, sin necesidad de dispositivos especiales que centralicen la gestión de la red. Existen muchos tipos de redes ad-hoc que añaden características especiales a esta definición. Principalmente, las redes ad-hoc móviles (*Mobile Ad-hoc Network*, MANET) y las redes mesh.

#### Redes MANET.

Una red ad-hoc móvil, o MANET, es un tipo de red ad-hoc en la que las estaciones tienen propiedades de autoconfiguración y movilidad. Es decir, están montadas sobre plataformas móviles. Los dispositivos, al moverse, cambian sus enlaces con los demás dinámicamente, según la potencia de la señal, el estado del enlace o el ancho de banda del mismo. Cada dispositivo

inalámbrico que forma una red MANET debe ser capaz de enrutar paquetes hacia otros nodos de la red. El mayor reto de este tipo de redes es, por lo tanto, mantener la información acerca de la topología de la red correcta y actualizada en todos los nodos a medida que estos se van desplazando.

Existen escenarios reales donde este tipo de redes son especialmente útiles. Por ejemplo, en el caso de un accidente, catástrofe natural u otras situaciones de emergencia, en las que los canales normales de comunicación (redes cableadas, teléfonos móviles) no son viables. Las redes MANET permiten a los equipos médicos y de salvamento estar interconectados mientras se mueven por el escenario, ya sea en un vehículo motorizado o a pie. También son útiles en situaciones de tráfico rodado urbano, en un escenario en el que los vehículos están equipados con sistemas de conexión inalámbrica que les permiten comunicarse entre ellos y con unidades estáticas que les ofrecen información sobre el estado del tráfico.

**Redes mesh.**

Dentro de las redes ad-hoc, existe un tipo especial llamado mesh, en el que se centra nuestro trabajo. La topología mesh combina las características de los modos infraestructura y ad-hoc.

Una red mesh básica es igual que una red ad-hoc. Un conjunto de dispositivos conectados de manera inalámbrica de forma distribuida, sin estructuras centrales que gestionen el funcionamiento y el tráfico de datos. Pero aparte de los nodos mesh básicos, encargados de enrutar el tráfico y de actualizar la información sobre la topología de la red, existen algunos nodos que pueden funcionar como AP's a los que se conectan clientes equipados con interfaces inalámbricas. Además, algunos nodos pueden funcionar como pasarela hacia otro tipo de redes.

Para describir la arquitectura y funcionamiento de una red mesh, nos basaremos en el borrador de estándar del IEEE, el 802.11s [17]. Agradecemos a los miembros del IEEE 802.11s *Task Group* habernos proporcionado el borrador actualizado, de abril de 2011. Veamos los distintos componentes que define dicho estándar:

- *Mesh Station* (Mesh STA): unidad básica de la red. Una mesh STA puede comunicarse con otras mesh STA de la red. Pero, al contrario que otros dispositivos 802.11, pueden intercambiar paquetes a través

de varios saltos inalámbricos, permitiendo alcanzar mesh STA que no están dentro del rango propio de cobertura inalámbrica. Como un punto de acceso, una mesh STA tiene la capacidad de reenviar (y por lo tanto enrutar) tramas de datos para las que dicha mesh STA no es la destinataria.

- *Mesh Gate:* dispositivo encargado de integrar una red mesh con un sistema de distribución (*Distribution System*, DS) 802.11. De esta manera, la red puede conectarse con redes 802.11 cuyos puntos de acceso formen parte del DS.

El conjunto de mesh STA y mesh *gates* conforman un MBSS (*Mesh Basic Service Set*).

Estos son los dispositivos que son exclusivos de arquitecturas mesh 802.11s. Pero en el borrador también se explica que existe la posibilidad de integrar en un mismo dispositivo cualquier combinación de mesh STA, punto de acceso, mesh *gate* y portal (dispositivo que conecta redes 802.11 con otras de otro tipo, como 802.3). Generalmente encontraremos las siguientes combinaciones de funcionalidades en una red mesh:

- *Mesh Access Point* (MAP): es una mesh STA con funcionalidades de punto de acceso. Es decir, además de las tareas propias de una mesh STA, un MAP tiene la capacidad de interconectar dispositivos inalámbricos 802.11, formando una WLAN, y añadirlos a la red mesh.
- *Mesh Portal* (MPP): permite conectar la red mesh con otras redes que no son 802.11. Una red Ethernet cableada, por ejemplo.

Más allá de la arquitectura y los elementos hardware de una red mesh, se hace necesario contar con aplicaciones y protocolos que permitan a la red funcionar. El punto clave en la operatividad de estas redes es el enrutamiento de paquetes. Es decir, decidir qué ruta, qué serie consecutiva de nodos, debe atravesar un paquete para llegar de un origen a un destino. Además, estos protocolos deben ser capaces de descubrir cambios en la topología o en los enlaces entre nodos y actualizar la información correspondiente.

A día de hoy, todavía no existe un estándar aprobado internacionalmente que defina el funcionamiento de las redes mesh, por lo que existen numerosos protocolos y tecnologías que implementan este tipo de redes.

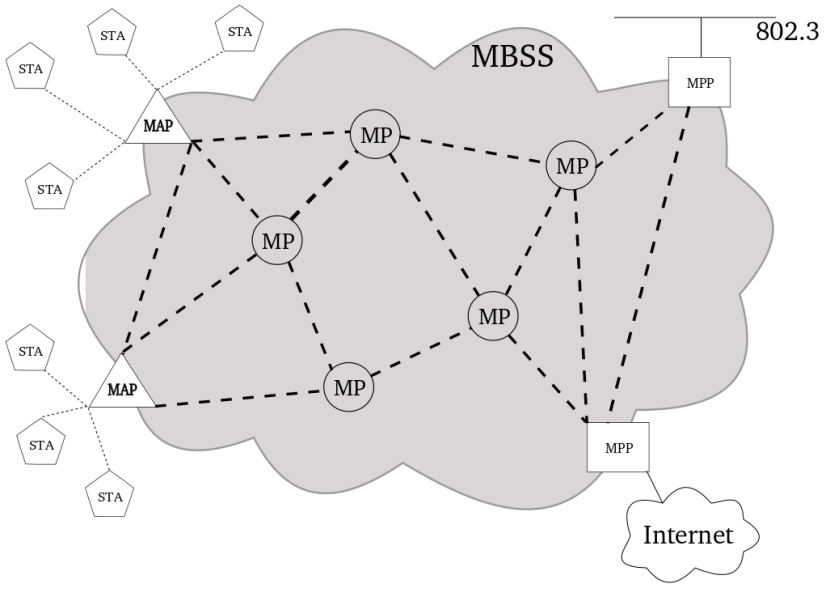


Figura 1.5: Red mesh.

Podemos clasificar los protocolos de enrutamiento mesh según varios parámetros o características. Si atendemos a la forma en que se descubren y seleccionan las rutas:

- Proactivos: periódicamente se emiten paquetes de datos utilizados para descubrir nuevos nodos en la red y la ruta hacia ellos, presuponiendo que en algún momento dichas rutas serán necesarias y utilizadas. Se usa una tabla para mantener actualizadas todas las rutas a los distintos nodos. Las principales desventajas que presenta este tipo de protocolo son la generación de tráfico de mantenimiento y actualización de las tablas de enrutado, que disminuye el *throughput* (aprovechamiento real del ancho de banda de la red), y la lenta reacción ante cambios o fallos en la topología.
- Reactivos: buscan las rutas cuando se necesitan, mediante la "inundación" (*flooding*) de la red de paquetes de petición de ruta. Las desventajas de los protocolos reactivos son el alto tiempo de latencia hasta que la ruta es descubierta y la posibilidad de saturación de la red debida a la técnica de *flooding*.
- Híbridos: Combinan elementos proactivos y reactivos.

Si nos atenemos al nivel de funcionamiento en el modelo OSI, podemos

separar los protocolos de enrutamiento en:

- Nivel 2 (capa de enlace de datos): en el modelo de red OSI, esta capa es la encargada del direccionamiento físico, la detección y corrección de errores, el control de flujo y el acceso al medio. Los protocolos de enrutamiento que trabajan en este nivel son transparentes para la capa superior (capa de red, encargada del direccionamiento IP), manteniendo la pila TCP/IP existente inalterada y evitando las complejas tareas de configuración y administración de IP's. Proporcionan un acceso más rápido y completo a la información de la capa física y MAC, mejorando la eficiencia de las decisiones de encaminamiento y reduciendo el tiempo de reacción ante problemas inesperados. Además, no están sujetos a esquemas de direccionamiento concretos de la capa 3 (IPv4 frente a IPv6, por ejemplo). Por otro lado, el enrutamiento en esta capa no puede aprovechar la información sobre la arquitectura de la red que proporciona el direccionamiento IP, ya que las direcciones MAC son "planas" (sin estructura jerárquica). Por lo tanto, la escalabilidad de una red que utiliza un esquema de este tipo es reducida y no se pueden enrutar paquetes hacia otras redes, siendo utilizados principalmente en redes de tamaño pequeño o medio.

- Nivel 3 (capa de red): la capa de red es la encargada del direccionamiento lógico y el encaminamiento de paquetes. En resumen, es responsable de proporcionar los medios necesarios para transmitir secuencias de datos de longitud variable desde un origen a un destino a través de una o más redes, así como de asegurar la calidad de servicio (*Quality of Service*, QoS). La ventaja del enrutamiento a este nivel es la posibilidad de segmentar las redes (mediante la división en subredes basada en la jerarquía del direccionamiento IP), permitiendo una mejor escalabilidad, y la habilidad para interconectar diferentes redes. Además permite una comunicación más segura (mediante listas de control de acceso, por ejemplo).

Una última posible clasificación sería la siguiente, dependiendo de la información que cada nodo tiene sobre la topología de la red:

- Conocimiento total: cada nodo mantiene información sobre la topología de la red entera.

- Conocimiento parcial: cada nodo sólo tiene información parcial sobre

la topología. Generalmente, el nodo posee información sobre los enlaces con sus nodos vecinos.

## 1.4. Orígenes e historia de las redes ad-hoc y mesh.

### 1.4.1. 1970-1980: Primeras redes ad-hoc.

Para hablar de los orígenes de las redes mesh debemos remontarnos a las primeras redes inalámbricas ad-hoc. Las primeras redes ad-hoc fueron las redes *packet radio* de los años 70. Se trata de redes conmutadas para el intercambio de datos por radiofrecuencia. Una de las primeras de estas redes fue ALOHAnet, desarrollada por el profesor Norman Abramson, de la Universidad de Hawai. Operaba en UHF, a 9600 baudios. De esta red se derivó el protocolo ALOHA (*Areal Locations of Hazardous Atmospheres*) y otros como CSMA (*Carrier Sense Multiple Access*), del que Ethernet fue la primera implementación.

Podemos considerar ALOHAnet como una de las primeras redes de ordenadores de la historia. Como ya se ha dicho, fue desarrollada en la Universidad de Hawai, con financiación de la agencia DARPA (*Defense Advanced Research Projects Agency*). La novedad que introducía ALOHAnet con respecto a otras redes existentes en la época, como ARPANET, era su carácter inalámbrico. La intención de Abramson era conectar las distintas islas que forman el archipiélago hawaiano, por lo que no se podía utilizar tecnología cableada. Utilizaba el sistema *packet radio*. Debido a este hecho, a que los nodos usaban un medio compartido para comunicarse, la misma frecuencia de radio, se necesitaba algún tipo de mecanismo que regulase el acceso de los distintos nodos al medio.

Para resolver este problema, ALOHAnet introdujo un nuevo protocolo: el Acceso Múltiple por Detección de Portadora (CSMA). Básicamente, se trata de que cada nodo, antes de emitir, "escucha" el canal. Si detecta que nadie está usando el medio, envía su información. Normalmente esto significaría que el primer nodo que empiece a transmitir tendría la posesión del medio por tanto tiempo como quisiera, lo que supone que los otros nodos no podrían tomar parte en la comunicación. Para evitar este problema, ALOHAnet hizo que los nodos partieran sus mensajes en pequeños paquetes, y que los enviasen

de uno en uno y dejando huecos entre ellos. Esto permitía a los otros nodos enviar sus paquetes entre medias, por lo que todo el mundo podía compartir el medio al mismo tiempo. Si dos nodos intentan emitir justo en el mismo momento, tendremos de nuevo un problema de colisión de paquetes. Para evitar esto, ALOHAnet introdujo la siguiente solución. Después de enviar un paquete, el nodo escucha para ver si dicho paquete le es devuelto por un *hub* (repetidor, reenvía la información a todos sus nodos vecinos) central. Si lo recibe, es que ha tenido éxito y sigue con el envío del siguiente paquete. En caso contrario, el nodo espera un tiempo aleatorio e intenta volver a transmitir el paquete. Como cada nodo esperaría un tiempo aleatorio, alguno debería ser el primero en reintentarlo, y el resto de nodos podrían ver que el canal está en uso al intentar emitir. En la mayoría de los casos, esto serviría para evitar las colisiones.

El problema es que, en una situación de saturación de la red, el número de colisiones puede crecer de forma que se llegue a un colapso por congestión. Los cálculos demuestran que el uso máximo del canal está en el 18 %. Para intentar atenuar este problema, se introdujo una nueva versión del protocolo: ALOHA ranurado (*Slotted* ALOHA, en inglés). En esta versión, el tiempo se divide en ranuras o *time slots*, cuya duración es igual al tiempo necesario para transmitir un paquete. Los nodos solo pueden emitir al comienzo de un slot. Existe un reloj central que envía la señal de comienzo de un slot a todos los nodos. Si dos o más nodos pretenden emitir en el mismo slot, se produce una colisión. Cada nodo esperará un tiempo aleatorio y lo volverá a intentar. De esta forma se reduce el número de colisiones y aumenta el uso máximo del canal hasta un 36 %.

En 1977, DARPA crea la red *packet radio* PRnet en la bahía de San Francisco. Junto a la empresa SRI, llevaron a cabo una serie de pruebas en las que se consiguieron interconectar y enrutar paquetes entre diferentes redes como ARPANET, PRnet y SATNET.

Prnet fue diseñado con los siguientes objetivos:

- Transparencia: la forma de operar de la red debe ser transparente para los usuarios.
- Conectividad: se debe proporcionar una conectividad lógica total entre todos los nodos.
- Movilidad: se debe soportar nodos móviles.

- Conectividad entre redes: se deben ofrecer funcionalidades de encaminamiento y pasarela hacia redes de otro tipo.

- Coexistencia: Prnet debe poder coexistir junto a otros usuarios de una determinada frecuencia.

- *Throughput* y retardo bajo: paquetes de tamaño variable, latencia de 0.1s, zona de cobertura de 100 millas cuadradas, 100-400 kb/s *throughput* (cantidad de información real transmitida por unidad de tiempo).

- Despliegue rápido: la red debe ser capaz de autoorganizarse una vez desplegada.

- Encaminamiento: *broadcast* y punto a punto.

- Direccionamiento: soporte para distintos grupos *broadcast*.

- Seguridad: resistente a suplantación de identidad, *jamming* (intento de romper la conectividad de la red eliminando enlaces clave), detección y localización de nodos.

Para implementar el enrutamiento de paquetes, PRnet utiliza un tipo de encaminamiento basado en vector distancia. Cada nodo emite un paquete broadcast de actualización de rutas (llamado PROP, *Packet Radio Organization Packet*) cada 7.5 segundos. Estos paquetes contienen las direcciones del emisor y el receptor, el número de saltos realizado hasta el momento, y el número de saltos que faltan hasta llegar al destino. Los nodos que reciben un PROP actualizan su tabla de enrutamiento, si es necesario.

Estas tecnologías, inicialmente desarrolladas por organismos estatales y militares, pronto empezaron a interesar a grupos de aficionados a las comunicaciones por radio. En 1978, Robert Rouleau, un radioaficionado, y la *Western Quebec VHF/UHF Amateur Radio Club*, empezaron a realizar pruebas experimentales de transmisión de datos en código ASCII sobre frecuencias VHF con equipos caseros, en Montreal, Canadá. En EE.UU. pronto nacería la red AMPRNet. Dicha red todavía existe en la actualidad, implementa el protocolo TCP/IP y está mantenida por una comunidad de radioaficionados.

Casi al mismo tiempo empezaron a desarrollarse sistemas comerciales de *packet radio* como DCS o Mobitex.

### 1.4.2. 1980-2000: Redes MANET y difusión de 802.11

El siguiente paso lógico en el desarrollo de las redes ad-hoc era conseguir que los nodos tuvieran la capacidad de moverse, aparecer y desaparecer. Este tipo de red se denomina MANET (*Mobile Ad-hoc Network*). Nuevamente nos encontramos, en el origen de esta tecnología, con DARPA y la investigación militar. Esta agencia puso en marcha, en 1983, el proyecto SURAN (*Survivable Radio Network*) cuya finalidad era desarrollar una red ad-hoc móvil de bajo coste y que pudiese implementar protocolos *packet* radio más complejos que PRNET. En los años 90 se iniciaron distintos proyectos en el ámbito militar y de defensa, como GloMo (*Global Mobile*) y NTDR (*Near Term Digital Radio*).

En 1994 aparece la primera versión del protocolo de enrutamiento DSR para redes ad-hoc multisalto, del que luego se derivarían otros como AODV.

Es a finales de los 90, con la amplia difusión de Internet y de los dispositivos portátiles (móviles, PDA's, *laptops*), cuando las redes inalámbricas empezaron a ser una gran alternativa a las redes cableadas para aplicaciones civiles y comerciales. En 1997, el IEEE aprobó el estándar 802.11 que define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en una red local inalámbrica o WLAN. Aparecieron también otros estándares como Bluetooth e HIPERLAN.

En 1998 se publica la primera versión de AODV, protocolo de enrutamiento para redes ad-hoc móviles. Este protocolo es uno de los más usados hoy en día para este tipo de redes y ha servido de inspiración a muchos otros.

En 1999 se aprueba la revisión 802.11b de la norma original, la 802.11. Esta revisión aumentaba la tasa de transferencia hasta los 11 Mbit/s. Ese mismo año nace la asociación WECA (*Wireless Ethernet Compatibility Alliance*), creada por Nokia y Symbol Technologies. Tenía como finalidad crear una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos. Esta asociación pasó a denominarse Wi-Fi Alliance en 2003. De esta forma, en abril de 2000 WECA certifica la interoperabilidad de equipos según la norma IEEE 802.11b, bajo la marca Wi-Fi.

802.11-1997 fue el primer estándar aprobado, aunque no fue hasta el año 1999, con la aprobación de la revisión 802.11b, cuando la tecnología inalámbrica empezó a tener una amplia difusión entre usuarios finales.

### 1.4.3. 2000-2011: amplia adaptación de 802.11 y primeras redes inalámbricas comunitarias.

En la primera década del siglo XXI la tecnología 802.11 llega a los hogares con conexión a Internet, mediante routers y puntos de acceso inalámbricos que permiten conectar diferentes dispositivos entre sí, y a Internet, sin cables, dentro de una casa, oficina, colegio, etc. Las interfaces de red inalámbricas y los routers y puntos de acceso son cada vez más baratos.

En esta década se aprueban los estándares 802.11g (2003) y 802.11n (2009), los más utilizados en la actualidad para redes WLAN. A diferencia de las otras versiones, 802.11n puede trabajar en dos bandas de frecuencias: 2,4 GHz (la que emplean 802.11b y 802.11g) y 5 GHz (la que usa 802.11a). Gracias a ello, 802.11n es compatible con dispositivos basados en todas las ediciones anteriores de Wi-Fi. Además, es útil que trabaje en la banda de 5 GHz, ya que está menos congestionada y en 802.11n permite alcanzar un mayor rendimiento. El estándar 802.11n fue ratificado por la organización IEEE el 11 de septiembre de 2009 con una velocidad de 600 Mbps en capa física.

En el campo de las redes mesh aparecen nuevos protocolos de enrutamiento como OLSR (2003), B.A.T.M.A.N. (2005) y Babel (2007). Además, el IEEE empieza a trabajar en su estándar para redes mesh, 802.11s. En 2004 se crea el 802.11s Task Group.

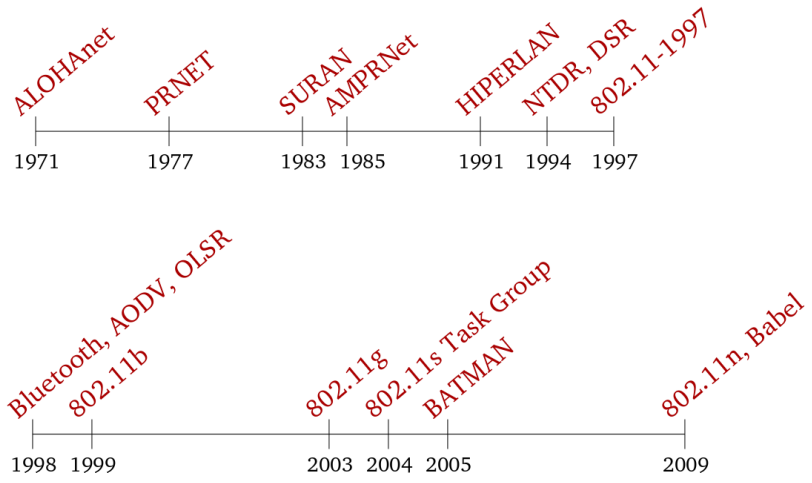


Figura 1.6: Línea temporal de la evolución de protocolos e implementaciones para redes ad-hoc y mesh (la línea superior e inferior están a distinta escala).

De forma paralela, grupos de usuarios con conocimientos informáticos

y de administración de redes empiezan a darse cuenta del potencial de esta nueva tecnología. De forma autogestionada y utilizando tecnología 802.11 se empiezan a crear las primeras redes inalámbricas comunitarias. Además de la tecnología inalámbrica, cada vez más barata, el otro factor fundamental para el desarrollo de este tipo de comunidades fue el software libre. En esa época la comunidad del software libre era ya muy potente, y aplicaciones como Apache ya copaban ciertos campos de la informática y de las redes de ordenadores. Además existían implementaciones libres para la gestión y el enrutamiento de redes ad-hoc, que sirvieron de base a las redes inalámbricas comunitarias para desarrollar su propio software. Estas comunidades empiezan a organizarse cada vez mejor, ofreciendo medios para registrar nuevos nodos, soporte técnico para nuevos usuarios, foros, listas de correo, etc. Comienzan a tener más visibilidad en las áreas donde operan.

Actualmente son numerosos y diversos los proyectos de redes mesh existentes. Muchos son proyectos experimentales asociados a universidades, otros son promovidos por comunidades de usuarios y también existen empresas que venden y despliegan redes de este tipo.

# Capítulo 2

# Estado del arte

En este capítulo del proyecto vamos a hacer un repaso teórico a los protocolos más importantes utilizados en redes mesh, centrándonos, sobre todo, en las técnicas de enrutamiento. Destacaremos sus características principales, su funcionamiento y los diferentes métodos utilizados para resolver las dificultades propias de este tipo de redes. El documento no pretender ser una referencia técnica a la que acudir para cada uno de los protocolos, sino una breve descripción de sus principios básicos y su funcionamiento.

El capítulo está dividido en dos partes:

- Protocolos: describiremos los protocolos de enrutamiento más utilizados en este tipo de redes y para los que existen implementaciones con licencia libre. Los protocolos que estudiaremos y que después serán utilizados en el entorno real de prueba serán: AODV, OLSR, Babel, B.A.T.M.A.N., IEEE 802.11s y DSR. Como ya hemos explicado antes, con DSR y OLSR nos limitaremos a describirlos en el estado del arte.
- Redes mesh en funcionamiento. Enumeración y descripción de varias redes mesh que están funcionando hoy en día alrededor del mundo, fijándonos, sobre todo, en aquellas que surgen de iniciativas comunitarias de usuarios y que usan software libre.

Seguidamente pasamos a explicar los distintos protocolos. El esquema descriptivo para cada uno de ellos es el siguiente:

- Características principales:

  - Desarrollador(es): autores de la especificación del protocolo.
  - Fecha de la primera versión: fecha en la que aparece por primera vez un documento, ya sea un RFC, un artículo, un borrador u otro tipo de texto, describiendo el protocolo.
  - Versión actual: estado actual del protocolo, determinado por la versión actual del RFC o la revisión del documento original.
  - Tipo: tipo de protocolo en cuanto a la forma de descubrir y actualizar las rutas. Como se explica en 1.3.2 los tipos pueden ser: proactivo, reactivo e híbrido.
  - Nivel OSI: nivel o capa del modelo OSI en el que actúa. Puede actuar en el nivel 2 (enlace de datos) o 3 (red).
  - Conocimiento de la topología: define el grado de conocimiento que alcanza un nodo en la red, pudiendo llegar a ser completa o simplemente parcial, limitándose a saber cómo llegar a los nodos con los que le es imprescindible comunicarse.

- Introducción: breve descripción del origen del protocolo y sus propiedades más significativas. También se nombrarán algunas implementaciones con una breve explicación.

- Funcionamiento: descripción del protocolo a nivel funcional, en la que se explican los mecanismos de descubrimiento y actualización de rutas, reacción ante fallos o cambios en la topología y mensajes de control.

## 2.1. AODV

- Desarrolladores: C. Perkins, E. Belding-Royer y S. Das
- Fecha de la primera versión: Noviembre de 1997 [19]
- Versión actual: RFC-3561, Julio de 2003 [10]
- Tipo: Reactivo
- Nivel OSI: Nivel de Red
- Conocimiento de la topología: Parcial, sólo vecinos.

### 2.1.1. Introducción

AODV es un protocolo de enrutamiento utilizado en redes ad-hoc, como pueden ser las redes mesh. Ha sido desarrollado por Nokia, la Universidad de California y la Universidad de Cincinnati, por C. Perkins, E. Belding-Royer y S. Das.

Es un protocolo reactivo, es decir, sólo se intenta descubrir una ruta cuando un nodo de la red la requiere. Esto provoca una gran latencia en la primera comunicación mientras se descubre un camino. Sin embargo, la principal ventaja de los protocolos reactivos es el menor consumo de ancho de banda y de CPU, ya que no envían paquetes a no ser que sea estrictamente necesario.

Para trazar las rutas, AODV utiliza un vector distancia representado por la dirección del nodo destino y el número de saltos. Además de las características generales de los protocolos reactivos, AODV tiene unas peculiaridades que lo distinguen [15]:

- Ningún nodo tiene conocimiento total de la topología de la red. Un nodo sólo tiene conocimiento de los nodos con los que necesita comunicarse. De los nodos conoce a cuántos saltos de distancia se encuentran y hacia dónde debe enviar el primer salto para llegar a ellos. AODV encamina salto a salto, es decir, no se sabe la ruta que van a seguir los paquetes cuando se generan, sino que cada nodo, cuando recibe un paquete, decide cuál es el próximo salto para que la información llegue a su destino final.
- Para poder distinguir si la información recibida es más moderna que la que se tiene actualmente en caché, se emplean "horas lógicas". Una hora lógica es un identificador del nodo, la IP en este caso, y un número de secuencia asociado a éste, que es incrementando con cada información que se envía. Toda información enviada o guardada por un nodo lleva consigo una hora lógica para poder comparar cúal de las dos informaciones es más actual.
- Toda información acaba caducando, es decir, cada conocimiento que se tiene de la red se desechará si no es renovado antes de un tiempo especificado.

El RFC-3561 para AODV [10] no especifica nada sobre IPv6, y se centra absolutamente en IPv4. Sin embargo existe un borrador que complementa al

RFC para hacer funcionar AODV con IPv6 escrito por los mismos autores [1]. Actualmente se encuentra en fase borrador.

Existe varias implementaciones de AODV para sistema libres, tanto para la versión de IPv4 como para la versión de IPv6. La mayoría de ellos no contienen mucha información y otros parecen abandonados. Podemos citar algunos como:

- AODV-UU:[2] Creado desde la Universidad de Uppsala. Funciona con un módulo del kernel que sólo es compatible con algunas versiones específicas de éste, tanto de la rama 2.4 como de la 2.6. En versiones actuales del kernel no funciona ya que el módulo se ha quedado desactualizado respecto a las nuevas versiones. También tiene un demonio que es el que realiza la mayor parte de la lógica de control de AODV. Cumple con el estándar definido en el RFC-3561 de AODV. Existe, además, una versión para IPv6 y parches para poder hacer uso de *multicast*.

- Meshias: Funciona íntegramente en espacio de usuario. Tiene la ventaja de que seguirá funcionando con nuevos kernel y no hay que mantener un módulo actualizado. Como contrapartida, en espacio de usuario la velocidad del demonio no será todo lo óptima que podría ser si estuviera completamente integrado en el núcleo.

- AODV-UCSB:[3] Antigua implementación para Linux 2.4. Fue realizada por la Universidad de California, Santa Barbara. Implementa la versión IPv6 del protocolo. Llegó a ser funcional pero ahora solamente recomiendan su uso para proyectos de investigación y en ningún caso para producción. El proyecto ha dejado de actualizarse.

- Kernel AODV:[4] Fue desarrollado por el *National Institute of Standards and Technology* contratado por el Gobierno Federal de Estados Unidos. Funciona solamente en la rama 2.4 del kernel de Linux. Su última versión data de 2004. Ya no es mantenido pero su código es libre de ser redistribuido y modificado. Cumple con el estándar definido en el RFC-3561 de AODV.

[1] `http://www.cs.ucsb.edu/~ebelding/txt/aodv6.txt`
[2] `http://sourceforge.net/projects/aodvuu/`
[3] `http://moment.cs.ucsb.edu/AODV/aodv.html`
[4] `http://www.antd.nist.gov/wctg/aodv_kernel/`

### 2.1.2. Funcionamiento

Según el RFC-3561 [10] el protocolo AODV realiza dos tareas principales, la búsqueda y el de mantenimiento de rutas.

#### Búsqueda de rutas

La búsqueda de rutas se ejecuta cuando un nodo quiere enviar información a otro pero éste no se encuentra en su tabla de rutas. El nodo fuente, que así llamaremos al que quiere enviar la información, envía un paquete de tipo *Route Request* (RREQ) en *broadcast*. El envío en *broadcast* significa que es recibido por todos los nodos que estén en su rango de transmisión. Un RREQ lleva la siguiente información:

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de AODV los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.
- *Opciones:* Permiten modificar el comportamiento natural de la petición:
  - *Multicast:* Existen un par de opciones reservadas especialmente para multicast.
  - *RREP gratuito:* Si un nodo intermedio responde a la petición, enviará también otro RREP al nodo destino de la petición para que sea notificado.
  - *Sólo destino:* Sólo podrá responder a la petición el nodo destino y por tanto los nodos intermedios siempre retransmitirán la petición.
  - *Número de secuencia desconocido:* Se suele activar cuando no hemos recibido información del nodo destino y no podemos enviar su número de secuencia.
- *Identificador RREQ:* Un número de secuencia que junto con el nodo origen identifica la petición de manera única.
- *IP Destino:* La IP del nodo del que queremos la ruta.
- *Número de secuencia de destino:* El último número de secuencia recibido en el pasado por el nodo origen del nodo destino. Este atributo junto con la IP de destino forman la hora lógica destino.

- *IP Origen:* La dirección del nodo emisor de la petición.
- *Número de secuencia del origen:* El número de secuencia actual para ser usado en la tabla de rutas de los nodos que reciben la petición. Este atributo junto con la IP de origen forman la hora lógica origen.
- *Contador de saltos:* Utilizado para ir midiendo el número de saltos que está realizando la petición.
- *TTL:* Atributo que contiene cualquier paquete UDP y que mide cuántos saltos puede dar un mensaje a través de los nodos. Cobra especial importancia ya que limita la profundidad a la que llegará la petición.

Cuando un nodo recibe un RREQ comprueba si ha recibido anteriormente una petición con el mismo identificador y origen. Si es así descarta el mensaje silenciosamente. Si por el contrario no lo había recibido antes pueden ocurrir tres cosas:

1. *Que el nodo sea el destino de esa petición.* Entonces procederá a responder con un paquete del tipo RREP. Veremos más adelante qué información contiene y cómo se transmite.
2. *Que tenga al destino en su tabla de rutas.* Esto puede ocurrir porque él ya haya establecido una comunicación previa. Antes de contestar comprueba que la hora lógica de destino del paquete sea igual o menor que la que se ha guardado en la tabla de rutas, ya que la información podría haber quedado obsoleta. En este caso el nodo deberá responder con un paquete RREP. Veremos más adelante que información contiene y como se transmite.
3. *Que no tenga conocimiento sobre el destino.* Cuando esto ocurre, el nodo reenvía el RREQ con el objetivo de llegar al máximo de nodos posible y encontrar al nodo destino. Esto ocurrirá siempre y cuando el TTL no haya llegado a cero. Igualmente aumentará el contador de saltos en uno. Además se mantendrá la información de la petición temporalmente para que se puede desechar si volviese a llegar. El nodo que envió la petición, que no tiene porque ser el origen sino sólo el salto anterior, también es guardado con el objetivo de poder reconstruir la ruta hacia atrás como veremos más adelante. Toda esta información es guardada durante un tiempo por el nodo. Pasado este tiempo la información se desechará.

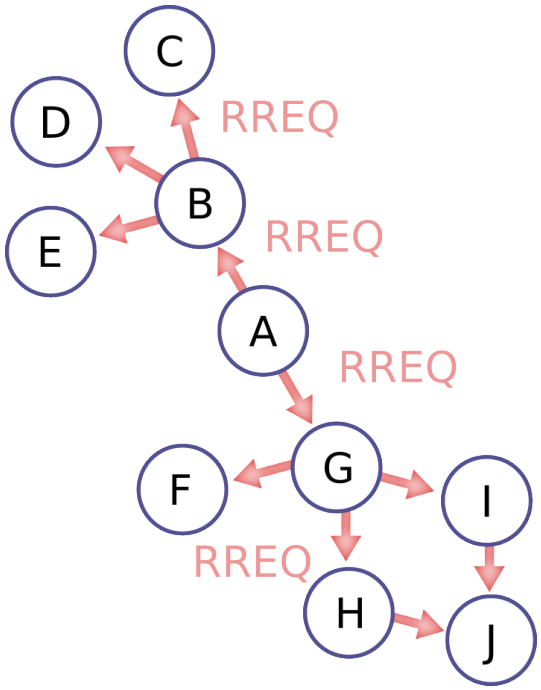


Figura 2.1: La expansión de una petición de ruta del nodo A

En la figura 2.1 podemos observar como el nodo A realiza una petición con un paquete RREQ. El paquete se extiende por toda la red buscando su destino siendo reenviado por los nodos intermedios. En la figura observamos como B, G, H e I reenvían el paquete que les ha llegado de A porque ni son el destino ni lo conocen. Como ya hemos explicado, puede ocurrir que un nodo reciba varios paquetes RREQ para una misma petición, en este caso el nodo J la recibe dos veces a través de I y H. Descartará silenciosamente la petición que llegue en segundo lugar.

Un nodo que contesta lo hace a través de un paquete denominado *Route Reply* (RREP) que contiene la siguiente información:

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de AODV los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.
- *IP destino:* La dirección del nodo destino.
- *Número de secuencia destino:* El número de secuencia asociado a la ruta.

- *IP Origen:* La dirección IP del nodo que originó la petición RREQ y al que va dirigido esta respuesta.
- *Contador de saltos:* El número de saltos desde la dirección IP de origen hasta la dirección IP de destino.
- *Tiempo de vida:* El tiempo en milisegundos en el que cada nodo que reciba el RREP considerará la ruta como válida.

El RREP recorre el mismo camino, pero en sentido inverso, que ha ido recorriendo el RREQ hasta llegar al nodo que ha contestado la petición. Sin embargo, un paquete RREP no se envía por broadcast, sino que cada nodo lo recibe de uno de los nodos a los que previamente envió el RREQ y lo reenvía al nodo del cual recibió el RREQ siguiendo un camino completamente unidireccional hasta llegar al nodo fuente. Por esta razón todos los nodos guardan temporalmente el origen de los paquetes RREQ que han recibido como ya habíamos comentado anteriormente.

Cada vez que un nodo recibe un RREP, sin importar si es un simple intermediario o no, se añade o actualiza la entrada en la tabla de rutas, incluyendo su hora lógica. Una singularidad de AODV es que una entrada en la tabla de rutas solamente contiene el destino final, o sea el origen del RREP, la distancia y el nodo vecino a través del cual se llega a ese destino. De este modo, un nodo no conoce la ruta completa hasta llegar al destino, sólo sabe a que distancia está en número de saltos y cuál es el nodo vecino que consigue llevar hasta él. La ruta, por tanto, se va construyendo sobre la marcha, de manera que un paquete no sabe en principio qué ruta va a seguir. Cada vez que un nodo recibe un paquete que tiene que reenviar, mira su actual tabla de rutas y lo manda siempre a un nodo vecino sin saber por qué otros nodos pasará esa información.

Otra propiedad de las rutas es que son temporales; se inicializan con un tiempo de caducidad para la nueva entrada. El tiempo por defecto son tres segundos, lo cual se antoja muy corto para la mayoría de los casos, pero este campo es configurable y debe ser adaptado según la topología de la red. Pasado ese tiempo, y si no ha sido actualizada por ningún dato recibido, la ruta será eliminada. Si fuera necesario volverla a utilizar para enviar información habría que reconstruirla con una nueva petición y esperar a la respuesta.

Además un nodo debe guardar, por cada entrada de la tabla de rutas, una lista de precursores que utilizan esta ruta. Cada vez que un nodo reenvía información de un nodo vecino, éste último es añadido a la lista de precursores

de la ruta que se haya utilizado. También se añade a la lista de precursores al nodo que se le envía un RREP, ya que a partir de entonces el nodo nos usará para llegar al nodo destino de la petición, convirtiéndose así en el primer precursor de la ruta. La lista de precursores se usará para el mantenimiento de rutas.

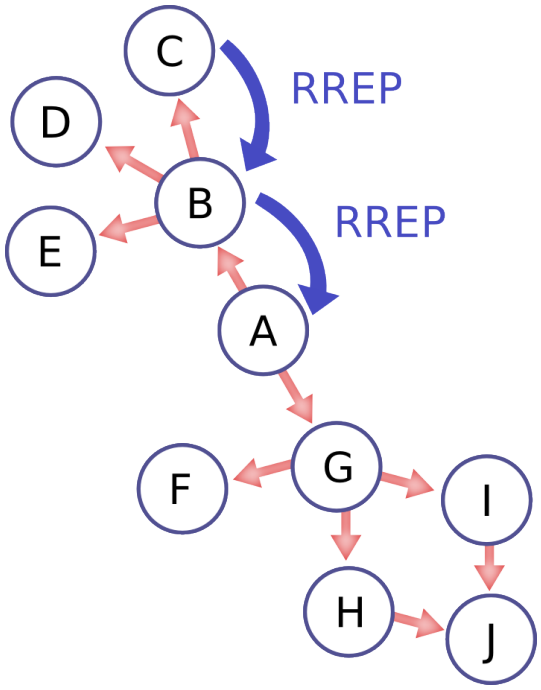


Figura 2.2: Respuesta del nodo C a la petición de ruta de A

Siguiendo el ejemplo anterior, la figura 2.2 muestra como el nodo C recibe una petición de ruta hacia él enviado por A. C contesta con un paquete RREP. Como la petición le llegó a través de B, la respuesta se la envía directamente a él. B al recibir el RREP enviado por C, lo vuelve a enviar pero esta vez con destino A. Es interesante notar como la respuesta, en contraposición con la petición, es enviada de manera unidireccional para intentar saturar la red lo menos posible.

Para evitar desechar rutas válidas constantemente, cada vez que un nodo recibe un paquete, ya sea un RREP, un RREQ, o cualquier paquete de datos que no sea del protocolo AODV, el emisor de esa información es actualizado en la tabla de rutas del nodo receptor reinicializando su tiempo de caducidad y refrescando la hora lógica si procede. Esto provoca que si una ruta se utiliza asiduamente y funciona correctamente no caduque.

**Mantenimiento de rutas**

Los nodos también realizan tareas de mantenimiento de rutas, en caso de que alguna se vuelva inoperativa. Su funcionamiento se basa en el envío de paquete llamados *Route Error* (RERR). Un paquete de este tipo contiene la siguiente información:

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de AODV los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.
- *Número de rutas inválidas:* Un paquete RERR puede contener la dirección de más de un nodo que ha quedado inaccesible. Este campo contendrá el número de destinos que han dejado de ser válidos enviados en este mensaje.
- *Lista de nodos inaccesibles:* Es una lista que contiene los nodos que han dejado de ser accesibles. Su tamaño es indicado por el campo anterior. Cada elemento de esta lista tiene la dirección del nodo y su último número de secuencia asociado guardado por el nodo que ha generado el paquete.
- *TTL:* Atributo que contiene cualquier paquete UDP y que mide cuántos saltos puede dar un mensaje a través de los nodos. El valor para los RERR siempre es de uno.

Existen tres situaciones por las que un nodo generará un paquete de tipo RERR:

1. Si detecta que el enlace se ha roto para el siguiente salto de una ruta mientras se estaban transmitiendo datos. Para este caso el nodo observa en su tabla de rutas todos los nodos destinos que han dejado de ser inaccesibles a causa de la pérdida de conectividad con su vecino y generará un paquete RERR con todos ellos.
2. Si recibe un paquete de datos para un destino que no tiene en su tabla de rutas. A diferencia del anterior caso, aquí sólo habrá un nodo inaccesible en el paquete RERR y es el destino desconocido del paquete de datos recibido.

3. Si recibe un paquete RERR de un vecino que afecte a una o más rutas activas. Entonces el nodo forjará un nuevo RERR que contendrá como nodos inaccesibles aquellos que cumplan tres condiciones:

   *a*) Que estuvieran como inaccesibles en el RERR recibido.

   *b*) Que estuvieran como destino en alguna entrada en la tabla de rutas.

   *c*) Que en la entrada de la tabla de rutas el primer salto fuese el vecino que envío el RERR.

Los nodos que deben recibir los RERR son todos los vecinos que pertenezcan a la lista de precursores de las rutas que tengan como destino al menos a uno de los nodos inalcanzables. Recordemos que la lista de precursores no es más que una lista de los nodos que han ido utilizando esa ruta a través del nodo. Explicado de una manera más simple, se debe enviar el RERR a todos los vecinos que hayan usado alguna vez una ruta que hayamos tenido que eliminar para notificarles que no podrán usarla más. Si una ruta tiene vacía la lista de precursores no haría falta enviar ningún RERR de ese destino.

En el caso de que fuera un único nodo el que necesita recibir el RERR, entonces éste debería ser enviado de manera unidireccional hacía ese vecino. En otro caso, el RERR es enviado a todos los vecinos por *broadcast* y con valor del campo TTL a uno para asegurar que no es reenviado.

También es posible agrupar varios destinos que sean inaccesibles en un sólo paquete RERR y enviarlos a todos los vecinos simultáneamente por *broadcast*. Cada vecino que reciba el mensaje comparará uno por uno buscando si existen nodos afectados en su tabla de rutas. De esta forma, enviando solamente un paquete, nos podemos ahorrar varios mensajes. Cuando el uso del *broadcast* se considere inapropiado el nodo puede separar las direcciones una a una y enviar iterativamente paquetes específicos a los vecinos correspondientes, aunque el coste de ancho de banda sea mayor.

Cuando un nodo recibe un paquete RERR, comprueba cada nodo y número de secuencia; si la información recibida es más actual porque su número de secuencia es mayor que la guardada, la ruta se invalida. Posteriormente procederá a comprobar si vecinos suyos han utilizado la recién invalidada ruta. Si es así procederá a generar el RERR correspondiente y a enviárselo para que queden notificados. Esto seguirá de manera encadenada hasta llegar al último nodo que haya usado en algún momento el camino roto.

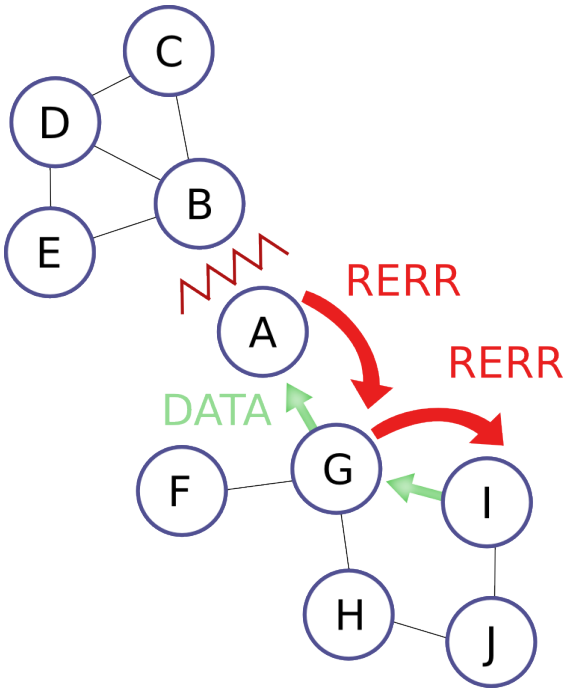


Figura 2.3: Comunicación infructuosa del nodo I hacia C

En esta ocasión, podemos observar en la imagen 2.3 que el nodo I quería enviar información al nodo C. Pero cuando la información llega al nodo intermediario A, éste se encuentra en una de las situaciones en las que se debe producir el envío de un RERR, es decir, percibir que el enlace con el nodo que debe transmitir información se ha perdido, en este caso con B. Entonces el nodo A comprueba la lista de precursores que han utilizado la ruta hacia C y sólo encuentra al nodo G, que es quién estaba realizando la conexión actualmente. Cuando G recibe el RERR y elimina la ruta hacia C, se encuentra en otra situación en la que se debe generar un RERR. El nodo ha recibido un RERR de A sobre la ruta C, de la cual tiene como precursor al nodo I. Como es un solo destino el envío del mensaje no se realiza en broadcast sino de manera unidireccional hacia el nodo I porque el nodo H o F nunca han usado esa ruta antes.

## 2.2. DSR

- Desarrolladores: D. Johnson, Y. Hu y D. Maltz
- Fecha de la primera versión: Marzo de 1998 [20]

- Versión actual: RFC-4728, Febrero de 2007 [8]
- Tipo: Reactivo
- Nivel OSI: Nivel de Red
- Conocimiento de la topología: Parcial, con rutas completas.

### 2.2.1. Introducción

DSR es un protocolo de enrutamiento muy similar a AODV. Ambos son reactivos y mantienen los mismos tipos de paquetes en su funcionamiento. Para realizar el análisis de DSR haremos múltiples referencias al estudio ya realizado en este mismo documento a AODV 2.1 comparando ambos protocolos y viendo sus puntos débiles y fuertes.

Sus características básicas son las mismas [8]:

- Funciona bajo demanda, es decir, sólo se buscarán rutas cuando sea necesario enviar información a un nodo. Esto provoca una alta latencia en las primeras comunicaciones con nodos que no hayan sido descubiertos aún.
- Detecta rápidamente cambios en la arquitectura de la red adaptando las rutas cuando estos se producen.
- Trabaja bien incluso en condiciones donde la movilidad es alta.
- Incorpora un mecanismo para evitar la formación de bucles.

Tiene, sin embargo, otras propiedades que lo caracterizan y diferencian de AODV:

- Permite tener varias rutas para un mismo destino permitiendo así balanceo de carga y mayor robustez.
- Para encaminar un paquete, éste incorpora toda la ruta completa con todos los nodos por los que pasará hasta llegar al destino. Con este mecanismo un nodo puede obligar que sus mensajes pasen forzosamente por determinados nodos, o todo lo contrario, evitar que sus mensajes transiten algún nodo que se quiera sortear. En AODV esto es completamente imposible.

- Su diseño no es escalable, por lo que a partir de un número de nodos las cabeceras crecen demasiado y deja de funcionar correctamente.

La principal ventaja de este tipo de protocolos reactivos es que reducen la carga de la red debida a tráfico de control y actualización de rutas, ya que no es necesario inundar la red periódicamente con estos mensajes, como en los protocolos proactivos.

Algunas implementaciones del protocolo DSR son las siguientes:

- *DSR-UU:*[5] Creado desde la Universidad de Uppsala tiene una arquitectura parecida a AODV-UU 2.1.1. Funciona en Linux y necesita un par de módulos compilados para el kernel e incluso genera una interfaz de red virtual. También se apoya en un demonio a nivel de usuario que es dónde se realizan la mayor parte de las operaciones lógicas de la solución.

- *Monarch Project:*[6] Es una implementación de DSR bastante antigua para FreeBSD 3.3 y 2.2.7. Nunca salió una versión definitiva del mismo pero tenía la mayoría de las características descritas en unos de los borradores de la época. Su uso siempre se recomendó para investigadores de redes y nunca para un sistema en producción.

- *Piconet:*[7] Implementación muy antigua de los primeros borradores de DSR escrito por Alex Song para su tesis doctoral. Funciona bajo versiones del kernel de Linux muy antiguas como 2.4.3.

### 2.2.2. Funcionamiento

DSR es un protocolo bastante extenso (el RFC tiene más del doble de páginas que el de AODV), con muchas funcionalidades opcionales que mejoran la eficiencia del protocolo. Sin embargo, aquí nos centraremos en su funcionamiento básico sin entrar en detalles de características opcionales. Para más información se recomienda la lectura del RFC [8].

[5] `http://dsruu.sourceforge.net/`
[6] `http://www.monarch.cs.rice.edu/dsr-impl.html`
[7] `http://piconet.sourceforge.net/`

Como en AODV, DSR tiene dos tareas principales en su funcionamiento [24], el de búsqueda y el de mantenimiento de rutas. Empezaremos por la primera.

### Búsqueda de rutas

Cuando un nodo fuente tiene un paquete para enviar, lo primero que hace es buscar en su tabla de rutas un camino hacia el destino. Si lo tiene, crea un nuevo paquete añadiendo a la cabecera la ruta completa, en donde se indican los saltos que debe seguir hasta llegar al destino y finalmente lo envía. Aquí nos encontramos la primera diferencia con AODV, en donde si el nodo encontraba en su tabla de rutas el destino, éste enviaba simplemente el paquete al primer vecino sin añadir ni modificar nada más. En DSR se añade toda la ruta primero antes de ser enviado.

El nodo emisor tiene en su tabla de rutas todos los caminos hacia los nodos que ha descubierto previamente. En el caso de que no encontrara ninguna ruta, es cuando se inicia el mecanismo de descubrimiento de rutas para encontrar dinámicamente un nuevo camino hacia el nodo destino.

Lo primero que hace el nodo fuente es enviar un paquete por *broadcast* denominado Route Request (RREQ), el cual es recibido por todos los nodos que se encuentren dentro del rango de transmisión del nodo fuente. El paquete RREQ de DSR suele tener mayor tamaño que el de AODV. Contiene:

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de DSR los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.
- *Tamaño:* Qué tamaño tiene el paquete no incluyendo el discriminador ni esta opción. Hay que recordar que los RREQ van creciendo de tamaño según van saltando de nodo en nodo.
- *Identificador:* Valor incremental generado por el nodo fuente para identificar de manera única una petición.
- *IP destino:* La dirección del nodo objetivo de la petición.
- *Lista de IPs de ruta:* Cada vez que se reenvía la petición el nodo se agrega al final de la lista e incrementa el tamaño en 4.

- *TTL:* Atributo que contiene cualquier paquete UDP y que mide cuántos saltos podrá dar un mensaje. Cobra especial importancia ya que limita la profundidad a la que llegará la petición.

Cuando un nodo recibe un RREQ y comprueba que no es el destino de la ruta, mira si ya ha recibido recientemente otro RREQ con la misma fuente, destino e identificación o si su propia dirección ya aparece en el paquete. En cualquiera de los dos casos el nodo descarta el paquete silenciosamente. Como en AODV, en DSR también es necesario ir recordando las peticiones que ya han sido contestadas para evitar la saturación de la red respondiendo varias veces una misma petición. También se comprueba que no aparezcan en la ruta que se ha ido contruyendo para evitar que se produzcan bucles.

En el momento en el que no se cumple ninguna de las dos condiciones anteriores y por tanto es la primera vez que el nodo recibe la petición, comprueba primero en su tabla de rutas si existe alguna hacia el nodo destino. Si fuera así, el nodo responde con un *Route Reply* (RREP) hacia el origen en lugar de reenviar el RREQ. De este paquete ya hablaremos más adelante. Por otra parte, si el nodo no encuentra ninguna ruta, entonces añade su dirección en el paquete y lo reenvía mediante *broadcast* para intentar llegar al nodo destino o a otro que conozca un trayecto hacia él. Si finalmente el nodo destino recibe un RREQ, procederá a contestar la petición con un RREP al nodo fuente.

Para ilustrar el procedimiento de envío de un RREQ tenemos la figura 2.4, donde podemos observar una secuencia de cómo se transmitiría un RREQ en DSR. El nodo A inicia la petición y envía una petición poniéndose así misma como primer nodo. Cuando los nodos B, E y H reciben la petición y se cercioran de que no la han recibido antes se añaden automáticamente a la lista de paquetes. Una vez hecho esto procederán a enviarlo de nuevo por *broadcast*. Seguramente lleguen esos paquete también a A, que los descartará silenciosamente. Posteriormente los nodos C y F recibirán la petición, junto con E, que al ser la segunda vez que la recibe, esta vez a través de H, la descartará. Tanto C como F vuelven a enviar la petición agregándose a la lista que se va generando. Finalmente llegará a D a quién le corresponderá contestar ya que ninguno de los nodos anteriores conocían una ruta hasta él.

Habíamos dejado la explicación de como un nodo respondía a la petición. Un paquete RREP es muy sencillo ya que sólo contiene:

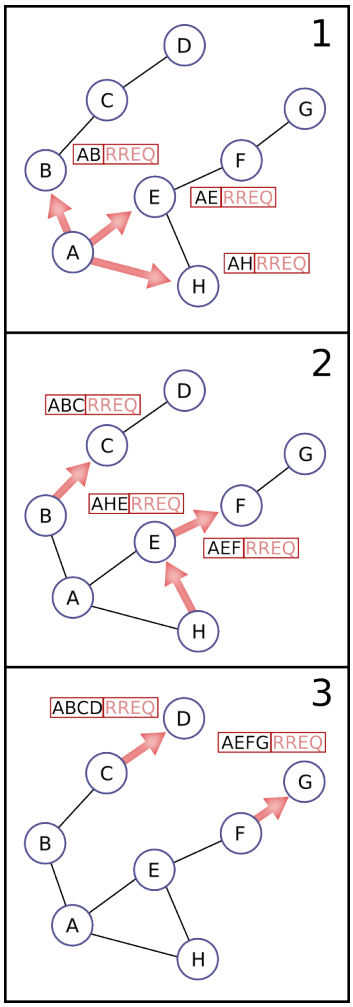


Figura 2.4: Búsqueda de ruta del nodo A hacia el D

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de DSR los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.
- *Tamaño:* Qué tamaño tiene el paquete no incluyendo el discriminador ni este campo. La ruta con la respuesta puede tener un tamaño variable.
- *Lista de IPs de ruta:* La lista invertida de las IP que se han ido acumulando durante la petición.

Existían dos situaciones en las que un nodo podía contestar con un RREP. Cuando el nodo recibe una petición de ruta en la que él es el objetivo,

enviará un RREP que contiene una copia de las direcciones acumuladas en el RREQ recibido pero de manera invertida. En el caso de que la petición llegue a un nodo que conozca una ruta hasta el nodo destino, entonces éste procederá a responder con una ruta en la que los nodos son aquellos que conforman su ruta hacia el destino, añadiendo al final los nodos por los que ha ido pasando el RREQ recibido. Este último proceso no es más que unir una parte del camino que se ha descubierto con el RREQ, a otra parte ya conocida por el nodo que responde, que es su ruta hacia el destino.

Además a diferencia del RREQ y como ocurre en AODV, los RREP no se transmiten en *broadcast*, sino en *unicast*, saltando de nodo a nodo siguiendo la ruta marcada por el propio paquete hasta llegar al nodo fuente.

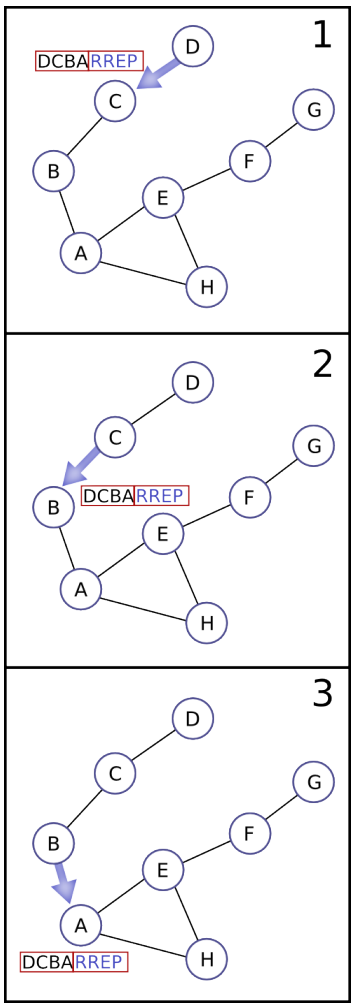


Figura 2.5: Respuesta de la petición de ruta de A

Siguiendo el ejemplo anterior 2.4, podemos observar en esta nueva figura 2.5, como el nodo destino D responde a la petición iniciada por A. La cabecera es la misma que la que llegó al nodo D pero ordenada de manera inversa. Se puede observar cómo el mensaje se va transmitiendo de manera unidireccional hasta llegar al nodo emisor, en contraposición a los RREQ que se extienden por toda la red.

Una vez el nodo origen recibe el RREP, éste guarda dicha ruta en su tabla de rutas. Esta ruta será incluida en la cabecera de cada paquete que el nodo envíe, de esta manera todos los nodos que van recibiendo el paquete sabrán cual es su próximo salto. Esto conlleva enviar más datos para transmitir la misma información y por tanto una disminución del ancho de banda que gastaremos en enviar una y otra vez la ruta en los mensajes.

Este volumen de datos extra puede ir incrementándose aún más si se utiliza IPv6, en vez de IPv4, ya que las direcciones de IPv6 ocupan 128 bits y las de IPv4 solamente 32. Hay que percatarse también de que a medida que las rutas se hacen más largas por el crecimiento de la red, el tamaño de la cabecera también lo hará. Uno de los principales problemas de DSR es su falta de escalabilidad por estos problemas, obligando a las redes a segmentarse para un mejor funcionamiento.

Esta es una de las ventajas de AODV respecto DSR. Y es que AODV no necesita modificar la naturaleza de los paquetes ajenos al protocolo. Un paquete sería exactamente igual enviado desde una red wifi convencional que una mesh dirigida por AODV, cosa que no ocurre con DSR.

Cuando los nodos construyen rutas de encaminamiento válidas, éstas son guardadas aproximadamente durante unos 300 segundos. Si en algún momento se recibieran datos desde un nodo, el contador para esa ruta se reinicializaría de nuevo a los 300 segundos.

Por último, vemos en la figura 2.6 como el nodo A, que ya recibió la ruta a través del RREP, puede enviar la información al nodo D agregando la ruta completa en la cabecera y saltando de nodo en nodo hasta el destino.

### Mantenimiento de rutas

Cuando se envía un paquete a través de una ruta, cada nodo es responsable de confirmar que el paquete se ha recibido en el siguiente nodo. Este reconocimiento se suele hacer a nivel de enlace. Después de haber retrans-

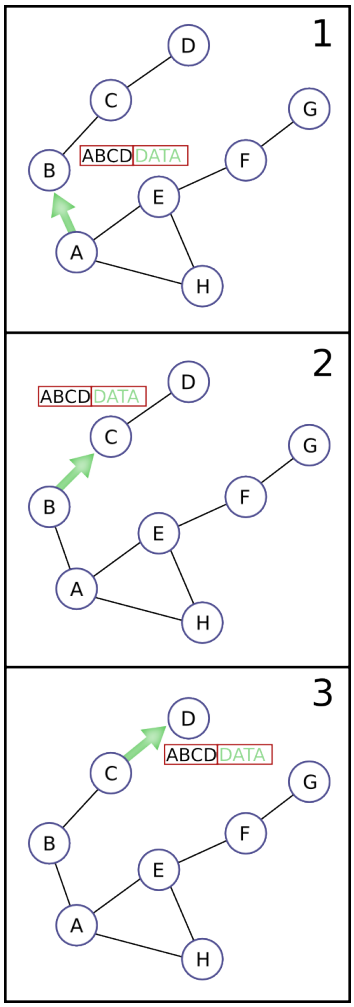


Figura 2.6: Envío de datos desde el nodo A al D

mitido un paquete un número máximo de veces, si el paquete no ha podido ser recibido por el siguiente salto, entonces el nodo considera que el enlace está roto. En ese caso, el nodo debe eliminar ese enlace de su caché y enviar un *Route Error* (RERR) a cada nodo que haya enviado un paquete con una ruta que use el enlace roto.

Un paquete RERR está formado por los siguientes atributos:

- *Discriminador:* Campo que se utiliza para identificar el tipo de paquete que se ha recibido. En el caso de DSR los tipos de paquete que podemos encontrar son RREQ, RREP y RERR.

- *Tamaño:* Qué tamaño tiene el paquete no incluyendo el discriminador ni este campo.
- *Tipo de error:* Existen tres tipos de errores:
  1. *NODE_UNREACHABLE:* Este es el tipo de error que nos interesa y el que hemos estado explicando previamente. Un RERR enviado con este tipo de error es lanzado cuando un nodo detecta que no puede comunicarse con otro y necesita enviarle algún paquete.
  2. *OPTION_NOT_SUPORTED:* El protocolo DSR tiene varias funcionalidades que son opcionales y no todas las implementaciones tienen porque proveerlas. Cuando un nodo recibe alguna opción extra que no tiene implementada devuelve un RERR con este tipo de error.
  3. *FLOW_STATE_NOT_SUPPORTED:* Existe una funcionalidad adicional de DSR que es conocer el estado de flujo de una ruta. Es un poco especial porque necesita una cabecera algo distinta para implementarse. Cuando los nodos reciben un paquete de este tipo y no tienen implementada esa opción deben responder con un RERR con este tipo de error.
- *Dirección origen del error:* La dirección del nodo que origina el error de la ruta. Por ejemplo, en el caso de nodo inalcanzable, este campo tendría el valor del nodo con el que se perdió la comunicación.
- *Dirección de destino:* La dirección del nodo al que este paquete debe ser entregado. Por ejemplo, cuando el tipo de error es de nodo inalcanzable, este campo se establecerá con la dirección del nodo fuente que originó la comunicación infructuosa de manera que se le notifiqué que ya no podrá usar más esa ruta.
- *Información específica del error:* Para la situación en el que el error sea nodo inalcanzable, este campo contendrá el nodo con el que se perdió la conexión.

Cuando un nodo recibe un paquete RERR de tipo nodo inalcanzable comprueba si tiene alguna ruta que contenga el salto que se ha roto. En caso de que fuera así, procedería a eliminar a todas ellas. Aquí podemos observar otra diferencia entre AODV y DSR. En ambos protocolos el error es notificado hasta el nodo fuente, pero en AODV se invalida completamente la ruta, mientras que en DSR se envía simplemente el salto que ha dejado de

funcionar, y de esta forma, aprovechando el conocimiento de rutas completas, descartar otras rutas que también han quedado invalidadas.

Posteriormente comprobaría si la dirección de destino del paquete es él mismo. En el supuesto de que él no fuera el destino final del paquete, éste procedería a reenviar el paquete para que siguiera su ruta.

Cuando el RERR llega finalmente al nodo fuente, además de eliminar todas las rutas que han quedado inutilizables empieza lo que se denomina "salvamento del paquete". Se buscan rutas alternativas para los paquetes que no se han podido transmitir a causa del fallo de la ruta. Si se encuentra alguna se procederá a reenviar los mensajes con la ruta alternativa que no pudieron llegar anteriormente. De esta forma se intenta realizar de nuevo la comunicación de manera absolutamente transparente para el usuario, aunque se hayan producido errores durante los primeros intentos.

En el caso que no tenga ninguna otra ruta en su caché, el nodo origen tiene que iniciar un nuevo descubrimento de ruta para encontrar un nuevo camino hacia el nodo destino. En este proceso se envía, junto con la petición de ruta, el mensaje de error recibido (*piggybacking* del mensaje de error) para informar a los demás nodos del enlace roto y así evitar que respondan con una ruta que utilice el enlace recién invalidado. En AODV, para evitar estos casos se utiliza simplemente el reloj lógico.

## 2.3. IEEE 802.11s

- Desarrollador: IEEE 802.11s Task Group.
- Fecha de la primera versión: Septiembre de 2003 [18]
- Versión actual: Draft 10.01, Abril de 2011 [17]
- Tipo: Híbrido.
- Nivel OSI: Nivel de Enlace de Datos.
- Conocimiento de la topología: Parcial, sólo vecinos.

### 2.3.1. Introducción

Revisión del 802.11 del IEEE para redes mesh. Está en estado de borrador. Define cómo se conectan dispositivos inalámbricos para formar una WLAN (Wireless Local Area Network) mallada o mesh. Proporciona una arquitectura y protocolos que permiten el reenvío de tramas y la selección de camino en el nivel 2 (enlace de datos) del modelo OSI.

802.11s nació como Grupo de Estudio (*Study Group*) del IEEE 802.11 en 2003. Se convirtió en Grupo de Trabajo (*Task Group*) en Julio de 2004. Actualmente sigue en proceso de aprobación, la cual está prevista para mediados de 2011, según las previsiones del IEEE.

El enrutamiento se hace mediante HWMP (*Hybrid Wireless Mesh Protocol*). Este protocolo debe ser implementado obligatoriamente por todos los nodos mesh, aunque se permite usar protocolos adicionales.

La principal ventaja de este estándar es que introduce un mecanismo de enrutamiento en la capa 2 (MAC), haciéndolo aparecer como un sistema LAN (802.x) para protocolos de capas superiores. Además, define aspectos como acceso al medio, sincronización o seguridad, no sólo cuestiones de encaminamiento mesh.

Pero el hecho de que el enrutamiento de IEEE 802.11s funcione en la capa de enlace de datos también se convierte en una desventaja, ya que de esta manera no podemos aprovechar la estructura jerárquica de protocolos de direccionamiento superiores, como IP, ni interconectar diferentes redes. Este hecho hace que sea complicado enrutar paquetes sólo con HWMP en redes mesh de tamaño medio o grande. Por ello se hace necesario combinar IEEE 802.11s con otros protocolos de capas superiores.

Existen tres implementaciones de IEEE 802.11s en la actualidad:

- Open802.11s[8]: implementación para el núcleo Linux a partir de la versión 2.6.26. Desarrollada por un consorcio empresarial formado por Nortel, Cozybit, OLPC (One Laptop per Child) y Google.
- Los equipos XO del OLPC[9]: implementan 802.11s en las tarjetas de red inalámbricas, permitiendo que el ordenador funcione como MP aunque el equipo esté en modo *standby*.

[8] `http://open80211s.org/`
[9] `http://wiki.laptop.org/go/Mesh_Network_Details`

- WifiMesh[10]: implementación para sistemas FreeBSD desarrollada por dicha comunidad. Presente a partir de la versión 8.0. Es incompatible con la versión que incluye el núcleo Linux, ya que están basadas en versiones diferentes del protocolo.

### 2.3.2. Funcionamiento

802.11s cubre diversos aspectos para redes mesh: seguridad, acceso al medio, sincronización, etc. Nosotros nos vamos a centrar en la parte más relevante para nuestro proyecto, el enrutamiento. Veamos, por lo tanto, el funcionamiento de HWMP, extraído de [5].

#### HWMP

HWMP combina características del protocolo AODV (ver 2.1) y técnicas de enrutamiento basadas en árbol. La combinación de elementos proactivos y reactivos permite una óptima y eficiente selección de ruta en una amplia variedad de redes mesh (con y sin infraestructura).

HWMP utiliza un conjunto de mensajes basados en los que utiliza el protocolo AODV, adaptados al direccionamiento MAC de la capa 2, para el descubrimiento de rutas reactivamente. Adicionalmente, se usan otros mensajes para, de forma proactiva, construir un árbol de vectores distancia a partir de un nodo raíz. Este último método necesita que el nodo a partir del cual se calculará el árbol esté configurado como *root* o raíz. Pueden existir varios nodos raíz en una misma red. Esta técnica suele utilizarse para construir árboles de rutas hacia nodos que cumplen un papel especial dentro de la red, como por ejemplo los nodos que actúan como portal (MPP) hacia otras redes.

Se contemplan dos modos de funcionamiento, dependiendo de la configuración del MP: bajo demanda y construcción proactiva de árbol, basados en las dos técnicas descritas en el párrafo anterior. Los dos modos no son excluyentes, se pueden usar de forma simultánea.

Los dos modos de funcionamiento utilizan mensajes y reglas de control comunes:

[10] `http://wiki.freebsd.org/WifiMesh`

- Los mensajes de control de HWMP, que explicaremos más adelante, son:
  - Path Request (PREQ).
  - Path Reply (PREP).
  - Path Error (PERR).
  - Root Announcement (RANN).
- La métrica para medir el coste de cada enlace determina la elección de la ruta. Para propagar esta información sobre la métrica de cada enlace, se usa un campo en los mensajes PREQ, PREP y RANN.
- Para mantener la red libre de mensajes en bucle, cada MP utiliza un número de secuencia que se propaga a otros MP's en los mensajes de control.

#### *Modo bajo demanda o reactivo*

Cuando un MP necesita encontrar una ruta mediante el modo bajo demanda, propaga a toda la red un mensaje PREQ con el MP destino y con número de secuencia 0. Un mensaje PREQ contiene, entre otras, las siguientes informaciones:

- *Discriminador:* Determina el tipo de paquete dentro del protocolo (PREQ, PREP, PERR o RRAN).
- *Tamaño:* Tamaño del paquete.
- *Contador de saltos:* Saltos desde el nodo origen hasta el actual.
- *Tiempo de vida en saltos:* Máximo número de saltos permitido para este mensaje.
- *Identificador de PREQ:* Identificador único para este PREQ.
- *Dirección del origen:* Dirección MAC del nodo que originó el PREQ.
- *Número de secuencia:* Número de secuencia para el nodo origen.
- *Tiempo de vida:* Tiempo durante el cual el mensaje se considera información válida.

- *Métrica:* Métrica acumulada desde el origen hasta el nodo actual.
- *Configuración para el destino:* Se define un bit que determina la forma en que actuarán los nodos intermedios al recibir y retransmitir un PREQ: TO (*Target Only*). Si TO vale 0, un nodo intermedio responderá con un PREP al nodo emisor de un PREQ si tiene una ruta válida para el destino solicitado. En caso contrario (TO=1, valor por defecto), sólo el destino responderá con un PREP.
- *Dirección del destino:* Dirección MAC del nodo destino.
- *Número de secuencia hacia destino:* El número de secuencia más reciente recibido por el origen, en un mensaje con información de ruta hacia el destino.

Al recibir un PREQ, un MP intermedio crea una ruta hasta el MP origen o actualiza la que ya tenía y propaga el PREQ si este contiene un número de secuencia mayor, o si el número de secuencia es igual pero el PREQ ofrece una métrica mejor que la ruta actualmente almacenada. Es decir, un mensaje PREQ no solo permite al nodo que lo origina encontrar una ruta al destino, sino que sirve para que los nodos intermedios por los que pasa actualicen su ruta hacia el nodo emisor del PREQ.

Cuando se retransmite un PREQ, su campo métrica es actualizado para reflejar la métrica acumulada para la ruta hasta el nodo que originó el PREQ.

Una vez que el mensaje ha llegado al nodo destino, éste genera un mensaje de respuesta PREP destinado exclusivamente al origen. Es decir, el PREP no inunda toda la red, si no que sigue el camino inverso al trazado por el PREQ. Este proceso está basado en AODV, véase 2.1 y 2.2. Veamos el formato de PREP:

- *Discriminador:* Determina el tipo de paquete dentro del protocolo (PREQ, PREP, PERR o RRAN).
- *Longitud:* Longitud del paquete.
- *Número de saltos:* Saltos desde el nodo origen hasta el actual.
- *Tiempo de vida en saltos:* Máximo número de saltos permitido para este mensaje.

- *Dirección del destino:* Dirección MAC del nodo para el que se ha realizado una solicitud de ruta.
- *Número de secuencia:* Número de secuencia para el nodo origen.
- *Tiempo de vida:* Tiempo de vida del PREQ al que se responde con este PREP.
- *Métrica:* Métrica acumulada desde el destino de la ruta hasta el nodo actual.
- *Número de MP's dependientes:* Cuando sea aplicable, número de MP's que han establecido una ruta al nodo raíz a través del nodo actual.
- *Direcciones de los MP's dependientes:* Direcciones MAC de los nodos que han establecido una ruta al nodo raíz a través del nodo actual.
- *Números de secuencia de los MP's dependientes:* Los números de secuencia asociados a las direcciones MAC de los nodos que han establecido una ruta al nodo raíz a través del nodo actual.

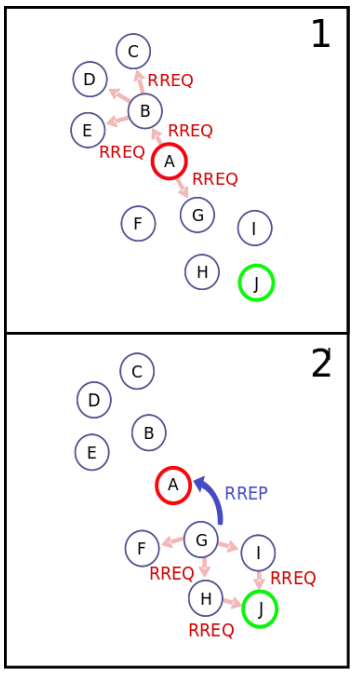


Figura 2.7: A necesita una ruta hacia J. Los nodos intermedios pueden responder y deben reenviar el PREQ

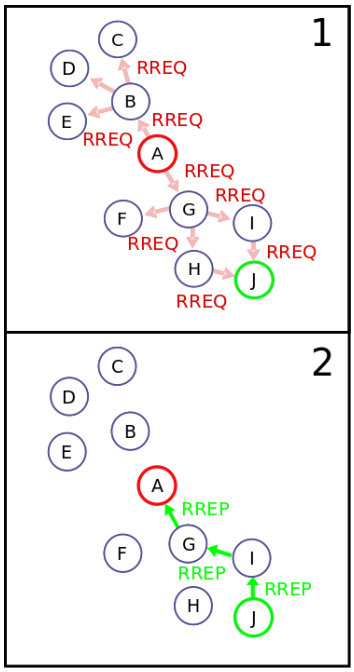


Figura 2.8: A necesita una ruta hacia J. Sólo el nodo destino puede responder con la ruta.

El nodo que solicita una ruta con PREQ puede solicitar que sólo responda con un PREP el nodo destino (Fig. 2.8) o que respondan los nodos intermedios que tengan una ruta al destino (Fig. 2.7). Esto se configura mediante el ya mencionado bit TO. Este mecanismo permite a un MP establecer de forma rápida una ruta, utilizando un PREP generado por un MP intermedio, y empezar a enviar mensajes de datos con un bajo retraso debido al descubrimiento de ruta, aunque de esta manera puede que no obtengamos la ruta de menor métrica.

Cuando el destino envía un PREP de vuelta al origen, los MP's intermedios crean una ruta al destino y retransmiten el PREP hacia el origen. Nuevamente vemos cómo la respuesta con PREP cumple dos funciones: que el nodo origen reciba la ruta hacia el destino y que los nodos intermedios actualicen o creen su ruta hacia el destino.

El nodo emisor, de igual manera, crea una ruta al destino al recibir el PREP. Si después el destino recibe más PREQ con una métrica mejor, actualiza su ruta al origen y responde con un nuevo PREP y la nueva ruta. Por lo tanto, se establece una ruta bidireccional origen-destino óptima en cuanto a la métrica.

Como vemos, este modo de funcionamiento de 802.11s es prácticamente idéntico a AODV. La nueva aportación de 802.11s es la parte proactiva, en la que se introduce un nuevo mensaje de control, RRAN.

### *Construcción proactiva de árbol*

Existen dos mecanismos que permiten propagar información de enrutamiento para alcanzar el nodo raíz de forma proactiva. El primero utiliza mensajes PREQ proactivos para crear rutas entre el nodo raíz y todos los demás nodos de la red. El segundo método usa mensajes RANN (*Root Announcement*) para diseminar la información de ruta hacia el nodo raíz.

Un nodo configurado como raíz, enviará periódica y reactivamente mensajes PREQ o RRAN.

#### -PREQ proactivo.

El proceso de construcción del árbol comienza con el envío proactivo de un mensaje PREQ *broadcast* por parte del nodo raíz, TO a 1. El PREQ contiene la métrica de distancia (puesta a 0 inicialmente por el nodo raíz) y un número de secuencia. Este mensaje es enviado periódicamente por el nodo raíz, con números de secuencia crecientes.

Cualquier nodo que "escuche" un PREQ proactivo, actualiza (o crea) su ruta hasta el nodo raíz, actualiza la métrica y el número de saltos del mensaje PREQ y reenvía la versión actualizada. De esta forma, la información sobre la presencia y distancia a los nodos raíz de la red es diseminada por toda la red.

Al igual que en el caso reactivo, un nodo actualizará su ruta hacia el nodo raíz si, y sólo si, el número de secuencia del PREQ es mayor o si es igual pero el mensaje ofrece una mejor métrica para la ruta hasta el nodo raíz.

Un MP que reciba un PREQ, responderá al nodo raíz con un PREP proactivo dependiendo del bit "Proactive PREP" del PREQ. Si este bit es 0, el MP responderá con un PREP sólo si es necesario (si, por ejemplo, debe establecer una ruta bidireccional con el nodo raíz para enviar paquetes de datos). Si el bit vale 1, el nodo responde con un PREP. El PREP establece una ruta entre el MP y el nodo raíz.

**-RRAN proactivo.**

Periódicamente, el MP raíz inunda la red con mensajes RANN. Estos mensajes contienen información sobre la métrica de las rutas hacia el nodo raíz. Veamos el formato:

- *Discriminador:* Determina el tipo de paquete dentro del protocolo (PREQ, PREP, PERR o RRAN).
- *Longitud:* Longitud del paquete.
- *Nodo raíz:* Determina si el nodo es un nodo raíz.
- *Número de saltos:* Saltos desde el nodo origen hasta el actual.
- *Tiempo de vida en saltos:* número de saltos que todavía puede realizar este mensaje.
- *Dirección del origen:* Dirección MAC del nodo raíz.
- *Número de secuencia:* Número de secuencia para el nodo raíz.
- *Tiempo de vida:* Tiempo durante el cual el mensaje se considera información válida.
- *Métrica:* Métrica acumulada desde el origen hasta el nodo actual.

Los mensajes RRAN sirven para diseminar la información acerca de la ruta hacia el/los nodo(s) raíz, pero, a diferencia de los paquetes PREQ o PREP, no se utiliza para actualizar o crear rutas.

Cuando un RRAN es recibido por un nodo que debe crear o actualizar su ruta hasta el nodo raíz, se le responde a este mediante un PREQ unicast a través del MP por el que se recibió el RRAN.

Este PREQ de respuesta seguirá las mismas reglas de procesamiento que en el caso bajo demanda.

A su vez, el nodo raíz responde con un PREP unicast a cada PREQ que recibe. El mensaje PREQ crea la ruta inversa del nodo raíz al MP origen, y el PREP la ruta del MP origen al raíz. Podemos ver este proceso en la figura 2.9

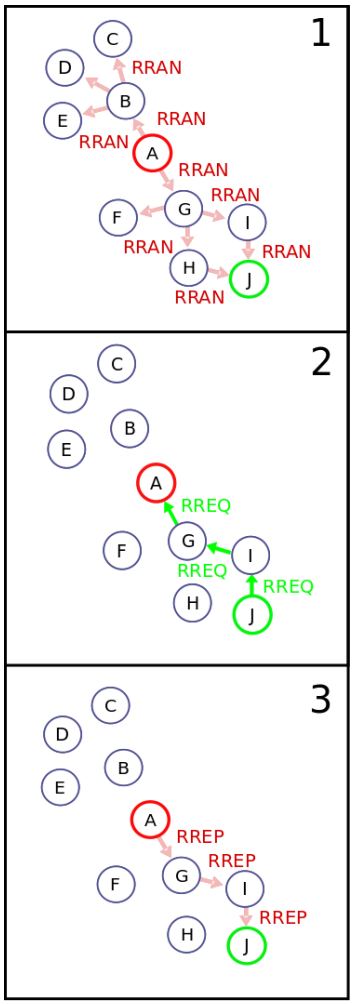


Figura 2.9: RRAN proactivo. J actualiza su ruta al nodo raíz A

### *Emisión de mensajes de error, PERR*

El mensaje PERR es utilizado para anunciar destinos inalcanzables, ya sea por enlaces rotos, por caída de nodos o por carecer de información para llegar a dicho destino.

Un nodo generará un mensaje de error en cualquiera de los siguientes casos:

- El nodo detecta que el siguiente salto para una de sus rutas activas es inutilizable.

- El nodo recibe un paquete de datos dirigido a otro nodo (que no sea él mismo) para el que no posee información de ruta.

El mensaje se envía a los precursores del nodo que utilizaban la ruta que ha dejado de estar activa.

Veamos a continuación el formato de un mensaje PERR:

- *Discriminador:* Determina el tipo de paquete dentro del protocolo (PREQ, PREP, PERR o RRAN).
- *Longitud:* Longitud del paquete.
- *Número de destinos:* Número de nodos destino para los que se anuncia un fallo en la ruta.
- *Direcciones destino:* Direcciones MAC de los nodos destino para los que se anuncia un fallo en la ruta.
- *Números de secuencia destino:* Números de secuencia de los nodos destino para los que se anuncia un fallo en la ruta.

Cuando un nodo detecta un enlace roto con un vecino y dicho enlace forma parte de una ruta activa, el nodo en cuestión emitirá un mensaje PERR anunciado todos los nodos que dejan de ser alcanzables por esa ruta. Antes de enviar un PERR el nodo emisor incrementa el número de secuencia para los destinos inalcanzables debido al error e invalida las rutas para esos destinos. Esta parte también es igual en AODV.

Los nodos que reciben un PERR actualizan su información de rutas (si procede) y lo reenvían a sus vecinos precursores en la ruta invalidada.

## 2.4. B.A.T.M.A.N.

- Desarrollador: A. Neumann, C. Aichele, M. Lindner, S. Wunderlich (Comunidad Freifunk).
- Fecha de la primera versión: Abril de 2008 [9]
- Versión actual: RFC Draft draft-wunderlich-openmesh-manet-routing-00, Abril de 2008 [9]

- Tipo: Proactivo.
- Nivel OSI: Nivel de Red.
- Conocimiento de la topología: Parcial, sólo vecinos.

### 2.4.1. Introducción

El protocolo B.A.T.M.A.N. ha sido diseñado por Freifunk, una comunidad wireless sin ánimo de lucro, con base, principalmente, en Alemania. Forman grupos de usuarios interconectados mediante tecnología mesh y además desarrollan software libre como, por ejemplo, la implementación de B.A.T.M.A.N.

En B.A.T.M.A.N. ningún nodo tiene la información completa de la topología de la red. En cambio, para construir su tabla de enrutamiento, almacena la mejor dirección para llegar a cada nodo. Cuando hablamos de "dirección", nos referimos al enlace inalámbrico de un salto que conecta a un nodo con otro. Para cada enlace inalámbrico de un salto, se cuentan los paquetes que llegan de cada nodo. El enlace por el que lleguen más paquetes de un nodo en concreto, será el mejor enlace para enviar paquetes a dicho nodo. De esta manera, se "comparte" el conocimiento de la topología de la red, y cada nodo solo almacena la mejor dirección para cada destinatario.

Actualmente, encontramos las siguientes implementaciones de B.A.T.M.A.N., todas impulsadas por la comunidad Freifunk a través del proyecto Openmesh[11] y con licencia GPLv2[12]:

- Batmand[13]: para sistemas GNU/Linux. Funciona como un demonio en espacio de usuario y opera en la capa de red.
- BMX (BatMan-eXperimental)[14]: *fork* del proyecto original. En un principio se ideó como una versión de pruebas en la que experimentar con nuevas características y conceptos, pero el proyecto evolucionó hacia una dirección distinta a B.A.T.M.A.N. y ahora es un proyecto completamente diferente.

[11] `http://www.open-mesh.org/`

[12] GNU General Public License, version 2. `http://www.gnu.org/licenses/gpl-2.0.html`

[13] `http://www.open-mesh.org/wiki/open-mesh/BranchesExplained#batmand`

[14] `http://www.bmx6.net/`

- Batman-adv[15]: implementación en forma de módulo del kernel para sistemas GNU/Linux. Opera en la capa de enlace de datos. Desde la versión 2.6.38. está incluido en el núcleo Linux.

### 2.4.2. Funcionamiento

Las características principales de B.A.T.M.A.N. son:

- B.A.T.M.A.N. distingue entre nodos e interfaces. Un nodo puede tener más de una interfaz formando parte de la red.
- El paquete utilizado para informar a los demás sobre la presencia de un nodo se llama OGM (*OriGinator Message*).
- Se utilizan números de secuencia para diferenciar información nueva y obsoleta.
- Se hace uso de una ventana deslizante para almacenar los números de secuencia, hasta que son considerados fuera de rango (fuera del borde inferior de la ventana). El tamaño de la ventana es configurable mediante la constante WINDOW_SIZE.
- La cantidad de números de secuencia almacenados en la ventana se usa como métrica para determinar la calidad de los enlaces y rutas.

#### Generación y difusión de OGM's

Como hemos mencionado antes, un nodo B.A.T.M.A.N. utiliza paquetes OGM para anunciar su presencia al resto de la red. Esto lo hace de forma periódica. El intervalo entre la emisión de dos OGM's se define en la variable ORIGINATOR_INTERVAL. El OGM es retransmitido a todos sus vecinos. Estos, a su vez, lo retransmiten a los suyos. El número de OGM's recibidos de un nodo dado a través de cada vecino se usa para calcular la calidad de la ruta. Es decir, si recibimos OGM's de un nodo determinado a través de varios vecinos, aquel vecino por el que hayamos recibido más OGM's de ese nodo será el siguiente salto en la ruta hacia el nodo emisor del OGM. Los OGM retransmitidos por enlaces pobres o lentos tardarán más en propagarse, o se producirán pérdidas o corrupciones de datos, favoreciendo la rápida y

[15] `http://www.open-mesh.org/wiki/open-mesh/BranchesExplained#batman-adv`

correcta propagación de los OGM que viajen por buenos enlaces. Podemos ver este proceso en la figura 2.10: G recibe el OGM1 de J a través de H e I. Se produce una pérdida de datos a través de I y el OGM2 sólo le llega por H. Como G ha recibido dos OGM's de J a través de H y sólo uno a través de I, G designa a H como su pasarela hacia J. Para saber si se ha recibido un OGM más de una vez, estos llevan asociado un número de secuencia.

Un OGM contiene, por lo menos, la IP del emisor, la IP del nodo que retransmite el paquete, un TTL (*Time To Live*) y un número de secuencia, generado por el emisor, que permite saber si un paquete ha sido recibido más de una vez. Veamos la información más relevante que transporta un OGM:

- *Tiempo de vida:* Define un número máximo de saltos que el OGM puede atravesar.
- *Pasarela:* Si el nodo emisor del OGM es una pasarela hacia Internet, codifica el ancho de banda aproximado de subida y de bajada.
- *Número de secuencia:* Número de secuencia del OGM. Sirve para determinar si la información que transporta el OGM es nueva o está obsoleta.
- *Dirección del origen:* Dirección IPv4 de la interfaz por la que el origen envió el OGM.

### Mantenimiento y actualización de rutas

Para registrar y mantener actualizada la información sobre las rutas se utiliza una lista, en la que se registran todos los nodos de los que hemos recibido por lo menos un OGM. Para ello se guardan, entre otras cosas, su dirección IP, el tiempo transcurrido desde que se recibió el último OGM, su actual número de secuencia y una lista de información de vecinos. En esta lista se mantiene, para cada vecino del nodo, una estructura de ventana deslizante que va almacenando los últimos OGM's recibidos.

La ventana deslizante tiene un tamaño determinado por la variable WINDOW_SIZE. En ella se guarda el número de secuencia del último OGM recibido y los WINDOW_SIZE-1 números anteriores. Para cada número que esté dentro de los límites de la ventana se refleja si se ha recibido o no el OGM correspondiente. Esto nos sirve para calcular la métrica de la ruta hacia el emisor del OGM a través del vecino asociado a la ventana.

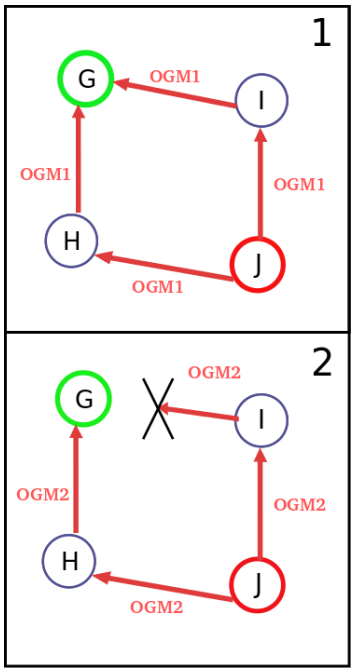


Figura 2.10: Selección de ruta según número de OGM's recibidos.

Si recibimos un OGM con un número de secuencia dentro de los límites de la ventana, no se actualiza nada. Pero si recibimos un número de secuencia que no está comprendido en la ventana, al actual número de secuencia se le asigna el que hemos recibido y por lo tanto la ventana se mueve.

Una ruta hacia un nodo será borrada cuando no se reciba un nuevo OGM de dicho nodo por un tiempo superior a WINDOW_SIZE y al intervalo PURGE_TIMEOUT.

### Anuncio de pasarela

Además, si el nodo ofrece un acceso a redes distintas a B.A.T.M.A.N., debe anexar un mensaje de extensión HNA (*Host Network Announcement*) para cada red que desee anunciar. Veamos la estructura de un HNA:

- *Máscara de red:* Número de bits que indican el prefijo de la red anunciada.
- *Dirección de red:* Dirección de la red que se anuncia.

### Procesamiento y retransmisión de OGM's

Cuando un nodo recibe un OGM, lo retransmite, como mucho una vez, y solo si ha sido recibido del vecino designado como el mejor enlace para llegar al emisor del OGM, consiguiendo una dispersión o *flooding* selectivo.

Al recibir un OGM de otro nodo, nos encontramos con dos posibilidades:

1. El MP receptor no tiene al emisor en su tabla de rutas: se crea una nueva entrada en la tabla de rutas, almacenando la interfaz por la que se recibió y el vecino a través del cual se alcanza al emisor. Esto se hace incluso para los vecinos de un salto.
2. El MP receptor ya tiene al emisor en su tabla de rutas: si el número de secuencia del OGM ya está contenido en la ventana deslizante asociada al enlace por el que se recibió el OGM, no se actualiza nada. Si el número de secuencia es nuevo (no está contenido en la ventana), se actualizan las ventanas deslizantes de todos los enlaces conocidos para llegar al emisor. Después de actualizar, el enlace con la ventana deslizante que contenga la mayor cantidad de números de secuencia para ese emisor pasará a ser la pasarela hacia dicho nodo.

El nodo retransmitirá el OGM si:

- Ha sido recibido de un vecino de un salto.
- Ha sido recibido por el enlace designado como pasarela hacia el emisor del OGM y el OGM no es un duplicado.

## 2.5. Babel

- Desarrollador: Juliusz Chroboczek
- Fecha de la primera versión: Abril de 2009 [22]
- Versión actual: RFC-6126, Abril de 2011 [6]
- Tipo: Proactivo
- Nivel OSI: Nivel de Red
- Conocimiento de la topología: Parcial, sólo vecinos.

### 2.5.1. Introducción

Babel es un protocolo desarrollado por Juliusz Chroboczek, profesor de la Unidad de Formación e Investigación de la Universidad Diderot de París. Utiliza una técnica basada en vector de distancias, inspirada en los protocolos AODV (descrito en 2.1), DSDV[16], y EIGRP[17] de Cisco para enrutar paquetes en una red mesh.

Funciona con IPv4 e IPv6 y fue diseñado, en un principio, para redes inalámbricas ad-hoc, por lo que es un protocolo muy robusto en presencia de nodos móviles, impidiendo la formación de bucles sin fin y ofreciendo una rápida convergencia.

Además de Babel, el profesor Chroboczek ha desarrollado un protocolo para la configuración y asignación de direcciones IP (versiones 4 y 6), análogo a DHCP. El protocolo se llama AHCP[18] (*Ad-Hoc Configuration Protocol*).

Babel está basado en el algoritmo de Bellman-Ford[19], al que incorpora ciertos refinamientos que previenen la formación de bucles o, al menos, aseguran que un bucle desaparecerá después de un cierto tiempo y no volverá a aparecer.

Las siguientes implementaciones están disponibles[20]. Las dos han sido desarrolladas por J. Chroboczek y se distribuyen bajo licencia libre:

- Babeld: demonio para espacio de usuario en sistemas GNU/Linux. Opera en la capa de red.
- BabelZ: rama experimental del proyecto que introduce un refinamiento para minimizar la influencia de interferencias en las comunicaciones. Demonio en espacio de usuario para sistemas GNU/Linux. Capa de red.

[16]Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Charles E. Perkins, Pravin Bhagwat. `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.555`

[17]Introduction to EIGRP. Cisco Systems. `http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080093f07.shtml`

[18]AHCP — the Ad-Hoc Configuration Protocol. `http://www.pps.jussieu.fr/~jch/software/ahcp/`

[19]The Bellman-Ford Algorithm. The Babel Routing Protocol (RFC 6126). `http://tools.ietf.org/html/rfc6126#section-2.2`

[20]Existe información y enlaces de descarga para ambas en `http://www.pps.jussieu.fr/~jch/software/babel/`

### 2.5.2. Funcionamiento

A continuación definiremos las peculiaridades y conceptos básicos que diferencian a Babel de otros protocolos de enrutamiento para redes ad-hoc y mesh, basándonos en [6]. Seguidamente pasaremos a describir el funcionamiento y formato de los paquetes.

- Nodos e interfaces: igual que B.A.T.M.A.N., Babel distingue entre nodos e interfaces. Un nodo puede tener más de una interfaz formando parte de la red.

- Métrica: Babel se basa en la calidad de los enlaces entre nodos para calcular la métrica de una ruta. Se define el coste de un enlace entre A y B como C(A,B) y la métrica de A a un destino S como D(A). Para determinar estas métricas se utilizan dos tipos de paquetes: *Hello* y IHU (*I Heard yoU*).

- Algoritmo de Bellman-Ford: este algoritmo calcula la ruta más corta entre dos nodos en un grafo dirigido y ponderado. Babel se basa en él para el cálculo de rutas. El algoritmo se ejecuta en paralelo para cada nodo, calculando las distancias a los demás nodos de la red. Para la explicación teórica que sigue, se define un nodo destino S de referencia para el que se calculará la distancia mínima.

- Condición de viabilidad: cuando el enlace entre dos nodos vecinos se rompe, pueden crearse bucles en la actualización de rutas[21]. Para evitar esto, se define la condición de viabilidad como aquella que permite a un nodo rechazar un anuncio de ruta por parte de otro nodo, si dicho anuncio puede crear un bucle sin fin. Las condiciones de viabilidad pueden ser muy diversas. Babel usa la misma que el protocolo EIGRP. Dado un nodo A, se define la distancia de viabilidad de A a S, FD(A), como la menor métrica que A ha anunciado alguna vez para llegar a S. Es decir, una actualización de ruta D(B) hacia S anunciada por un vecino B de A será viable si D(B) es estríctamente menor que la distancia de viabilidad de A, FD(A) (D(B) $<$ FD(A)).

- Números de secuencia: Babel utiliza números de secuencia para las rutas, una solución introducida por DSDV y AODV. Además de la

[21]The Babel Routing Protocol. J. Chroboczek. `http://www.pps.jussieu.fr/~jch/software/babel/draft-chroboczek-babel-routing-protocol-05.html#anchor8`

métrica, cada ruta transporta un número de secuencia, un número entero no decreciente que se propaga inalterado por toda la red. El único que puede incrementar el número de secuencia es el nodo origen. El par (métrica, núm. secuencia) se conoce como distancia. De esta forma, podremos saber si la información recibida sobre una ruta es nueva o antigua.

### Búsqueda de rutas

Para cada nodo S destino, un nodo A mantiene dos campos de datos: la distancia estimada hacia S, D(A), y el próximo salto (de entre sus vecinos) para llegar a S, NH(A), del inglés *Next Hop*. Inicialmente D(A) vale infinito y NH(A) está indefinido. A medida que se van calculando los costes de los enlaces y las métricas de las rutas, y después de que los nodos hayan anunciado sus distancias a S (proceso de convergencia de la red) tendremos una situación como la que se ilustra en la figura 2.11.

Con el fin de advertir de su presencia al conjunto de sus vecinos, un nodo envía periódicamente paquetes *Hello* por todas sus interfaces, en modo *broadcast*. Veamos la estructura de un paquete *Hello*:

- *Discriminador:* Indica que se trata de un paquete *Hello*.
- *Longitud:* Longitud del cuerpo del mensaje.
- *Número de secuencia:* Valor del número de secuencia del paquete para la interfaz que lo envía.
- *Intervalo:* Límite máximo de tiempo en el que se volverá a enviar otro paquete *Hello*.

Cada nodo mantiene un historial de los paquetes *Hello* recibidos para cada uno de sus vecinos en una estructura de datos de ventana deslizante, al igual que B.A.T.M.A.N.

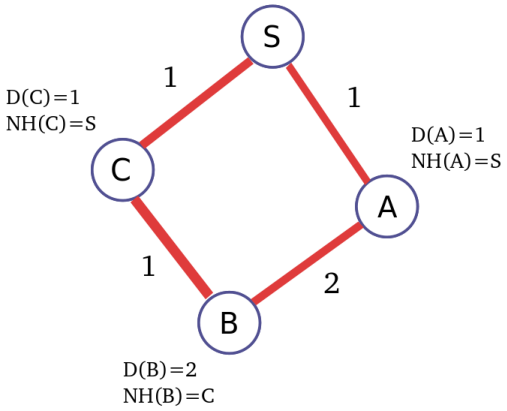


Figura 2.11: Distancia y próximo salto de A, B y C para llegar a S, una vez alcanzada la convergencia de la red.

Cuando recibe un paquete *Hello*, el nodo receptor contesta con un paquete IHU (*I Heard You*). De esta manera, un nodo A, que ha enviado un paquete *Hello*, computa el coste del enlace con un vecino B, que ha respondido con un paquete IHU. El formato de este último paquete es el siguiente:

- *Discriminador:* Indica que se trata de un paquete IHU.
- *Longitud:* Longitud del cuerpo del mensaje.
- *Codificación del campo dirección:* Indica cuál es el formato del campo dirección de destino.
- *Coste de recepción:* El coste de recepción, desde el punto de vista del nodo que recibe el IHU. Este valor se calcula a partir del historial de paquetes *Hello*.
- *Intervalo:* Límite máximo de tiempo en el que se volverá a enviar otro paquete IHU.

- *Dirección:* Dirección del nodo destino, en el formato especificado en el campo Codificación del campo dirección.

Periódicamente, cada nodo B envía una actualización de ruta a todos sus nodos vecinos, con su D(B). Cuando un nodo A recibe la actualización de B, comprueba si B es su siguiente salto para llegar a S. Si ese es el caso y se cumple la condición de viabilidad, entonces NH(A)=B y D(A)=C(A,B) + D(B). En caso contrario, A compara C(A,B) + D(B) con su valor actual de D(A). Si es menor, es decir, la si ruta es mejor, entonces A asigna NH(A)=B y D(A)=C(A,B) + D(B). Si se rompe un enlace de un nodo A con un vecino B, el coste del enlace pasa a valer infinito.

Cada nodo mantiene una tabla, en la que se almacena, para cada vecino, la siguiente información:

- Interfaz por la que se llega al vecino.
- Dirección de la interfaz del vecino.
- Historial de paquetes *Hello* del vecino.
- Coste de transmisión, contenido en el último paquete IHU del vecino.
- Número de secuencia esperado en el próximo paquete *Hello* del vecino.

De esta manera, cada nodo mantiene actualizado el coste de transmisión con cada uno de sus vecinos.

Además, el nodo almacena una tabla de rutas que refleja (entre otras cosas), para cada nodo S, el vecino para llegar a S y la métrica de la ruta. El nodo anuncia, de forma periódica y en *broadcast*, dicha tabla a sus vecinos. También hace lo mismo si se produce un cambio significativo en una de sus rutas almacenadas.

## 2.6. OLSR

- Desarrolladores: T. Clausen y P. Jacquet
- Fecha de la primera versión: Noviembre de 1998 [21]
- Versión actual: RFC-3626, Octubre de 2003 [7]

- Tipo: Proactivo
- Nivel OSI: Nivel de Red
- Conocimiento de la topología: Completa

### 2.6.1. Introducción

Fue creado por Thomas Clausen y Philippe Jacquet en el proyecto Hipercom INRIA. Se trata de un protocolo proactivo basado en la optimización de los clásicos protocolos *link-state* o estado de enlace.

A diferencia de AODV, OLSR es un protocolo proactivo, es decir, periódicamente se envían mensajes para mantener las tablas de enrutamiento actualizadas. OLSR funciona bien en redes con alto número de usuarios (nodos) y con una topología cambiante. Para llevar un control, se intercambian periódicamente mensajes de tal forma que se va aprendiendo la topología de la red y el estado de los nodos vecinos.

OLSR es adecuado para redes en las que el tráfico se produce entre un gran conjunto de nodos, y no entre un pequeño grupo solamente. Cuanto mayor es la red, en número de nodos y en densidad, mayor es la optimización de OLSR con respecto al protocolo *link state* clásico [23].

Se diseñó para trabajar en un ambiente completamente distribuido, sin necesidad de una entidad central. Tampoco requiere que la transmisión de los paquetes del protocolo se realice de forma fiable. Dichos paquetes se envían periódicamente, y se asume que algunos se perderán, algo que ocurre con frecuencia en redes inalámbricas.

El intercambio de muchos paquetes en una red mesh provoca una congestión en la red y supone un grave problema en las comunicaciones. Para solucionar esto, OLSR utiliza los *Multi Point Relay* (MPR). Los MPRs son los nodos de la red encargados de retransmitir los paquetes *broadcast*. El número de MPR que retransmiten un paquete *broadcast* es siempre menor que el número total de nodos, por lo que con esta técnica se aminora considerablemente el coste de las inundaciones de paquetes *broadcast*. Sin embargo, el protocolo sigue proporcionando las rutas óptimas, pues dichos paquetes siguen llegando a todos los nodos.

Como ya hemos dicho, OLSR es un protocolo proactivo, esto quiere decir que el intercambio de información sobre la topología de la red se realiza

periódicamente, en lugar de bajo demanda, como ocurriría en un protocolo reactivo. Los protocolos proactivos tienen la ventaja de ser más rápidos en las primeras conexiones, ya que la topología de la red se conoce previamente y no es necesario calcularla antes de enviar cada paquete. La desventaja de los protocolos proactivos es que requieren que se envíe periódicamente paquetes de control, lo que supone una sobrecarga para la red. La sobrecarga también aparece en los protocolos reactivos, pero sólo cuando el número de peticiones es relativamente alto.

Actualmente se está desarrollando la versión 2 de este protocolo donde las diferencias claves son el diseño flexible y modular usando componentes compartidos como el formato de paquetes "packetbb" o el descubrimiento de vecinos por el protocolo NHDP [16]. Estas características están siendo incorporadas a muchos de la siguiente generación de protocolos para redes ad-hoc.

Para OLSR existen varias implementaciones actuales que funcionan en varios sistemas:

- *NRL-OLSR:*[22] Implementación realizada por el *Naval Research Laboratory* de los Estados Unidos. Soporta IPv6 y varios sistemas operativos como Windows, MAC OS y Linux. Altamente configurable y con posibilidad de mostrar mucha información sobre el estado actual del nodo.

- *Olsrd:*[23] Bajo la licencia BSD, esta es la implementación más extendida del protocolo. Funciona bajo multitud de sistemas como Windows, Linux, MAC OS X, BSD o Android. Cumple completamente con el RFC, tanto las funciones básicas como las adicionales. Permite el uso de IPv4 tanto como de IPv6. Utilizados en redes con miles de nodos actualmente [24].

- *INRIA OLSR:*[25] Creado por INRIA es una implementación que funciona bajo Linux y Windows que sólo soporta IPv4. Escrito en C++, también cumple de manera completa el RFC. Ha sido probado en simulaciones con cientos de nodos.

[22] `http://cs.itd.nrl.navy.mil/work/olsr/index.php`
[23] `http://www.olsr.org/`
[24] `http://wind.awmn.net/?page=nodes`
[25] `http://hipercom.inria.fr/olsr/`

### 2.6.2. Funcionamiento

Para empezar a explicar el protocolo OLSR vamos a ilustrar el funcionamiento de su tecnología más característica, los nodos MPR [7]. Como hemos explicado antes, un protocolo proactivo corre el riesgo de saturar la red con envíos masificados. Para impedir la congestión de la red, OLSR usa los nodos MPR. Esto consiste en seleccionar un mínimo conjunto de nodos vecinos, aquellos con los que tiene comunicación directa sin necesidad de usar nodos intermediarios, que sean capaces de llegar a todos los nodos que se encuentran a dos saltos de distancia, los vecinos de mis vecinos.

De esta forma, un nodo selecciona su conjunto de nodos MPR, y sólo intercambiará mensajes de control con ellos. Así se evita el envio masivo de mensajes en *broadcast* a toda la red.

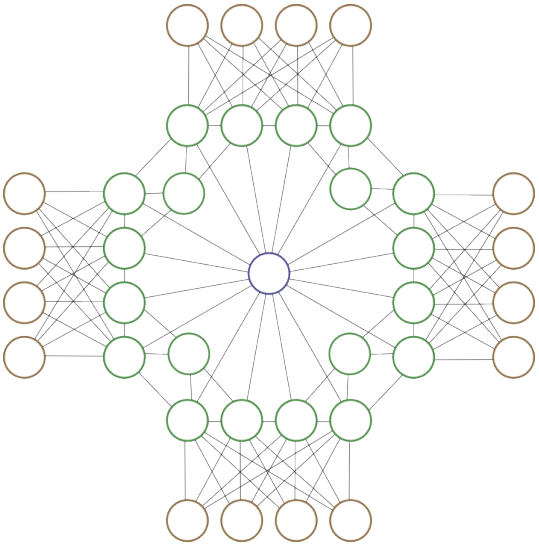

Figura 2.12: Ejemplo de conexiones en una red mesh

Podemos ver un ejemplo en la ilustración 2.12. Sin duda, inundar la red con mensajes broadcast provocaría que un nodo recibiera innecesariamente el mismo mensaje repetidas veces y que todos tuvieran que replicarlo una vez. Para este caso, algo extremo por la concentración de nodos, se producirían 37 envíos de mensajes *broadcast* y cada nodo podría recibir unos 9. Esto tiene un coste enorme tanto en ancho de banda, como en CPU para procesar los mensajes que llegan, y de batería por el envío. Esto es muy negativo para las ad-hoc en general.

OLSR optimiza el envío indiscriminado de mensajes *broadcast*. Como

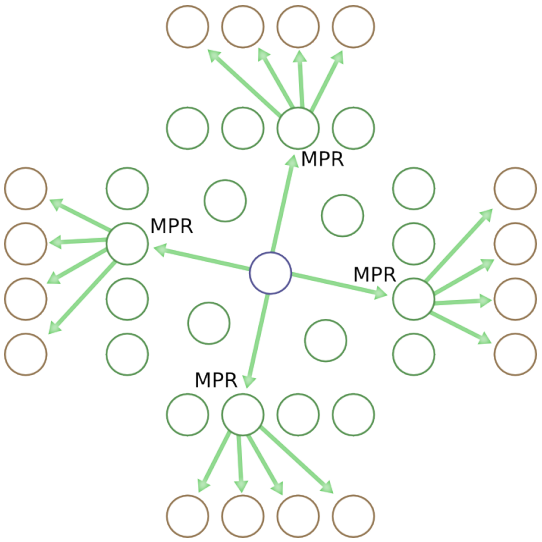


Figura 2.13: Selección de nodos MPR

podemos observar en la imagen 2.13 el nodo central ha seleccionado unos nodos llamados MPR. La característica de estos nodos es que son capaces de llegar a todos los nodos que se encuentran a dos saltos de distancia y siendo el mínimo conjunto, es decir, evitando tener varios nodos que lleguen a los mismos vecinos innecesariamente. Cuando el nodo necesite enviar información de control al resto de la red enviará un mensaje broadcast con un TTL mayor que uno, como se hace en otros protocolos como AODV o DSR. La diferencia es que en OLSR los envíos *broadcast* sólo serán retransmitidos por aquellos nodos que hayan sido declarados como MPR por el nodo emisor. De esta manera tan simple nos ahorramos muchas transmisiones y procesados innecesarios obteniendo el mismo resultado: que todos los nodos de la red reciban el mensaje. En este caso pasaríamos al envío de tan sólo 5 mensajes y cada nodo lo recibiría como máximo dos veces.

Antes de continuar explicando cómo los nodos averiguan cuáles son sus nodos MPR, estudiemos cómo se realizan las comunicaciones en OLSR. El paquete básico de OLSR está dividido en tres partes: la cabecera del paquete, la cabecera del mensaje y los datos del mensaje. Un paquete OLSR enviado tendrá solamente una cabecera de paquete con la siguiente información:

- *Longitud del paquete:* El tamaño que tiene el paquete en bytes, incluyendo la cabecera.
- *Número de secuencia:* Número de secuencia del paquete que se va in-

crementado en una unidad cada vez que se envía. Se usa para saber cómo de actual es la información que contiene.

En un mismo paquete OLSR puede haber varios mensajes, cada uno con su cabecera y su información. La cabecera de un mensaje tiene los siguientes campos:

- *Discriminador:* Para conocer el tipo de mensaje que contiene esta cabecera. En OLSR existen cuatro tipos de mensajes: HELLO, TC, MID y HNA.
- *Tiempo de vida:* Indica durante cuánto tiempo después de la recepción del mensaje la información que contiene debe ser considerada válida.
- *Longitud del mensaje:* El tamaño de la información del mensaje que viene justo después de esta cabecera.
- *Dirección origen:* La dirección del nodo que creó el mensaje originalmente. Es importante recalcar que esta información puede ser distinta a la que se encuentre en la cabecera IP. Esto puede ocurrir porque un nodo haya hecho de intermediario desde el nodo origen hasta el destino. Por tanto este campo nunca debe modificarse.
- *TTL:* A diferencia de AODV o DSR que utilizan el TTL de la cabecera IP para medir el número de saltos que puede tomar un paquete, OLSR reserva un campo en la cabecera del mensaje para este cometido. Esto es así porque en un mismo paquete pueden coexistir varios mensajes y cada uno podría tener un TTL diferente.
- *Número de saltos:* El número de saltos que ha realizado el paquete desde el nodo origen hasta el nodo que ha recibido este mensaje.
- *Número de secuencia:* Número de secuencia del mensaje. Debe ser único para cada uno de los mensajesque envíe el nodo fuente. Como el resto de número de secuencias que encontramos en los demás protocolos, es utilizado para conocer si el mensaje ya ha sido recibido por el nodo y poder descartar los duplicados o anticuados.

Después de la cabecera del paquete nos encontramos con la cabecera del primer mensaje y su información relacionada. Es posible que detrás de este primer mensaje nos encontremos con varios más consecutivamente, de

manera que se aproveche el envío de un paquete al máximo. Como vemos, OLSR está optimizado para minimizar el número de mensajes transmitido entre los nodos.

Para confeccionar la lista de los nodos MPR, cada nodo envía periódicamente un mensaje HELLO a todos los nodos vecinos transmitiéndolo por broadcast. Este tipo de paquete tiene asignado el TTL a uno, por lo cual el mensaje sólo llega a los nodos que se encuentran a un salto de distancia. No es retransmitido por toda la red ya que solamente tiene interés para los vecinos del nodo.

Un mensaje HELLO contiene otra pequeña cabecera con la siguiente información:

- *Intervalo de tiempo:* Este campo especifica el intervalo de emisión usado para transmitir los mensajes HELLO por el nodo.
- *Disposición:* Este campo especifica la disponibilidad del nodo para llevar y retransmitir tráfico hacia otros nodos. Es útil para los nodos conocer la disposición a la hora de seleccionar sus MPR.

Después de la cabecera nos encontramos con una lista de los vecinos organizados de la siguiente forma:

- *Código de enlace:* El tipo de conexión que mantiene actualmente con los vecinos que se especifican a continuación. Este campo realmente se divide en dos:
  1. *El estado del enlace:* Los valores que puede tener este campos son simétrico, asimétrico, no especificado o perdido.
  2. *El tipo de vecino:* Con valores muy parecidos a los anteriores, indica si los nodos son vecinos MPR y si tienen conexiones simétricas o asimétricas.
- *Tamaño del mensaje de enlace:* El tamaño de la lista de vecinos que viene posteriormente.
- *Direcciones de vecinos:* La lista de direcciones de los vecinos que tienen el tipo de conexión determinado anteriormente.

Se puede apreciar que en un mismo paquete HELLO puede venir varias listas de vecinos clasificados por el campo "código del enlace" que depende del tipo de enlace que mantienen con el nodo emisor.

Estos mensajes HELLO contienen los nodos vecinos del nodo emisor. De esta manera cada nodo conoce a sus nodos vecinos, pero también a los vecinos de éstos; es decir, tiene conocimiento de todos los nodos que se encuentran a dos saltos de distancia.

Cada nodo generará dos tablas con la información obtenida de los paquetes HELLO. Una primera donde se encuentran todos sus nodos vecinos, es decir, todos los nodos de los que va recibiendo los paquetes HELLO. Junto a la dirección del nodo, se guarda además, el estado del enlace que mantiene con ellos. Los posibles valores que puede tomar este atributo son: MPR, bidireccional o unidireccional. El enlace se considera unidireccional cuando el nodo recibe paquetes HELLO pero no se encuentra incluido en ellos como vecino. Esto le sugiere al nodo que recibe los paquetes pero que no ocurre a la inversa, por lo que el nodo externo no le puede considerar vecino.

Con la información de la tabla anterior, más el conocimiento de cuáles son los nodos que se encuentran a dos saltos y quién es el vecino que los conecta, se genera una segunda tabla. Para ello se emplea un algoritmo de mínimos conjuntos, teniendo en cuenta la disponibilidad de los nodos, y se calcula cuáles son sus nodos MPR. Una vez realizados los cálculos, en esta segunda tabla se guardan los nodos de dos saltos y a través de qué nodo MPR se accede ellos. Por tanto, simplemente con los mensajes HELLO, ya tenemos conocimiento completo de cómo es nuestra red con un radio de dos saltos.

Aparte de la información de los nodos vecinos, los mensajes HELLO añaden cuáles de ellos han sido seleccionados por el nodo emisor para que sean sus MPR. Por lo cual, cuando se recibe un paquete HELLO, además de conocer la topología de la red de uno o más saltos, sabrá si ha sido seleccionado por su vecino como MPR y por tanto estará en la obligación de reenviar los mensajes de control que le envíe a partir de ahora.

Por otro lado, existen los mensajes Topology Control (TC). Un mensaje de este tipo contiene los siguientes campos:

- *Número de secuencia de transmisión de vecinos:* Un número de secuencia para conocer si la información transmitida es moderna o no.

- *Lista de vecinos públicos:* La lista con las direcciones de lo vecinos del nodo.

Un mensaje TC es enviado por un nodo a la red para declarar un conjunto de enlaces, los cuales deben incluir al menos los enlaces con todos sus nodos MPR. Los TC tienen el TTL máximo permitido, con el objetivo de llegar a todos los nodos de la red. Son transmitidos por *broadcast*, pero al ser un mensaje de control de OLSR, sólo lo reenviarán los nodos MPR del emisor y si no lo habían hecho previamente.

Cuando un nodo detecta que le llega un mensaje de control de OLSR por *broadcast*, comprueba si el nodo que ha enviado el mensaje, no el nodo fuente y originario de la información, sino el último salto, está en su lista de MPR. Si es así, entonces él volverá a enviar el mensaje de nuevo por *broadcast*; y por lo tanto, todos sus enlaces MPR que no hubiesen recibido el mensaje previamente volverán a enviarlo de nuevo, y así sucesivamente, extendiéndose por la red como una columna vertebral. De esta manera se consigue que la información llegue a todos los nodos de la red pero que sólo lo retransmitan unos pocos, evitando la saturación.

Todos los nodos de la red envían periódicamente y de forma asíncrona mensajes de tipo TC. Puede ocurrir que un nodo detecte un cambio en la topología y apresure el envío del paquete TC; asimismo, si detecta que la red no ha sufrido ningún cambio puede omitir enviarlo. Con estos paquetes que contienen, al menos, la tabla de los MPR seleccionados por el nodo fuente, los nodos generan una tercera tabla denominada topológica con información de toda la red, al igual que tienen una con los vecinos y otra con los nodos de dos saltos generadas con la información transmitida por los mensajes HELLO [25].

Gracias a las tablas creadas por los mensajes de control, los nodos pueden generar y mantener la tabla de rutas con todos los nodos de la red.

En la sección 10 de su RFC [7] podemos encontrar el algoritmo, basado en el de Dijkstra, para generar la tabla de rutas a partir de las tres anteriores tablas. Cuando alguna de las tres tablas es modificada porque se ha producido algún cambio en la red, la tabla de rutas se actualiza para añadir las modificaciones precisas y que todas las rutas se mantengan válidas. En la tabla de rutas encontraremos además de la dirección destino, a qué distancia se encuentra y cuál es el próximo salto.

Además de estos dos mensajes, en OLSR podemos encontrar otros dos.

Los Host and Network Association (HNA) que permite conectar la red OLSR con otras redes a través de nodos enlace. También están los Multiple Interfaces Declaration (MID) que permiten a un nodo tener varias interfaces de red inalámbrica funcionando para la misma red OLSR. Sin embargo el alcance de este documento no llega para estudiar a fondo estas tecnologías y otras que hemos dejado en el tintero, sino que busca dar una introducción al descubrimiento y manejo de las rutas de los protocolos principales. Pueden ahondar más si les interesa en el RFC de OLSR [7].

## 2.7. Ejemplos de redes mesh en funcionamiento

En este capítulo estudiaremos una serie de ejemplos significativos de diferentes redes mesh que funcionan actualmente alrededor del mundo. Nos interesa mostrar, sobre todo, que es posible desplegar redes mesh de forma autónoma, sin intervención de administraciones públicas ni empresas, reutilizando componentes, con tecnología relativamente barata y software de licencia libre. Por eso centraremos este punto en las redes inalámbricas comunitarias, aunque dedicaremos también una parte a explicar el modelo de red inalámbrica municipal.

### 2.7.1. Redes inalámbricas comunitarias.

Una red inalámbrica comunitaria es aquella puesta en marcha por un grupo de personas interesadas en las redes de ordenadores, la tecnología en general y en muchos casos en el software libre. Son iniciativas autofinanciadas y autogestionadas y la mayoría de las veces, abiertas. Es decir, son redes a las que cualquiera que posea el equipamiento necesario puede unirse. Un número elevado de ellas utilizan topología y protocolos mesh, aunque no es una condición imprescindible. En estas comunidades se comparten todo tipo de recursos, información y servicios, como por ejemplo:

- Sistemas de intercambio de archivos P2P.
- Repositorios de software.
- Servidores de correo.

- Servidores ftp.
- Servidores http.
- Sistemas de mensajería instantánea o chats.
- Juegos online multijugador.

Además, los usuarios de estas redes pueden decidir compartir su conexión a Internet con los demás usuarios de la red, aunque formar parte de una comunidad inalámbrica no implica necesariamente obtener acceso gratuito a Internet.

Veamos algunos ejemplos.

### Guifi.net

Guifi.net[26] es una red informática de telecomunicaciones libre, abierta y neutral, mayoritariamente inalámbrica. La red, está formada por particulares, empresas y otras organizaciones. A día de hoy comprende cerca de 18.400 km de enlaces, 18.243 nodos, de los cuales 12.065 están activos. Esto convierte a guifi.net en una de las mayores y más exitosas redes mesh, creada por una comunidad de voluntarios, del mundo entero. La mayoría de éstos nodos se encuentran ubicados en Catalunya, aunque se están abriendo nuevas zonas en el resto de la Península Ibérica.

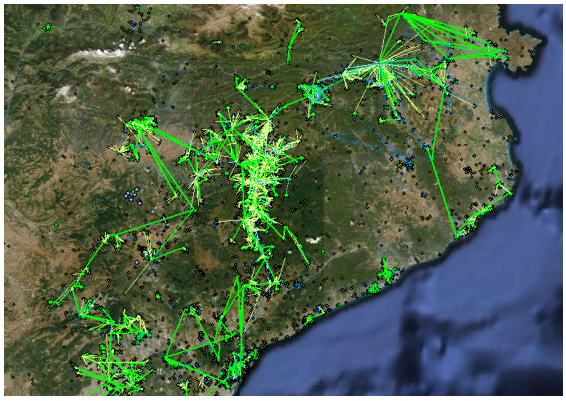

Figura 2.14: Vista parcial del despliegue de Guifi.net en Catalunya.

[26] `http://guifi.net/`

Los fundamentos básicos de la red se resumen en la Licencia Procomún de la Red Abierta, Libre y Neutral (Procomún de la XOLN)[27]:

- Eres libre de utilizar la red para cualquier propósito en tanto que no perjudiques al funcionamiento de la propia red o a la libertad de otros usuarios.
- Eres libre de saber cómo es la red, sus componentes y cómo funciona.
- Eres libre de hacer uso de la red para cualquier tipo de comunicación y difundir su funcionamiento.
- Incorporándote a la red ayudas a extender estas libertades en las mismas condiciones.

Guifi.net se creó en 2004 por un grupo de personas interesadas en las comunidades wireless. En mayo de ese año se configuraron los primeros enlaces permanentes y estables entre varias localidades de la comarca catalana de Osona, provincia de Barcelona. Desde entonces la red ha crecido a una gran velocidad, ha participado en numerosos proyectos para llevar conectividad wireless a zonas donde no existe acceso a Internet y ha recibido diversos premios en Catalunya y en Europa. En 2009 la Fundación Guifi.net se inscribe en el registro de operadores de telecomunicaciones de la Comisión del Mercado de las Telecomunicaciones. Esto permite a Guifi.net operar como un proveedor de acceso a Internet más, sin complicaciones legales.

La red se vertebra en dos grandes partes: la parte troncal y los puntos Guifi.net o nodos. La primera se encarga de intercomunicar las distintas zonas y la segunda es el conjunto de puntos que dan acceso a la red (puntos de acceso mesh), a los que se conectan los clientes o conexiones simples.

La parte troncal consta de conexiones inalámbricas punto a punto de larga distancia, que interconectan zonas lejanas entre sí. Dado su carácter de "columna vertebral", deben ser los enlaces más estables, gestionados con especial atención y ofrecer la máxima disponibilidad, ancho de banda y seguridad.

A los puntos troncales se conectan los diferentes puntos de acceso (generalmente routers mesh). Estos ofrecen conectividad a los distintos clientes (PC's, portátiles, PDA's, móviles, etc.). Cualquier usuario o grupo de ellos puede compartir con los demás una conexión a Internet.

[27] `http://www.guifi.net/es/ProcomunXOLN`

Guifi.net utiliza distintos protocolos: BGP (*Border Gateway Protocol*) para la parte troncal, OSPF (*Open Shortest Path First*) para zonas donde no hay frecuentes cambios de topología ni un alto número de nodos y OLSR y BMX (*BatMan eXperimental*) para zonas con cambios frecuentes de topología.

### Freifunk

Freifunk[28] es una iniciativa no comercial para fomentar la creación de redes inalámbricas libres en Alemania. El proyecto comenzó en 2003 y tiene como finalidad extender el conocimiento acerca del despliegue de redes abiertas inalámbricas y comunitarias, en colaboración con otros colectivos y personas interesadas en el tema, usando simpre software libre. Actualmente existen numerosas comunidades impulsadas por Freifunk en distintas ciudades alemanas y en otros puntos del planeta.

La comunidad Freifunk ha desarrollado su propio firmware para dispositivos wifi de la familia WRT54G, que permite que funcionen como puntos de acceso mesh. Utiliza OLSR y B.A.T.M.A.N. (del que son los principales desarrolladores) como protocolos de enrutamiento. Hoy en día la comunidad Freifunk sigue creciendo y participa activamente en una gran diversidad de proyectos como OLPC, OpenWRT e incluso el desarrollo de hardware mesh libre, como el Mesh Potato del proyecto Villagetelco[29].

### Nepal Wireless

Nepal Wireless[30] es una iniciativa personal del profesor nepalí Mahabir Pun. En 1997, la escuela donde trabajaba recibió varios ordenadores reciclados, donados por un colegio australiano. Entonces Mahabir decidió llevar Internet al pueblo para que la gente pudiese conectarse usando los ordenadores recién llegados. En un primer momento intentó conseguir que una línea telefónica (por radio o satélite) llegase hasta el pueblo, pero empresas y políticos lo ignoraron. Necesitaba encontrar una forma barata y casera de llevar Internet al pueblo.

Después de escribir una carta, explicando su situación y la de su pueblo,

[28] `http://start.freifunk.net/`
[29] `http://www.villagetelco.org/`
[30] `http://www.nepalwireless.net/`

a la BBC inglesa todo cambió. A través de esta carta pudo contactar con voluntarios que viajaron hasta Nepal para ayudarle con su cometido. Esto fue en 2002. Estos voluntarios fueron los primeros en hablarle de la tecnología Wifi. Las primeras pruebas se realizaron ese mismo año, consiguiendo conectar dos pueblos, Nangi y Ramche, que distan un kilómetro y medio.

En la actualidad, y después de varios años de esfuerzos y de intentar conseguir financiación, el proyecto ha conseguido conectar 42 pueblos rurales nepalís. Y está en desarrollo una nueva fase que pretende conectar otros 19 municipios.

En este ejemplo vemos con claridad el potencial que tiene la tecnología mesh para llevar Internet a zonas empobrecidas y de difícil acceso, e interconectar comunidades, reduciendo así la brecha digital y favoreciendo la democratización de la tecnología.

### CUWiN (the Champaign-Urbana Community Wireless Network)

CUWiN[31] es una organización formada por desarrolladores y voluntarios que trabaja para extender y favorecer el uso de tecnologías inalámbricas comunitarias en forma de redes malladas o mesh. Es una iniciativa del Urbana-Champaign Independent Media Center (IMC), entidad de carácter social y político que trabaja en el campo de la información "alternativa" e integrante de la red IMC mundial. Está localizada en la ciudad de Urbana, condado de Champaign, Illinois, EE.UU.

El proyecto empezó en el año 2000, a partir de un grupo de programadores, técnicos wireless y activistas sociales interesados en la creación de una red inalámbrica comunitaria en la ciudad de Urbana. Actualmente mantienen dicha comunidad wireless, además de desarrollar software para redes mesh y proporcionar servicios de consultoría y formación sobre la materia.

El trabajo de CUWiN en el desarrollo de nuevo software mesh ha sido bastante fructífero hasta la fecha, habiendo desarrollado su propio protocolo de enrutamiento, Hazy Sighted Link State (HSLS)[32].

[31] `http://www.cuwireless.net/`
[32] `http://www.cuwin.net/manual/techdocs/hsls`

### 2.7.2. Redes inalámbricas municipales.

Numerosas administraciones locales en diferentes países han puesto en marcha, en los últimos años, redes de comunicaciones inalámbricas para proporcionar diferentes recursos y servicios a la ciudadanía. Muchas de ellas utilizan redes mesh, por ser relativamente baratas y porque son la mejor opción (combinadas con otros tipos de tecnologías) a la hora de proporcionar conectividad en escenarios urbanos, en los que la topología cambia constantemente.

Las características y servicios ofrecidos son diversos y cambian según el caso. Hay algunas de pago, otras de acceso gratuito, algunas sólo ofrecen ciertos servicios y recursos vinculados con la administración, otras ofrecen acceso a Internet, en algunas ciudades sólo se ofrece la conexión en ciertos centros y edificios municipales y en otras se cubren barrios enteros.

En el caso del Estado Español, son pocos los ejemplos de redes wifi municipales promovidas por ayuntamientos. Muchos intentos de poner en marcha estas iniciativas ha chocado con las restricciones impuestas por la CMT (Comisión del Mercado de Telecomunicaciones). Dicha agencia estatal establece que las redes de este tipo no pueden ser financiadas con dinero público. Estas deben ser desplegadas y gestionadas por empresas privadas o el consistorio que quiera instalarlas debe registrarse como operador. Veamos, a modo de ejemplo, parte de la resolución de la CMT sobre la propuesta del *Institut Municipal d'Informàtica* (Barcelona), relativa a proporcionar acceso inalámbrico público a Internet en distintos puntos de la ciudad:

"*[...]PRIMERO.- El Institut Municipal d'Informàtica podrá prestar el servicio de acceso básico a Internet de forma gratuita durante un año, siempre que se respeten las condiciones señaladas en relación con dicho servicio en el texto de la presente Resolución. En particular:*

- *El servicio podrá prestarse únicamente en los horarios indicados de apertura de los centros del proyecto.*
- *La duración de la conexión estará limitada a sesenta minutos por usuario y día.*
- *La velocidad de transmisión deberá estar efectivamente limitada a un caudal de hasta –o alrededor de- 200 Kpbs.*
- *No se podrá acceder a actividades distintas del servicio de navegación*

*por Internet (Internet browsing), esto es, se impedirá el acceso a aplicaciones peer to peer para descarga de contenidos audiovisuales, telefonía IP (VoIP), videoconferencia u otras.*

- *Se impedirá el acceso a páginas web con contenidos para adultos o que puedan resultar nocivos para la juventud y la infancia.*

*[...]*"

En Marzo del año 2011 la CMT ha autorizado a las comunidades de vecinos a compartir la conexión a Internet entre todos los habitantes de un edificio[33], lo que constituye una gran noticia en la que sin duda una de las tecnologías que mejor podría adaptarse a este escenario sería una red mesh.

[33] `http://www.elpais.com/articulo/economia/vecinos/edificio/podran/compartir/wifi/comunitario/elpepueco/20110329elpepueco_4/Tes`

# Capítulo 3

# Descripción del sistema

En este capítulo pasamos a describir el sistema, tanto software como hardware, que se ha utilizado para implementar cada uno de los nodos mesh que integran la red.

## 3.1. Sistema utilizado

### 3.1.1. Hardware

Cada uno de los nodos que forman parte de la red experimental se compone de los siguientes elementos hardware:

- Placa base: PC Engines Alix 2c2. Especificaciones técnicas:
  - CPU: 500 MHz AMD Geode LX800.
  - DRAM: 256 MB DDR DRAM.
  - Almacenamiento: CompactFlash socket.
  - Alimentación: jack DC o POE pasivo (7V a 20V).
  - Tres LEDS frontales, botón de encendido.
  - Expansión: 2 slots miniPCI, bus LPC.
  - Conectividad: 2 adaptadores Ethernet Via VT6105M 10/100.
  - Entrada/salida: puerto serie DB9, dos puertos USB 2.0.

  - Medidas: 152.4 x 152.4 mm.
  - Firmware: tinyBIOS.

- Adaptador inalámbrico miniPCI: Atheros Communications Inc. AR5413 802.11abg.
- Unidad de almacenamiento: tarjeta compact flash de 4 GB.
- Entrada/salida: cable serie conectado al puerto del nodo y a un equipo externo. La mayoría de equipos de hoy en día no poseen puerto serie, por lo que se hace necesario también un adaptador serie-usb.

Figura 3.1: Vista frontal de uno de los nodos.

Figura 3.2: Vista cenital de uno de los nodos, sin la carcasa.

Figura 3.3: Vista trasera de uno de los nodos.

### 3.1.2. Software

Para el sistema operativo decidimos basarnos en una distribución Debian GNU/Linux a la que se le han añadido los paquetes de software y scripts necesarios para implementar funcionalidad mesh. El proceso que se ha seguido para generarlo y probarlo es el siguiente:

1. Mediante la herramienta *debootstrap*[1] generamos un sistema base Debian GNU/Linux en un directorio llamado chroot. Este directorio constituirá la partición del sistema.

2. Utilizando la aplicación *chroot* se instalan los paquetes software necesarios.

3. En otro directorio copiamos syslinux, para la gestión del arranque, el kernel y el initrd. Este directorio constituirá la segunda partición.

4. Generamos una imagen de disco con el sistema, a partir de los dos directorios citados anteriormente. La estructura de disco queda de esta manera:

```
# fdisk -l /dev/sdd

Disco /dev/sdd: 4009 MB, 4009549824 bytes
124 cabezas, 62 sectores/pista, 1018 cilindros
Unidades = cilindros de 7688 * 512 = 3936256 bytes
```

[1] `http://wiki.debian.org/Debootstrap`

```
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador de disco: 0xcce1380e

Dispositivo Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sdd1   *           1           9        32768    6   FAT16
/dev/sdd2               9           1019     3881784  83  Linux
```

5. Para probar el sistema utilizamos la herramienta de virtualización kvm[2].

6. Cuando necesitamos modificar archivos o instalar nuevas aplicaciones utilizamos chroot sobre el directorio del mismo nombre.

7. Para pasar el sistema a un nodo, copiamos la imagen a la tarjeta compact flash.

Veamos ahora en detalle el sistema operativo utilizado:

- Sistema base: Debian Sid.
- Kernel: 2.6.38-2-486.
- Gestor de arranque: Syslinux.
- Sistema de archivos: debido al número limitado de escrituras que soportan las tarjetas compact flash, hemos decidido configurar el sistema en dos modos, cada uno con distintos sistemas de ficheros:
  - Modo lectura: este es el modo habitual, el que usamos durante el funcionamiento normal de la red y durante las pruebas y que previene el deterioro de las tarjetas de memoria por escrituras continuadas. Utiliza la herramienta fsprotect[3] para montar un sistema de archivos aufs[4] y tmpfs[5]. Una vez apagado o reiniciado el sistema, todos los cambios realizados en el sistema de ficheros se pierden.
  - Modo lectura/escritura: utilizado para montar el sistema de ficheros en las ocasiones en las que se haga necesario hacer algún cambio permanente. Monta la partición del sistema en ext2[6].

[2] http://www.linux-kvm.org/page/Main_Page
[3] http://freshmeat.net/projects/fsprotect
[4] http://aufs.sourceforge.net/
[5] http://lxr.linux.no/#linux+v2.6.39/Documentation/filesystems/tmpfs.txt
[6] http://e2fsprogs.sourceforge.net/ext2.html

- Paquetes de software adicionales instalados:
  - Herramientas de red: iw, iproute, conntrack, conntrackd, wireless-tools, arptables, httptermd, http-client-benchmark.
  - Sistema de ficheros: fsprotect, aufs-tools.
  - Protocolos de enrutamiento mesh: batmand, babeld, meshias.
  - Otros: screen, pciutils, bash-completion, vim.

Además de los anteriores paquetes de software se ha hecho necesario implementar distintos scripts de shell para diferentes tareas:

- change-ips.sh (Ver 5.2): cambia las direcciones IP en el fichero /etc/network/interfaces.
- chroot-fsprotect.sh (Ver 5.2): remonta el sistema de archivos en lectura/escritura. Utilizado para realizar cambios permanentes estando en modo lectura.
- chroot-script.sh (Ver 5.2): utilizado para realizar chroot en la carpeta del sistema, montando y desmontando los directorios necesarios (/dev, /sys y /proc). Necesario para instalar paquetes de software antes de generar la imagen de disco.
- compress.sh (Ver 5.2): comprime el directorio del sistema, respetando los permisos y propietarios.
- create-disk.sh (Ver 5.2): genera la imagen de disco del sistema, creando las dos particiones y copiando los directorios chroot y syslinux.
- create-disk-cf.sh (Ver 5.2): análogo al anterior pero en vez de generar una imagen de disco, la copia en una tarjeta compact flash.
- extract.sh (Ver 5.2): descomprime el directorio chroot respetando permisos y propietarios.
- pull-chroot.sh (Ver 5.2): descarga el directorio chroot comprimido desde el repositorio svn utilizado durante el proyecto. Utiliza la herramienta rsync.
- push-chroot.sh (Ver 5.2): análogo al anterior pero para subir el chroot comprimido.

### 3.1.3. Entorno de despliegue de la red y topologías modeladas.

La red ha sido desplegada en un piso amplio, de 150 $m^2$. Se han instalado cuatro nodos repartidos por las distintas estancias de la casa. Dada la potencia de las antenas, los nodos estaban todos dentro del rango de cobertura inalámbrica de los demás, por lo que se ha hecho necesario utilizar herramientas como iptables para modelar la topología deseada.

- Topología 1: en este modelo todos los nodos tienen enlaces directos entre ellos menos el 5 y el 7, que deben elegir entre pasar por el 3 o el 6 para comunicarse. Con esta topología hemos puesto a prueba el tiempo de convergencia de la red ante cambios como la caída de un nodo.

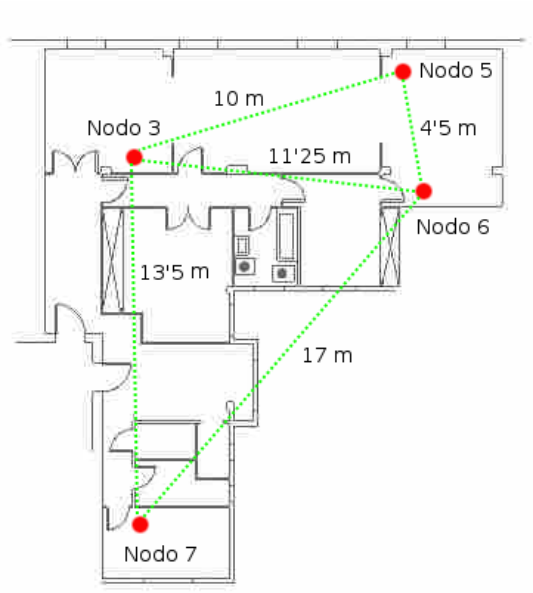


Figura 3.4: Topología para probar el cálculo de rutas.

- Topología 2: es un modelo lineal o en cadena. Necesario para conseguir rutas de tres saltos con los cuatro nodos disponibles.

Hay que destacar los inconvenientes propios de este tipo de despliegue. En un principio, la intención era desplegar la red en un entorno más ajustado a la realidad, en un espacio urbano, con los nodos separados de manera que

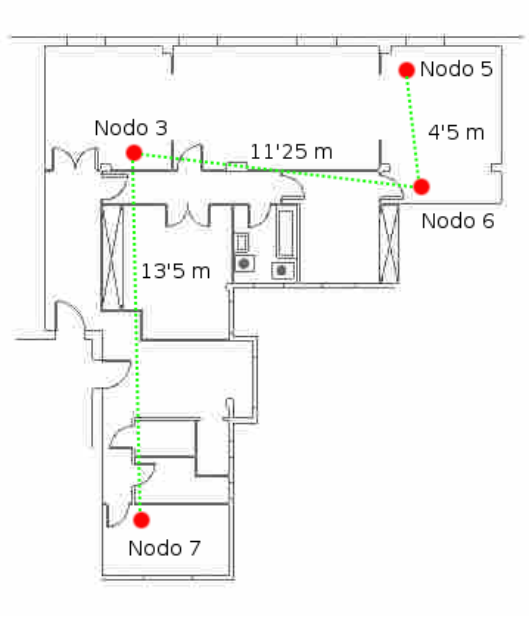


Figura 3.5: Topología para probar el throughput en base al número de saltos.

no todos tuvieran conexión inalámbrica directa con los demás. Pero, ante las dificultades técnicas que esta opción presentaba, finalmente se optó por montar una red mesh experimental en una vivienda que, aunque amplia, no permite separar un par de nodos lo suficiente como para evitar que caigan en el rango de cobertura inalámbrica el uno del otro.

Este hecho presenta un problema. Si todos los nodos comparten el mismo medio, el mismo canal de transmisión, sólo uno de ellos podrá transmitir información en un momento determinado, ya que de otra manera se producirían interferencias y corrupción de datos. Por lo tanto, en un entorno de estas características, en un momento en el que todos los nodos tengan datos que enviar, el *throughput* observado será igual al *throughput* conseguido entre dos nodos vecinos dividido por el número total de nodos menos uno. No siempre nos encontraremos con una situación en la que todos los nodos necesitan enviar datos a la vez, pero el tráfico de control de los protocolos, sobre todo los proactivos, también empeora este fenómeno.

Las tarjetas inalámbricas utilizadas tienen una capacidad de ancho de banda de 54 Mbps. Este es un parámetro ideal, pero, realmente, nunca suelen sobrepasar los 22 Mbps, debido a la inestabilidad de las comunicaciones inalámbricas, las interferencias de otras redes dentro del alcance y el *overhead* que producen los protocolos de red.

Para comprobar el *throughput* máximo que podemos alcanzar hemos medido la velocidad de transmisión entre dos nodos, con los demás apagados. En el modo ad-hoc, sin ningún protocolo de enrutamiento funcionando, obtenemos una velocidad máxima de 18 Mbps. Y en el modo mesh de Open80211s de 12'56 Mbps. Las pruebas se han realizado usando el canal 11, por ser el menos usado por las redes inalámbricas del entorno.

Además de estos factores, existen otros como *drivers* que no aprovechan todo el potencial de las interfaces de red, la configuración por defecto de dichas interfaces y algunos parámetros del núcleo linux que pueden afectar al rendimiento de la red. Podemos configurar algunos de estos parámetros para intentar subir el *throughput*. O por lo menos asegurarnos de que no son los causantes del bajo rendimiento. Uno de los problemas causados por una mala configuración de estas variables es el conocido como *bufferbloat*[28]. Consiste en la aparición de latencias altas, variaciones en la latencia de transmisión para paquetes del mismo tamaño (*jitter*) y reducción del *throughput* debido a un *buffering* excesivo de los paquetes de datos antes de ser enviados. En la mayoría de *routers* comerciales este parámetro no es configurable, pero en nuestro sistema sí podemos modificar el valor del parámetro *txqueuelen* para evitar el *bufferbloat*. Este parámetro define la longitud de la cola de transmisión o *buffer*. Por defecto tiene un valor de 1000 paquetes. Es decir, no se empezará a transmitir hasta que no se hayan almacenado 1000 paquetes de datos en la cola. Para cambiar este valor podemos usar el comando *ifconfig*:

```
# ifconfig <dispositivo> txqueuelen <valor>
```

En nuestro sistema hemos utilizado un valor de 10 para *txqueuelen*. Aun así, no nos aseguramos la desaparición del *bufferbloat*, ya que también existen buffers en el *driver* de la interfaz inalámbrica y en el propio hardware.

Además, se ha utilizado un pequeño *script*, desarrollado por József Kadlecsik (Netfilter Core Team) y György Pásztor que modifica algunos parámetros del kernel de Linux para optimizar el rendimiento de la red (Ver 5.2).

A pesar de todo esto, las pruebas realizadas nos han ayudado a llegar a algunas conclusiones interesantes que pueden ser extrapolables a entornos reales.

# Capítulo 4

# Pruebas, resultados y evaluación.

En este capítulo describiremos las pruebas que se han llevado a cabo sobre la red, las características que queríamos analizar con ellas, los resultados obtenidos, comparando unos protocolos de enrutamiento con otros, y las conclusiones a las que hemos llegado.

## 4.1. Parámetros a evaluar.

Los parámetros que hemos considerado más significativos para una correcta evaluación y comparación de los distintos protocolos de enrutamiento son los siguientes:

- RTT (*Round-Trip Time*): la traducción al castellano sería tiempo de ida y vuelta. Constituye el tiempo en el que un paquete de datos es enviado desde un nodo, llega al destino y recorre el camino inverso hasta llegar de nuevo al emisor. A partir de este parámetro podemos medir la rapidez con que cada protocolo es capaz de enrutar un paquete, decidiendo el camino adecuado entre todos los posibles. Se ha comparado este parámetro en función del número de saltos que debe atravesar la información, de 1 a 3 saltos. Para ello hemos utilizado la aplicación *ping*, que se adapta perfectamente a este propósito.
- Sesiones por segundo y *throughput*: la intención es medir el aprove-

chamiento real del ancho de banda (*throughput*) frente al número de sesiones simultáneas haciendo peticiones a un nodo determinado, confrontándolos a su vez con el número de saltos realizados y el tamaño del objeto requerido. Hemos determinado, de esta manera, cómo influye el tráfico de control y enrutamiento en la degeneración de la eficiencia de cada protocolo. Para ello hemos utilizado las aplicaciones *httptermd* y *http-client-benchmark*.

- Tiempo de reconfiguración de rutas frente a la caída de un nodo: uno de los puntos fundamentales de los protocolos de enrutamiento mesh es la capacidad de reacción ante cambios en la topología, ya sea porque se añaden nuevos nodos a la red o porque alguno de ellos deja de estar operativo. En esta prueba hemos medido el tiempo que emplea cada protocolo en detectar, buscar y activar una nueva ruta ante la caída de un nodo que formaba parte de una ruta activa.

## 4.2. Pruebas realizadas.

Veamos de forma detallada las pruebas realizadas, encaminadas a medir cada uno de los parámetros descritos en el apartado anterior. Para las dos primeras se han utilizado los parámetros por defecto de todos los protocolos. En la última se han modificado algunos de estos parámetros, como se explica más adelante.

- RTT: la prueba consiste en evaluar el tiempo en que un paquete de datos es enviado al destino y es devuelto al origen por el primero. Para ello utilizamos el programa *ping*. Dicha aplicación utiliza el protocolo ICMP. El emisor envía un datagrama ECHO_REQUEST y el destino responde con un ECHO_RESPONSE. El tiempo empleado en todo el proceso constituye el RTT. El datagrama tiene un tamaño total de 64 bytes (56 de datos más 8 de la cabecera ICMP). Este parámetro se ha medido para uno, dos y tres saltos de enlaces entre vecinos, enviando un total de 50 datagramas en cada prueba. Para realizarlas se ha modelado una topología lineal, forzando que cada nodo esté conectado por un enlace de un salto sólo con el nodo siguiente, en los extremos, y con el siguiente y el anterior en el caso de los nodos intermedios. Para conseguirlo, hemos utilizado las siguientes herramientas según el protocolo bajo estudio:

- Open80211s: al ser un protocolo de nivel 2, no podemos utilizar filtros que trabajen con el protocolo IP para modelar la topología. Se ha utilizado el software *iw* y concretamente la opción *station* para bloquear enlaces punto a punto:

  ```
  # iw dev <dispositivo> station set <dirección_mac> \
  <plink_action> block
  ```

  De esta manera bloqueamos el enlace directo entre el nodo en el que ejecutamos el comando y el identificado por *dirección_mac*.

- B.A.T.M.A.N. y Meshias: hemos empleado la herramienta *iptables* para bloquear los paquetes de los nodos deseados, identificándolos por su dirección MAC. Al ser un protocolo de nivel 3, conseguimos bloquear el tráfico IP:

  ```
  # iptables -A INPUT -m mac --mac-source <dirección_mac> \
  -j DROP
  ```

- Babel: Babel es un protocolo híbrido, trabaja con IPv4 e IPv6. Por defecto identifica a los nodos por su dirección de IPv6, por lo que hemos utilizado la aplicación ip6tables:

  ```
  # ip6tables -A INPUT -s <dirección_IPv6> -j DROP
  ```

Finalmente, ejecutamos el comando ping para medir el RTT para uno, dos y tres saltos. Dicho comando nos da, al final, el RTT mínimo, máximo, medio y la desviación de la media:

```
$ ping -c 50 <dirección_ip>
```

- Sesiones por segundo y *throughput*: para esta prueba se ha vuelto a utilizar la misma topología que en la anterior, ya que también calcula en función del número de saltos inalámbricos. Una vez configurada la topología hemos utilizado dos herramientas instaladas en cada nodo:

  - HTTPTerm: aplicación diseñada por Willy Tarreau[1]. Se trata, básicamente, de un servidor de objetos *http*. Se puede configurar para que sirva objetos de distinto tamaño dependiendo del puerto al que se realiza la petición. Nosotros hemos utilizado los siguientes:
    - Puerto 8000: devuelve un objeto con código de respuesta 304, es decir, es un objeto de tamaño 0, sin contar las cabeceras http.

[1] `http://1wt.eu/`

  - Puerto 8001: objeto de 64 bytes.
  - Puerto 8002: objeto de 512 bytes.
  - Puerto 8003: objeto de 1024 bytes.

- Http-client-benchmark: herramienta desarrollada por Pablo Neira Ayuso[2], tutor de este proyecto. Se trata de un simple cliente que realiza peticiones http a una IP. Puede simular un número concurrente de sesiones por segundo y devuelve como salida:
  - Número de peticiones HTTP (total acumulado).
  - Número de peticiones HTTP por segundo (actual y media acumulada).
  - Throughput en KB/s.
  - Número de errores en el establecimiento de la comunicación TCP.
  - Número de respuestas de error TCP RST.
  - Número de peticiones HTTP descartadas por *timeout*.

  El comando utilizado es el siguiente:

```
$ client -t <timeout_http> -u <num_usuarios> \
-G <dirección_ip>:<puerto>
```

  Para cada salto (de uno a tres), hemos realizado peticiones para 20 usuarios concurrentes. Para esta cantidad de usuarios hemos hecho pruebas con objetos HTTP de 0, 1024, 5120, 51200 y 10240000 bytes (puertos 8000, 8003, 8005, 8008 y 8011, respectivamente).

- Reconfiguración de rutas frente a la caída de un nodo: como hemos explicado antes, se ha medido el tiempo en que cada protocolo recalcula su tabla de rutas cuando un nodo deja de estar operativo repentinamente. Para ello hemos modelado, usando las herramientas descritas en la prueba que mide el RTT, la topología descrita en 3.4.

  La implementación práctica de la prueba consiste en realizar una llamada ping de un extremo al otro de la ruta activa (de dos saltos inalámbricos de longitud) y en un momento determinado desconectar el nodo intermedio de dicha ruta. Se han empleado 50 datagramas y la caída del nodo se produce en el segundo 15 una vez iniciada la llamada a ping, que tiene el siguiente formato:

[2] `http://1984.lsi.us.es/~pablo/index.html`

```
$ ping -c 50 -T tsandaddr <dirección_ip>
```

Con la opción *-T tsandaddr* pedimos al comando *ping* que nos muestre el camino recorrido en cada petición, además de un registro temporal.

Con la salida de este comando podemos ver cuántos paquetes se han perdido y cuánto tiempo ha tardado en restablecerse la comunicación.

## 4.3. Resultados y comparativa.

Pasamos a exponer los resultados obtenidos en cada prueba para cada uno de los protocolos estudiados.

### 4.3.1. RTT.

| Saltos | Open80211s | | | | Babel | | | | B.A.T.M.A.N. | | | | Meshias | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Mín. (ms) | Máx. (ms) | Media (ms) | % paquetes perdidos | Mín. (ms) | Máx. (ms) | Media (ms) | % paquetes perdidos | Mín. (ms) | Máx. (ms) | Media (ms) | % paquetes perdidos | Mín. (ms) | Máx. (ms) | Media (ms) | % paquetes perdidos |
| **1** | 0.419 | 9.291 | 2.356 | 0 | **0.346** | **2.731** | **0.615** | 0 | 0.355 | 3.475 | 0.944 | 0 | 0.780 | 12.235 | 2.506 | 0 |
| **2** | 0.718 | 13.868 | 3.093 | 0 | **0.614** | **2.932** | **0.975** | 0 | 0.668 | 21.362 | 2.758 | 0 | 1.358 | 12.401 | 3.844 | 0 |
| **3** | 1.047 | 29.548 | 4.639 | 0 | **0.978** | **27.160** | **2.331** | 0 | 1.070 | 37.723 | 8.870 | 0 | 1.915 | 14.188 | 5.710 | 0 |

Figura 4.1: Resultados para el RTT. En negrita, los resultados óptimos.

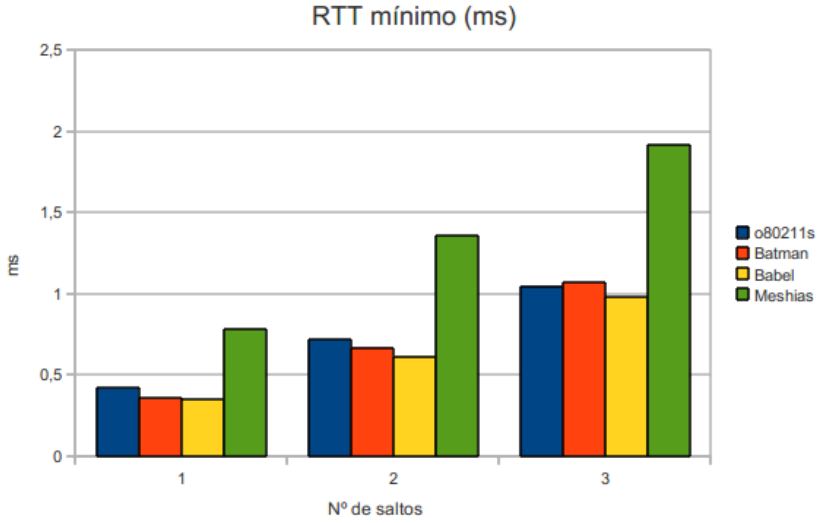


Figura 4.2: RTT mínimos.

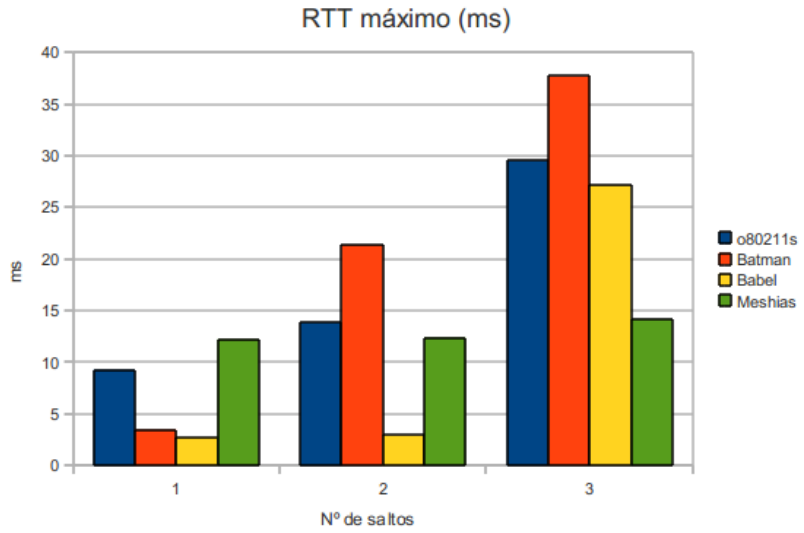


Figura 4.3: RTT máximos.

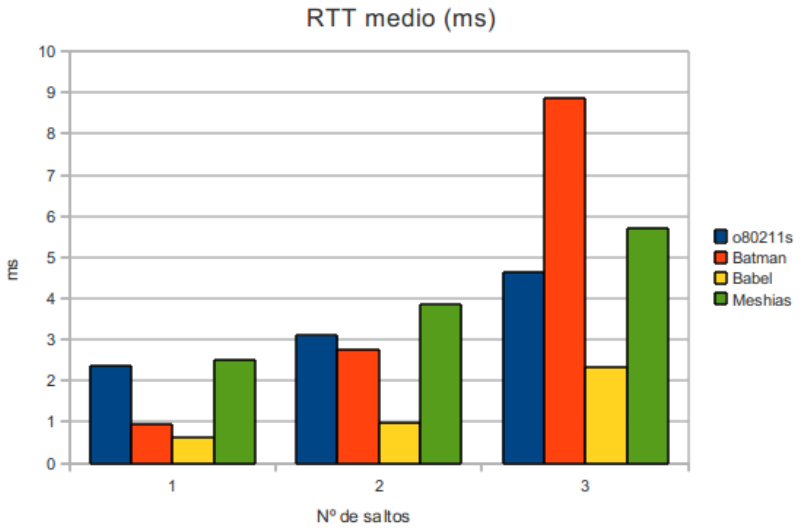


Figura 4.4: RTT medios.

Como se puede observar, Babel es el protocolo que ofrece unos mejores resultados para esta prueba. Mientras que para el RTT mínimo no se observan grandes diferencias entre protocolos (excepto Meshias), sí que las encontramos para los RTT máximos y medios. La uniformidad de resultados en el RTT mínimo se debe a que en una situación en la que el canal no está saturado y las rutas están calculadas todos los protocolos alcanzan la velocidad máxima disponible, sin diferencias significativas. En los RTT máximos y medios, los diferentes enfoques a la hora de implementar el enrutamiento mesh hacen visibles mayores diferencias, sobre todo entre Babel y B.A.T.M.A.N. Babel es el que ofrece mejores resultados. Una posible explicación es el intervalo de tiempo en el que los protocolos envían mensajes de descubrimiento y actualización de vecinos y rutas. Por defecto, Babel envía un mensaje *Hello* (de 8 bytes) cada 4 s. Además, cada 16 segundos un nodo envía una actualización de su tabla de rutas a los demás. En cambio, B.A.T.M.A.N. envía un mensaje OGM (12 bytes) cada segundo. Por lo tanto, éste último protocolo genera más tráfico de control que el primero, afectando así a la velocidad del tráfico. Aunque el tamaño de los paquetes de control es pequeño, a medida que aumenta el número de saltos afecta al RTT, que en cualquier caso no supera en ningún caso los 40 ms.

### 4.3.2. Sesiones por segundo y *throughput*.

Veamos los resultados obtenidos con respecto al *throughput* y las sesiones HTTP atendidas por segundo. Para cada protocolo se han medido estos valores para uno, dos y tres saltos, y para objetos HTTP de distinto tamaño:

- Objeto 1: 0 bytes (sin contar las cabeceras HTTP).
- Objeto 2: 1024 bytes.
- Objeto 3: 5120 bytes.
- Objeto 4: 51200 bytes.
- Objeto 5: 10240000 bytes.

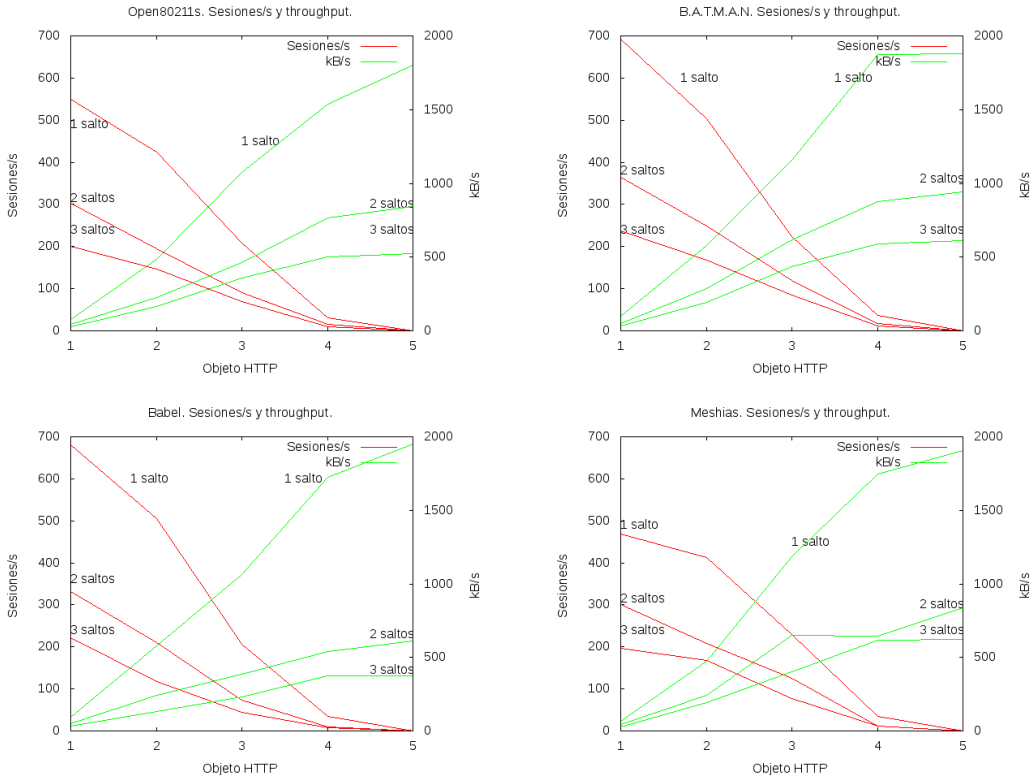


Figura 4.5: *Throughput* y sesiones por segundo frente al número de saltos y tamaño del objeto HTTP.

Como se observa, los resultados son muy parecidos pra todos los protocolos. Los valores representados son la media de los resultados obtenidos después de realizar las pruebas varias veces para cada caso. El máximo *throughput* medio conseguido es de casi 2000 kB/s, lo que supone un aprovechamiento del ancho de banda del 86 %, aunque en alguna de las pruebas se ha conseguido llegar a picos de 2200 kB/s para un objeto de 10240000 y un salto inalámbrico, en el protocolo B.A.T.M.A.N. Como hemos comentado anteriormente, el máximo *throughput* alcanzado en el modo ad-hoc (sin protocolos de enrutamiento) con solo dos nodos encendidos era de 18 Mbps, que equivalen a 2304 kB/s. Por lo tanto, en determinados momentos se ha llegado a un aprovechamiento del ancho de banda del 95 %.

Si prestamos atención a las gráficas vemos que el *throughput* crece de manera inversa al número de sesiones por segundo a medida que el tamaño del objeto aumenta, para todos los protocolos. Es decir, al hacer peticiones de objetos pequeños podemos atender muchas sesiones a la vez, pero el aprovechamiento del canal es bajo. Por contra, con objetos de tamaño grande, seremos capaces de conseguir un *throughput* alto, pero las sesiones atendidas por segundo irán disminuyendo.

Otro fenómeno que se observa es la disminución del *throughput* a medida que aumenta el número de saltos. Esto es debido a que todos los nodos comparten el mismo medio inalámbrico, como se explicó en el capítulo anterior. Cuando hacemos pruebas de un salto inalámbrico, sólo dos nodos envían y reciben datos, por lo que podemos acercarnos al *throughput* máximo, aunque el tráfico de control de enrutamiento generado por los otros nodos sigue afectando pero de manera poco significativa. Pero una vez que los datos deben de atravesar más de un salto el *throughput* empieza a disminuir debido a que sólo un nodo puede transmitir datos a la vez. Por lo tanto, para dos saltos (tres nodos implicados) obtendremos aproximadamente la mitad del *throughput*. En general, para N nodos obtendremos el máximo *throughput* en una comunicación de un salto dividido entre N-1.

- *Throughput* máximo con 2 nodos (1 salto): 2000 kB/s.
- *Throughput* máximo con 3 nodos (2 saltos): 2000 kB/s / 2 = 1000 kB/s.
- *Throughput* máximo con 4 nodos (3 saltos): 2000 kB/s / 3 = 666'67 kB/s.

Se pueden comprobar estos cálculos, de manera aproximada, en las gráficas presentadas.

### 4.3.3. Reconfiguración de rutas frente a la caída de un nodo.

Para esta prueba, al contario que en las anteriores, hemos modificado la configuración por defecto de los protocolos Babel y B.A.T.M.A.N. para adaptarlos a una situación en la que la red mesh tiene un alto grado de movilidad, anteponiendo un rápido tiempo de recuperación al overhead generado por los mensajes proactivos de descubrimiento de nodos y actualización de la topología. En cuanto a Open80211s no hemos modificado la configuración ya que se trata de un protocolo reactivo. Meshias ha sido excluido de esta prueba ya que se demostró la existencia de un *bug* que impedía la reconfiguración de las rutas en un tiempo razonable. Los parámetros modificados para Babel y B.A.T.M.A.N. son:

- Babel: este protocolo envía, por defecto, un mensaje *Hello* cada 4 segundos y un *dump* completo de su tabla de rutas cada 16 segundos. Hemos reducido estas variables a la mitad, quedando en 2 y 8 segundos, respectivamente.
- B.A.T.M.A.N: el intervalo temporal por defecto para el envío de OGM's es de un segundo. Hemos reducido este tiempo en la misma proporción que Babel, quedando en 500 ms.

Como vemos en la figura 4.6 Open80211s tiene el tiempo de recuperación más rápido, perdiendo la comunicación sólo durante un segundo. B.A.T.M.A.N. tarda 12 segundos en encontrar la nueva ruta y Babel 7 segundos. Teniendo en cuenta que sólo existía otra ruta posible y que la red es de un tamaño muy reducido, parecen tiempos altos en el caso de estos dos últimos protocolos. La diferencia entre ellos y Open80211s puede estribar en que éste trabaja en espacio del kernel, más rápido, y en nivel de enlace de datos, que suele tener latencias más bajas que la capa 3.

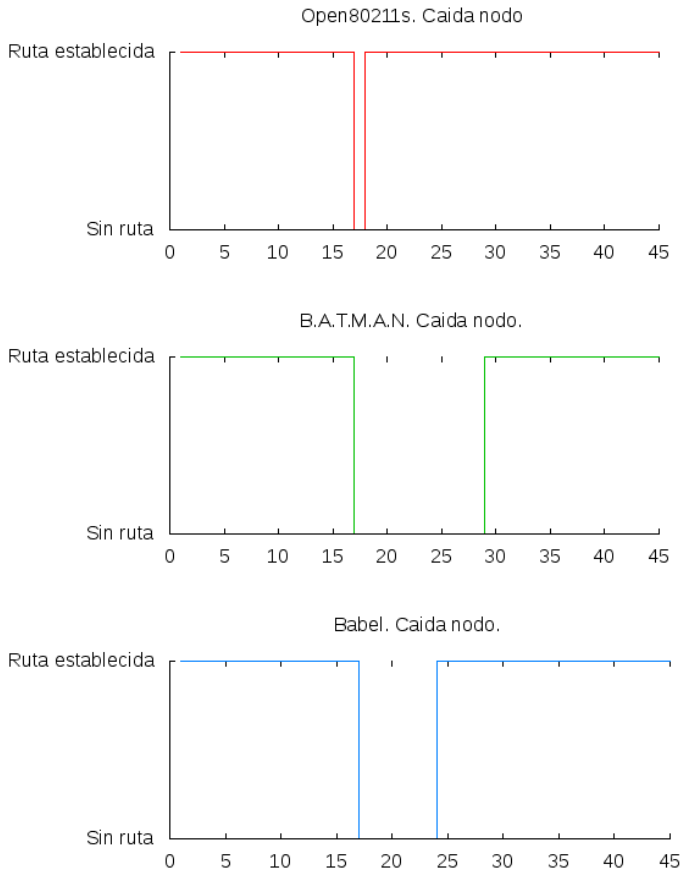


Figura 4.6: Tiempo de respuesta ante la caída de un nodo.

# Capítulo 5

# Conclusiones y trabajos futuros.

## 5.1. Conclusiones.

A raíz del estudio, experimentación y análisis realizados de los distintos protocolos, estudiados de forma teórica y experimental, podemos extraer las siguientes conclusiones:

- El campo de las redes inalámbricas mesh es relativamente nuevo y, por lo tanto, queda todavía mucho camino por recorrer. No olvidemos que todavía no existe un estándar internacional, ya que el IEEE 802.11s se encuentra en fase de borrador. Lo que queremos expresar es que los desarrollos que encontramos en la actualidad tiene carácter experimental en su mayoría y existen pocas redes mesh en producción. Los protocolos de enrutamiento existentes todavía no están maduros y faltan aspectos por implementar, sobre todo relativos a seguridad y calidad de servicio. En concreto, en lo que a software libre y sistemas GNU/Linux se refiere, existen pocos drivers para interfaces inalámbricas que soporten el modo mesh o incluso el ad-hoc. Y, aunque los soporten, muchas veces ofrecen un pobre rendimiento.
- Trabajar con redes inalámbricas es una tarea compleja. La saturación del espectro radioeléctrico, la proliferación de redes wifi en los últimos años y las interferencias que se producen entre ellas hace difícil disponer de entornos donde realizar pruebas experimentales sin que dichos fac-

tores influyan de manera significativa. Además, cuando aparecen fallos y comportamientos inesperados, es complicado discernir si son debidos a las citadas interferencias o a problemas de diseño e implementación.

- Las redes inalámbricas mesh tienen un enorme potencial. Como ya hemos dicho en capítulos anteriores, la relativa facilidad con la que es posible extender el rango de cobertura de estas redes añadiendo más nodos, y el hecho de que sean compatibles con las tecnologías inalámbricas ya existentes hacen que tengan mucho que aportar en el futuro. Ya sea para hacer llegar la conexión a Internet a lugares donde no llega el ADSL comercial o para compartir y abaratar las conexiones ya existentes en zonas urbanas. El único impedimento para un fuerte desarrollo de esta tecnología lo constituyen las actuales operadoras de acceso a Internet (y las autoridades reguladoras), reacias a permitir despliegues de redes mesh que podrían hacerles perder clientes y beneficios. Pero, como era de esperar, la sociedad ya ha cogido ventaja y hoy en día tenemos ejemplos de comunidades de usuarios que han implementado redes de este tipo. La asociación de estas comunidades de usuarios y desarrolladores, con la comunidad del software libre ya está dando grandes resultados y avances.

- En cuanto a la parte técnica y de evaluación de los protocolos, no podemos decir que hayamos llegado a la conclusión de que uno de los que hemos estudiado sea claramente superior a los demás. El rendimiento ofrecido es bastante parecido, aunque Babel y B.A.T.M.A.N. destacan por encima de los otros dos. Veamos, de forma concreta, a qué conclusiones podemos llegar después de estudiar y probar cada uno de ellos:

  - Open80211s: aunque no es un producto acabado, ya que faltan por implementar partes del borrador del IEEE 802.11s (el desarrollo de la funcionalidad de Mesh Portal), es una implementación completamente funcional y que ya se usa, por ejemplo, en los equipos del OLPC, además de estar incluida en el núcleo Linux. Es sencilla de usar, si se posee una tarjeta inalámbrica que esté soportada. En cuanto al rendimiento, ha mostrado el menor tiempo de recuperación antes fallos en rutas activas, bastante por delante de los demás protocolos. Tiene la desventaja de funcionar a nivel de enlace de red, lo que lo hace poco escalable.
  - B.A.T.M.A.N: el planteamiento teórico de este protocolo es interesante por su simplicidad, basado en intercambiar solamente la cantidad de información estrictamente necesaria para que todos

los nodos sean capaces de enviar, por la mejor ruta, la información al destino. Los paquetes de control (OGM) son de tamaño muy reducido, favoreciendo un bajo *overhead*, aunque el valor por defecto para el intervalo entre el envío de uno y el siguiente (un segundo) puede ser demasiado bajo en determinadas situaciones. En la prueba del RTT mostró grandes variaciones entre los valores mínimo y máximo. Ante cambios en la topología fue el que ofreció peores resultados.

- Babel: la principal característica de Babel es que evita la formación de bucles sin fin en el cálculo de rutas, así como su robustez ante cambios en la topología. En su contra debemos decir que el tiempo inicial de convergencia se ha mostrado bastante alto. En la prueba de recuperación ante la caída de un nodo sí obtuvimos buenos resultados, ofreciendo un rápido cambio de ruta.

- Meshias: esta implementación de AODV es la que ha mostrado peor rendimiento, aunque en los resultados para el RTT máximo ofrece buenos tiempos, así como en el cálculo del *throughput*, en el que está prácticamente al mismo nivel que los demás. Es un protocolo en versión alfa y, por lo tanto, eran de esperar problemas de este tipo. En la prueba de recuperación ante la caída de un nodo no fue capaz de recalcular una ruta válida.

En definitiva, podemos decir que para cada situación concreta en la que se quiera desplegar una red mesh será más conveniente utilizar uno u otro protocolo. Aunque volvemos a decir que B.A.T.M.A.N. y Babel están un escalón por encima de los otros dos. A partir de los resultados obtenidos, recomendaríamos usar Babel para redes con un alto grado de movilidad y cambios en la topología y B.A.T.M.A.N. para redes más estáticas en las que es más importante la estabilidad y la rapidez en las comunicaciones.

- Finalmente, destacar que durante el proyecto hemos tenido que enfrentarnos a numerosas dificultades que nos han impedido realizar un estudio experimental en un entorno real. Aun así, creemos que tan sólo el hecho de haber conseguido realizar el despliegue de la red y las pruebas de evaluación constituyen un éxito en sí mismos y un proceso personal de aprendizaje muy importante.

## 5.2. Trabajos futuros.

El siguiente paso después de realizar este trabajo es el despliegue de la red en un entorno urbano real, en el que se puedan realizar estas y otras pruebas para recoger resultados que nos ayuden a decidir cuál de los protocolos ofrece un mejor rendimiento. Para ello será necesario adquirir antenas con una potencia adecuada, encontrar emplazamientos que ofrezcan una buena localización en cuanto a altura, campo de visión y suministro eléctrico. También sería interesante añadir más protocolos, como por ejemplo Batman-adv, y completar el estudio de todas las funcionalidades que ofrecen los incluidos en este trabajo, como la interconexión de redes, por ejemplo.

Una vez desplegada la red se podrían añadir dispositivos que ofrezcan conexión a Internet y así probar el rendimiento y el enrutamiento de datos desde dentro hacia fuera de la red.

Una vez completados estos pasos, la idea sería generar una comunidad de usuarios y desarrolladores alrededor de la red, y que poco a poco vaya convirtiéndose en una red en producción, que sea de utilidad para la comunidad. Para ello se hará necesario profundizar en los conocimientos sobre conectividad inalámbrica y mesh, así como desarrollar un sistema estable, con los scripts y drivers necesarios para gestionar la red de manera sencilla.

Un úlitmo paso podría ser, si obtenemos buenos resultados, comercializar la solución desarrollada para ofrecer pequeñas y medianas redes mesh a comunidades de vecinos o pequeñas localidades. Para ello será necesario resolver cuestiones legales que no nos impidan ofrecer dicho servicio.

# Anexo.

Incluimos en este capítulo de Anexo el código de los distintos scripts usados y desarrollados para la generación y gestión del sistema utilizado.

### change-ips.sh

```
#!/bin/bash

if [ $(id -u) != "0"  ];
then
        echo "Debe tener privilegios de superusuario para ejecutar este script."
 1>&2
        exit 1
fi

if [ $# -lt 1 ];
then
        echo "No se ha especificado ningún parámetro!"
        exit 1
fi

mount -oremount,rw /fsprotect/system

mount --bind /proc /fsprotect/system/proc
mount --bind /sys /fsprotect/system/sys
mount --bind /dev /fsprotect/system/dev
mount --bind /dev/pts /fsprotect/system/dev/pts

chroot /fsprotect/system /bin/bash -c "
```

```
sed -i -r 's/address *10\.0\.0\.[0-9]+/address 10\.0\.0\.$1/g' \
/etc/network/interfaces;
sed -i -r 's/address *10\.0\.[0-9]+\.1/address 10\.0\.$1\.1/g' \
/etc/network/interfaces;
sed -i -r 's/broadcast *10\.0\.[1-9]+[0-9]*\.255/broadcast \
10\.0\.$1\.255/g' /etc/network/interfaces;
echo Nodo-$1 > /etc/hostname;
exit
"

umount /fsprotect/system/proc
umount /fsprotect/system/sys
umount -l /fsprotect/system/dev
umount /fsprotect/system/dev/pts

mount -oremount,ro /fsprotect/system

shutdown -r now
```

### chroot-fsprotect.sh

```
#!/bin/bash

mount -oremount,rw /fsprotect/system

mount --bind /proc /fsprotect/system/proc
mount --bind /sys /fsprotect/system/sys
mount --bind /dev /fsprotect/system/dev
mount --bind /dev/pts /fsprotect/system/dev/pts

chroot /fsprotect/system /bin/bash

umount /fsprotect/system/proc
umount /fsprotect/system/sys
umount -l /fsprotect/system/dev
umount /fsprotect/system/dev/pts

mount -oremount,ro /fsprotect/system
```

**chroot-script.sh**

```
#!/bin/bash

mount --bind /proc chroot/proc
mount --bind /sys chroot/sys
mount --bind /dev chroot/dev
mount --bind /dev/pts chroot/dev/pts

chroot chroot /bin/bash

umount chroot/proc
umount chroot/sys
umount -l chroot/dev
umount chroot/dev/pts
```

**compress.sh**

```
#!/bin/bash

if [[ $EUID -ne 0 ]]; then
   echo "Este script se debe ejecutar como root" 1>&2
   echo "Cuando se comprime el chroot la mayoria de ficheros \
   son propietarios de root" 1>&2
   echo "Un usuario normal no tiene permiso para leerlos todos" 1>&2
   exit 1
fi

tar --numeric-owner -cj -f chroot.tar.bz2 chroot
```

**create-disk.sh**

```
#!/bin/bash

LOOP=/dev/loop0
MBRSYSLINUX=/usr/lib/syslinux/mbr.bin
MNTPOINT=/tmp/mnt/
IMG=disk.img
```

```
OFFSET1=1048576
SIZE1=32768
OFFSET2=34603008

[ -f $MBRSYSLINUX ] || (echo "MBR for syslinux not found" && exit 1)

echo Creando imagen vacia
dd if=/dev/zero of=$IMG bs=1G count=1

echo Particionando
fdisk -u=sectors $IMG  < input.fdisk

echo Anyadiendo master boot record de SYSLINUX
dd bs=440 count=1 conv=notrunc if=$MBRSYSLINUX of=$IMG

echo Montando loop particion 1
losetup -o $OFFSET1 $LOOP $IMG

echo Formateando particion 1
mkfs.vfat -F 16 $LOOP $SIZE1

echo Montando particion
mkdir -p $MNTPOINT
mount $LOOP $MNTPOINT

echo Copiando necesario para syslinux
cp -rf syslinux $MNTPOINT
cp chroot/vmlinuz chroot/initrd.img $MNTPOINT

echo Desmontando particion
umount $LOOP

echo Instalando syslinux
syslinux $LOOP

echo Desmontando loop
losetup -d $LOOP

echo Montando loop particion 2
losetup -o $OFFSET2 $LOOP $IMG
```

```
echo Formateando particion 2
mkfs.ext2 -b 1024 $LOOP

echo Montando particion 2
mount $LOOP $MNTPOINT

echo Copiando chroot
cp -rfa chroot/* $MNTPOINT

echo Desmontando particion
umount $LOOP

echo Desmontando loop
losetup -d $LOOP

echo Cambiando permisos de la imagen generada
chmod +rwx $IMG

echo "#######################################################"
echo "Imagen generada. Para probarla ejecutar qemu (kvm)"
echo "La maquina virtual se controla desde la misma consola \
imitando al puerto serie"
echo ""
echo "kvm -hda $IMG -serial stdio"
```

**create-disk-cf.sh**

```
#!/bin/bash

MBRSYSLINUX=/usr/lib/syslinux/mbr.bin
MNTPOINT=/tmp/mnt/

if [ "$(id -u)" != "0" ]; then
echo "Debe tener privilegios de superusuario para \
ejecutar este script." 1>&2
exit 1
fi

[ -f $MBRSYSLINUX ] || (echo "MBR for syslinux not found" && exit 1)
```

```
DEV=‘fdisk -l | grep -w $1:‘

if [ "$DEV" == ’’ -o "$1" == ’’ ]; then
echo No se ha especificado un dispositivo válido
exit 1
fi

echo "Ha seleccionado el dispositivo $1. Esta es la geometría \
y las particiones del dispositivo:"
fdisk -l $1
echo ""

RESP=""
while [ "$RESP" != "s" -a "$RESP" != "n" ]
do
echo -ne "ATENCIÓN! Esta operación destruirá todos \
los datos contenidos en $1. Está seguro de que desea continuar (s/n)? "
read RESP
done

if [ "$RESP" == "n" ]; then
exit 1
fi

#Desmontamos todas las particiones del dispositivo
#que pudiesen estar montadas
umount $1*

PART1=$1"1"
PART2=$1"2"

echo Particionando
fdisk $1 < input-cf.fdisk

echo Anyadiendo master boot record de SYSLINUX
dd bs=440 count=1 conv=notrunc if=$MBRSYSLINUX of=$1

echo Formateando particion 1
mkfs.vfat -F 16 $PART1
```

```
echo Montando particion
mkdir -p $MNTPOINT
mount $PART1 $MNTPOINT

echo Copiando necesario para syslinux
cp -rf syslinux $MNTPOINT
cp chroot/vmlinuz chroot/initrd.img $MNTPOINT

echo Desmontando particion
umount $MNTPOINT

echo Instalando syslinux
syslinux $PART1

echo Formateando particion 2
mkfs.ext2 -b 1024 $PART2

echo Montando particion 2
mount $PART2 $MNTPOINT

echo Copiando chroot
cp -rfa chroot/* $MNTPOINT

echo Desmontando particion
umount $MNTPOINT

echo "#######################################################"
echo "Imagen generada. Para probarla ejecutar qemu (kvm)"
echo "La maquina virtual se controla desde la misma consola \
imitando al puerto serie"
echo ""
echo "kvm -hda $1 -serial stdio"
```

**extract.sh**

```
#!/bin/bash

if [[ $EUID -ne 0 ]]; then
   echo "Este script se debe ejecutar como root" 1>&2
```

```
   echo "Cuando se descomprime el chroot la mayoria \
   de ficheros deben ser propietarios de root" 1>&2
   echo "Un usuario normal no puede hacer eso" 1>&2
   exit 1
fi

tar --numeric-owner -xj -f chroot.tar.bz2 chroot
```

### pull-chroot.sh

```
#!/bin/bash

echo Descargando la imagen
rsync -avzP 1984:chroot.tar.bz2 chroot.tar.bz2
```

### push-chroot.sh

```
#!/bin/bash

echo Subiendo la imagen
rsync -avz chroot.tar.bz2 1984:chroot.tar.bz2
```

### Script para optimizar parámetros del kernel para conseguir un mayor throughput

```
#!/bin/sh
# 4KB send buffer, 20,480 connections max at worst case
echo 83886080  > /proc/sys/net/core/wmem_max
echo 83886080  > /proc/sys/net/core/wmem_default
# 16KB receive buffer, 20,480 connections max at worst case
echo 335544320 > /proc/sys/net/core/rmem_max
echo 335544320 > /proc/sys/net/core/rmem_default
# Max open files
echo 65536 > /proc/sys/fs/filemax
# Fast port recycling (TIME_WAIT)
echo 1 >/proc/sys/net/ipv4/tcp_tw_recycle
echo 1 >/proc/sys/net/ipv4/tcp_tw_reuse
```

```
# TIME_WAIT buckets increased
echo 65536 > /proc/sys/net/ipv4/tcp_max_tw_buckets
# FIN timeout decreased
echo 15 > /proc/sys/net/ipv4/tcp_fin_timeout
# SYN backlog increased
echo 65536 > /proc/sys/net/ipv4/tcp_max_syn_backlog
# SYN cookies enabled
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
# Local port range maximized
echo "1024 65535" > /proc/sys/net/ipv4/ip_local_port_range
# Netdev backlog increased
echo 100000 > /proc/sys/net/core/netdev_max_backlog
```

# Bibliografía


[1] Abolhasan, M.; Hagelstein, B.; and Wang, J. C.-P.*Real-world performance of current proactive multi-hop mesh protocols.* Real-world Performance of Current Proactive Multi-hop Mesh Protocols. 2009.

[2] Hiertz G. R., Max S., Zhao R., Denteneer D., Berlemann L. *Principles of IEEE 802.11s.* Computer Communications and Networks, Honolulu. 2007.

[3] Murray D., Dixon M., Koziniec T. *An Experimental Comparison of Routing Protocols in Multi Hop Ad Hoc Networks.* Australasian Telecommunication Networks and Applications Conference, ATNAC, Auckland,Nueva Zelanda. 2010.

[4] Akyildiz I. F., Wang X., Wang W. *Wireless mesh networks: a survey.* Elsevier. 2005.

[5] Joshi A., Gossain H., Jetcheva J., Audeh M., Bahr M., Kruys J., Lim A., Rahman S., Kim J., Conner S., Strutt G., Liu H., Hares S. *HWMP Specification* IEEE. 2006.

[6] Chroboczek J. *The Babel Routing Protocol* Internet Engineering Task Force (IETF). 2011.

[7] Clausen T., Jacquet P. *Optimized Link State Routing Protocol (OLSR)* Internet Engineering Task Force (IETF). Octubre 2003.

[8] Johnson D., Hu Y., Maltz D. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks for IPv4* Internet Engineering Task Force (IETF). 2007.

[9] Neumann A., Aichele C., Lindner M., Wunderlich S. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)* Internet Engineering Task Force (IETF). 2008.

[10] Perkins C., Belding-Royer E., Das S. *Ad hoc On-Demand Distance Vector (AODV) Routing* Internet Engineering Task Force (IETF). 2003.

[11] Bakht H. *The history of mobile ad-hoc networks* `http://www.computingunplugged.com/issues/issue200508/00001598001.html` Computing Unplugged Magazine, ZATZ Publishing. 2005.

[12] Javier Martín Solera. *Nuevos protocolos de encaminamiento para redes móviles Ad-Hoc* Proyecto Fin de Carrera de la Universitat Politécnica de Catalunya, Castelldefels, Febrero de 2006.

[13] D. M. Urra. *Comparativa de implementaciones de protocolos reactivos de encaminamiento en redes Ad-Hoc* Proyecto Fin de Carrera de la Universitat Politécnica de Catalunya, Castelldefels, Febrero de 2006.

[14] Alejandro Medina Santos *Comparativa de los protocolos AODV y OSLR con un emulador de redes Ad-Hoc* Proyecto Fin de Carrera de la Universitat Politécnica de Catalunya, Castelldefels, Febrero de 2006.

[15] Miguel Angel Ortuño Pérez *Encaminamiento en redes Ad-Hoc* Septiembre de 2010

[16] Thomas Heide Clausen *The optimized link state routing protocolo version 2* `http://www.ietf.org/proceedings/63/slides/manet-5.pdf`

[17] IEEE 802.11s Task Group *IEEE P802.11s/D10.01. Draft STANDARD for Information Technology. Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 10: Mesh Networking* IEEE 802.11s Draft 10.01 Abril de 2011

[18] IEEE 802.11s Task Group *IEEE P802.11s/D0.01. Draft STANDARD for Information Technology. Telecommunications and information exchange between systems. Local and metropolitan area networks. Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 10: Mesh Networking* IEEE 802.11s Draft 0.01 Marzo de 2006

[19] C. Perkins *Ad hoc On-Demand Distance Vector (AODV) Routing Draft 00* Draft IETF MANET AODV 00 Noviembre de 1997

[20] J. Broch, D. Johnson y D. Maltz *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks* Draft IETF MANET DSR 00 Marzo de 1998

[21] P. Jacquet, P. Muhlethaler y Amir Qayyum *Optimized Link State Routing Protocol* Draft IETF MANET OLSR 00 Noviembre de 1998

[22] Juliusz Chroboczek *The Babel routing protocol* draft-chroboczek-babel-routing-protocol-00 Abril de 2009

[23] Francisco Javier Campos Berga *Anonimato en redes ad-hoc mediante integración de los protocolos HIP y OLSR* Julio de 2009

[24] Jordi Chameta Ugas *Estudio y análisis de prestaciones de redes móviles Ad Hoc mediante simulaciones NS-2 para validar modelos analíticos* Proyecto Fin de Carrera de la Universitat Politécnica de Catalunya, Barcelona Noviembre de 2009

[25] Qamar Abbas Tarar *Optimized link state routing protocol for ad-hoc networks* University of Cyprus, Computer Science Departament

[26] Capacity of Wireless Networks. `http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/18/17872/00825799.pdf` IEEE.

[27] Capacity of Ad-Hoc Wireless Network. `http://pdos.csail.mit.edu/papers/grid:mobicom01/paper.pdf` Massachusetts Institute of Technology.

[28] *The criminal mastermind: bufferbloat!*. Jim Gettys. http://gettys.wordpress.com/2010/12/03/introducing-the-criminal-mastermind-bufferbloat/

# Agradecimientos.

A mi padre y a mi madre, todo lo bueno de mi persona (que algo habrá) es gracias a ellos. A mi hermana, por ser la mejor hermana del mundo. A mis amigos y amigas, por aguantarme, por apoyarme y por hacer mi vida más feliz. A Pablo, amigo y tutor de este proyecto, que me dio la idea y me ha guiado durante todo el proceso. A Alex, que me ha ayudado tanto. Y a mis compañeras y compañeros de La Huelga, por aguantarme en las asambleas y porque sé que esperan pacientemente a que les ponga wifi gratis.

# Licencia



## Definiciones

1. La obra es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

2. En esta licencia se considera una prestación cualquier interpretación, ejecución, fonograma, grabación audiovisual, emisión o transmisión, mera fotografía u otros objetos protegidos por la legislación de propiedad intelectual vigente aplicable.

3. La aplicación de esta licencia a una colección (definida más adelante) afectará únicamente a su estructura en cuanto forma de expresión de la selección o disposición de sus contenidos, no siendo extensiva a éstos. En este caso la colección tendrá la consideración de obra a efectos de esta licencia.

4. El titular originario es:

   *a*) En el caso de una obra literaria, artística o científica, la persona natural o grupo de personas que creó la obra.

   *b*) En el caso de una obra colectiva, la persona que la edite y divulgue bajo su nombre, salvo pacto contrario.

   *c*) En el caso de una interpretación o ejecución, el actor, cantante, músico, o cualquier otra persona que represente, cante, lea, recite, interprete o ejecute en cualquier forma una obra.

   *d*) En el caso de un fonograma, el productor fonográfico, es decir, la persona natural o jurídica bajo cuya iniciativa y responsabilidad se realiza por primera vez una fijación exclusivamente sonora de la ejecución de una obra o de otros sonidos.

   *e*) En el caso de una grabación audiovisual, el productor de la grabación, es decir, la persona natural o jurídica que tenga la iniciativa y asuma la responsabilidad de las fijaciones de un plano o secuencia de imágenes, con o sin sonido.

   *f*) En el caso de una emisión o una transmisión, la entidad de radiodifusión.

   *g*) En el caso de una mera fotografía, aquella persona que la haya realizado.

   *h*) En el caso de otros objetos protegidos por la legislación de propiedad intelectual vigente, la persona que ésta señale.

5. Se considerarán obras derivadas aquellas obras creadas a partir de la licenciada, como por ejemplo: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de la obra con una imagen en movimiento (synching) será considerada como una obra derivada a efectos de esta licencia.

6. Tendrán la consideración de colecciones la recopilación de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos que por la selección o disposición de sus contenidos constituyan creaciones intelectuales. La mera incorporación de una obra en una colección no dará lugar a una derivada a efectos de esta licencia.

7. El licenciador es la persona o la entidad que ofrece la obra o prestación bajo los términos de esta licencia y le concede los derechos de explotación de la misma conforme a lo dispuesto en ella. Usted es la persona o la entidad que ejercita los derechos concedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra o la prestación, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos concedidos mediante esta licencia a pesar de una violación anterior.

8. La transformación de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

9. Se entiende por reproducción la fijación directa o indirecta, provisional o permanente, por cualquier medio y en cualquier forma, de toda la obra o la prestación o de parte de ella, que permita su comunicación o la obtención de copias.

10. Se entiende por distribución la puesta a disposición del público del original o de las copias de la obra o la prestación, en un soporte tangible, mediante su venta, alquiler, préstamo o de cualquier otra forma.

11. Se entiende por comunicación pública todo acto por el cual una pluralidad de personas, que no pertenezcan al ámbito doméstico de quien la lleva a cabo, pueda tener acceso a la obra o la prestación sin previa distribución de ejemplares a cada una de ellas. Se considera comunicación pública la puesta a disposición del público de obras o prestaciones por procedimientos alámbricos o inalámbricos, de tal forma que cualquier persona pueda acceder a ellas desde el lugar y en el momento que elija.

12. La explotación de la obra o la prestación comprende la reproducción, la distribución, la comunicación pública y, en su caso, la transformación.

13. Los elementos de la licencia son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta licencia: Reconocimiento, NoComercial, CompartirIgual.

14. Una licencia equivalente es:

    *a*) Una versión posterior de esta licencia de Creative Commons con los mismos elementos de licencia.

    *b*) La misma versión o una versión posterior de esta licencia de cualquier otra jurisdicción reconocida por Creative Commons con los mismos elementos de licencia (por ejemplo: Reconocimiento-NoComercial-CompartirIgual 3.0 Japón).

    *c*) La misma versión o una versión posterior de la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) con los mismos elementos de la licencia.

## Límites de los derechos

Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de propiedad intelectual o cualesquiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como la copia privada o la cita, u otras limitaciones como la resultante de la primera venta de ejemplares (agotamiento).

## Concesión de licencia

Conforme a los términos y a las condiciones de esta licencia, el licenciador concede, por el plazo de protección de los derechos de propiedad intelectual y a título gratuito, una licencia de ámbito mundial no exclusiva que incluye los derechos siguientes:

1. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación.

2. Derecho a incorporar la obra o la prestación en una o más colecciones.

3. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación lícitamente incorporada en una colección.

4. Derecho de transformación de la obra para crear una obra derivada siempre y cuando se incluya en ésta una indicación de la transformación o modificación efectuada.

5. Derecho de reproducción, distribución y comunicación pública de obras derivadas creadas a partir de la obra licenciada.

6. Derecho a extraer y reutilizar la obra o la prestación de una base de datos.

Estos derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos en el momento de la concesión de esta licencia. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los establecidos en la sección 4.f, así como los derechos morales irrenunciables reconocidos por la ley aplicable. En la medida en que el licenciador ostente derechos exclusivos previstos por la ley nacional vigente que implementa la directiva europea en materia de derecho sui generis sobre bases de datos, renuncia expresamente a dichos derechos exclusivos.

# Restricciones

La concesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

1. Usted puede reproducir, distribuir o comunicar públicamente la obra o prestación solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI). Usted no puede ofrecer o imponer ninguna condición sobre la obra o prestación que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede sublicenciar la obra o prestación. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra o prestación con medidas tecnológicas que controlen el acceso o el uso de una manera contraria a los términos de esta licencia. Esta sección 4.a también afecta a la obra o prestación incorporada

en una colección, pero ello no implica que ésta en su conjunto quede automáticamente o deba quedar sujeta a los términos de la misma. En el caso que le sea requerido, previa comunicación del licenciador, si usted incorpora la obra en una colección y/o crea una obra derivada, deberá quitar cualquier crédito requerido en el apartado 4.d, en la medida de lo posible.

2. Usted puede distribuir o comunicar públicamente una obra derivada en el sentido de esta licencia solamente bajo los términos de la misma u otra licencia equivalente. Si usted utiliza esta misma licencia debe incluir una copia o bien su URI, con cada obra derivada que usted distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término respecto a la obra derivada que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías cuando distribuya o comunique públicamente la obra derivada. Usted no puede ofrecer o imponer ningún término respecto de las obras derivadas o sus transformaciones que alteren o restrinjan los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede reproducir, distribuir o comunicar públicamente la obra derivada con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Si utiliza una licencia equivalente debe cumplir con los requisitos que ésta establezca cuando distribuya o comunique públicamente la obra derivada. Todas estas condiciones se aplican a una obra derivada en tanto que incorporada a una colección, pero no implica que ésta tenga que estar sujeta a los términos de esta licencia.

3. Usted no puede ejercitar ninguno de los derechos concedidos en la sección 3 anterior de manera que pretenda principalmente o su actuación se dirija a la obtención de un beneficio mercantil o una contraprestación monetaria. El intercambio de la obra por otras obras protegidas por la propiedad intelectual mediante sistemas de compartir archivos no se considerará como una manera que pretenda principalmente o se encuentre dirigida hacia la obtención de un beneficio mercantil o una contraprestación monetaria, siempre que no haya ningún pago en relación con el intercambio de las obras protegidas.

4. Si usted reproduce, distribuye o comunica públicamente la obra o la prestación, una colección que la incorpore o cualquier obra derivada, debe mantener intactos todos los avisos sobre la propiedad intelectual

e indicar, de manera razonable conforme al medio o a los medios que usted esté utilizando:

   *a*) El nombre del autor original, o el seudónimo si es el caso, así como el del titular originario, si le es facilitado.

   *b*) El nombre de aquellas partes (por ejemplo: institución, publicación, revista) que el titular originario y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable.

   *c*) El título de la obra o la prestación si le es facilitado.

   *d*) El URI, si existe, que el licenciador especifique para ser vinculado a la obra o la prestación, a menos que tal URI no se refiera al aviso legal o a la información sobre la licencia de la obra o la prestación.

   *e*) En el caso de una obra derivada, un aviso que identifique la transformación de la obra en la obra derivada (p. ej., "traducción castellana de la obra de Autor Original,.º "guión basado en obra original de Autor Original").

   Este reconocimiento debe hacerse de manera razonable. En el caso de una obra derivada o incorporación en una colección estos créditos deberán aparecer como mínimo en el mismo lugar donde se hallen los correspondientes a otros autores o titulares y de forma comparable a los mismos. Para evitar la duda, los créditos requeridos en esta sección sólo serán utilizados a efectos de atribución de la obra o la prestación en la manera especificada anteriormente. Sin un permiso previo por escrito, usted no puede afirmar ni dar a entender implícitamente ni explícitamente ninguna conexión, patrocinio o aprobación por parte del titular originario, el licenciador y/o las partes reconocidas hacia usted o hacia el uso que hace de la obra o la prestación.

5. Para evitar cualquier duda, debe hacerse notar que las restricciones anteriores (párrafos 4.a, 4.b, 4.c y 4.d) no son de aplicación a aquellas partes de la obra o la prestación objeto de esta licencia que únicamente puedan ser protegidas mediante el derecho sui generis sobre bases de datos recogido por la ley nacional vigente implementando la directiva europea de bases de datos

6. Para evitar cualquier duda, el titular originario conserva:

   *a*) El derecho a percibir las remuneraciones o compensaciones previstas por actos de explotación de la obra o prestación, calificadas

por la ley como irrenunciables e inalienables y sujetas a gestión colectiva obligatoria.

*b*) El derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión colectiva de derechos, cualquier remuneración derivada de actos de explotación de la obra o prestación que usted realice que no queden sujetos a esta licencia de acuerdo con lo establecido en el apartado 4.c.

## Exoneración de responsabilidad

A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA O LA PRESTACIÓN TAL CUAL (ON AN .ᴬS-IS"BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O LA PRESTACIÓN O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

## Limitación de responsabilidad

SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA O LA PRESTACIÓN, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

## Finalización de la licencia

- Esta licencia y la concesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos

de la misma. Las personas o entidades que hayan recibido de usted obras derivadas o colecciones bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

- Conforme a las condiciones y términos anteriores, la concesión de derechos de esta licencia es vigente por todo el plazo de protección de los derechos de propiedad intelectual según la ley aplicable. A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra o la prestación en condiciones distintas a las presentes, o de retirar la obra o la prestación en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente, sin perjuicio del derecho moral de arrepentimiento en los términos reconocidos por la ley de propiedad intelectual aplicable.

## Miscelánea

- Cada vez que usted realice cualquier tipo de explotación de la obra o la prestación, o de una colección que la incorpore, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra o la prestación en las mismas condiciones y términos que la licencia concedida a usted.

- Cada vez que usted realice cualquier tipo de explotación de una obra derivada, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra objeto de esta licencia en las mismas condiciones y términos que la licencia concedida a usted.

- Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los términos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

- No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.
- Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra o la prestación objeto de la licencia. No caben interpretaciones, acuerdos o condiciones con respecto a la obra o la prestación que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación que le haga llegar usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.