PROYECTO FIN DE MÁSTER EN

SISTEMAS INTELIGENTES

CURSO 2008-2009

PROTOCOLO DISTRIBUIDO PARA LA CONFIGURACIÓN DINÁMICA DE DIRECCIONES EN REDES MÓVILES AD HOC

Ana Lucila Sandoval Orozco

Director:

Alberto Díaz Esteban

Departamento de Ingeniería del Software e Inteligencia Artificial

Colaborador externo de dirección:

Luis Javier García Villalba

Departamento de Ingeniería del Software e Inteligencia Artificial

---

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

La abajo firmante, matriculada en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "***Protocolo Distribuido para la Configuración Dinámica de Direcciones en Redes Móviles Ad hoc***", realizado durante el curso académico 2008-2009 bajo la dirección de Alberto Díaz Esteban y con la colaboración externa de dirección de Luis Javier García Villalba en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

_____

Ana Lucila Sandoval Orozco

Abstract

Mobile ad hoc networks (MANETs) are multihop wireless networks of mobile nodes without any fixed or preexisting infrastructure. The topology of these networks can change randomly due to unpredictable mobility of nodes and propagation characteristics. In most networks, including MANETs, each node needs a unique identifier to communicate. This work presents a distributed protocol for dynamic IP address assignment to nodes in MANETs. Nodes of a MANET synchronize from time to time to keep record of IP address assignment in the entire network and detect any IP address leaks. The proposed stateful autoconfiguration scheme uses the OLSR proactive routing protocol for synchronization and guarantees unique IP address under a variety of network conditions including message losses and network partitioning. Simulations results show that the protocol incurs low latency and communication overhead for an IP address assignment.

Keywords

Mobile ad hoc network, MANETs, IP address assignment, autoconfiguration, dynamic host configuration, stateful protocol, synchronization, OLSR proactive routing protocol.

Resumen

Las redes móviles ad hoc (MANETs) son redes inalámbricas multisalto de nodos móviles sin infraestructura previa alguna. La topología de estas redes puede cambiar aleatoriamente debido a la movilidad imprevisible de los nodos y a las características de propagación. En la mayoría de las redes, MANETs incluidas, cada nodo necesita un identificador único para comunicarse. Este trabajo presenta un protocolo distribuido para la asignación dinámica de direcciones IP a los nodos de redes móviles ad hoc. Los nodos de una red MANET se sincronizan periódicamente para mantener actualizada la asignación de direcciones IP en la red y detectar cualquier inconsistencia en la misma. El esquema de autoconfiguración de estado completo propuesto utiliza el protocolo de encaminamiento proactivo OLSR para la sincronización y garantiza unicidad en las direcciones IP bajo una amplia variedad de condiciones de red que incluyen pérdidas de mensaje y partición de redes. Los resultados de la simulación demuestran que el protocolo tiene baja latencia y sobrecarga.

Palabras clave

Redes móviles ad hoc, MANETs, asignación de dirección IP, autoconfiguración, configuración dinámica de host, protocolo de estado completo, sincronización, protocolo de encaminamiento proactivo OLSR.

Agradecimientos

Son muchas las personas a quienes debería nombrar en estas líneas, pero me quedaré con las más trascendentales, con aquellas que no han bajado la guardia y siempre me han apoyado, tanto a lo largo del desarrollo de este Proyecto Fin de Máster como a lo largo de mi vida.

Primero doy gracias a Dios por estar siempre presente en cada momento de mi vida.

A mis padres que desde pequeña me han guiado y acompañado en los momentos en que más les he necesitado.

A mis hermanas que son mis ojos y no puedo sino darles las gracias por siempre estar presentes, ser mi apoyo y mi puesta a tierra.

A Javier García que siempre ha confiado en mis ideas, en mi trabajo y, sobre todo, ha puesto en mí la mentalidad de que se puede ser cada vez mejor en lo que uno hace. Gracias por la confianza y el apoyo que me ha brindado desde el primer día en que llegué a Madrid.

A Lourdes por su amabilidad y sonrisa.

Al Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA) por el apoyo logístico y las facilidades brindadas y, muy especialmente, a Alberto Díaz, por acceder a ser mi Director.

Finalmente, y no menos importante, gracias a todo el equipo del Grupo de Análisis, Seguridad y Sistemas (GASS), por la amistad y las tardes de cervezas, en donde no sólo éramos un grupo de trabajo, sino también un grupo de amigos.

Lista de acrónimos

AA	Address Agent
AODV	Ad hoc On-demand Distance Vector
APAC	Agent-based Passive AutoConfiguration
AROD	Autoconfiguration with address Reservation and Optimistic duplicated address Detection
ARP	Address Resolution Protocol
CSMA	Carrier Sense Multiple Access
DAD	Duplicate Address Detection
DAD_REP	Detection Adress Detection REsPonse
DAD_REQ	Detection Adress Detection REQuest
DARPA	Defense Advanced Research Projects Agency
DHCP	Dynamic Host Configuration Protocol
DHCP-PD	Dynamic Host Configuration Protocol – Prefix Delegation
DNS	Domain Name Servers
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
DYMO	DYnamic Manet On-demand
EMAP	Extensible Manet Autoconfiguration Protocol
FBCB2	Force XXI Battle Command, Brigade-and-Below
GNU	General Public License
GSM	Global System for Mobile communications
HCQA	Hybrid Centralized Query-based Autoconfiguration

ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
MAC	Media Access Control
MANET	Mobile Ad hoc NETwork
MPR	Multipoint Distribution Relay
NDP	Neighbor Discovery Protocol
NS-3	Network Simulator 3
OLSR	Optimized Link State Routing
PACMAN	Passive AutoConfiguration for Mobile Ad hoc Networks
PAN	Personal Area Networks
PDA	Personal Digital Assistant
PDAD	Passive Duplicate Address Detection
PPP	Point to Point Protocol
QoS	Quality of Service
RF	Radio Frequency
RFC	Request For Comments
RIP	Routing Information Protocol
RREP	Route REPly
RREQ	Route REQuest
SDAD	Strong Duplicate Address Detection
SLAAC	StateLess Address AutoConfiguration

SURAN	SUrvivable RAdio Network
TBRPF	Topology dissemination Based on Reverse-Path Forwarding
TC	Topology Control
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
TTL	Time To Live
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
WDAD	Weak Duplicate Address Detection
WG	Work Group
WiFi	Wireless Fidelity
WRP	Wireless Routing Protocol
WSN	Wireless Sensor Networks
ZRP	Zone-based hierarchical link state Routing Protocol

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Una red móvil ad hoc o *Mobile Ad hoc NETwork* (MANET) es un conjunto de nodos móviles que se comunican entre sí a través de enlaces inalámbricos (*wireless*). Al contrario de las redes convencionales, una red móvil ad hoc no necesita la existencia de una infraestructura previa ya que cada nodo se apoya en los demás para conseguir comunicarse con otro creando la llamada comunicación multisalto.

Este tipo de redes tiene varios inconvenientes que una red convencional no presenta. La topología de este tipo de redes puede cambiar rápidamente y de una forma impredecible. Además, pueden surgir variaciones en las capacidades de los nodos y enlaces, errores frecuentes en la transmisión y falta de seguridad. Por último, se deben tener en cuenta los recursos limitados de los nodos ya que normalmente una red ad hoc estará formada por dispositivos alimentados por baterías.

Para comunicarse entre ellos [2] los nodos ad hoc necesitan configurar sus interfaces con direcciones locales que son válidas dentro de una red ad hoc. Los nodos ad hoc también pueden necesitar configurar globalmente direcciones de encaminamiento para comunicarse con otros dispositivos en Internet. Desde la perspectiva de la capa IP, una red ad hoc se presenta como una red multisalto de nivel 3 constituida por una colección de enlaces.

El resto de este capítulo está organizado como sigue: El apartado 1.1 presenta el objeto de investigación de este trabajo. En el apartado 1.2 se analizan algunos trabajos relacionados. En último lugar, el apartado 1.3 resume la estructura del resto del trabajo.

1.1. Objeto de la investigación

En una red móvil ad hoc autónoma los nodos pueden identificarse unívocamente a través de una dirección IP con la única premisa de que esta dirección sea distinta a la de cualquier otro nodo de la red.

El proceso de configuración es el conjunto de pasos a través de los cuales un nodo consigue obtener su dirección IP dentro de la red. Existen dos mecanismos de configuración de direcciones: sin estado (*stateless*) y de estado completo (*stateful*).

La configuración de direcciones sin estado propone que sea el propio nodo el encargado de generar su dirección IP. La dirección se obtiene de la concatenación de un prefijo de red conocido y un número teóricamente único dentro de la red generada por el nodo. Este mecanismo puede exigir la inclusión de un módulo encargado de comprobar la unicidad de la dirección generada llamado Detección de Direcciones Duplicadas (*Duplicate Address Detection*, DAD).

Por otro lado, la configuración de direcciones de estado completo se basa en la utilización de servidores que controlan y asignan las direcciones a todos los nodos de la red. *Dynamic Host Configuration Protocol* (DHCP) [14] es un ejemplo de configuración de estado completo. Sin embargo, dada la naturaleza multisalto de las redes móviles ad hoc, este protocolo no puede ser aplicado directamente.

Este trabajo propone un protocolo de autoconfiguración de estado completo, que garantiza unicidad en las direcciones IP bajo una amplia variedad de condiciones de red que incluyen pérdidas de mensaje y partición de redes.

1.2. Trabajos relacionados

Las redes móviles ad hoc presentan características especiales que deben tenerse en cuenta a la hora de implementar un protocolo de configuración de direcciones. Existen muchas soluciones para redes convencionales (por ejemplo, RFCs 3315 [14], 4861 [34], 4862 [49], …) pero en su diseño no se tuvieron en cuenta las redes móviles ad hoc. Es necesario, pues, dar soporte multisalto, soporte a topologías dinámicas y soporte a la unión (*merging*) y partición (*partitioning*) de redes, eventos que son típicos en las redes móviles ad hoc.

Hay numerosos trabajos que realizan propuestas para la configuración de direcciones en una red móvil ad hoc utilizando tanto el mecanismo sin estado como el de estado completo. Quizás la *Internet Engineering Task Force* (IETF) [47] tenga el grupo de trabajo más conocido llamado *Ad-Hoc Network Autoconfiguration Work Group* (Autoconf WG) [2] cuya principal finalidad es describir el modelo de direccionamiento para redes ad hoc y cómo los nodos en estas redes configuran sus direcciones. Se exige que tales modelos no causen problemas a los demás componentes de un sistema ad hoc tales como aplicaciones estándar que se ejecuten en un nodo ad hoc o nodos de Internet conectados a los nodos ad hoc. La labor de este grupo puede incluir el desarrollo de nuevos protocolos si los mecanismos de autoconfiguración IP existentes resultan inadecuados. Sin embargo, la primera tarea de este grupo de trabajo es describir un modelo de direccionamiento práctico para redes ad hoc.

Cheshire *et al.* [10] describen un método de autoconfiguración del *host* eligiendo aleatoriamente una dirección local de enlace dentro del rango 169.254.1.0 - 169.254.254.255. Después de seleccionar la dirección, el *host* comprueba si la dirección está en uso por otro nodo. Esta aproximación se centra en redes cableadas y asegura la unicidad de la dirección local de enlace. Se requiere que cada nodo en la red esté dentro del rango de comunicación de

los otros nodos, lo cual no siempre es posible en el caso de una red móvil ad hoc. Para extender la solución a redes móviles ad hoc, mensajes de detección de conflicto tendrán que ser difundidos por inundación (*flooding*) por la red.

Perkins *et al.* [43] propone una solución para la autoconfiguración de direcciones en redes ad hoc. Se elige aleatoriamente una dirección dentro del rango 2048 a 65534 de la dirección de bloque 169.254/16. Un nodo envía *Route Requests* (RREQs) para la dirección IP solicitada. Si no recibe *Route Reply* (RREP) dentro de un período de tiempo, intenta RREQ RETRIES veces. Al finalizar todos los intentos, si no se recibe respuesta, se asume que la dirección IP elegida está libre. El nodo se asigna esa dirección IP. En este caso la latencia es el valor del *timeout* multiplicado por RREQ RETRIES. Esta aproximación requiere que el protocolo de encaminamiento tenga una fase de *descubrimiento de rutas*. No aborda el problema de la partición de redes.

La autoconfiguración sin estado de IPv6 [49] especifica los pasos de un nodo que quiera configurar sus interfaces en IPv6. Los pasos incluyen la construcción de una dirección local de enlace, Detección de Direcciones Duplicadas (DAD), y construcción de una dirección local de sitio. La Detección de Direcciones Duplicadas en redes móviles ad hoc requiere de difusión, lo que hace que esta aproximación no sea escalable. Para abordar el problema de la escalabilidad en [53] se propone una extensión construyendo una estructura jerárquica. Pero el coste que requiere mantener tal estructura jerárquica puede ser elevado.

En MANETconf [36] un nodo que desea entrar en la red móvil ad hoc solicita información de configuración de sus vecinos. Uno de estos vecinos inicia el proceso de asignación de dirección IP para el nuevo nodo. Para cada asignación de dirección IP, esta aproximación requiere un *broadcast* a toda la red, originando problemas de escalabilidad. Sin embargo, esta aproximación gestiona la partición y unión de redes.

El protocolo *Weak Duplicate Address Detection* (WDAD) [52] precisa que cada nodo en la red tenga una clave única. WDAD requiere que los paquetes destinados a un nodo no deben ser encaminados a ningún otro, incluso si los dos nodos han elegido la misma dirección. Esto se logra utilizando la información de la clave en la detección de dirección duplicada. En esta aproximación se requiere modificar el protocolo de encaminamiento en lo relativo a los paquetes de control para transportar la información de la clave. En el esquema WDAD puede perderse el paquete en el intervalo que transcurre entre la ocurrencia de direcciones IP duplicadas en la red y su detección correspondiente. En la versión mejorada de WDAD [52] se elimina el inconveniente anterior utilizando números de secuencia y un contador.

La Detección Pasiva de Direcciones Duplicadas (*Passive Duplicate Address Detection*, PDAD) presentada en [55] intenta detectar direcciones duplicadas sin difundir información de control adicional. Basado en el encaminamiento clásico de estado de enlace propone tres esquemas basados en números de secuencia, en el principio de localidad y en el vecindario. Los esquemas PDAD propuestos utilizan identificadores de origen aleatorios para detectar direcciones duplicadas dentro del vecindario a dos saltos. Esta aproximación requiere el uso de un protocolo de encaminamiento de estado de enlace.

El esquema de autoconfiguración de direcciones sin estado presentado en [24] consta de tres fases: (1) selección de dirección aleatoria, (2) verificación de la unicidad de la dirección y, (3) asignación de la dirección a la interfaz de red. La verificación de la unicidad se hace por un esquema DAD híbrido de dos fases: (a) fase de *Strong Duplicate Address Detection* (SDAD) y, (b) fase de WDAD. Dentro de una red ad hoc conectada, un nodo se configura a sí mismo con una dirección IP utilizando SDAD. Durante la SDAD, el nodo elige una dirección tentativa y comprueba si está duplicada enviando el mensaje AREQ por *broadcast* con la dirección tentativa elegida durante un número determinado

de veces. Si no recibe respuesta al mensaje AREQ, el nodo se configura a sí mismo con la dirección tentativa. WDAD utiliza una clave además de la dirección IP para detectar direcciones duplicadas durante el encaminamiento ad hoc. Este esquema asegura que durante la resolución de un conflicto de direcciones, las sesiones que utilizan direcciones conflictivas son mantenidas hasta que se cierran.

Dynamic Address Allocation Protocol (DAAP) propuesto en [40] requiere que los nuevos nodos soliciten al líder de la red la dirección IP. El líder es el nodo con la dirección IP más alta en la red. Contempla la partición y la unión de redes. El identificador único utilizado para la identificación de la red es la dirección MAC del nodo iniciador de la red. Esto podría originar un problema de múltiples redes que tengan el mismo identificador cuando el nodo iniciador se mueva fuera de la red y forme otra red. No contempla la pérdida de mensajes.

Dynamic Registration and Configuration Protocol (DRCP) [28] [29] extiende DHCP a redes inalámbricas. En este protocolo cada nodo actúa a la vez como servidor y como cliente y posee un conjunto de direcciones. La distribución del conjunto de direcciones se hace utilizando DAAP. Cada nodo obtiene el conjunto de direcciones solicitando la mitad de las direcciones del conjunto de direcciones del nodo vecino. El protocolo no contempla la partición de redes ni la pérdida de mensajes.

Zhou *et al.* [56] proponen una solución que se deriva del esquema de la generación de secuencia donde una secuencia consta de números dentro de un rango R que se genera utilizando una función f(n). Esta función se elige de tal forma que en las secuencias generadas por ella, el intervalo entre dos ocurrencias del mismo número sea muy grande y la probabilidad de que más de una ocurrencia del mismo número en un número limitado de secuencias diferentes iniciadas por distintas semillas sea extremadamente bajo. El

protocolo requiere que el primer nodo elija un número aleatorio del rango R como su dirección IP y utilice un valor de estado aleatorio como semilla para la función f(n). Cuando un nuevo nodo se une a la red, el nodo configurado genera otro entero y un nuevo valor de estado usando f(n). El nuevo nodo obtiene estos valores y se configura a sí mismo. En esta aproximación, el bloque requerido de direcciones IP puede ser significativamente más grande que el número de nodos en la red. Incluso si el intervalo mínimo entre dos ocurrencias de la misma dirección en la secuencia es extremadamente grande, todavía es posible que dos nodos tengan la misma dirección IP si los nodos entran y salen de la red con elevada frecuencia.

Ros *et al.* [45] proponen *Extensible Manet Autoconfiguration Protocol* (EMAP), que sigue con la idea del sistema de detección de direcciones duplicadas. Esta aproximación requiere una adaptación de los protocolos de encaminamiento para redes móviles ad hoc existentes. No contempla explícitamente la unión de redes.

Thoppian y Prakash [50] proponen una asignación dinámica de direcciones basada en el denominado *buddy system* (sistema de amigos) que maneja movilidad de los nodos durante la asignación de direcciones, pérdida de mensajes, partición y unión de redes. Sin embargo, la asignación de direcciones IP puede generar alta sobrecarga de los mensajes de control mientras hace una búsqueda global y la recuperación de direcciones (para evitar pérdidas de direcciones) requiere de mensajes de difusión por inundación (*flooding*). Además, la unión y partición pueden incurrir en una alta sobrecarga debido a la naturaleza global de este protocolo.

Kim *et al.* [26] presentan AROD (*Address autoconfiguration with Address Reservation and Optimistic duplicated address*), un mecanismo de autoconfiguración en el que la reserva de direcciones se basa en la existencia de unos nodos que tienen una dirección IP reservada para entregársela a los nodos

que entren nuevos. Diferencia dos clases de nodos: Agentes tipo 1 con direcciones IP reservadas y Agentes tipo 2, que no tienen direcciones IP reservadas. Este protocolo considera su proceso de detección de direcciones duplicadas como un proceso DAD optimista, porque solamente se lleva a cabo una vez cuando entra un nodo y se le asigna una dirección IP reservada. Contempla la posibilidad de cambiar el número de direcciones IP reservadas para los nodos tipo 1. Si aumenta el número, la latencia de asignación de direcciones IP es menor pero, por el contrario, aumenta la sobrecarga al tener que realizar más procesos DAD y viceversa. Esta posibilidad permite variar latencia versus sobrecarga.

Finalmente señalar que en [5] [6] [7] Bernardos *et al.* realizan un riguroso estudio de la problemática de la autoconfiguración en redes móviles ad hoc, presentando una revisión detallada de los protocolos de autoconfiguración más representativos.

Las soluciones descritas anteriormente han supuesto contribuciones significativas para nuestra comprensión del problema. Sin embargo, pensamos que todas estas aproximaciones manejan únicamente un subconjunto de las condiciones de red enumeradas a continuación:

1) Topología dinámica: Los nodos en la red se mueven arbitrariamente y pueden entrar y salir de la red dinámicamente.

2) Pérdidas de mensajes y fallos en los nodos: Las pérdidas de mensajes pueden ser bastante frecuentes y pueden duplicar la asignación de dirección IP si no se maneja correctamente. Los nodos pueden abandonar la red de forma abrupta debido a un fallo en el enlace o un accidente.

3) Partición y unión de redes: La red puede dividirse en múltiples redes y,

posteriormente, unirse con otras. Durante la unión de redes es posible tener direcciones IP duplicadas en la red fusionada.

4) Peticiones concurrentes de direcciones: Múltiples nodos pueden querer unirse a la red simultáneamente.

5) Energía y ancho de banda limitados: Los nodos en una red móvil ad hoc son de energía limitada y los enlaces tienen un limitado ancho de banda. Por tanto, la sobrecarga de comunicación en la que se incurra debería ser baja.

En este trabajo se propone una solución similar a DAAP [28] [29] y a [31] que garantiza unicidad en la asignación de direcciones IP bajo un amplio conjunto de condiciones de red. En nuestra aproximación, la mayoría de las asignaciones de direcciones implican comunicación local originando baja sobrecarga de comunicación y baja latencia.

1.3. Estructura del trabajo

El resto del trabajo está organizado en 8 capítulos con la estructura que se comenta a continuación.

El Capítulo 2 realiza un estado del arte de las redes ad hoc incluyendo un repaso cronológico de la evolución de las redes ad hoc, un análisis de las características básicas de este tipo de redes, una presentación del protocolo de comunicación que actualmente se utiliza en ellas, una clasificación de las redes ad hoc y un resumen de las principales aplicaciones de las redes móviles ad hoc.

El Capítulo 3 aborda el problema de la autoconfiguración en redes móviles

ad hoc. Comienza señalando la no aplicabilidad de las soluciones estándar. Posteriormente, presenta una clasificación de los protocolos de autoconfiguración para redes ad hoc analizando los más representativos. Finaliza con un estudio del protocolo de Moshin y Prakash [31], que supuso el punto de partida del protocolo que se propone en este trabajo.

El Capítulo 4 se centra en el encaminamiento de las redes móviles ad hoc. Se muestra especial atención al protocolo OLSR [11] ya que participa activamente en el protocolo de autoconfiguración propuesto en este trabajo.

El Capítulo 5 con las ideas básicas para entender el funcionamiento del protocolo de autoconfiguración para redes móviles ad hoc que se propone en este trabajo.

El Capítulo 6 presenta una especificación detallada del denominado *Distributed Dynamic Host Configuration Protocol* (D2HCP) ó Protocolo Distribuido para la Configuración Dinámica de Direcciones en Redes Móviles Ad Hoc, la contribución de este Proyecto Fin de Máster.

El Capítulo 7 contiene los resultados obtenidos de las simulaciones realizadas en el software NS-3 [48].

Por último, el Capítulo 8 muestra las principales conclusiones extraídas de este trabajo así como algunas líneas futuras de trabajo.

2. REDES MÓVILES AD HOC

Una red móvil ad hoc es un conjunto de nodos móviles que se comunican entre sí a través de enlaces inalámbricos (*wireless*). Al contrario de las redes convencionales, una red móvil ad hoc no necesita la existencia de una infraestructura previa ya que cada nodo se apoya en los demás para conseguir comunicarse con otro creando la llamada comunicación multisalto. La Figura 2.1 presenta un ejemplo típico de una red móvil ad hoc.

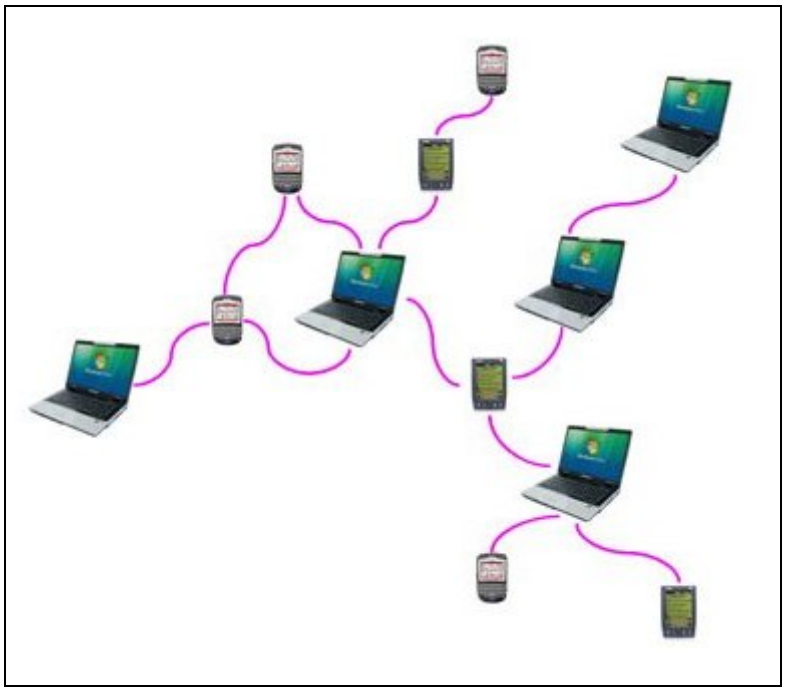

Fig. 2.1. Red móvil ad hoc.

El capítulo se estructura de la siguiente forma: En el apartado 2.1 se hace un repaso cronológico de la evolución de las redes ad hoc. En el apartado 2.2 se analizan las características básicas de este tipo de redes. En el apartado 2.3 se presta atención al protocolo de comunicación que actualmente se utiliza en este tipo de redes, el IEEE 802.11 [20]. El apartado 2.4 realiza una clasificación de las redes ad hoc. Por último, el apartado 2.5 resume sus principales aplicaciones.

2.1. Evolución Histórica

En muy pocos años, el campo de las redes ad hoc ha tenido una rápida expansión visible en la proliferación de dispositivos inalámbricos de bajo coste como ordenadores portátiles, asistentes personales digitales (PDAs), teléfonos móviles, etc.

A comienzos de los años 70 un trabajo pionero en radio de la Universidad de Hawai introduce el primer sistema que usa el medio de la radio para la transmisión de información. Conocido ampliamente como ALOHA [1], fue desarrollado por Abramson y Kuo.

El trabajo realizado en Hawai llevó en 1972 al desarrollo de una arquitectura distribuida consistente en una red de difusión de radio con mínimo control central llamada PARNET [13] bajo el patrocinio de DARPA (*Defense Advanced Research Projects Agency*). El proyecto ayudó a establecer el concepto de redes móviles ad hoc. PARNET permitía la comunicación directa entre usuarios móviles sobre grandes áreas geográficas, ancho de banda compartido y protección contra los efectos de múltiples caminos.

Los rápidos avances de la tecnología de la radio en los años 70 provocó la aparición de múltiples sistemas de comunicación móvil como teléfonos celulares e inalámbricos, sistemas de radio búsqueda, satélites móviles, etc.

Posteriormente, DARPA desarrolló el proyecto SURAN (*Survivable Radio Networks*) en 1983 que trata las tareas de escalabilidad de la red, seguridad, capacidad de proceso y gestión de energía. Se dedicaron esfuerzos para desarrollar dispositivos de bajo coste y con poco gasto de energía que pudieran soportar los avanzados protocolos de encaminamiento, escalar a miles de nodos las redes y dar soporte para ataques a la seguridad. El resultado fue la aparición

de la tecnología conocida como LPR (*Low-cost Packet Radio*) en 1987.

A mitad de los 90 se produce un nuevo avance con la llegada de tarjetas de radio 802.11 para ordenadores personales y portátiles. En los artículos [16][21] se propone por primera vez la idea de una colección de *hosts* móviles con una infraestructura mínima, y el IEEE (*Institute of Electrical and Electronic Engineers*) acuña el término *redes ad hoc*.

Durante el mismo tiempo, el Departamento de Defensa de Estados Unidos continuaba trabajando con proyectos como el GloMo (*Global Mobile Information Systems*) o el NTDR (*Near-term Digital Radio*). El objetivo del GloMo era permitir la conectividad multimedia de tipo *Ethernet*, en cualquier momento y en cualquier lugar, entre los dispositivos inalámbricos. NTDR son protocolos que se basan en dos componentes: agrupamiento y encaminamiento. Los algoritmos de agrupamiento organizan dinámicamente una red en líderes de grupo y miembros de grupo. Los líderes forman la columna vertebral de la red y los miembros se comunican entre sí a través de dicha columna. NTDR inicialmente fue un prototipo para la Armada de los Estados Unidos y en la actualidad algunos países lo utilizan como base para otros protocolos.

La definición de estándares como IEEE 802.11 [20] provocó el rápido crecimiento de las redes móviles en campos no sólo militares, sino también en el mundo comercial.

2.2. Características

Como su propio nombre indica la característica principal de una red móvil ad hoc es la movilidad de los nodos, que pueden cambiar de posición rápidamente. La necesidad de crear redes de forma rápida en lugares sin infraestructura suele implicar que los nodos exploren el área y, en algunos casos, se deban unir para

conseguir un objetivo. El tipo de movilidad que desarrollen los nodos puede tener una influencia a la hora de elegir el protocolo de encaminamiento que aumente el rendimiento de la red.

Otro de los aspectos importantes en las redes ad hoc es la llamada auto-organización que se estudia en profundidad en [15]. La idea principal se basa en la coordinación y colaboración de todos los nodos de la red para conseguir un mismo objetivo. Se han propuesto varios métodos de auto-organización para redes en general y para redes ad hoc en particular. La auto-organización puede desglosarse en las siguientes capacidades:

- *Auto-reparación*: mecanismos que permitan detectar, localizar y reparar automáticamente los fallos siendo capaces de distinguir la causa del error. Por ejemplo, sobrecarga o mal funcionamiento.

- *Auto-configuración*: métodos de generación de configuraciones adecuadas en función de la situación actual dependiendo de las circunstancias ambientales. Por ejemplo, conectividad o parámetros de calidad de servicio.

- *Auto-gestión*: capacidad de mantener dispositivos o redes dependiendo de los parámetros actuales del sistema.

- *Adaptación*: adecuación a los cambios de las condiciones ambientales. Por ejemplo, cambio en el número de nodos vecinos.

A continuación se presentan el resto de características de las redes móviles ad hoc:

- *Ausencia de infraestructura*: Al contrario que las redes convencionales que cuentan con la existencia de elementos físicos, las redes móviles se

forman autónomamente.

- *Topología dinámica*: Los nodos se pueden mover arbitrariamente haciendo que algunos enlaces se destruyan y otros se creen cuando un nodo se acerque a otros que antes tenía fuera de su alcance.

- *Ancho de banda limitado*: En la mayoría de las ocasiones será menor que el de una conexión cableada, afectado además por las interferencias de las señales electromagnéticas.

- *Variación en la capacidad de los enlaces y los nodos*: Los nodos pueden disponer de varias interfaces de radio que difieren entre sí en capacidad de transmisión/recepción y en la banda de frecuencia en la que trabajan. Esta característica complica el desarrollo de los protocolos de encaminamiento en gran medida.

- *Conservación de energía*: Algunos o todos los nodos de una red móvil ad hoc son alimentados por baterías y no tienen posibilidad de recargarlas. Para estos nodos, el criterio más importante a la hora de diseñar sistemas y protocolos será la optimización de la conservación de energía.

- *Escalabilidad*: En muchas aplicaciones las redes ad hoc pueden llegar a tener miles de nodos lo que conlleva dificultad en tareas como direccionamiento, encaminamiento, gestión de localización, gestión de configuración, interoperabilidad, seguridad, etc.

- *Falta de seguridad*: La seguridad juega un papel importante en las redes ad hoc dado el carácter vulnerable de los enlaces inalámbricos que se forman. Los protocolos de encaminamiento deben proporcionar una comunicación segura. Existen áreas de investigación en este sentido que sugieren incluir datos de sensores externos e información geográfica y

topográfica en el propio algoritmo de encaminamiento.

- *Encaminamiento multisalto*: Los nodos actúan como *routers* para retransmitir los paquetes intercambiados entre nodos cuyo alcance no permite una comunicación directa.

- *Entorno imprevisible*: Las redes ad hoc pueden darse en terrenos en los que las situaciones no son las más óptimas debido a condiciones peligrosas o desconocidas. Pueden darse casos donde los nodos se destruyan, se estropeen o comiencen a producir fallos.

- *Comportamiento de los terminales*: Una de las principales claves para que una red móvil ad hoc tenga un funcionamiento adecuado es la confianza que cada nodo debe tener sobre los demás. Sin esta confianza sería imposible crear un protocolo de encaminamiento ya que la información debe transmitirse por varios nodos intermedios. Normalmente, los protocolos de encaminamiento que descubren los terminales intermedios se basan en las respuestas que dan los nodos sobre el coste de la comunicación. Existen nodos maliciosos que podrían intencionadamente informar de forma incorrecta sobre los costes con la finalidad de recibir todos los paquetes, poder manipularlos, alterarlos o incluso eliminarlos. Algunas soluciones al respecto se encuentran en [17].

2.3. Estándar IEEE 802.11

El IEEE 802.11 es un estándar de protocolo de comunicaciones que define el uso de los dos niveles más bajos de la arquitectura OSI (capa física y capa de enlace de datos), especificando sus normas de funcionamiento en una red inalámbrica. La primera propuesta de este estándar mantenía tasas de transmisión de 1 y 2 Mbps en la banda de frecuencias ISM (*Industrial Scientific and Medical*), situada

en 2.4 GHz. Además, se especificaban como tecnologías en la capa física los infrarrojos y el canal radio. Con los años se ha llegado a distintas versiones del estándar. Se citan los más importantes a continuación:

- *IEEE 802.11a*: hasta 54 Mbps a 5 GHz. Utiliza la tecnología OFDM (*Orthogonal Frecuency-Division Multiplexing*) en la capa física.

- *IEEE 802.11b*: hasta 11 Mbps a 2.4 GHz. Actualmente es el más utilizado. Utiliza la tecnología DSSS (*Direct Sequence Spread Spectrum*) en la capa física.

- *IEEE 802.11e*: pretende proporcionar calidad de servicio QoS (*Quality of Service*) para su uso en servicios como VoIP (Voz sobre IP) y *Streaming*. Una aproximación para otorgar calidad de servicio es la de diferenciar los paquetes clasificándolos en un número pequeño de tipos de servicios y utilizar mecanismos de prioridad para proporcionar una calidad de servicio adecuada a cada tráfico.

- *IEEE 802.11f*: desarrolla especificaciones para la implementación de puntos de acceso y sistemas de distribución para evitar problemas de interoperabilidad entre distintos fabricantes y distribuidores de equipos.

- *IEEE 802.11g*: hasta 54 Mbps a 2.4 GHz. Soporta tanto OFDM como DSSS en la capa física

2.4. Clasificación

La terminología de redes ad hoc aún no está muy asentada y no existe una clasificación clara. A continuación se exponen varias clasificaciones situando el lugar en el que se encuentran las redes móviles ad hoc.

Existen redes ad hoc *con infraestructura* donde los nodos se mueven mientras se comunican con una estación base fija. Cuando un nodo se mueve fuera del rango de una estación fija entra en el alcance de otra estación. Por otro lado se encuentran las redes ad hoc *sin infraestructura* donde no existen estaciones base fijas y todos los nodos de la red necesitan actuar como *routers*. *Las redes móviles ad hoc son redes ad hoc sin infraestructura*.

Otra clasificación de las redes ad hoc incluye las *redes de un solo salto* y las *redes multisalto*. Los nodos de las redes de un solo salto se comunican únicamente con los nodos que tiene a su alcance. En las redes ad hoc multisalto, los nodos que no pueden comunicarse directamente utilizan nodos intermedios para retransmitir la información. *Las redes móviles ad hoc son redes ad hoc multisalto*.

Por último hay una clasificación que incluye las redes móviles ad hoc como un tipo independiente. Se incluyen tres tipos de redes ad hoc:

- *Redes móviles ad hoc*.

- *Redes de sensores*: También denominadas WSN (*Wireless Sensor Networks*). Formadas de dispositivos sensoriales, generalmente compuestos por un sensor tradicional y un conversor analógico-digital. La unidad de proceso está compuesta de un microprocesador y una pequeña memoria. Pueden incluir sistemas de localización y sistemas de movilidad. En estas redes el número de nodos suele ser mucho mayor que en una red móvil ad hoc pero la movilidad se considera escasa o nula (solamente cambia la topología con la pérdida o desconexión de nodos). Es habitual el flujo de información desde muchos orígenes hasta un nodo llamado sumidero (*sink*) que se encarga de procesar la información y enviársela al destino.

- *Redes híbridas*: También denominadas mixtas, son redes ad hoc que usan

infraestructuras IP si están disponibles.

A su vez las redes móviles ad hoc se pueden dividir en dos tipos en función de si están conectadas o no a otras redes:

- *Redes móviles ad hoc autónomas*: Son redes que no están conectadas a ninguna otra red. Los nodos de la red se pueden identificar unívocamente a través de una dirección IP con la única premisa de que sea distinta a la de cualquier otro nodo de la red.

- *Redes móviles ad hoc subordinadas*: Son redes conectadas a una o más redes externas. Se obliga a usar un direccionamiento IP topológico correcto y encaminable globalmente. Un ejemplo típico de red móvil ad hoc subordinada es una red móvil ad hoc que es parte de Internet.

2.5. Aplicaciones

Es fácil encontrar situaciones donde se ve la utilidad de las redes móviles ad hoc. Uno de los ejemplos más clásicos (aunque también discutido) es una reunión de trabajo: un grupo de personas con ordenadores portátiles o PDAs. Son de distintas empresas y por tanto sus direcciones son distintas. Tal vez en la sala haya acceso a Internet y puedan usar por ejemplo IP móvil, pero ¿para qué pasear sus datagramas por toda la ciudad o todo el país cuando están en la misma habitación? Sus equipos probablemente estén dotados de puertos de infrarrojos o Bluetooth que les permitan formar una red para la ocasión. En algunos casos, simplemente no habrá infraestructuras de apoyo. Pensemos en poblaciones aisladas o de orografía difícil, situaciones de emergencia, desastres naturales donde las infraestructuras hayan desparecido, etc.

Otro ejemplo son las denominadas PAN (*Personal Area Networks*): redes

formadas por los dispositivos de una persona, como su reloj, su agenda y su teléfono móvil. Una red así puede querer entrar en contacto con la red de otra persona que en ese momento esté próxima.

La capacidad de desplegarse inmediatamente y la no dependencia de un único punto de fallo hace a estas redes muy interesantes para el uso militar. El campo militar es posiblemente el más desarrollado actualmente. Así, el ejército estadounidense ya dispone de un sistema basado en este tipo de redes, el FBCB2 (*Force XXI Battle Command, Brigade-and-Below*). Uno de sus objetivos es distinguir las fuerzas propias de las fuerzas del enemigo, ofreciendo a los soldados una visión del campo de batalla similar a la de un videojuego. Los equipos de la generación inmediatamente anterior estaban basados en comunicaciones por satélite, con latencias de cinco minutos. En abril de 2003 el FBCB2 se utilizó en la Segunda Guerra del Golfo, lo que supuso probablemente el primer uso bajo fuego real de una red móvil ad hoc.

Otro motivo por el que una red móvil ad hoc puede ser ventajosa es el coste. Aunque exista una infraestructura de red, si pertenece a una entidad ajena es muy posible que nos cobre por su uso, mientras que si tenemos nuestros equipos desplegados dispondremos ya de una red sin coste adicional. Por ejemplo, los coches que pasan por una autopista podrían formar fácilmente una red móvil ad hoc, independiente de su capacidad de conectarse a otras redes como GSM o similar. Por último, supongamos que tenemos estaciones capaces de comunicarse empleando un satélite. Estos equipos de comunicaciones son caros, pero bastaría con que algunos tengan capacidad de conectarse al satélite para que todos dispusieran de conectividad. Y no todos los capaces de conectarse al satélite necesitarían estar conectados simultáneamente.

Resulta evidente que el potencial de este tipo de redes es muy grande.

3. AUTOCONFIGURACIÓN EN REDES MÓVILES AD HOC

Los nodos de una red necesitan de algún mecanismo para intercambiarse mensajes. El protocolo TCP/IP permite comunicar a los diferentes nodos de una red asociando a cada nodo de la misma una dirección IP distinta.

En redes cableadas o en redes inalámbricas con infraestructura se dispone de un servidor o de un nodo que actúa como tal que asigna correctamente las direcciones IP.

En redes móviles ad hoc no se dispone de una entidad centralizada que pueda realizar esta función. Por tanto, es necesario un protocolo que realice la configuración de la red de forma dinámica y automática, que utilizará todos los nodos de la red (o sólo una parte de ellos) como si fuesen servidores que gestionan direcciones IP.

3.1. El Problema de la Autoconfiguración

Debido a la topología dinámica de las redes móviles ad hoc (constante movimiento de los nodos que pueden entrar y salir de la red frecuentemente e incluso simultáneamente), los protocolos de autoconfiguración se enfrentan a diversos problemas para garantizar la unicidad de las direcciones IP y permitir la partición y la unión de redes.

Para garantizar el correcto funcionamiento de la red los protocolos pretenden lograr los siguientes objetivos:

- *Lograr la unicidad de las direcciones IP*: Asegurar que dos o más nodos no obtengan la misma dirección IP.

- *Funcionar correctamente*: Una dirección IP está asociada a un nodo solamente por el tiempo que permanece en la red. Cuando un nodo deja la red, su dirección IP debe quedarse disponible para ser asociada a otro nodo.

- *Solucionar los problemas derivados de la pérdida de mensajes*: En caso de que algún nodo falle u ocurra pérdida de mensajes, el protocolo debe actuar lo suficientemente rápido para evitar que dos o más nodos posean la misma dirección IP.

- *Permitir el encaminamiento multisalto*: Un nodo no se configurará con una dirección IP si no hay ninguna disponible en toda la red. De esta forma, si cualquier nodo de la red posee una dirección IP libre ésta debe asociarse al nodo que está solicitando una dirección IP, aunque esté a dos o más saltos de distancia.

- *Minimizar el tráfico de paquetes adicionales en la red*: El protocolo debe minimizar el número de paquetes intercambiados entre los nodos en el proceso de autoconfiguración. En otras palabras, el tráfico de paquetes de control debe perjudicar lo menos posible al tráfico de paquete de datos ya que, en caso contrario, el rendimiento de la red disminuiría.

- *Verificar la existencia de solicitudes concurrentes de dirección IP*: Cuando dos nodos solicitan una dirección IP en el mismo instante de tiempo, el protocolo debe realizar el tratamiento pertinente para que no sea suministrada la misma dirección IP a los dos nodos.

- *Ser flexible a la partición y a la unión de redes móviles ad hoc*: El protocolo debe poder lograr la unión de dos redes móviles ad hoc distintas así como la partición en dos o más redes.

- *Realizar la sincronización*: El protocolo debe adaptarse a los rápidos cambios de la topología de las redes inalámbricas debido a la frecuente movilidad de los nodos. La sincronización se realiza periódicamente para mantener una configuración lo más actualizada posible de la topología de la red.

3.2. Aplicabilidad de las Soluciones Estándar

La aplicabilidad de los protocolos estándar es insuficiente para redes MANET [3]. A continuación se presentan dos de estos protocolos.

3.2.1. SLAAC/NDP

SLAAC (*StateLess Address AutoConfiguration*) [49] es un estándar que permite la autoconfiguración automática de una dirección IPv6 sin necesidad de un nodo encaminador (*router*). Para ello se ayuda del protocolo NDP (*Neighbor Discovery Protocol*) [34], un estándar para transmitir los mensajes y descubrir a sus vecinos.

Un nodo crea automáticamente una dirección IPv6 uniendo su identificador de *host* (suele ser la dirección MAC) con un prefijo local conocido y realiza un proceso DAD mediante la difusión (*broadcast*) de mensajes NDP a los vecinos.

Si la dirección IPv6 no es única el proceso de autoconfiguración se detiene y se tendrá que hacer manualmente. Si por el contrario la dirección es única, deberá pedir mediante mensajes NDP el prefijo de red y, posteriormente, volverá a comprobar con DAD si su dirección IPv6 es realmente única.

La aplicabilidad de este protocolo en redes móviles ad hoc queda limitada

porque usa el protocolo NDP para enviar los mensajes y NDP asume que en la red todos los nodos están conectados entre sí. Consecuentemente, sólo soporta un único salto y lo más frecuente es que la red móvil ad hoc sea multisalto, no alcanzando a la mayoría de los nodos para realizar los procesos DAD y no pudiendo, por tanto, asegurar que la dirección IPv6 obtenida sea única.

3.2.2. DHCP-PD

DHCP-PD (*Dynamic Host Configuration Protocol – Prefix Delegation*) [51] es una opción de DHCPv6 [14] que provee un mecanismo para la delegación de los prefijos de direcciones IPv6 y permite la asignación automática de uno de estos. Para ello, un nodo que desea obtener una dirección IPv6, envía un mensaje DHCP con la opción *prefix delegation* activada para obtener un prefijo de un servidor DHCP de la red.

La aplicabilidad en redes móviles ad hoc es insuficiente porque está basado en DHCP, por lo que asume que todos los nodos pueden conectarse ya sea directamente o a través de varios saltos con un servidor DHCP, y debido a la topología de las redes móviles ad hoc, la conexión directa al servidor DHCP no suele ser frecuente con la consecuencia de que la conexión mediante varios saltos puede producir que el servidor sea inalcanzable.

3.3. Clasificación de los Protocolos de Autoconfiguración

Los protocolos de autoconfiguración pueden clasificarse dependiendo del modo de gestión de direcciones en:

- *Protocolos de estado completo* (*Stateful*): Los nodos conocen el estado de la red, es decir, mantienen tablas con las direcciones IP de los nodos.

- *Protocolos sin estado* (*Stateless*): La dirección IP de un nodo está gestionada por el mismo. Generalmente crean una dirección aleatoria y realizan un proceso de detección de direcciones duplicadas para verificar su unicidad.

- *Protocolos híbridos*: Combinan mecanismos de los dos anteriores para mejorar la escalabilidad y la fiabilidad de la autoconfiguración. El precio es un alto nivel de complejidad en los algoritmos.

3.3.1. Protocolos de Estado Completo.

3.3.1.1. Protocolo de Nesargi y Prakash (MANETConf)

MANETConf [36], que es una mejora de [35], está basado en la existencia de una tabla común distribuida a través de la cual todos los nodos son capaces de asignar direcciones IP.

Cuando un nodo quiere entrar en la red envía mensajes de difusión y al primero que le conteste lo elige como nodo iniciador y le pide una dirección IP. El nodo iniciador elige una de las direcciones IP libres que hay en la red y antes de asignarla pide permiso al resto de nodos, porque puede darse el caso de que otro nodo la haya elegido para otro o porque puede que las tablas no estén totalmente sincronizadas debido al retardo de los mensajes.

Si la respuesta de los nodos es positiva, le asigna la dirección IP al nodo entrante y lo comunica por difusión para que el resto de nodos actualicen sus tablas.

Si hay algún nodo que no responde, se pone en contacto con él directamente (*unicast*) para obtener respuesta. Si aún así no logra obtenerla, dará por hecho que el nodo ha dejado la red, y lo comunicará al resto de nodos de la red para

que actualicen sus tablas.

3.3.1.2. Protocolo de Moshin y Prakash

El esquema proactivo de Mohsin y Prakash [31] trata de solucionar el problema de la asignación de direcciones IP mediante la división binaria de bloques de direcciones libres.

Estas divisiones se hacen en potencias de 2. De esta forma pueden existir nodos con varios o ningún bloque. Además, cada nodo de la red dispone de una tabla con el estado de todos los nodos de la red, es decir, conoce los bloques de direcciones libres de los otros nodos, así como la dirección IP de la interfaz de red de cada nodo.

El proceso de asignación de direcciones puede proceder de cualquier nodo. Siguiendo esta idea, un nodo no configurado (nodo cliente) envía un mensaje de difusión del tipo REQUEST para que un nodo de la red lo configure. Al ser un mensaje de difusión podrían producirse varias respuestas, pero elegirá al nodo que realice la primera respuesta (REPLAY), y este actuará como nodo servidor. Si este nodo dispone de varios bloques, le entregará uno al nodo cliente, y elegirá la primera IP del bloque como propia. Si por el contrario dispone de un bloque, entonces lo dividirá en 2 partes iguales y le entregará una mitad al nodo cliente, y la otra se la quedará como propia.

Si se diera el caso en que no dispusiera de ningún bloque se proponen ciertas soluciones basadas en la idea de que el servidor busque en sus vecinos próximos si alguno tiene algún bloque disponible. En caso de no encontrar direcciones libres intentará obtener un bloque en vecinos de un salto mayor, y así sucesivamente. Otra solución es buscar el nodo que tenga mayor rango de direcciones libres, para entregar la mitad de este bloque al nodo que está entrando en la red.

La entrada y salida de los nodos puede ser de forma abrupta o voluntaria, para cada caso hay un esquema a seguir. Si la salida es abrupta, el nodo que actuó como servidor en su configuración verá en su tabla de rutas que el bloque del nodo que abandonó la red no está y añadirá el bloque a su bloque de direcciones libres. En el caso de una salida voluntaria el nodo que sale, notifica a algún vecino suyo la intención de marcharse y el vecino busca al nodo que configuró al cliente para que se quede con su bloque.

La mayor ventaja de este protocolo es que funciona bien para unión y división de redes, ya que soluciona el problema de las direcciones duplicadas que se producen en estos casos. Cada nodo que inicia la red genera un número aleatorio llamado *PartitionID* que será un número identificativo de la red. De esta manera cuando se produce una división o partición de la red, el primer nodo que se retire de la red original creará otro *PartitionID*. En el momento en que dos redes con *PartitionID* diferentes se unan, primero se comprobará la consistencia de sus direcciones IP. Este proceso consiste en comprobar la existencia de dos nodos con una misma dirección. En caso afirmativo se produce un cambio del nodo perteneciente a la red con menor rango de direcciones IP libres. La nueva dirección IP será perteneciente al mayor rango de la red con mayor número de direcciones libres.

El mayor inconveniente de este protocolo es que la sincronización depende de la existencia de un *broadcast* confiable y en un entorno distribuido móvil tal cosa no existe, por lo que uno puede plantearse dudas sobre la robustez de este protocolo.

3.3.1.3. Protocolo de Thoppian y Prakash

Una mejora del esquema anterior (particularmente en lo relativo a la sincronización) puede encontrarse en [50], donde Thoppian y Prakash proponen una asignación dinámica de direcciones basada en el denominado

buddy system (sistema de amigos) que maneja movilidad de los nodos durante la asignación de direcciones, pérdida de mensajes, partición y unión de redes. Sin embargo, la asignación de direcciones IP puede generar alta sobrecarga de los mensajes de control mientras hace una búsqueda global y la recuperación de direcciones (para evitar pérdidas de direcciones) requiere de mensajes de difusión por inundación (*flooding*). Además, la unión y partición pueden incurrir en una alta sobrecarga debido a la naturaleza global de este protocolo.

3.3.1.4. EMAP

EMAP (*Extensible Manet Autoconfiguration Protocol*) [45] es un protocolo de autoconfiguración que se basa en la idea del protocolo de mensajes REQUEST/REPLAY para su funcionamiento.

La principal ventaja de este protocolo es la posibilidad de hacerlo extensible, es decir, que en el futuro se pueden incluir nuevas funcionalidades que quedan tratadas de forma teórica como el descubrimiento de servidores DNS (*Domain Name Server*).

Este protocolo trata también la posibilidad de comunicaciones exteriores a la red móvil ad hoc a través de Internet.

El mecanismo de descubrimiento de rutas entre nodos sigue la línea del protocolo AODV (*Ad Hoc On-Demand Distance Vector*) [41].

La principal idea de este protocolo de autoconfiguración es la de poseer diferentes direcciones por parte de un nodo no configurado que va a unirse a la red ya creada. De esta manera existen tres direcciones para las comunicaciones interiores: la dirección temporal (*temporary address*), la dirección tentativa (*tentative address*), y la dirección local de red móvil ad hoc (*mobile ad hoc network local address*).

Cuando un nodo quiere entrar en la red genera aleatoriamente dos direcciones IP válidas de la red (con dirección de red conocida), y las considera como dirección temporal y dirección tentativa. Estas direcciones IP se encapsulan en el mensaje DAD_REP (*Detection Adress Detection REsPonse*) para saber si es una dirección válida. El nodo queda esperando un DAD_REQ (*Detection Adress Detection REQuest*). Si transcurre el tiempo de espera para este mensaje el nodo asume que puede usar su dirección tentativa como única, y se la asigna a su interfaz de red. Si este nodo recibe un mensaje DAD_REP a su dirección temporal, y este mensaje contiene el origen con la dirección tentativa que había propuesto, sabe que esta dirección tentativa está siendo usada y comienza nuevamente el proceso anterior creando otro par de direcciones.

3.3.2. Protocolos sin Estado

3.3.2.1. Proceso de Detección de Direcciones Duplicadas

La Detección de Direcciones Duplicadas (*Duplicate Address Detection*, DAD) es un proceso que utilizan los protocolos para comprobar la unicidad de las direcciones IP. Este proceso requiere un tiempo relativamente largo para completarse, por lo que se han implementado diferentes soluciones que lo reducen.

Existen tres tipos de procesos DAD:

- **SDAD:** *Strong Duplicate Address Detection* [43] es la base de los protocolos sin estado. Consiste en un sencillo mecanismo en el que el nodo elige dos direcciónes IP, una temporal y una tentativa.

  La dirección temporal sólo la usará para la inicialización mientras detecta si la tentativa es única o no. El método de detección consiste en enviar un mensaje ICMP destinado directamente a esa dirección. Si recibe

respuesta, esa dirección IP está siendo usada por lo que reanudará el proceso. Si no recibe respuesta, enviará el mensaje un número determinado de veces para asegurarse de que es única.

Al ser un mecanismo muy sencillo, no asegura la unicidad de la dirección IP ya que el proceso se limita sólo a la fase de inicialización, y para desconexiones temporales o pérdida de la red no funcionaría. Además, cuando la red es grande y quedan pocas direcciones IP libres, añade mucha sobrecarga hasta que encuentra una dirección IP única.

- **WDAD:** *Weak Duplicate Address Detection* [52] establece la idea de tolerar durante un tiempo las direcciones duplicadas en la red. Para ello, cada nodo al iniciarse creará una clave que enviará siempre junto a su dirección IP. Cuando un nodo reciba un mensaje comprobará en su tabla si esa dirección IP ya está asignada y mirará si las claves coinciden, si no coinciden, marcará esa dirección como inválida y se tomarán acciones para que sean únicas (estas acciones no están definidas en WDAD).

  Este proceso tiene que soportar la identificación de un nodo mediante un par clave-IP y depende totalmente del protocolo de encaminamiento. Sólo funcionará con un proactivo que actualiza las rutas constantemente, pero con un reactivo habrá nodos que nunca puedan detectar la duplicidad de direcciones IP.

  No añade sobrecarga adicional al protocolo de encaminamiento, pero en cambio si añade la sobrecarga de enviar siempre junto a la dirección IP la clave.

- **PDAD:** En *Passive Duplicate Address Detection* [55] la idea se basa en que en vez de detectar o resolver direcciones IP duplicadas enviando información de control, cada nodo investiga y deduce si existe una

dirección duplicada por eventos que nunca ocurrirían si todas las direcciones IP fueran únicas.

Se proponen tres detecciones pasivas, que son necesarias unir para el correcto funcionamiento de la detección:

- PDAD-SN (*Sequence Numbers*). Este sistema se basa en la idea de que los protocolos de encaminamiento usan números de secuencia en sus mensajes para actualizar las rutas. Usando estos números de secuencia, y la idea de que dos nodos con una distancia entre ellos de dos saltos no tienen el mismo vecindario, se solucionan algunos conflictos. Además, se tiene en consideración la posibilidad de que estos números de secuencia lleguen al máximo y empiecen a producirse números desde cero de nuevo.

- PDAD-LP (*Locality Principle*). De menor potencia que el anterior, se basa en la frecuencia de actualización de las tablas de rutas. En función de esta frecuencia se pueden detectar direcciones duplicadas tomando un umbral de tiempo para visualizar el estado de las tablas de rutas. Hay que tener en cuenta el protocolo de encaminamiento usado. Debemos considerar diferentes umbrales, ya que se puede dar el caso de que dos mensajes con el mismo origen se confundan con una dirección duplicada, si el tiempo es demasiado corto y, por consiguiente, el protocolo modifica las rutas demasiado rápido.

- PDAD-NH (*Neighborhood*). Teniendo en cuenta que un nodo conoce sus vecinos, y los de un nodo que haya enviado un paquete del estado de su enlace, diferencia si hay conflicto o no en función de si en un paquete, el origen de este mensaje es un vecino, y contiene la dirección.

La ventaja es que no añade sobrecarga a la red adicional, pero solamente se puede utilizar con protocolos de encaminamiento proactivos.

3.3.2.2. APAC

APAC (*Agent based Passive Autoconfiguration*) [24] es un protocolo de autoconfiguración basado en PDAD. Su característica principal es el uso de ciertos nodos que centralizan el reparto de direcciones.

El mecanismo por el que un nodo configura su dirección IP al entrar en la red consiste en preguntar si tiene a un salto de distancia algún nodo de tipo AA (*Address Agent*). En ese caso, el nodo AA le proporcionará una dirección IP.

En caso de no tener respuesta de ningún nodo AA, el nodo entrante se configura para funcionar en modo AA, y hará de servidor de direcciones para los próximos nodos que lleguen. Cuando se configura como AA, el nodo genera aleatoriamente un número identificador *agentID*, para poder formar una tabla con las direcciones que asignará a los nodos que lleguen. Esas direcciones son de la forma *agentID+hostID*.

Cuando un nodo se mueve en la red y sale del radio de cobertura del AA que le proporcionó su dirección IP, deberá pedir otra dirección a otro nodo AA que tenga dentro de su nuevo radio de cobertura. Esto se complementa con un mecanismo para no interrumpir las comunicaciones en curso.

Al darse esta situación, el AA anterior marcará su dirección IP como libre para poder ser asignada a algún otro nodo más adelante.

La detección de direcciones duplicadas se realiza mediante el proceso PDAD. Una vez detectado algún conflicto, se informa al AA que asignó esa dirección IP conflictiva. Ese nodo AA entonces generará un nuevo *agentID* y avisará a todos

los nodos dependientes de él para que cambien su dirección del tipo *agentID+hostID* al nuevo *agentID*.

En cuanto a la división y unión de redes, se utiliza el mismo mecanismo explicado anteriormente. En caso de división, los nodos AA marcarán las direcciones que hayan salido de la red como libres. Y en el caso de unión de dos redes, el mecanismo para la detección de direcciones duplicadas sigue funcionando correctamente.

3.3.2.3. AROD

En AROD (*Address autoconfiguration with address Reservation and Optimistic duplicated address Detection*) [26], la reserva de direcciones se basa en la existencia de unos nodos que tienen una dirección IP reservada para entregársela a los nodos que entren nuevos.

Existirán dos tipos de nodos:

- Agentes tipo 1 con una dirección IP reservada, aparte de las direcciones IP que tengan sus interfaces de red. Cuando un nodo entra en la red, se le asignará esta IP reservada inmediatamente.

- Agentes tipo 2, que no tienen direcciones IP reservadas. Si un nodo que entra nuevo le pide una dirección IP a uno de estos, pide prestada la dirección reservada de uno de sus vecinos que sea de tipo 1, y se la asigna al nuevo inmediatamente.

Establece un mecanismo para que la red no se quede sin nodos de tipo 1, cada vez que se asigna una dirección IP a un nuevo nodo, el nodo que ha entregado la dirección genera dos direcciones IP aleatorias (una para sí mismo y otra para el nuevo nodo que ha entrado) y se hace un proceso DAD para

detectar si esas direcciones son únicas.

Una vez realizado el proceso se pueden dar las siguientes posibilidades:

- Las dos direcciones IP son únicas, por lo tanto, los dos nodos se convierten en nodo tipo 1.

- Si sólo una es única, se convertirá en nodo tipo 1 el que dio la dirección IP.

- Si ninguna es única, los dos nodos se quedarán como tipo 2.

Este protocolo considera su proceso de detección de direcciones duplicadas como un proceso DAD optimista, porque solamente se lleva a cabo una vez cuando entra un nodo y se le asigna una dirección IP reservada.

Contempla la posibilidad de cambiar el número de direcciones IP reservadas para los nodos tipo 1. Si aumenta el número, la latencia de asignación de direcciones IP es menor, pero por el contrario aumenta la sobrecarga al tener que realizar más procesos DAD, y viceversa. Esta posibilidad permite variar latencia versus sobrecarga.

3.3.3. Protocolos Híbridos

3.3.3.1. HCQA

HCQA (*Hybrid Centralized Query-based Autoconfiguration*) [46] fue el primer protocolo de autoconfiguración híbrido.

Un nodo que quiera entrar en la red realiza un proceso SDAD. Si el proceso es exitoso, el nodo deberá registrar su dirección IP tentativa con una AA (*Address Authority*). Para ello, esperará un mensaje del AA y cuando reciba ese

mensaje enviará una petición de registro y el AA se lo confirmará. El nodo al comenzar todo este proceso inicia un contador, si ese contador expira, volverá a empezar de nuevo el proceso hasta poder registrar la dirección IP.

Cuando se crea la red, el primer nodo se convierte en AA, elige un identificador para la red único (por ejemplo la dirección MAC) y mediante mensajes de difusión la anuncia periódicamente para identificar la red. Si algún nodo no la recibe se considera que la red ha sido dividida y creará su propia red convirtiéndose en AA.

Este protocolo añade robustez al proceso SDAD y garantiza la no duplicidad de las direcciones IP y a la vez proporciona un buen mecanismo de partición de redes.

Pero tiene dos principales problemas, primero la sobrecarga producida por el proceso SDAD y los mensajes periódicos del AA, y segundo, que la red depende de una entidad central con la que todos los nodos deben comunicarse directamente para poder registrar su dirección IP, con lo que se añade mucha latencia en la entrada de los nodos a la red.

3.3.3.2. PACMAN

PACMAN (*Passive Autoconfiguration for Mobile ad hoc Networks*) [54] es un protocolo de autoconfiguración pasivo para redes MANET.

Utiliza elementos de protocolos con y sin estado, por lo que podría considerarse híbrido en cierto modo.

Su funcionamiento se basa en que cada nodo se asigna a sí mismo una dirección al entrar a la red, y en la monitorización pasiva de las comunicaciones para la detección de direcciones duplicadas.

Para conseguir la mínima sobrecarga en las comunicaciones, se comparte información entre las diferentes capas de red. En concreto se monitoriza la información que maneja el protocolo de encaminamiento.

El método usado para elegir la propia dirección IP que se usará sigue un algoritmo probabilístico. Para intentar que la probabilidad de elegir una dirección IP actualmente en uso por otro nodo sea cercana a cero, este algoritmo tiene en cuenta, entre otros factores, una tabla de asignaciones. Esta tabla se crea con información extraída del protocolo de encaminamiento sobre las direcciones IP que ya están en uso.

PACMAN usa el proceso PDAD para monitorizar las comunicaciones en busca de direcciones duplicadas. Esto es necesario debido a que el mecanismo utilizado para la asignación de direcciones no garantiza unicidad (aunque intenta reducir la probabilidad de colisión), y a que pueden producirse uniones de redes que contengan nodos con las mismas direcciones IP.

A grandes rasgos hay dos tipos de eventos que indican duplicidad de direcciones IP:

Por un lado están los eventos que nunca ocurren si la dirección es única, y siempre si la dirección está duplicada. Estos eventos confirman que hay un problema detectado.

Y por otra parte tenemos los eventos que ocurren pocas veces si la dirección es única, y a menudo si la dirección está duplicada. De este modo se detecta la posibilidad de que existan problemas, por lo que se trata de algoritmos probabilísticos.

Cuando se detecta que dos nodos están usando la misma dirección IP, se le notifica el problema a uno de ellos mediante un mensaje *unicast* para que

cambie su dirección.

Además, se tiene en cuenta el problema que surge cuando se cambia una dirección que tiene alguna comunicación en marcha. Para solucionarlo, al cambiar de dirección un nodo avisa a los nodos con los que tiene comunicaciones en curso de su nueva dirección IP, para que hagan un encapsulamiento de los mensajes adecuadamente.

3.4. Análisis detallado del Protocolo de Moshin y Prakash

Se trata de un protocolo de autoconfiguración [31] para redes móviles ad hoc que garantiza la unicidad de las direcciones IP usadas en la red usando como base la idea de la división binaria de direcciones.

En este protocolo, todos los nodos de la red son los encargados de asignar direcciones IP a los nuevos nodos que entran. Cada uno tiene asignadas una serie de direcciones IP que puede asignar a los nuevos nodos. También es responsabilidad de todos los nodos el mantener actualizada la información sobre las direcciones IP libres disponibles propias y del resto de nodos de la red. Mediante un sistema de sincronización simple, los nodos son capaces de mantener unicidad en las direcciones IP asignadas, garantizando la no duplicidad.

El protocolo especifica cómo actuar en los sucesos de entrada o salida de nodos de la red, así como en la partición y en la unión de redes.

3.4.1. Estructuras de datos

Cada nodo debe mantener dos tablas que se irán actualizando constantemente mediante un mecanismo de sincronización para poder reflejar el estado de la

red actual. Recordemos que la topología dinámica de este tipo de redes conlleva sucesos como la entrada y salida de nodos, o la partición y unión de redes.

- *Free_IP_blocks*: Tabla que contiene los bloques de direcciones IP libres para atender a los nuevos nodos. Cada bloque almacenado debe cumplir la condición de ser disjunto del resto de bloques de la red. En esta tabla también se almacenará la dirección IP del propio nodo.

- *Allocated_IP_blocks*: Tabla que almacena la topología de la red. Cada entrada de la tabla se refiere a un nodo, almacenando los siguientes datos:

    - *Dirección IP*: Dirección IP asignada a la interfaz.

    - *Free_IP_blocks*: Bloques de direcciones IP libres (indicando en qué bloque se encuentra la dirección IP de la interfaz).

    - *TimeStamp* (TS): Antigüedad del nodo, necesaria a la hora de recuperar direcciones que algún nodo deja disponibles al abandonar la red.

    - *Father*: Dirección IP del nodo que hizo de servidor cuando entró a la red.

Además cada nodo debe conocer el identificador de la red, *Partition ID* (PID), generado por el nodo que creó la red. Este PID es necesario en la partición y unión de redes.

3.4.2. Formato de los mensajes

En la tabla 3.1. se listan los mensajes de este protocolo y su funcionalidad.

MENSAJE	DESCRIPCIÓN
ADDR_REQ	Mensaje que el nodo cliente envía al servidor para solicitar una dirección IP.
ADDR_REP	Respuesta del servidor al nodo cliente cuando recibe el mensaje de solicitud de dirección IP.
SERVER_POLL	El nodo cliente envía ese mensaje a un nodo servidor para indicarle que lo eligió como servidor para la configuración de su dirección IP.
IP_ASSIGNED	Mensaje enviado por el nodo servidor al nodo cliente asociando la dirección IP.
IP_ASSIGNMENT_OK	Mensaje enviado por el nodo cliente al nodo servidor confirmando que el proceso de autoconfiguración fue realizado con éxito.
NODE_UP	Mensaje enviado para notificar la entrada de un nuevo nodo, de forma que el resto de ellos puedan actualizar las tablas.
NODE_UP_REPLY	Mensaje de confirmación de actualización de tablas tras la entrada de un nodo.
GRACEFULL_DEPARTURE	Mensaje enviado para entregar las direcciones libres que deja un nodo al salir avisando a la red.
NODE_DOWN	Mensaje que se envía a todos los miembros de la red para notificarles la desaparición de un nodo.
NODE_DOWN_REPLY	Mensaje de respuesta al NODE_DOWN para indicar que los cambios en las tablas han sido realizados.

Tabla. 3.1. Descripción de los mensajes del protocolo de Moshin y Prakash.

3.4.3. Funcionamiento general

Cada nodo guarda en sus estructuras de datos, entre otros, una serie de bloques de direcciones IP libres. Estos bloques son la base fundamental en la que se sostiene el mecanismo de asignación de direcciones por división binaria.

La Figura 3.1. muestra cómo se realiza la división binaria de esos bloques de direcciones libres, partiendo de un nodo que creó la red con 256 direcciones IP.

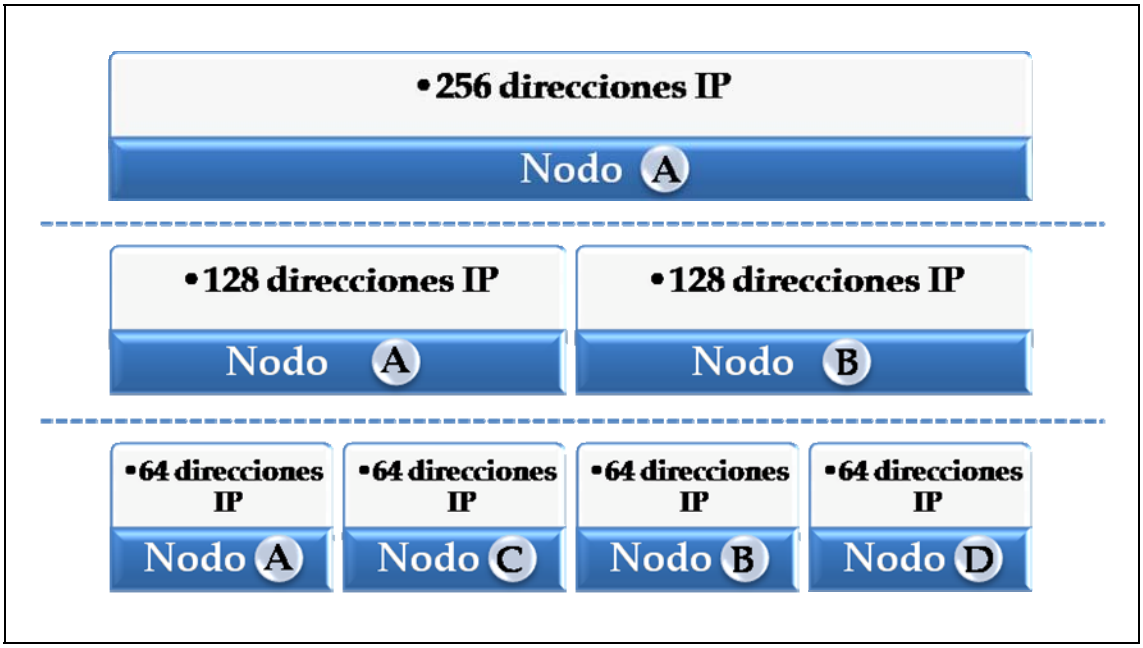

Fig. 3.1. Funcionamiento del modelo de división binaria.

Inicialmente, el nodo A posee todo el rango de direcciones IP, menos una que está asociada a su interfaz de red, por lo que tiene un bloque de 255 direcciones IP libres para entregar a otros nodos.

Cuando el nodo A recibe una solicitud de entrada a la red del nodo B, divide su conjunto de direcciones IP libres en dos mitades, suministrando la mitad para el nodo B y quedándose con la otra mitad. Ahora, el nodo A y el nodo B tienen cada uno 128 direcciones, siendo una dirección IP para su interfaz de red y 127 para servir a otras peticiones.

Usando el mismo mecanismo, el nodo A divide su conjunto de direcciones IP libres en dos mitades nuevamente, suministrando la mitad para el nodo C y quedándose con la otra mitad. El mismo procedimiento ocurre con el nodo B que atiende una petición del nodo D.

El mecanismo de división binaria asegura que todos los nodos de la red posean conjuntos disjuntos de direcciones IP, evitando que una misma dirección IP pueda ser usada por dos o más nodos aún cuando se produzca la unión de dos redes ad hoc. Es decir, asegura completamente la unicidad de las direcciones IP.

Tomando como base este modelo de distribución de direcciones, se muestran a continuación todos los escenarios que pueden tener lugar teniendo en cuenta la topología dinámica de las redes MANET.

3.4.3.1. Inicialización de la red MANET

La inicialización de un nodo consiste en el intento de conexión a una red y, si ésta no existe, en la creación de la suya propia.

Para asociarse a una red un nodo envía mensajes ADDR_REQ (*address_request*) solicitando una dirección IP y un bloque de direcciones libres. Si en un número definido de intentos no recibe respuesta alguna, creará una red propia y generará un PID que quedará asociado a la nueva red. De este modo se convertirá en el nodo líder que poseerá todas las direcciones libres de la red. Para su interfaz reservará la primera dirección.

3.4.3.2. Entrada de nodos

Cuando un nodo desea unirse a la red, con el objetivo de obtener una dirección IP envía un mensaje ADDR_REQ por difusión. Estos primeros mensajes se

envían usando la capa MAC, ya que no se dispone aún de una dirección IP para el nuevo nodo. Cualquier nodo perteneciente a la red responde a la petición del nodo cliente enviando un mensaje ADDR_REP (*Address Response*). Este mensaje contiene el mayor bloque *Free_IP_Blocks* de entre los que dispone, ya que el nodo puede poseer más de uno con diferente número de direcciones IP.

El nodo cliente puede recibir más de un mensaje proveniente de diferentes nodos, y elegirá como nodo servidor al que le haya enviado el mayor *Free_IP_Blocks* enviándole un mensaje SERVER_POLL.

Una vez se haya recibido el mensaje SERVER_POLL del nodo cliente confirmando la intención de obtener una dirección IP, el nodo servidor divide su *Free_IP_Blocks* por la mitad, suministrando una mitad al nodo cliente y quedándose la otra mitad para atender futuras peticiones.

Esta mitad del bloque es enviada a través del mensaje IP_ASSIGNED. El nodo cliente asocia la primera dirección IP de este bloque recibido como su dirección de red, y marca al bloque como el *Free_IP_Blocks* que almacena su propia dirección IP

Para confirmar que la configuración fue realizada con éxito, el nodo cliente envía un mensaje IP_ASSIGNMENT_OK al nodo servidor.

A continuación informa a cada uno de los nodos de la red su presencia en la misma enviando un mensaje NODE_UP a cada uno las veces que sean necesarias, hasta recibir las correspondientes respuestas de confirmación NODE_UP_REPLY de cada nodo. Estos mensajes ya se envían mediante datagramas IP.

El nodo cliente establecerá en sus estructuras de datos internas como *father* al nodo que le entregó el *Free_IP_Blocks*.

La figura 3.2. muestra el intercambio de mensajes durante el proceso de la entrada de un nodo en la red.

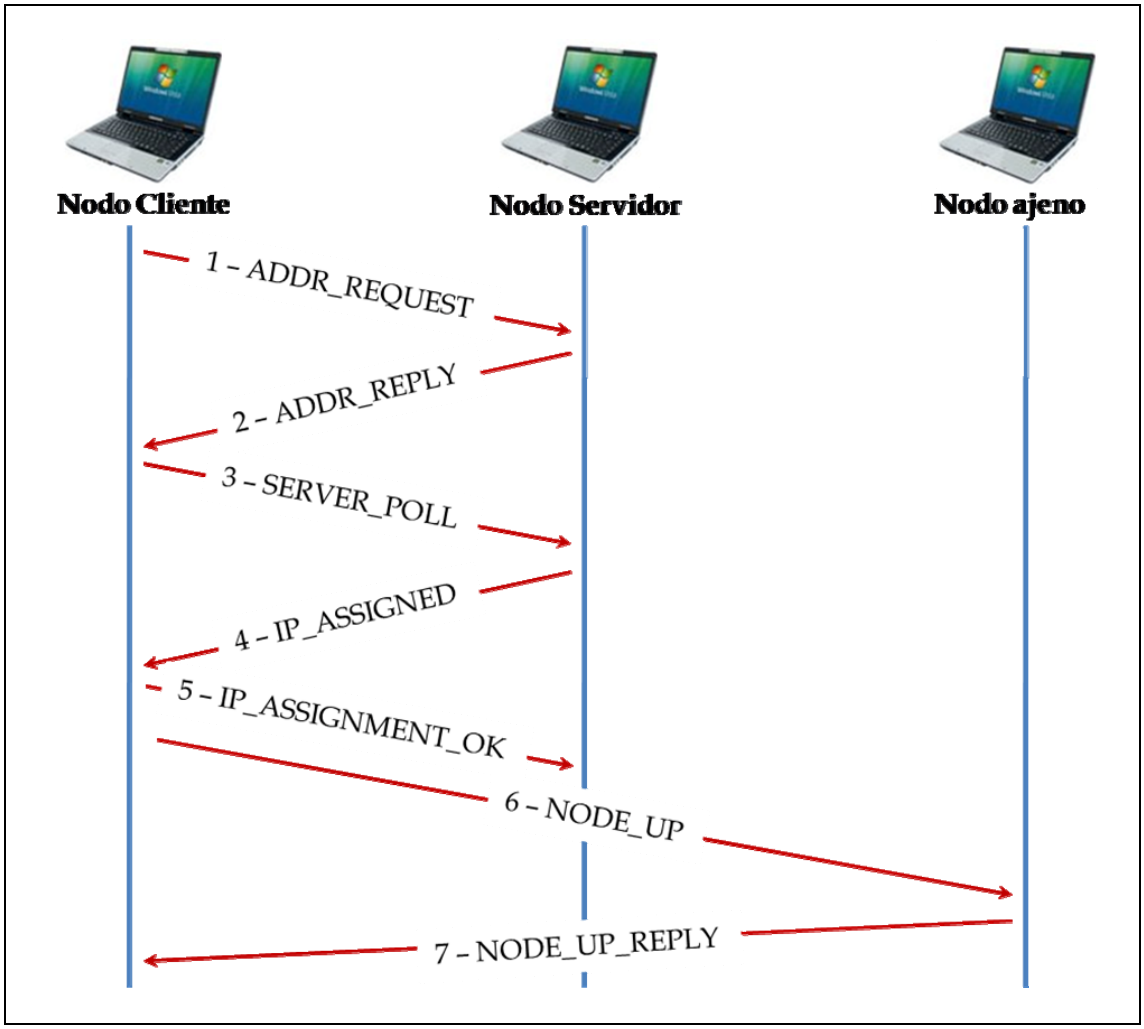

Fig. 3.2. Intercambio de mensajes en el proceso de entrada de un nodo en la red.

Puede darse el caso de pérdidas de mensajes en el proceso de entrada de los nodos. Si el nodo servidor no recibe el IP_ASSIGNMENT_OK, manda un mensaje ping al nodo cliente para verificar que fue definitivamente configurado. Si recibe respuesta, eso implica que el proceso de autoconfiguración fue realizado con éxito Por lo tanto, el mensaje de confirmación se perdió.

Por otro lado, si el nodo cliente no recibe algún mensaje NODE_UP_REPLY, comprobará que los nodos siguen aún en la red mediante el envío de mensajes ping, y si es así reenviará el mensaje NODE_UP, hasta recibir respuesta.

Es posible que un nodo servidor no tenga ninguna dirección IP disponible en su *Free_IP_Blocks* al recibir un mensaje de petición de un nodo entrante. La solución es encaminar la petición hacia los nodos vecinos en la red. Para mantener una distribución uniforme de las direcciones IP, el nodo servidor mira en su tabla *allocated_ip_blocks* para buscar cuál es el nodo que posee mayor número de direcciones libres.

Si ninguno de los nodos posee una dirección IP libre, el nodo servidor responde al cliente enviando un mensaje DENY informando que no hay direcciones IP disponibles en ese momento.

3.4.3.3. Salida de nodos

La salida de nodos en este protocolo se trata de dos maneras diferentes, diferenciando entre salida fácil y abrupta. Una salida abrupta está causada por ejemplo por la desconexión de un nodo, el agotamiento de la batería o simplemente el desplazamiento hacia una zona fuera del rango de cobertura de la red.

3.4.3.3.1. Salida fácil

Cuando un nodo desea abandonar la red lo notifica facilitando la tarea de recuperar las direcciones que dejará libres. El proceso se puede observar en la figura 3.3.

Para realizar la notificación comprueba si su padre se encuentra en la red. Si su padre responde a mensajes ping, entonces le envía un mensaje

GRACEFUL_DEP y abandona la red.

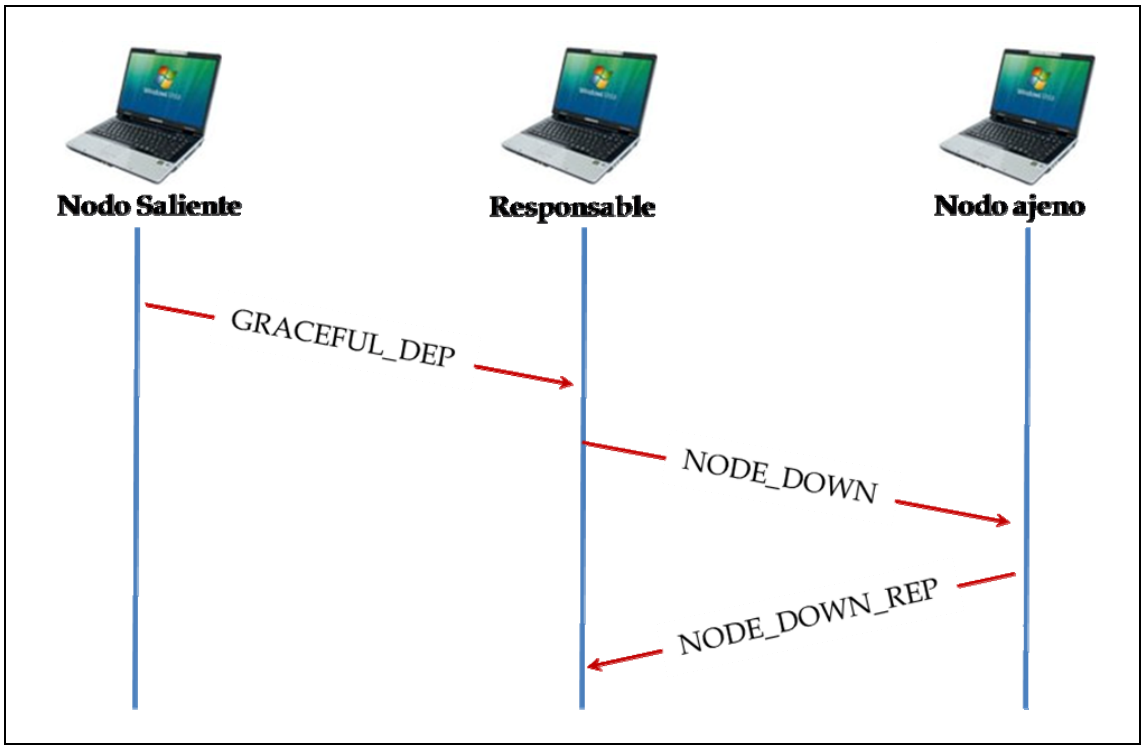

Fig. 3.3. Intercambio de mensajes en el proceso de salida fácil de un nodo.

Si su padre no se encuentra en la red, el nodo responsable de recoger sus direcciones es el nodo con mayor antigüedad. Por lo tanto, es a él a quien envía el mensaje GRACEFUL_DEP.

Tanto si es el padre como si es el nodo de mayor antigüedad, el nodo responsable recoge las direcciones IP y actualiza su tabla con los nuevos cambios. Después manda un mensaje NODE_DOWN a todos los nodos de la red avisando que un nodo ha abandonado la red. De este modo todos actualizan las tablas, y para confirmarlo responden con el mensaje NODE_DOWN_REP. Si algún nodo no responde con este mensaje, se reenvía el mensaje NODE_DOWN hasta que todos los nodos tengan constancia del cambio.

Si se pierde algún mensaje de salida, se considerará salida abrupta.

3.4.3.3.2. Salida abrupta

Si un nodo sale de una forma abrupta no tiene ocasión de avisar a ningún otro de su salida. Por lo tanto sus *Free_IP_Blocks* se perderían si no se tuviese un mecanismo diseñado para estas situaciones.

Los nodos comprueban periódicamente que su nodo padre y sus nodos hijos siguen en la red mediante mensajes ping.

Si los nodos hijos han abandonado la red, el nodo recupera las direcciones que le asignó.

Por otra parte, si es su nodo padre el que ha abandonado la red, el nodo se encargará de sus direcciones sólo si es el nodo más antiguo de la red. Notificará mediante mensajes NODE_DOWN la caída del nodo, para que el resto de integrantes de la red mantengan sus tablas actualizadas. El intercambio de mensajes en este proceso puede observar en la Figura 3.4.

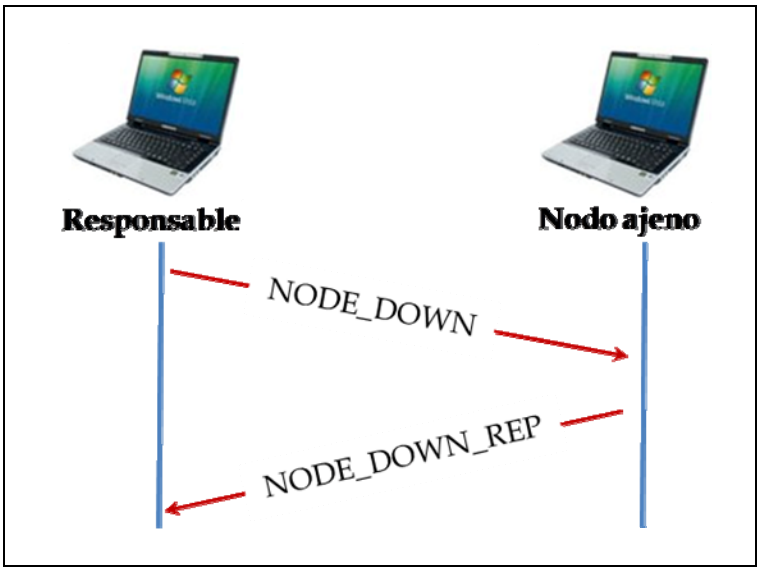

Fig. 3.4. Intercambio de mensajes en la salida abrupta de un nodo.

3.4.3.4. Sincronización

Para que todos los nodos tengan actualizadas las tablas que representan el estado de la red, es necesario un proceso de sincronización que permita detectar los cambios producidos en esta.

Cada nodo dispone de una tabla *Allocated_ip_block*, que contiene información de todos los nodos de la red: los padres de cada uno, sus direcciones libres, y su TS. Tiene que ser igual en todos los nodos, y ante posibles cambios en la red, debe ser actualizada en el menor tiempo posible.

En el caso de entrada de nodos en la red la sincronización se realiza enviando los mensajes NODE_UP y NODE_UP_REPLY.

Cuando se produce una salida fácil se informa de la salida mediante el mensaje GRACEFUL_DEP, y el nodo que recogerá sus direcciones IP avisará al resto de nodos mediante el mensaje NODE_DOWN.

Si se da el caso de que la salida es abrupta, también puede detectarse ya que se dispone de un temporizador de sincronización para que cada nodo realice una comprobación de los nodos de los que es responsable cada cierto tiempo. Cada nodo se responsabiliza de los nodos a los que ha entregado direcciones libres, y de su nodo *father*. En caso de que un nodo no tenga a otro responsable de él, lo será el nodo más antiguo de la red. Si en una de estas comprobaciones mediante mensajes ping un nodo comprueba la desaparición de otro, realizará la sincronización notificando al resto de la caída del nodo.

3.4.3.5. Partición y fusión de redes

El protocolo soporta la partición y fusión de redes. El principio que rige este funcionamiento es el que cada red tiene un PID. Cuando un subgrupo de la red

la abandona, esto es, se produce una partición, el mecanismo de actualización detectará el cambio como si se tratase de varias salidas abruptas al mismo tiempo. Por lo tanto, la sincronización está resuelta en el caso de división de redes.

Al fusionarse dos redes, se dispone de un mecanismo que intenta resolver los conflictos que se puedan dar (direcciones duplicadas). Se basa en el PID y el campo TS de cada nodo para detectar cuando dos nodos diferentes comparten dirección IP.

4. ENCAMINAMIENTO EN REDES MÓVILES AD HOC

4.1. Protocolos de Encaminamiento en Redes Móviles Ad Hoc

En redes móviles ad hoc los protocolos convencionales o bien tendrán un rendimiento muy pobre, o bien serán simplemente inaplicables. Como alternativa se desarrollan protocolos específicos de encaminamiento. Con frecuencia se les denomina de nivel 2.5, ya que es habitual encontrarlos por encima de protocolos de enlace como IEEE 802.11 y por debajo del protocolo de red IP.

El concepto de encaminamiento básicamente comprende dos actividades. En primer lugar, determinar los caminos óptimos y, en segundo lugar, transferir los grupos de paquetes de información a través de la red. Los algoritmos utilizan varias métricas para calcular el mejor camino para que los paquetes lleguen a su destino. Estas métricas son medidas estándar como podría ser el número de saltos que son usados por el algoritmo para determinar el camino óptimo. El proceso para determinar el camino inicializa y mantiene tablas de encaminamiento que contienen la información total de cada ruta. La información que se almacena para cada ruta varía de un algoritmo a otro.

Las redes móviles ad hoc se construyen de forma dinámica cuando un conjunto de nodos crean rutas entre sí para conseguir la conectividad entre ellos. Los nodos de la red móvil ad hoc pueden actuar como origen o destino de una comunicación, pero también como *routers* cuando una relación entre nodos no se puede realizar directamente por motivos de alcance. De esta forma se crean comunicaciones multisalto. Un protocolo de encaminamiento de una red móvil ad hoc necesita proveer un mecanismo que mantenga las rutas hacia los

destinos frente al movimiento de los nodos que puede provocar que las rutas se destruyan, y sea necesario encontrar una ruta alternativa para mantener la comunicación entre los nodos.

El objetivo de un protocolo de encaminamiento para redes móviles es conseguir el envío de un mensaje de un nodo a otro sin existir un enlace directo. La mayoría de protocolos de encaminamiento para redes móviles ad hoc provienen de adaptaciones realizadas sobre protocolos de redes fijas, siendo su principal problema la cantidad de fallos que se producen en la comunicación debido a la movilidad de los nodos.

Los protocolos de encaminamiento para redes móviles ad hoc deben satisfacer básicamente los siguientes criterios [4]:

- *Señalización mínima*: La reducción de los mensajes de control ayuda a conservar la capacidad de las baterías y la comunicación de los nodos.

- *Mantenimiento dinámico de topología*: El algoritmo deberá ser capaz de localizar una nueva ruta rápidamente cuando se rompe un enlace.

- *Libre de bucles*: Se pretende evitar el problema de tener paquetes circulando perdidos por la red.

- *Capacidad multisalto*: Debe asegurarse el reenvío de paquetes a través de los nodos de la red dado que habitualmente el destino no se encuentra dentro del alcance de la fuente.

- *Tiempo de procesamiento mínimo*: Se requieren algoritmos con cálculos computacionales que no sean excesivamente complejos para disminuir el tiempo de procesamiento y alargar de esta forma el tiempo de vida de la batería.

Además, debe admitir diversos modos de operación [27]:

- *Distribuido*: Propiedad esencial de las redes MANET.

- *Inactivo*: Los protocolos de encaminamiento deberán estar preparados para afrontar aquellos períodos de tiempo en los cuales los nodos frenan su actividad y permanecen inactivos para ahorrar energía.

- *Bajo demanda*: La adaptación del encaminamiento a los patrones de tráfico particulares de cada situación hace posible reducir el gasto de ancho de banda y energía, aunque se amplía el tiempo de obtención de la ruta.

- *Soporte de enlaces unidireccionales*: Los protocolos de encaminamiento en muchas ocasiones han sido diseñados y funcionan correctamente sólo con enlaces bidireccionales y esto no debería ser así, porque en la práctica podemos encontrarnos con la existencia de enlaces unidireccionales que sean clave para el intercambio de información en redes móviles ad hoc.

Se han diseñado numerosos protocolos de encaminamiento para redes MANET atendiendo a estos criterios.

La finalidad del MANET WG [30] del IETF es estandarizar la funcionalidad de un protocolo de encaminamiento IP para aplicaciones de encaminamiento inalámbrico dentro de topologías tanto estáticas como dinámicas como consecuencia de la movilidad de los nodos u otros factores. Los enfoques están destinados a ser relativamente generales pues deben ser adecuados en múltiples entornos inalámbricos y hardware y dirigidos a escenarios donde las redes móviles ad hoc estén desplegadas en la frontera de una infraestructura IP. Las infraestructuras *mesh* híbridas (por ejemplo, una mezcla de *routers* móviles y fijos) deberían ser soportados por las especificaciones de las redes móviles ad hoc.

4.2. Clasificación

Desde que se empezaron a estudiar las redes móviles ad hoc se han propuesto diversas clasificaciones de los protocolos de encaminamiento que se resumen en [22].

En función de la información de estado que almacenan los nodos de la red los protocolos pueden ser clasificados en protocolos basados en la topología y protocolos basados en el destino. En los primeros cada nodo toma decisiones basándose en una completa información de la topología de la red. Los segundos son protocolos que manejan vectores de distancias, en los que cada nodo intercambia con sus vecinos las distancias que conoce a otros nodos.

Otra clasificación propone dividir los protocolos en función de la estructura, diferenciándose varios niveles. La Figura 4.1. presenta esta clasificación formulado, especificando algunos de los protocolos representativos de cada categoría:

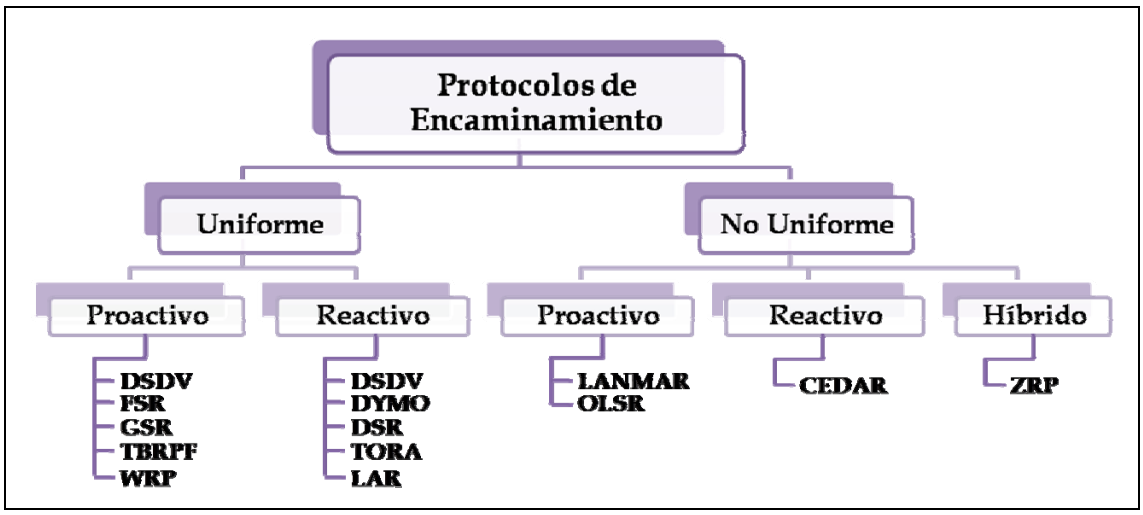

Fig. 4.1. Taxonomía de protocolos de encaminamiento en redes móviles ad hoc.

El primer nivel se refiere a la homogeneidad o heterogeneidad de las

funciones de los nodos en el encaminamiento, distinguiéndose dos tipos:

- *Protocolos uniformes o de estructura plana*: Ningún nodo de la red realiza un papel distinto al de los demás, todos ellos envían y responden a los mensajes de control del mismo modo.

- *Protocolos no uniformes*: Típicos de estructuras jerárquicas en las que algunos nodos desarrollan papeles especiales e incluso pueden dotarse de capacidades particulares en términos de cómputo, energía o almacenamiento entre otros. Esto les permite soportar algoritmos más complejos, reducir la sobrecarga debida a la comunicación y ofrecer la posibilidad de balanceo de carga mientras mantienen sus características, incluso ante incrementos del número de nodos en la red. Por el contrario, generan cierto coste de mantenimiento de la estructura y necesitan en muchos casos la disponibilidad de nodos heterogéneos.

Dentro de estas dos categorías anteriores, los protocolos presentan una nueva peculiaridad relativa al procedimiento adoptado para el descubrimiento del camino a establecer y su mantenimiento. Esta clasificación, sin duda, es la más difundida surgiendo los siguientes tipos de protocolos:

- *Protocolos proactivos*: En este tipo de encaminamiento cada nodo mantiene información de cómo llegar a cualquier otro nodo de la red e intercambia esta información con todos sus vecinos. La información de encaminamiento es normalmente almacenada en un número diferente de tablas. Periódicamente se actualizan las tablas si la topología de red cambia. La diferencia entre los protocolos de este tipo se encuentra en la forma de actualizar y detectar la información de encaminamiento y el tipo de información que se guarda en cada tabla. La ventaja que aportan estos protocolos es la baja latencia ya que las rutas están siempre disponibles. Sin embargo, esto conlleva un consumo de energía muy alto en los nodos

y se puede producir una sobrecarga de mensajes en la red debido a la inundación periódica de mensajes. DSDV (*Destination-Sequenced Distance-Vector*) [42], WRP (*Wireless Routing Protocol*) [33], GSR (*Global State Routing*) [9], FSR (*Fisheye State Routing Protocol*) [18], OLSR (*Optimized Link State Routing*) [11] o TBRPF (*Topology Broadcast Reverse Path Forwarding*) [37] son ejemplos de protocolos de encaminamiento proactivos. En general, estos protocolos tratan de evitar bucles en las rutas, consumo excesivo de memoria y reducción del tamaño de los paquetes que contienen la información de las tablas de encaminamiento. Dentro de los protocolos proactivos se pueden distinguir dos subtipos de protocolos según su comportamiento: los conducidos por eventos (*event-driven*), que envían paquetes con información sobre las rutas sólo cuando éstas sufren algún cambio, y los que refrescan la información periódicamente (*regular updated*). El protocolo OLSR, que ha sido utilizado para este trabajo y que será analizado en detalle en el siguiente apartado, entra dentro de la segunda categoría.

- *Protocolos reactivos*: Estos protocolos tratan de reducir la sobrecarga que producen los protocolos proactivos. Para ello proponen que los nodos de la red móvil ad hoc, cuando no tienen una ruta a un destino, la calculen sólo cuando es necesaria, es decir, cuando el nodo tenga que comenzar un intercambio de paquetes con el destino. El descubrimiento de una ruta normalmente se realiza por inundación de mensajes de solicitud por toda la red. Estos protocolos conllevan una alta latencia, provocada por el descubrimiento de rutas. Sin embargo, la sobrecarga de mensajes por la red se reduce. AODV (*Ad hoc On-demand Distance Vector*) [41], DYMO (*DYnamic Manet On-demand*) [8], DSR (*Dynamic Source Routing*) [23], ROAM (*Routing On-demand Acyclic Multi-path*) [44], LMR (*Lightweight Mobile Routing*) [12], LAR (*Location Arder Routing*) [25] o TORA (*Temporally-Ordered Routing Algorithm*) [39] son algunos de los protocolos

de encaminamiento reactivos. La mayoría de ellos tienen el mismo coste de encaminamiento en el peor escenario posible ya que casi todos siguen la misma filosofía para el descubrimiento de rutas.

- *Protocolos híbridos*: Combinando los protocolos proactivos y reactivos nacen los protocolos híbridos que pretenden minimizar los inconvenientes de ambos. La idea de estos protocolos es que los nodos de la red trabajen de forma proactiva con los nodos más cercanos y de forma reactiva con el resto de nodos. La parte reactiva controla la sobrecarga y el consumo de memoria al calcular las rutas sólo cuando son necesarias. En contraste, la parte proactiva necesita actualizar periódicamente la información almacenada y mantiene rutas que quizás nunca serán utilizadas, añadiendo una innecesaria sobrecarga. El caso más conocido de protocolo híbrido es ZRP (*Zone Routing Protocol*) [19].

El Grupo de Trabajo MANET tiene previsto desarrollar dos especificaciones de protocolo de encaminamiento estándar, denominadas *Reactive MANET Protocol* (RMP) y *Proactive MANET Protocol* (PMP), si bien también puede decidir un enfoque mixto. Soportará IPv4 e IPv6, requisitos de seguridad y otros aspectos, y prestará atención especial al protocolo OSPF-MANET [30] que viene desarrollando el OSPF WG [38]. El OSFP WG desarrolla extensiones del protocolo OSPF para diferentes escenarios, siendo OSPF-MANET la extensión de OSPF a redes móviles ad hoc.

4.3. Protocolo OLSR

El protocolo de encaminamiento OLSR (*Optimized Link State Protocol*) pertenece al grupo de los protocolos para MANETs que están definidos como RFC (*Request For Comments*). OLSR se especifica en la RFC 3626 [11], siendo una optimización del clásico protocolo de estado de enlace u OSPF (*Open Shortest*

Path First) [32], pero adaptado a redes móviles ad hoc.

Al tratarse de un protocolo proactivo la información sobre las rutas hacia todos los nodos se mantiene siempre actualizada, para que esté disponible en caso de que sea necesaria. Como se ha indicado anteriormente, OLSR es *regular updated*, esto es, cada cierto tiempo paquetes de información sobre las rutas son transmitidos, aunque no se hayan detectado cambios.

La principal aportación de OLSR que lo diferencia de otros protocolos similares son las optimizaciones que se realizan para que la sobrecarga producida por las actualizaciones periódicas sea mínima. El modo en que se hace llegar la información sobre rutas a toda la red es mediante inundación controlada: designando a ciertos nodos encargados de enviarse entre ellos la información, haciéndola llegar posteriormente al resto de la red, y comprobando que no se envían datos duplicados.

Debido a las optimizaciones que aplica, obtiene buenos resultados en redes grandes y densas. Sus optimizaciones se notan en el rendimiento cuanto más grandes sean las redes, sobre todo si el tráfico es esporádico entre pares de nodos que varíen irregularmente, en lugar de comunicaciones regulares entre nodos concretos.

4.3.1. Funcionamiento del Protocolo

Lo primero que hace un nodo al iniciarse es detectar con qué otros nodos tiene conexión a nivel de enlace. Para ello periódicamente se emiten mensajes HELLO. Estos mensajes no se retransmiten por los nodos que los reciben, ya que su finalidad es que los nodos se den a conocer a sus vecinos de un salto, es decir, nodos con los que existe conectividad a nivel de enlace. Otra funcionalidad de los mensajes HELLO, aparte de dar a conocer al propio nodo, es anunciar los vecinos ya conocidos del nodo emisor. De esta forma, un nodo

que escucha estos mensajes no sólo descubre a sus vecinos de un salto, sino que además adquiere conocimiento de sus vecinos de dos saltos de distancia.

La clave del protocolo está en los Multipuntos de Retransmisión (*Multipoint Relay*), abreviadamente MPR. Una vez que un nodo conoce el conjunto de sus vecinos de dos saltos, elige de entre sus vecinos de un salto un grupo de nodos MPR que retransmitan sus mensajes, de forma que le proporcionen acceso hacia todos los vecinos de dos saltos. Los nodos elegidos como MPR son notificados de su condición., manteniendo información sobre quiénes le han elegido como MPR (denominados selectores MPR) en una estructura llamada MPR *selector set*.

Uno de los cometidos de los MPR es retransmitir los mensajes de difusión generados por alguno de los nodos en su MPR *selector set*. De este modo, los mensajes llegan a toda la red, pero intentando que la saturación sea mínima.

La otra tarea que debe realizar un nodo que ha sido seleccionado como MPR es generar y retransmitir mensajes *Topology Control* (TC), que dan a conocer al resto de la red los nodos de los que el emisor tiene constancia. Los mensajes TC se generan de forma periódica, y contienen una lista con las direcciones de los nodos del MPR *selector set*, nodos que han elegido al emisor del mensaje como MPR (a diferencia de otros protocolos que anunciarían a cualquier nodo cercano). Con esto se consigue que la información sobre la topología generada sea mínima, y que su diseminación por la red se haga de forma controlada.

También es importante conocer las estructuras de datos internas que maneja OLSR. En particular, la más relevante es la tabla de rutas. La información que se tiene de cada nodo de la red en la tabla de rutas es una entrada con los siguientes campos:

R_dest_addr *R_next_addr* *R_dist* *R_iface_addr*

Esa entrada significa que el nodo identificado por la dirección *R_dest_addr* está a una distancia estimada de *R_dist* saltos, que el vecino con la dirección de interfaz *R_next_addr* es el siguiente salto en la ruta hacia *R_dest_addr*, y que este vecino es alcanzable a través de la interfaz local con la dirección *R_iface_addr*.

En OLSR se tiene en consideración la posibilidad de que un nodo tenga más de una interfaz de red participando al mismo momento en la red. Cada dirección de interfaz está asociada con una dirección principal, única para cada nodo. Esta dirección principal será la misma que la dirección de la interfaz en caso de que ese nodo tenga una única interfaz utilizando OLSR.

4.3.2. Formato del paquete OLSR

Las comunicaciones en OLSR se realizan usando un formato de paquete común para toda la información relacionada con el protocolo. De este modo, se facilitan futuras ampliaciones del protocolo sin romper la compatibilidad con versiones anteriores. Además, esto también facilita agrupar diferentes *tipos* de información dentro de una misma transmisión.

Los paquetes se encapsulan dentro de datagramas UDP para su transmisión por la red.

Cada paquete encapsula a su vez uno o varios mensajes. Los mensajes comparten un formato de cabecera común, lo que permite que un nodo sea capaz de aceptar y retransmitir (si procede) mensajes de tipo desconocido.

Los mensajes pueden ser transmitidos por inundación a la red en su totalidad, o la transmisión puede ser limitada a nodos dentro de un determinado diámetro -refiriéndonos a número de saltos- desde el emisor del mensaje. Por lo tanto, transmitir un mensaje al vecindario de un nodo es un caso especial de transmisión por inundación. Cuando se transmiten mensajes de

control, las retransmisiones duplicadas son eliminadas de forma local, ya que cada nodo guarda información sobre los mensajes de control que ya ha transmitido anteriormente.

Los paquetes en OLSR usan el puerto UDP 698, asignado por el *Internet Assigned Numbers Authority* (IANA).

Los campos de cualquier paquete OLSR (omitiendo las cabeceras IP y UDP) se indican en la Figura 4.2.

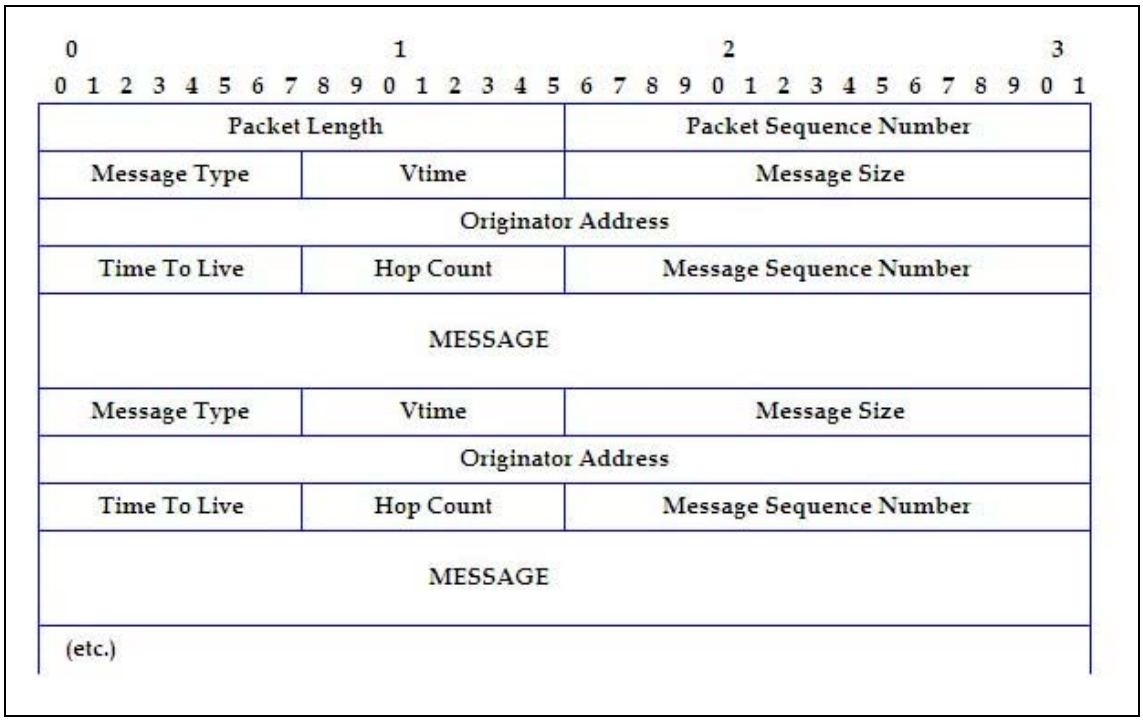

Fig. 4.2. Formato del paquete OLSR.

4.3.2.1. Cabecera del paquete

La Figura 4.3. muestra los campos de la cabecera del paquete OLSR.

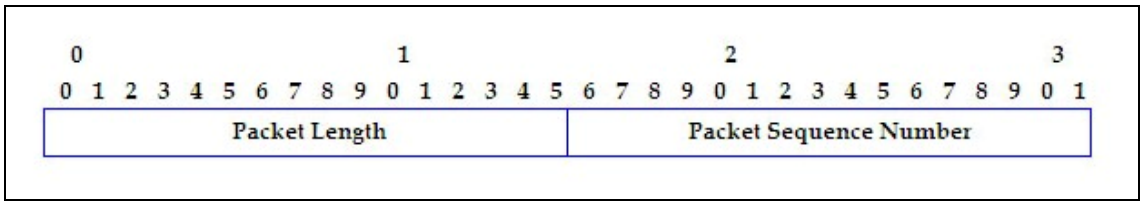

Fig. 4.3. Cabecera del paquete OLSR.

- *Packet Length*: [16 bits]. En este campo se define el tamaño del paquete OLSR en bytes.

- *Packet Sequence Number*: [16 bits]. Este campo se utiliza para definir el número de secuencia del paquete. Debe ser incrementado en uno cada vez que un nuevo paquete OLSR es transmitido.

4.3.2.2. Cabecera del mensaje

En la Figura 4.4. se representan los campos de cabecera del mensaje:

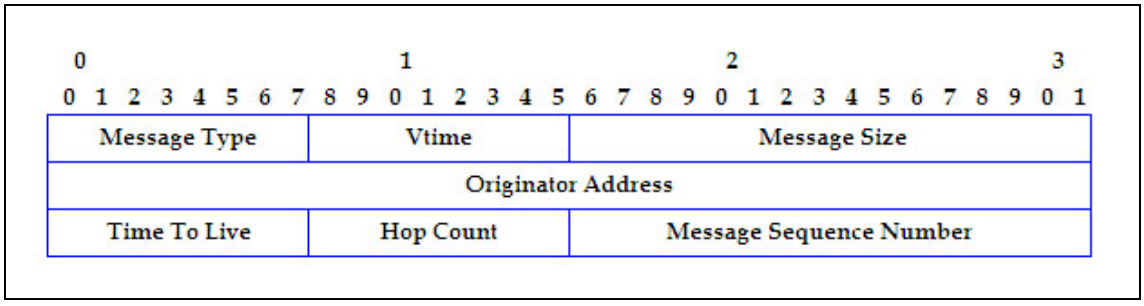

Fig. 4.4. Cabecera del mensaje.

- *Message Type*: [8 bits]. Este campo indica qué tipo de mensaje se encuentra en el campo MESSAGE. Los tipos en el rango 0-127 son reservados.

- *Vtime*: [8 bits]. Campo que indica el tiempo de validez de la información contenida en el mensaje.

- *Message Size*: [16 bits]. Tamaño del mensaje en bytes, incluyendo la

cabecera y el campo MESSAGE.

- *Originator Address*: [32 bits]. Indica a dirección principal del nodo que originalmente generó el mensaje. No debe confundirse este campo con la dirección origen de la cabecera IP, que se modifica cada vez que el paquete OLSR se retransmite por un nodo intermedio.

- *Time To Live*: [8 bits]. Contiene el número máximo de saltos que el mensaje será transmitido. Antes de retransmitir un mensaje, al valor de este campo se le debe restar uno.

- *Hop Count*: [8 bits]. Número de saltos que el mensaje ha dado. Se inicializa a 0, y se incrementa en 1 en cada retransmisión.

- *Message Sequence Number*: [16 bits]. Al generar un mensaje, el nodo que lo genera le asigna un número de secuencia único que identifica a cada mensaje en este campo. El número de secuencia se incrementa en 1 para cada mensaje originado por el nodo

4.3.3. Mensaje HELLO

El protocolo OLSR utiliza el intercambio periódico de mensajes HELLO para descubrir a los nodos vecinos y el estado de la red a nivel de enlace.

4.3.3.1. Formato del mensaje HELLO

Los mensajes HELLO siguen un formato similar al del paquete general, de modo que puedan incluir información para la detección del estado de los enlaces de la red, para transmitir señales para la detección de nodos vecinos, para selección de MPRs y para tener en cuenta futuras ampliaciones.

El formato del mensaje se indica en la Figura 4.5.

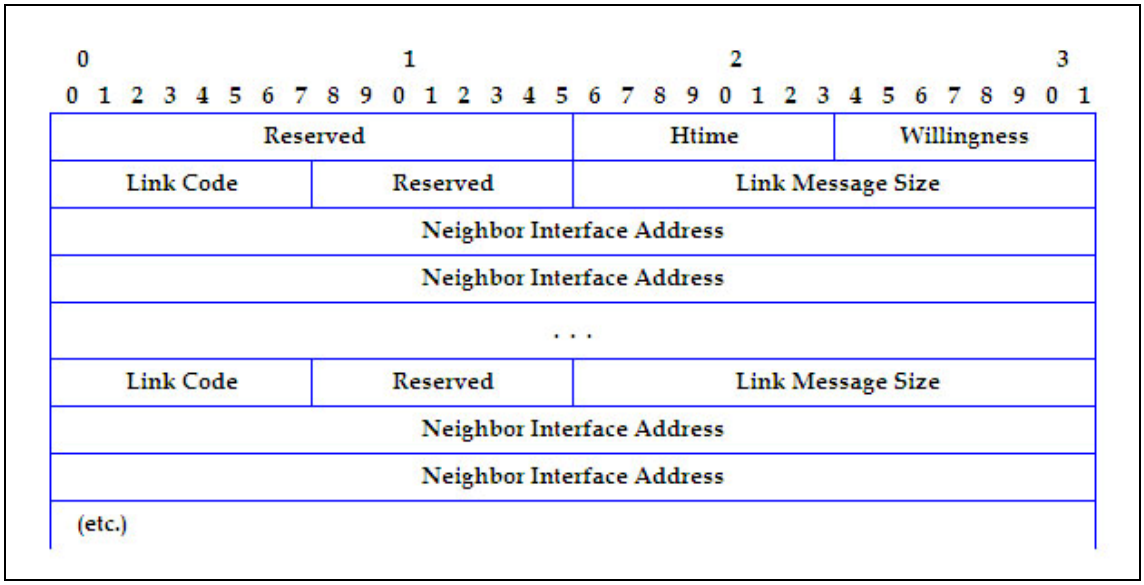

Fig. 4.5. Formato del mensaje HELLO.

Se envía como datos dentro del paquete general OLSR descrito anteriormente, configurando el campo *Message Type* con el valor HELLO_MESSAGE y el campo TTL con 1.

- *Reserved*: [16 bits]. Reservado, se establece con el valor 0000000000000.

- *HTime*: [8 bits]. Este campo especifica el intervalo de emisión de mensajes HELLO usado por el nodo, esto es, el tiempo hasta el envío del próximo HELLO.

- *Willingness*: [8 bits]. Indica la disponibilidad que tiene un nodo para re-enviar tráfico a otros nodos, si la disponibilidad de un nodo se define como WILL_NEVER, éste nunca será elegido como un nodo MPR.

- *Link Code*: [8 bits]. Este campo especifica el tipo de enlace que el nodo emisor tiene con los vecinos en su lista. Como mínimo, OLSR requiere los siguientes tres tipos de enlaces:

- *ASYM_LINK*: Indica que los enlaces entre el nodo emisor y sus vecinos son de tipo asimétricos (es decir, solamente es posible "escuchar" al vecino, pero no es posible establecer un enlace bidireccional con éste).

- *SYM_LINK*: Indica que el enlace entre el emisor y sus vecinos son simétricos (existe un enlace bidireccional).

- *MPR_LINK*: Indica, que los nodos definidos en la lista han sido seleccionados por el emisor como MPR.

- *Reserved*: [8 bits]. Este campo se reserva para uso futuro, y debe ser puesto a '00000000000000000000000'.

- *Link Message Size*: [16 bits]. Este campo define el tamaño del mensaje de enlace, el cual se mide desde el inicio del campo *Link Code* hasta el siguiente campo de *Link Code*. Si no existe otro campo de *Link Code* dentro del mensaje HELLO, entonces el valor del campo *Link Code*" se mide hasta el fin del mensaje HELLO.

- *Neighbor Interface Address*: [32 bits]. Este campo define la lista de vecinos que se han etiquetado con un *Link Code* en particular.

4.3.3.2. Procesamiento del mensaje HELLO

Los nodos procesan los mensajes HELLO recibidos para la detección de conexiones a nivel de enlace, detección de vecinos y selección de MPR.

4.3.4. Descubrimiento de vecinos

4.3.4.1. Detección de conexiones a nivel de enlace

Cada nodo guarda información sobre sus conexiones a nivel de enlace con otros nodos en una estructura llamada *Link Set*. Con estas conexiones nos referimos más concretamente a las interfaces de red utilizando OLSR y su capacidad de intercambiar paquetes OLSR.

El mecanismo usado para esta detección es el intercambio periódico de mensajes HELLO. Para considerar una conexión válida se debe comprobar que existe comunicación (recepción de HELLO) en ambos sentidos.

Cada nodo vecino tiene asociado un estado en relación a la conexión: *simétrico* o *asimétrico*. El primero indica que la comunicación en ambos sentidos ha sido confirmada; y el segundo se usa para indicar que se han recibido mensajes HELLO generados por el nodo vecino, pero no se ha confirmado aún que el nodo vecino es capaz de recibir los HELLO generados localmente.

La confirmación de que un nodo vecino es capaz de recibir los mensajes HELLO emitidos se tiene al encontrar la dirección propia en los HELLO del vecino.

4.3.4.2. Detección de vecinos

Cada nodo mantiene un conjunto de tuplas de vecinos (*neighbor tuples*) basadas en la información sobre las conexiones almacenadas en el *Link Set*.

Cada tupla de vecino consta de los datos:

N_*neighbor_main_addr* *N_status* *N_willingness*

donde *N_neighbor_main_addr* es la dirección principal del nodo vecino, *N_status* se refiere al estado de la conexión (simétrica o asimétrica), y *N_willingness* es un entero entre 0 y 7 que representa la disposición o intención del vecino de retransmitir tráfico de otros nodos.

Aparte del conjunto de vecinos inmediatos o de un salto, con los que se tiene conexión a nivel de enlace, cada nodo guarda información sobre el conjunto de nodos de dos saltos de distancia en la estructura *2-hop Neigbor Set*. Para ello, por cada nodo vecino de un salto, se guarda el conjunto de sus vecinos de un salto, ya que están anunciados en los mensajes HELLO que se reciben periódicamente.

4.3.5. Multipuntos de Retransmision (MPR)

Los MPR (Multipoint Relay) se usan para inundar mensajes de control de un nodo a toda la red minimizando las retransmisiones. Por lo tanto, el concepto de MPR se considera una optimización del mecanismo habitual de inundación de mensajes en una red.

Como se ilustra en la Figura 4.6, cada nodo en la red elige, independientemente de los demás, su propio conjunto de MPR de entre sus vecinos de un salto con conexión simétrica. El conjunto de nodos seleccionados se conoce como el conjunto MPR de ese nodo. Los vecinos del nodo N que no están dentro del grupo MPR, reciben y procesan la información de los mensajes de difusión, pero no retransmiten la información proveniente del nodo N.

La sobrecarga del tráfico de control generada por el protocolo de enrutamiento es directamente proporcional al tamaño del conjunto de nodos MPR en la red. A su vez, los nodos MPR mantienen información sobre el conjunto de vecinos a un salto que lo han seleccionado como MPR; este conjunto se conoce como conjunto selector de MPR de un nodo. Está

información se adquiere de los mensajes HELLO recibidos de los vecinos a un salto.

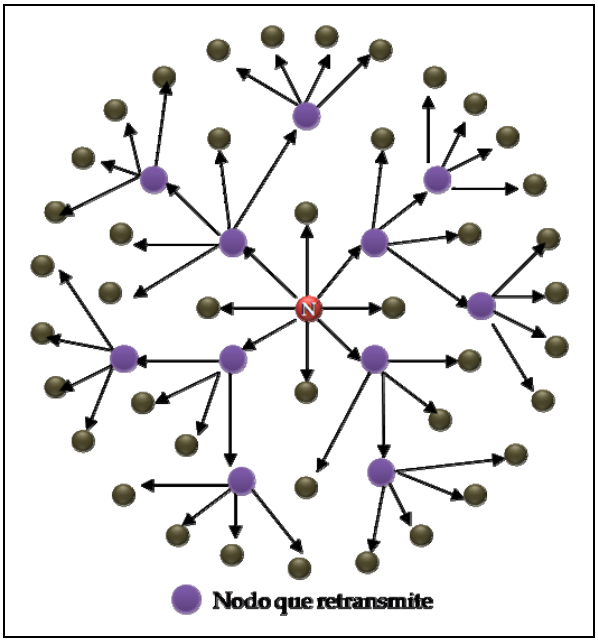

Fig. 4.6. Proceso de selección de nodos MPR.

Aunque la inundación de mensajes pura es más confiable y robusta, ésta consume una gran cantidad del ancho de banda. El empleo de nodos MPR proporciona resultados igualmente buenos, con mucho menos tráfico de control. En la Figura 4.7 se ilustra una comparación, en términos de retransmisiones, para hacer llegar un mensaje de difusión a 3 saltos en la red.

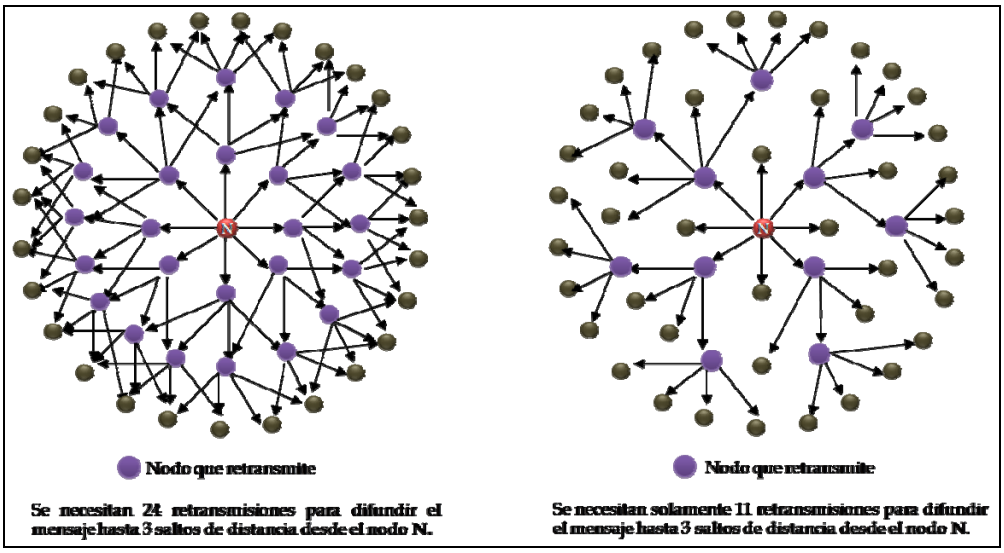

Fig. 4.7. Diferencia entre la difusión pura y el uso de nodos MPR.

4.3.5.1. Selección de MPR

Cuando se ha elegido a un vecino como MPR, se anuncia en los mensajes HELLO colocando el valor MPR_NEIGH en lugar de SYM_NEIGH en el campo *Link Type* anterior a la dirección del vecino elegido como MPR.

El conjunto de MPR es calculado por cada nodo de forma que, a través de los vecinos elegidos, el nodo sea capaz de alcanzar a todos los vecinos de dos saltos. Más concretamente, a los vecinos de dos saltos exactos, por lo que no se está contando a los vecinos de un salto.

Aunque el conjunto de MPR no debe ser mínimo para garantizar el correcto funcionamiento, cuanto más pequeño sea se consigue menor sobrecarga producida por los mensajes de control del protocolo OSLR.

Cada nodo guarda además en el *MPR selector set* el conjunto de nodos que le han elegido como MPR. Son detectados al procesar los mensajes HELLO recibidos.

4.3.6. Descubrimiento de la topología en OLSR

4.3.6.1. Funcionamiento

La detección de conexiones y vecinos proporciona a cada nodo una lista de nodos con los que comunicarse directamente y, haciendo uso de nodos MPR, un mecanismo para inundaciones optimizado. Basándose en esto, se genera información sobre la topología y se distribuye por la red.

Los mensajes *Topology Control* (TC) los generan los nodos que han sido elegidos como MPR por algún vecino suyo. Sirven para anunciar un conjunto de enlaces entre el emisor y otros nodos, que por lo general será su MPR Selector Set, es decir, los vecinos que han elegido al nodo emisor como MPR. Estos mensajes TC se emiten a toda la red por inundación.

Debido a limitaciones de tamaño de los mensajes en la red, la lista de direcciones anunciadas puede ser parcial en cada mensaje TC. Sin embargo, al unir todos los mensajes TC emitidos deben encontrarse todas las direcciones del MPR *Selector Set*.

4.3.6.2. Formato de los mensajes TC

Los mensajes TC tienen el siguiente formato mostrado en la Figura 4.8:

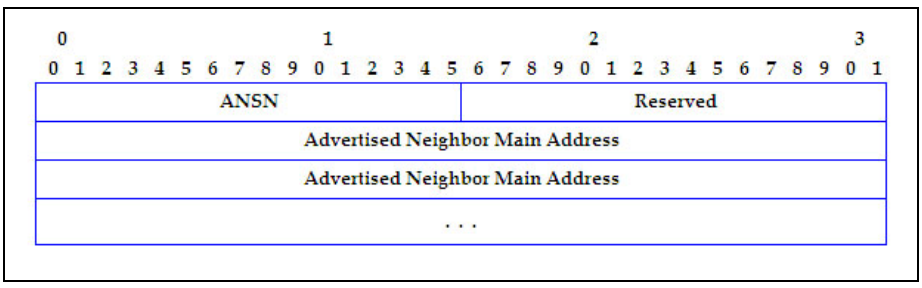

Fig. 4.8. Formato del mensaje TC.

Esto se envía como datos dentro del paquete OLSR, configurando el campo *Message Type* con el valor TC_MESSAGE y el campo TTL con el valor 255 (el máximo).

- *Advertised Neighbor Sequence Number* (ANSN): [16 bits]. Un número de secuencia que se asocia con el conjunto de vecinos que se anuncia en este mensaje. Cada vez que un nodo detecta cambios en el conjunto de sus vecinos, se incrementa este número. Sirve para que los nodos que reciben este mensaje sepan si se trata de información más reciente que la que puedan tener actualmente.

- *Reserved*: [16 bits]. Este campo está reservado. Se le da el valor 0000000000000000.

- *Advertised Neighbor Main Address*: [32 bits]. Campo que contiene la dirección principal de un nodo vecino.

4.3.7. Cálculo de las tablas de rutas

Cada nodo mantiene actualizada una tabla con rutas hacia el resto de nodos de la red. Esta tabla se basa en la información obtenida sobre los nodos vecinos y los mensajes de control TC.

El formato de las entradas en esta tabla es el siguiente:

a. *R_dest_addr* *R_next_addr* *R_dist* *R_iface_addr*

b. *R_dest_addr* *R_next_addr* *R_dist* *R_iface_addr*

c. …

Cada entrada significa que el nodo con la dirección *R_dest_addr* se encuentra

a una distancia de *R_dist* saltos del nodo local, que el nodo vecino con la dirección de interfaz de red *R_next_addr* es el siguiente salto en la ruta hacia *R_dest_addr*, y que este nodo es alcanzable desde la interfaz local con la dirección *R_iface_addr*. Se mantiene una entrada por cada nodo de la red para el que se tiene ruta conocida.

Los nodos con una ruta desconocida no se incluyen. La actualización de la tabla se realiza en caso de aparición o desaparición de un nodo, ya sea vecino inmediato, vecino de dos saltos de distancia, o cualquier otro nodo conocido a través de los mensajes de control TC. También se actualiza la tabla cuando cambia información sobre las interfaces múltiples que puedan estar asociadas a los nodos.

Esta actualización de la tabla es un proceso interno, que no desencadena el envío de ningún mensaje.

5. D2HCP: CONSIDERACIONES DE DISEÑO

El objetivo de este capítulo es desarrollar las principales ideas inherentes al diseño del protocolo D2HCP (*Distributed Dynamic Host Configuration Protocol*), sin entrar en demasiados detalles técnicos que obstaculicen su explicación. De modo complementario, el capítulo 6 está dedicado a la especificación formal del mismo.

Así, se expondrá cómo se ha mejorado el protocolo de Moshin y Prakash [31] optimizando sus estructuras de datos y la forma en la que se recuperan bloques de direcciones IP libres.

También se hablará del nuevo proceso de sincronización, usando el protocolo OLSR y de la inicialización de las estructuras de datos a la hora de entrar un nodo en la red.

Por último, se razonarán las dificultades a los que los nuevos mecanismos propuestos deben hacer frente.

5.1. Mejoras sobre el protocolo de Moshin y Prakash

En el protocolo de Moshin y Prakash [31] se tiene que por cada nodo de la red hay que conocer la siguiente información: su dirección IP, los bloques de direcciones libres *Free_IP_blocks*, el nodo *father* que le suministró la dirección IP y su marca de tiempo TS.

Recordemos que *Free_IP_blocks* se refiere a varios bloques disjuntos de direcciones IP, del tipo [.5 - .10], [.15 - .20], [.48 - .120].

Los campos *father* y TS se usan para determinar quién es el responsable de recuperar las direcciones libres que un nodo deja disponibles al abandonar la red. Este responsable es el nodo *father*, o en caso de no estar disponible, el nodo de la red con menor TS.

El hecho de que cada nodo pueda tener muchos bloques diferentes de direcciones libres hace que no se pueda determinar cuánto puede llegar a ocupar el campo *Free_IP_blocks*.

Se concluyó que si se cambiase el modo en el que se recuperan las direcciones de un nodo que sale de la red, los campos *father* y *TS* podrían omitirse.

Las diferencias de este nuevo mecanismo con respecto al original se resumen en los siguientes puntos:

- Cada nodo tiene como información asociada para la autoconfiguración un único bloque de direcciones IP libres contiguas, que incluyen a la propia dirección IP del nodo.

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Con este nuevo comportamiento se consiguen varias mejoras.

Por una parte ya no son necesarios los campos *father* y TS, ya que el propio bloque de direcciones libres de cada nodo determina quién es el responsable de recogerlo en caso de que un nodo abandone la red.

Se garantiza además que cada nodo tiene un solo bloque de direcciones IP libres en todo momento, haciéndolo más manejable. Se conoce cuánto ocupará esta información, y se garantiza que este tamaño será constante. Esto es importante para poder estudiar los requisitos de memoria de los nodos, o la sobrecarga que supondría su transmisión por la red.

En el protocolo original, cada nodo debe comprobar periódicamente mediante mensajes *ping* si siguen activos tanto el nodo "padre" como sus "hijos". El nodo padre es el que le suministró su dirección IP, y los nodos hijos son los nodos a los que se les ha facilitado una dirección IP y un bloque de direcciones libres.

Con las mejoras propuestas, se consigue que los nodos no tengan que comprobar activamente si otros nodos siguen participando en la red. Será en el momento de detectar que otro nodo ha abandonado la red, cuando cada nodo comprobará si el bloque de direcciones que queda disponible debe ser recogido por él mismo, o por otro de los nodos de la red.

En el siguiente apartado veremos un ejemplo que muestra este comportamiento de forma más clara.

5.2. Elección de OLSR como mecanismo de sincronización

Después de estudiar el comportamiento y los aspectos clave de los protocolos

de encaminamiento más conocidos, se decidió usar OLSR como base para la sincronización del protocolo de autoconfiguración por varios motivos.

El primero y más importante es que al ser un protocolo proactivo y con refresco de las rutas de forma periódica, nos da la seguridad de poder conocer la topología de la red en todo momento. Aunque algún paquete de actualización se pierda, las rutas se actualizarán igualmente con las siguientes transmisiones.

La otra gran ventaja que se notó es que OLSR aporta la infraestructura para distribuir información a toda la red usando MPR, mecanismo que podría ser utilizado para hacer llegar nuevos mensajes de control a todos los nodos de la red.

Por último, el estar optimizado para funcionar en redes a gran escala también influyó en la decisión de utilizarlo, ya que se pretendía someter al protocolo de autoconfiguración resultante a pruebas exhaustivas en escenarios complejos.

Para terminar con el análisis de OLSR se intentó concretar qué información se podía deducir de este protocolo de encaminamiento, monitorizando su comportamiento o sus estructuras de datos, sin modificar ni su comportamiento ni sus paquetes. Se centró el interés en interpretar los datos que OLSR maneja, y los cambios que suceden con estos datos.

Debido a que se tiene información actualizada sobre la topología de la red, podemos determinar la entrada o la salida de cualquier nodo de forma inmediata y sencilla. Además, se tendrá información acerca de esos nodos, como por ejemplo su IP.

Aunque parezca información escasa, más adelante veremos que será más que suficiente para desarrollar ideas más complejas.

También se vio que monitorizando las estructuras de datos que OLSR maneja se puede conocer el conjunto de nodos considerados vecinos, es decir, con los que se tiene un enlace directo o que se encuentran a un salto de distancia.

De nuevo, puede parecer información poco relevante, pero permitiría distinguir entre nodos vecinos o lejanos para controlar el comportamiento de mecanismos más complejos. Por ejemplo, esta información podría ser usada para decidir enviar ciertas peticiones sólo a los nodos vecinos, con el fin de tener respuestas inmediatas, o evitar inundar toda la red con mensajes de control.

5.3. Sincronización usando OLSR

La solución que se propone a la necesidad de un mecanismo de sincronización para el protocolo D2HCP es integrar en parte el protocolo de encaminamiento OLSR.

Al integrar la autoconfiguración con OLSR, conseguimos aprovechar el mecanismo provisto por este último para detectar los cambios en la red. Recordemos que OLSR se trata de un protocolo de encaminamiento proactivo, y que mantiene información actualizada sobre las rutas hacia todos los nodos de la red. Por ello, es capaz de detectar la entrada de un nuevo nodo a la red (descubrimiento de una ruta nueva), y la salida de nodos (eliminación de una de las rutas conocidas).

Por cada nodo de la red, OLSR mantiene una entrada en su tabla de rutas. Esta tabla consta de los campos:

R_dest_addr *R_next_addr* *R_dist* *R_iface_addr*

Podemos ver un ejemplo en la Tabla 5.1, donde se omiten los campos referentes a las rutas por simplicidad:

R_dest_addr	*R_next_addr*	*R_dist*	*R_iface_addr*
.1	…	…	…
.128	…	…	…
.65	…	…	…

En el protocolo propuesto necesitamos conocer el bloque de direcciones IP que tiene asignado cada nodo de la red, de modo que cada nodo tendrá almacenada una tabla similar a la siguiente:

IP	**Free_IP_blocks**
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

Puede verse la relación entre una y otra de forma clara. En cada tabla tenemos una entrada por cada nodo de la red. También se cumple que ambas tablas deben estar actualizadas constantemente: deben reflejar la entrada o salida de cualquier nodo de la red.

Por lo tanto, ya que OLSR se encarga de conocer la topología de la red en todo momento, basta con monitorizar su tabla de encaminamiento para detectar cuándo un nodo se une o abandona la red. Es decir, la tabla de bloques de direcciones IP libres se actualizará a la par junto a la tabla de encaminamiento con el mecanismo provisto por OLSR.

Veamos un ejemplo que ilustra el funcionamiento y la actualización de los

campos. Hablaremos de las estructuras relacionadas con la sincronización, omitiendo los mensajes necesarios para realizar estos pasos.

La siguiente tabla muestra todos los nodos de una red. Tenemos un único nodo, con 254 direcciones IP libres.

IP	Free_IP_blocks
.1	.1 - .254

Esta es la tabla que maneja el protocolo internamente.

Como se ve, la propia dirección del nodo está dentro del bloque de direcciones IP libres. Más adelante veremos por qué esto es necesario, y cómo se garantiza que no sea un problema.

Supongamos ahora que un nuevo nodo entra en la red y que obtiene una dirección IP y un bloque de direcciones IP libres del nodo con la dirección IP 1.

El bloque de direcciones libres [.1 - .127] se divide en dos sub-bloques de 127 direcciones cada uno:

[.1 - .127] y [.128 - .254]

El nodo con la dirección .1 debe entregar uno de estos dos sub-bloques al nuevo nodo cliente, y éste será el que no contiene su propia dirección (.1).

El nodo que entra en la red tiene por tanto asignado el rango de direcciones libres [.128 - .254], del que tomará la primera dirección para su propio uso.

En el caso de que el nuevo nodo tenga más de una interfaz de red participando en la misma red móvil ad hoc, usará tantas direcciones como

interfaces. Esas direcciones serán las primeras del rango. La primera de las direcciones de sus interfaces será además su dirección principal, que identificará al nodo. Cada dirección secundaria está ligada a la dirección principal del nodo.

En este momento, el nuevo nodo .128 estará correctamente configurado y comenzará a utilizar el encaminamiento OLSR. Por lo tanto, el nodo .1 descubrirá gracias a OLSR una nueva ruta hacia el nodo .128.

Esta nueva ruta de la tabla de encaminamiento se añade a la tabla de autoconfiguración:

IP	Free_IP_blocks
.1	.1 - .127
.128	.128 - .254

En este momento un nuevo nodo entra en la red, y de nuevo es el nodo .1 quien le proporciona una dirección IP y un bloque de direcciones libres.

En este caso, .1 dispone de 127 direcciones libres que debe dividir en dos sub-bloques. Como no puede crear dos sub-bloques de 63.5 direcciones cada uno, creará los sub-bloques

[.1 - .64] y [.65 - .127]

de 64 y 63 direcciones respectivamente. De nuevo, la dirección que está usando (.1) está contenida en el sub-bloque izquierdo, por lo que se asigna el derecho al nuevo nodo:

IP	Free_IP_blocks
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

Veamos ahora qué ocurre si el nodo .128 abandona la red. Al dejar de participar en la red, las direcciones libres [.128 - .254] no serán asignadas por .128 a nuevos nodos.

Estas direcciones no pueden desperdiciarse, por lo que alguien tiene que recogerlas. El encargado de ello es el único nodo que puede añadir esas direcciones por la derecha al bloque de direcciones IP libres que ya tiene. Ese nodo es .65.

IP	Free_IP_blocks
.1	.1 - .64
.65	.65 - .254

Recordemos que esta tabla es una estructura interna que maneja el protocolo. Cada uno de los nodos de la red tiene una copia, y cada nodo realiza las actualizaciones que estamos explicando de forma local, sin intercambiar ningún mensaje.

Aquí hay que destacar que el nodo .1 fue quien facilitó la entrada a la red al nodo .128. En el protocolo anterior diríamos que .1 es el padre de .128. Sin embargo, hemos visto que el encargado de recoger las direcciones que .128 deja disponibles es otro nodo diferente, determinado únicamente por el bloque de direcciones que tiene asignado.

Como vemos, el bloque de direcciones libres del nodo .65 ha crecido, pero

sigue manteniéndose como un único bloque de direcciones contiguas.

En el caso de que .1 dejase la red, su bloque de direcciones [.1 - .64] debe ser recogido por otro nodo. En este caso, no hay ningún nodo que pueda añadir este bloque al suyo por la derecha, ya que la dirección más baja de la red (.1) está contenida en él.

Por lo tanto, sus direcciones tienen que ser recogidas por el nodo que pueda añadir este bloque que queda disponible al suyo por la izquierda. De nuevo es el nodo .65 el responsable.

IP	Free_IP_blocks
.65	.1 - .254

Se observa una situación nueva: la dirección IP del nodo (.65) está contenida en mitad de su bloque de direcciones IP libres, en lugar de ser la primera de él.

Si no permitiésemos esto, el bloque no sería uno sólo, sino que tendríamos las direcciones libres

[.1 - .64] + [.66 - .254]

y ya no se cumpliría que cada nodo tiene un único bloque de direcciones libres.

En realidad esto no supone un problema a la hora de realizar la división binaria de direcciones libres. La entrada de un nodo en este momento supondría que el nodo .65 ha de dividir su bloque de direcciones. Su propia dirección IP queda en el sub-bloque izquierdo, y puede desprenderse de las direcciones .128 - .254 sin problemas.

En este punto la tabla quedaría de la siguiente forma:

IP	Free_IP_blocks
.65	.1 - .127
.128	.128 - .254

Si otro nodo entra en la red, y es .65 de nuevo quien le proporciona su dirección IP, las direcciones IP libres tendría que dividirlas en los sub-bloques:

[.1 - .64] y [.65 - .127]

De los dos bloques, uno de ellos es el que contiene la dirección IP del nodo que facilita la entrada al nuevo (la dirección IP .65 está en el bloque [.65 - .127]).

En este caso, al contrario de lo que ha sucedido antes, se entregará al nuevo nodo el bloque con las direcciones más bajas, para evitar que la dirección IP .65 sea usada por cualquier otro nodo ya que en realidad no está libre.

IP	Free_IP_blocks
.65	.65 - .127
.128	.128 - .254
.1	.1 - .64

5.4. Escenarios Problemáticos en la Sincronización

Al realizar la actualización de la tabla de autoconfiguración cada nodo de forma local, sin intercambiar mensajes que confirmen los cambios, debe tener la certeza de que ninguna situación puede desencadenar que un nodo tenga almacenada información que no se corresponda a la situación real de la red.

Existen varias situaciones conflictivas con el modelo de sincronización propuesto.

Un nodo perteneciente a la red ofrece la mitad de su bloque de direcciones IP libres a los nodos que intentan acceder, y así lo solicitan. Se debe tener en cuenta que tras ofrecer esas direcciones IP, el nodo servidor podría recibir otra solicitud diferente. En ese caso, el nodo no debería responder a esa solicitud hasta tener confirmación de que el ofrecimiento anterior ha sido aceptado, o rechazado.

Conocer si el nodo cliente al que se le han ofrecido las direcciones IP libres ha aceptado la solicitud es simple, ya que el protocolo OLSR detectará una nueva ruta hacia un nodo con la primera dirección IP del bloque ofrecido.

Para detectar si una solicitud no ha sido aceptada, el nodo servidor pone en marcha un temporizador desde el momento en que se envía el primer ofrecimiento. Si al expirar este temporizador no se tiene constancia de que el nodo cliente ha aceptado el ofrecimiento, esto significa que ha sido rechazado, ya que la oferta habrá caducado y el nodo cliente no podrá usar las direcciones ofrecidas.

Otro escenario problemático al que se debe prestar atención es la detección de entradas o salidas de nodos fuera de orden. Para explicar este escenario, desarrollamos a continuación un ejemplo práctico.

Supongamos que la siguiente tabla representa parte de una red en tres instantes de tiempo diferentes: t_1, t_2 y t_3. Se representan tres nodos y el bloque de direcciones que corresponde a cada nodo en los diferentes instantes. En un primer momento, el único nodo de la red representada es el nodo A. Más adelante se incorporan a ésta el nodo B y luego C, siendo en ambos casos el nodo A quien les facilite la entrada.

Nombre	IP	t_1	t_2	t_3
A	1	1 – 100	1 – 50	1 – 25
B	51		51 – 100	51 – 100
C	26			26 – 50

Si ésta es parte de una red más grande, estos cambios son detectados por otro nodo perteneciente a ella, aquí no representado. En su tabla de autoconfiguración, tiene almacenada la información de que el nodo A usa la dirección IP 1 y el bloque de direcciones IP libres [1 – 100]. Mediante la monitorización de la tabla de encaminamiento del protocolo OLSR, detectará la entrada de los nodos B (51) y C (26).

Pero por problemas de interferencias, saturación de la red, o pérdida de mensajes, podría ocurrir que OLSR incluyese antes una ruta hacia el nodo C que hacia el nodo B. Desde el punto de vista de la autoconfiguración, lo que pasaría es que en el instante t_2 aparece un nodo con la dirección IP 26. Esto implica que esa dirección ha sido facilitada por A, ya que tenía asignado el bloque de direcciones [1 – 100], lo cual es incorrecto. A habría dividido su bloque de direcciones en dos sub-bloques, [1 – 50] y [51 – 100], por lo que la aparición de un nodo con la dirección IP 26 no es posible.

Nombre	IP	t_1	t_2	t_3
A	1	1 – 100	1 – **50**	
B	-			
C	**26**		**51** – 100	

Un problema muy similar a éste se da si un nodo abandona la red en un momento cercano en el que el servidor que debe recoger las direcciones que quedan disponibles está dando entrada a un nuevo cliente. Otro nodo de la red

podría detectar la nueva entrada y la salida del nodo de manera que la tabla de autoconfiguración resultante no tuviese sentido.

La solución propuesta consiste en establecer temporizadores, de manera que se garantice que la entrada de un nuevo nodo en la red no esté cercana a otra entrada, o salida, que modifique los bloques de direcciones IP involucrados en esa nueva entrada.

Tras otorgar un bloque de direcciones IP libres a un nuevo nodo cliente, el servidor establece un tiempo durante el cual no atenderá más peticiones. Asimismo, el nodo que acaba de entrar en la red esperará el mismo tiempo hasta comenzar a atender peticiones de otros nodos.

Siguiendo con el ejemplo anterior, tras facilitar la entrada al nodo B, el nodo A no atendería la solicitud de C hasta que pasase un tiempo suficiente como para que cualquier nodo de la red haya podido detectar la nueva presencia de B. Por lo tanto, no existe riesgo de que la entrada del nodo C sea detectada antes que la de B por algún otro nodo.

De manera similar, cuando un nodo detecta que debe recoger las direcciones que un nodo que acaba de abandonar la red deja disponibles, espera un tiempo suficiente antes de volver a atender peticiones de clientes. Así, el resto de la red actualizará el bloque de direcciones correspondiente antes de que se divida y se entregue la mitad a algún cliente.

5.5. Inicialización

Hemos visto que podemos actualizar la información necesaria para sincronizar el protocolo D2HCP aprovechando las actualizaciones de rutas de OLSR. Pero para poder actualizar la tabla, un nodo que acaba de entrar a la red debe

primero construirla.

Proponemos dos soluciones: comenzar con una tabla, o ir creándola con información añadida a las actualizaciones periódicas del protocolo de encaminamiento.

En el código para realizar las simulaciones se implementó la primera solución, ya que es más simple y eficiente al no requerir modificar el protocolo OLSR.

5.5.1.1. Paso de tabla entera en un mensaje

Cuando un nuevo nodo accede a la red, depende de la intervención de un nodo cualquiera de los que ya pertenecen a la red para obtener una dirección IP válida y un bloque de direcciones IP libres, que el nuevo nodo podrá proporcionar a su vez a futuros nodos que deseen entrar a la red.

Durante ese proceso, el nuevo nodo está configurándose para poder hacer uso de la red, pero todavía no pertenece a ésta. Por ello, se decidió que en ese mismo momento sería lógico que el nodo servidor transmitiese una copia de su tabla de autoconfiguración al nodo cliente.

De este modo, se decidió modificar el mensaje IP_ASSIGNMENT para que contenga la tabla mencionada.

Esto quiere decir que en el momento en el que un nodo comienza a participar en la red dispone de una tabla de autoconfiguración que puede actualizar con la información aportada por OLSR.

Hay que aclarar que aunque la información sobre los bloques de direcciones libres y la topología de la red están relacionadas, en este primer momento se

dispone de la tabla de autoconfiguración, pero OLSR comenzará a funcionar de aquí en adelante. La tabla de rutas se encuentra vacía.

Si OLSR detecta una ruta nueva hacia un nodo antes desconocido, se actualizará además la tabla de autoconfiguración. Si ese nodo que OLSR descubre como nuevo se encuentra en la tabla que se recibió del nodo servidor, no habrá que actualizar ninguna entrada.

Se puede dar el caso de que tras recibir esa tabla de autoconfiguración con todos los nodos de la red, se produzca la salida de uno de ellos. Como se ha dicho, durante un tiempo inicial OLSR no contiene ninguna ruta, y las irá descubriendo desde ese momento. Por lo tanto, desde el punto de vista de OLSR, el que un nodo desaparezca de la red en esos primeros instantes no implica borrar una ruta, ya que no se disponía de ninguna. En este caso la detección de esa salida se realiza mediante un temporizador.

Al recibir la tabla, se inicia un temporizador. Cuando este expira, se borrarán de la tabla de autoconfiguración los nodos para los cuales OLSR no tenga ruta conocida. De este modo, el problema que se daría porque un nodo abandone la red sin que OLSR lo detecte queda resuelto.

Tras cubrir cómo se resolvieron la sincronización y la inicialización, se observa que el protocolo OLSR es monitorizado, pero no se modifica ni su comportamiento, ni sus mensajes. Esto implica que la solución propuesta genera una **sobrecarga nula** sobre el tráfico originado por OLSR una vez que el nuevo nodo esté configurado correctamente.

5.5.1.2. Construirla a partir de información incluida en los mensajes OLSR

Al estudiar la forma de proporcionar a los nuevos nodos de la red una tabla de

autoconfiguración se barajó una segunda opción.

Esta segunda propuesta consiste básicamente en que cada nodo anunciaría su propio bloque de direcciones IP libres en los mensajes de control OLSR.

De esta forma, un nodo que entre en la red sería capaz de conocer los bloques que corresponden a cada nodo de la red al recibir las actualizaciones OLSR. Así construiría su tabla de autoconfiguración al mismo tiempo que la de encaminamiento.

Los mensajes que necesitarían nuevos formatos son los siguientes:

Los mensajes HELLO, usados para que los nodos se den a conocer a sus vecinos inmediatos, tienen los campos mostrados en la Figura 5.1:

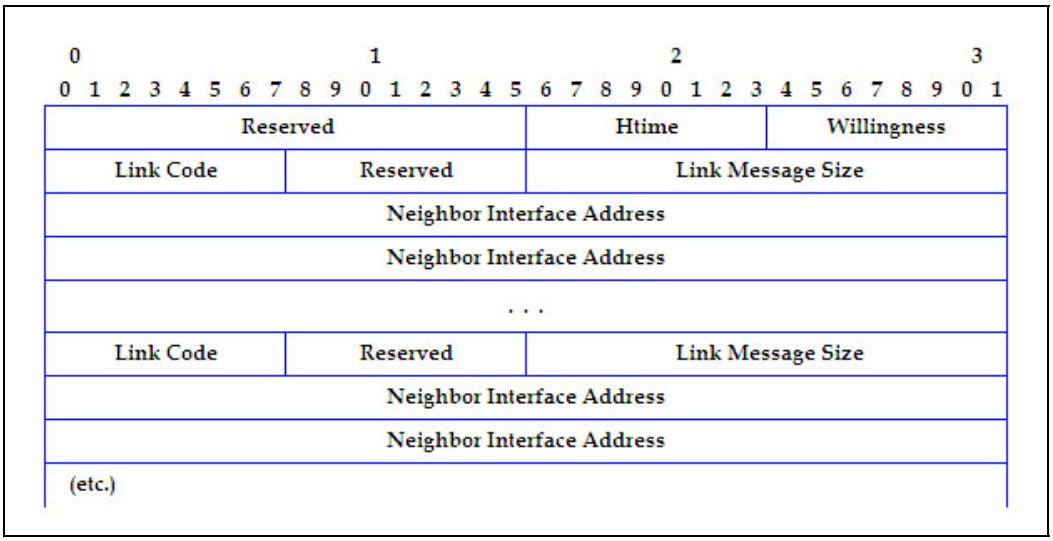

Fig. 5.1. Formato del mensaje HELLO de OLSR.

Para que el nodo que lo emite anuncie su bloque de direcciones IP libres, habría que incluir dos campos, la primera y la última dirección del bloque. En la Figura 5.2. se presenta el mensaje HELLO modificado.

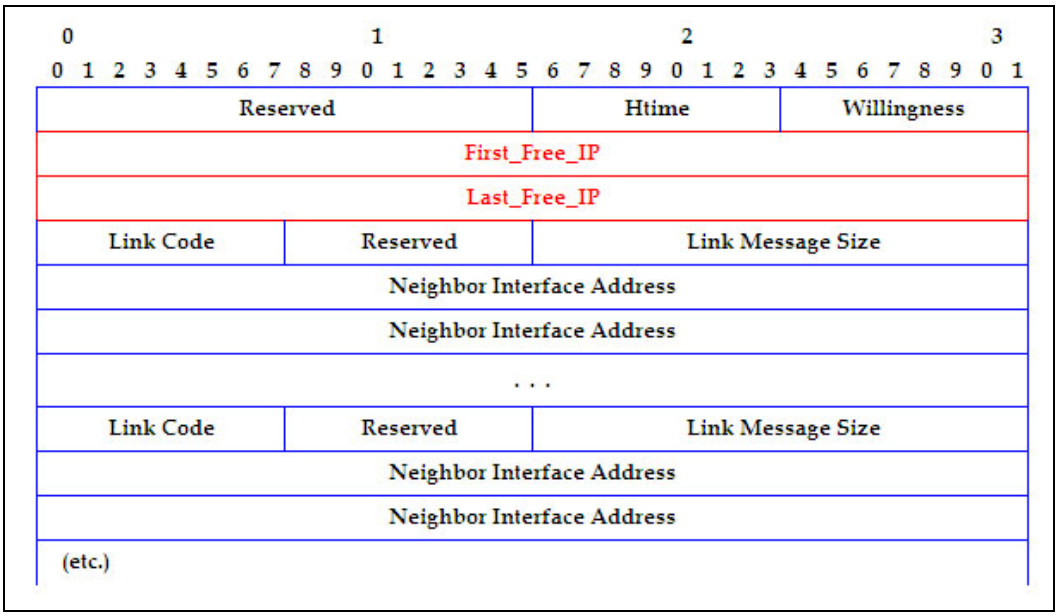

Fig. 5.2. Formato del mensaje HELLO de OLSR modificado.

El otro tipo de mensaje de control que se utiliza para difundir la información sobre la topología de la red es el mensaje TC. En la Figura 5.3. se muestra el formato de este tipo de mensaje:

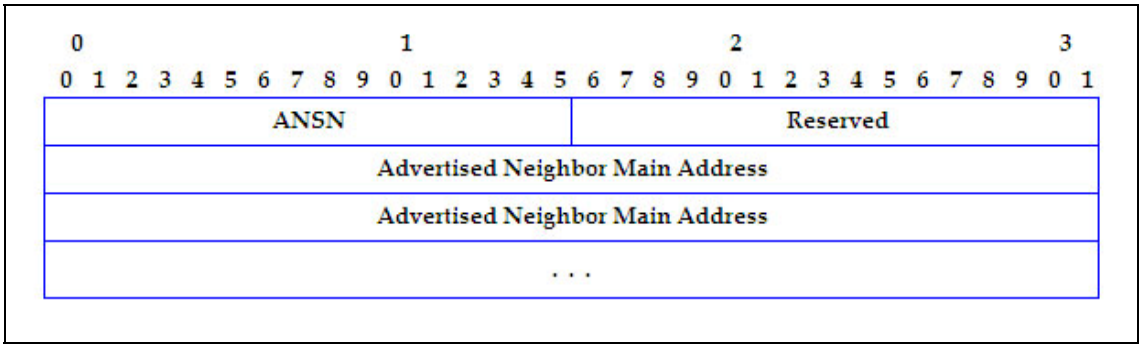

Fig. 5.3. Formato del mensaje TC de OLSR.

Para incluir el bloque de direcciones libres de cada nodo, se necesitarían dos campos más por cada nodo anunciado. El formato resultante sería como se muestra en la Figura 5.4.

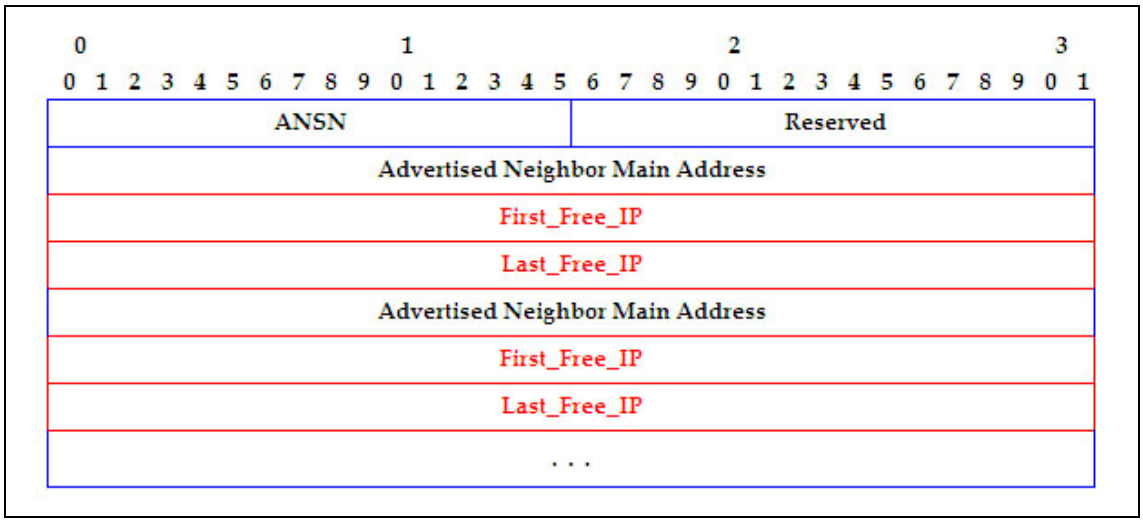

Fig. 5.4. Formato del mensaje TC de OLSR.

Modificar los mensajes del protocolo OLSR implicaría una mayor dificultad, y una dependencia mayor entre el protocolo de autoconfiguración y el protocolo de encaminamiento.

Aparte de esto, se puede ver que se generaría una sobrecarga con respecto al tráfico normal de OLSR. Por otra parte, esta sobrecarga está definida desde el primer momento: será de 64 bytes en los mensajes HELLO, y 64 bytes por cada vecino contenido en un mensaje TC. Conociendo este dato, se podría estudiar en qué tipos de redes la sobrecarga no sería un factor problemático, y si las ventajas superarían a las desventajas.

Realizar estos cambios tendría otras implicaciones. Por ejemplo, al difundir el bloque de direcciones libres de cada nodo, los nodos de la red no actualizarían de forma local más que su propio bloque de direcciones libres.

En lugar de suponer que al abandonar la red un nodo A, otro nodo B recogerá las direcciones que A deja disponibles, el resto de la red podría esperar a que B anuncie que de hecho las ha recogido. Por ello, podría incluirse algún método de comprobación, como podría ser el envío de varios mensajes *ping*

desde B hasta A, ya que el único nodo que debe asegurarse de la salida de A es B, y no toda la red.

Otra idea que podría desarrollarse sobre la base de difundir los bloques de direcciones libres de cada nodo es la posibilidad de que un nodo que quiera entrar a la red sea capaz de conocer el estado de ésta de antemano, escuchando los mensajes OLSR. Esto podría facilitar otros mecanismos de autoconfiguración más complejos, como podría ser el que el nodo que quiera entrar a la red eligiese al nodo de la red que más le interese, según sus criterios, como su servidor para que le facilite la entrada.

6. D2HCP: ESPECIFICACIÓN

6.1. Introducción

En este capítulo se detalla la especificación del protocolo D2HCP.

D2HCP es un protocolo de autoconfiguración que gestiona la entrada y salida de nodos en redes MANET.

El protocolo hace que los nodos de una red MANET colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existiendo un tipo de nodo especial que centralice la gestión de la misma.

Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo, la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR.

Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a una red intenta contactar con cualquier nodo ya perteneciente a ella, y podrá recibir varias respuestas de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia que supone la gestión distribuida.

La estructura del resto del capítulo, que contiene la especificación del protocolo D2HCP, es la siguiente: comienza con las estructuras de datos usadas,

continúa con una explicación de los mensajes que se intercambian entre los nodos para la entrada y salida de éstos, y a continuación se detalla cómo se lleva a cabo la sincronización en el protocolo.

A continuación se hablará de cómo se solucionan las posibles pérdidas de mensajes en la red usando los temporizadores adecuados y realizando determinadas acciones cuando estos expiran para reestablecer el proceso de autoconfiguración.

Para acabar, se explicará el formato de los mensajes y los diagramas de estados para cada modo de funcionamiento que puede asumir un nodo.

6.2. Estructuras de datos

Las estructuras de datos de este protocolo pueden clasificarse en las propias que maneja el mecanismo de autoconfiguración y las pertenecientes al protocolo de encaminamiento OLSR.

OLSR almacena internamente una tabla de rutas que es actualizada periódicamente. Esta tabla contiene información sobre la ruta a cada nodo, almacenada en los siguientes campos:

- *R_dest_addr*: Dirección IP del nodo destino.
- *R_next_addr*: Dirección IP del siguiente salto en la ruta.
- *R_dist*: Distancia al nodo destino.
- *R_iface_addr*: Dirección IP de la interfaz de salida al nodo destino.

Las estructuras necesarias para el mecanismo de la autoconfiguración son:

- Direcciones IP de las interfaces del nodo

- Máscara de red

- Free_IP_Blocks: tabla de bloques libres de cada nodo de la red. Tendrá la siguiente forma:

IP	Free_IP_blocks
.1	.1 - .64
.128	.128 - .254
.65	.65 - .127

6.3. Entrada y salida de los nodos

El protocolo usa un número concreto de mensajes para cada operación. Todas las operaciones están definidas buscando un óptimo funcionamiento y una baja latencia.

En esta sección se habla de cómo se establece la comunicación entre los nodos y los mensajes transmitidos durante la salida y la entrada de nodos en la red.

6.3.1. Entrada de nodos

La entrada de un nodo a la red implica la necesidad de encontrar un nodo que actúe como servidor. Una vez encontrado, éste le facilita la entrada proporcionándole un bloque de direcciones IP y una tabla *Free_IP_Blocks* que representa el estado de los nodos de toda la red.

Hasta que el nodo no tenga asignada una dirección IP, su comunicación con

los nodos que podrán actuar como servidores será a través de la capa MAC.

Este mecanismo de configuración utiliza 4 tipos de mensajes en la mayoría de los casos. Si no hay nodos en su rango con direcciones IP libres, se usarán 6 tipos de mensajes en total.

En la Figura 6.1 se muestra el esquema de intercambio de mensajes que se explica a continuación:

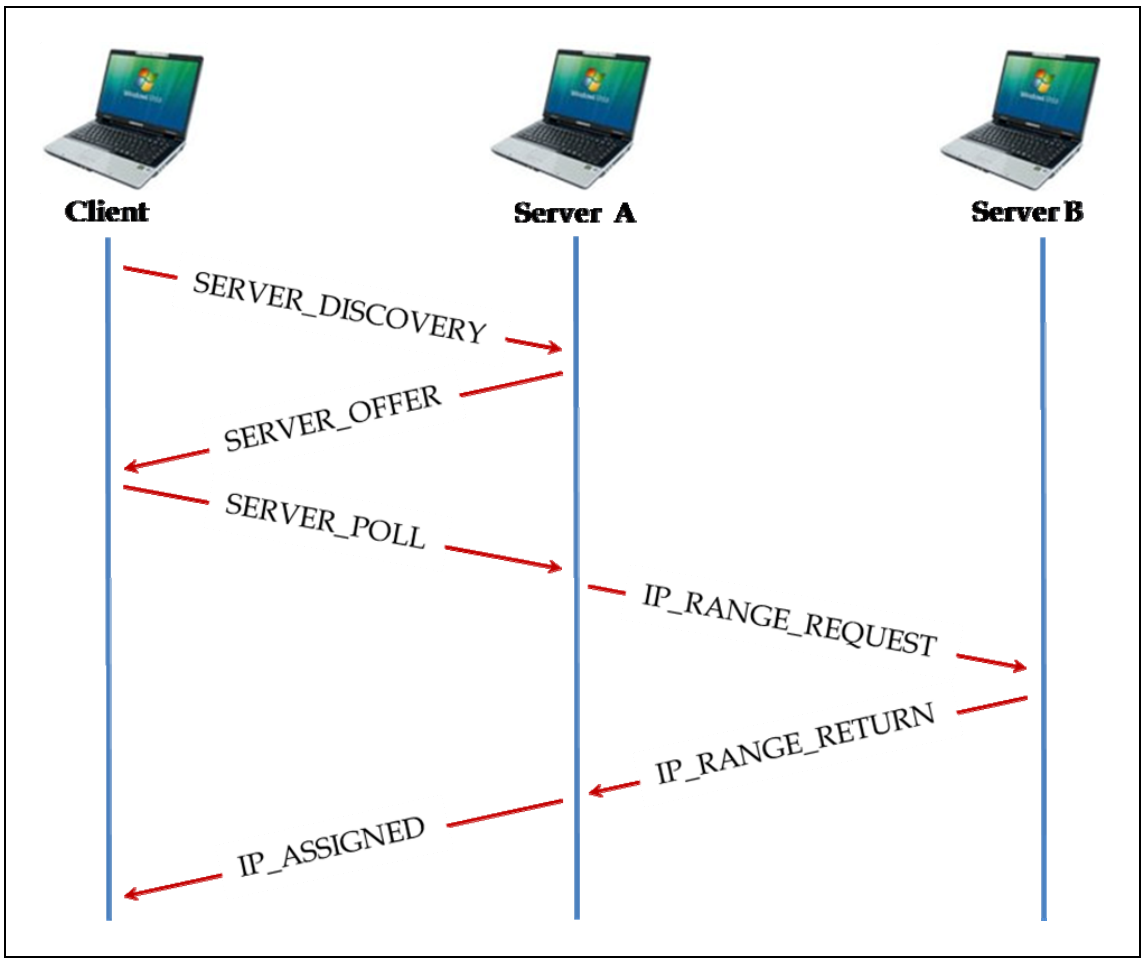

Fig. 6.1. Intercambio de mensajes en el proceso de entrada de un nodo a la red.

I. SERVER_DISCOVERY

El nodo cliente que desea entrar en una red inicia el proceso con un mensaje de

este tipo. Se transmite por la capa MAC, con la dirección de difusión como destino. En el mensaje se indican el número de direcciones IP que se necesitan (igual al número de interfaces).

Si el nodo dispone de más de una interfaz de red, el mensaje se transmite por todas ellas, usando el campo ID para que las diferentes interfaces no se confundan con varios nodos.

II. SERVER_OFFER

Los nodos de la red que reciben el mensaje SERVER_DISCOVERY responden con este mensaje, también usando la capa MAC, en el que ofrecen un número de direcciones IP. El número de direcciones ofrecidas es la mitad del rango disponible.

El mensaje SERVER_DISCOVERY incluye un campo *Count* que indica cuántos intentos se han realizado por parte del cliente. Dependiendo de su valor, los nodos servidores se comportarán del siguiente modo:

- *Count* = 1: El nodo servidor responderá con un SERVER_OFFER si dispone de direcciones suficientes y los campos R (*Ready*) y L (*Local*) tendrán el valor 1 (puede asignar las direcciones ofrecidas en este momento, y son direcciones del bloque del propio nodo).

- *Count* = 2: El nodo servidor responderá con un SERVER_OFFER si los campos pueden tomar el valor R = 1 y L = 1. En caso de no ser posible, responderá también si se da la situación de que tiene suficientes direcciones y R = 0, L = 1 (el servidor no puede asignar las direcciones en este momento, pero dispone de ellas).

- *Count* > 2: Si el nodo dispone de direcciones y por su estado es capaz,

enviará un SERVER_OFFER con R = 1, L = 1. Si puede, lo enviará con R=0, L = 1. Y por último, si no dispone de suficientes direcciones libres, enviará el mensaje con los campos R = 1, L = 0 (disponibilidad inmediata de las direcciones, pero las direcciones ofrecidas son de otro nodo de la red).

III. SERVER_POLL

Tras un tiempo de escucha, el nodo cliente habrá recibido varios mensajes SERVER_OFFER. De no ser así, volvería a intentarlo.

Ordenará los mensajes recibidos según los siguientes criterios:

- Se descartan los servidores que no estén disponibles, es decir, con R =0. Los SERVER_OFFER con R = 0 no se usan para poder responder con un SERVER_POLL, pero tienen la función de informar al cliente de que existe un nodo servidor de una red aunque en ese momento no pueda facilitarle el acceso.

- Se dan prioridad a las direcciones locales: preferirá los mensajes con el campo L = 1.

- Por último, se ordenarán por número de direcciones ofrecidas, de mayor a menor.

Según ese orden de preferencia, enviará al primer servidor un mensaje SERVER_POLL (de nuevo, por la capa MAC) para indicarle que le ha elegido para que le asigne un bloque de direcciones IP libres.

IV. IP_RANGE_REQUEST

Si las direcciones ofrecidas por el nodo servidor no eran propias, sino de un

tercer nodo de la red, con este mensaje se le solicitan formalmente a ese nodo. Al ser una comunicación entre dos nodos ya configurados correctamente, se realiza en la capa IP.

V. IP_RANGE_RETURN

Ese tercer nodo de la red autoriza al nodo que envió el mensaje IP_RANGE_REQUEST a asignar al nodo cliente el bloque de direcciones indicado en este mensaje. También es un mensaje enviado por IP.

VI. IP_ASSIGNED

Tras recibir el SERVER_POLL, si las direcciones ofrecidas eran del propio nodo servidor, o tras el mensaje IP_RANGE_RETURN en caso de que se haya tenido que pedir las direcciones a un tercer nodo, el nodo servidor envía este mensaje al cliente. Este mensaje es transmitido por la capa *MAC*.

En él se indica el bloque de direcciones libres que se le asignan al cliente, y la tabla *Free_IP_Blocks* que representa el estado de la red. La tabla que se transmite en este mensaje no refleja la entrada del nodo cliente.

Tras este intercambio de mensajes, el nodo cliente elige como su dirección IP la primera del bloque que se le ha asignado. En caso de tener más de una interfaz de red, usará las primeras del bloque en orden, y será la primera de todas la que use como dirección principal que identifica al nodo.

6.3.2. Salida de nodos

El mecanismo de salida de nodos no requiere del intercambio de ningún mensaje. El nodo que quiera abandonar la red no tiene que avisar a ningún otro nodo de su salida, evitando la sobrecarga que estos mensajes producen.

El resto de nodos de la red, se darán cuenta de la salida del nodo mediante las actualizaciones periódicas de rutas que el protocolo OLSR realiza cada cierto tiempo. Observarán que se ha perdido la ruta hacia ese nodo, y por lo tanto lo eliminarán de su tabla *Free_IP_Blocks*, añadiendo su bloque de direcciones libres al nodo que corresponda según se ha explicado en el punto anterior.

6.4. Sincronización

La sincronización se lleva a cabo monitorizando la tabla de rutas del protocolo de encaminamiento OLSR.

La entrada o salida de un nodo a la red se detecta cuando OLSR añade una nueva ruta a su tabla de encaminamiento, o borra una de las existentes. Al detectar la entrada o la salida de un nodo de la red, se actualiza localmente, y sin intercambiar ningún mensaje, la tabla *Free_IP_Blocks*.

Para ello, se siguen las siguientes reglas:

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Cuando se detecta la salida de un nodo, se debe eliminar su entrada, y

actualizar la del nodo a quien corresponden las direcciones IP que han quedado disponibles.

Al detectar la entrada de un nuevo nodo, se creará una nueva entrada en la tabla para él, y se actualizará el bloque de direcciones libres del nodo que le facilitó su dirección IP. Para saber quién fue ese nodo que actuó como servidor, basta con buscar qué nodo tiene la dirección IP del nuevo nodo en su bloque de direcciones libres.

6.5. Formato de los mensajes

Todos los mensajes enviados en el protocolo van empaquetados con el formato presentado en la Figura 6.2:

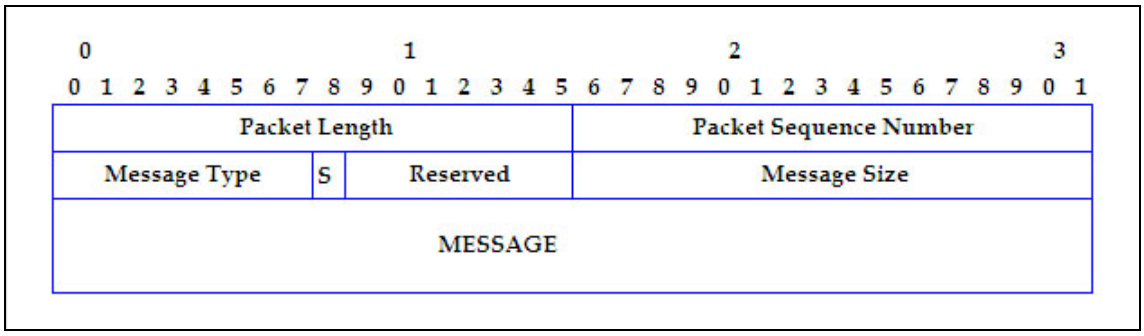

Fig. 6.2. Paquete del protocolo D2HCP.

Estos mensajes serán encapsulados a su vez con las cabeceras correspondientes a la capa MAC o TCP/IP, dependiendo del tipo de mensaje que se incluya en el campo MESSAGE.

6.5.1. Cabecera del paquete

La primera fila de la Figura 6.2. contiene los campos de la cabecera del paquete.

- *Packet Length*: Longitud del paquete, incluyendo la cabecera (2 bytes).

- *Packet Sequence Number*: Número de secuencia (2 bytes). En cada mensaje diferente que envía un nodo este campo se incrementa en uno. Sirve para poder detectar paquetes duplicados.

6.5.2. Cabecera de los mensajes

La segunda fila de la Figura 6.2. constituye la cabecera para cada uno de los mensajes del protocolo:

- *Message Type*: Tipo de Mensaje (1 byte).

- *S (Security)*: Reservado para la implementación de seguridad. (1 bit).

- *Reserved*: Reservado para futuras ampliaciones de funcionalidad (7 bits).

- *Message Size*: Tamaño del Mensaje (2 bytes).

A continuación se muestra el formato de cada tipo de mensaje, contenidos en el campo MESSAGE.

6.5.3. SERVER_DISCOVERY

La Figura 6.3. muestra el formato del mensaje SERVER_DISCOVERY.

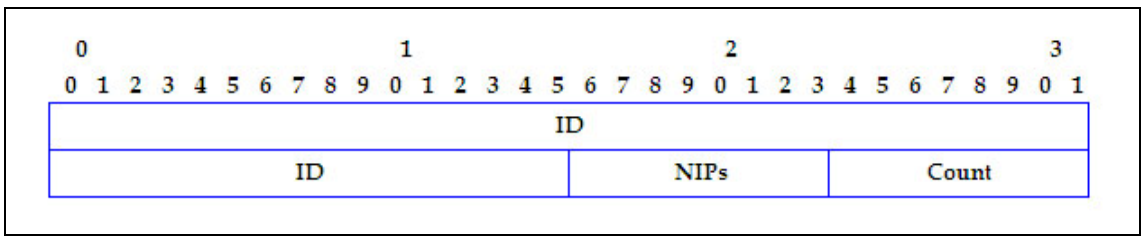

Fig. 6.3. Formato del mensaje SERVER_DISCOVERY.

- *ID*: Identificador del nodo (6 bytes). El nodo debe elegir, si tiene más de una, la dirección MAC de una de sus interfaces. Este campo identificativo tiene el mismo valor para cualquier mensaje SERVER_DISCOVERY y SERVER_POLL emitido por el nodo aunque se haga desde distintas interfaces.

- *NIPs*: Cantidad de direcciones IP solicitadas por el nodo. Será igual al número de interfaces del nodo cliente (1 byte).

- *Count*: Número de veces que se ha intentado la petición SERVER_DISCOVERY (1 byte).

6.5.4. SERVER_OFFER

La Figura 6.4. muestra el formato del mensaje SERVER_OFFER.

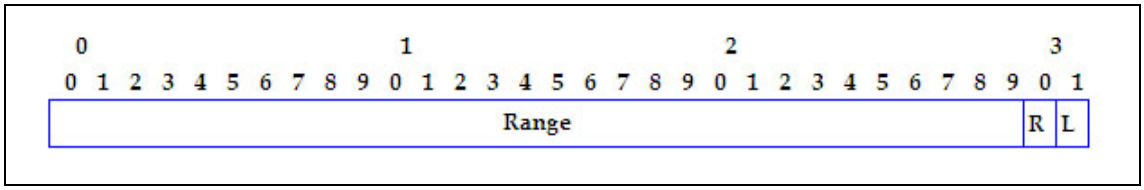

Fig. 6.4. Formato del mensaje SERVER_OFFER.

- *Range*: Número de direcciones IP que se ofrecen (30 bits).

- *Ready* (R): Indica si el nodo que ofrece las direcciones IP está listo para asignarlas de inmediato, o si las ofrece para comunicar su existencia aunque en este momento no esté en disposición de asignarlas (1 bit).

- *Local* (L): Indica si el rango ofrecido es del nodo emisor, o por el contrario se pedirá a su vez a un tercer nodo de la red (1 bit).

6.5.5. SERVER_POLL

La Figura 6.5. muestra el formato del mensaje SERVER_POLL.

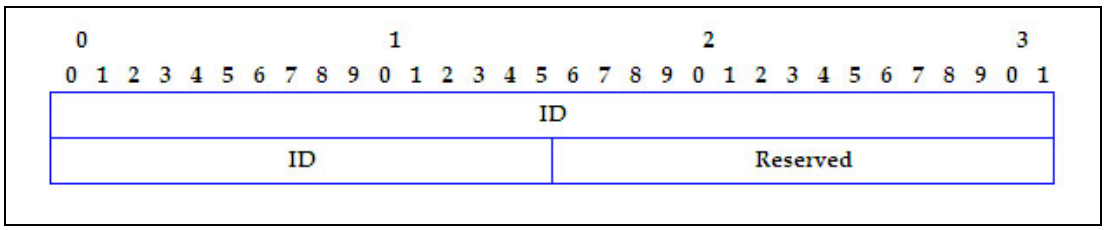

Fig. 6.5. Formato del mensaje *SERVER_POLL.*

- *ID*: Identificador del nodo (6 bytes) Es el mismo identificador que el elegido en el mensaje SERVER_DISCOVERY.

- *Reserved*: Reservado para futuras implementaciones (2 bytes).

6.5.6. IP_ASSIGNED

La Figura 6.6. muestra el formato del mensaje IP_ASSIGNED.

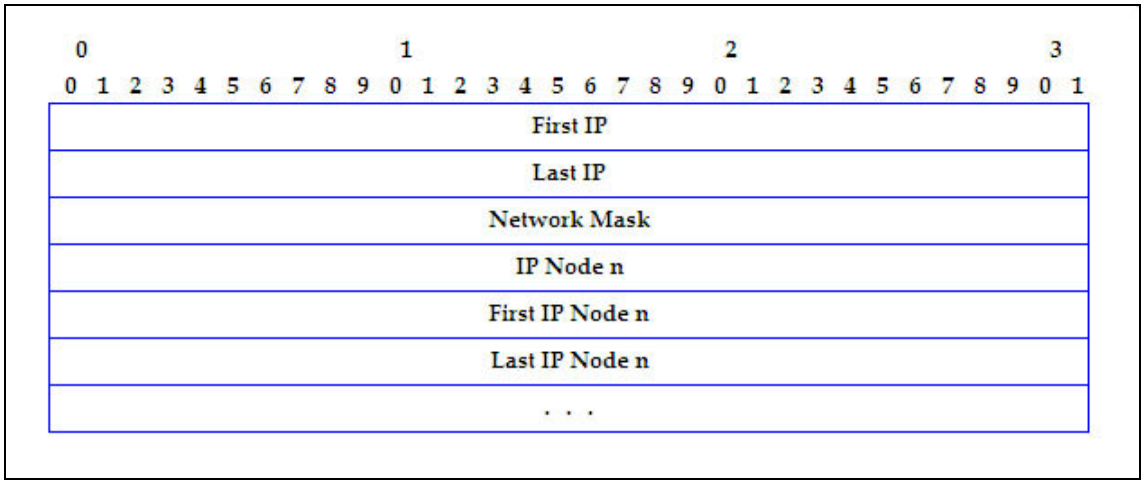

Fig. 6.6. Formato del mensaje IP_ASSIGNED.

- *First IP*: Dirección IP inicial del bloque de direcciones libres (4 bytes).

- *Last IP*: Dirección IP final del bloque de direcciones libres (4 bytes).

- *Network Mask*: Máscara de red (4 bytes).

- *IP Node n, First IP Node n, Last IP Node n*: Representa una entrada de la tabla de bloques libres de todos los nodos de la red. Cada nodo está representado por estos 3 campos de 4 bytes, siendo cada uno: la dirección IP del nodo, la dirección IP inicial y la final de su bloque de direcciones libres, respectivamente.

6.5.7. IP_RANGE_REQUEST

La Figura 6.7. muestra el formato del mensaje IP_RANGE_REQUEST.

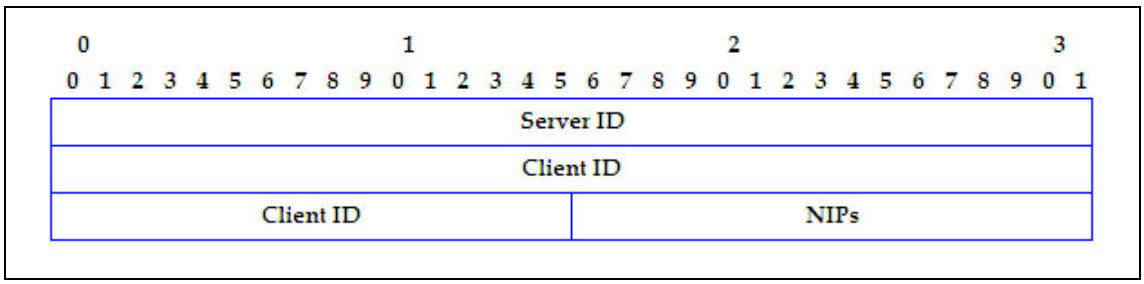

Fig. 6.7. Formato del mensaje IP_RANGE_REQUEST.

- *Server IP*: Dirección del servidor que solicita el rango para cliente (4 bytes). Puede ser diferente de la dirección IP del emisor del mensaje, al tratarse de redes multisalto.

- *Client ID*: Identificador del nodo cliente. Tiene el mismo valor que el campo ID de los mensajes SERVER_DISCOVERY y SERVER_POLL (6 bytes).

- *NIPs*: Número de direcciones IP solicitadas (2 bytes).

6.5.8. IP_RANGE_RETURN

La Figura 6.8. muestra el formato del mensaje IP_RANGE_RETURN.

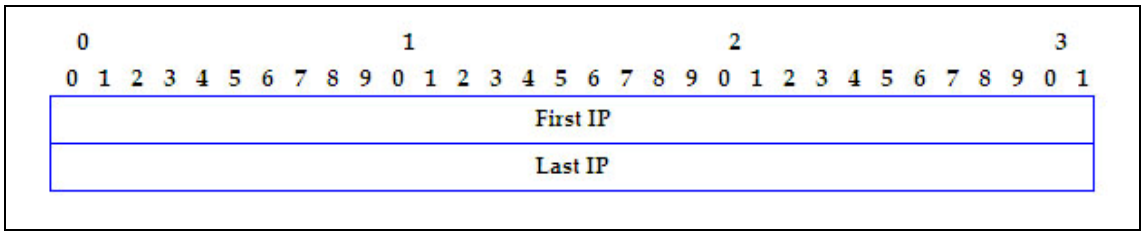

Fig. 6.8. Formato del mensaje IP_RANGE_RETURN.

- *First IP*: Dirección IP inicial del bloque de direcciones libres (4 bytes).

- *Last IP*: Dirección IP final del bloque de direcciones libres (4 bytes).

6.6. Temporizadores

El carácter inalámbrico y móvil de las redes MANET provoca que en este tipo de redes se den a menudo situaciones en las que se pierden mensajes, o tardan en llegar a su destino más que el tiempo estimado. Por ello, exponemos a continuación una serie de temporizadores usados para resolver situaciones de este tipo:

- *SERVER_DISCOVERY_TIMER*: Tras enviar el mensaje SERVER_DISCOVERY, el nodo cliente iniciará este temporizador al pasar al estado WAITING_REPLY. Durante este tiempo el nodo está a la espera de mensajes SERVER_OFFER de los posibles nodos cercanos pertenecientes a una red. Cuanto más largo sea el temporizador mayor tiempo se dedicará a recibir mensajes de este tipo, por lo que habrá más

para procesar y, por lo tanto, será más fácil obtener un bloque de direcciones. Pero también implica el aumento de la latencia al obtener una dirección IP.

Si este temporizador expira y el nodo cliente no ha recibido ningún SERVER_OFFER, bien por pérdidas de mensajes, o porque no hay ningún nodo servidor, volverá a enviar un nuevo SERVER_DISCOVERY. Esta acción se repetirá un número máximo de veces (SDISCOVERY_MAX_RETRY) y, si sigue sin recibir mensajes, iniciará su propia red.

- *SERVER_OFFER_TIMER*: Tras el mensaje SERVER_OFFER, el nodo pasa al estado WAITING_POLL, e inicia este temporizador. Al expirar, el estado cambiará a IDLE.

- *SERVER_POLL_TIMER*: Una vez enviado el mensaje SERVER_POLL, el nodo cliente esperará al mensaje IP_ASSIGNED el tiempo que determine este temporizador. Si este mensaje no llega, se retransmitirá el SERVER_POLL hasta un número máximo de intentos, definido como SPOLL_MAX_RETRY. Si se supera el número máximo de intentos, comenzará el proceso de configuración de nuevo.

- *IP_RANGE_REQUEST_TIMER*: El nodo servidor que manda un mensaje IP_RANGE_REQUEST a otro nodo de la red inicia este temporizador en ese momento. Al igual que con el SERVER_POLL_TIMER, si expira este temporizador se reenviará el mensaje IP_RANGE_REQUEST hasta un número máximo de intentos, RREQUEST_MAX_RETRY.

- *ACCEPTED_OFFER_TIMER*: Este temporizador se activa tras enviar un mensaje IP_ASSIGNED o un mensaje IP_RANGE_RETURN. Durante este tiempo, el nodo servidor no puede responder a solicitudes de tipo

SERVER_POLL o IP_RANGE_REQUEST. Esta restricción se levantará al expirar el temporizador (la oferta caducó sin ser aceptada), o al detectar que un nodo con la primera dirección IP de las ofrecidas ha entrado en la red (el bloque de direcciones ofrecido fue aceptado).

Hay que tener en cuenta que aunque no pueda asignar direcciones IP, el nodo servidor seguirá respondiendo a peticiones SERVER_DISCOVERY dando el valor 0 al campo R (READY) en el mensaje SERVER_OFFER. De este modo, se informa al nodo cliente de la existencia del servidor, aunque no sea capaz de asignar direcciones IP inmediatamente.

- *NODE_DOWN_TIMER*: Cuando OLSR borra la ruta hacia un nodo, no se elimina inmediatamente de la tabla *Free_IP_Blocks*. En su lugar, se inicia este temporizador. Si antes de que el temporizador expire se vuelve a descubrir una ruta hacia el nodo, eso quiere decir que desapareció momentáneamente, pero no abandonó la red. Por lo tanto, se cancela la eliminación de la tabla *Free_IP_Blocks*. En el caso de que el temporizador expire y no se haya recuperado una ruta, se da al nodo por perdido y se elimina su entrada de la tabla *Free_IP_Blocks*, actualizando las que correspondan.

- *INIT_TABLE_TIMER*: Al recibir la tabla de autoconfiguración *Free_IP_Blocks* en el mensaje IP_ASSIGNED, el nodo cliente activa este temporizador. Durante ese tiempo, la tabla contiene nodos para los que OLSR aún no tiene una ruta conocida. Al expirar el temporizador, se comprueba qué nodos de la tabla *Free_IP_Blocks* no tienen entrada en la tabla de rutas de OLSR: esos nodos se eliminan (actualizando las entradas que correspondan), ya que son nodos que pertenecían a la red al recibir la tabla, y la han abandonado antes de que OLSR supiese de su existencia.

- *INIT_ASSIGN_TIMER*: Este temporizador lo usan tanto el nodo cliente

como el servidor cuando han recibido o asignado, respectivamente, un bloque de direcciones IP. Es decir, el servidor lo inicializa al comprobar la entrada de un nodo con la primera dirección IP del bloque ofrecido en el mensaje IP_ASSIGNED, y el cliente lo inicia tras recibir el mensaje IP_ASSIGNED y configurar su dirección.

Así se da tiempo a que toda la red pueda actualizar su tabla *Free_IP_Blocks* antes de que se produzcan más cambios. Durante ese tiempo, ignorarán mensajes SERVER_POLL o IP_RANGE_REQUEST, aunque responderán a los SERVER_DISCOVERY.

- *NODE_DOWN_ASSIGN_TIMER*: Cuando un nodo ya configurado detecta la salida de otro, y comprueba que le corresponde recoger las direcciones IP que quedan libres, pone en marcha este temporizador. Más concretamente, el temporizador se activará cuando se detecte la eliminación de la tabla de encaminamiento OLSR, es decir, se activará al mismo tiempo que el temporizador NODE_DOWN_TIMER.

  Hasta que no expire, el nodo ignorará las peticiones SERVER_POLL e IP_RANGE_REQ. De ese modo, se dará un margen de tiempo para asegurarse de que todos los nodos en la red detectan la salida mencionada y actualizan su tabla *Free_IP_Blocks*, antes de asignárselas a algún otro nuevo nodo. Por lo tanto, la duración de este temporizador debe ser mayor que la del NODE_DOWN_TIMER para asegurar que el resto de nodos de la red no sólo han detectado la eliminación de una ruta sino que lo han eliminado de la tabla *Free_IP_Blocks*.

  El nodo seguirá respondiendo a los mensajes SERVER_DISCOVERY fijando el valor 0 en el campo R del mensaje SERVER_OFFER.

- *SLEEP_TIMER*: Temporizador usado por un nodo cliente cuando detecta nodos cercanos pertenecientes a alguna red, pero que no están en disposición de asignar direcciones IP en este momento. De este modo, da un margen de tiempo para permitir que acaben los procesos que les impiden asignar direcciones.

6.7. Diagramas de estado

Dependiendo de si están en proceso de entrar en la red, o si ya pertenecen a ésta, se distinguen dos tipos de nodos: cliente y servidor. En los siguientes apartados se muestran y explican los diagramas de estados que rigen el comportamiento de ambos tipos de nodos.

El estado en el que se encuentra un nodo cambia al producirse envíos o recepciones de mensajes, o cuando determinados temporizadores expiran.

6.7.1. Nodo servidor

Llamamos nodo servidor a todos los nodos de la red que están correctamente configurados, es decir, que poseen una dirección IP válida con la que comunicarse con el resto de nodos, y un bloque de direcciones IP libres.

Con este bloque de direcciones libres facilitan el acceso a los nuevos nodos, que llamaremos clientes.

El diagrama de estados se muestra en la Figura 6.9:

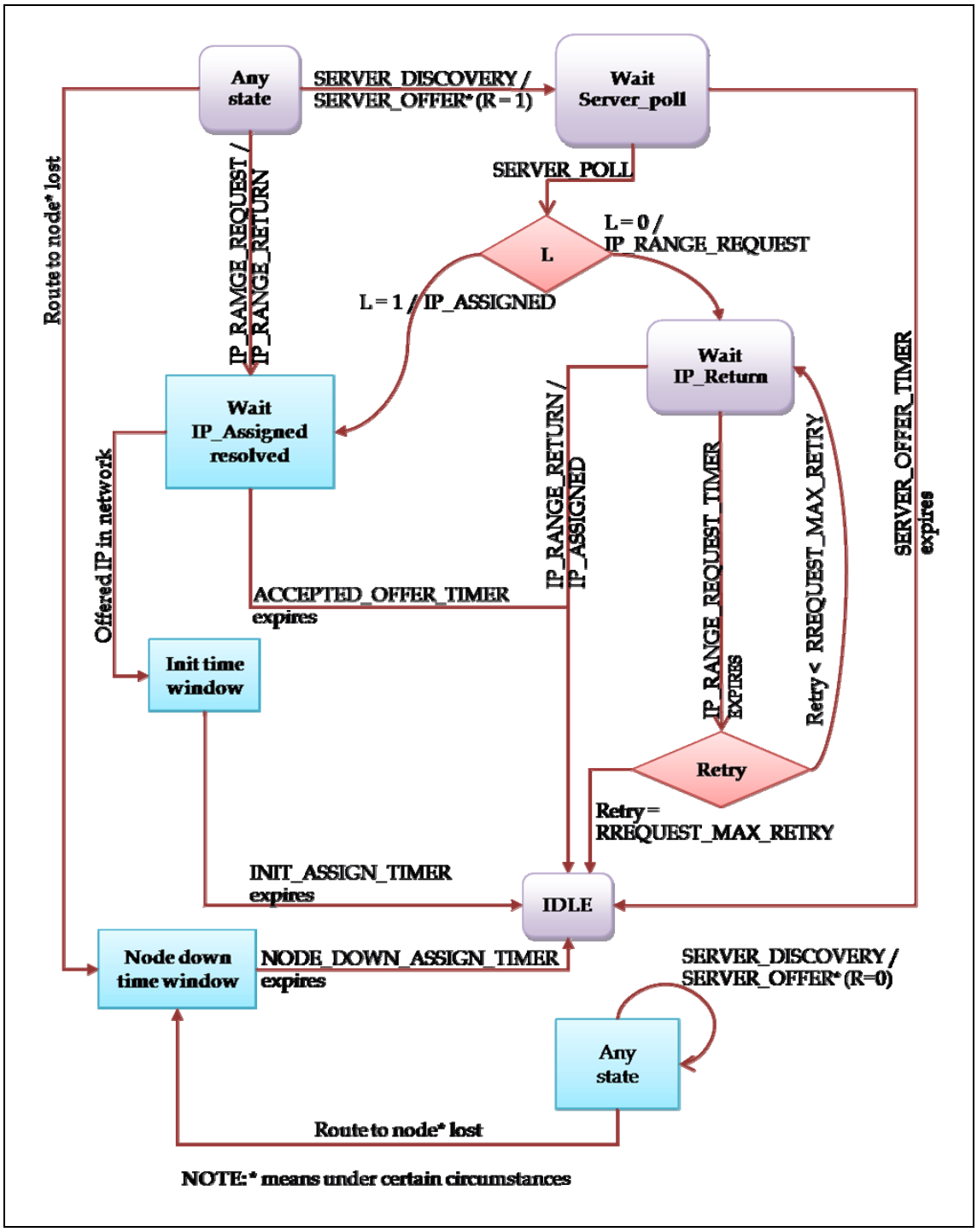

Fig. 6.9. Diagrama de estados del nodo servidor.

Como puede verse, existen dos tipos de estados: los representados con rectángulos redondeados y claros, y los que se encierran en rectángulos con esquinas y algo más oscuros. Los llamaremos estados del tipo ready o

not_ready, respectivamente.

Desde cualquier estado, el nodo servidor está a la espera de mensajes SERVER_DISCOVERY. Responderá siempre con un mensaje SERVER_OFFER, pero dependiendo de si el estado actual del nodo es de tipo ready o not_ready, responderá dando al campo R (READY) el valor 1 ó 0. De este modo, un nodo siempre anuncia su presencia, aunque en ese preciso momento no sea capaz de asignar direcciones IP a un cliente. Esto se indica en el diagrama de estados con los estados Any state.

I. Any state (ready)

Desde cualquier estado de tipo ready, el nodo servidor responderá a un mensaje SERVER_DISCOVERY con uno de tipo SERVER_OFFER (campo R con valor 1). Eso hará que el servidor pase al estado Wait SERVER_POLL.

Un nodo puede responder al mismo tiempo peticiones SERVER_DISCOVERY de nodos diferentes, y estar a la espera de cualquiera de los correspondientes SERVER_POLL.

También se responderán a mensajes de tipo IP_RANGE_REQUEST con uno del tipo IP_RANGE_RETURN. Esto significa que si el nodo se encuentra en cualquier estado ready, dará la mitad de su bloque de direcciones libres a cualquier otro nodo de la red sin direcciones propias y que necesita facilitar la entrada a un cliente.

II. Any state (not_ready)

Mientras el nodo se encuentre en un estado de tipo *not_ready*, responderá a los mensajes SERVER_DISCOVERY con un mensaje SERVER_OFFER dando el valor 0 al campo R.

III. Wait SERVER_POLL

En este estado el nodo espera un tiempo determinado por el temporizador SERVER_OFFER_TIMER la recepción de un mensaje SEVER_POLL. Este mensaje indica que el cliente a quien envió el SERVER_OFFER le ha elegido como su servidor para el proceso de autoconfiguración.

Tras la recepción del mensaje, el nodo enviará un mensaje IP_ASSIGNED al cliente en caso de tener direcciones disponibles localmente (el campo L del mensaje SERVER_OFFER tenía valor 1); y pasará al estado Wait IP_Assigned resolved. Si las direcciones que ofreció no eran locales, deberá pedirlas a un nodo de la red con el mensaje IP_RANGE_REQUEST; y pasará a estar en el estado Wait IP_RReturn.

En caso de que no se reciba ningún mensaje SERVER_OFFER antes de que el temporizador expire, el nodo pasará a estar en el estado IDLE.

IV. Wait IP_ASSIGNED resolved

En este estado, de tipo not_ready, el nodo está esperando a conocer si el nodo cliente recibió correctamente el bloque de direcciones ofrecido mediante un mensaje IP_ASSIGNED, o a través de un mensaje IP_RANGE_RETURN y un intermediario. El resultado puede ser que el cliente se haya configurado correctamente, o que no haya recibido el bloque de direcciones.

Esto último puede producirse por varios motivos, como pueden ser problemas de interferencias en la recepción del mensaje, moviendo del nodo cliente fuera del rango de cobertura, etc.

Si aparece un nuevo nodo en la red usando la primera dirección IP del

bloque ofrecido en el mensaje IP_ASSIGNED o IP_RANGE_RETURN, significa que el nodo cliente terminó de configurarse. En ese momento el nodo servidor cambia su estado a *Init time window*.

Si el nodo no fue capaz de terminar el proceso de autoconfiguración, el temporizador ACCEPTED_OFFER_TIMER expirará. El nodo pasa en ese caso al estado IDLE.

V. Init time window

Este estado sirve para dar un margen de tiempo que permita que todos los nodos de la red sean capaces de detectar la entrada del cliente recientemente configurado, antes de volver a dividir el propio bloque de direcciones IP libres.

Si no se diese este margen, y se atendiesen peticiones nuevas de inmediato, podrían darse problemas de sincronización si otros nodos detectasen las nuevas incorporaciones a la red en un orden incorrecto.

VI. Wait ip_return

En este estado el nodo está pendiente de recibir un mensaje IP_RANGE_RETURN.

Cuando reciba ese mensaje, en el que un nodo de la red le indica un bloque que puede ofrecer al cliente en espera, enviará al cliente un mensaje IP_ASSIGNED. Tras esto, el nodo habrá terminado su función como servidor, y pasará al estado IDLE.

Si no se recibe el mensaje IP_RANGE_RETURN antes de que el temporizador IP_RANGE_REQUEST_TIMER expire, se volverá a intentar la petición enviando de nuevo un mensaje IP_RANGE_REQUEST un número máximo de veces RREQUEST_MAX_RETRY. Estos sucesivos intentos se envían en cada

ocasión a un nodo diferente. Si se sobrepasa el límite de intentos, entonces el nodo desistirá y cambiará su estado a IDLE.

VII. IDLE

Este es el estado de reposo, o en el que el nodo está ocioso. Cuando se encuentra en este estado, el nodo no está realizando ninguna operación relacionada con la autoconfiguración.

Se trata por lo tanto de un estado de espera.

VIII. Node down time window

Este nodo proporciona un margen de tiempo cuando el nodo debe recoger las direcciones IP de un nodo que ha abandonado la red.

Más exactamente, el nodo cambia a este estado al detectar que ha perdido la ruta hacia un nodo de cuyo bloque de direcciones es responsable. Esta transición se hace desde cualquier otro estado, ya sea de tipo ready o not_ready.

Tras el tiempo determinado por el temporizador NODE_DOWN_ASSIGN_TIMER, el nodo volverá al estado de reposo IDLE.

No se debe confundir este temporizador con el NODE_DOWN_TIMER. Aunque se inicien al mismo tiempo al detectar el mismo evento, los procesos involucrados son independientes.

6.7.2. Nodo cliente

El procedimiento que sigue un nodo que desea acceder a una red se describe en la Figura 6.10:

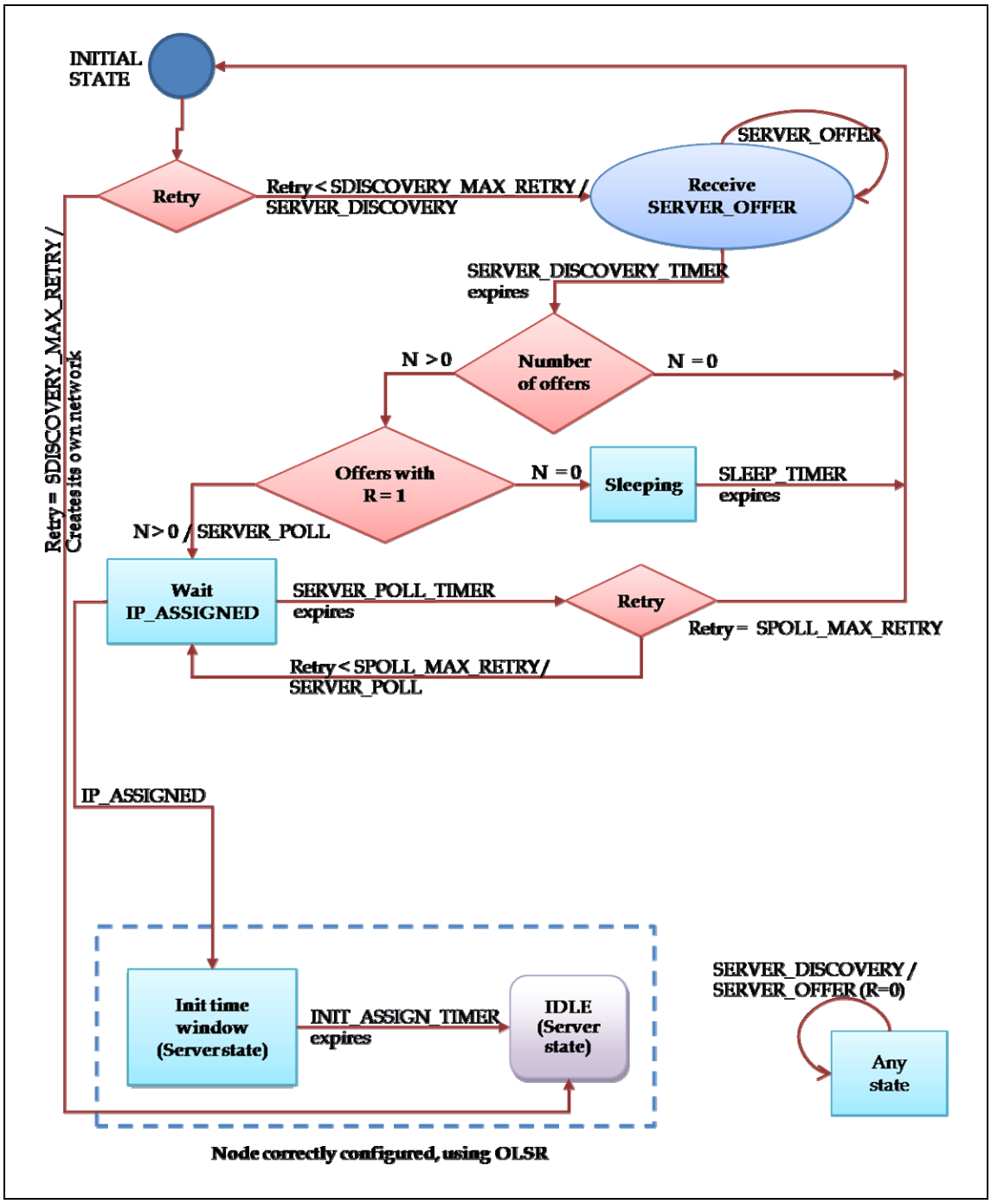

Fig. 6.10. Diagrama de estados del nodo cliente.

El diagrama cuenta con estados de tipo not_ready, explicados en el diagrama de estados del nodo servidor y representados con el mismo estilo de rectángulos. Al igual que si se tratase de un nodo ya configurado, el nodo que está en proceso de configuración de su dirección IP responde a peticiones

SERVER_DISCOVERY con mensajes SERVER_OFFER. En estos mensajes se da el valor 0 al campo R, ya que el nodo no está en disposición de asignar direcciones IP, solo pretende anunciar su presencia.

I. Initial state

Estado inicial, en el que comienza el proceso de autoconfiguración. Si el número de intentos es menor al máximo, SDISCOVERY_MAX_RETRY, entonces el nodo emite un mensaje SERVER_DISCOVERY por cada una de sus interfaces de red que vayan a utilizar la red MANET. Cambia su esado a Receive SERVER_OFFER.

Si se ha llegado al límite de intentos, entonces el nodo desiste de su intención de encontrar una red a la que unirse y crea una nueva. Pasará a ser un nodo servidor, comenzando en el estado IDLE del diagrama de estados del servidor.

II. Receive SERVER_OFFER

Se trata de un estado de espera, durante el cual se recogen los mensajes SERVER_OFFER de posibles nodos próximos. Al terminar la espera, determinada por el temporizador SERVER_DISCOVERY_TIMER, se procesan las respuestas.

Si no se ha recibido ninguna respuesta SERVER_OFFER, se vuelve al estado inicial. Si hay alguna oferta, se comprueba el número de ellas con el valor 1 en el campo R.

Si de entre las ofertas ninguna tenía el bit R a 1, eso quiere decir que hay nodos cercanos pertenecientes a una red, pero de momento no son capaces de asignar direcciones IP. Por lo tanto, el nodo pasa al estado Sleeping.

Si había alguna oferta con el bit R a 1, se ordenan los servidores por

preferencia y se envía un mensaje SERVER_POLL al primero de ellos. En este caso el nodo cambia su estado a Wait IP_ASSIGNED.

Este estado no es de ninguno de los tipos explicados en el diagrama de estados de un nodo cliente. Esto significa que no responde a mensajes SERVER_DISCOVERY, ya que de momento no conoce si hay alguna red cercana a la que unirse.

III. Sleeping

El nodo pausa sus intentos de entrar en la red durante el tiempo determinado por el temporizador SLEEP_TIMER. Esto es así porque se han recibido mensajes SERVER_OFFER de nodos cercanos que de momento no son capaces de asignar direcciones IP, y lo que se pretende que tras este tiempo ya sean capaces de facilitar la entrada a la red.

Al expirar el temporizador, el nodo volverá al estado inicial.

IV. Wait IP_ASSIGNED

En este estado, el cliente se encuentra a la espera de un mensaje IP_ASSIGNED por parte del servidor al que se envió el mensaje SERVER_POLL.

Si no llega el mensaje esperado, el temporizador SERVER_POLL_TIMER expira. En ese caso, se volverá a intentar enviar un SERVER_POLL al siguiente servidor de la lista generada tras la finalización del temporizador SERVER_DISCOVERY_TIMER. Si se acaba la lista de servidores, o se llega al límite de intentos SPOLL_MAX_RETRY, el nodo vuelve al estado inicial.

Al recibir el mensaje IP_ASSIGNED, el nodo configura su dirección (o direcciones, en caso de disponer de varias interfaces de red). En ese momento, ya participa normalmente en la red, y pasa a ser un nodo servidor.

El estado con el que comienza su comportamiento como servidor es el *Init time window*.

7. SIMULACIONES Y RESULTADOS

Puesto que en estas redes es impredecible el número de nodos que formarán la red, la escalabilidad del protocolo es una de las principales cuestiones a tener en cuenta. Por lo tanto, es fundamental evaluar el impacto del incremento del número de nodos en la red en distintos parámetros como la latencia en la asignación de dirección, la sobrecarga debido al tráfico de control o el retardo en la sincronización. Además del número de nodos presentes en la red, es necesario tener en cuenta la frecuencia de la salida y entrada de nodos en la red. Cuando un nodo sale de la red, se deben actualizar las tablas de direcciones libres de la red. Si esto no se realiza de forma rápida, los nodos de la red pueden no atender peticiones de nuevas entradas de la red por interpretar que no disponen de direcciones libres.

Para evaluar el comportamiento del protocolo D2HCP se ha utilizado el simulador de redes Network Simulator 3 (NS-3) [48]. Se han simulado diferentes escenarios de redes MANET para evaluar el rendimiento bajo diferentes circunstancias.

7.1. Escenarios de simulación

La tabla 7.1 resume los parámetros principales utilizados durante las simulaciones.

A la hora de realizar estas simulaciones se ha mantenido constante el número de entradas en la red por unidad de tiempo. Este factor es importante, particularmente en redes de alta densidad.

Parámetro	Valor
Área de simulación	1500 m x 1500 m
Número de nodos móviles	De 50 a 1600
Patrón de movilidad	*Random Waypoint (setdest)*
Protocolo de encaminamiento	OLSR
Alcance o Cobertura del nodo	125 metros
Número de Simulaciones	10

Tabla 7.1.Parámetros de las simulaciones.

7.2. Resultados

En primer lugar se evaluó la latencia en el proceso de asignación de direcciones. Las Figuras 7.1 y 7.2 muestran la evolución del valor de la latencia en función del número de nodos en la red. En la Figura 7.1 se presentan los valores empleando direcciones IPv4 clase C, es decir, con 254 direcciones disponibles. En la Figura 7.2 se han utilizado direcciones IPv4 clase B, que proporcionan 65534 direcciones. Se observa en la Figura 7.1 que el aumento del tiempo de asignación de dirección empieza a crecer de forma más rápida a partir de los 125 nodos. Sin embargo, utilizando direcciones clase B (Figura 7.2), el tiempo experimenta un crecimiento muy leve hasta los 1600 nodos que se han simulado.

Estos resultados indican que el parámetro que condiciona la latencia en mayor medida es el porcentaje de direcciones ocupadas. Cuando este porcentaje se aproxima al 50%, el número de nodos que no disponen de direcciones que ofrecer aumenta de forma directamente proporcional. Esto no permite realizar una asignación local y es necesario solicitar la dirección a otro nodo, aumentando el tiempo necesario para completar el proceso. En todo caso la latencia promedio en el proceso de asignación de direcciones es baja.

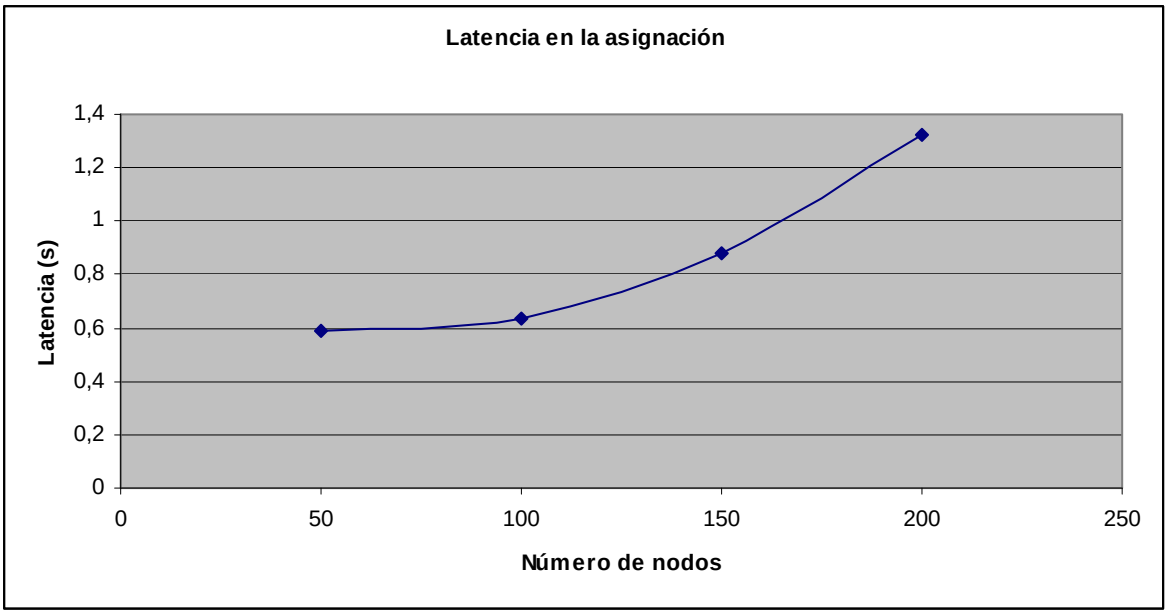

Fig. 7.1. Latencia en la asignación de direcciones IPv4 en una red clase C.

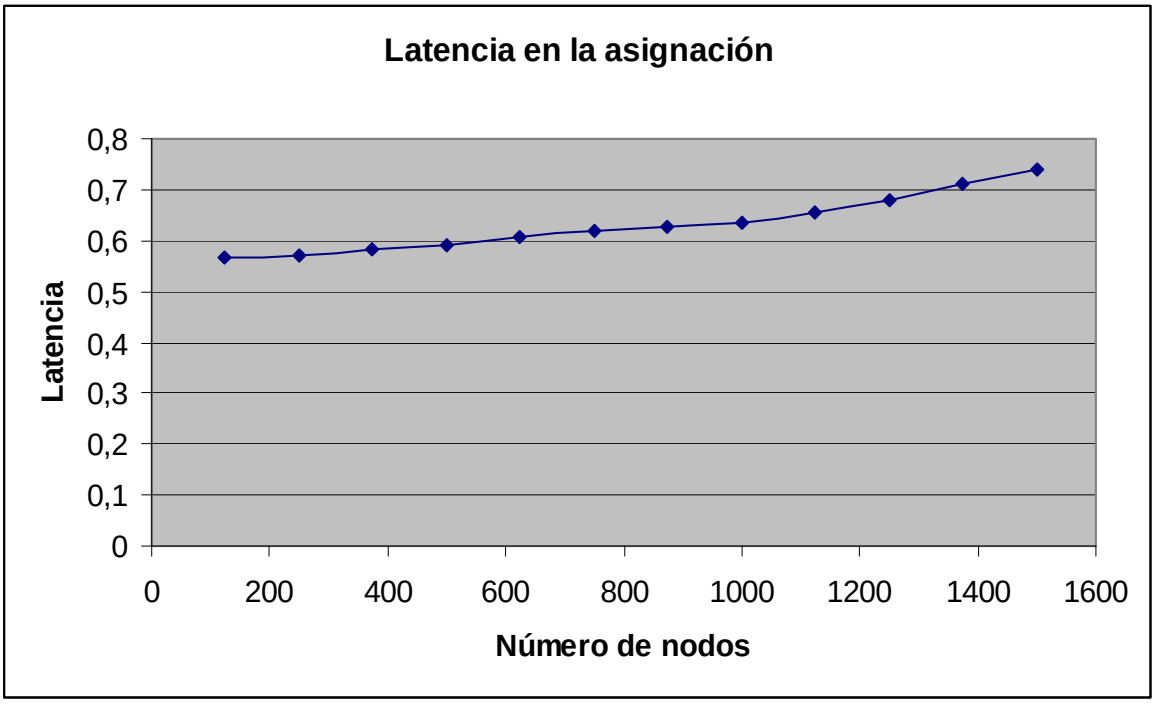

Fig. 7.2. Latencia en la asignación de direcciones IPv4 en una red clase B.

Frente a los protocolos de autoconfiguración basados en la detección de direcciones duplicadas (DAD), el protocolo D2HCP presenta además una gran reducción en la sobrecarga de paquetes de control en la red. En efecto, en la mayoría de los casos, la configuración se realizará de forma local, es decir, un vecino asignará la dirección al nuevo nodo. Esto implica el envío de cuatro paquetes de control que no se propagan al resto de la red. En el caso de que no se puedan asignar direcciones localmente se realiza una transmisión *unicast* con el servidor elegido, lo que provoca una sobrecarga mucho menor que un envío *broadcast*.

La probabilidad de que no se pueda asignar direcciones localmente depende de la relación entre el número de nodos presentes en la red y el número de direcciones disponibles.

En la Figura 7.3 se puede observar el número medio de paquetes de control involucrados en cada proceso de configuración de dirección.

En las simulaciones se han utilizado direcciones IP clase C, por lo que disponemos de 254 direcciones de red. Se puede comprobar en la Figura 7.3 que cuando existen pocos nodos en la red, el número de mensajes de control necesarios para realizar la configuración se acerca al mínimo, puesto que en la mayoría de los casos se puede realizar la configuración de forma local.

Sin embargo, cuando el número de direcciones libres se acerca a 0, no se puede realizar la configuración localmente y se debe recurrir a nodos lejanos para realizar dicha configuración, aumentando el número de mensajes enviados.

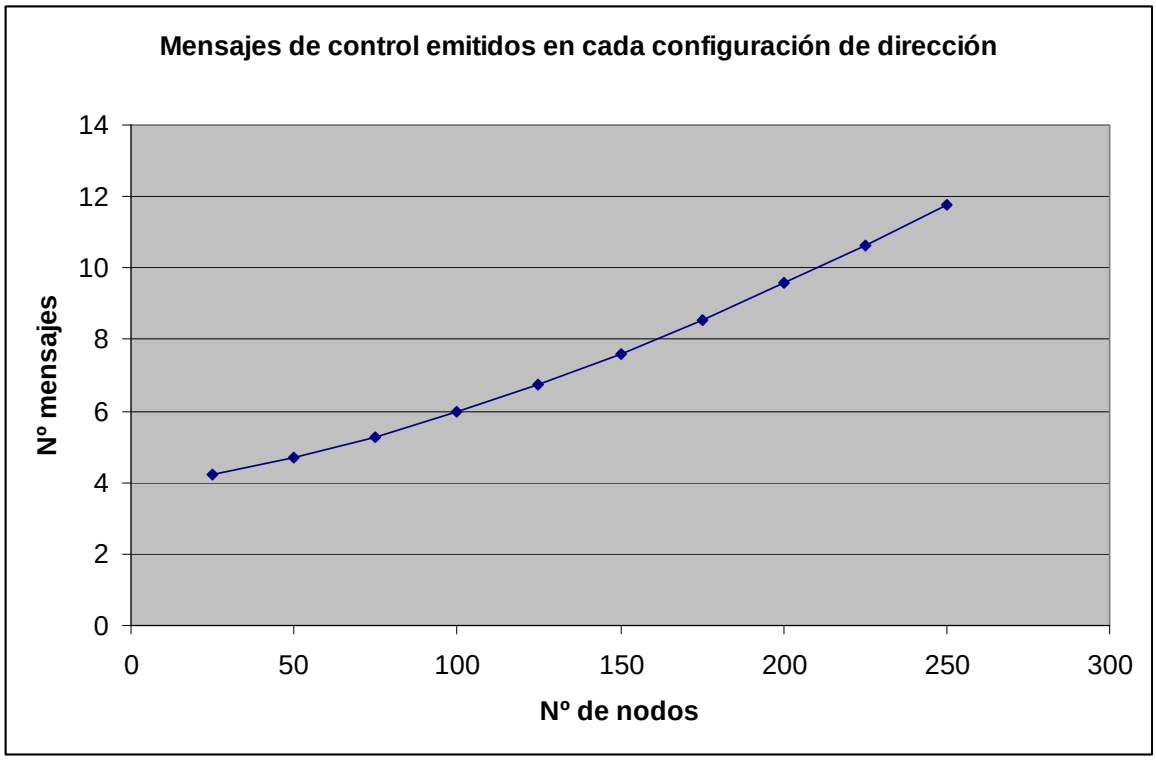

Fig. 7.3. Número medio de mensajes de control enviados en cada configuración de dirección.

En la Figura 7.4 se puede ver la evolución del número de mensajes de control necesarios para asignar una dirección en función del número de peticiones realizadas por segundo. En el caso de la línea rosa se ha realizado la simulación en un escenario con 250 nodos. En el caso de la línea azul se ha utilizado un escenario de similares características pero con 225 nodos. La frecuencia de salida de nodos de la red utilizada es similar a la frecuencia de entrada para mantener la disponibilidad de direcciones.

Como se aprecia en la Figura 7.4 en el caso de una red con 225 nodos (en torno al 90% de ocupación) el número de mensajes de control necesarios es prácticamente independiente del número de peticiones realizadas por segundo, lo que se traduce en que el protocolo soporta eficientemente la escalabilidad de la red.

Sólo en el caso límite (en torno al 100% de ocupación) el protocolo disminuye su rendimiento en términos de sobrecarga. En efecto, el principal problema de rendimiento encontrado se da en la situación que refleja la línea rosa de la Figura 7.4. En situaciones en las que se producen errores en la elección del servidor remoto, la latencia aumenta de forma directamente proporcional al número de mensajes de control enviados. Sin embargo, este incremento en la sobrecarga es aceptable, puesto que sigue siendo menor que la sobrecarga producida por los algoritmos DAD.

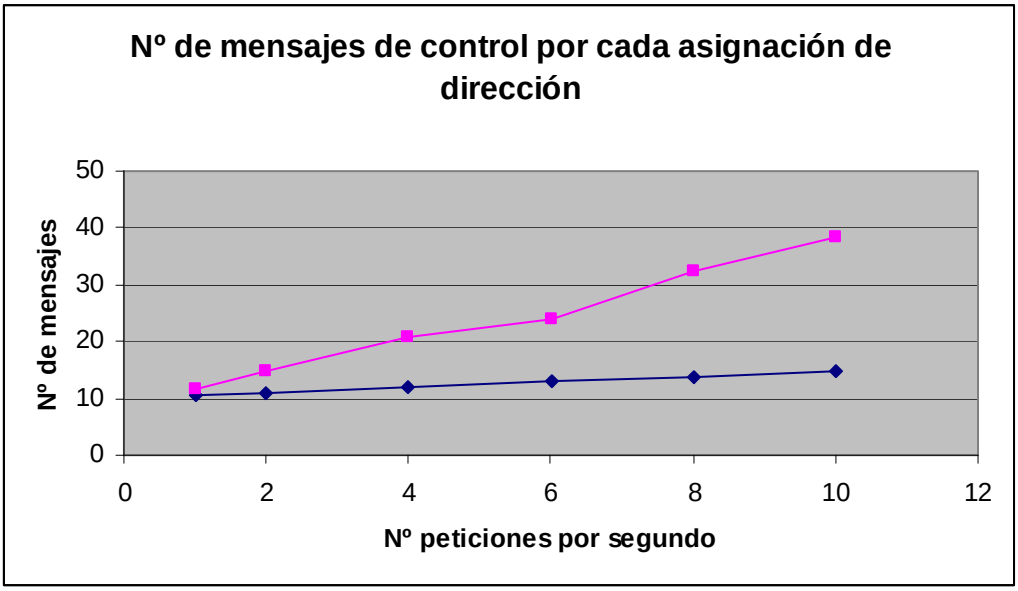

Fig. 7.4. Número de mensajes de control frente al número de peticiones por segundo.

8. CONCLUSIONES Y TRABAJO FUTURO

Se ha diseñado un protocolo de autoconfiguración para redes móviles ad hoc denominado D2HCP (*Distributed Dynamic Host Configuration Protocol*). Este protocolo se clasifica como un protocolo *stateful* o de estado completo. Se trata de un protocolo de autoconfiguración de direcciones IPv4 para redes móviles ad hoc aisladas.

En este protocolo cada nodo es responsable de la administración de un rango de direcciones. Cuando un nuevo nodo desea comenzar a participar en la red, uno de los nodos ya pertenecientes a la red, cede la mitad de su rango de direcciones al nuevo nodo. En el caso de que ningún nodo adyacente disponga de direcciones libres, pero existan direcciones libres, se realiza una petición a un nodo de la red que disponga de direcciones libres. En este modo de funcionamiento se basa el carácter distribuido del protocolo.

Para mantener una información actualizada acerca de las direcciones libres que posee cada nodo, se monitoriza el tráfico de paquetes de control del protocolo OLSR. Dicho protocolo intenta mantener en cada nodo un conocimiento actualizado de la topología completa de la red. Este protocolo se ha diseñado para trabajar de forma conjunta con OLSR. Aunque por la flexibilidad de su diseño podría operar con cualquier otro protocolo proactivo.

D2HCP garantiza unicidad en las direcciones IP bajo una amplia variedad de condiciones de red que incluyen pérdida de mensajes, peticiones concurrentes y partición de redes. Los resultados de la simulación demuestran que el protocolo tiene baja latencia y sobrecarga. Cabe destacar asimismo la escalabilidad que presenta el protocolo frente a otras propuestas existentes en la literatura, su flexibilidad que facilita la extensión del protocolo con nuevas funcionalidades,

así como su proceso de sincronización que introduce sobrecarga nula.

8.1. Trabajo Futuro

Como posibles trabajos futuros pueden señalarse los siguientes:

- Detección de la unión o fusión de redes permitiendo reasignar las direcciones que entren en conflicto (algo relativamente fácil ya que sería introducir un nuevo mensaje).

- Extensión del protocolo a redes subordinadas, con acceso a Internet o a otro tipo de redes, para lo que habría que tener en cuenta la topología de la red para realizar el proceso de autoconfiguración de direcciones.

- Estudio del rendimiento del protocolo en cooperación con otros protocolos de encaminamiento proactivos (esta primera versión D2HCP se ha diseñado para trabajar de forma conjunta con OLSR).

- Adición de un módulo de seguridad que lo proteja contra diferentes ataques para poder proporcionar una autoconfiguración segura.

REFERENCIAS

[1] Norman Abramson: "The ALOHA system-another alternative for computer communications". *Proceedings of the Fall 1970 AFIPS Computer Conference*, pp. 281-285, 1970.

[2] Ad-Hoc Network Autoconfiguration Work Group (autoconf). http://tools.ietf.org/wg/autoconf/.

[3] E. Baccelli: "Address Autoconfiguration for MANET: Terminology and Problem Statement". *Internet Draft*, February 2008. http://tools.ietf.org/html/draft-ietf-autoconf-statement-04.

[4] Elizabeth M. Belding-Royer: "Routing Approaches in Mobile Ad hoc Networks". *Mobile Ad hoc Networking*, (Stefano Basagni, Marco Conti, Silvia Giordano, Ivan Stojmenovic eds.). Wiley Inter-Science. 1st Edition. New Jersey. Chapter 10, pp. 275-300, 2004.

[5] C. Bernardos, M. Calderon, H. Moustafa: "Ad-Hoc IP Autoconfiguration Solution Space Analysis". *Internet Draft*, November 2008. http://tools.ietf.org/pdf/draft-bernardos-autoconf-solution-space-02.pdf.

[6] C. Bernardos, M. Calderon, H. Moustafa: "Survey of IP address autoconfiguration mechanisms for MANETs". *Internet Draft*, November 2008. http://tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-04.

[7] C. Bernardos, M. Calderon, H. Moustafa: "Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs". *Internet Draft*, November 2008.
http://www.it.uc3m.es/cjbc/papers/draft-bernardos-autoconf-evaluation-considerations-03.txt.

[8] I. Chakeres, Charles E. Perkins: "Dynamic MANET On-demand (DYMO) Routing". *Internet Draft*, March 2009 (Trabajo en progreso).
http://tools.ietf.org/html/draft-ietf-manet-dymo-17.

[9] Tsu-Wei Chen, Mario Gerla: "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks". *Proceedings of IEEE International Conference on Communications (ICC'98)*, June 1998.

[10] S. Cheshire, B. Aboba, E. Guttman: "Dynamic Configuration of IPv4 Link-Local Addresses". *RFC 3927*, May 2005.
http://www.ietf.org/rfc/rfc3927.

[11] T. Clausen, P. Jacquet: "Optimized Link State Routing Protocol (OLSR)". *RFC 3626*, October 2003.
http://www.ietf.org/rfc/rfc3626.

[12] M.S. Corson, A. Ephremides: "A distributed routing algorithm for mobile wireless networks". *ACM/Baltzer Wireless Networks,* Vol. 1, No. 1, pp. 61–81, February 1999.

[13] Falko Dressler: "Self-Organization in Ad Hoc Networks: Overview and Classification". *Technical Report*. Autonomic Networking Group, University of Erlangen, February 2006.

[14] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney: "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)". *RFC 3315*, July 2003. http://www.ietf.org/rfc/rfc3315.txt.

[15] L. M. Feeney: "An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks", *Mobile Networks and Application*. Vol. 6, No. 3, pp. 239 – 249, 2001.

[16] James A. Freebersyser, Barry Leiner: "A DoD perspective on mobile ad hoc networks". *Ad Hoc Networking*, (Charles E. Perkins ed.). Addison-Wesley, Chapter 2, pp. 29-51, 2001.

[17] Ricardo Puttini, Maíra Hanashiro, Fábio Miziara, Rafael de Sousa, L. Javier García Villalba, C. J. Barenco: "On the Anomaly Intrusion-Detection in Mobile Ad Hoc Network Environments". *Lecture Notes in Computer Science*, Springer-Verlag, LNCS 4217, Vol. 4217/2006, pp. 182-193, 2006.

[18] Mario Gerla, Xiaoyan Hong, Guangyu Pei: "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks". *Internet Draft*, June 2002. http://tools.ietf.org/html/draft-ietf-manet-fsr-03.

[19] Zygmunt J. Haas, Marc R. Pearlman, Prince Samar: "The Zone Routing Protocol (ZRP) for Ad Hoc Networks". *Internet Draft*, July 2002. http://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04.

[20] IEEE 802.11-1999: "IEEE Standard for Local and Metropolitan Area Networks – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". *IEEE Standard Association, The Working Group for Wireless LAN*, Edición de 1999, ratificada en Junio 2003.

[21] Sushant Jain: "Energy Aware Communication in Ad – Hoc Networks". *Technical Report UW-CSE 03-06-03,* Computer Science and Engineering, University of Washington, Seattle, January 2003. ftp://ftp.cs.washington.edu/tr/2003/06/UW-CSE-03-06-03.pdf.

[22] Geetha Jayakumar, G. Gopinath: "Ad Hoc Mobile Wireless Networks Routing Protocols – A Review". *Journal of Computer Science*, Vol. 3, No. 8, pp. 574-582, 2007.

[23] D. Jonson, Y. Hu, D. Maltz: "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4". *RFC 4728,* February 2007. http://www.ietf.org/rfc/rfc4728.

[24] Longjiang Li, Yunze Cai, Xiaoming Xu, Yonggang Li: "Agent-Based Passive Autoconfiguration for Large Scale MANETs". *Wireless Personal Communications*, Vol. 43, No. 4, pp. 1741-1749, December 2007.

[25] Young-Bae Ko, Nitin H. Vaidya: "Locationaided routing (LAR) in mobile ad hoc networks". *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networkings.* pp. 66-75, 1998.

[26] Namhoon Kim, Soyeon Ahn, Younghee Lee: "AROD: An address autoconfiguration with address reservation and optimistic duplicated address detection for mobile ad hoc networks". *Computer Communications.* Vol. 30, No. 8, pp. 1913-1925, June 2007.

[27] Joseph P. Macker, M. Scott Corson: "Mobile Ad hoc Networks (MANETs): Routing Technolgy for Dynamic, Wireless Networking". *Mobile Ad hoc Networking*, (Basagni Stefano, Conti Marco, Giordano Silvia, Stojmenovic Ivan eds.). Wiley Inter-Science. 1st Edition. New Jersey. Chapter 9, pp. 255-274, 2004.

[28] A. J. McAuley and K. Manousakis: “Self-Configuring Networks”. *Proceedings of MILCOM 2000, 21st Century Military Communications Conference*, 2000.

[29] Archan Misra, Subir Das, Anthony McAuley, Sajal K. Das: “Autoconfiguration, registration and mobility management for pervasive computing”. *IEEE Personal Communications*. Vol. 8, No. 4, pp. 24 - 31, August 2001

[30] Mobile Ad-hoc Networks Work Group (manet). http://tools.ietf.org/wg/manet/.

[31] Mansoor Mohsin, Ravi Prakash: “Ip address assingment in a mobile ad hoc network”. *Proceedings of Military Communications Conference (MILCOM)*. Vol. 2, pp. 856-861, September 2002. http://www.utdallas.edu/~ravip/papers/milcom02.pdf.

[32] J. Moy: “OSPF Version 2”. *RFC 2328*, April 1998. http://www.ietf.org/rfc/rfc2328.

[33] Shree Murthy, J. J. Garcia-Luna-Aceves: “A routing protocol for packet radio networks”. *Proceedings of First Annual ACM International Conference on Mobile Computing and Networking*. pp. 86–95, 1995.

[34] T. Narten, E. Nordmark, W. Simpson, H. Soliman: "Neighbor Discovery for IP version 6 (IPv6)". *RFC 4861,* September 2007. http://www.ietf.org/rfc/rfc4861.txt.

[35] Sanket Nesargi, Ravi Prakash: “DADHCP: Distributed Dynamic Configuration of Hosts in a Mobile Ad Hoc Network”. *Technical Report,* UTDCS-04-01, University of Texas at Dallas, Dept. of Computer Science, January 2001.

[36] Sanket Nesargi, Ravi Prakash: "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network". *Proceedings of IEEE INFOCOM 2002*. pp. 1059-1068, June 2002.
http://www.utdallas.edu/~ravip/papers/infocom2002.pdf.

[37] R. Ogier, F. Templin, M. Lewis: "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)". *RFC 3684*, February 2004.
http://www.ietf.org/rfc/rfc3684.

[38] Open Shortest Path First IGP Work Group (OSPF).
http://tools.ietf.org/wg/autoconf/.

[39] V. Park, S. Corson: "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification". *Internet Draft*, July 2001.
http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04.

[40] P. Patchipulusu: "Dynamic Address Allocation Protocols For Mobile Ad Hoc Networks". *Master's Thesis*, Texas A&M University. August 2001.

[41] Charles E. Perkins, Elizabeth M. Belding-Royer, S. Das: "Ad hoc On-Demand Distance Vector (AODV) Routing". *RFC 3561,* July 2003.
http://tools.ietf.org/html/rfc3561.

[42] Charles E. Perkins, Pravin Bhagwat: "Highly Dynamic Destination-Sequence Distance-Vector Routing (DSDV) for Mobile Computers". *Computer Communications Review*. Vol. 24, No. 4, pp. 234-244, 1994.

[43] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Elizabeth M. Belding-Royer, Yuan Sun: "IP Address Autoconfiguration for Ad Hoc Networks". *Internet Draft*, November 2001.
http://tools.ietf.org/html/draft-perkins-manet-autoconf-01.

[44] Jyoti Raju, J. J. Garcia-Luna-Aceves: "A New Approach To On-Demand Loop-Free Multipath Routing". *Proceedings of 8th Annual IEEE International Conference on Computer Communications and Networks (ICCCN)*, pp. 522–527, October 1999.

[45] F. Ros, P. Ruiz, Charles E. Perkins: "Extensible MANET Auto-configuration Protocol (EMAP)". *Internet Draft*, March 2006. http://tools.ietf.org/html/draft-ros-autoconf-emap-02.

[46] Yuan Sun, Elizabeth M. Belding-Royer: "Dynamic Address Configuration in Mobile Ad Hoc Networks," *Technical Report*, UCSB 2003-11, Department of Computer Science, University at Santa Barbara, June 2003.

[47] The Internet Engineering Task Force (IETF). http://www.ietf.org/.

[48] The ns-3 network simulator. http://www.nsnam.org/.

[49] S. Thomson, T. Narten, T. Jinmei: "IPv6 Stateless Address Autoconfiguration". *RFC 4862,* September 2007. http://www.ietf.org/rfc/rfc4862.txt.

[50] Mansi Ramakrishnan Thoppian, Ravi Prakash: "A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks". *IEEE Transactions on Mobile Computing*. Vol. 5. No. 1, pp. 4-19, January 2006. http://www2.computer.org/portal/web/csdl/doi/10.1109/TMC.2006.2.

[51] O. Troan, R. Droms: "IPv6 Prefix Options for DHCPv6". *RFC 3633*, December 2003. http://www.ietf.org/rfc/rfc3633.txt.

[52] Nitin H. Vaidya: “Weak Duplicate Address Detection in Mobile Ad Hoc Networks”. *Proceedings of ACM MobiHoc 2002*, pp. 206–216, June 2002.

[53] Kilian Weniger, Martina Zitterbart: “IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks”. *Proceedings of European Wireless 2002*, Florence, Italy, Feb 2002. http://www.iponair.de/publications/Weniger-EuroWireless02.pdf.

[54] Kilian Weniger: “PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks”. *IEEE Journal on Selected Areas in Communications (JSAC) Special Issue 'Wireless Ad hoc Networks'*. Vol. 23, No. 3, pp. 507-519, March 2005. http://www.tm.uka.de/doc/2004/autoconf_jsac_epub.pdf.

[55] Kilian Weniger: “Passive Duplicate Address Detection in Mobile Ad Hoc Networks”. *Procedings of IEEE WCNC 2003*, March 2003. http://www.tm.uka.de/doc/2003/passive_dad_lsr_wcnc03.pdf.

[56] Hongbo Zhou, Lionel M. Ni, Matt W. Mutka: “Prophet Address Allocation for Large Scale MANETs”. *Proceedings of IEEE INFOCOM 2003*, March 2003.

ANEXO A: SIMULADOR DE REDES NS-3

El simulador NS-3 (Network Simulator 3 [46]) está diseñado dentro del proyecto NSNAM (Network Simulator Network ANimator) siendo un simulador de redes de eventos discretos.

Este simulador está escrito en el lenguaje C++, y se ofrece bajo la versión 2 de la GNU (General Public License).

Las principales características de este simulador de redes son:

- Fácilmente extensible
- Código abierto y libre para el estudio o modificaciones
- Multiplataforma
- Desarrollo en comunidad

Protocolos y estructuras implementadas

NS-3 tiene implementadas diferentes capas del modelo TCP/IP, y en ellas se encuentran diferentes protocolos y estructuras de datos contempladas en los protocolos de redes actuales.

Al ser un simulador que se desarrolla en comunidad y con código abierto, las implementaciones existentes han podido aumentar gracias a desarrolladores de todo el mundo.

La Figura A.1. muestra una tabla que resume las estructuras y protocolos ya

implementados oficialmente tanto en NS-3, como en su antecesor NS-2:

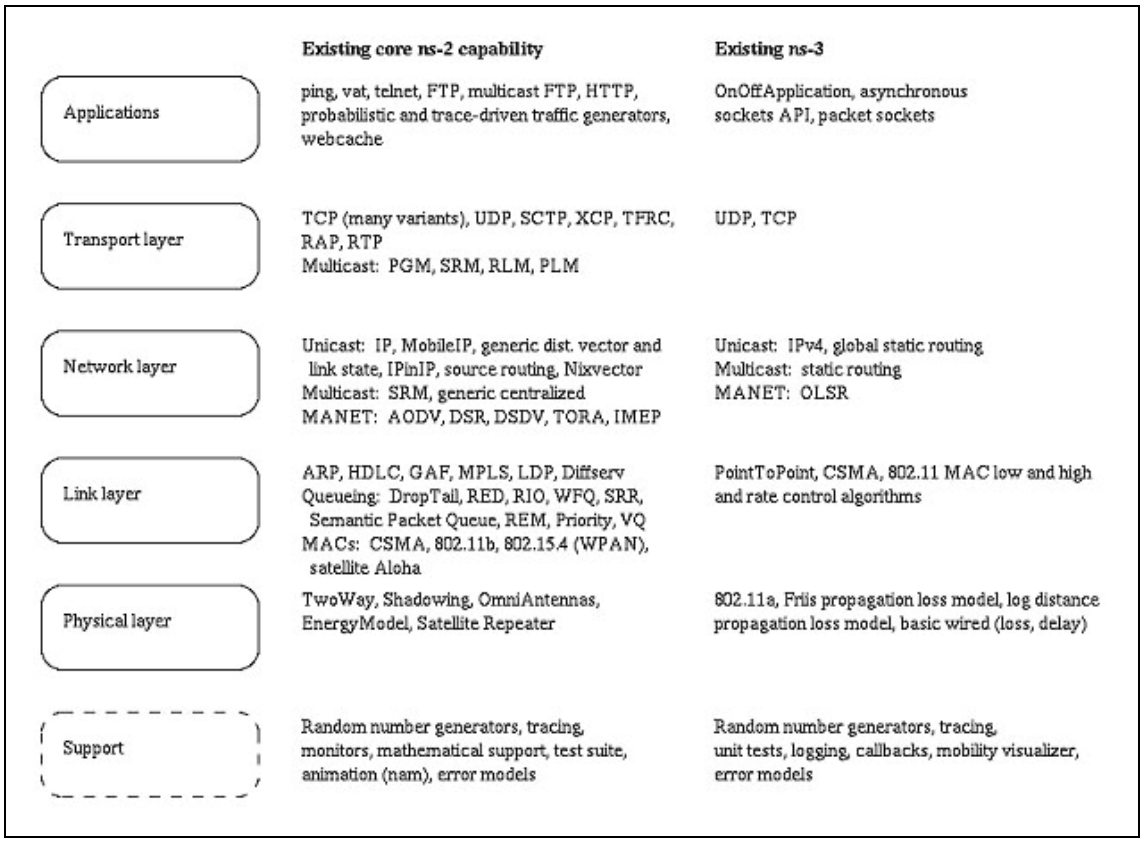

Fig. A.1. Tabla resumen de estructuras y protocolos implementados en NS-2 y NS-3.

Escenarios e implementación

Para la generación de escenarios se implementan Scripts escritos en C++ y Python. Estos escenarios cuentan con determinados objetos, llamados entidades (entities) que son utilizados de forma análoga a objetos del mundo real, como son sockets, aplicaciones o canales.

Las entidades mínimas que se deben programar en los escenarios son:

- *Nodos*: forman la red, se conectan a través de los canales de transmisión.

- *Aplicaciones*: simulan el envío y recepción de datos en los nodos.

- *Paquetes*: objetos que se envían, por ejemplo, un datagrama UDP.

- *Protocolos*: se encuentran entre la aplicación y el interfaz de red. Simulan conexiones, acceso a un medio compartido, direccionamiento, encaminamiento, etc.

- *Interfaces de red*: es la interfaz entre el protocolo y el canal de transmisión, de esta manera se implementan Ethernet, CSMA (Carrier Sense Multiple Access), PPP (Point to Point Protocol), etc.

- *Canales*: Modelan el medio de transmisión, introduciendo determinados parámetros como retardos, tasa de transmisión, etc. Estos canales unen los nodos.

Diagrama de las capas del protocolo TCP/IP

La Figura A.2. muestra el tránsito de información a través de las diferentes capas del modelo TCP/IP representado con las entidades del simulador NS-3.

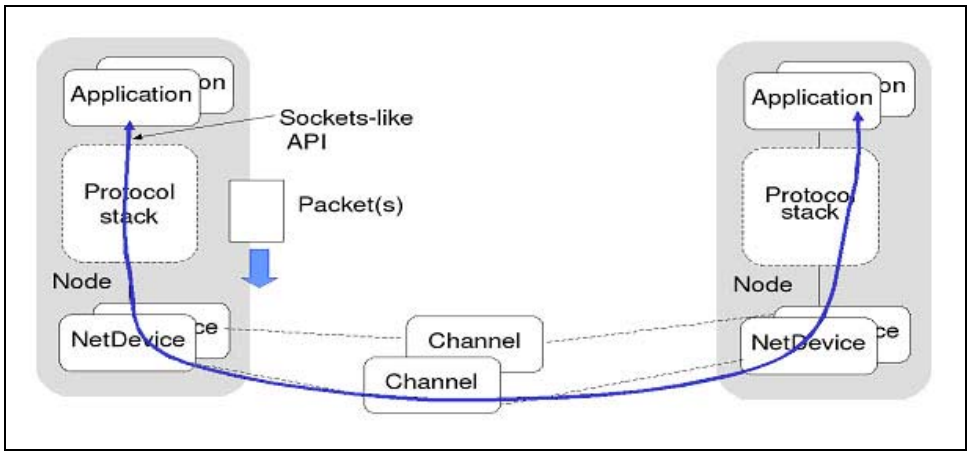

Fig. A.2 Tránsito de información entre capas TCP/IP de NS-3.

Ejemplo de un escenario

El siguiente Script es un ejemplo que viene dado por los desarrolladores para la comprensión del sistema de entidades. Este Script llamado simple.cc muestra cómo crear nodos, y asignarle diferentes configuraciones.

Primero crea un nodo al que simula una conexión a Internet y un intercambio de mensajes, para después mostrar el tráfico procesado por el socket simulado:

- En primera instancia se debe crear un NodeContainer que será el contenedor que tendrá los nodos de la simulación, en este caso creamos 1 nodo.

- El siguiente paso es la asignación de los diferentes Helpers. Los Helpers sirven para la creación de los diferentes protocolos, y estructuras de datos dentro de las redes de comunicación. De esta manera, los Helpers pueden considerarse como un grupo de clases que implementan las diferentes configuraciones aplicables a un nodo, a un contenedor o a un dispositivo. Por ejemplo, en este caso se le dota al nodo con el protocolo necesario para simular Internet, mediante el InternetStackHelper. Análogamente se pueden asignar un Helper de encaminamiento como OLSR-helper a los diferentes nodos, o un Helper que implementa las direcciones IP versión 4 (ipv4-address-helper).

- Por último queda lanzar la ejecución con los parámetros deseados.

Código del ejemplo:

```
Void RunSimulation (void)
{
    NodeContainer c;
    c.Create (1);
    InternetStackHelper internet;
    internet.Install (c);
    TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
    Ptr<Socket> sink = Socket::CreateSocket (c.Get (0), tid);
    InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (),
        80);
    sink->Bind (local);
    Ptr<Socket> source = Socket::CreateSocket (c.Get (0), tid);
    InetSocketAddress remote = InetSocketAddress (Ipv4Address::
        GetLoopback (), 80);
    source->Connect (remote);
    GenerateTraffic (source, 500);
    PrintTraffic (sink);
    Simulator::Run ();
    Simulator::Destroy ();
}
```

Estructura de clases

El siguiente esquema resume el sistema de módulos disponibles en el simulador NS-3. Esta estructura de clases facilita la tarea del programador, ya que es fácil encontrar los elementos deseados para una simulación, así como las configuraciones requeridas para la estructura de la red recreada. Estos son los principales módulos:

- *Simulator*: se centran las clases involucradas en la simulación en sí, como son Time y Scheduler. Objetos de estas clases son necesarias para cualquier simulación en un escenario del simulador NS-3, ya que se usan para simular todos los eventos que se suceden en una simulación, como la entrada o salida de nodos de una red.

- *Node*: encontramos todas las clases usadas para la configuración de nodos, como la dirección IP, el canal de transmisión, los diferentes dispositivos de red o sockets.

- *Devices*: modelan diferentes accesos a un medio de transmisión, como un puente (bridge), el modelado del protocolo punto a punto (PPP), el protocolo CSMA o el medio WIFI.

- *InternetStack*: se centra en la implementación de los protocolos necesarios para Internet, como son TCP, UDP o ARP.

- *Helpers*: se engloban todos los Helpers creados hasta el momento por el equipo NSNAM. Encontramos Helpers destinados a aplicar diferentes configuraciones a los dispositivos de la red como InternetStackHelper; pero también encontramos otros que son útiles para crear un conjunto de objetos con una propiedad en común como CsmaHelper (el cual crea dispositivos capacitados con el protocolo CSMA).

- *Applications*: en este módulo existen un conjunto de clases que simulan el comportamiento de la capa de aplicación del modelo TCP/IP, como son los mensajes UDP Echo.

- *Mobility*: se describen una serie de modelados de movilidad, tales como la posición en ejes cartesianos, la velocidad o la dirección que van tomando diferentes nodos.

- *Routing*: corresponde con los protocolos de encaminamiento implementados en este simulador. Hasta el momento están implementados OLSR y el encaminamiento global.