Universidad Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

DESCUBRIMIENTO DE SERVICIOS CROSS-LAYER BASADO EN OLSR PARA REDES MANET

Autora:

Mª Isabel Vara Lorenzo

Directora:

Mª Celeste Campo Vázquez

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Leganés, noviembre 2015

Universidad Carlos III de Madrid
www.uc3m.es

(a entregar en la Oficina de Posgrado, una vez nombrado el Tribunal evaluador , para preparar el documento para la defensa de la tesis)

TESIS DOCTORAL

DESCUBRIMIENTO DE SERVICIOS CROSS-LAYER BASADO EN OLSR PARA REDES MANET

Autora: *Mª Isabel Vara Lorenzo*

Directora: **Mª Celeste Campo Vázquez**

Firma del Tribunal Calificador:

Firma

Presidente: (Nombre y apellidos)

Vocal: (Nombre y apellidos)

Secretario: (Nombre y apellidos)

Calificación:

Leganés, de de

A mis padres, Rosalía e Ignacio,
a mi hermana Lucía, y a mi hija Leire.

El esfuerzo solo proporciona plenamente su recompensa, después de que una persona se niega a darse por vencida.
Napoleon Hill

Agradecimientos

Está claro que querer es poder. Fueron las palabras de Celeste, mi directora de tesis, cuando después de mucho esfuerzo publicamos el resultado de nuestro trabajo. Tengo mucho que agradecerle. Se lo tengo que agradecer todo. Tengo claro que no hubiera llegado hasta aquí sin ella. Esta tesis ha sido fruto del trabajo conjunto que hemos realizado. Esta tesis ha sido mérito de las dos. Gracias, Celeste. Por lo fácil que ha sido trabajar contigo, por tu incuestionable profesionalidad, por tu dedicación, por todas las noches sin dormir, por todas las horas que has empleado en esta tesis, por tus ánimos, por el conocimiento que me has transmitido y también, por creer en mi y no haberme dejado sola todos estos años. Y sobre todo, gracias por haberme ayudado a cumplir un sueño. Un sueño por la ilusión que tenía puesta en esta tesis y el convencimiento, a la vez, de que no podría lograrlo.

Quiero agradecer a mi hija, que aunque más veces de las que ella hubiera querido y menos de las que yo necesitaba, ha entendido que tenía que trabajar para ser Doctora. Gracias por haberme arrancado una sonrisa en mis momentos tristes. Y gracias por todos los *Te quiero* y los abrazos que me das, que me han dado fuerzas para llegar hasta aquí. Gracias, cariño, porque sin ti, no soy nada.

Gracias a las personas que han estado cerca de mí, a mi hermana, que me han visto sufrir, llorar este último año, aunque también de alegría. No ha sido un año fácil, y tampoco os lo he puesto fácil. Gracias por estar ahí, por compartir mi día a día, por tener paciencia, por escucharme. Gracias por estar pendientes de mí y por toda la ayuda y ánimos que me habéis transmitido para terminar esta tesis.

Gracias también a mis padres que durante tantos años creyeron en mí y me transmitieron su ilusión para que nunca abandonara esta aventura en la que me embarqué hace más de diez años, y que tan difícil ha sido compaginar con una carrera profesional en el sector privado y una maternidad que me hecho descubrir lo grande que puede llegar a ser el amor. Gracias también a ellos, hoy estoy escribiendo este capítulo de *Agradecimientos* de esta Tesis Doctoral que me llena tanto de orgullo.

Por último, quiero agradecer a mis compañeros de trabajo su gran generosidad durante este último año, sus ánimos, su paciencia y la ayuda que me han ofrecido. En especial a Eduardo, por todo, porque sin ti, todo hubiera sido más difícil, porque contigo, todo ha sido más fácil.

Eskerrik asko,
María Isabel Vara Lorenzo
Madrid, noviembre 2015

Resumen

El auge que en los últimos años ha tenido el uso de dispositivos móviles, su integración plena en la vida de las personas, así como el desarrollo expansivo de las redes inalámbricas, y en especial de las redes MANET (*Mobile Ad hoc Network*), hace que actualmente sea difícil imaginar un mundo sin dispositivos inteligentes personales que nos acompañen a todas partes como, por ejemplo, los *smartphones* y *wearables*.

Las redes MANET [Ahvar et al, 2007], [Tyagi et al, 2010] están compuestas por nodos móviles autónomos que se unen voluntariamente formando una red entre ellos. Son redes en las que no existe una infraestructura de red fija y la administración se realiza de forma descentralizada. Esto permite que se cree una red prácticamente de la nada, sin necesidad de intervención humana ni configuraciones previas. Los nodos que integran la red participan en la toma de decisiones, tienen su propio conjunto de protocolos de encaminamiento, en el que toman parte de forma activa y tienen mecanismos de gestión de red y procesos de intercambio de información propios. La disponibilidad de estos nodos es generalmente corta. Son nodos que entran y salen de la red sin previo aviso, con lo que la topología de la red está continuamente cambiando de forma dinámica y aleatoria. La mayoría de estos nodos son dispositivos con limitado poder de procesamiento, limitada capacidad de memoria y baja capacidad de almacenamiento de energía.

Que las redes MANET sean capaces de soportar descubrimiento de servicios se antoja indispensable. Debido a la movilidad de estas redes, en cualquier momento puede cambiar la topología y los servicios y recursos que se ofrecen en la red. Los dispositivos deben poder descubrir de forma automática los servicios que están disponibles en la red, así como los nodos que proporcionan estos servicios.

En esta tesis doctoral proponemos un nuevo mecanismo de descubrimiento de servicios basado en OLSR (*Optimized Link State Routing Protocol*) [Clausen et al, 2003] para redes MANET, que permite que de forma automática un dispositivo o nodo descubra los servicios ofrecidos por otros nodos que lo rodean. SD-OLSR (*Service Discovery over OLSR*) es un mecanismo que permite

anunciar servicios con una sobrecarga introducida en la red pequeña, una tasa de descubrimiento de servicios alta, una tasa baja de falsos descubrimientos y un tiempo en descubrir servicios mínimo.

Inicialmente definimos el mecanismo de descubrimiento de servicios sobre la primera versión del protocolo OLSR (OLSRv1). Este mecanismo ofrecía la posibilidad de que un nodo anunciara y preguntara por servicios disponibles en la red. Dadas las características proactivas del protocolo de encaminamiento OLSR vimos que no era necesario inundar la red con mensajes preguntando por servicios que los nodos ya anunciaban periódicamente. Limitar los mensajes sólo al anuncio de servicios hizo que las prestaciones del protocolo mejoraran.

En abril de 2014 se estandariza la segunda versión de OLSR, OLSRv2 [Clausen et al, 2014]. Una de las líneas que manteníamos abierta era integrar y optimizar el mecanismo propuesto para OLSRv1, sobre OLSRv2, aprovechando la versatilidad y mejoras realizadas en la segunda versión del protocolo de encaminamiento.

Los resultados que hemos obtenido, nos hacen pensar que SD-OLSRv2 puede ser un serio candidato para descubrir servicios en redes MANET con un número de nodos grande. Cuanto más grande es la red más posibilidades existen de que se pierdan paquetes y en consecuencia de que los mensajes de petición de servicio que envíen los nodos, no obtengan la respuesta con el servicio solicitado. Sin embargo, los nodos en SD-OLSRv2 están continuamente anunciando servicios, con lo que las consecuencias de las posibles pérdidas de paquetes son menores. De hecho, SD-OLSRv2 ofrece una alta tasa de descubrimiento de servicios, superior al 90 %, incluso cuando los nodos se mueven a velocidades altas. La tasa de falsos descubrimientos también es pequeña, menor del 0.5 %. Además, el tiempo que tarda un nodo en descubrir un servicio es casi instantáneo, del orden de decenas de milisegundos. Y todos los nodos tienen un conocimiento global de los servicios que se ofrecen en la red al mismo tiempo.

Abstract

The growth that the use of mobile devices has experienced, its full integration in people's lives, as well as the expansive development of wireless networks, and especially MANET (*Mobile Ad Hoc Network*) networks, makes it difficult to think of a world without intelligent personal devices coming with us everywhere, as for example smartphones and wearables. MANET networks [Ahvar et al, 2007], [Tyagi et al, 2010] consist of a collection of wireless autonomous mobile nodes that join voluntarily creating a network between them. They are networks with neither fixed infrastructure requirements nor centralized management for their operation. The network nodes work together with minimal central control and human intervention. All of the nodes of these networks behave as routers and take part in discovery and maintenance of routes to other nodes in the network. They also have their own network management mechanisms and information exchanging processes. The availability of these nodes is generally short. These nodes are free to move, to join or to leave the network at any time. They move randomly and organize themselves on a random basis. This high mobility of nodes causes the network topology to change in a random and dynamic way. Most of these nodes are devices with a limited processing power, a limited memory capacity and a low capacity of energy storage.

MANET networks ability to support service discovery is completely indispensable. Due to the high mobility of nodes, the topology and the services and resources offered in the network may vary. Devices must be able to discover automatically the services that are available on the network as well as the nodes that provide these services.

In this doctoral thesis we propose a new service discovery mechanism based on OLSR (*Optimized Link State Routing Protocol*) [Clausen et al, 2003] for MANET networks which allows any device or node to automatically discover the services offered by other nodes surrounding it. SD-OLSR (*Service Discovery over OLSR*) is a mechanism that allows to announce services with a small network overhead, a high rate of services discovery, a low rate of false discoveries and a minimum time lapse to discover services.

We first define the service discovery mechanism integrated into the first version of the OLSR (OLSRv1) protocol. This mechanism provided the possibility that a node could announce and ask for available services on the network. Given the proactive features of the OLSR proactive protocol, we saw that it was not necessary to flood up the network with messages asking for services the nodes already announced periodically. Limiting these messages only to the announcement of services made the protocol's performance improve.

In April 2014 the second version of OLSR is standardized, OLSRv2 [Clausen et al, 2014]. One of the lines we kept open was integrating and optimizing the mechanism integrated into OLSRv1 on OLSRv2, taking advantage of the versatility and improvements achieved in second version of OLSR routing protocol.

The results we have achieved make us believe that SD-OLSRv2 may be a serious candidate to discover services in MANET networks with big amount of nodes. The bigger the network, the bigger the chances of losing packages and, consequently, of failing to obtain the service response to the service requested by the nodes. However, the nodes in SD-OLSRv2 are continuously announcing services, therefore the consequences of the possible losses of packages are small. As a matter of fact, SD-OLSRv2 offers a high rate of service discovery, over 90 %, even when the nodes are moving at a high speed. The rate of false discoveries is also small, under 0,5 %. Besides, a node discovers a given service almost instantaneously, it takes it some tens of milliseconds. And every node has the global knowledge of the services being offered on the network at the same time.

Índice general

Índice de cuadros

Índice de figuras

Parte I

Introducción

La parte I de esta tesis introduce el concepto de redes MANET y la necesidad de definir protocolos de descubrimiento de servicios específicos sobre ellas. En el capítulo 1, listamos los problemas derivados de aplicar un protocolo de descubrimiento de servicios tradicional, diseñado para redes fijas sobre redes MANET, y detallamos los objetivos que nos hemos planteado de cara al trabajo realizado en esta tesis.

Capítulo 1

Introducción

Si nos fijamos a nuestro alrededor, la visión futurista que Mark Weiser describió en su artículo *"Computer for the 21st Century"* [Weiser, 1991] está siendo una realidad. Weiser describe entornos saturados de elementos con capacidades de cómputo y comunicación, totalmente integrados en nuestras vidas y que nos proporcionan información asociada a nuestras necesidades y al entorno en el que nos encontramos en cada momento. Estos elementos con capacidades de cómputo y comunicación son capaces de conocerse entre ellos y colaborar entre sí para intentar resolver los problemas del usuario. El usuario no se debe preocupar de configurar los elementos ni de saber cómo conectarlos entre sí, simplemente debe utilizar el potencial de computación que le ofrecen. Son los propios dispositivos los que tienen que saber, qué otros dispositivos existen a su alrededor y qué servicios pueden proporcionar cada uno de ellos. Esta nueva era de la computación imaginada por Weiser es conocida como computación ubicua. La computación ubicua busca, entre otros objetivos, que el usuario no perciba la tecnología que le rodea pero sí que sienta sus beneficios.

En la actualidad, gracias fundamentalmente a los avances realizados en la microelectrónica y también a la evolución de los protocolos inalámbricos, existe un gran número de dispositivos con capacidad de cómputo y comunicación que forman parte de nuestra vida: sensores electrónicos de reducido tamaño, bajo coste y bajo consumo, capaces de capturar y procesar información del entorno o del propio usuario, por ejemplo para monitorización de incendios, gases, contaminación... ; actuadores que ejecutan remotamente acciones físicas como abrir puertas, accionar interruptores; electrodomésticos del hogar como lavadoras y frigoríficos inteligentes e incluso dispositivos en entornos vehiculares como coches y autobuses, capaces de comunicarse entre ellos e intercambiar información sin necesidad de ninguna infraestructura previa. Todos estos dispositivos además dis-

ponen de capacidad para comunicarse con otros dispositivos de forma autónoma e inalámbrica, sin necesidad de ningún elemento intermedio.

La historia de las redes inalámbricas comienza en los años 70 con proyectos como *ALOHA* [Abramson, 1970], que tenía por objetivo conectar las instituciones educativas de las islas Hawai. Más adelante, en 1973 la *Defense Advanced Research Projects Agency* (DARPA) comenzó a trabajar en un nuevo proyecto *Packet Radio Networks* (PRNETs)[Jubin et al, 1987] que pretendía que las unidades militares pudieran comunicarse a través de radio con total movilidad y de modo cooperativo, de tal forma que cada nodo podía ser, en cualquier momento, un nodo terminal o un nodo conmutador de paquetes.

El rápido avance de la tecnología radio en los años 70 provocó la aparición de múltiples sistemas de comunicación móvil como teléfonos inalámbricos, sistemas de radio búsqueda, satélites móviles, etc. Posteriormente, en 1983, DARPA desarrolló el proyecto *Survivable Radio Networks* (SURAN)[Schacham et al, 1987] que trataba 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. También se dedicaron esfuerzos a escalar las redes a miles de nodos y dar soporte para ataques a la seguridad. El resultado fue la aparición de la tecnología conocida como *Low-Cost Packet Radio* (LPR)[Fifer et al, 1987] en 1987.

A mitad de los años 90 con la llegada de las tarjetas de radio 802.11 para ordenadores personales y portátiles, se produce un nuevo avance en la evolución de las redes inalámbricas. [Freebersyser, 2001] y [Jain, 2003] proponen por primera vez la idea de una colección de nodos móviles con una infraestructura mínima, y el *Institute of Electrical and Electronics Engineers* (IEEE) acuña el término redes *Ad hoc*.

Las redes *Ad hoc* ya no son exclusivas del ámbito militar como lo fueron en sus orígenes las redes inalámbricas. Gracias al éxito de las comunicaciones inalámbricas y al incremento de los dispositivos portátiles con capacidad de computación y de conectividad a redes, las redes *Ad hoc* han suscitado el interés de la comunidad académica y científica. Como punto de partida en la investigación de las redes *Ad hoc* se puede establecer el año 1995 cuando en una conferencia del *Internet Engineering Task Force* (IETF) se discutía sobre las redes inalámbricas por satélite, las redes gubernamentales y las redes autoconfigurables. La idea principal era la de adaptar los protocolos existentes en Internet a estas redes inalámbricas. Posteriormente en 1997, el IETF formó el grupo de trabajo *MANET Working Group*

(MANET-WG)[1] especialmente dedicado al estudio de las redes *Ad hoc*, con el objetivo de estandarizar los aspectos y protocolos más importantes referentes a este tipo de redes y también, con el objetivo de que este tipo de redes pudieran ser utilizadas no sólo en el ámbito científico y académico sino también en el ámbito comercial.

1.1. Redes móviles *Ad hoc*

Las redes inalámbricas se pueden clasificar en:

- **Redes con infraestructura**, donde los nodos se comunican siempre con una estación base fija mientras se mueven.
- **Redes sin infraestructura**, donde los nodos no necesitan una estación base fija para comunicarse mientras se mueven. En este caso, para que la comunicación sea posible todos los nodos que integran la red deben ser capaces de actuar como routers.

Las *Redes Móviles Ad hoc* (MANET) son redes inalámbricas sin infraestructura, que se crean de la nada, sin ningún tipo de planificación previa y respondiendo a la distribución que los nodos tienen en cada momento. Son redes totalmente descentralizadas, que carecen completamente de ningún tipo de elemento central o punto de acceso fijo, sobre el que recaiga la administración de la red.

Se caracterizan principalmente por la movilidad de los nodos que integran la red y la comunicación salto a salto entre ellos. Los nodos que no pueden comunicarse directamente, utilizan nodos intermedios para propagar la información. Estos nodos disponen, por lo tanto, de su propio conjunto de protocolos de encaminamiento en el que toman parte de forma activa. La disponibilidad de estos nodos es generalmente corta, la mayoría son dispositivos con limitado poder de procesamiento, limitada capacidad de memoria y baja capacidad de almacenamiento de energía. Debido a su movilidad, hacen que la topología de la red cambie de forma dinámica y aleatoria.

A continuación se resumen las características principales de este tipo de redes, que tendrán que tenerse en cuenta en la definición de los protocolos que se desarrollen para ellas:

- *Ausencia de infraestructura.* Las redes MANET las forman nodos autónomos. No existe ningún tipo de entidad central que se encargue de la gestión

[1] https://tools.ietf.org/wg/manet/

y del control de la red. Los nodos pueden desempeñar el papel de *host* o de *router* en cualquier momento.

- *Topología dinámica.* Al no existir ninguna infraestructura fija y dadas las características móviles de la red MANET, la topología de la misma puede cambiar de forma dinámica y aleatoria. Los nodos se pueden mover arbitrariamente creando y destruyendo enlaces.

- *Ancho de banda limitado.* Los enlaces inalámbricos se caracterizan por tener un ancho de banda reducido y ser más propensos a errores que los enlaces de una red cableada.

- *Variación en la capacidad de los enlaces y los nodos.* Los nodos pueden tener varias interfaces con diferentes capacidades de transmisión y que operan en diferentes frecuencias.

- *Consumo de energía.* La red la integran nodos móviles que funcionan alimentados por baterías de vida limitada. Es muy importante, por lo tanto, que en una red MANET se reduzca el consumo de energía al máximo.

- *Escalabilidad.* La red MANET se crea de la nada y la pueden llegar a formar miles de nodos, lo que conlleva retrasos y dificultad en tareas como direccionamiento, encaminamiento, seguridad, etc.

- *Falta de seguridad.* Dada la vulnerabilidad inherente a los enlaces inalámbricos, la seguridad es una limitación de estas redes.

- *Encaminamiento multisalto.* Son los propios nodos que integran la red los que actúan de *routers* en el caso de que no exista comunicación directa entre el nodo origen y el nodo destino.

- *Entorno imprevisible.* Las redes MANET están pensadas para operar en ambientes hostiles, campos de batalla, zonas catastróficas o incendios, donde la principal necesidad radica, en crear rápidamente un entorno de comunicación efectivo entre los diferentes elementos involucrados. Es posible que en estos escenarios los nodos puedan destruirse o comiencen a producir fallos.

1.2. Motivación de la tesis

Dentro de las MANET se abordan principalmente cinco grandes retos:

- *Encaminamiento.* Dado que la topología en este tipo de redes se mantiene en constante cambio, el envío de paquetes entre nodos se convierte en todo un reto.

- *Seguridad.* Además de las vulnerabilidades inherentes a las redes inalámbricas, las redes MANET tienen otros problemas de seguridad. Por ejemplo, un nodo sin autorización puede entrar en la red sin ser detectado. No existe ninguna entidad central que valide usuarios. Este nodo puede simplemente escuchar el tráfico, o enviar paquetes falsos, pero también podría participar en el encaminamiento de paquetes lo que implicaría que la red dejara de funcionar de forma correcta.

- *Calidad de servicio.* Otro de los aspectos importantes es la garantía de calidad de servicio que la red debe ofrecer a los usuarios que hacen uso de ella. La calidad de servicio en una red MANET es mucho más crítica que en las redes tradicionales, ya que es mucho más difícil garantizar límites de retardo entre redes, *jitter* o probabilidad de pérdida de paquetes, cuando se opera en un entorno altamente dinámico.

- *Consumo de energía.* Dado que los nodos que participan en una red MANET basan su funcionamiento en el uso de baterías, es necesario que se mejoren los procesos de comunicación y procesamiento de datos para optimizar el consumo de energía, haciendo que la vida útil de estos nodos se alargue lo máximo posible.

- *Descubrimiento de servicios.* Como ya hemos visto, las redes *Ad hoc* se encuentran intrínsecamente ligadas con el concepto de computación ubicua y por tanto, el descubrimiento de servicios es un componente necesario para su correcto funcionamiento. El auge que los últimos años ha tenido la *Internet de las cosas* (Internet of Things, IoT) [Atzori et al, 2010] ha contribuido también a que cada vez sean más necesarias redes que ofrezcan servicios que se auto-configuren solos y que se auto-descubran solos. Los nodos o servicios de la IoT deben ser dinámicamente descubiertos y utilizados por otros otros nodos o servicios de una forma sencilla y transparente. Para ello es necesario diseñar protocolos de descubrimiento de servicios específicos que permitan que los nodos descubran servicios dinámicamente.

Tras el estudio realizado de la situación actual de la literatura científica respecto al descubrimiento de servicios, se ha observado que es un tema que ha suscitado un gran interés, tanto en su aplicación a redes con infraestructura como a redes MANET. Durante los últimos años se han definido diversos protocolos que abordan el problema descubrimiento de servicios para ambos tipos de redes:

[Ververidis et al, 2008] y [Mian et al, 2009]. Sin embargo no todos los protocolos aquí descritos emplean los mecanismos más adecuados para funcionar en redes MANET.

Muchas de estas soluciones tienen contribuciones interesantes pero haciendo un análisis podemos concluir que algunas no logran resolver las complejidades de los entornos móviles, sobre todo en redes con un gran número de nodos. También, la mayoría, sitúa el descubrimiento de servicios a nivel de aplicación, lo que implica que por debajo de ese mecanismo de descubrimiento de servicios debe existir un mecanismo de descubrimiento de rutas para acceder al servicio descubierto. Esto, generalmente, trae consigo un incremento en la sobrecarga de paquetes que se transmiten por la red.

La principal motivación para investigar el descubrimiento de servicios en redes MANET es que, dado el carácter dinámico de estas redes, donde la topología de la red cambia de forma arbitraria e impredecible debido a la movilidad de los nodos, es muy difícil definir un protocolo de descubrimiento de servicios con una tasa de descubrimiento de servicios alta y que a la vez, la sobrecarga que se introduzca en la red debido al mecanismo de descubrimiento sea pequeña. Para que la tasa de descubrimiento de servicios sea lo más alta posible y un nodo tenga una visión de los servicios que se ofrecen en la red lo más real posible, los nodos cliente deberían estar continuamente preguntando por los servicios disponibles en la red y/o los nodos servidores anunciando continuamente sus servicios. Sin embargo, inundar continuamente la red de mensajes de descubrimiento y/o anuncio de servicios supone aumentar considerablemente la sobrecarga que se introduce en la misma. Llegar a un equilibrio entre tasa de descubrimiento de servicios y sobrecarga introducida en la red es todo un reto, sobre todo en redes con un número de nodos elevado.

1.3. Problema

Históricamente los protocolos de descubrimiento de servicios diseñados para redes con infraestructura se basan en el uso de un directorio central en el que los nodos de la red registran sus servicios. Este tipo de soluciones no son idóneas para redes MANET donde, debido a la movilidad de los nodos es difícil confiar en un nodo central en el que registrar los servicios. Además de que resulta difícil que en una red MANET donde los dispositivos tienen capacidades limitadas de memoria y baja capacidad de almacenamiento de energía, un nodo pueda asumir el rol de servidor central.

El descubrimiento de servicios en redes MANET requiere un sistema descen-

tralizado y que cada nodo sea independiente del resto para anunciar o registrar sus servicios.

Estas soluciones distribuidas, que eliminan el repositorio centralizado de servicios, emplean dos mecanismos básicos para que los nodos que integran la red descubran los servicios que se ofrecen en la misma:

- **Método *push***: en el que los servicios se anuncian periódicamente y los nodos clientes escuchan y almacenan estos anuncios para seleccionar posteriormente el servicio que les interesa.

- **Método *pull***: en el que los nodos cliente realizan peticiones y los servicios se descubren bajo demanda.

Un factor importante a tener en cuenta en estas soluciones en las que, o bien los anuncios de servicios, o bien las peticiones o bien ambos se envían a todos los nodos que integran la red, es que puede suponer un incremento elevado del ancho de banda que se consume.

Además, debido al carácter dinámico de una red MANET, los servicios estarán disponibles un tiempo limitado, por lo que es necesario implementar mecanismos de descubrimiento de servicios que permitan detectar lo antes posible tanto la disponibilidad como indisponibilidad de esos servicios. Todo ello sin sobrecargar en exceso la red.

Durante los últimos años se han desarrollado mecanismos de descubrimiento de servicios específicos para redes MANET, definidos a nivel de aplicación y también soluciones que integran el descubrimiento de servicios en los protocolos de encaminamiento de la red MANET. Término que se conoce como descubrimiento de servicios *cross-layer*. En el primer caso, el descubrimiento de servicios es independiente del protocolo de encaminamiento que se utilice en la red. En el segundo caso, el envío de mensajes de descubrimiento de servicios forma parte del protocolo de encaminamiento, con el objetivo de reducir el número de mensajes enviados a través de la red, y permitir además, conocer al mismo tiempo los cambios producidos en la topología de la red y los servicios que se ofrecen en la misma.

En el capítulo 2 realizaremos un estudio más detallado del problema de descubrimiento de servicios y justificaremos la necesidad de definir un nuevo mecanismo de descubrimiento de servicios adaptado a las características de las redes MANET.

1.4. Objetivos

Se han establecido los siguientes objetivos:

- *Objetivo 1.* Analizar y estudiar las diferentes tecnologías existentes para el descubrimiento de servicios en redes MANET.

  Una vez analizadas las diferentes propuestas definidas hasta la fecha, nos planteamos otro objetivo:

  - *Objetivo 1.1.* Dada la movilidad de los nodos que integran este tipo de redes, que hace que las rutas cambien durante el tiempo de vida de la red MANET, es necesario definir un mecanismo que permita integrar los procesos de creación y mantenimiento de rutas con el de descubrimiento de servicios para así, aumentar las prestaciones del mecanismo de descubrimiento de servicios. Esto nos lleva a cumplir otro objetivo más:
  - *Objetivo 1.2.* Hay que adaptarse a las modificaciones y/o mejoras que el protocolo de encaminamiento ofrezca en el tiempo.

- *Objetivo 2.* Definir un mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento proactivo OLSR para redes MANET, con un número de nodos elevado mediante el cual, los nodos servidores den a conocer los servicios que ofrecen a todos los nodos que integran la red.

  Los objetivos que nos fijamos de cara a la validación del mecanismo propuesto son:

  - *Objetivo 2.2.* Desarrollar la solución o soluciones planteadas para el descubrimiento y anuncio de servicios.
  - *Objetivo 2.3.* Evaluar la solución planteada mediante un simulador de red y realizar posibles modificaciones sobre la propuesta inicial para mejorar el mecanismo de descubrimiento de servicios.
  - *Objetivo 2.1.* Encontrar el mayor porcentaje de servicios que se ofrecen en la red, con la menor tasa de falsos descubrimientos y el mínimo tiempo en descubrir un servicio, dadas las características de movilidad de los nodos que integran la red MANET.

- *Objetivo 3.* Documentar el trabajo realizado y difundirlo en foros y revistas de investigación.

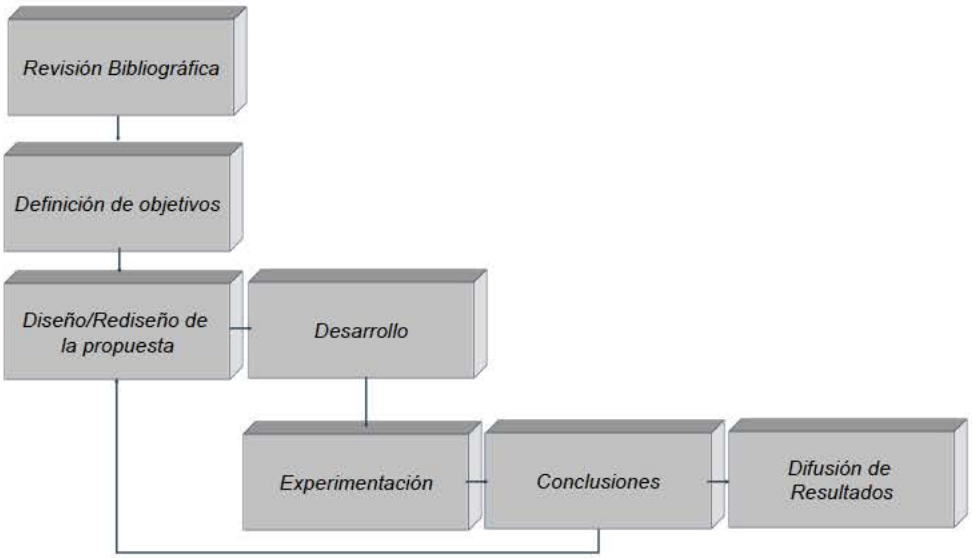

Figura 1.1: Metodología de diseño

1.5. Plan de trabajo

Con objeto de conseguir los objetivos marcados en la sección anterior se ha establecido una metodología de trabajo que se representa de forma gráfica en la figura 1.1.

- Revisión bibliográfica sobre el descubrimiento de servicios centrada en las redes MANET. En esta fase se ha requerido también una revisión de los principales protocolos de encaminamiento que existen para redes MANET.

- La revisión bibliográfica ha dado lugar a una clasificación de las soluciones existentes referentes al descubrimiento de servicios y a un estudio exhaustivo del funcionamiento de las versiones 1 y 2 del protocolo de encaminamiento OLSR de cara a integrar el mecanismo de descubrimiento de servicios sobre él.

- Definición de los requisitos de la solución.

- Diseño y desarrollo de la solución propuesta.

- Evaluación de la solución propuesta.

- Rediseño y mejora del mecanismo de descubrimiento de servicios propuesto y adaptación a las nuevas mejoras definidas en la versión 2 del protocolo de encaminamiento OLSR.
- Elaboración de las conclusiones y difusión de los resultados.

1.6. Estructura de la memoria

La memoria de esta tesis se estructura como sigue:

En el capítulo 2, se analiza el problema del descubrimiento de servicios de forma teórica y se realiza un estudio detallado de los mecanismos de descubrimiento de servicios existentes en la literatura tanto en redes con infraestructura fija como en redes MANET. Los protocolos de descubrimiento de servicios para redes MANET los clasificamos en protocolos de descubrimiento de servicios basados en directorio y protocolos de descubrimiento de servicios que no utilizan directorios para descubrir y anunciar servicios. Dentro de estos últimos se encuentran los protocolos de descubrimiento de servicio *cross-layer*.

El capítulo 3, presenta una clasificación de los protocolos de encaminamiento y describe las principales diferencias entre ellos. Se explican en detalle las dos versiones del protocolo OLSR, OLSRv1 [Clausen et al, 2003] y OLSRv2 [Clausen et al, 2014] que son en los que se basa esta tesis.

En el capítulo 4, se proporciona un diseño razonado de nuestra propuesta para el descubrimiento de servicios integrado en la versión 1 del protocolo de encaminamiento OLSR. También se realiza una descripción detallada de cómo se ha implementado el mecanismo de descubrimiento de servicios en el simulador de red NS-2. Por último, se analizan las prestaciones del mecanismo propuesto respecto a los parámetros: sobrecarga introducida en la red y tasa de descubrimiento de servicios.

En el capítulo 5, se proporciona un diseño razonado de nuestra propuesta para el descubrimiento de servicios integrado en la versión 2 del protocolo de encaminamiento OLSR. También se realiza una descripción detallada de cómo se ha implementado el mecanismo de descubrimiento de servicios en el simulador de red NS-2. Por último, se analizan las prestaciones del mecanismo propuesto respecto a los parámetros: tiempo en descubrir un servicio, tasa de descubrimiento de servicios, tasa de descubrimientos falsos y sobrecarga introducida en la red. También se presenta una comparativa del mecanismo de descubrimiento de servicios propuesto con el protocolo de descubrimiento de servicios SD-AODV.

Finalmente, en el capítulo 6 se resumen las principales conclusiones del trabajo

realizado y se enumeran una serie de posibles líneas de continuación del mismo.

Como parte de la memoria hemos incluido una serie de apéndices para aclarar y completar algunas de las contribuciones de esta tesis doctoral. Así:

En el apéndice A, se describe la metodología de simulación empleada para realizar el diseño y el estudio de prestaciones del mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSR.

En el apéndice B, se describe cómo introducir nuevas instrucciones Tcl para simular en el simulador de red NS-2[Simulador NS-2] los escenarios que nos permitan validar las prestaciones de los mecanismos de descubrimiento de servicios SD-OLSRv1 y SD-OLSRv2.

1.7. Historia de la tesis

En esta sección, realizamos un breve resumen de la historia de esta tesis, y enumeramos las publicaciones realizadas durante estos últimos años y en las que se reflejan parte de sus resultados.

El objetivo con el que se inició esta tesis doctoral fue diseñar un mecanismo de descubrimiento de servicios para redes MANET con OLSR como protocolo de encaminamiento. Redes MANET con OLSR como protocolo de encaminamiento porque eran sobre las que en su momento menos se había investigado y menos soluciones había a la hora de implementar un mecanismo de descubrimiento de servicios eficaz. Inicialmente se diseñó un mecanismo de descubrimiento de servicios integrado en la primera versión de OLSR, OLSRv1. El formato de paquete del protocolo OLSRv1, permitía fácilmente añadir nuevas funcionalidades al protocolo, definiendo nuevos tipos de mensajes que iban encapsulados en el campo MESSAGE del paquete OLSR. Para permitir el descubrimiento de servicios, se definió un nuevo tipo de mensaje, *Service Discovery Message* (SDM) que permitía a un dispositivo tanto descubrir servicios que ofrecen otros nodos, como anunciar los servicios que un nodo posee. En [Vara et al, 2006] abordábamos nuestro primer estudio del estado del arte de descubrimiento de servicios en redes MANET y definíamos la primera versión del mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSRv1. En [Jodra et al, 2006] describíamos el comportamiento del protocolo en redes de diferentes tamaños, con los nodos servidores inundando periódicamente la red con mensajes de anuncio de servicios y los nodos clientes enviando peticiones solicitando un servicio cada vez que lo necesitaban.

Posteriormente, comenzamos a estudiar la posibilidad de que el mecanismo

inicialmente definido tipo *push/pull* funcionara únicamente como un mecanismo tipo *push* y contemplara sólo el anuncio periódico de servicios, aprovechando las características proactivas del protocolo de encaminamiento OLSR. Los propios nodos servidores eran los que de forma activa anunciaban periódicamente los servicios que disponían. Los nodos clientes almacenaban esos servicios de forma local y cuando querían acceder a un servicio de un tipo determinado, comprobaban si era uno de los servicios anunciados previamente en la red. En caso de no tener ese servicio almacenado localmente, esperaban a que se produjera un anuncio de ese tipo en la red.

Con esta modificación en el protocolo vimos que se mejoraban notablemente las prestaciones del mismo en cuanto a sobrecarga introducida en la red, y que la tasa de servicios descubiertos no empeoraba respecto a la versión del mecanismo de descubrimiento de servicios propuesto inicialmente.

Este hecho coincide en el tiempo con la evolución del protocolo OLSR de la versión 1 a la versión 2. Aprovechando los buenos resultados obtenidos con SD-OLSRv1, nuestro objetivo fue trasladar los conocimientos obtenidos a OLSRv2, mejorando ciertos aspectos del mecanismo definido para OLSRv1. Durante el tiempo que tardó la versión 2 de OLSR en estandarizarse (abril del 2014), trabajamos en la versión actual del mecanismo de descubrimiento de servicios. Este nuevo trabajo se recoge en la publicación [Vara et al, 2015].

Los primeros *drafts* del protocolo OLSRv2 aparecen en el año 2005. En el año 2006 aparece el *draft* del protocolo *MANET Neighborhood Discovery Protocol* (NHDP) [Clausen et al, 2011], protocolo genérico para el descubrimiento de vecinos, y también se especifica un formato genérico para el envío de los paquetes de control para cualquier protocolo definido para redes *Ad hoc*, *Generalized MANET Packet/Message Format* [Clausen et al, 2015].

En la figura 1.2 se muestra gráficamente la evolución del protocolo OLSRv2.

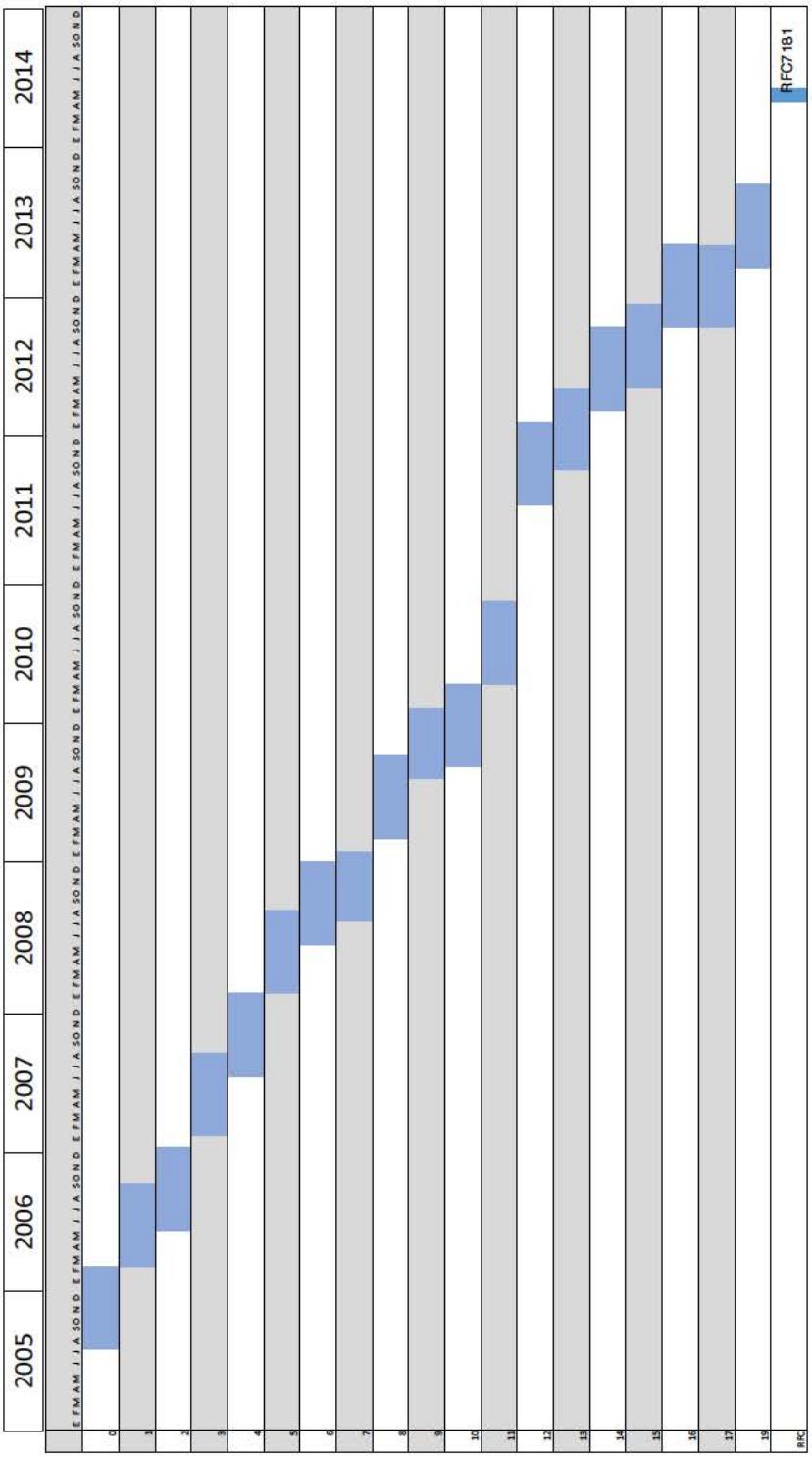

Figura 1.2: Proceso de estandarización del protocolo de encaminamiento OLSRv2

Parte II

Estado del arte

En la parte II realizamos un estudio preciso de los protocolos de descubrimiento de servicios diseñados hasta la fecha en el ámbito de las redes MANET así como una clasificación de los principales protocolos de encaminamiento definidos para este tipo de redes.

El capítulo 2 se centra en el problema del descubrimiento de servicios. Definimos lo que es un servicio y realizamos un estudio teórico de los mecanismos de descubrimiento de servicios que posteriormente pasamos a detallar. Hacemos una breve introducción de los protocolos de descubrimiento de servicios diseñados para redes fijas y nos centramos más en los protocolos diseñados para entornos MANET.

En el capítulo 3 introducimos el concepto de protocolos de encaminamiento en redes MANET y presentamos la clasificación más común que se hace de ellos, centrándonos en describir de una forma precisa el protocolo de encaminamiento proactivo OLSR.

Capítulo 2

Estudio de mecanismos de descubrimiento de servicios

El concepto de descubrimiento de servicios no es algo nuevo, existen varias propuestas al respecto tanto para redes fijas como para redes MANET, [Ververidis et al, 2008] y [Mian et al, 2009]. El objetivo principal con el que surgieron estos protocolos fue facilitar a un dispositivo móvil el descubrimiento, configuración y uso de los servicios que se ofrecían en la nueva red a la que se conectaba. En general, las diferentes propuestas abordan el problema de distintas formas, desde las que han sido diseñadas para su uso con un determinado protocolo de red, hasta las que están asociadas a un determinado lenguaje de programación.

En este capítulo realizamos un estudio detallado del problema del descubrimiento de servicios y analizamos las soluciones más relevantes que se han proporcionado estos últimos años. Así, en la sección 2.1, abordamos el estudio del concepto de servicio, viendo las diferentes definiciones que se han dado y relacionándolas con el problema del descubrimiento. En la sección 2.2, realizamos un análisis teórico del problema del descubrimiento de servicios. En la sección 2.3, enumeramos los principales protocolos de descubrimiento de servicios diseñados hasta la fecha para redes con infraestructura. En las secciones 2.4 y 2.5, nos centramos en el estudio de los protocolos de descubrimiento de servicios más relevantes propuestos hasta la fecha para redes MANET. Finalizamos, sección 2.6, con las conclusiones del estudio realizado y que nos han llevado a la definición de un nuevo mecanismo de descubrimiento de servicios para redes MANET.

2.1. Definición y descripción de servicios

Los protocolos de descubrimiento de servicios permiten localizar al servidor que ofrece un determinado servicio, pero ¿qué es un servicio?. Varias han sido las respuestas que se han dado en la literatura en los últimos años, citamos a continuación las más relevantes [Campo C., 2004]:

"...service can be defined as any hardware or software entity that resides on any mobile device or platform and has distinct functional attributes that characterizes it"[Chakraborty, 2001]

"...a particular logical function that may be invoked via some network protocol, such as printing, storing a file on a remote disk, or even perhaps requesting delivery of a pizza"[Willians, 2002]

"...we defined such "services" as applications with well-known interfaces that perform computation or actions on behalf of client users"[Czerwinski et al, 1999]

"...a service is an entity that can be used by a person, a program, or another service. A service may be a computation, a storage, a communication channel to another user, a software filter, a hardware device, or another user [...] A service has an interface which defines the operations than can be requested of that service"[Jini, 1999]

En general, podemos decir que existen dos alternativas a la hora de abordar la definición del concepto de servicio: los que lo definen desde el punto de vista de una plataforma de computación distribuida y los que lo definen desde el punto de vista de protocolos de comunicación. En el primer caso se habla de interfaces, en el segundo de protocolos. En ambos casos, un servicio es una entidad que realiza una determinada función de la que un usuario o aplicación puede beneficiarse, la diferencia está en la forma en la que se accede a él y en la forma en la que se describe.

Consideremos por ejemplo una impresora, el usuario puede necesitar en algún momento solicitar la impresión de un documento que está editando en su *smartphone*. Si hablamos del servicio impresión a nivel de protocolo de aplicación, esto se traducirá en localizar un elemento en la red que soporte el protocolo `lpr` o `ipp`(*internet printING protocol*); si lo hacemos desde el punto de vista de interfaces necesitamos un elemento que soporte una determinada interfaz, por ejemplo

`Printer`. Una vez descubierto el servicio, en el primer caso el *smartphone* necesitará conocer el protocolo `lpr` o `ipp`, la dirección IP y el número de puerto que le permita comunicarse con la impresora y enviarle el documento; en el segundo caso, el acceso al servicio consistirá en invocar métodos de la interfaz `Printer` sobre un determinado objeto, cuya localización y acceso remoto aisla la plataforma de computación distribuida con la que se esté trabajando.

La descripción de un servicio debe proporcionar información necesaria para poder acceder a él y las características particulares que nos permitan diferenciar ese servicio de otros similares. En la mayoría de los formatos de descripción de servicios, esta información se proporciona a través de lo que se denominan atributos del servicio. Los tipos de atributos que tiene un servicio dependen del servicio que describen, para permitir la interoperabilidad se deben definir plantillas de descripción de servicios, en las que se indican los diferentes atributos y los valores que pueden tomar.

Un atributo común en la descripción de servicios, es el tipo de servicio. Un tipo engloba a un conjunto de servicios con características comunes. En la mayoría de formatos de descripción de servicios, se emplean jerarquías de tipos de servicios que van desde servicios más abstractos a más concretos. El tipo de servicio es el atributo que se emplea para realizar búsquedas, el servidor comprueba que ofrece un servicio del tipo solicitado antes de responder.

En nuestra propuesta de mecanismo de descubrimiento de servicios, como descripción de servicios utilizaremos un valor constante para todos los anuncios de servicios. Se trata de una descripción de servicios muy simplificada basada en el formato definido en el protocolo SLP, pero que para el objetivo de esta tesis doctoral, que es la validar el mecanismo de descubrimiento de servicios en entornos MANET en cuanto a tasa de descubrimiento de servicios y tiempo en descubrir un servicio, el campo descripción del servicio no tiene un impacto importante.

2.2. Estudio teórico de mecanismos de descubrimiento de servicios

En esta sección realizamos un estudio sobre las diversas alternativas teóricas que existen para el descubrimiento de servicios de forma dinámica. Estas alternativas no se consideran protocolos de descubrimiento de servicios como tales, aunque son la base sobre las que se han definido los protocolos que han aparecido estos últimos años.

2.2.1. Planteamiento del problema

En todas las redes de comunicación existen un conjunto de nodos, denominados servidores, que ofrecen determinados servicios que pueden ser utilizados por otros nodos conectados a la red, denominados clientes.

La manera tradicional en la que un nodo cliente descubre y accede a los servicios que ofrecen los nodos servidores, es mediante configuraciones realizadas por los administradores de la red a la que están conectados los nodos clientes. Cuando el número de servicios es muy elevado y además las redes son muy dinámicas: los nodos son móviles y por lo tanto, cambian de red frecuentemente; es necesario que estas configuraciones se realicen de forma automática, sin necesidad de intervenciones manuales continuas.

Para abordar este objetivo es necesario solucionar tecnológicamente dos problemas:

1. Descubrimiento de servicios, es decir, permitir que los nodos cliente sepan qué servicios están disponibles.

2. Acceso a los servicios, es decir, permitir que los clientes, una vez descubiertos los servicios que necesitan en la red, hagan uso de ellos.

En los siguientes apartados estudiamos las diferentes alternativas que se plantean para solucionar el problema del descubrimiento de servicios de forma dinámica y analizamos las ventajas e incovenientes de cada una de ellas, dependiendo del escenario de aplicación.

2.2.2. Posibles soluciones

Solución I: funcionamiento distribuido tipo *pull*

La primera solución que se plantea es que los nodos cliente cuando necesiten acceder a un servicio de un tipo determinado, envíen peticiones solicitándolo, a las que responden los nodos servidores que ofrecen dicho servicio. Las peticiones de servicio se envían por *multicast* o *broadcast* mientras que las respuestas de los servidores se envían por unicast al cliente que lo ha solicitado. A este modo de funcionamiento se le denomina distribuido de tipo *pull*.

Esta forma de realizar el descubrimiento de servicios es fácil de implementar tanto en los clientes como en los servidores, pero tiene problemas de escalabilidad

en redes con un gran número de nodos debido al consumo excesivo de ancho de banda.

Una manera de reducir este problema es introducir cachés de servicios remotos en los clientes, de tal forma que un cliente almacena de forma local los servicios que va descubriendo. Cuando este cliente quiera acceder de nuevo a un tipo determinado de servicio, antes de enviar un nuevo mensaje de petición a la red, comprueba si lo tiene almacenado en su caché y si es así, no envía la petición.

De esta forma se minimiza el número de mensajes transmitidos pero se introduce un nuevo problema: ¿el servicio almacenado en la caché sigue estando disponible en la red?. Para reducir esta incertidumbre, los servidores anuncian sus servicios con un tiempo de vida asociado, que indica cuál es el tiempo máximo que un cliente puede tener almacenado ese servicio en la caché.

Aunque el mecanismo anterior reduce la probabilidad de que un servicio almacenado en la caché no esté disponible, no la elimina. Existen algunas técnicas que se pueden emplear para reducir más este problema:

- Sondeo: el cliente de forma periódica comprueba si los servicios almacenados en su caché están disponibles o han sufrido alguna modificación, poniéndose en contacto con el servidor que los ofrece.

- Suscripción y notificación: los clientes pueden suscribirse en los servidores, para que se les notifique los cambios o caídas de los servicios almacenados en su caché, cuando éstos se producen.

Si los cambios o caídas de los servidores son muy frecuentes la introducción de estas técnicas tiene un consumo excesivo de ancho de banda por lo que en entornos muy dinámicos, no debe hacerse uso de ellas.

Solución II: funcionamiento distribuido tipo *push*

La segunda solución que se plantea es que sean los propios nodos servidores los que de forma activa anuncien periódicamente los servicios que ofrecen. Los nodos cliente almacenan estos anuncios de forma local y cuando quieren acceder a un servicio de un tipo determinado, comprueban si es uno de los servicios anunciados previamente en la red o esperan a que se produzca un anuncio de ese tipo. Estos anuncios se envían por difusión. A este modo de funcionamiento se le denomina distribuido de tipo *push*.

Esta forma de realizar el descubrimiento de servicios introduce mayor complejidad en los clientes y en los servidores, ya que los clientes deben almacenar todos los anuncios que escuchan en la red y los servidores periódicamente deben anunciar los servicios que tienen disponibles, aún cuando ningún cliente precise ese servicio.

Como los clientes almacenan de forma local los servicios que se anuncian en la red, se nos plantea el mismo problema que al emplear cachés en el modo *pull*: ¿el servicio estará disponible cuando el cliente quiera acceder a él?. En el modo *push* los servidores envían los anuncios según una periodicidad que comunican al cliente, de tal forma, que un cliente cuando no recibe un anuncio en el intervalo previsto, elimina este servicio de su caché. En este caso el tiempo de vida de un servicio en la caché de los clientes, es el periodo de tiempo entre anuncios consecutivos del servidor que lo ofrece.

Entre anuncios consecutivos de un servidor se pueden producir caídas del mismo o modificaciones, que el cliente no detectará hasta el próximo anuncio. Por lo tanto, parece recomendable que los servidores anuncien con una frecuencia elevada sus servicios, pero de esta forma se consume un gran ancho de banda con el consecuente problema de escalabilidad del protocolo. Si se quiere reducir este consumo, minimizando la probabilidad de que un cliente no detecte cambios en el servicio, se pueden introducir técnicas de sondeo y de suscripción y notificación como se ha explicado en la solución tipo *pull*.

2.3. Descubrimiento de servicios

Una vez visto el problema del descubrimiento de servicios a nivel teórico, en las siguientes secciones vamos a realizar una clasificación de las soluciones más relevantes existentes en la actualidad para el descubrimiento de servicios en redes MANET. Pero antes, vamos a ver brevemente las soluciones que fueron diseñadas específicamente para redes con una infraestructura fija.

2.3.1. Redes con infraestructura

Estas propuestas no son aplicables a las redes MANET ya que no fueron diseñadas teniendo en cuenta las características fundamentales de este tipo de redes que ya hemos visto en la sección 1.1: la movilidad de los nodos, la ausencia de infraestructrua, topología dinámica, ancho de banda limitado, escalabilidad, encaminamiento multisalto, etc. Sin embargo, sí que definieron aspectos básicos del proceso de descubrimiento de servicios que se han utilizado posteriormente para

la definición de propuestas de descubrimiento de servicios para redes MANET.

En [Richard et al, 2000], [Bettstetter, 2000], [Willians, 2002] y [Dabrowski et al, 2005] podemos encontrar una comparación de los principales protocolos tradicionales de descubrimiento de servicios. Entre ellos destacan el protocolo SLP (*Service Location Protocol*) [Veizades et al, 1997], [Guttman, 1999], [Guttman et al, 1999], el protocolo SSDP (*Simple Service Discovery Protocol*) [Goland et al., 1999], UPnP (*Universal Plug-and-Play*) [UpnP, 2015], Jini, actualmente conocido como *Apache River* [Apache Software Foundation, 2013], Salutation [Miller and Pascoe, 2000] y SDP [SDP, 2001].

Vamos a describir un poco más en detalle el protocolo SLP, puesto que es en el que nos hemos basado, a la hora de definir el formato de descripción de servicios propuesto en el mecanismo de descubrimiento de servicios basado en OLSR para redes MANET que hemos diseñado. Además es un buen ejemplo de protocolo de descubrimiento de servicios funcional, lo suficientemente simple y completo, que nos da una idea de las tareas que hay que definir y especificar a la hora de diseñar un protocolo de descubrimiento de servicios.

SLP es uno de los protocolos que más éxito ha tenido en los últimos años. Se trata de un protocolo de descubrimiento de servicios propuesto por el IETF que se definió por primera vez en [Guttman, 1999] y su segunda versión [Guttman et al, 1999] contempla también aspectos como seguridad, extensiones de atributos, etc. Los clientes en SLP se identifican como *User Agentes* (UAs) y lo servicios se anuncian como *Service Agents* (SAs). SLP ofrece la posibilidad en redes grandes de utilizar *Directory Agents* (DAs) que son los responsables de almacenar información sobre los servicios que se están anunciando en la red. Cuando SLP funciona con DAs, los SAs registran en los DAs los servicios que ofrecen y los UAs envían peticiones de servicio *unicast* a los DAs con los servicios que buscan. Cuando SLP funciona sin DAs, los UAs envían peticiones de servicio por *multicast* a toda la red y los SAs que ofrecen el servicio responden por *unicast*. La mayoría de los mensajes que se envían son paquetes de texto que se envían al puerto 427 de UDP (*User Datagram Protocol*). La búsqueda de servicios se puede hacer por tipo o por búsqueda de atributos específicos. Cada servicio es definido por una URL (*Uniform Resource Locator*) donde viene codificado el tipo de servicio que está definido en la RFC 2609 (*Service Templates and Service Schemes*) [Guttman et al, 1999] . Vamos a ver algunos ejemplos de tipos de servicios y de URLs asociadas a servicios:

```
"weather.nasa:wtp"
"service:weather.nasa:wtp://weather.nasa.com:12000"
```

que siguen generalmente la estructura:

```
"servicio:nombre_del_servicio.nombre_proveedor:protocolo://ordenador:puerto"
```

Las descripciones de los servicios se realizan mediante plantillas definidas en la RFC 2609 para la definición de servicios. El núcleo principal de esas plantillas es una lista de identificadores de atributos y valores que contemplan las características principales de un determinado servicio. En cualquier búsqueda se especifica una lista de atributos que el servicio debe satisfacer con lo que elabora una respuesta donde incluye su URL. Un ejemplo de descripción de un servicio de impresión sería:

```
service:printer://arco.inf-cr.uclm.es:1030/queue3
scopes = planta, efca, administrator
printer-name = Despacho
printer-model = HP 2600
printer-location = tercera planta
color-supported = true
sides-supported = one-sided
```

En esta descripción se incluye la primera línea que indica el tipo de servicio y su localización (en este ejemplo el servicio es de tipo *printer* y se localiza en la dirección de la máquina *arco*) para, a continuación, listar una serie de atributos con su valor asociado entre los cuales se incluyen el *modelo*, el *nombre asociado*, la *localización*, etc.

2.3.2. Clasificación de protocolos de descubrimiento de servicios

El problema del descubrimiento de servicios en redes MANET ha sido objeto de numerosos trabajos de investigación que han tratado de adecuar las características de las soluciones tradicionales definidas para una red con infraestructura fija a este tipo de redes.

La clasificación que vamos a presentar en cuanto a protocolos de descubrimiento de servicios los divide en aquellos que hacen uso de directorios de servicio donde se centraliza y almacena toda la información de los servicios que se ofrecen en la red, y protocolos que no hacen uso de directorios de servicio y donde la información referente a los servicios que se ofrecen en la red se distribuye entre todos los nodos que integran la misma. Dentro de estos últimos vamos a ver los protocolos de descubrimiento de servicios integrados en un protocolo de encaminamiento de la red MANET, término que se conoce como descubrimiento de servicios *cross-layer*.

2.4. Protocolos de descubrimiento de servicios basado en directorios

Tal como hemos visto en la sección 2.3.1 los protocolos de descubrimiento de servicios tradicionales para redes con infraestructura hacen uso de directorios donde los nodos que ofrecen los servicios registran sus servicios, y los nodos clientes van a buscar los servicios que necesitan.

Sin embargo, estos mecanismos tal cual no se pueden aplicar a protocolos de descubrimiento de servicios diseñados específicamente para redes MANET: los directorios serían nodos de la propia red MANET, que pueden cambiar de posición, las rutas hacia esos directorios también pueden variar dada la movilidad de los nodos, hasta el punto de que los directorios puedan ser inaccesibles. Es necesario por lo tanto, encontrar mecanismos que seleccionen de forma dinámica y teniendo en cuenta los cambios en la topología de la red, los nodos más adecuados para funcionar como directorios de servicios. Además, debido a la naturaleza distribuida de la red MANET, estos directorios se suelen comunicar entre sí con el objetivo de intercambiar información de los servicios y así que toda la red MANET tenga un conocimiento global de los servicios que se ofrecen.

En los últimos años han surgido propuestas que estudian y han dado solución al problema de la selección de directorios en redes MANET. Soluciones que también tienen en cuenta el hecho de que los nodos seleccionados como directorios pueden fallar. [Raychoudhury et al, 2011] resuelven este problema mediante la elección incremental de nuevos directorios. [Cui et al, 2011] demuestran que aunque no se produzca el fallo total del directorio es posible migrar el contenido almacenado en ese nodo a otro nodo de la red que disponga de mejores recursos computacionales para mantener la información.

Existen otras soluciones de descubrimiento de servicios que proponen un *cluster* de nodos proveedores de servicio que ejercen de directorios de servicio y responden a las peticiones de solicitud de servicio enviadas por los nodos cliente. El *clustering* consiste en un proceso de jerarquización en el que los nodos de la red MANET se auto organizan en grupos llamados *clusters* con sus consecuentes beneficios para la red, como reducir el tráfico global de la red ya que el número de nodos de un cluster es menor que el número de nodos de toda la red, optimizar el proceso de encaminamiento y ayudar en el manejo de la topología, ya que cada nodo sólo requiere almacenar una fracción del total de la información de encaminamiento. Algunos ejemplos de este tipo de soluciones los encontramos en [Buvana et al, 2015] que proponen un modelo para reducir la energía consumida en la red y mejorar la escalabilidad del sistema. Para ello, seleccionan un subconjunto de nodos llamado *clusterhead* para formar un servicio de directorios donde

los nodos del cluster registran sus servicios. Cada nodo de la red está asociado a un *clusterhead*.

Siguiendo con la agrupación de nodos en *clusters*, [Monire et al, 2012] proponen una solución donde los nodos cercanos se agrupan en *clusters*. Cada *cluster* abarca un diámetro de dos saltos y elige a un *clusterhead* que se encarga de coordinar el resto de miembros del *cluster*. Las rutas entre los diferentes *cluster* se descubren dinámicamente. Cada nodo mantiene una tabla con los nodos vecinos y otra con los *cluster* colindantes. [Siddarth et al, 2013] proponen una arquitectura de descubrimiento de servicios basada en procesos de inteligencia colectiva *Swarm Intelligence* (SI). La SI se trata de una rama de la inteligencia artificial inspirada en la naturaleza y basada en el estudio de comportamientos colectivos presentes en la naturaleza, generalmente de caracter descentralizado y auto organizativo. Este comportamiento social, define los movimientos de las variables de decisión en el espacio de búsqueda y las orienta hacia soluciones óptimas. En este caso, los autores utilizan la SI para establecer el camino más corto entre los nodos que forman un *cluster* y también el camino más corto para comunicar diferentes *clusters* entre sí. Cada nodo envía su petición de servicio con sus requisitos en cuanto a parámetros de calidad de servicio (QoS) se refiere al *clusterhead*. Cada *clusterhead* busca el nodo servidor que ofrezca el servicio basándose en los parámetros de QoS definidos de entre unas tablas de servicio y unas tablas de nodos servidores que tiene almacenadas.

2.5. Protocolos de descubrimiento de servicios no basados en directorios

A diferencia de las soluciones basadas en el uso de directorios, este tipo de propuestas se basa en la difusión de anuncios de servicios y solicitudes de búsqueda de servicios a través de los nodos de la red. De esta forma se evita centralizar toda la información referente a anuncios de servicios en un conjunto de nodos seleccionados como directorios de servicios.

Una de las primeras propuestas de este tipo de soluciones es el protocolo de descubrimiento y anuncio de servicios denominado DEAPspace Algorithm [Nidd, 2001]. A través de este algoritmo, un nodo puede detectar la presencia de nodos próximos, compartir información de los servicios que están disponibles en la red y detectar cuando un nodo deja de estar disponible. El protocolo ha sido diseñado para operar de forma eficiente en redes inalámbricas *Ad-hoc* de un solo salto, y su objetivo principal es dar respuesta a los cambios frecuentes que se producen en este tipo de entornos, teniendo en cuenta las restricciones de potencia y

limitaciones de los dispositivos. El algoritmo se basa en un método "*push*" puro, en el que todos los dispositivos mantienen una "*world view*" (visión del mundo) que transmiten cada cierto periodo de tiempo a sus vecinos mediante mensajes de *broadcast* y que actualizan escuchando el "*world view*" que tienen los demás.

En este tipo de protocolos de descubrimiento de servicios y dadas las características de las redes MANET es muy importante controlar la distancia a la que los nodos difunden sus anuncios de servicios. Normalmente esta distancia se controla mediante contadores de tiempo que se definen en el propio mensaje de anuncio de servicios. Estos contadores de tiempo se van decrementando por cada salto que el paquete realiza en la red. En este sentido, [Ratsimor et al., 2002] proponen una arquitectura distribuida basada en políticas. Emplean una caché de servicios dentro de una red P2P (*Peer to Peer*). Presentan el concepto de *alianza* de un nodo, que es un conjunto de nodos cuya información de servicio local es almacenada por este nodo. La política local de un nodo establece la forma en que quiere anunciarse a los otros nodos del vecindario. La arquitectura del nodo y los componentes de la plataforma son: *gestor de políticas*, responsable de controlar por ejemplo con qué frecuencia se anuncian servicios locales, qué servicios se almacenan en la caché, ..., *gestor de anuncios*, responsable de anunciar mediante mensajes periódicos de *broadcast* los servicios ofrecidos por los agentes locales, *gestor de caché*, responsable de almacenar de forma local los servicios ofrecidos por agentes que residen en dispositivos remotos, *gestor de reenvíos*, responsable de reenviar peticiones de servicios de dispositivos vecinos, para que se propagasen a otros dispositivos de la red.

Otra solución similar a DEAPspace es el protocolo *Pervasive Discovery Protocol* (PDP) [Campo et al, 2006]. PDP es un protocolo totalmente distribuido, simple y poco costoso computacionalmente que reduce el consumo de baterías ya que disminuye considerablemente el número de mensajes transmitidos por búsqueda. Se trata de un protocolo de descubrimiento de servicios adaptado a entornos de computación ubicua, que permite a los dispositivos obtener información sobre la existencia y localización de servicios proporcionados en el entorno próximo que les rodea. PDP elimina la necesidad de que el usuario conozca previamente el tipo y los servicios que pueden proporcionarle otros dispositivos con los que puede comunicarse en algún momento. El usuario, a través de sus aplicaciones, podrá solicitar la búsqueda de un tipo de servicio al que quiere tener acceso, y si así lo requiere, podrá conocer también todos los servicios que los dispositivos que le rodean pueden ofrecerle.

Group Based Service Discovery (GSD) [Chakraborty et al, 2006] es un protocolo de descubrimiento de servicios que sigue los modelos *push* y *pull*. La novedad en este caso consiste en la retransmisión selectiva de los mensajes de petición de

servicios. Los nodos que reciben un mensaje de este tipo, lo retransmiten basándose en información relativa al grupo al que pertenece el servicio. GSD explota las capacidades semánticas ofrecidas por *DARPA Agent Markup Languaje* (DAML) para describir de forma efectiva los servicios de la red. Los servicios se clasifican dentro de la jerarquía de clase y subclase de DAML. Las peticiones de servicios se expresan en DAML y son emparejadas con descripciones de servicios usando un módulo de *service-matching*. De esta forma se unen la descripción y el descubrimiento de servicios para intentar minimizar el tráfico en la red.

Por otro lado, en el ámbito del uso de la tecnología de agentes móviles en redes MANET, [Neogy et al, 2012] presentan una protocolo de descubrimiento de servicios donde los agentes pueden descubrir los servicios ofrecidos por los diferentes nodos de la red e interactuar con ellos. Los agentes difunden la información de esos servicios al resto de nodos de la red. Del mismo modo, los nodos de la red comparten con los agentes móviles el conocimiento sobre los servicios que ellos tienen, de tal forma que los agentes móviles puedan tener un conocimiento de los servicios que ofrecen los nodos servidores sin necesidad de descubrirlos ellos mismos.

Por último, existen otras soluciones en las que son los nodos intermedios los que responden a las solicitudes de búsqueda de servicios generadas por cualquier nodo de la red. [Palmieri et al, 2013] proponen una solución donde cualquier nodo de la red al recibir anuncios de servicios del resto de nodos, los almacena y en caso de que otro nodo hiciera una petición de cualquier servicio que él tuviera almacenado lo podría responder. De esta forma se evita que las peticiones de servicio se retransmitan hasta los nodos que ofrecen los servicios y se reduce así el consumo de ancho de banda en la red.

En general, uno de los objetivos de todos estos protocolos es reducir el número de mensajes que se difunden en la red para reducir el consumo de ancho de banda en la red, crítico en las redes MANET.

2.5.1. Descubrimiento de servicios *cross-layer*

El envío de mensajes de anuncio y descubrimiento de servicios puede integrarse como parte de los protocolos de encaminamiento de la red MANET, con el fin de reducir el número de mensajes enviados a través de la red y permitir al mismo tiempo conocer los servicios que se anuncian en la red y las rutas para acceder a esos servicios. Al hecho de integrar funciones propias de la capa de aplicación como es el descubrimiento de servicios dentro de la capa de red del modelo *Open Systems Interconnection* (OSI) se conoce con el término *cross-layer*.

En esta sección vamos a ver en detalle en qué consiste un diseño *cross-layer* y sus ventajas a la hora de aplicarlo en las redes MANET. También describiremos los mecanismos de descubrimiento de servicios más relevantes surgidos en los últimos años integrados en protocolos de encaminamiento proactivos y protocolos de encaminamiento reactivos.

Diseño *cross-layer*

La arquitectura de las redes cableadas basa su diseño en el modelo de referencia *Open Systems Interconnection* (OSI) creado en los años 80 por la *International Standard Organization* (ISO), para solucionar el problema derivado de la gran cantidad de redes con implementaciones hardware y software diferentes que existían y para las que era muy difícil comunicarse entre sí.

El modelo OSI se basa en una arquitectura de capas donde la función y los servicios de cada una de las capas están perfectamente definidos. Los protocolos definidos para cada capa tienen en cuenta los servicios que proporciona la capa inferior pero les es indiferente cómo define cada capa estos servicios. Del mismo modo las interfaces de comunicación entre las diferentes capas se limitan a intercambiar las primitivas que se han especificado en el modelo OSI.

Las ventajas de este modelo radican en que:

- Divide la comunicación de red en partes más pequeñas y sencillas.
- Normaliza los componentes de red de tal forma que permite el desarrollo y el soporte de productos de diferentes fabricantes.
- Los cambios de una capa no afectan a las demás capas y por lo tanto, cada capa puede evolucionar más rápido.

Sin embargo, este modelo basado en capas no resulta suficientemente flexible en una red MANET, dados los continuos cambios en la topología de estas redes y la limitación de los nodos que las integran [Tian et al, 2005], [Saleem et al, 2013]. Las redes MANET son redes inalámbricas, multisalto que no requieren de una infraestructura definida, que se forman espontáneamente y cuyos nodos están en continuo movimiento. Además, generalmente los nodos que integran estas redes tienen reducidas capacidades de procesamiento y batería. Todo esto hace que la sobrecarga debido a paquetes de control en estas redes sea grande. Por ello, resulta vital optimizar el rendimiento de las redes MANET. En base a este hecho, ha surgido una nueva tendencia en el desarrollo de protocolos específicos para este

tipo de redes, denominada diseño *cross-layer* que mejora la eficiencia de estas redes, mediante el intercambio de información entre las capas del modelo OSI.

La idea original del diseño *cross-layer* es mantener la funcionalidad asociada a cada capa, y permitir que protocolos de diferentes capas se coordinen y compartan parámetros para mejorar el rendimiento de la red. Realmente el concepto *cross-layer* no es completamente nuevo en el entorno de las redes de comunicaciones. También ha sido propuesto como una optimización de la arquitectura de capas en redes cableadas, lo que se ha llamado *cross-layer feedback* [Clark, 1985] [Clark et al, 1990]. Entendiendo por *cross-layer feedback* a la interacción de una capa con cualquier otra capa de la arquitectura OSI.

Al contrario que ocurre con el modelo OSI y una arquitectura sujeta a un modelo de capas, una solución *cross-layer* implica el incumplimiento de todas las reglas definidas en el modelo OSI. En un modelo de capas se optimiza cada capa sin tener en cuenta el resto, sin embargo, en un diseño *cross-layer* se optimiza una visión global conjunta, de tal forma que es posible que todas las capas necesarias de la arquitectura reaccionen conjuntamente, adaptándose a las condiciones del entorno. Es por ello que es adecuado su uso para redes MANET.

Existen diferentes propuestas de diseño *cross-layer* en la literatura. En [Raisinghani et al, 2003] los autores presentan una visión general de las propuestas existentes basándose en las capas que interaccionan entre ellas. Todas estas propuestas incumplen ciertos aspectos de la arquitectura basada en capas. Este incumplimiento se puede deber a que se creen nuevas interfaces, se combinen funcionalidades de capas adyacentes, se definan nuevos protocolos resultado de la combinación de capas adyacentes y lo que se denomina calibración vertical, es decir, que se ajusten parámetros específicos de cada capa considerando la consecuente adaptación en el resto de capas.

- *Creación de nuevas interfaces.* Una arquitectura basada en capas impide la comunicación directa entre dos capas no adyacentes y la comunicación entre capas adyacentes se limita a las primitivas definidas para tal efecto. Un diseño *cross-layer* de este tipo permite, sin embargo, crear nuevas interfaces para compartir información entre capas.

- *Integración de dos capas adyacentes.* En este caso la nueva capa ofrecerá los servicios de las dos capas que integra. No es necesario que se creen nuevas interfaces, ya que esta nueva capa puede hablar con el resto de capas con las interfaces ya definidas en la arquitectura original.

- *Calibración vertical.* En este caso se condiciona el funcionamiento de una capa al funcionamiento de otra con lo que no se puede reemplazar ni modificar

una capa sin rediseñar la otra.

El diseño *cross-layer* aplicado a las redes MANET, hace referencia a diseñar los protocolos específicos para este tipo de redes explotando la dependencia de las capas de la arquitectura OSI, y no diseñar los protocolos de manera independiente para cada una de las capas. De esta forma se obtienen mejores prestaciones en el rendimiento de los protocolos, dado el carácter dinámico e inalámbrico de estas redes [Srivastava et al, 2005]. En el caso de las redes MANET el incumplimiento de la arquitectura basada en capas se basa principalmente en una interacción entre diferentes capas.

Hoy en día existen diferentes propuestas *cross-layer* en el ámbito de las redes MANET. Cada una de ellas se ha desarrollado de manera independiente y no tiene por qué compartir ningún criterio con otra propuesta *cross-layer*. Dado el abanico de posibilidades de diseño que presenta un planteamiento *cross-layer* y la dificultad para definir el método más adecuado y/o común para cualquier diseño, actualmente los esfuerzos se están centrando en estudiar protocolos de encaminamiento eficientes que garanticen una QoS óptima y una gestión de la energía eficiente. En este contexto, se han desarrollado diferentes propuestas [Antonopoulos et al, 2014], [Zuo et al, 2014].

Un diseño *cross-layer* aplicado al descubrimiento de servicios se refiere a desarrollar protocolos de descubrimiento de servicios integrados en un protocolo de encaminamiento de red. De tal forma que los mensajes de anuncio de servicios se retransmitan por toda la red al mismo tiempo que lo hacen los mensajes de descubrimiento de rutas. Tradicionalmente, en las redes cableadas, los protocolos de descubrimiento de servicios se han situado en la capa de aplicación. Para ello lo primero que se hacía era descubrir el proveedor de servicios que ofrecía el servicio buscado y posteriormente buscar la ruta para acceder a ese proveedor. Esto obligaba a desencadenar dos procesos diferentes, cada uno de los cuales sobrecargaba la red con una gran cantidad de mensajes. En una red MANET cada mensaje que se transmite por la red consume ancho de banda además de recursos en cuanto a procesamiento de datos y energía de los nodos móviles. Las soluciones basadas en un diseño *cross-layer* referentes al descubrimiento de servicios disminuyen considerablemente el ancho de banda consumido en la red y la cantidad de mensajes transmitidos [García-Macías et al, 2005], [Halkes et al, 2006], [Radhamani et al, 2011]. Además este tipo de soluciones reducen el tiempo en acceder a un servicio ya que los procesos de descubrimiento de servicios y de rutas se realizan al mismo tiempo y no de manera independiente como ocurría cuando el descubrimiento de servicios se situaba en la capa de aplicación.

En este contexto se han definido varias propuestas que se explican en detalle a continuación.

Descubrimiento de servicios en redes proactivas

La principal ventaja de integrar el descubrimiento de servicios en un protocolo de encaminamiento proactivo es que dada la naturaleza proactiva de estos protocolos, los nodos tienen un conocimiento exhaustivo del estado de la red, de modo que cuando necesitan una ruta, ésta ya es conocida y está lista para utilizarse de manera inmediata. Lo mismo ocurre con un servicio. Cuando un nodo necesite un servicio que se ofrezca en la red, éste ya conocerá qué nodo ofrece ese servicio.

Describimos a continuación los principales protocolos de descubrimiento de servicios integrados en protocolos de encaminamiento proactivos.

[Li et al, 2005] proponen un mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSR que descubre servidores *Session Initiation Protocol* (SIP) en una red MANET. Para ello, definen un nuevo tipo de mensaje que llaman *Service Location Extension* (SLE). Los nodos servidores SIP utilizan este mensaje para periódicamente anunciar sus servicios y los nodos cliente también utilizan el mensaje SLE para hacer peticiones de servicio buscando los nodos SIP.

Mercury [Flathagen et al, 2008]. Al igual que nuestro mecanismo de descubrimiento de servicios diseñado sobre la OLSRv1, *Mercury* introduce un nuevo mensaje denominado *Mercury Service Discovery* (MSD) dentro del paquete genérico OLSR. Al igual que la primera versión que nosotros definimos [Vara et al, 2006] y [Jodra et al, 2006], *Mercury* tiene la posibilidad tanto de solicitar como de anunciar servicios (modos *pull* y *push*). Una de las diferencias con nuestro mecanismo de descubrimiento de servicios es que *Mercury* describe los servicios utilizando un filtro Bloom. Este tipo de filtros son capaces de en un pequeño espacio de memoria (128 bytes) contener un resumen de todos los servicios disponibles en el dispositivo (hasta un máximo de 256). Para ello crea una función *hash* para cada servicio, mapeando su resultado en la lista de servicios disponibles. El problema de utilizar este tipo de filtros es que puede llegar a dar un falso positivo si un nodo ofrece muchos servicios y se solapan sus resultados en el espacio de memoria asignado.

Descubrimiento de servicios en redes reactivas

La mayoría de las soluciones para descubrimiento de servicios integradas en un protocolo de encaminamiento reactivo utilizan *Ad-hoc On-Demand Distance Vector* (AODV) [Chakeres et al, 2007] como protocolo de encaminamiento reactivo.

El primero que introdujo esta idea fue [Koodli et al, 2002]. Vamos a explicar

más en detalle esta propuesta de descubrimiento de servicios puesto que es en la que se basan la gran mayoría de soluciones que integran el descubrimiento de servicios en protocolos reactivos y también porque es la solución con la que comparamos en la sección 5.3 nuestro mecanismo de descubrimiento de servicios SD-OLSRv2. En el *draft* del protocolo proponen que un nodo que necesite un servicio descubra el nodo que ofrece el servicio al mismo tiempo que la ruta hacia ese nodo. En el caso de que el nodo que busca el servicio ya conozca el nodo proveedor del servicio y la ruta hacia él, no es necesario iniciar un proceso de petición del servicio. En caso contrario, el nodo lanza un mensaje de petición de rutas con una extensión para la solicitud del servicio que se llama *Service Request* (SREQ). Este proceso termina cuando el nodo recibe un mensaje de respuesta al mensaje de petición de rutas con una extensión con la respuesta del servicio solicitado, *Service Reply* (SREP).

En las figuras 2.1 y 2.2 se muestra el formato de una petición de servicio SREQ y una respuesta SREP. El significado de cada campo es el siguiente. Para una petición de servicio SREQ:

- *Type*. Campo de 1 byte. Por determinar
- *Length*: Campo de 1 byte. Indica la longitud de de la petición de servicio.
- *ServiceTypeLen*: Campo de 1 byte. Indica la longitud del campo *Service Type* del servicio.
- *Reserved*: Campo de 1 byte. Su uso está reservado.
- *Service Type*: Tipo de servicio. Se define según se especifica en el protocolo SLP [Guttman, 1999].
- *Service Predicate*. Service predicate. Se define según se especifica en el protocolo SLP [Guttman, 1999].

Y para una respuesta SREP:

- *Type*. Campo de 1 byte. Por determinar
- *Length*: Campo de 1 byte. Indica la longitud de de la petición de servicio.
- *Lifetime*: Campo de 2 bytes. Tiempo de vida del servicio.
- *ServiceURLLen*: Campo de 1 byte. Longitud del campo *service URL*.

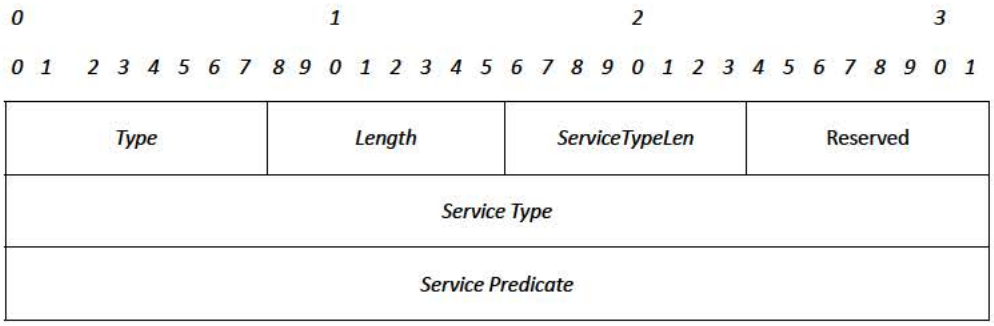

Figura 2.1: Petición de servicio, SREQ, del protocolo SD-AODV

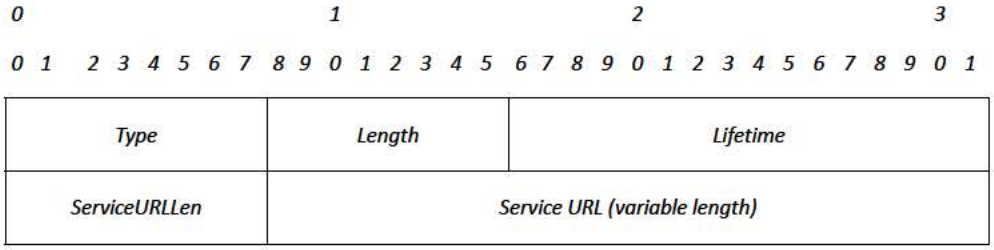

Figura 2.2: Respuesta a la petición de servicio, SREP, del protocolo SD-AODV

- *Service URL*: *Uniform Resource Locator* (URL) del servicio. Se define según se especifica en el protocolo SLP [Guttman, 1999].

Como vemos tanto el tipo de servicio como la descripción están basados en el formato definido por el protocolo SLP [Guttman, 1999].

Partiendo de esta solución a modo de *draft* que acabamos de ver, han sido muchos los que han ido creando soluciones y añadiendo extensiones sobre la misma. En [García-Macías et al, 2005], los autores comparan el descubrimiento de servicios integrado en el protocolo de encaminamiento AODV, con un protocolo de descubrimiento de servicios definido a nivel de aplicación denominado Nom. En [Athanaileas et al, 2007] y [Fan et al, 2007] añaden extensiones al protocolo SD-AODV para que los servicios se puedan seleccionar basándose en parámetros de QoS.

SEDIRAN[Obaid et al, 2007], y la propuesta de [Engelstad et al, 2006] son otros ejemplos de este tipo de arquitecturas. SEDIRAN es un protocolo de descubrimiento de servicios integrado en el protocolo de encaminamiento AODV que encapsula tres nuevos paquetes en el mensaje *Routing Replay* (RREP) del protocolo AODV: la solicitud del servicio *Discovery Request* (DREQ), la respuesta a dicha solicitud, *Discovery Reply* (DREP), y el anuncio de un servicio *Advertisement Message* (ADVM). Por lo tanto, para difundir tanto las solicitudes como los anuncios de los servicios por toda la red, el protocolo utiliza el mensaje de respuesta RREP del protocolo de encaminamiento AODV, que no está diseñado para enviar mensajes por *broadcast*. Para ello, es necesario modificar el protocolo AODV para incluir el mecanismo de descubrimiento de servicios. Esto conlleva que aquel nodo que no tenga instalado SEDIRAN no puede reenviar los mensajes que le lleguen, haciendo que el mecanismo de descubrimiento de servicios no funcione correctamente.

En SD-AODV, [Engelstad et al, 2006] modifican los formatos de los mensajes *Routing Request Packets* (RREQ) y RREP del protocolo de encaminamiento AODV para solicitar servicios y responder a las solicitudes realizadas respectivamente. Este protocolo permite el uso tanto de la arquitectura distribuida como de la arquitectura híbrida. En el caso de la arquitectura híbrida los anuncios de los nodos directorio son encapsulados en los mensajes RREQ y los registros de servicios en el directorio son encapsulados en los mensajes de respuesta RREP. Además, cada nodo posee una caché interna que contiene una lista con la información de los servicios de los demás nodos de la red.

[Zhong et al, 2012] también proponen un mecanismo de descubrimiento de servicios llamada SCAODV que integra el mecanismo de descubrimiento de servicios sobre el protocolo de encaminamiento AODV.

Recientemente, [Bhumika, 2015] ha ampliado la solución de descubrimiento de servicios integrada en el protocolo de encaminamiento AODV definida por [García-Macías et al, 2005], para descubrir servicios en la red MANET de forma eficiente y también para comunicarse con el nodo servidor que ofrezca una mayor confianza en términos de seguridad.

2.6. Conclusiones

En este capítulo hemos realizado un estudio teórico del problema del descubrimiento de servicios, analizando las diferentes soluciones teóricas que existen. Primero, en la sección 2.1 hemos definido lo que es un servicio. A continuación en la sección 2.2 hemos visto las soluciones teóricas de funcionamiento distribuido, denominadas modo *push*, en la que los servidores se anuncian periódicamente y así los clientes los descubren, y modo *pull*, en la que los clientes son los que preguntan por servicios cuando los necesitan y son los servidores los que responden a estas peticiones.

En la sección 2.3, hemos descrito las soluciones que se han proporcionado en los últimos años al problema del descubrimiento de servicios en redes con infraestructura fija. Hemos visto que estas soluciones no son aptas para funcionar en redes MANET dadas las características inherentes a este tipo de redes como son la movilidad de los nodos y una topología de red cambiante.

En las secciones 2.4 y 2.5, hemos presentado una revisión bibliográfica de los principales mecanismos de descubrimiento de servicios definidos hasta la fecha, específicos para funcionar en redes MANET, que nos ha permitido conocer las características que un mecanismo de descubrimiento de servicios para redes MANET debe cumplir. En estas secciones hemos visto protocolos de descubrimiento de servicios para redes MANET que definen una arquitectura de red basada en directorios y protocolos de descubrimiento de servicios que no utilizan directorios para centralizar la información de los servicios que se anuncian en la red, sino que por el contrario, difunden esta información a toda la red. Sobre cuál de las dos soluciones es mejor no existe un consenso claro. [Ververidis et al, 2008] dicen que el motivo principal para esta indecisión es que existen muchos factores que hay que evaluar cuando se diseña un protocolo de descubrimiento de servicios nuevo para una red MANET: movilidad de los nodos, periodicidad con la que se envían mensajes de anuncio de servicios o peticiones de solicitud de servicios etc. Si la red es muy móvil es mejor diseñar un protocolo de descubrimiento de servicios sin directorios para evitar los problemas derivados de tener que enviar anuncios frecuentes. Sin embargo, si se trata de una red con un número elevado

de peticiones de servicio, es preferible un protocolo de descubrimiento de servicios basado en una arquitectura de directorios ya que todos las peticiones se realizarían a los mismos nodos y no sería necesario difundir estas peticiones a toda la red. Sin embargo, [Engelstad et al, 2005] sí que concluyen que para redes MANET una solución de descubrimiento de servicios basada en una arquitectura sin directorios puede ser mejor que una basada en directorios.

Respecto a integrar o no el mecanismo de descubrimiento de servicios en la capa de red OSI, hemos visto que tradicionalmente, en las redes cableadas, los protocolos de descubrimiento de servicios se han situado en la capa de aplicación. Primero, se descubría el proveedor de servicios que ofrecía el servicio buscado y posteriormente se buscaba la ruta para acceder a ese proveedor. Esto obligaba a desencadenar dos procesos diferentes, cada uno de los cuales sobrecargaba la red con una gran cantidad de mensajes. Tal como hemos visto, en una red MANET cada mensaje que se transmite por la red consume ancho de banda además de recursos en cuanto a procesamiento de datos y energía de los nodos móviles. Las soluciones basadas en un diseño *cross-layer* referentes al descubrimiento de servicios disminuyen considerablemente el ancho de banda consumido en la red y la cantidad de mensajes transmitidos [García-Macías et al, 2005], [Halkes et al, 2006], [Radhamani et al, 2011].

Uno de los objetivos planteados en la realización de esta tesis doctoral es la definición de un mecanismo de descubrimiento de servicios para redes MANET con un número de nodos elevado, donde el número de mensajes transmitidos a la red sea pequeño de tal forma que no se sobrecargue en exceso la misma y donde el tiempo de respuesta para encontrar un servicio sea pequeño. Para ello vemos necesario integrar el mecanismo de descubrimiento de servicios en el protocolo de encaminamiento de la red.

Otro de los objetivos planteados es que el mecanismo propuesto pudiera evolucionar y adaptarse a la evolución del protocolo de encaminamiento.

Ninguna de los soluciones que hemos visto sobre OLSR se han adaptado a los cambios cuando se ha estandarizado la versión 2 de OLSR. Todas además, no se benefician del carácter proactivo del protocolo OLSR y definen mensajes tanto para anunciar servicios como para hacer peticiones de servicios.

Por lo tanto, vemos necesario definir un nuevo protocolo de descubrimiento de servicios que proporcione una solución eficiente para redes MANET con protocolos de encaminamiento OLSR. En los capítulos 4 y 5 realizaremos un diseño razonado del mecanismo de descubrimiento de servicios propuesto para ambas versiones del protocolo de encaminamiento OLSR.

Como se indica en la sección 2.1 de este capítulo, no es objetivo de esta

tesis doctoral definir un formato de descripción de servicios, y el mecanismo de descubrimiento de servicios que vamos a proponer es independiente de este formato y de cómo se realiza la correspondencia entre el servicio buscado y el servicio ofrecido.

Capítulo 3

Encaminamiento en redes MANET

Este capítulo se centra en la problemática del encaminamiento en redes MANET. En un primer lugar, explicamos por qué son necesarios protocolos de encaminamiento diseñados específicamente para redes MANET y cuáles son las características que tienen que cumplir para funcionar adecuadamente en este tipo de entornos. A continuación, mostramos la clasificación más habitual de este tipo de protocolos y por último, describimos en detalle las dos versiones del protocolo de encaminamiento OLSR.

3.1. Protocolos de encaminamiento

El concepto *encaminamiento* se define como el proceso de transmisión de información desde un nodo origen a un nodo destino por el camino más óptimo. Un protocolo de encaminamiento tiene, por lo tanto, dos tareas básicas que realizar. En primer lugar, recoger información del estado de la red y mantenerla actualizada y en segundo lugar, encontrar el camino óptimo para transferir los paquetes a través de la red. Para determinar el camino óptimo desde un nodo origen a un nodo destino cada nodo mantiene tablas de encaminamiento con la información total de cada ruta. Dependiendo del tipo de protocolo de encaminamiento que se utilice, estas tablas de encaminamiento guardarán una u otra información.

Los protocolos de encaminamiento utilizados en las redes convencionales, tanto fijas como cableadas se han diseñado suponiendo que la topología de la red es estable y poco cambiante. Los protocolos de encaminamiento más utilizados en este tipo de redes son los algoritmos de *vector de distancia* [Bellman, 1957] y

estado del enlace [Ford et al, 1962]. En el primer caso, cada nodo mantiene una estructura de datos llamada vector de distancias, donde almacena al menos la dirección IP (*Internet Protocol*) del nodo destino, la dirección IP del siguiente salto y la distancia más corta para alcanzarlo. Cada nodo de la red intercambia periódicamente sus vectores de distancia con sus nodos vecinos. La principal desventaja de este algoritmo es la lenta convergencia y la tendencia a crear bucles. En el segundo caso, la información de encaminamiento se transmite en forma de *Link State Packet* (LSP). Los paquetes que envía cada nodo incluyen información acerca de sus nodos vecinos y cualquier cambio en alguno de sus enlaces provoca inmediatamente la inundación de la red con mensajes anunciando estos cambios. De esta forma cada nodo es capaz de reconstruir y mantener un mapa de la topología global de la red a través de los LSP que recibe, pudiendo calcular por sí mismo las rutas hacia cualquier otro nodo. El principal inconveniente de este algoritmo es que puede introducir mucha sobrecarga en la red en el caso de que ésta sea muy dinámica.

Este tipo de protocolos no se pueden aplicar a las redes MANET debido a la gran cantidad de intercambio de mensajes para actualización de rutas que requieren, lo que supondría un consumo importante de recursos de ancho de banda, capacidad de batería y procesamiento en los nodos de la red MANET. Es necesario que los protocolos de encaminamiento que se definan para las redes MANET, se adapten rápidamente a los constantes cambios de topología que existen en estas redes, además de que deben trabajar con tasas altas de error para los enlaces debido a las características inalámbricas de estas redes.

Un protocolo de encaminamiento diseñado para redes MANET debe satisfacer los siguientes criterios:

- *Señalización mínima.* Cuantos menos mensajes de señalización y control se difundan, menos consumo de ancho de banda habrá en la red.
- *Tiempo de procesamiento mínimo.* El tiempo que un nodo tarde en procesar los mensajes de encaminamiento debe ser mínimo de cara a alargar el tiempo de vida de la batería de los nodos.
- *Mantenimiento de rutas.* El algoritmo debe ser capaz de localizar una nueva ruta rápidamente cuando la topología de la red cambie.
- *Capacidad multisalto.* Los nodos de la red deben poder reencaminar los paquetes dado que normalmente el nodo destino no se encuentran en comunicación directa con el nodo origen.

Teniendo en cuenta estos criterios se han diseñado protocolos de encaminamiento específicos para redes MANET que se describen en la sección siguiente.

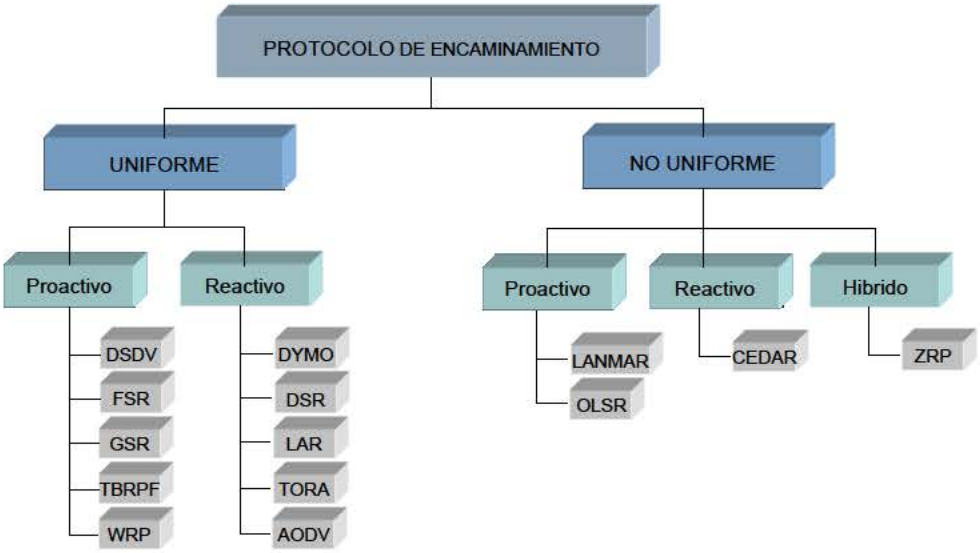

Figura 3.1: Clasificación de los protocolos de encaminamiento en redes MANET

3.2. Clasificación de los protocolos de encaminamiento

Existen una gran cantidad de protocolos de encaminamiento definidos para estas redes que se pueden clasificar atendiendo a diferentes criterios. En [Jayakumar et al, 2007] se resumen las diversas clasificaciones que existen.

Tal como se muestra la figura 3.1, la clasificación más común divide los protocolos de encaminamiento en varios niveles. El primer nivel se refiere a la homogeneidad o heterogeneidad de las funciones que realizan los nodos respecto al encaminamiento. Así, podemos distinguir:

- *Protocolos uniformes.* En este caso todos los nodos de la red realizan las mismas funciones y envían y responden a los mensajes de control del mismo modo.
- *Protocolos no uniformes.* En este caso no todos los nodos desarrollan las mismas funciones. Existen nodos que desempeñan funciones especiales en cuanto al tratamiento de la información de encaminamiento se refiere. Dentro de este tipo se pueden distinguir entre los que hacen uso del método de selección de vecinos, en el caso de que los nodos realicen el encaminamiento sólo a través de algunos nodos vecinos, y los que emplean el método de particionamiento, en el caso de que la red se divida en subredes.

Dentro de estas dos categorías, los protocolos se pueden clasificar nuevamente en protocolos proactivos, protocolos reactivos y protocolos híbridos atendiendo al procedimiento adoptado para el descubrimiento y mantenimiento de rutas.

3.2.1. Protocolos proactivos

Los protocolos de encaminamiento proactivos o basados en tablas son aquellos algoritmos donde cada nodo de la red mantiene en todo momento información actualizada de las rutas hacia todos los nodos que integran la red. Cada nodo almacena esta información en tablas de encaminamiento que se actualizan periódicamente y también cuando existe algún cambio en la topología de la red. Este tipo de protocolos está basado en los protocolos de encaminamiento vector de distancia y estado del enlace definidos para redes tradicionales. La diferencia entre ellos está en la forma de actualizar y detectar la información de encaminamiento y el tipo de información que se almacena en cada tabla. Los protocolos proactivos más conocidos son:

- *Destination-sequenced Distance Vector*(DSDV) [Perkins et al, 1994] basado en el algoritmo de vector de distancia.
- *Optimized Link State Routing* (OLSR) [Clausen et al, 2003], [Clausen et al, 2014] basado en el algoritmo de estado de enlace.

Aunque existen también otros como los que hemos visto en la figura 3.1.

- *Fisheye State Routing*(FSR) [Gerla et al, 2002].
- *Global State Routing* (GSR) [Chen al, 1998].
- *Topology Dissemination Based on Reverse-Path Forwarding* (TBRPF) [Ogier et al, 2004].
- *Wireless Routing Protocol* (WRP) [Murphy et al, 1996].
- *Landmark Routing* (LANMAR) [Pei et al, 2000].

La ventaja que ofrecen este tipo de protocolos es la baja latencia ya que las rutas a cada destino siempre están disponibles. Sin embargo, el hecho de que periódicamente se propaguen las rutas hacia todos los nodos de la red, causa que exista una gran cantidad de mensajes de señalización que incluyen caminos que nunca son utilizados y que afectan al ancho de banda, el *throughput* y el consumo de energía. En general este tipo de protocolos resulta adecuado cuando:

- Las rutas deben estar disponibles lo antes posible.
- La longitud de las rutas es indeterminada, pueden ser cortas, largas o una combinación de ambas.
- El número de rutas que se utilizan es alto frente al número de rutas disponibles.

3.2.2. Protocolos reactivos

Los protocolos de encaminamiento reactivos o bajo demanda son aquellos algoritmos en los que únicamente se crean rutas cuando un nodo origen necesita enviar información a un nodo destino. El objetivo es reducir el consumo de energía y la sobrecarga que introducen en la red los protocolos proactivos. Si un nodo origen no conoce la ruta hacia un nodo destino, el descubrimiento de la ruta lo realiza enviando mensajes de solicitud de ruta en modo *broadcast* por toda la red. Este tipo de protocolos resulta adecuado cuando:

- Es asumible una latencia alta para el primer paquete.
- La topología de la red no es muy dinámica. Las rutas cambian, pero una vez descubierta una ruta puede ser utilizada varias veces antes de que deje de ser válida.
- El número de rutas que se utilizan es bajo, frente al número de rutas disponibles.

Este tipo de protocolos se puede dividir en dos tipos:

- **Basados en la fuente** (*Source-Based*). En este tipo de protocolos, en la cabecera de los paquetes de datos se transporta la ruta completa del nodo origen al nodo destino. De esta forma los nodos intermedios saben hacia dónde tienen que reencaminar los paquetes y no son necesarias tablas de encaminamiento en los nodos intermedios. En el caso de que la red la integren muchos nodos y el camino entre el nodo origen y el nodo destino sea muy largo, este tipo de protocolos no es aconsejable ya que la cabecera aumenta de tamaño y se pierde ancho de banda. De la misma forma, en el caso de redes con mucha movilidad es posible que el enlace se pierda o se rompa y la ruta que se transporta en la cabecera del paquete de datos deje de ser válida.

- **Salto a salto** (*hop-by-hop*) o **punto a punto** (*point-to-point*). En este tipo de protocolos, en la cabecera de los paquetes de datos únicamente se transporta la dirección del nodo destino y la dirección del próximo salto. En este caso los nodos intermedios sí que tienen que mantener tablas de encaminamiento para conocer cuál es el siguiente nodo al que tienen que encaminar el paquete de datos. Esto supone un gasto computacional en los nodos mayor que en los protocolos *basados en la fuente* pero en contrapartida se adapta mejor a los cambios de topología de la red.

Los principales protocolos reactivos son:

- *Dynamic Source Routing* (DSR) [Jonson et al, 2007] basado en la fuente.
- *Ad-hoc On-Demand Distance Vector* (AODV) [Chakeres et al, 2007] que funciona salto a salto.

Existen también otros protocolos de encaminamiento reactivos como los que hemos visto en la figura 3.1.

- *Dynamic MANET On-demand*(DYMO) [Chakeres et al, 2009].
- *Location aided routing* (LAR) [Ko et al, 1998].
- *Temporally-Ordered Routing Algorithm* (TORA) [Park et al, 2001].
- *Core Extraction Distributed Ad Hoc Routing* (CEDAR) [Sivakumar et al, 1999].

3.2.3. Protocolos híbridos

Los protocolos de encaminamiento híbridos son una mezcla de los protocolos proactivos y protocolos reactivos, y combinan las ventajas de ambos. Este tipo de protocolos suele dividir la red en un número determinado de zonas o *clusters* creando así diferentes grupos de nodos. En general, el encaminamiento entre nodos cercanos se realiza mediante un protocolo proactivo y el de nodos lejanos con un protocolo reactivo. Un ejemplo de este tipo de protocolos es el protocolo *Zone Routing Protocol* (ZRP) [Haas et al, 2002].

3.3. Protocolo OLSR

El protocolo proactivo OLSR es una optimización del protocolo de estado de enlace *Open Shortest Path First* (OSPF) [Moy, 1998] pero adaptado para redes MANET. A día de hoy existen dos versiones del protocolo proactivo OLSR, ambas definidos como *Request For Comments* (RFC). La versión 1 del protocolo que se especifica en la RFC 3626 [Clausen et al, 2003] y la versión 2 del protocolo que se especifica en la RFC 7181 [Clausen et al, 2014].

Al tratarse de un protocolo proactivo cada nodo mantiene una ruta al resto de nodos de la red que se actualiza periódicamente. Para ello los nodos que integran la red intercambian periódicamente mensajes de estado de enlace y no sólo cuando se detectan cambios en la topología de la red.

La principal ventaja de OLSR frente a otros protocolos proactivos es que OLSR minimiza tanto el tamaño de los mensajes de encaminamiento como el número de nodos que reenvían estos mensajes por *broadcast* a la red, de tal forma que la sobrecarga introducida en la red debido a las actualizaciones periódicas de rutas es pequeña. Para ello, sólo ciertos nodos de la red son los encargados de reenviar la información de encaminamiento y responsables de que esta información llegue al resto de nodos de la red. Esto permite que OLSR sea un protocolo óptimo para operar en redes grandes y densas. Además OLSR, puede reaccionar rápidamente ante cambios en la topología de la red y por eso los nodos pueden elegir la mejor ruta en cada instante, dependiendo siempre de la frecuencia con la que se actualicen rutas. Otra de las ventajas del protocolo OLSR es que al enviar periódicamente los mensajes de control para actualización de rutas, puede aceptarse una pérdida razonable de estos mensajes, pérdidas que son frecuentes en las redes MANET debido a la movilidad de los nodos y a las características inherentes al medio inalámbrico.

El funcionamiento básico del protocolo OLSR es el mismo para ambas versiones del protocolo. Las particularidades de la versión 1 y de la versión 2 las describiremos en secciones aparte, así como un resumen con las diferencias entre ambas.

3.3.1. Funcionamiento básico del protocolo

Cuando un nodo entra a formar parte de una red MANET lo primero que hace es detectar con qué otros nodos vecinos tiene conexión a nivel de enlace. Entendiendo por conexión, las interfaces de red que tenga OLSR. Para ello los nodos intercambian periódicamente mensajes *Hello*. Los mensajes *Hello* no se

retransmiten por toda la red. Su objetivo es que los nodos conozcan a otros nodos con los que tienen conectividad a nivel de enlace, es decir, que conozcan y descubran a sus nodos vecinos. Además, los mensajes *Hello* anuncian los vecinos ya conocidos por el nodo que envía el mensaje *Hello*. De esta forma los nodos que están a un salto de distancia de este nodo emisor, no sólo conocen a su vecino sino que también conocen a los vecinos de su vecino, es decir, a los nodos que están a dos saltos de distancia.

Como ya hemos visto inundar periódicamente la red de paquetes de control para actualizar rutas congestiona la red. Para evitar esto, OLSR utiliza la técnica de *MultiPoint Relay* (MPR). Una vez que un nodo conoce todos los nodos que se encuentran a dos saltos de distancia de él, selecciona de entre sus nodos vecinos a un salto, aquellos nodos que le proporcionen acceso a todos los nodos a dos saltos. A este grupo de nodos se le conoce como nodos MPR. Se informa a los nodos MPR sobre qué nodos los han elegido como tales en una estructura que se llama *selector set MPR*. Una de las responsabilidades de los nodos MPR es retransmitir los mensajes de difusión que generen los nodos selectores de tal forma que esta información se difunda a toda la red. Otra de las responsabilidades de los nodos MPR es generar y retransmitir mensajes *Topology Control* (TC). Los mensajes TC también se generan de forma periódica y contienen una lista con las direcciones de todos los nodos que han elegido al nodo como nodo MPR. Es decir, la información de los mensajes TC no anuncia a cualquier nodo vecino del nodo MPR, sino únicamente a los nodos que lo han elegido como MPR. De esta forma se consigue minimizar la información sobre la topología que viaja por la red. Esta información sobre la topología de la red únicamente la reenvían y la procesan los nodos MPR. No se inunda la red con mensajes de control, sino que la diseminación de estos mensajes por la red se hace de forma controlada.

Los nodos de la red MANET tienen la posibilidad de tener más de una interfaz de red. Cada dirección de interfaz está asociada con una interfaz principal, única para cada uno.

3.3.2. OLSR versión 1

A continuación vamos a detallar el funcionamiento de la versión 1 del protocolo de encaminamiento OLSR definido en la RFC3626 [Clausen et al, 2003].

Tablas en OLSRv1

Cada nodo en la red MANET que emplea OLSRv1 almacena información que recoge de la red gracias al intercambio de paquetes de control. Esta información se

guarda en los repositorios: *Multiple Interface Association Information Base*, *Local Link Information Base*, *Neighborhood Information Base*, *MPR Set* y *Topology Information Base*. Vamos a ver en detalle cada uno de ellos:

- *Multiple Interface Association Information Base*. Por cada nodo destino en la red, se almacenan tuplas de las direcciones de las interfaces (*Interface Association Tuples*).

- *Local Link Information Base*. Se almacena información sobre los enlaces hacia los vecinos. Cada nodo almacena en una estructura llamada *Link Set* la información del estado de sus conexiones a nivel de enlace con sus nodos vecinos.

- *Neighborhood Information Base*. En este caso se almacena información sobre los vecinos a un salto, los vecinos a dos saltos, los MPRs y los nodos selectores de MPRs. Respecto a los vecinos a un salto con los que tiene conexión, cada nodo almacena, un conjunto de tuplas de vecinos (*neighbour tuples*) con la siguiente información:

  - **N_neigbour_main_addr**: Dirección principal del nodo vecino.
  - **N_status**: Estado de la conexión (simétrica o asimétrica).
  - **N_willingness**: Entero entre 0 y 7 que representa la intención del nodo vecino de retransmitir tráfico de otros nodos.

  Respecto a los vecinos a dos saltos con los que tiene conexión, cada nodo almacena, un conjunto de tuplas de vecinos a dos saltos (*2-hop tuples*) donde guarda información sobre los enlaces simétricos entre los nodos vecinos y los nodos vecinos a dos saltos.

  Respecto al conjunto MPR, cada nodo mantiene un conjunto de vecinos que han sido seleccionados como MPR. También mantiene el conjunto de nodos que han seleccionado al nodo como MPR.

- *Topology Information Base*. Cada nodo mantiene información sobre la topología de la red. Esta información se obtiene de los mensajes TC y se utiliza para el cálculo de rutas. Cada entrada de esta tabla contiene las tuplas: *T_dest_addr* que es la dirección principal del nodo destino que es alcanzado en un salto por el nodo con dirección principal *T_last_ addr*. *T_last_ addr* es el nodo MPR del nodo *T_dest_addr*. También contiene el parámetro *T_seq* con el número de secuencia y un *T_time* con el tiempo de expiración de la tupla.

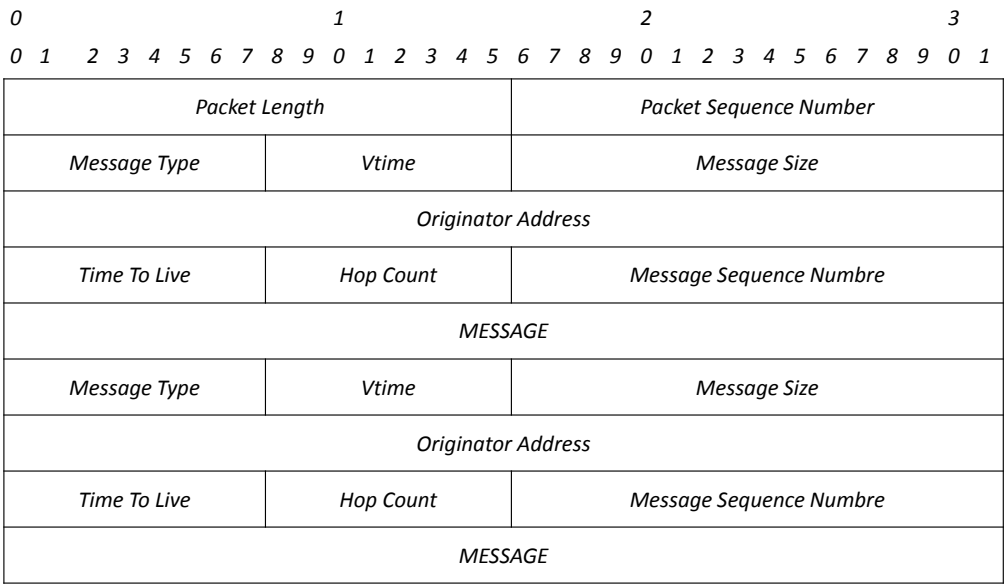

Figura 3.2: Formato de un paquete del protocolo OLSRv1

Formato del paquete OLSRv1

La primera versión del protocolo OLSR utiliza un formato de paquete común para todos los mensajes que intercambian los nodos que integran la red.

Los paquetes se transmiten encapsulados en datagramas *User Datagram Protocol* (UDP) utilizando el puerto 698 asignado por el *Internet Assigned Numbers Authority* (IANA). Cada paquete puede encapsular a su vez uno a varios mensajes. Todos los mensajes comparten una cabecera común al formato del paquete lo cual permite que los nodos de la red puedan aceptar y retransmitir mensajes de un tipo desconocido o nuevo para ellos.

El formato del paquete OLSR versión 1 se muestra en la figura 3.2 (omitiendo las cabeceras IP y UDP).

Los campos correspondientes a la cabecera OLSRv1 son: *packet lengh* y *packet sequence number*. El primero define el tamaño del paquete OLSR en bytes. El segundo define el número de secuencia del paquete. Este número de secuencia se incrementa en uno por cada paquete OLSR que se transmite en la red.

Los campos correspondientes a la cabecera de cada mensaje son los siguientes:

- *Message Type*. Indica el tipo de mensaje que se encuentra en el campo

MESSAGE.

- *Vtime*. Indica el tiempo en el que la información del mensaje es válida.
- *Message Size*. Indica el tamaño del mensaje en bytes, incluyendo la cabecera y el campo MESSAGE.
- *Originator Address*. Indica la dirección principal del nodo origen del mensaje.
- *Time To Live* (TTL). Indica el número máximo de saltos permitidos para retransmitir el mensaje. Los mensajes se pueden transmitir a toda la red o se pueden limitar a los nodos que se encuentren a un número de saltos del nodo origen.
- *Hop Count*. Número de saltos que da el mensaje. Se incrementa en uno en cada retransmisión.
- *Message Sequence Number*. Número de secuencia del mensaje que genera un nodo y luego se retransmite en la red. Sirve de indentificador del mensaje.

Mensajes *Hello* y descubrimiento de vecinos en OLSRv1

OLSR utiliza los mensajes *Hello* para descubrir a los nodos vecinos y el estado de la red a nivel de enlace. Los mensajes *Hello* se envían como datos dentro del paquete general OLSR descrito anteriormente, configurando el campo *Message Type* con el valor *Hello_Message* y el campo *TTL* con el valor 1. Estos mensajes no se retransmiten a toda la red.

A continuación se detallan los campos del mensaje *Hello* que se describe en la figura 3.3.

- *Reserved*. Son 6 bits que se establecen con el valor 0x0000.
- *HTime*. Este campo indica la periodicidad con la que se envían mensajes *Hello*.
- *Willingness*. Indica la disponibilidad que tiene un nodo para ser elegido como nodo MPR y poder reenviar tráfico a otros nodos.
- *Link Code*. Este campo indica el tipo de enlace que hay entre el nodo que envía el mensaje y el listado de nodos vecinos. Este tipo de enlace puede ser: *enlace asimétrico*, cuando no es posible establecer un enlace bidireccional

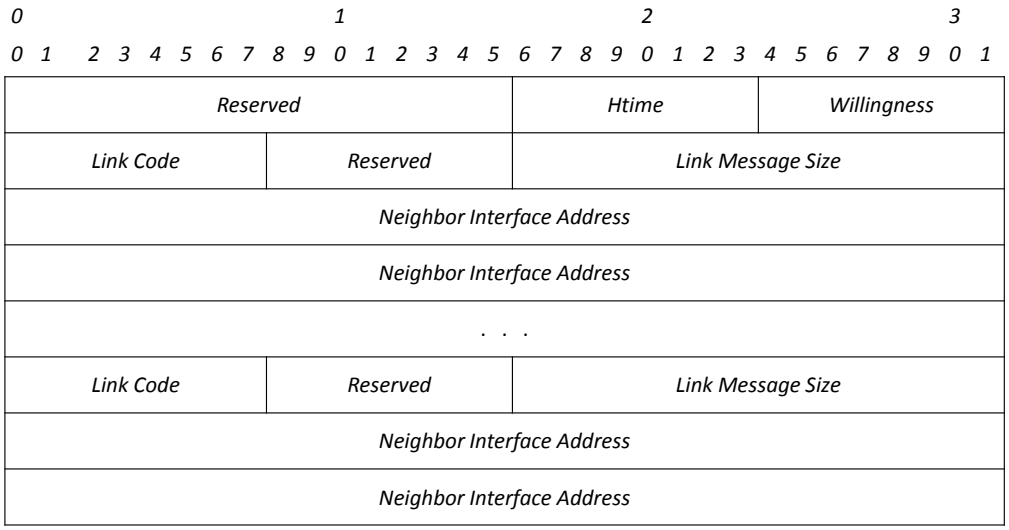

Figura 3.3: Formato del mensaje *Hello* de OLSRv1

con el nodo vecino; *enlace simétrico*, cuando existe un enlace bidireccional y *enlace MPR*, cuando los nodos definidos en la lista han sido seleccionados por el nodo emisor del mensaje como MPR.

- *Reserved*. 8 bits que se ponen a 0.
- *Link Message Size*. Este campo define el tamaño del mensaje en bytes.
- *Neigbor Interface Address*. Este campo define la lista de vecinos que se han etiquetado con un *Link Code* en particular.

Para que la conexión entre dos nodos vecinos sea válida el mensaje *Hello* se ha tenido que recibir en ambos sentidos.

Multipoint Relays (MPR)

Cada nodo de la red selecciona su conjunto de nodos MPR de entre sus vecinos a un salto. En el caso de la figura 3.4 el nodo N selecciona los nodos 1, 2, 4 y 5 como nodos MPR ya que a través de ellos puede llegar a todos los vecinos a dos saltos de distancia. A este conjunto de nodos se le conoce como conjunto de MPRs seleccionados del nodo N. Para realizar esta selección no existe ningún

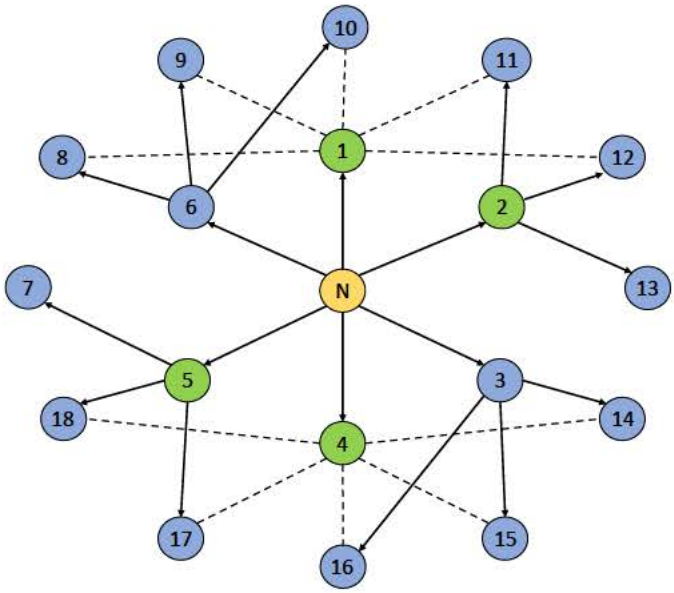

Figura 3.4: Selección de nodos MPR

método establecido. El único requisito es que el conjunto MPR tiene que ser suficientemente grande para que el nodo N pueda alcanzar todos los vecinos a dos saltos. El resto de nodos que se encuentran a un salto de distancia del nodo N y que no han sido seleccionados como MPR reciben y procesan los mensajes de difusión del nodo N pero no los retransmiten. Sólo retransmiten estos mensajes el conjunto de nodos MPR. De esta forma se consigue llegar a todos los nodos de la red sin necesidad de que todos los nodos reenvíen mensajes y por lo tanto se consigue reducir el tráfico de control de la red. En la figura 3.5 se muestra un ejemplo donde todos los nodos de la red retransmiten mensajes de control y otro ejemplo donde se utiliza la técnica MPR. Cuanto menor sea el número de nodos seleccionados como MPR menor será la sobrecarga que se introduzca en la red. El conjunto de MPRs se calcula cada vez que se detecta un cambio en la topología de la red.

Hay que destacar que el conjunto de MPRs seleccionados por un nodo se reporta en los mensajes *Hello* enviados por este nodo. De esta forma se puede construir el *selector set MPR* que son aquellos nodos que han seleccionado a uno en particular como MPR.

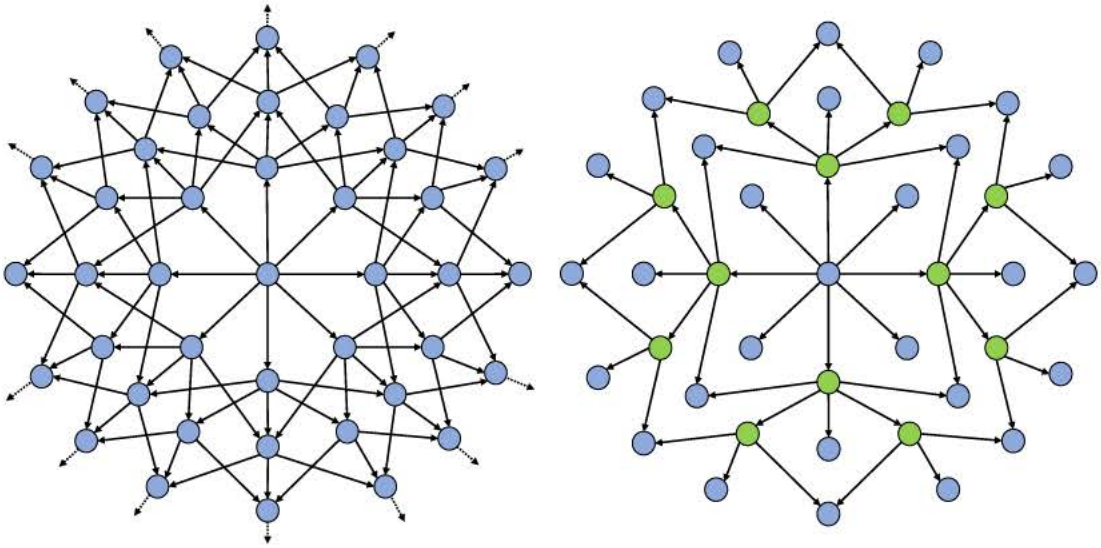

Figura 3.5: *Regular flooding vs MPR flooding*

Mensaje TC

Los mensajes *Topology Control* TC son generados por los nodos que han sido elegidos como nodos MPR. Cada nodo no necesita informar de todos sus vecinos sino únicamente de su conjunto de selectores MPR. De esta forma, los nodos que no son MPR no necesitan difundir los mensajes TC. Los mensajes TC se envían periódicamente, aunque si se detecta alguna modificación en el conjunto de selectores MPR hay que enviar un mensaje TC inmediatamente con los cambios.

Los mensajes TC tienen el formato que se muestra en la figura 3.6.

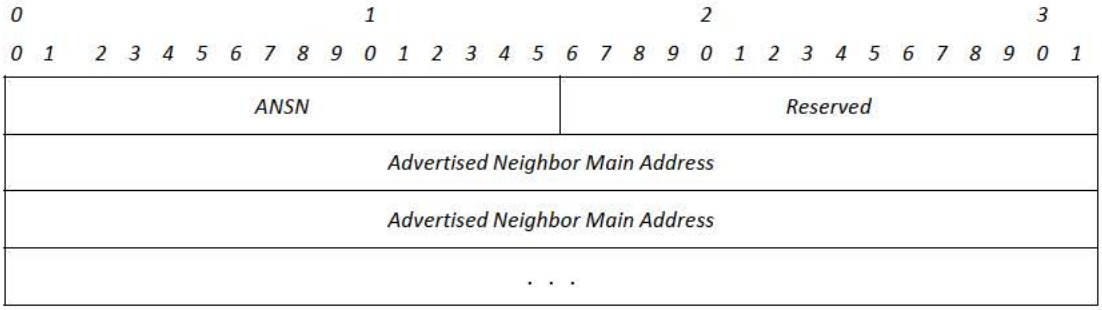

Figura 3.6: Formato del mensaje TC de OLSRv1

- *Advertised Neighbour Sequence Number* (ANSN). 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 vecinos, envía otro mensaje TC incrementando este número. De esta forma los nodos que reciben este mensaje saben si la información es más reciente que la que tienen.

- *Reserved*. Campo de 16 bits reservado.

- *Advertised Neighbour Main Address*. Campo que contiene la dirección principal de un nodo vecino.

Este mensaje se envía dentro del campo de datos del paquete OLSR, configurando el campo *Message Type* con el valor *TC_Message* y el campo *TTL* con el valor máximo (255).

Cuando un nodo recibe un mensaje TC realiza los siguientes pasos:

- Si existe alguna tupla en la tabla de topología donde *T_last_addr* sea igual a la dirección del nodo MPR origen del mensaje y exista un *T_seq* mayor a ANSN, el mensaje se descarta puesto que se trata de un mensaje viejo.

- Todas las tuplas de la tabla de topología donde *T_last_addr* sea igual a la dirección de origen del mensaje y exista un *T_seq* menor al ANSN del mensaje son eliminadas.

- Para cada una de las entradas de la tabla de topología donde *T_dest_addr* sea igual a las direcciones informadas en el mensaje TC y *T_last_addr* sea igual a la dirección de origen del mensaje, se actualiza el tiempo de vida de la tupla en la tabla de topología.

- El resto de las direcciones informadas en el mensaje TC que no estaban en la tabla de topología, se agregan a la tabla de topología.

Cálculo de las tablas de rutas

Cada nodo mantiene actualizada una tabla con rutas con la cual puede llegar a cualquier nodo de la red y enviar datos hacia cualquier destino. Esta tabla se basa en la información que se obtiene de la tabla de enlaces y la tabla de topología de la red. Cualquier modificación en alguna de estas tablas supone un nuevo cálculo de todas las rutas.

El formato de entradas de esta tabla es el siguiente:

- *R_dest_addr, R_next_addr, R_dist, R_iface_addr*
- *R_dest_addr, R_next_addr, R_dist, R_iface_addr*

R_dest_addr es la dirección principal de un nodo que se encuentra a una distancia de *R_dist saltos* del nodo actual. *R_next_addr* es el nodo vecino que tiene un enlace simétrico con el nodo actual y es el siguiente salto en la ruta hacia *R_dest_addr*. Este nodo es alcanzable a través de la interfaz local con dirección *R_iface_addr*.

Existe una entrada en esta tabla por cada nodo de la red para el que existe una ruta conocida. Para construir una ruta se utiliza el algoritmo del camino más corto (*Shortest Path Algorithm*).

Esta tabla se actualiza cuando existe algún cambio en la topología de la red porque exista un cambio en la lista de vecinos y también cuando alguna de las entradas de las tablas expira con lo que es necesario recalcular la tabla de rutas.

3.3.3. OLSR versión 2

A continuación vamos a detallar el funcionamiento de la versión 2 del protocolo de encaminamiento OLSR definido en la RFC7181 [Clausen et al, 2014].

Tablas en OLSRv2

OLSRv2 define los llamados *repositorios de información* que consisten en una serie de tablas de información donde el protocolo guarda información de estado.

- *Interface Information Base*. Cada nodo mantiene esta tabla por cada una de sus interfaces de red. Guarda información de los enlaces que se pueden alcanzar a uno y dos saltos por cada interfaz. Por cada entrada guarda la siguiente información:
  - *Link Set*. Se guarda la información relativa a los enlaces de otros nodos que son o han sido vecinos a un salto de distancia.
  - *2-Hop Set*. Se guarda la información de las direcciones de red de los vecinos a dos saltos de distancia y de las direcciones de red de los vecinos a un salto de distancia a través de los cuales se pueden alcanzar los vecinos a dos saltos. Se guarda: la dirección IP de la interfaz del nodo, la dirección IP del nodo vecino a través del cual alcanzo al

situado a dos saltos, dirección IP del nodo situado a dos saltos, el tiempo en el que la tupla expira y ha de eliminarse.

- *Neighbor Information Base*. Esta tabla se mantiene por cada nodo de la red. Contiene información relativa a las direcciones de red de los vecinos situados a un salto de distancia. Está presente en el protocolo NHDP y en el protocolo OLSRv2. Guarda la siguiente información:

  - *Neighbor Set*. Se guardan las direcciones IP de todos los nodos vecinos a un salto de distancia.
  - *Lost Neighbor Set*. Se guardan las direcciones IP de los nodos que fueron vecinos a un salto de distancia pero con los que ahora se ha perdido el enlace.

  Además de esta información definida en la RFC 6130, el protocolo OLSRv2 añade los siguientes campos:

  - *N_orig_addr*. Es la dirección del nodo vecino.
  - *N_in_metric*. Es la métrica de cualquier enlace del nodo vecino hacia cualquiera de las interfaces del nodo.
  - *N_out_metric*. Es la métrica de cualquier enlace de las interfaces del nodo hacia el nodo vecino.
  - *N_will_flooding*. Define la predisposición del nodo vecino a ser seleccionado como nodo MPR.
  - *N_flooding_mpr*. Flag que indica si el nodo vecino ha sido seleccionado como MPR por el nodo.
  - *N_mpr_selector*. Flag que indica si el nodo vecino ha seleccionado al nodo como MPR.
  - *Advertised Neighbor Sequence Number (ANSN)*. Sirve para comprobar si la información que transmiten los mensajes es reciente. Este número aumenta cuando la información de los mensajes TC se actualiza.

- *Topology Information Base*. Esta tabla se mantiene por cada nodo de la red. Se actualiza con la información que cada nodo procesa de los mensajes TC. Guarda información de la topología de la red:

  - *Advertising Remote Router Set*. Guarda cada nodo del que se han recibido mensajes TC. De esta forma se puede determinar si un mensaje TC contiene información actualizada o no.

- *Router Topology Set* Guarda la lista de enlaces existentes entre pares de nodos de la red. Guarda: la dirección destino, la dirección a partir de la cual se alcanza el destino, número de secuencia del link, y el tiempo en el que la tupla expira y ha de eliminarse.
- *Routable Address topology Set*. Se guarda información topológica sobre las rutas de la red, incluye información sobre a través de qué nodos alcanzar un destino. Guarda: la dirección origen de un nodo que puede alcanzar la dirección destino en un salto y el tiempo en el que la tupla expira y ha de eliminarse.
- *Routing Set*. Guarda información de rutas que utilizará el nodo para encaminar los paquetes. Guarda: la interfaz destino, la interfaz a la que hay que mandar el paquete para llegar al destino, interfaz propia por la que hay que enviar el paquete para que llegue al destino.

- *Received Message Information Base*. Cada nodo guarda información sobre los mensajes TC que recibe en cada interfaz, los mensajes TC que procesa y los mensajes TC que reenvía hacia otros nodos, con el fin de que no se procesen o se retransmitan por error dos veces. También guarda los mensajes fragmentados recibidos, en espera de recibir el resto de mensajes para procesarlos. Los mensajes recibidos, procesados y retransmitidos se guardan en *Received Set*, *Processed Set* y *Forwarded Set*. Para cada uno de ellos se guarda: tipo de mensaje recibido/procesado/retransmitido, dirección origen del mensaje, número de secuencia del mensaje, tiempo en el que la tupla expira y ha de eliminarse.

Formato del paquete OLSRv2

La segunda versión del protocolo OLSR, OLSRv2 presenta una arquitectura más modular y flexible que OLSRv1. OLSRv2 utiliza el formato genérico definido por el grupo MANET para el envío de los paquetes de control de cualquier cualquier protocolo para redes MANET, *Generalized MANET Packet/Message Format* definido en la RFC 5444 [Clausen et al, 2015] y *Type Length Value* (TLV) [Clausen et al, 2009] estandarizados en la RFC 5497.

En las figuras 3.7 y 3.8 vemos gráficamente el formato de paquete que utiliza OLSRv2. Este formato permite que un paquete OLSRv2 pueda integrar uno o varios mensajes. Cada mensaje a su vez contiene una cabecera donde se identifica el tipo de mensaje y un cuerpo del mensaje. Esto también es así en la primera versión de OLSR. La diferencia es que en este caso OLSRv2 ha adoptado un formato de paquete/mensaje estandarizado y no propietario del protocolo OLSR.

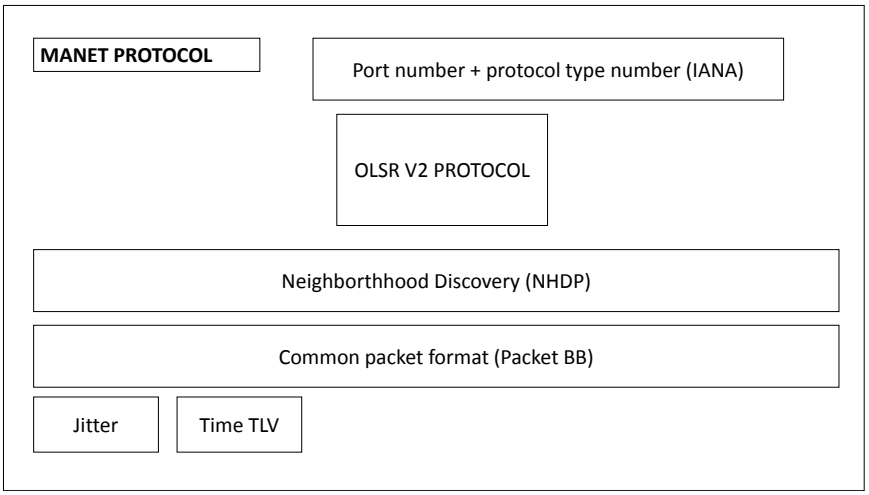

Figura 3.7: Estructura del protocolo de encaminamiento OLSRv2

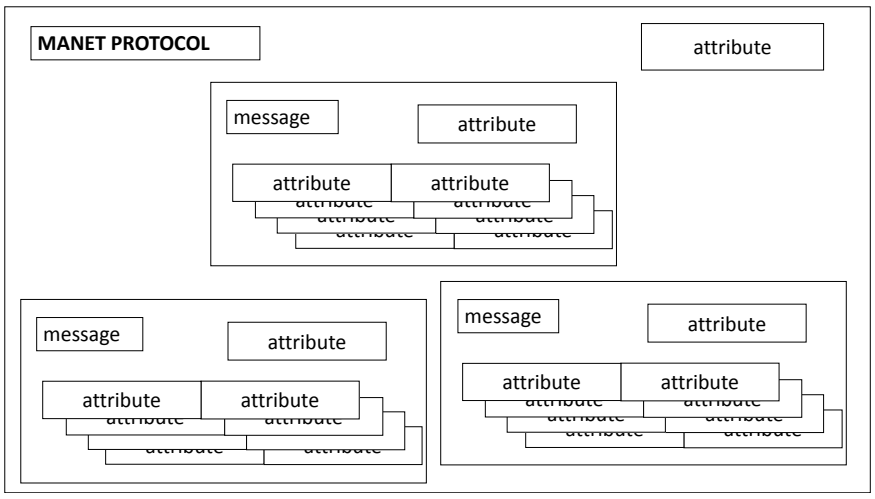

Figura 3.8: Formato genérico del Paquete/Mensaje de una red MANET

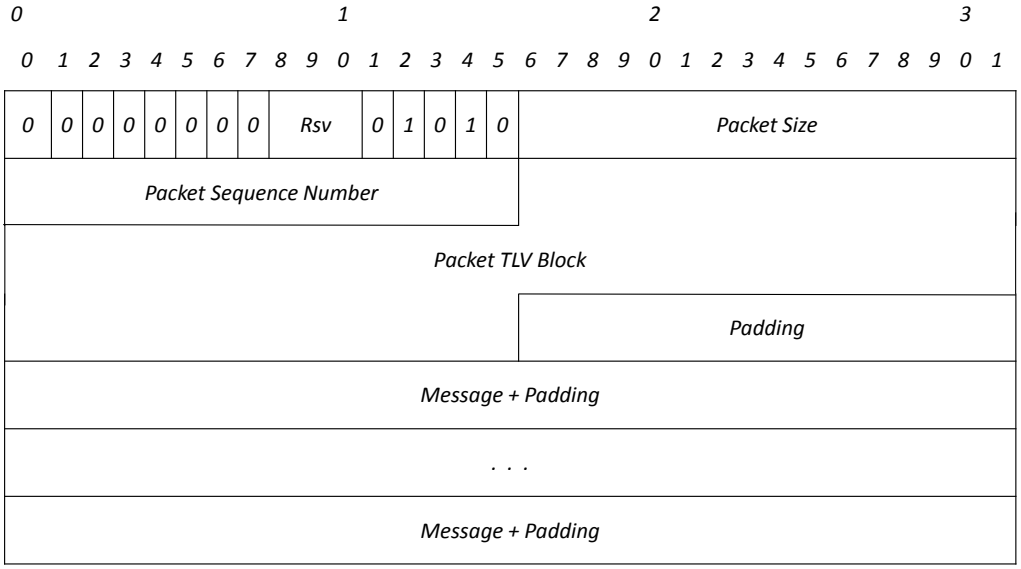

Figura 3.9: Formato de un paquete genérico en una red MANET

Tal como se muestra en la figura 3.9, cada paquete OLSRv2 consta de una cabecera y un campo de mensaje.

La cabecera del paquete OLSR contiene:

- Versión: son 4 bits que definen la versión. En la versión actual del protocolo los 4 bits están puestos a 0.
- *Flags*: son también 4 bits que especifican qué va a contener el resto de la cabecera del paquete: Si el bit 0 es un 0, el número de secuencia del paquete no se incluye en la cabecera. Si el bit 1 es 0 no se incluyen bloques TLV en la cabecera. Los bits 2 y 3 están reservados y deben estar a 0 cuando se transmita un paquete.
- Número de secuencia del paquete: Especifica el número de secuencia del paquete.
- Bloque TLV.

Cada mensaje consta a su vez de una cabecera y un bloque TLV tal como se muestra en la figura 3.10.

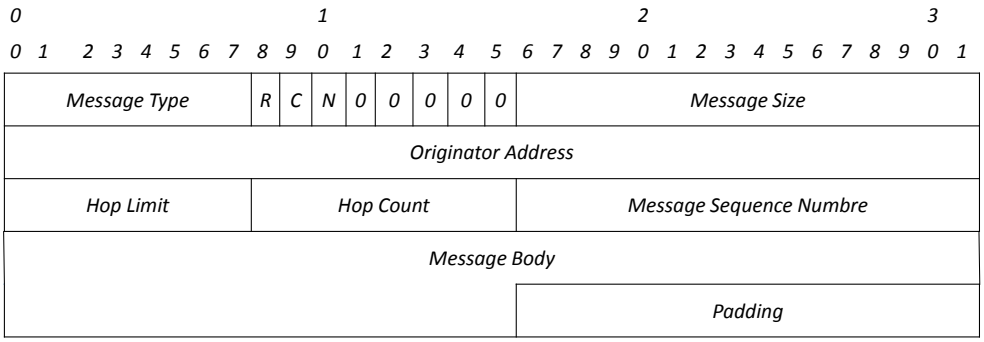

Figura 3.10: Formato de un mensaje genérico en una red MANET

OLSRv2 utiliza dos tipos de mensajes: mensajes *Hello* y mensajes TC. Los mensajes *Hello* se definen dentro del protocolo NHDP [Clausen et al, 2011] de la RFC 6130 y los mensajes TC se definen en la propia RFC de OLSRv2.

La cabecera del mensaje OLSRv2 contiene:

- Tipo de mensaje: Especifica el tipo de mensaje.
- *Flags*: Son 4 bits que definen el resto de la cabecera del mensaje. Si el bit 0 es 0 el campo con la dirección origen no se incluye. Si el bit 1 es 0 el campo *hop-limit* no se incluye. Si el bit 2 es 0 el campo *hop-count* no se incluye. Si el bit 3 es 0 el campo número de secuencia no se incluye.
- Longitud de la dirección del mensaje. Indica la longitud de la dirección en octetos. Para direcciones IPv4 es un 3 y para direcciones IPv6 es un 15.
- Tamaño del mensaje. Número de octetos que componen el mensaje incluyendo la cabecera del mensaje.
- Dirección origen. Dirección origen del nodo que envía el mensaje.
- *Hop limit*. Número máximo de saltos que a un paquete le está permitido ser retransmitido.
- *Hop count*. Número de saltos que un paquete realiza.
- *Sequence number*. Número de secuencia que identifica el mensaje.

OLSRv2 utiliza dos tipos de atributos TLVs para los mensajes:

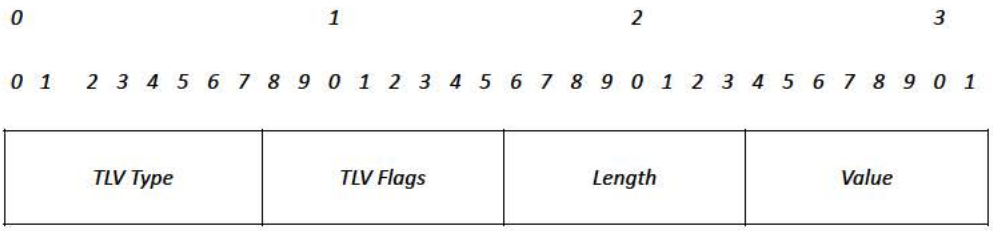

Figura 3.11: Formato estructura TLV

- MPR_WILLING. Se utiliza en los mensajes *Hello*. Indica la predisposición que un nodo tiene a ser MPR.
- CONT_SEQ_NUM. Se utiliza en los mensajes TC. Se corresponde con el ANSN almacenado en el *Neighbour Information Base*.

Estructura TLV

Una estructura TLV añade a un paquete o un mensaje un atributo. La interpretación y el procesamiento de los atributos TLV se tienen que hacer dentro del protocolo de encaminamiento de la red MANET que los utilice. El formato en el que se define cualquier estructura TLV sigue las recomendaciones de la RFC 5444. Este formato se muestra en la figura 3.11.

Cada atributo TLV se identifica por los siguientes campos:

- *TLV Type*. Es un campo de 8 bits que especifica el tipo de TLV: TLV asociado a un paquete, TLV asociado a un mensaje.
- *TLV Flags*. Es un campo de 8 bits que identifica el resto de campos de TLV.
- *Lenght*. Longitud del TLV.
- *Value*. Valor del TLV.

Los TLVs se agrupan en bloques TLV. El formato de un bloque TLV se muestra en la figura 3.12.

- *TLV Lenght*. Es un campo de 16 bits que contiene el número total de octetos de los TLVs que contiene el bloque.
- *TLVs*. Atributo TLV que se define como hemos visto en la figura 3.11.

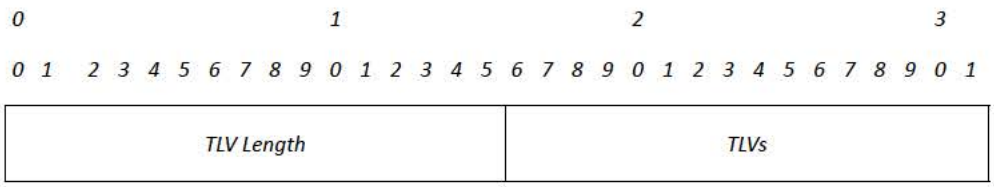

Figura 3.12: Formato bloque TLV

Mensajes *Hello* y descubrimiento de vecinos

Los mensajes *Hello* se generan, se transmiten, se reciben y se procesan por el protocolo NHDP siguiendo la RFC 6130. El protocolo NHDP es un protocolo genérico definido para redes MANET para descubrir nodos vecinos a uno y dos saltos de distancia.

Una de las características comunes a los protocolos proactivos es la necesidad de que un nodo conozca sus nodos vecinos. Los protocolos de encaminamiento previos a la definición del protocolo NHDP habían adoptado cada uno de ellos su propia solución para determinar el conjunto de vecinos de un nodo, aunque todos ellos compartían ciertas características comunes. Este era el caso de la versión 1 de OLSR. Siguiendo la tendencia de aunar esfuerzos y proponer soluciones globales, el grupo MANET ha definido el protocolo NHDP. Este protocolo se basa en la solución adoptada por el protocolo OLSR versión 1, con lo que además de conocer vecinos a un salto, NHDP es capaz de detectar vecinos a dos saltos. Al igual que ocurría en la versión 1 del protocolo OLSR, NHDP se basa en la difusión de mensajes *Hello*. Los mensajes *Hello* se transmiten también periódicamente y como consecuencia de un cambio topológico en la red o como una combinación de ambas estrategias.

El protocolo NHDP realiza las siguientes funcionalidades:

- Anuncia la presencia de un nodo y de todas sus interfaces.
- Descubre enlaces a nodos vecinos.
- Comprueba la bidireccionalidad de los enlaces a nodos vecinos.
- Anuncia los enlaces encontrados en los mensajes *Hello*, de tal forma que se pueden descubrir también los nodos vecinos a dos saltos de distancia.
- Guarda una base de información con todos los enlaces descubiertos, los vecinos a un salto y los vecinos a dos saltos de distancia.

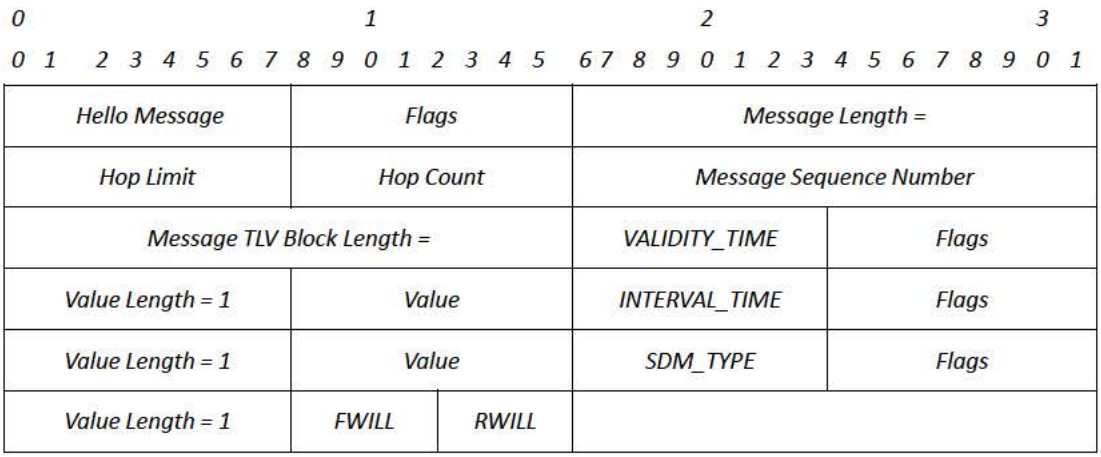

Figura 3.13: Formato de un mensaje *Hello* en OLSRv2

El formato del mensaje *Hello* sigue la RFC 5444, ver figura 3.13. Se utilizan dos tipos de atributos TLVs para los mensajes *Hello*: VALIDITY_TIME e INTERVAL_TIME definidos en la RFC 5497. El primero define el tiempo durante el cual la información que contiene el mensaje es válida. El segundo, indica el tiempo máximo que transcurrirá hasta que se reciba otro mensaje igual procedente del mismo nodo origen.

Para el caso del protocolo OLSRv2 a la definición de mensajes *Hello* hecha en la RFC 6130 se le añaden los siguientes elementos:

- La dirección del nodo origen del mensaje.
- El atributo TLV MPR_WILLING con el que cada nodo indica lo predispuesto que está a actuar como un nodo MPR en la red.
- Al mensaje *Hello* también se le añaden extensiones para permitir métricas de enlace para anunciar enlaces con el resto de nodos de la red y para indicar qué tipo de métrica de enlace se está usando.

Los mensajes *Hello* se transmisten en la red por el protocolo NHDP periódicamente y cada vez que el conjunto de nodos MPRs cambia.

Mensaje TC

Los mensajes TC se definen en el protocolo OLSRv2. Se generan a partir de la información que se guarda en las tablas que almacena cada nodo. Se reenvían

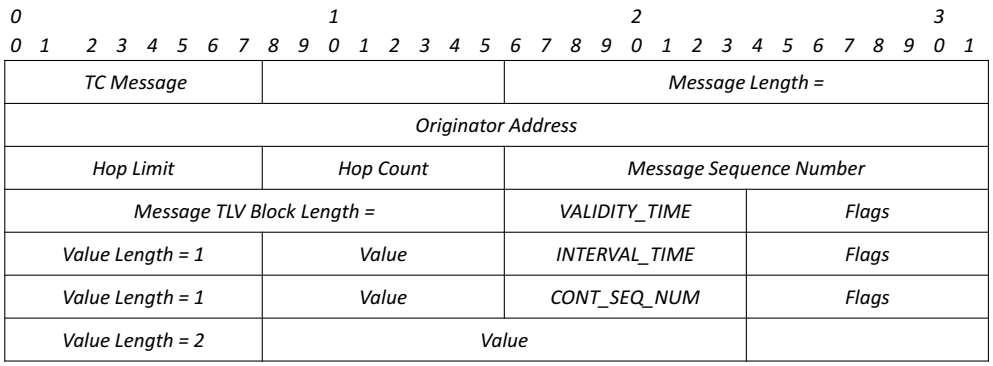

Figura 3.14: Formato de un mensaje TC en OLSRv2

por los nodos MPR periódicamente de forma proactiva por todas las interfaces del nodo. La periodicidad con la que se envían estos mensajes puede ser fija o dinámica en función de ciertos parámetros de la red como por ejemplo la congestión. Además de enviarse periódicamente los mensajes TC también se envían cuando se detecta un cambio en la topología de la red.

La información que procesan los nodos de los mensajes TC se utiliza para actualizar la información del repositorio *Topology Information Base*.

Como se muestra en la figura 3.14 un mensaje TC contiene los siguientes campos:

- *Message Originator Address*. Dirección del nodo origen.
- *Msg_seg_num*. Número de secuencia del mensaje que se retransmite.
- *Msg_hop_limit*. Límite en el número de saltos del mensaje que se retransmite.
- *Msg_hop_count*. En el caso en que el mensaje TC contenga cualquiera de los atributos TLV VALIDITY_TIME y/o INTERVAL_TIME este campo valdrá cero.
- *Msg_tlv_CONT_SEQ_NUM*. Se corresponde con el ANSN de la tabla *Neighbor Information Base*. Sirve para comprobar si la información que transmiten los mensajes es reciente. Este número aumenta cuando la información de los mensajes TC se actualiza.

Cálculo de las rutas OLSRv2

El conjunto de rutas de un nodo representa los caminos desde ese nodo a todos los nodos destino de la red. Estos caminos se calculan en base al gráfico de la topología de la red que se construye a su vez en base a la información que se obtiene de los mensajes *Hello* y TC y que se guarda en los repositorios de OLSRv2.

Si existen cambios en el conjunto de rutas de un nodo éstas no se retransmiten por la red. Sí que se tienen que reflejar estos cambios en la tabla de rutas IP.

Para construir una ruta no se utiliza el camino más corto como en OLSRv1 sino la mejor métrica de enlace posible.

3.3.4. Diferencias entre OLSRv1 y OLSRv2

En esta sección vamos a resumir las principales diferencias entre ambas versiones del protocolo OLSR.

- En OLSRv1 los mensajes TC y *Hello* comparten una cabecera de paquete común definida en OLSRv1. OLSRv2, sin embargo, utiliza el Generalized MANET Packet/Message Format definido en la RFC 5444 y los TLVs definidos en la RFC 5497.
- OLSRv1 utiliza los mensajes *Hello* para descubrir sus nodos vecinos. OLSRv2 utiliza el protocolo NHDP para descubrir vecinos.
- OLSRv2 elige la mejor ruta hacia un destino en función de una métrica del enlace, mientras que OLSRv1 utiliza la métrica del número de saltos (no siempre las rutas con menos saltos son las más idóneas).
- OLSRv2 incluye extensiones para seguridad.

Parte III

Contribuciones

En la parte III describimos y evaluamos el mecanismo de descubrimiento de servicios integrado sobre las dos versiones del protocolo de encaminamiento OLSR que hemos diseñado.

En el capítulo 4 nos centramos en el mecanismo de descubrimiento de servicios definido sobre OLSRv1. Realizamos un diseño razonado de nuestra propuesta y describimos cómo la hemos implementado sobre el simulador de red NS-2 para finalmente detallar las prestaciones que hemos evaluado.

En el capítulo 5 realizamos lo mismo pero para la versión del mecanismo de descubrimiento de servicios que hemos evolucionado aprovechando la evolución de la versión 1 a la versión 2 del protocolo de encaminamiento OLSR.

Por último, finalizamos la tesis con un capítulo de conclusiones, capítulo 6, donde reflejamos las principales conclusiones que hemos sacado del trabajo realizado y las posibles líneas futuras sobre las que continuar investigando.

Capítulo 4

Diseño y evaluación del protocolo SD-OLSRv1

Después del estudio realizado en el capítulo 2, hemos visto que las soluciones basadas en un diseño *cross-layer* referentes al descubrimiento de servicios, disminuyen considerablemente el ancho de banda consumido en la red y la cantidad de mensajes transmitidos [García-Macías et al, 2005], [Halkes et al, 2006] y [Radhamani et al, 2011]. Por ello en esta tesis, nos planteamos definir un nuevo mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSR, que hemos denominado *Service Discovery over OLSR* (SD-OLSR) que permita descubrir el máximo número de servicios posibles sin que esto sobrecargue en exceso la red.

SD-OLSR es un mecanismo de descubrimiento de servicios tipo *push*, integrado en el protocolo de encaminamiento OLSR que presenta buenas prestaciones atendiendo a los criterios de tasa de descubrimiento de servicios y tiempo en descubrir un servicio. Aprovechamos la naturaleza proactiva del protocolo OLSR y también el método de reenvío de paquetes mediante nodos MPR, propio de OLSR, para anunciar periódicamente los servicios disponibles en cada nodo. El mecanismo ha sido adaptado a las dos versiones del protocolo OLSR estandarizadas.

SD-OLSRv1 es un mecanismo de descubrimiento de servicios totalmente distribuido, cuyo funcionamiento no depende de directorios. Cada nodo anuncia periódicamente los servicios que tiene disponibles mediante mensajes que se integran en el paquete OLSR. Estos mensajes se transmiten por difusión a través de los nodos MPR, de tal forma que los anuncios de servicios llegan a toda la red, y cualquier nodo puede conocer la existencia del servicio anunciado. Dependiendo de la versión del protocolo SD-OLSR definida, cada nodo puede almacenar local-

mente una o dos cachés donde se guardan los servicios que él anuncia y los que va escuchando en la red. La información almacenada en estas cachés se consulta cada vez que un nodo solicita un servicio, para comprobar si el servicio está disponible o no en la red. Inicialmente, la versión 1 del mecanismo definido para OLSRv1, permitía también que un nodo que buscara un servicio pudiera enviar una petición de búsqueda del mismo, según el método distribuido de tipo *pull*, cuando no encontrara ese servicio en su caché.

La utilización del formato del paquete definido en OLSRv1 para encapsular los mensajes de anuncio de servicios, hace posible el correcto funcionamiento del mecanismo de descubrimiento de servicios aunque no todos los nodos que forman parte de la red MANET lo tengan implementado.

Una vez descritas las principales características del mecanismo de descubrimiento de servicios diseñado, en la sección 4.1 realizaremos una explicación formal de SD-OLSRv1. A continuación, en la sección 4.2 describimos la implementación del mecanismo de descubrimiento de servicios sobre el protocolo de encaminamiento OLSRv1 dentro del simulador de red NS-2 versión 2.26.

En la sección 4.3 analizaremos en detalle las prestaciones del mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSRv1. En concreto, veremos cómo se comporta SD-OLSRv1 en cuanto a la sobrecarga introducida en la red y la tasa de descubrimiento de servicios se refiere.

4.1. Descripción de SD-OLSRv1

En esta sección vamos a describir en detalle el funcionamiento del mecanismo de descubrimiento de servicios SD-OLSRv1: cuál es el formato del nuevo mensaje SDM que hemos definido, sección 4.1.1, cuál es el procedimiento que realiza un nodo para generar mensajes SDM, sección 4.1.2, cuál es el procedimiento que realiza un nodo para procesar los mensajes SDM y en función de qué parámetros guarda en su caché de servicios los servicios anunciados, sección 4.1.3, la posibilidad en la versión 1 de SD-OLSRv1 que tienen los nodos de hacer peticiones de servicio, sección 4.1.4, y la obligación de los nodos de indicar la indisponibilidad de sus servicios cuando abandonan la red, sección 4.1.5.

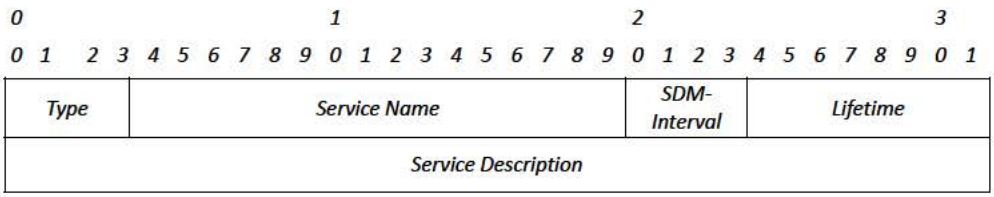

Figura 4.1: Formato del mensaje SDM

4.1.1. Formato del mensaje de descubrimiento de servicios

El formato de paquete del protocolo OLSRv1, nos permite añadir nuevas funcionalidades al protocolo de encaminamiento, definiendo nuevos tipos de mensajes que irán encapsulados en el campo MESSAGE del protocolo OLSRv1. Para el caso que nos ocupa, definimos un nuevo tipo de mensaje al que llamamos *Service Discovery Message* (SDM). SDM es un mensaje de pequeño tamaño, 8 bytes, de tal forma que el ancho de banda consumido por cada mensaje que se envía de este tipo es pequeño.

En un primer momento, en la definición del mecanismo de descubrimiento de servicios sobre OLSRv1, contemplamos la posibilidad de que los nodos fueran capaces tanto de anunciar servicios como de pedir un servicio que necesitaran y no tuvieran almacenado. Para ello definimos el mensaje SDM con el formato de la figura 4.1.

A continuación detallamos el significado de cada uno de los campos:

- *Type*: El campo *Type* indica si el mensaje es una petición o un anuncio de servicio. Un valor 1 significa que se trata de una petición de servicio y un valor 0 que es un mensaje para anunciar un servicio.

- *Service Name*: El campo *Service Name* indica dependiendo del valor del campo *Type*, el nombre del servicio que se anuncia o que se solicita.

- *SDM_INTERVAL*: El campo SDM_INTERVAL indica cada cuánto tiempo se envía un mensaje de anuncio de servicio. Se trata de un parámetro configurable por el usuario, de tal forma que se puede variar dependiendo del entorno en el que se opere. Cuanto menor sea este parámetro, el mensaje se transmitirá con más frecuencia, y la tasa de descubrimiento de servicios será mayor. En contrapartida, la sobrecarga que se introduzca en la red también aumentará.

- *LifeTime*: El campo *LifeTime* describe el tiempo durante el cual el servicio está disponible para el resto de nodos. Un valor 0 indica que el servicio deja de estar disponible.

- *Service Description*: El campo *Service Description* contiene información del servicio que un nodo solicita o del servicio que un nodo anuncia.

Tal como hemos dicho, este mensaje viaja dentro del paquete OLSRv1. Como ya hemos explicado en la sección 3.3.2, el paquete OLSRv1 tiene una cabecera donde los campos más importantes de cara al mensaje SDM son:

- *Originator Address*: Indica quién es el nodo que origina el mensaje. Nos informa del nodo proveedor del servicio, en el caso de que se trate de un mensaje de anuncio de servicios, o del nodo que busca un servicio en el caso de que se trate de una petición.

- *Time To Live*: Determina el número de saltos que tiene permitido como máximo este mensaje a lo largo de la red.

- *Hop Count*: Indica el número de saltos que lleva el mensaje. Se incrementa de uno en uno y se inicia en 0.

- *Message Sequence Number*: Es el identificador del mensaje. Cada vez que un servicio es anunciado, este valor se incrementa en uno.

4.1.2. Generación del mensaje SDM

Siempre que un nodo entre a formar parte de una red MANET anuncia sus servicios al resto de nodos de la red siguiendo el modo *push* del modelo teórico de descubrimiento de servicios descrito en la sección 2.2.2. Cada servicio se envía encapsulado en el paquete OLSR como parte del mensaje SDM según el formato descrito en la figura 4.2.

Se envían tantos mensajes SDM como servicios tenga el nodo para anunciar. Teniendo en cuenta que el mensaje SDM tiene un tamaño de 8 bytes, el tamaño total del paquete OLSR cuando se anuncian servicios será de:

```
Tamaño paquete con anuncio de servicios en OLSR v1:
   Cabecera paquete OLSRv1 + Mensaje SDM * Número de servicios = 16 + 8 * S bytes
```

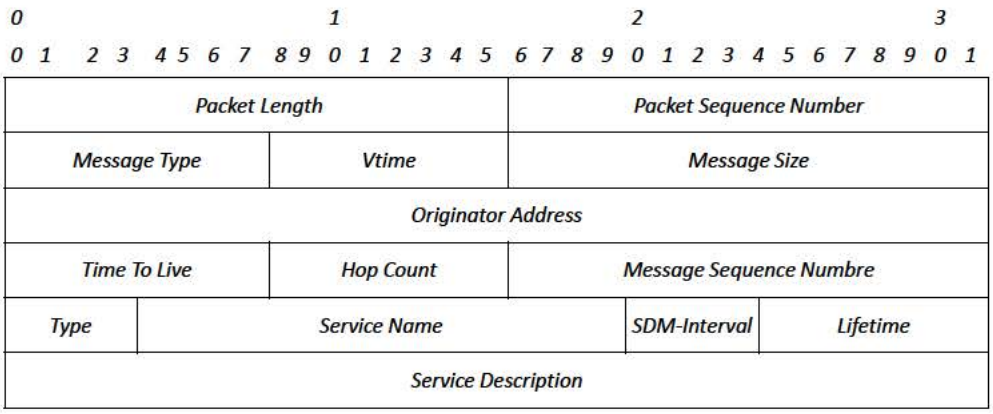

Figura 4.2: Mensaje SDM integrado en el protocolo OLSRv1

Los mensajes SDM se envían periódicamente para que los demás nodos tengan la información sobre los servicios disponibles en la red lo más actualizada posible. Este intervalo de tiempo, común para todos los anuncios de servicios que realicen los nodos, se denomina SDM_INTERVAL y es un parámetro configurable por el usuario. Una vez configurado este parámetro es fijo. Como ya hemos dicho cuanto más pequeño sea el valor del parámetro SDM_INTERVAL, más actualizada tendrán el resto de nodos su caché de servicios, pero el ancho de banda consumido también será mayor. Al contrario, cuanto mayor sea el valor del parámetro SDM_INTERVAL, menor será el ancho de banda utilizado pero es posible que se produzcan falsos positivos al no estar la caché de servicios actualizada. Es decir, es posible que un nodo se conecte a otro nodo para utilizar un servicio que él creía activo y que el servicio ya no esté disponible porque por ejemplo, el nodo haya abandonado la red.

4.1.3. Procesado del mensaje SDM y caché de servicios

Cada nodo que integra la red MANET almacena en una caché de servicios los servicios propios del nodo y también los servicios que va descubriendo en la red. En esta caché de servicios, existe un espacio para almacenar los servicios propios del nodo y otro, para almacenar los servicios que va descubriendo.

Cuando un nodo recibe un mensaje de anuncio de servicios extrae la información necesaria del paquete que recibe, y almacena, si procede, su contenido en el espacio habilitado para ello dentro de la caché de servicios.

Cada entrada en la caché de servicios, bien sea para servicios propios del nodo como para los servicios que va descubriendo, sigue el formato de la figura 4.3.

<Service Name>	*IP Address*	*<Lifetime>*	*<Service Description>*
Printer	x x.x.x	100 seconds	Color Printer

Figura 4.3: Formato caché de servicios en SD-OLSRv1

Si el servicio es nuevo se crea una nueva entrada en la caché guardando el nombre del servicio *Service Name*, la dirección IP del nodo que anuncia el servicio, *IP Address*, el tiempo de vida del servicio *Lifetime*, y por último, la descripción detallada del servicio *Service Description*. Si dicha entrada ya existe, lo único que se hace es actualizar el tiempo de vida del servicio.

La caché de servicios irá aumentando con el número de servicios que se vayan guardando, hasta llegar a un punto en que se llene. Cuando ocurre esto, hay que ir borrando servicios para dar cabida a otros nuevos. Los primeros que se van a borrar son aquellos cuyo parámetro *Lifetime* esté a punto de expirar. De la misma forma, todos los servicios cuyo *Lifetime* haya expirado, se borran automáticamente de la caché de servicios.

Es posible que el mismo servicio lo anuncien varios nodos. Como la caché de servicios de un nodo tiene un tamaño limitado, antes de guardar la información referente a un servicio, cada nodo comprueba si esa información ya ha sido previamente almacenada en su caché, porque haya sido anunciada por otro nodo. En caso de que ese servicio ya haya sido anunciado por otro nodo y se guarde copia de la información en la caché, se compararán los tiempos de vida de cada uno de los servicios y se almacenará en la caché el servicio del nodo que mayor tiempo de vida tenga.

También es posible que un nodo adquiera un nuevo servicio que ya se encontraba almacenado en su caché. En ese caso el servicio almacenado se borra y se sustituye por el servicio propio del nodo, independientemente del tiempo de vida de cada uno de ellos. Siempre es más rápido utilizar un servicio propio que uno de otro equipo. El algoritmo se puede resumir en la figura 4.4.

4.1.4. Descubrimiento de servicios

Inicialmente SD-OLSRv1 permitía enviar mensajes SDM de petición de servicios cuando un nodo necesitaba un servicio que no tenía almacenado en su caché. Estas peticiones se enviaban siguiendo el modo *pull* del modelo teórico de descubrimiento de servicios descrito en la sección 2.2.2. Sobre todo pensamos en este

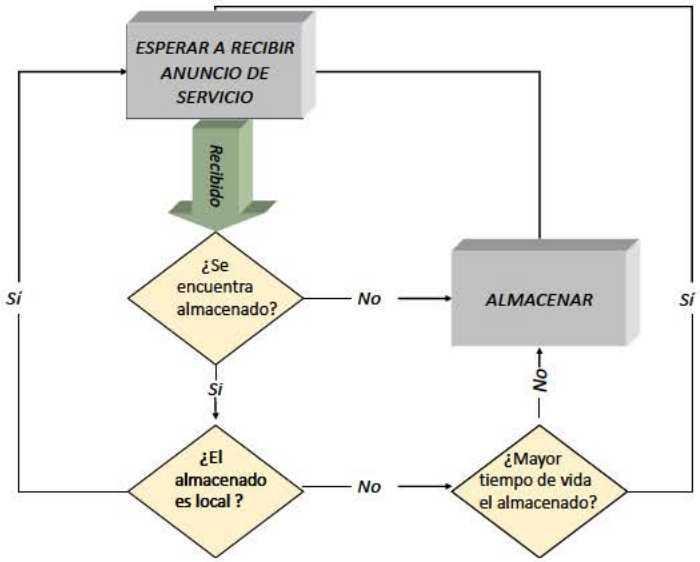

Figura 4.4: Recepción de anuncio de servicios en SD-OLSRv1

caso para los momentos en los que un nodo se incorpora a la red y quiere utilizar un servicio que se acaba de anunciar. Cuando este nodo consulte su caché de servicios no dispondrá de ninguna entrada con ese servicio, y de esta forma tendrá la posibilidad de enviar un mensaje de petición de servicios mediante difusión a toda la red.

La diferencia con el mensaje SDM de anuncio de servicios es que en este caso, el campo *Type* del mensaje SDM se refiere a un mensaje de petición de servicios, y el campo *Service Name* identifica al servicio que el nodo quiere buscar en lugar de identificar al servicio que el nodo anuncia.

Los campos SDM_INTERVAL y *LifeTime* viajan vacíos, puesto que ahora no se trata de anunciar un servicio cada cierto tiempo, sino de descubrir un servicio, y esto únicamente se hará cuando el nodo lo necesite.

Los nodos que son MPR son los encargados de retransmitir el mensaje de petición de servicio. Cuando un nodo recibe este mensaje, comprueba en su caché si alguno de los servicios que él soporta se corresponde con el servicio que se busca. Distinguimos dos casos:

- El servicio que se solicita lo ofrece el nodo que recibe el mensaje. En este caso, el nodo responderá con un mensaje SDM de anuncio de servicios. Es importante destacar que no enviará una respuesta *unicast* dirigida al nodo solicitante. Enviará un mensaje SDM idéntico a un mensaje de anuncio de

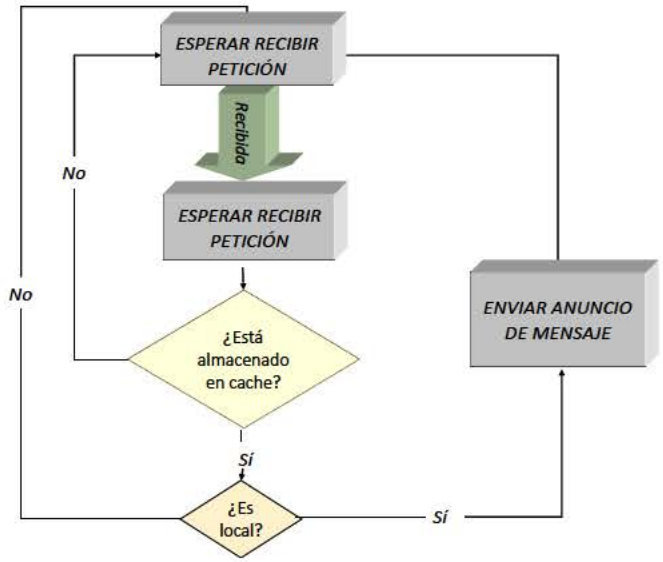

Figura 4.5: Recepción de un mensaje de petición de servicios en SD-OLSRv1

servicios. De esta manera se aumentará la sobrecarga en la red, pero, se conseguirá que si hay algún nodo en la misma situación, conozca también la existencia de ese servicio, y así se evitará el envío de peticiones posteriores del mismo servicio que también sobrecarguen la red.

- El servicio que se solicita no lo ofrece el nodo que recibe el mensaje. En este caso el nodo no hace nada y descarta el mensaje de petición de servicio.

Hay que destacar que antes de que un nodo responda al mensaje de petición de servicio, espera un tiempo aleatorio en el que comprueba, si algún otro nodo que también tuviera el servicio que se solicita hubiera respondido ya al mensaje de petición de servicios generado por el nodo origen. En caso afirmativo, el nodo ignora el mensaje y no lo responde. De esta forma se evitan múltiples respuestas ofreciendo el mismo servicio, con lo que se gana ancho de banda. Este algoritmo se puede resumir de forma gráfica en la figura 4.5.

Teóricamente a pesar de que este mecanismo pueda sobrecargar la red, algo que en el caso del protocolo OLSR se mejora con el mecanismo MPR, tiene la ventaja de que como el resto de nodos también escuchan la respuesta, pueden almacenar en su caché una entrada con las características del mismo, y así evitar buscarlo, si en un futuro necesitan un servicio de este estilo. De hecho, cuanto más se conozcan los nodos entre sí, menos peticiones de servicio se harán y, en consecuencia, menos se sobrecargará la red con nuevas peticiones. Obviamente,

se pueden dar casos en los que una petición no obtenga respuesta. En esos casos existen dos posibilidades. Que el nodo envíe peticiones indefinidamente hasta obtener respuestas o que no haga uso de ese servicio.

A pesar de la teoría, una vez realizadas las simulaciones y los resultados obtenidos, nos dimos cuenta que era mejor aprovechar el carácter proactivo del protocolo OLSR, ajustando la periodicidad con la que se envíaban los mensajes SDM de anuncio de servicios, que definir un mecanismo sobre un protocolo de encaminamiento proactivo que contemplara los modelos teóricos *push* y *pull*. Esta fue la versión dos de SD-OLSRv1. Para esta segunda versión del protocolo SD-OLSRv1 donde los nodos sólo anuncian servicios, cada nodo mantiene dos cachés de servicios diferentes, la primera donde se almacenan los servicios propios del nodo y la segunda donde se registran los servicios que anuncian el resto de nodos en la red. De esta forma, diferenciando los servicios ofrecidos por cada nodo de los servicios que se anuncian en la red, es más rápido consultar y guardar los servicios que se anuncian en la red. También es más rápido que un nodo anuncie sus servicios, puesto que éstos se guardan en una caché independiente.

4.1.5. Indisponibilidad de un servicio

Cuando uno de los nodos que integra la red MANET deja de tener disponible uno de los servicios que hasta ese momento ofrecía, tiene la obligación de mandar un mensaje de notificación al resto de nodos de la red indicando que el servicio en cuestión deja de estar disponible. De esta forma, los nodos que reciben esta notificiación pueden eliminar de sus respectivas cachés este servicio, y así evitar que se produzcan falsos descubrimientos. Del mismo modo, este servicio también se eliminará de la caché de servicios propios del nodo para que no vuelva a ser anunciado.

El mensaje de notificación de la indisponibilidad de un servicio tiene la misma estructura que el mensaje SDM de anuncio de servicios. La única particularidad es que el campo *LifeTime* tiene un valor igual a 0. De esta manera, cuando un nodo recibe este mensaje SDM de anuncio de servicios y va a actualizar su caché de servicios verá que el campo *LifeTime* es igual a 0, y por lo tanto, eliminará este servicio de su caché.

4.2. Implementación del mecanismo de descubrimiento de servicios SD-OLSRv1

En esta sección describimos la implementación del mecanismo de descubrimiento de servicios sobre el protocolo de encaminamiento OLSRv1, dentro del simulador de red NS-2 versión 2.26.

El mecanismo de descubrimiento de servicios sobre OLSRv1 lo hemos desarrollado en C++. Hemos partido de una implementación ya desarrollada para NS-2 del protocolo de encaminamiento OLSRv1, [UM-OLSR]. A continuación describimos en detalle los pasos que hemos seguido para implementarlo.

4.2.1. Creación de un nuevo mensaje SDM

Además de los mensajes que ya define el protocolo OLSR, mensajes *Hello*, mensajes MID y mensajes TC, tenemos que definir un nuevo tipo de mensaje, *Service Discovery Message* (SDM), para poder descubrir y anunciar servicios en la red. Este nuevo tipo de mensaje lo definimos en el fichero *OLSR_pkt.h* dentro de la implementación del protocolo OLSR que se ha utilizado para NS-2. Este nuevo tipo de mensaje tiene un tamaño de 8 bytes.

```
#define OLSR_SDM_MSG 4
#define OLSR_SDM_HDR_SIZE 8
```

También, dentro del fichero *OLSR_pkt.h*, definimos una estructura con los diferentes campos que forman el mensaje SDM, según se describe en la sección 4.1.1.

```
u_int8_t type_; /*Indicates whether the message
is an advertisement or a query request.*/
u_int8_t serviceName_; /*Name of the advertised/query message*/
u_int8_t sdminterval_; /*Indicates how often an advertisement message is transmitted.*/
u_int8_t lifeTime_; /*Indicates the time the
 service advertise will be availabe for the rest of the nodes*/
u_int32_t serviceDescription_;/*Service Description*/
```

Para el cálculo del tamaño del paquete OLSR que se transmite, además de los mensajes *Hello*, TC y MID definidos en el protocolo, hay que incluir el tamaño del mensaje SDM para descubrimiento de servicios (Cuadro 4.1).

```
inline u_int32_t size() {
    u_int32_t sz = OLSR_MSG_HDR_SIZE;
    if (msg_type() == OLSR_HELLO_MSG)
       sz += hello().size();
    else if (msg_type() == OLSR_TC_MSG)
       sz += tc().size();
    else if (msg_type() == OLSR_MID_MSG)
       sz += mid().size();
    else if (msg_type() == OLSR_SDM_MSG)
       sz += SDM().size();
    return sz;
}
```

Cuadro 4.1: Cálculo del tamaño de un paquete OLSR

De esta forma ya tenemos implementado el mensaje SDM tal como está definido en la descripción del mecanismo de descubrimiento de servicios de la sección 4.1.1. Cuando un nodo de la red tenga que procesar un paquete OLSR ya conocerá la existencia del nuevo mensaje.

Para poder evaluar la viabilidad del mecanismo de descubrimiento de servicios, extraemos del fichero de trazas generado en la simulación de un escenario, los valores que necesitamos para calcular la sobrecarga que el mecanismo de descubrimiento de servicios introduce en la red y la tasa de descubrimiento de servicios. Para ello tenemos que modificar el fichero de trazas para incluir la información referente a los mensajes SDM enviados (Cuadro 4.2).

La información que se escribe en el fichero de trazas referente al mensaje SDM es la siguiente:

- Nodo origen que envía el mensaje.
- Número de saltos que lleva acumulados el mensaje.
- Número de secuencia del mensaje.

Por lo tanto, un mensaje SDM enviado por el nodo 14 que lleva 3 saltos acumulados y tiene un número de secuencia de 67, queda reflejado del siguiente modo:

```
[-Pt SDM -Po 14 -Ph3 -Pms 67]
```

```
else if (newtrace_){
    sprintf(pt_->buffer() + offset, "-P olsr -Pn %d -Ps
            %d", op->count, op->pkt_seq_num());

    int len = strlen(pt_->buffer());
    for (int i = 0; i < op->count; i ++ ){
       const char *s;
       if(op->msg(i).msg_type() == OLSR_HELLO_MSG)
           s = "[-Pt HELLO -Po %d -Ph %d -Pms %d]";
       else if(op->msg(i).msg_type() == OLSR_TC_MSG)
           s = "[-Pt TC -Po %d -Ph %d -Pms %d]";
       else if(op->msg(i).msg_type() == OLSR_SDM_MSG)
           s = "[-Pt SDM -Po %d -Ph %d -Pms %d]";
       else
           s = "[-Pt UNKNOWN -Po %d -Ph %d -Pms %d]";
       sprintf(pt_->buffer() + len, s,
               op->msg(i).orig_addr(),
               op->msg(i).hop_count(),
               op->msg(i).msg_seq_num());
               len = strlen(pt_->buffer());
     }
}
```

Cuadro 4.2: Modificación en fichero de trazas para incluir referencias a mensajes SDM

Este mensaje forma parte de una traza completa con el siguiente formato:

```
r 3.119464459 _1_ RTR --- 7 OLSR 80 [0 ffffffff 0 800]
------- / [0:255 -1:255 32 0] [3 2 [-Pt SDM -Po 0 -Ph0
-Pms 1][-Pt SDM -Po 0 -Ph0 -Pms 2][-Pt SDM -Po 0 -Ph0 -Pms 3]]
```

En este caso se trata de un paquete OLSR recibido por el nodo 1 en el instante 3.11, enviado por el nodo 0 y formado por tres mensajes SDM con número de saltos 0 y números de secuencia 2, 3 y 4.

4.2.2. Envío de mensajes SDM

Uno de los parámetros que hay que definir en el mecanismo de descubrimiento de servicios es la periodicidad con la que se envían los mensajes SDM. Este parámetro se fija en el fichero *ns-2.26/tcl/lib/ns-default*.

```
Agent/OLSR set ival 3
```

Caché de servicios

Las memorias cachés las vamos a simular haciendo uso de ficheros que se ubicarán en el directorio */home*. En el caso de la primera versión de SD-OLSRv1 donde cada nodo tiene una única memoria caché en la que almacena todos los servicios, el fichero recibe el nombre de *servicecache*. En el caso de la segunda versión de SD-OLSRv1 donde cada nodo dispone de dos cachés diferentes, una para almacenar sus propios servicios y otra para almacenar los servicios que otros nodos anuncian en la red, los ficheros reciben el nombre de *localservicecache* y *remoteservicecache*. Para diferenciar las memorias caché de un nodo de las de sus nodos vecinos, a las memorias caché de cada nodo se les añade el número de nodo. De esta forma, el nodo número 1 guardará los servicios en el fichero */home/servicecache001*. Esto hace que el mecanismo de descubrimiento de servicios diseñado soporte un máximo de 1000 nodos.

Temporizadores

Para programar la periodicidad con la que se envían los mensajes SDM lo que hacemos es arrancar un temporizador que cuando vence envía el mensaje SDM.

```
OLSR::OLSR(nsaddr_t id) : Agent(PT_OLSR),
hello_timer_(this),
tc_timer_(this),
mid_timer_(this),
SDM_timer_(this){
```

Cuadro 4.3: Definición de temporizadores

Los temporizadores en C++ se apoyan en una clase base abstracta definida en el fichero *timerhandler.h*. Esta clase base está compuesta por las siguientes funciones de carácter público:

- `[double delay] void sched`. Programa un temporizador.
- `[double delay] void resched`. Programa un temporizador que no ha vencido.
- `[] void cancel`. Cancela un temporizador que no ha vencido aún.
- `[] int status`. Devuelve el estado de un temporizador.

Y los siguientes miembros de carácter protegido:

- `[Event* e] virtual void expire`. Este método es implementado por el agente OLSR.
- `[Event* e] virtual void handle`. Cuando se consume un evento, invoca al método *expire()* y fija el estado del timer de una manera apropiada.
- `int status`. Realiza un seguimiento del estado del temporizador.
- `Event event`. Evento que finalizará cuando el temporizador expire.

Por lo tanto, el simulador proporciona la clase base necesaria para dar soporte a los temporizadores y gracias a esto, el único fichero que deberemos modificar para programar la periodicidad de envío de mensajes SDM será el fichero *OLSR.cc*. En este fichero se incluirán las definiciones y el manejo de los eventos de los temporizadores.

El fichero *OLSR.cc* está compuesto por varias clases. La definición del temporizador se hace en la clase OLSR (Cuadro 4.3).

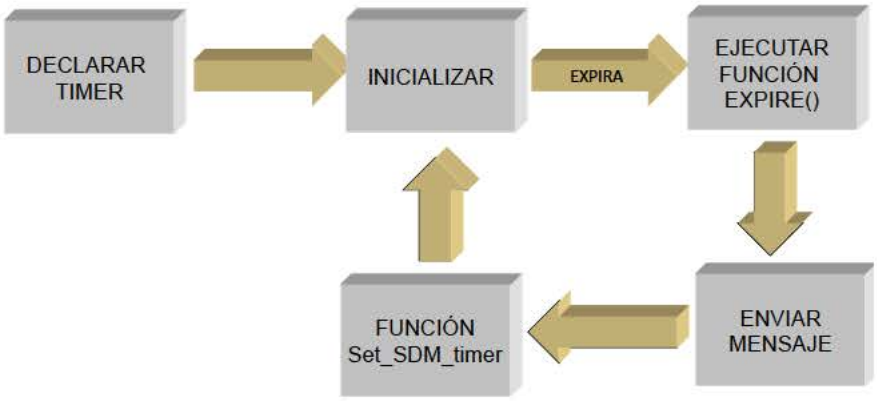

Figura 4.6: Comportamiento de los temporizadores en SD-OLSRv1

Una vez definido el temporizador, lo siguiente es inicializarlo. Cuando la simulación comienza, el temporizador debe fijarse a 0. Para ello llamamos a la función *resched* de la clase base definida en el fichero *timerhandler.h*. Desde ese momento el temporizador empezará a contar y cuando alcance el valor fijado en el fichero *ns-2.26/tcl/lib/ns-default*, el programa saltará automáticamente a la función *expire()* definida para fijar las acciones que se deben llevar a cabo cuando venza el temporizador. Una vez que todas las acciones que conlleva el vencimiento del temporizador se realicen, lo siguiente es volver a programar el temporizador. Utilizamos para ello la siguiente instrucción:

```
agent_->set_SDM_timer();
```

donde el identificador *agent_* hace referencia al nodo cuyo temporizador ha finalizado y la función *set_SDM_timer()* será la encargada de reprogramar el temporizador.

El diagrama de flujo que define el funcionamiento de los temporizadores se muestra en la figura 4.6.

Envío de anuncio de servicios

En una red MANET los nodos están en constante movimiento. Un nodo puede pertenecer a una red, pero debido a sus movimientos, o a los movimientos del resto

```
void OLSR_SDMTimer::expire(Event* e){
    if(agent_->nbset().size()>0){
    }
}
```

Cuadro 4.4: Comprobación de los vecinos de un nodo en OLSRv1

de nodos, puede dejar de pertenecer a ella en cualquier momento. Esto ocurre cuando su radio de alcance es inferior a la distancia existente hasta el nodo más cercano. En el caso del simulador NS-2 el radio de alcance por defecto definido para el protocolo MAC 802.11b es de 250 metros, aunque puede ser modificado a cualquier otro valor por el usuario.

Si un nodo deja de pertenecer a la red donde se están anunciando servicios, el protocolo SD-OLSRv1 se daría cuenta, puesto que dejaría de recibir mensajes SDM y también mensajes *Hello* y TC. En ese caso, si el nodo no tiene nodos vecinos, y tiene algún servicio que anunciar, cuando venza su temporizador dejará de anunciar servicios, puesto que éstos no llegarán a ningún lado. Para comprobar si el nodo tiene vecinos, hacemos uso de la estructura *nbset* donde se almacenan el conjunto de vecinos de un nodo (Cuadro 4.4).

Cuando transcurra el tiempo definido en el parámetro SDM_INTERVAL, y en el caso de que el nodo sí que tenga vecinos, mirará en su caché de servicios qué servicios tiene que anunciar. Dependiendo de la versión de SD-OLSRv1 que utilicemos el nodo buscará en la caché de servicios locales, o en la caché de servicios conjuntos. Se irá leyendo el fichero línea a línea y los servicios que haya que anunciar se guardarán en una cadena de caracteres que se pasará como argumento a la función *send_SDM*.

Además del nombre del servicio, la función *send_SDM* (Cuadro 4.5) necesita conocer el tiempo de vida del servicio, el tipo de servicio, si se trata de un anuncio o una petición de servicio (sólo en el caso de la primera versión de SD-OLSRv1), la periodicidad con la que se envía el servicio y la descripción del mismo. Los primeros valores los obtenemos de la caché de servicios. El valor del parámetro SDM_INTERVAL lo obtenemos del fichero *ns-default.tcl*. Como descripción del servicio, enviaremos un valor constante en todos los anuncios. Esto se ha decidido diseñar así, porque para el objetivo de esta tesis, que es evaluar las prestaciones del mecanismo de descubrimiento de servicios diseñado este dato no tiene impacto. Tampoco lo tiene en la comparativa con otros protocolos que utilizan un formato de descripción de servicios similar, todos basados en la descripción de servicios que hace el protocolo SLP. Este campo tendrá su importancia cuando este mecanismo

de descubrimiento de servicios se implemente en un escenario real, ya que los nodos deberán saber qué características tendrá el servicio que quieren utilizar. En esta tesis solamente se simula el descubrimiento de servicios, no el uso que se haga de los servicios anunciados.

Una vez construido el mensaje hay que encolarlo para que posteriormente sea transmitido.

En resumen, el proceso de envío de un anuncio de servicio se puede resumir en los siguientes pasos:

- Se crea un temporizador para el envío de mensajes.
- Se crea una función para manejar los eventos que el temporizador conlleva al expirar.
- Se recogen de la caché de servicios los datos del servicio que se va a anunciar.
- Se rellenan los campos del paquete OLSR y la función *send_SDM*.
- Se encola el mensaje.

4.2.3. Recepción de mensajes SDM

Cuando un nodo recibe un paquete lo primero que tiene que hacer es comprobar qué tipo de paquete es y qué mensajes soporta. La clase OLSR tiene una función *recv* que se encarga de ello. En caso de que el paquete sea un paquete OLSR, éste se procesará siguiendo las instrucciones de la RFC 3626. Se llamará a la función *recv_olsr* que es la encargada de recibir los paquetes OLSR. Esta función también pertenece a la clase C++ OLSR. En NS-2 todos los mensajes de los protocolos de encaminamiento se envían desde y al puerto *RT_PORT*. Por eso, lo primero que hará esta función *recv_olsr* es comprobar este puerto.

Una vez hecha esta comprobación, hay que identificar el tipo de mensajes que transmite el protocolo OLSR para procesar cada uno de ellos de manera adecuada. Al introducir un nuevo tipo de mensaje en SD-OLSRv1, tenemos que incluir también el código necesario para ser capaces de identificarlo y procesarlo adecuadamente (Cuadro 4.6).

En caso de tratarse de un mensaje SDM se llama a la función *process_SDM* que es la encargada de almacenar, si procede, en la caché de servicios el servicio anunciado. A esta función se le pasa como parámetro el paquete SDM y el identificador del nodo que ha recibido el mensaje.

```
/*Creates a new OLSR SDM message which is buffered
  for being sent later on*/

void
OLSR::send_SDM(int servicename, int lifetime, int servicetype) {
    OLSR_msg msg;
    double now = CURRENT_TIME;

    /* OLSR message header for SDM type of message*/
    msg.msg_type() = OLSR_SDM_MSG;
    msg.vtime() = OLSR::seconds_to_emf(OLSR_NEIGHB_HOLD_TIME);
    msg.orig_addr() = ra_addr();
    msg.ttl() = 255;
    msg.hop_count() = 0;
    msg.msg_seq_num() = msg_seq();

    /*SDM message header*/
    /*OLSR_msg structure is defined in OLSR_pkt.h*/

    /*Type of message depends on if the message is for advertise or
      query when the SDM message is transmitted because the timer expires
      is an advertise message*/
    msg.SDM().type() = servicetype;
    msg.SDM().sdminterval() = 3;

    /*The service name is extracted from de service
      cache(it is passed when send_SDM is called)*/

    msg.SDM().serviceName() = servicename;
    msg.SDM().lifeTime() = lifetime;
    msg.SDM().serviceDescription() = 2;
    msg.msg_size() = msg.size();
    enque_msg(msg, JITTER);
}

agent_->set_SDM_timer();
```

Cuadro 4.5: Envío de un mensaje SDM

```
/*Process the message according to its type*/
if (msg.msg_type() == OLSR_HELLO_MSG)
    process_hello(msg, ra_addr(), ih->saddr());
else if (msg.msg_type() == OLSR_TC_MSG)
    process_tc(msg, ih->saddr());
else if (msg.msg_type() == OLSR_MID_MSG)
    process_mid(msg, ih->saddr());
else if (msg.msg_type() == OLSR_SDM_MSG)
    process_SDM(msg, ra_addr());
else {
    debug("%f: Node %d can not process OLSR packet because does not
           implement OLSR type (%x)\n",
           CURRENT_TIME,
           OLSR::node_id(ra_addr()),
           msg.msg_type());
}
```

Cuadro 4.6: Identificación de mensajes en OLSRv1

Siguiendo con la mecánica del protocolo OLSR, lo primero que se hace en la función *proces_SDM* es comprobar que realmente el mensaje es de tipo SDM (Cuadro 4.7). Esto se llevará a cabo haciendo uso de la librería *assert*, la cual permite abortar la ejecución de un programa en caso de que la condición comprobada no se cumpla. Una vez comprobado que se trata de un mensaje SDM, bien un mensaje de anuncio de servicios o bien un mensaje de petición de servicios, se capturan los datos del mensaje: *nombre del servicio, tiempo de vida del servicio, tipo de mensaje y nodo que anuncia o solicita el servicio* dependiendo de si el mensaje es un mensaje de anuncio o un mensaje de petición de servicio. Los tres primeros parámetros se capturan del mensaje SDM y el último de ellos, el nodo que anuncia o solicita el servicio) de la cabecera del mensaje OLSR.

El tratamiento que se hace del mensaje SDM es diferente para los mensajes de anuncio de servicios que para los mensajes de petición de servicios. Estos últimos sólo están implementados en la primera versión desarrollada sobre OLSRv1. Vamos a ver a continuación el tratamiento que recibe cada uno de ellos.

Recepción de mensajes SDM de anuncio de servicios

Este tipo de mensaje se caracteriza porque el campo *Type* del mensaje SDM tiene un valor igual a 1 (sección 4.1.1).

La recepción de un mensaje SDM de anuncio de servicios trae consigo el alma-

```
/* Processes a SDM message following the implementation we have done
/* service discovery.
/*
/* \param msg the OLSR message which contains the SDM message.
/* \param sender_iface the address of the interface
/*  where the message was sent from.*/

void
OLSR::process_SDM(OLSR_msg& msg, nsaddr_t receiver_iface) {
    assert(msg.msg_type() == OLSR_SDM_MSG);

    u_int8_t serName;
    u_int8_t serLifeTime;
    u_int8_t serType;
    nsaddr_t sender_iface;

    /*Extract service information from SDM message
    serName = msg.SDM().serviceName();
    serLifeTime = msg.SDM().lifeTime();
    serType = msg.SDM().type();
    sender_iface = msg.orig_addr();
```

Cuadro 4.7: Procesamiento de mensajes SDM

```
if  (duplicate_service)
    then compare(ttl_service_stored && ttl_new service)
    save service_with_bigger_ttl
```

Cuadro 4.8: Procesado de mensajes SDM. Duplicación de servicios

```
if(service==1){ /*New services must be added*/
    fseek(fd,2L,SEED_END);
    position=ftell(fd)-2;
}
```

Cuadro 4.9: Escribir un servicio en la caché de servicios

cenamiento del servicio, si procede, en la caché de servicios del nodo que recibe el mensaje. Si el servicio recibido ya está almacenado y es un servicio que ofrece el propio nodo, no se hará nada con el mensaje. Pero si el servicio almacenado no es un servicio que ofrezca el propio nodo, se compararán los tiempos de vida del servicio almacenado con el del servicio recibido en el mensaje SDM y se guardará en la caché de servicios el de mayor tiempo de vida (Cuadro 4.8). Por último, en el caso de que el servicio no se encuentre almacenado, directamente se guarda.

Para conseguir reflejar esto en una simulación, es necesario tener un buen manejo de los ficheros. Se abrirá el fichero en modo lectura y se leerá el fichero fila a fila. El contenido de cada fila se almacenará en una cadena de caracteres. De esta cadena de caracteres, se deberán obtener los valores que sean de interés. En este caso, el valor que más interesa es el del nombre del servicio.

Una vez conocidos los datos de interés del mensaje SDM y de la memoria caché, se deben realizar las comprobaciones oportunas. La primera será ver si el servicio almacenado en esa línea del fichero se corresponde con el servicio anunciado. La presencia o no del servicio anunciado en el fichero se controlará mediante una variable *service*, que valdrá 0 si el servicio ya se encuentra almacenado y 1 si no es así.

Si una vez recorridas todas las filas del fichero, el servicio recibido no se encuentra almacenado, variable *service* igual a 1, se debe obtener la última posición del fichero, para que sea ahí donde posteriormente se escriba el nuevo servicio anunciado (Cuadro 4.9).

En caso de que el servicio sí que exista en la memoria caché hay que comprobar los tiempos de vida del servicio guardado y del nuevo. Se guardará el servicio cuyo proveedor ofrezca un tiempo de vida mayor (Cuadro 4.10).

```
if(storedProvider != receiver_interface){
    if(storedLife < serLifeTime){
        old=0;/*stored service has lower lifetime*/
        /*position where service must be updated*/
        position=ftell(fd) - start -17;
        fseek(fd, position, SEEK_SET);
        fprintf(fd, "%4.4ld\t%4.4d\t4.4d\tn\n"
        , sender_iface, serName, serLifeTime);
    }
}
```

Cuadro 4.10: Comprobación de si el servicio ya existe en la caché de servicios

En la figura 4.7 se muestra un diagrama de cómo se ha realizado el algoritmo.

Recepción de mensajes SDM de petición de servicios

Este tipo de mensaje se caracteriza porque el campo *Type* del mensaje SDM tiene un valor igual a 0 (sección 4.1.1).

La recepción de un mensaje SDM con una petición de servicio lleva consigo un proceso algo más sencillo que el caso anterior. En este caso lo que se debe hacer es comprobar si el servicio pedido lo ofrece el nodo que ha recibido la petición. En caso afirmativo, el nodo enviará un mensaje SDM anunciando ese servicio a toda la red. Es decir, no se responderá de modo *unicast* al nodo emisor de la petición, sino que se anunciará a todo el mundo, con el objetivo de dar a conocer ese servicio a toda la red y evitar futuras peticiones de ese servicio.

El proceso es muy similar al caso anterior. Se abre el fichero y se va leyendo fila a fila. Se compara si el servicio almacenado coincide con el que se ha solicitado en el mensaje SDM. En caso afirmativo se comprueba si el receptor de la petición es proveedor de ese servicio y de ser así, se realiza una llamada a la función *send_SDM*, pasándole los parámetros: *nombre del servicio, tiempo de vida y tipo de servicio*. Este último parámetro se pasará con un valor igual a 1 al tratarse de un mensaje de anuncio de servicio.

Destacar que para enviar este anuncio de servicio no se espera a que finalice el temporizador lanzado que controla el envío de anuncio de servicio periódicos. Se trata de un evento totalmente independiente a los demás.

En resumen, el proceso de recepción de un mensaje SDM se puede resumir en los siguientes pasos:

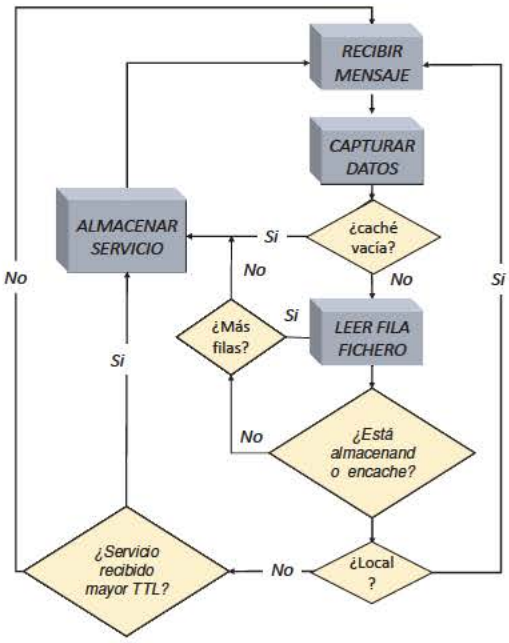

Figura 4.7: Recepción de mensajes SDM de anuncio de servicios en SD-OLSRv1

- Se crea una función para recibir todos los paquetes y que distinga los paquetes OLSR.

- Se crea una función capaz de procesar los paquetes OLSR, distinguiendo los diferentes tipos existentes.

- Se crea una función para cada tipo de mensaje. En este caso, *process_SDM*.

- Se distingue entre anuncios y peticiones de servicios.

- En el caso de recibir un mensaje de anuncio de servicios, se almacenan, si procede, los servicios, siguiendo el algoritmo explicado.

- En el caso de recibir un mensaje de petición de servicios, si el nodo que recibe el mensaje proporciona el servicio que se solicita, responde con un mensaje SDM de anuncio de servicios.

4.3. Prestaciones de SD-OLSRv1

Las prestaciones de SD-OLSRv1 las vamos a evaluar para las dos versiones del mecanismo de descubrimiento de servicios que se han diseñado. La primera versión, en la que el protocolo permite que los nodos que integran la red puedan anunciar sus servicios y también solicitar los servicios de terceros y la segunda versión, en la que el protocolo sólo permite que los nodos anuncien sus servicios a través de mensajes SDM.

Los parámetros que vamos a evaluar son los siguientes:

- Sobrecarga introducida en la red debido al mecanismo de descubrimiento de servicios.
- Tasa de descubrimiento de servicios que se descubren en la red con este mecanismo.

Las simulaciones realizadas muestran cómo varían los resultados de la simulación conforme el número de nodos que integran la red aumenta. Siempre que no se indique algún parámetro de simulación, se debe considerar el valor indicado en el escenario base.

4.3.1. Sobrecarga introducida por SD-OLSRv1

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv1 referentes a la sobrecarga introducida en la red, hemos considerado escenarios con una media de 10, 20, 30 y 50 nodos. Al comienzo de la simulación los nodos anuncian un servicio, diferente para cada nodo. A lo largo de la simulación cada nodo anuncia un segundo servicio.

Los nodos se mueven siguiendo el modelo de movilidad Random Waypoint. Cada nodo empieza en un lugar específico del escenario y pasado un tiempo, elige un nuevo destino aleatorio y se mueve hacia él a una velocidad uniforme entre 0 y un máximo de 10 m/s. En este modelo, los nodos se distribuyen en un área de 1000 x 1000 metros con un rango de transmisión de 250 metros. El modelo de propagación radio que utilizamos es el Two-Ray Ground. El tiempo de simulación se ha establecido en 400 segundos. Cada simulación la repetimos 10 veces para dar mayor veracidad a los resultados obtenidos. Estos valores permiten obtener un intervalo de confianza del 95 %.

En el caso de SD-OLSRv1 los anuncios de servicios se envían como un mensaje más del protocolo de encaminamiento OLSRv1. Por lo tanto, la frecuencia con la

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	cada 1, 5, 10 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 1, 5, 10 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 4.11: Parámetros de simulación Escenario 1, sobrecarga SD-OLSRv1

que se envían estos mensajes influye en el número de mensajes que se transmiten y en consecuencia en la sobrecarga introducida en la red. La frecuencia con la que se envían los mensajes *Hello* y TC del protocolo OLSRv1 la fijamos en 2 y 5 segundos respectivamente. Valores recomendados en la RFC del protocolo de encaminamiento OLSRv1 [Clausen et al, 2003].

Vamos a considerar varios escenarios para comparar la sobrecarga que introducen ambas versiones de SD-OLSRv1 respecto al protocolo de encaminamiento OLSRv1.

Escenario 1

En el escenario 1, la frecuencia de envío de mensajes SDM fijada con el parámetro SDM_INTERVAL es de 1, 5 y 10 segundos para ambas versiones de SD-OLSRv1. Además, para la versión 1 de SD-OLSRv1 en la que los nodos pueden hacer peticiones de servicios, configuramos que cada nodo realice una petición de servicio cada 5 segundos. En la tabla 4.11 se muestra un resumen con los parámetros de la simulación.

En las figuras 4.8, 4.9 y 4.10 comparamos las dos versiones del protocolo SD-OLSRv1 en función del número de nodos que integran la red y para diferentes

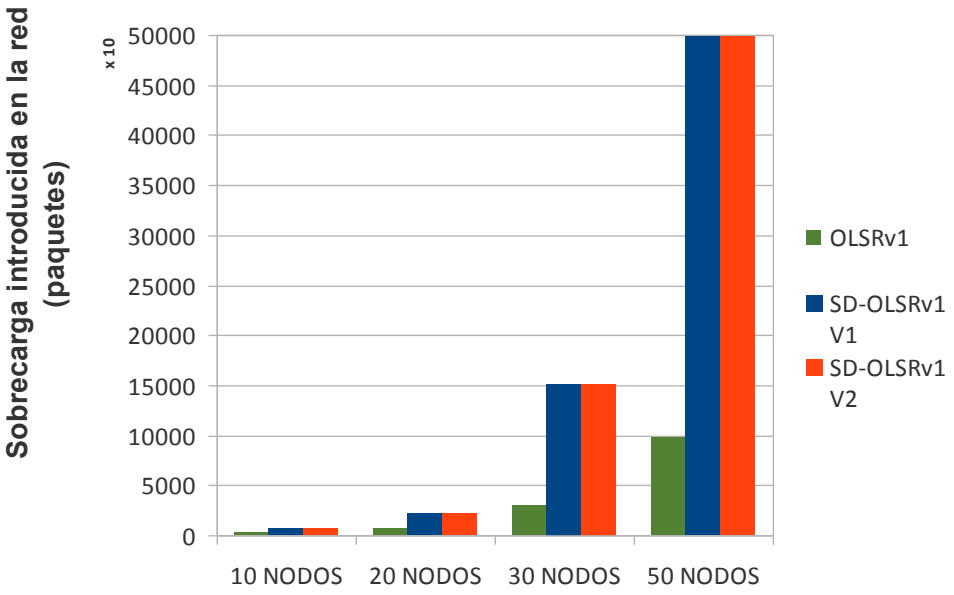

Figura 4.8: Sobrecarga introducida en la red por los mecanismos de descubrimiento de servicios SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a 1 segundo

valores del parámetro SDM_INTERVAL.

Tal como podemos observar, independientemente del número de nodos que integren la red, no existe ninguna diferencia en cuanto a sobrecarga se refiere, entre ambas versiones de SD-OLSRv1 cuando los nodos anuncian sus servicios cada segundo. Esto es debido a que para ambas versiones, los nodos están anunciando servicios de forma periódica y muy frecuente. De tal forma que todos los nodos de la red tienen conocimiento de los servicios que se ofrecen en la misma y por lo tanto, las peticiones de solicitud de servicio de la primera versión de SD-OLSRv1, programadas cada 5 segundos, no progresan en la red.

También observamos que cuando los nodos están enviando mensajes de anuncio de servicios cada segundo, la sobrecarga debido al mecanismo de descubrimiento de servicios medida en paquetes que se envían a la red es muy grande. En el caso de redes de 50 nodos el número de paquetes transmitidos llega a ser 5 veces más que los del protocolo de encaminamiento OLSRv1. A medida que la red es más grande, la sobrecarga aumenta considerablemente, puesto que aumenta el número de nodos y por lo tanto también el número de retransmisiones de paquetes. Cada nodo está enviando un mensaje SDM de anuncio de servicio cada segundo de la simulación. Mientras que en el protocolo de encaminamiento

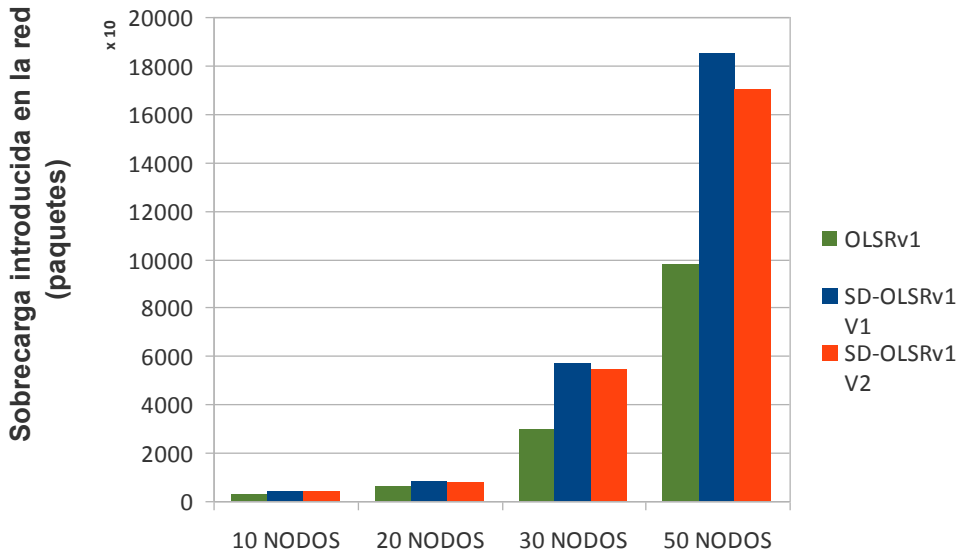

Figura 4.9: Sobrecarga introducida en la red por los mecanismos de descubrimiento de servicios SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a 5 segundos

OLSRv1 los mensajes *Hello* y TC se envían cada 2 y 5 segundos respectivamente.

Viendo las figuras 4.9 y 4.10 observamos que cuando la periodicidad de anuncio de servicios es de 5 y 10 segundos la sobrecarga de la primera versión de SD-OLSRv1 es mayor que la de la segunda versión. Esta diferencia es más apreciable a medida que la red es más grande y la periodicidad con la que se envían los servicios aumenta. Para estas frecuencias de envío de mensajes, cuando los servicios que se anuncian en la red varían en el tiempo, puede darse el caso de que un nodo solicite un servicio que se acaba de anunciar en la red pero que, por razones como que se haya perdido el mensaje de anuncio de servicio, éste no haya llegado al nodo que solicita el mensaje y por lo tanto, este nodo tenga que cursar la petición de servicio, puesto que el siguiente anuncio de servicios llegará 5 o 10 segundos después. No al segundo siguiente como ocurría en el caso anterior.

Cuanto más grande sea la red y más enlaces existan entre nodos, la posibilidad de que se pierdan mensajes se hace más grande. Y cuanto más grande sea la red, más retransmisiones supondrá un mensaje de petición de servicio y su correspondiente respuesta. Por eso, en redes con 50 nodos y una periodicidad de envío de anuncios de servicios de 10 segundos es cuando se aprecia más este aumento en el número de paquetes transmitidos de la primera versión de SD-OLSRv1 respecto

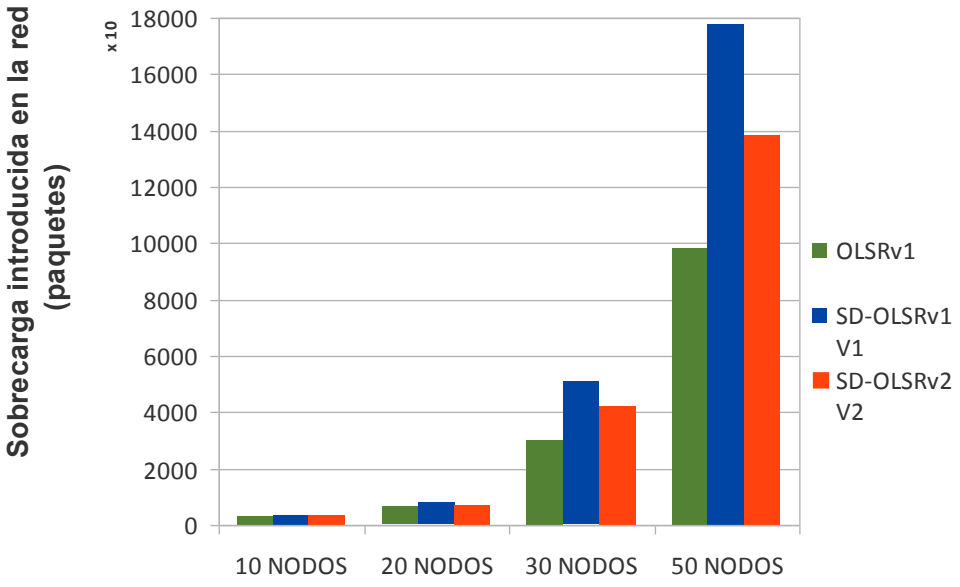

Figura 4.10: Sobrecarga introducida en la red por los mecanismos de descubrimiento de servicios SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a 10 segundos

a la segunda versión.

Para todos los escenarios vemos que la sobrecarga que introduce el mecanismo de descubrimiento de servicios, entendida como el número de paquetes que se transmiten en la red, es grande. Como era de esperar a medida que aumenta la frecuencia con la que se envían mensajes SDM de anuncio de servicios, esta sobrecarga también aumenta. Sin embargo, a pesar de que el número de paquetes aumenta considerablemente respecto al protocolo de encaminamiento OLSRv1, cada mensajes SDM ocupa 8 bytes, más 16 de la cabecera del protocolo OLSRv1, con lo que los kbytes que se consumen en la red son pocos.

Escenario 2

En el escenario 2 vamos a considerar los siguientes parámetros para cada una de las versiones de SD-OLSRv1:

- Para la versión 1, la frecuencia de envío de mensajes SDM fijada con el parámetro SDM_INTERVAL es de 400 segundos, que es el tiempo que dura la simulación. De esta forma los nodos sólo anunciarán los servicios una vez. Además los nodos harán peticiones de servicios cada 5 segundos. Cada vez que hacen una petición, todos los nodos solicitan el mismo servicio.

- Para la versión 2, los nodos anunciarán sus servicios cada 5 segundos, es decir el parámetro SDM_INTERVAL lo fijamos a 5 segundos, que es el valor intermedio entre el segundo y los 10 segundos simulados en el escenario anterior.

En la tabla 4.12 se muestra un resumen con los parámetros de la simulación.

Analizando los resultados obtenidos en la figura 4.11, observamos que en redes grandes la sobrecarga introducida por la versión 1 de SD-OLSRv1 es mayor que la sobrecarga introducida por la versión 2. Sin embargo, en redes pequeñas la sobrecarga de la versión 2 de SD-OLSRv1 es mayor que la de la versión 1.

Cuando la red es pequeña, no existen tantas pérdidas de paquetes. En la versión 2 de SD-OLSRv1 cada 5 segundos los nodos anuncian sus servicios, independientemente de que estos servicios ya sean conocidos por el resto de nodos de la red. En la versión 1, sólo se realizan peticiones de servicios cuando el nodo no tiene almacenado en su caché de servicios el servicio que solicita. Como se trata de redes pequeñas y el número de servicios es pequeño, en el tiempo de simulación no se están enviando peticiones de servicios cada 5 segundos porque los nodos

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	400 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 5 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 4.12: Parámetros de simulación Escenario 2, sobrecarga SD-OLSRv1

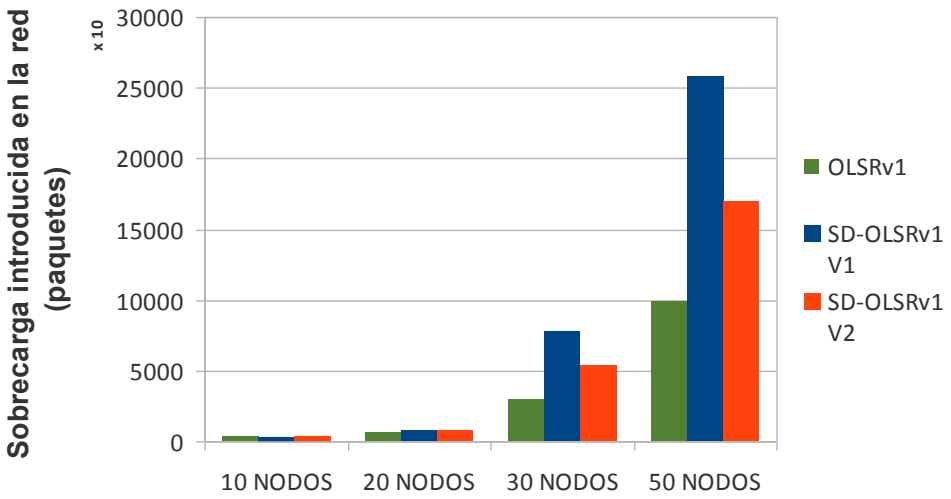

Figura 4.11: Sobrecarga introducida en la red por los mecanismos de descubrimiento de servicios SD-OLSRv1 versión 1 y 2 en el Escenario 2

ya tienen almacenados los servicios en la caché. Por lo tanto, la sobrecarga en la primera versión de SD-OLSRv1 es menor que en la segunda versión.

No ocurre lo mismo en redes grandes. A medida que la red se hace más grande y hay más servicios disponibles en la red, las peticiones de servicios de SD-OLSRv1 sí que progresan en la red. El consumo de ancho de banda por cada petición que progresa en la red es el doble que la de un anuncio de servicio. Puesto que cada petición implica dos mensajes SDM, el primero solicitando el servicio y el segundo anunciando, a modo de respuesta, el servicio.

Por lo tanto, en redes grandes, si queremos que todos los nodos de la red sepan de todos los servicios que se anuncian en la red (hacemos que todos los nodos en cada petición de servicio soliciten el mismo servicio), con la versión 1 de SD-OLSRv1 se consume más ancho de banda que con la versión 2. Además, los nodos tendrán un conocimiento de todos los servicios que se anuncian en la red cuando pregunten por todos los servicios y eso requiere un tiempo. Cada 5 segundos cada nodo pregunta por un servicio diferente. No ocurre lo mismo en la versión 2 de SD-OLSRv1 donde dado el mecanismo tipo *push* de anuncio de servicios, éstos se anuncian periódicamente en la red. Y estos anuncios llegan a todos los nodos al mismo tiempo.

Escenario 3

En el escenario 3 vamos a considerar los siguientes parámetros para cada una de las versiones de SD-OLSRv1:

- Para la versión 1, la frecuencia de envío de mensajes SDM fijada con el parámetro SDM_INTERVAL es de 400 segundos que es el tiempo que dura la simulción. De esta forma los nodos sólo anunciarán los servicios una vez. Además los nodos harán peticiones de servicios cada 5 segundos. Cada vez que hacen una petición, todos los nodos solicitan el mismo servicio.
- Para la versión 2, los nodos anunciarán sus servicios cada 100 segundos, es decir el parámetro SDM_INTERVAL lo fijamos a 100 segundos.

En la tabla 4.13 se muestra un resumen con los parámetros de la simulación.

Analizando los resultados que obtenemos en la figura 4.12, vemos que volvemos a tener dos situaciones diferentes dependiendo del número de nodos que integren la red.

En redes pequeñas la sobrecarga de la versión 2 de SD-OLSRv1 es ligeramente menor que la sobrecarga de la versión 1. En redes pequeñas la pérdida de mensajes

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	400 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 100 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cu v1

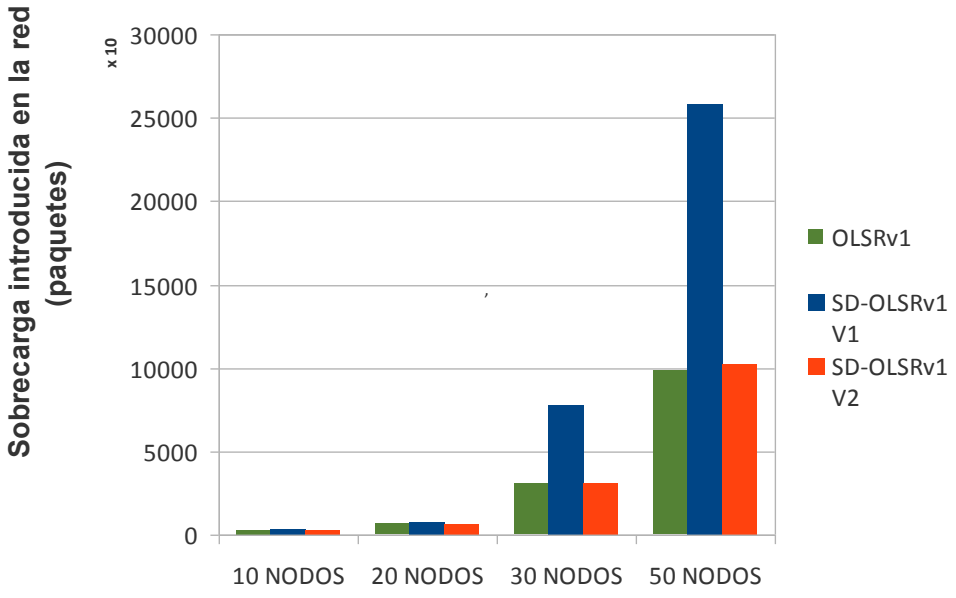

Figura 4.12: Sobrecarga introducida en la red por los mecanismos de descubrimiento de servicios SD-OLSRv1 versión 1 y 2 en el Escenario 3

debido a rotura de enlaces es pequeña y como vemos en la versión 1 de SD-OLSRv1 no se cursa cada petición de servicios que los nodos solicitan cada 5 segundos. Para la versión 2 de SD-OLSRv2 ya hemos dicho que cada nodo anuncia sus servicios cada 100 segundos. Sin embargo, los paquetes que se envían en la primera versión de SD-OLSRv1 sí que son ligeramente superiores puesto que cada petición de servicio desencadena dos mensajes SDM: un mensaje de petición de servicio y otro mensaje de respuesta a la petición del servicio.

En redes grandes la sobrecarga de la versión 2 de SD-OLSRv1 es mucho menor que la sobrecarga de la versión 1. Esto es debido a que el número de mensajes SDM que se envían en la versión 2 es mucho menor que los que se envían en la versión 1. En la versión 1 de SD-OLSRv1, a medida que aumenta el número de nodos que integran la red, los nodos necesitan enviar más mensajes SDM de petición de servicios puesto que los nodos no disponen de esos servicios en su caché.

Por los resultados obtenidos, observamos que en redes pequeñas de 10 nodos, la versión 2 de SD-OLSRv1 que anuncia servicios con una periodicidad de 1, 5 y 10 segundos introduce más sobrecarga en la red que el mecanismo de descubrimiento de servicios versión 1, donde los nodos hacen peticiones de servicio cada 5 segundos. Esto es debido a que a pesar de que estén programadas peticiones de servicios cada 5 segundos, éstas no progresan todas en la red, puesto que dado el tiempo de simulación y el número de nodos que integran la red, los nodos conocen los servicios que se ofrecen en la red sin necesidad de hacer peticiones cada 5 segundos. En cambio en la versión 2 de SD-OLSRv1 los nodos siempre están enviando mensajes de anuncios de servicio independientemente de que los nodos de la red ya conozcan estos servicios. En redes pequeñas, por lo tanto, la frecuencia con la que se envían mensajes de anuncio de servicio tiene que superior a los 10 segundos para que la sobrecarga introducida en la red sea más pequeña.

En redes grandes de 50 nodos, la sobrecarga introducida en la red es menor en la versión 2 de SD-OLSRv1 que en la versión 1 de SD-OLSRv1, cuando en esta versión configuramos el mecanismo de descubrimiento de servicios como un mecanismo tipo *pull* en el que los nodos envían peticiones de servicios cada 5 segundos y los servicios únicamente se anuncian una vez (mecanismo tipo *push*). En redes MANET con un número de nodos elevado, dada la movilidad de los nodos es más probable que ocurran roturas de enlaces y que, por lo tanto, no todos los mensajes que se envían en la red lleguen a su destino. Por este motivo, la mayor parte de las peticiones de servicio que realizan los nodos cada 5 segundos implican la transmisión de un mensaje SDM de petición de servicio. Por cada mensaje SDM de petición de servicio que se envía existe otro mensaje SDM como respuesta del nodo que ofrece el servicio. Esto implica el doble de mensajes SDM

que las de un mecanismo tipo *push* puro.

Por lo tanto, podemos concluir que en cuanto a sobrecarga de paquetes introducidos en la red se refiere, cuando se trata de redes grandes la versión 2 de SD-OLSRv1 con frecuencias bajas de envío de anuncios de servicios se comporta mejor que la versión 1 de SD-OLSRv1 cuando los nodos sólo hacen peticiones de servicios periódicas. En redes pequeñas, la frecuencia de envío de anuncio de servicio en la versión 2 de SD-OLSRv1 se tiene que aumentar para que la sobrecarga introducida sea igual a la de la versión 1 de SD-OLSRv1 cuando los nodos sólo hacen peticiones de servicios periódicas.

4.3.2. Tasa de descubrimiento de servicios con SD-OLSRv1

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv1 referentes a la tasa de descubrimiento de servicios, hemos considerado los mismos escenarios simulados en la sección anterior para ver qué porcentaje de servicios se descubre en cada uno de los escenarios y analizar así, la viabilidad del mecanismo de descubrimiento de servicios definido, teniendo en cuenta la sobrecarga introducida en la red y la tasa de descubrimiento de servicios.

En las figuras 4.13, 4.14 y 4.15 se muestran los resultados para los tres escenarios simulados.

Escenario 1

En la tabla 4.14 se muestra un resumen con los parámetros de la simulación.

Tal como observamos en la figura 4.13, la tasa de descubrimiento de servicios cuando los nodos anuncian los servicios cada segundo es prácticamente del 100 %, independientemente de la versión de SD-OLSRv1 que consideremos. Tal como hemos visto en la sección anterior, en la versión 1, los mensajes de petición de servicio no progresan en la red, ya que los nodos que integran la misma tienen un conocimiento preciso de los servicios que se anuncian, sin necesidad de enviar ninguna solicitud de servicio. A efectos prácticos, ambas versiones cuando los nodos anuncian sus servicios cada segundo se comportan igual y prevalece el mecanismo de descubrimiento de servicios tipo *push* de inundar la red de mensajes de anuncio de servicios, sobre el mecanismo tipo *pull* de la versión 1 de SD-OLSRv1.

Según se observa en la figura 4.14, cuando los servicios se anuncian cada 5 segundos la diferencia entre la versión 1 y la 2 sólo se aprecia cuando la red se hace

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	cada 1, 5, 10 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 1, 5, 10 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 4.14: Parámetros de simulación Escenario 1, tasa de descubrimiento de serv

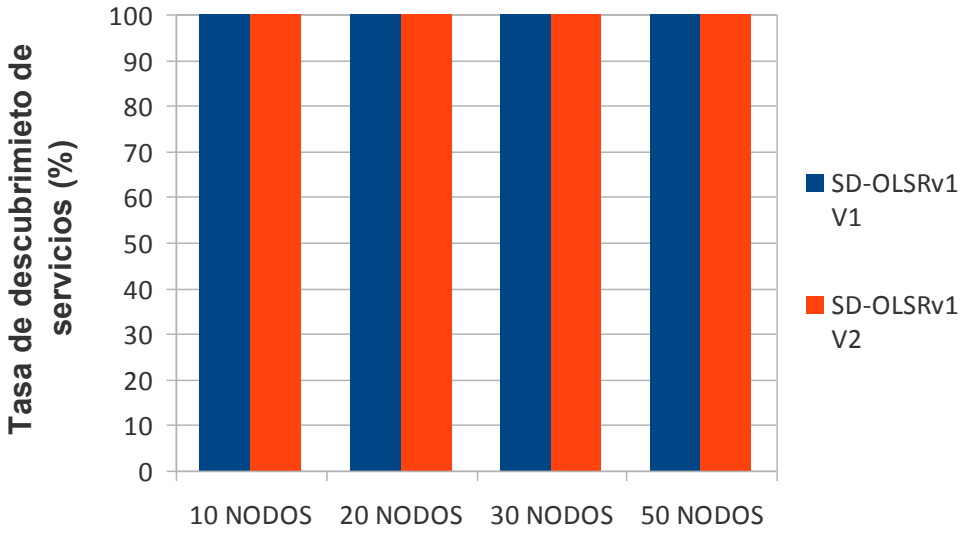

Figura 4.13: Tasa de descubrimiento de servicios con SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a segundo

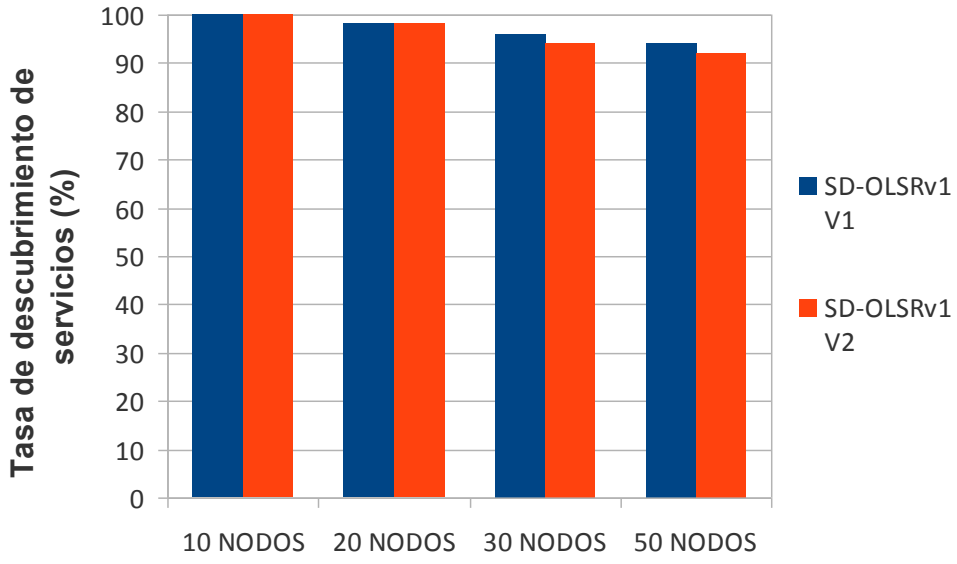

Figura 4.14: Tasa de descubrimiento de servicios con SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a 5 segundos

más grande. A medida que la red se hace más grande existen más probabilidades de enlaces rotos y de que los mensajes de anuncio de servicios que se envían cada 5 segundos no lleguen a todos los nodos de la red. Sobre todo cuando los servicios que se anuncian en la red no permanecen estables en el tiempo y siguen existiendo anuncios nuevos de servicios en el transcurso de la simulación. En este caso, la versión 1 de SD-OLSRv1 sí que ofrece una mayor tasa de descubrimiento de servicios a costa de aumentar la sobrecarga de paquetes transmitidos en la red, tal como hemos visto en la sección anterior, 4.3.1. Sin embargo no existe una correspondencia directa entre aumento en el número de paquetes transmitidos y aumento en la tasa de descubrimiento de servicios. Que aumente el número de paquetes transmitidos no significa que la tasa de descubrimiento de servicios aumente en la misma magnitud. Esto es debido a que también en redes de estas características, la petición o la respuesta el servicio se pueden perder.

De la figura 4.15 se desprende que esta diferencia de una mayor tasa de descubrimiento de servicios en la versión 1 respecto a la versión 2 es más evidente en redes de 30 nodos que en redes de 50. Es decir, cuando la red es muy grande, y se ofrecen muchos servicios, la tasa de descubrimiento de servicios en una red de 50 nodos es menor que en una red de 30 nodos con menos servicios. Esto es así porque cada 5 segundos los nodos únicamente solicitan un servicio. Cuanto

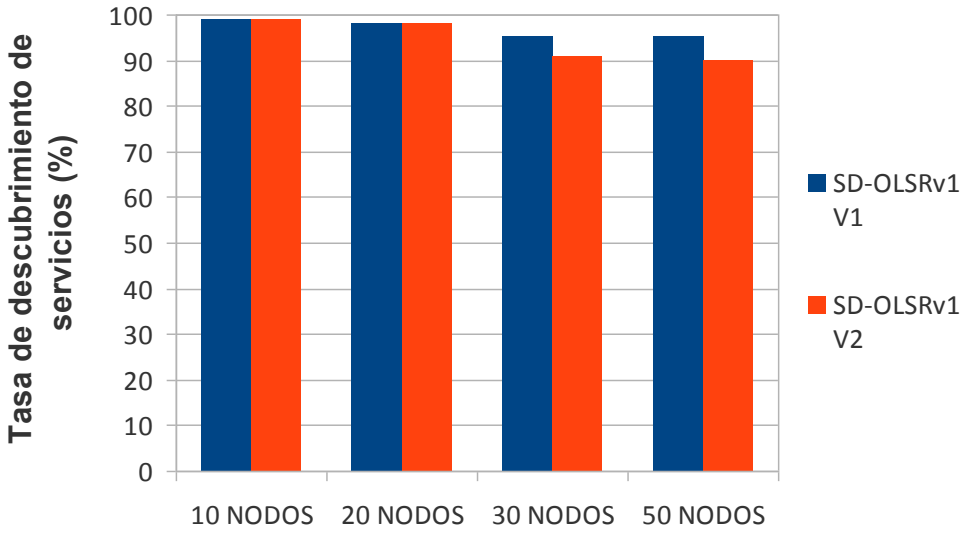

Figura 4.15: Tasa de descubrimiento de servicios con SD-OLSRv1 versión 1 y 2 con SDM_INTERVAL fijado a 10 segundos

más grande sea la red más tiempo tardarán en solicitar todos los servicios que se ofrezcan en la red.

Cuando la periodicidad de envío de anuncios de servicios se fija en 10 segundos la diferencia entre la tasa de descubrimiento de servicios con la versión 1 y la 2 es mayor que cuando la frecuencia de envío de anuncios de servicios era de 5 segundos. Aunque, en ambos casos, la diferencia no es muy grande. Y como ocurría en el caso anterior, en redes de 30 nodos se aprecia más la diferencia que en redes de 50 nodos.

Escenario 2

En la tabla 4.15 se muestra un resumen con los parámetros de la simulación.

Para este escenario simulado, en redes de 10 nodos el descubrimiento de servicios en ambas versiones de SD-OLSRv1 es del 100 %. Al tratarse de redes con pocos nodos, el tiempo que dura la simulación es suficientemente amplio como para que en la versión 1 de SD-OLSRv1 las peticiones de servicios que realizan los nodos sean suficientes para descubrir el 100 % de los servicios. En la versión 2 de SD-OLSRv1, los nodos están anunciando sus servicios cada 5 segundos, periodicidad más que suficiente para que en una red de 10 nodos, todos los nodos

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	400 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 5 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 4.15: Parámetros de simulación Escenario 2, tasa de descubrimiento de servicios SD-OLSRv1

conozcan todos los servicios que se ofrecen en la red.

Sin embargo, a medida que el tamaño de la red aumenta, la tasa de descubrimiento de servicios en la versión 1 de SD-OLSRv1 disminuye respecto a la tasa de descubrimiento de servicios en la versión 2. Cuantos más nodos integran la red, más difícil es en la primera versión de SD-OLSRv1, que la única vez que los nodos anuncian los servicios, éstos lleguen a todos los nodos. Y cuantos más nodos integran la red y más servicios se ofrecen en la misma, más difícil es que en el tiempo de simulación establecido, los nodos adquieran un conocimiento total de los servicios que se ofrecen en la red.

Estas conclusiones las extraemos de los resultados obtenidos en la figura 4.16.

Escenario 3

En la tabla 4.16 se muestra un resumen con los parámetros de la simulación.

En el escenario simulado, en redes pequeñas la tasa de descubrimiento de servicios en ambas versiones de SD-OLSRV1 es similar y próximo al 100 %.

En redes grandes, sin embargo, con la versión 1 de SD-OLSRv1 se descubren

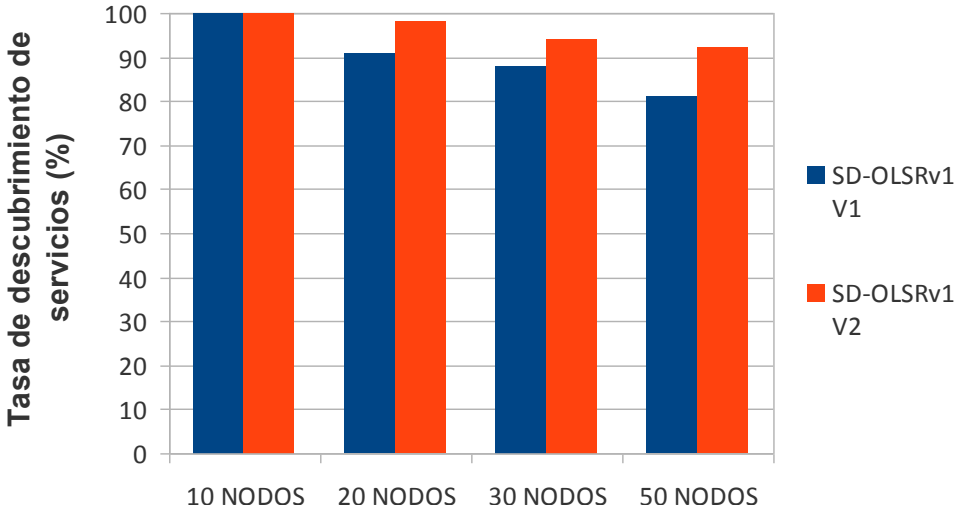

Figura 4.16: Tasa de descubrimiento de servicios con SD-OLSRv1 versión 1 y 2 en el Escenario 2

Parámetros de simulación	**Valor**
Versión NS-2	2.26
Implementación OLSRv1	UM-OLSR
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1 v1	400 s.
Petición de servicios SD-OLSRv1 v1	cada 5 s.
SDM_INTERVAL SD-OLSRv1 v2	cada 100 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 4.16: Parámetros de simulación Escenario 3, tasa de descubrimiento de servicios SD-OLSRv1

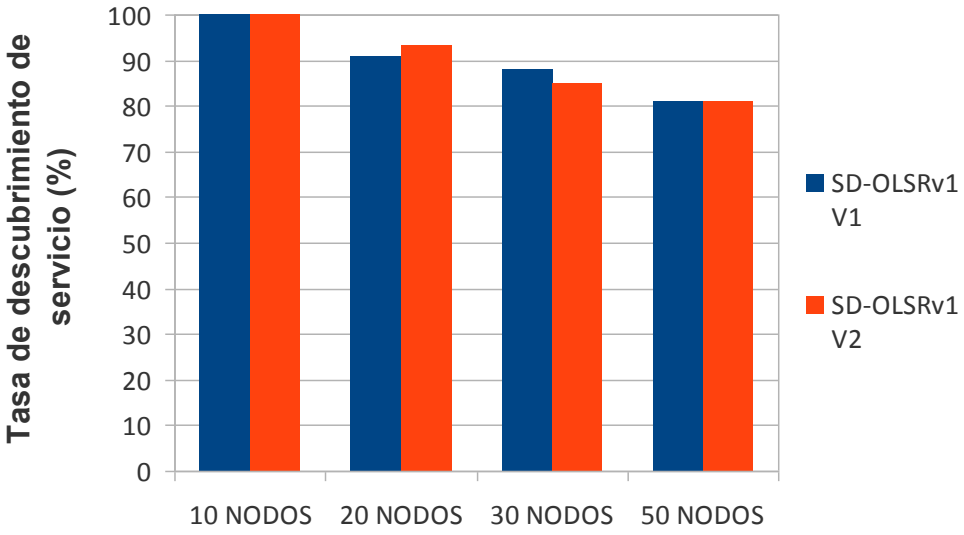

Figura 4.17: Tasa de descubrimiento de servicios con SD-OLSRv1 versión 1 y 2 en el Escenario 3

más servicios que con la versión 2. En redes tan grandes no es suficiente con que los nodos anuncien sus servicios con una periodicidad de 100 segundos, para que estos anuncios impliquen una tasa de descubrimiento de servicios alta. Aunque es cierto que cada vez que se anuncian los servicios éstos se anuncian a toda la red. Por cada vez que se anuncian servicios en la versión 2 existe un número mayor de nodos que conoce un número mayor de servicios. En la versión 1, en el escenario que estamos simulando, cuando un nodo quiere un servicio que no conoce, no espera a la siguiente actualización del servicio por parte del nodo, sino que él mismo hace una petición. Estas peticiones se realizan cada 5 segundos. Esto hace que al final de la simulación la tasa de descubrimiento de servicios en redes grandes, en la primera versión de SD-OLSRv1 sea mayor que en la segunda versión de SD-OLSRv1.

En cualquier caso, la tasa de descubrimiento de servicios en este escenario es inferior al de los escenarios simulados anteriormente.

Estos resultados los podemos observar en la figura 4.17.

Por los resultados obtenidos, observamos que para obtener una tasa alta de descubrimiento de servicios lo ideal en una red MANET con un protocolo de encaminamiento proactivo OLSR, es utilizar un mecanismo tipo *push* para descubrir servicios, y aprovechar la técnica de los nodos MPR que utiliza el protocolo

de encaminamiento OLSR, para que estos anuncios de servicios lleguen a todos los nodos de la red con la menor sobrecarga de paquetes introducida en la red posible.

Un mecanismo de descubrimiento de servicios tipo *pull*, por muchas peticiones de servicios que se realicen no iguala la tasa de descubrimiento de servicios de un método *push* y una periodicidad de envío de anuncio de servicios razonable. Este hecho es más evidente a medida que la red se hace más grande y hay más servicios que se ofrezcan en la red.

En redes pequeñas ambos mecanismos se comportan bien. Una red de 10 nodos es suficientemente pequeña como para que en 400 segundos de simulación ambos mecanismos de descubrimiento de servicios, la versión 1 con nodos haciendo peticiones cada 5 segundos siguiendo el mecanismo tipo *pull* y la versión 2 con nodos anunciando sus servicios cada 5, 10 o 100 segundos siguiendo el mecanismo tipo *push*, permitan descubrir el 100 % de los servicios que se ofrecen en la red.

4.4. Conclusiones

En este capítulo hemos descrito de forma razonada el mecanismo de descubrimiento de servicios sobre OLSRv1 y hemos analizado las prestaciones de SD-OLSRv1. Hemos comparado entre sí las dos versiones del mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSRv1. La primera versión, donde el protocolo contempla que los nodos envíen tanto peticiones de anuncio de servicios como mensajes periódicos anunciando sus servicios, y la segunda versión, donde los nodos únicamente anuncian sus servicios periódicamente a través de mensajes SDM.

Después de realizar las simulaciones y analizado los resultados, concluimos que aunque en un principio, definimos el mecanismo de descubrimiento de servicios sobre OLSRv1 como un mecanismo híbrido tipo *push-pull* con mensajes para anunciar y hacer peticiones de servicios, la conclusión que hemos sacado al comparar este mecanismo con la segunda versión propuesta, donde describimos el mecanismo como un mecanismo de descubrimiento de servicios tipo *push* puro, es que éste último se adapta mejor a las características del entorno de una red MANET con un protocolo de encaminamiento proactivo (OLSR). Sobre todo cuando el número de nodos que integran la red crece. Incluso en redes de 10 nodos, a pesar de que para ambas versiones de SD-OLSRv1 en todos los escenarios simulados la tasa de descubrimiento de servicios es cercana al 100 %, cuando la versión 1 de SD-OLSRv1 se comporta casi en exclusiva como un mecanismo tipo *pull*, la sobrecarga que se introduce en la red es ligeramente superior a la versión

2 de SD-OLSRv1 con los nodos anunciando servicios con una periodicidad de 100 segundos.

Tratándose de una red MANET OLSR proactiva, es mejor que sean los nodos que integran la red los que anuncien sus servicios periódicamente en lugar de tener lo nodos que hacer peticiones de servicio periódicas. OLSR ofrece un mecanismo de *broadcast* para envío de mensajes mejorado a través de los nodos MPR para precisamente inundar la red de mensajes a costa de una mínima sobrecarga. La periodicidad con la que se envíen los mensajes SDM será un parámetro que dependa del entorno y las características de la red en la que se implemente el mecanismo de descubrimiento.

Cuanto más frecuentemente se anuncien los servicios, mayor tasa de descubrimiento de servicios tendremos. Una periodicidad de 1 segundo supone una tasa de descubrimiento de servicios del 100 %. Una periodicidad en el envío de mensajes de 1 segundo también supone una mayor sobrecarga en el número de paquetes que se transmiten en la red. Hay que ajustar este parámetro para compensar la sobrecarga con la tasa de descubrimiento de servicios, dependiendo del entorno que se desee y las necesidades que se tengan en cuanto a tasa de descubrimiento de servicios.

La ventaja de la segunda versión de SD-OLSRv1 es que aprovechamos el carácter proactivo del protocolo de encaminamiento OLSR. Inundamos la red de mensajes de anuncio de servicios para que los nodos que integran la red, en cualquier momento, tengan un conocimiento lo más global posible de los servicios que se ofrecen. Teniendo en cuenta la movilidad de los nodos que caracteriza este tipo de redes, que los servicios se estén continuamente anunciando es una ventaja de cara a tener siempre en todos los nodos visibilidad de los servicios anunciados en la red. El inconveniente o el coste que hay que pagar por tener una red actualizada, es el incremento en el número de mensajes que se transmiten en la red. Aunque este hecho se mitiga en una red MANET con un protocolo de encaminamiento OLSR, debido precisamente, al funcionamiento del protocolo de encaminamiento. Gracias a los nodos MPR los mensajes que cada nodo envía llegan a toda la red con un número de retransmisiones mínimo.

En la primera versión de SD-OLSRv1 los mensajes de petición de servicios quedan supeditados a la frecuencia con la que un nodo anuncia sus servicios. Si la frecuencia de anuncio de servicios es razonable, los mensajes de petición de servicio raramente se llegan a enviar a la red. Si por el contrario, la frecuencia con la que un nodo anuncia sus servicios disminuye, las peticiones de servicio sí que se cursan, pero eso no garantiza que el servicio se descubra, sobre todo en redes muy grandes. Además de que el tiempo en descubrir el servicio aumenta respecto a una red donde los nodos anuncian sus servicios periódicamente. En una red

donde los nodos anuncian sus servicios periódicamente, los nodos que integran la red disponen en su caché de servicios un listado de estos servicios, mientras que en una red donde los nodos tienen que enviar un mensaje para solicitar un servicio, se emplea más tiempo en conocer el servicio, puesto que tienen que esperar la respuesta del proveedor del servicio. Otra diferencia respecto a la segunda versión de SD-OLSRv1, es que los nodos en SD-OLSRv1 no tienen un conocimiento global tan rápido de todos los servicios que se ofrecen en la red. Por cada petición que hacen descubren un nuevo servicio. Para descubrir todos los servicios necesitan más tiempo. No es como en la segunda versión de SD-OLSRv1 donde cada anuncio de servicio por parte de un nodo llega a todos los nodos de la red.

También hemos observado que cuanto más grande es la red, mejor se comporta la segunda versión de SD-OLSRv1 respecto a la primera. Cuanto más grande es la red, existen más probabilidades de pérdidas de paquetes debidas a enlaces rotos. No es lo mismo que un mensaje de anuncio de servicio se esté retransmitiendo en la red cada 1, 5, o 10 segundos, que un nodo haga la petición de un servicio una vez. Si esa petición o la respuesta a esa petición se pierde en la red, el nodo no llega a saber quién es el proveedor del servicio que busca.

Con todo esto podemos decir que, la versión 2 del mecanismo de descubrimiento de servicios propuesto mejora la versión inicial y es viable en redes MANET. La sobrecarga que introduce la definición de un nuevo tipo de mensaje SDM para anuncio de servicios depende de la periodicidad con la que se envíen mensajes SDM. Cuanto más frecuentemente se envíen los anuncios de servicio mayor sobrecarga se generará. Pero incluso con periodicidades muy altas, la tasa de descubrimiento de servicios es grande.

Capítulo 5

Diseño y evaluación del protocolo SD-OLSRv2

Tal como hemos visto en el capítulo 4, SD-OLSRv2 comparte características comunes con SD-OLSRv1:

- SD-OLSRv2 es un mecanismo de descubrimiento de servicios tipo *push* integrado en el protocolo de encaminamiento OLSR. SD-OLSRv2 se integra en la versión 2 de OLSR.
- SD-OLSRv2 aprovecha la naturaleza proactiva del protocolo OLSR y también el método de reenvío de paquetes mediante nodos MPR, propio de OLSR, para anunciar periódicamente los servicios disponibles en cada nodo.
- SD-OLSRv2 es un mecanismo de descubrimiento de servicios totalmente distribuido, cuyo funcionamiento no depende de directorios.
- La utilización del formato del paquete definido en OLSRv2 para encapsular los anuncios de servicios por parte de los nodos, hace posible el correcto funcionamiento del mecanismo de descubrimiento de servicios, aunque no todos los nodos que forman parte de la red MANET lo tengan implementado.

En la sección 5.1 describimos el mecanismo de descubrimiento de servicios SD-OLSRv2 y en la sección 5.2 cómo lo hemos implementado sobre el protocolo de encaminamiento OLSRv2 dentro del simulador de red NS-2 versión 2.34. A continuación, en la sección 5.3, analizamos en detalle sus prestaciones. En concreto, analizaremos las prestaciones de SD-OLSRv2 midiendo la sobrecarga introducida

en la red, el tiempo en descubrir un servicio, la tasa de descubrimiento de servicios y la tasa de descubrimientos falsos. Todo ello dependiendo del número de nodos, la velocidad media a la que se mueven los nodos y el número de nodos que hay ofreciendo el mismo servicio en la red.

5.1. Descripción de SD-OLSRv2

El funcionamiento del mecanismo de descubrimiento de servicios integrado en OLSRv2 no requiere la definición de un nuevo tipo de mensaje para que los nodos anuncien sus servicios como ocurría en SD-OLSRv1. En este caso, los nodos que tengan uno o varios servicios que anunciar lo harán en los mensajes *Hello* que cada nodo envía periódicamente. Estos mensajes *Hello* incluirán una estructura TLV con el nombre y la descripción del servicio que se anuncia. Los mensajes *Hello* los reciben todos los nodos que se encuentren a un salto de distancia del nodo que envía el mensaje. Estos nodos guardarán en una caché de servicios, la dirección del nodo origen que anuncia el servicio, el tipo de servicio que se anuncia, una breve descripción del mismo y el tiempo de vida que ese servicio estará disponible en la red. De todos los nodos que se encuentran a un salto del nodo que anuncia el servicio, sólo los nodos MPR serán los encargados de retransmitir ese servicio a la red, y lo harán en los mensajes TC. Los mensajes TC también incluirán la estructura TLV con el nombre y la descripción del servicio a reenviar. Por lo tanto, en los mensajes TC viaja la información de los servicios de los nodos que han elegido al nodo como MPR. Recordamos, que cada nodo selecciona de entre sus vecinos a un salto su conjunto de nodos MPR. En el protocolo OLSRv2, una de las responsabilidades de los nodos MPR es retransmitir los mensajes de difusión de los nodos selectores de tal forma que esa información se difunda por toda la red. En el caso de SD-OLSRv2, también van a retransmitir los servicios que los nodos selectores quieran anunciar. Cuando un nodo necesite un servicio en concreto buscará en su caché de servicios qué nodo le puede proporcionar el servicio.

También hemos aprovechado este nuevo diseño para mejorar el mecanismo de descubrimiento de servicios definido en SD-OLSRv1:

- Al igual que ocurría en SD-OLSRv1, cuando un nodo abandona la red envía un mensaje de notificación con el tiempo de vida del servicio o servicios anunciados fijado a 0, indicando que el servicio o los servicios que ese nodo anuncia dejan de estar disponibles. La diferencia con SD-OLSRv1 es que en SD-OLSRv2 los nodos consultarán también la información de las tablas de encaminamiento para ver cuándo un nodo abandona la red y eliminar

así los servicios que se mantienen vivos en la caché de servicios referentes a nodos que ya no están accesibles.

- Utilizamos los atributos ya definidos en la RFC 5497, [Clausen et al, 2009], VALIDITY_TIME e INTERVAL_TIME para configurar el tiempo de vida de un servicio en la red y la periodicidad con la que un nodo va a anunciar sus servicios.

5.1.1. Formato de la estructura TLV de anuncio de servicios

Tal como hemos visto en la sección 3.3.3 el protocolo de encaminamiento OLSRv2 ofrece una serie de ventajas sobre OLSRv1 que hacen pensar que el mecanismo de descubrimiento de servicios extendido a OLSRv2 mejorará los resultados obtenidos con SD-OLSRv1:

- OLSRv2 sigue la RFC 5444 para la definición de paquetes y mensajes y por lo tanto es un protocolo más modular que OLSRv1.
- La cabecera de los paquetes OLSRv2 permite discriminar fácilmente si un nodo tiene que procesar o reenviar el paquete cuando lo recibe.
- Los paquetes y mensajes en OLSRv2 pueden complementarse con atributos que siguen una estructura TLV.
- Para averiguar el camino más corto hacia un destino, OLSRv2 no utiliza la métrica del menor número de saltos entre el origen y el destino sino el estado del enlace.

Experimentalmente también se ha demostrado que el comportamiento de OLSRv2 mejora respecto a OLSRV1 [Dahiya et al, 2013]:

- El *throughput* en OLSRv2 es significativamente mayor que en OLSRv1, entendido como número de paquetes transmitidos con éxito por unidad de tiempo.
- El *average end-to-end-delay* (tiempo medio que tarda un paquete en llegar al destino) es menor en OLSRv2.
- El *average jitter* (variación en la velocidad de transmisión de los paquetes de datos) es un 50 % menor en OLSRv2 que en OLSRv1.

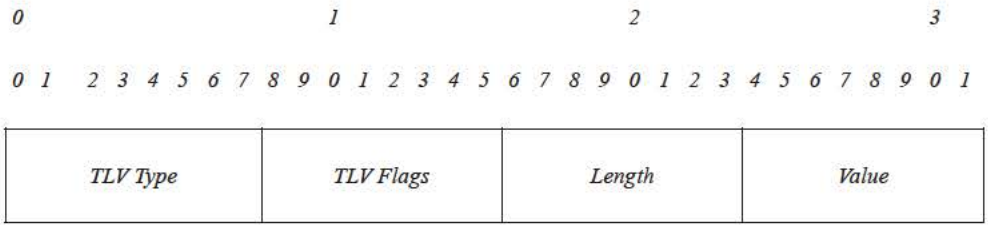

Figura 5.1: Estructura TLV

- El consumo de batería de los nodos al final de la simulación para un mismo escenario es menor en OLSRv2, lo que también supone una mejora de OLSRv2 respecto a OLSRv1, dada la limitación en cuanto a batería de los nodos que componen una red MANET.

Para integrar el mecanismo de descubrimiento de servicios sobre OLSRv2 utilizamos la estructura TLV estandarizada en la RFC 5497. Será necesario definir dos nuevas estructuras TLV para que un nodo anuncie los servicios que tiene disponibles: SDM_TYPE y SDM_DESCRIPTION. Cada estructura TLV tiene el formato de la figura 5.1.

En la figura 5.2 se muestra el formato del bloque TLV con los 4 TLVs necesarios para que un nodo anuncie un servicio, los dos que hemos definido nuevos, SDM_TYPE y SDM_DESCRIPTION y los dos que ya están definidos en la RFC 5497, VALIDITY_TIME e INTERVAL_TIME para configurar el tiempo de vida de un servicio en la red y la periodicidad con la que un nodo va a anunciar sus servicios.

Este bloque TLV se añade a los mensajes *Hello* siempre que un nodo tenga un servicio que anunciar y a los mensajes TC siempre que un nodo MPR retransmita el servicio anunciado por otro nodo.

El formato del paquete OLSRv2 con un mensaje *Hello* y un mensaje TC con la estructura TLV de anuncio de servicios se muestra en las figuras 5.3 y 5.4.

5.1.2. Generación del TLV SDM_TYPE

Los servicios que anuncie cada nodo formarán parte de los mensajes *Hello* y los mensajes TC definidos en el protocolo OLSRv2. Teniendo en cuenta el tamaño de la cabecera OLSRv2 y el tamaño de los mensajes *Hello* y TC, el tamaño de un paquete OLSRv2 que envíe un nodo anunciando un servicio será el siguiente:

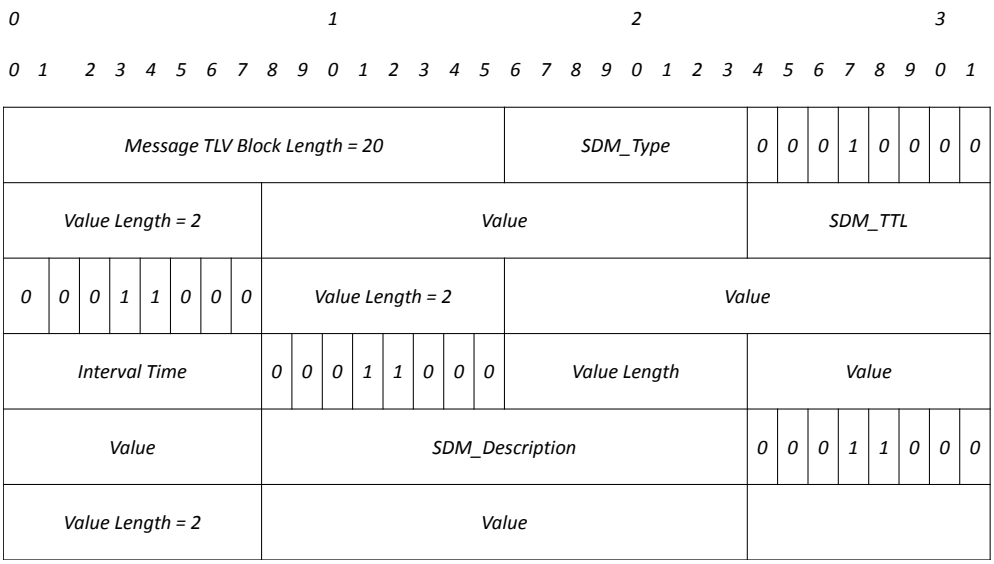

Figura 5.2: Bloque TLV para anuncio de servicios en SD-OLSRv2

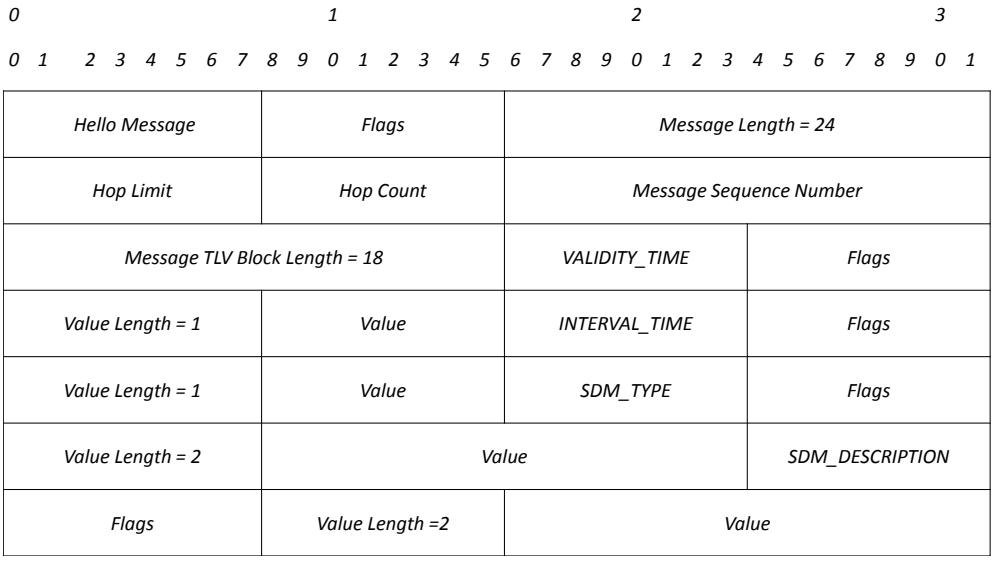

Figura 5.3: Mensaje *Hello* con anuncio de servicios en SD-OLSRv2

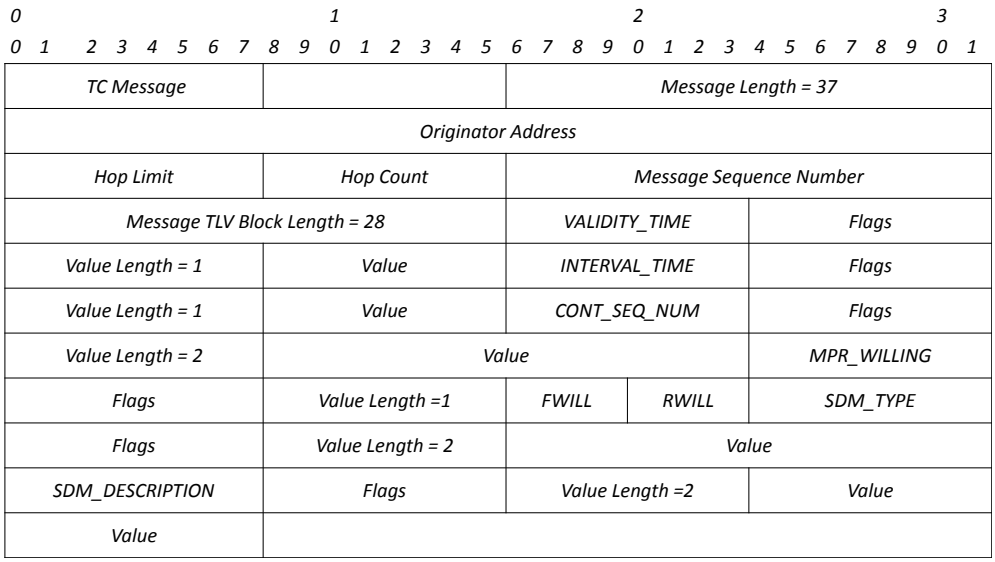

Figura 5.4: Mensaje TC con anuncio de servicios en SD-OLSRv2

```
Tamaño de un paquete con un mensaje Hello y anuncio de servicios en OLSR v2:
   Cabecera paquete OLSRv2 + Mensaje Hello + SDM_TYPE TLV = 34 bytes

Tamaño de un paquete con un mensaje TC y anuncio de servicios en OLSR v2:
   Cabecera paquete OLSRv2 + Mensaje TC + SDM_TYPE TLV = 39 bytes
```

En el caso de OLSRv1 cuando se transmitían mensajes *Hello* y mensajes SDM, estos suponían una sobrecarga en la red de 54 bytes. Cuando se transmitían mensajes TC y mensajes SDM, la sobrecarga introducida era de 52 bytes. Es decir, en OLSRv2 hemos reducido el ancho de banda consumido cuando un nodo tiene que anunciar servicios.

Los mensajes *Hello* con la estructura TLV de anuncio de servicio se enviarán siempre que:

- Un nodo tenga un nuevo servicio que anunciar.
- Un nodo abandone la red, en cuyo caso la estructura TLV VALIDITY_TIME tendrá un valor 0.
- Periódicamente cuando la estructura TLV INTERVAL_TIME haya vencido.

Cuanto menor sea el tiempo definido en el atributo INTERVAL_TIME, el resto de nodos de la red tendrán su caché de servicios con datos de servicios

<Service Name>	*<Lifetime>*	*<Service Description>*
Printer	100seconds	Color Printer

Figura 5.5: Caché de servicios locales en SD-OLSRv2

más precisos. A cambio el ancho de banda consumido en la red será mayor. Al contrario, cuanto mayor sea el tiempo definido en el atributo INTERVAL_TIME, menor será el ancho de banda consumido en la red, pero los nodos no tendrán su caché de servicios lo más actualizada posible. Esto implica que sea más común que se produzcan falsos descubrimientos.

5.1.3. Reenvío del TLV SDM_TYPE

Sólo los nodos definidos como MPR serán los encargados de retransmitir los mensajes que se anuncien en la red. Los anuncios de servicios se retransmiten como atributos definidos en una estructura TLV como parte de los mensajes TC.

5.1.4. Procesado del TLV SDM_TYPE y caché de servicios

Cada nodo dispone de dos cachés, una donde almacena los servicios que él proporciona al resto de nodos, y otra caché de servicios donde almacena los servicios que anuncian otros nodos en la red. A la primera la llamaremos caché de servicios locales y a la segunda la llamaremos caché de servicios remotos. La caché de servicios locales contiene la información que se muestra en la figura 5.5.

El parámetro *Service Name* indica el nombre del servicio que el nodo está anunciando. El parámetro *Lifetime*, el tiempo que ese servicio estará disponible en la red y por último el parámetro *Service Description*, muestra una descripción del servicio. Se almacenarán tantas entradas como servicios anuncie un nodo. La caché de servicios remotos se rellena con la información que recibe cada nodo procesando los mensajes *Hello* y TC que lleven la estructura TLV correspondiente a anuncios de servicios. Esta caché de servicios remotos contiene la información que se muestra en la figura 5.6.

Para el caso de que un nodo reciba un mensaje que ya tenía almacenado en su caché de servicios o para el caso de que el servicio anunciado coincida con uno de los servicios que el propio nodo dispone, el procedimiento de actuación es el

<Service Name>	*IP Address*	*<Lifetime>*	*<Service Description>*
Printer	x x.x.x	100 seconds	Color Printer

Figura 5.6: Caché de servicios remotos en SD-OLSRv2

mismo que el descrito para SD-OLSRv1 en la sección 4.1.3.

Siempre que un nodo tenga un servicio que anunciar o tenga que enviar una actualización de un servicio ya anunciado, utilizará los mensajes *Hello* a través de la estructura TLV definida. Los mensajes *Hello* tienen una periodicidad definida en el atributo INTERVAL_TIME. Ya hemos dicho anteriormente que cuanto menor sea este tiempo más precisa será la información que almacenen los nodos en cuanto a servicios disponibles en la red.

5.1.5. Descubrimiento de servicios

Cuando un nodo quiere hacer uso de un servicio, comprueba en su caché de servicios locales si ese servicio lo tiene él disponible. En caso negativo, busca en la caché de servicios remotos para ver si algún nodo en la red ofrece ese servicio. Si en ninguna de estas cachés hay una entrada para ese servicio, el servicio es inaccesible en el momento de la búsqueda. Si por el contrario, sí que hay una entrada para ese servicio, comprobamos qué nodo ofrece ese servicio.

Para garantizar que ese nodo sigue estando disponible y reducir así la tasa de falsos descubrimientos consultamos la información que cada nodo almacena en la *Topology Information Base*. Tal como hemos visto en la sección 3.3.3, esta información se actualiza con la información recibida en los mensajes TC en el *Advertising Remote Router Set*, el *Router Topoly Set* y el *Routable Address topology Set*. Además cada nodo mantiene un *Routing Set* que se actualiza con la información guardada en la *Topology Information Base*, *Local Information base* y *Neighbor Information Base*.

El primer conjunto, *Advertising Remote Router Set*, guarda información sobre qué nodos remotos transmiten mensajes TC. Cada entrada tiene el siguiente formato: AR_orig_addr, AR_seq_number, AR_time. Cuando expira el tiempo definido en el parámetro AR_time, la entrada es eliminada. En *Router Topoly Set*, se guarda información topológica referente a los enlaces entre los nodos que componen la MANET. Cada entrada contiene la siguiente información: TR_from_orig_addr, TR_to_orig_addr, TR_seq_number, TR_metric,

TR_time. TR_from_orig_addr es la dirección del nodo que puede alcanzar al nodo TR_to_orig_addr en un salto. Aquí también la entrada desaparece cuando expira el tiempo TR_time. En *Routable address Topology Set*, se guarda información topológica sobre las rutas de la red, incluye información sobre a través de qué nodos alcanzar un destino: TA_from_orig_addr, TA_dest_addr, TA_seq_number, TA_metric, TA_time. TA_from_orig_addr es la dirección origen de un nodo que puede alcanzar TA_dest_addr en un salto. Cuando expira el tiempo marcado en TA_time, la entrada desaparece. Por último el *Routing Set*, guarda información sobre el primer salto de un camino hacia cada nodo destino alcanzable por ese nodo. Cada entrada contiene: R_dest_addr, R_next_iface_addr, R_local_iface_addr, R_dist, R_metric. R_dest_addr, es la dirección del nodo destino; R_next_iface_addr, es la dirección de red del "siguiente salto" en el camino hacia el destino. R_local_iface_addr, la dirección origen a través de la cual se envía el paquete hacia el destino. R_dist, número de saltos hasta el destino. Esta información se actualiza (se añade o se eliminan entradas) con información recogida de *Link Sets*, *Neighbour Sets* y *Routable address Topology Set*. Por otro lado si una entrada se elimina de estos conjuntos, significa que ese nodo deja de estar disponible, o la ruta hacia él deja de estar disponible. Eliminamos también la entrada de la tabla de servicios referente al servicio ofrecido por ese nodo.

5.1.6. Indisponibilidad de un servicio

Para reducir la tasa de falsos descubrimientos, además de consultar la tabla de información topológica, cuando un nodo abandona la red y por lo tanto, sus servicios dejan de estar disponibles para el resto de nodos, envía un mensaje de notificación informando que sus servicios dejan de estar disponibles. Esto también lo hace a través de las estructuras TLV. Anunciará el servicio que deja de estar disponible y fija el tiempo de vida de ese servicio VALIDITY_TIME a 0. Cuando el resto de nodos reciban este mensaje, eliminarán de su caché de servicios el o los servicios asociados.

5.2. Implementación del mecanismo de descubrimiento de servicios SD-OLSRv2

En esta sección describimos la implementación del mecanismo de descubrimiento de servicios sobre el protocolo de encaminamiento OLSRv2, dentro del simulador de red NS-2 versión 2.34. La diferencia en cuanto a la versión del simulador NS-2 que utilizamos para evaluar las prestaciones de SD-OLSRv2, versión

2.34, respecto a la versión que utilizamos para evaluar las prestaciones de SD-OLSRv1, radica en el salto temporal que existe entre ambas implementaciones del mecanismo de descubrimiento de servicios.

El mecanismo de descubrimiento de servicios sobre OLSRv2 lo hemos desarrollado utilizando el lenguaje de programación Java. La elección del lenguaje de programación Java, viene impuesta porque hemos partido de una implementación ya desarrollada en Java para OLSRv2, [Bross, 2013]. La implementación se ha llevado a cabo utilizando Java SE 1.7 y las librerías de comunicación estándar proporcionadas por dicho lenguaje.

A continuación describimos en detalle los pasos que hemos seguido para su implementación.

5.2.1. Necesidad de la herramienta AgentJ

Para simular en NS-2 el mecanismo de descubrimiento de servicios que hemos implementado en Java necesitamos la herramienta AgentJ [Taylor et al, 2006]. AgentJ traslada los comandos Java de nuestro protocolo SD-OLSRv2 a comandos C++ que son los que entiende NS-2, y redirige las llamadas a sistema del protocolo en Java a sus equivalentes en C++, de tal forma que permite la ejecución del código Java en cada nodo de la red NS-2.

Gracias a esta herramienta, no es necesario modificar el código del mecanismo de descubrimiento de servicios hecho en Java para simularlo sobre NS-2. Tampoco es necesario, como sí ocurría con la implementación hecha en C++ de SD-OLSRv1, realizar ninguna modificación y posterior compilación del simulador NS-2 como por ejemplo, para que reconozca el nuevo mecanismo de descubrimiento desarrollado.

En la figura 5.7 se muestra la arquitectura de una aplicación Java que corre sobre un nodo NS-2 gracias a la herramienta AgentJ. La parte a la izquierda de la línea de puntos está implementada en el lenguaje de programación C++ y en el lenguaje interpretado OTcl. Esta parte incluye el nodo NS-2 y el agente AgentJ escrito en C++, que tiene que ser enlazado con el nodo NS-2. Por otro lado, para poder lanzar la aplicación Java y enviarle comandos desde el simulador NS-2, necesitamos un punto de entrada similar a una función *main()*. Este punto de entrada es una clase auxiliar Java que es la encargada de *enlazar* la aplicación Java con el simulador NS-2.

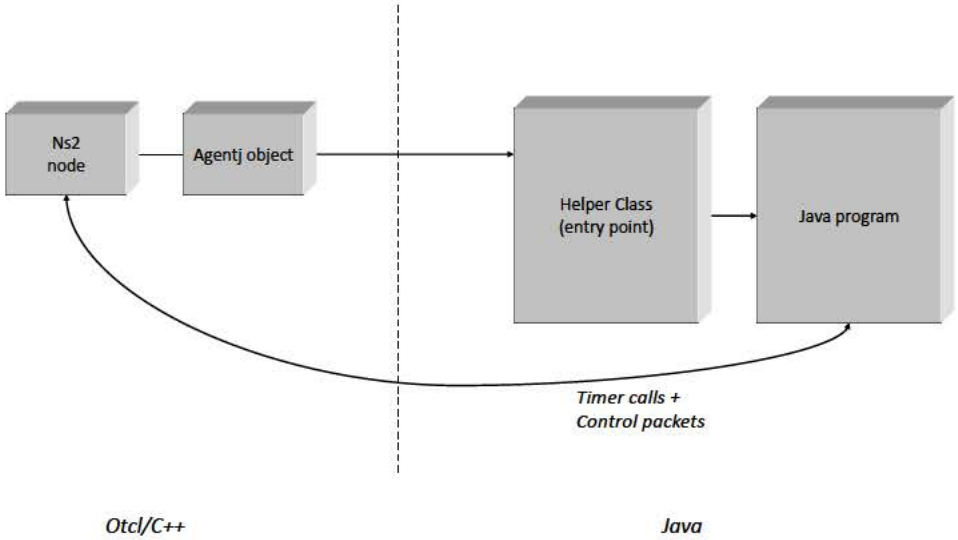

Figura 5.7: Arquitectura de un agente AgentJ que se asigna a un nodo NS-2 [Herberg U., 2009]

5.2.2. Clase Java *SDOLSRv2RoutingAgent*

Como ya hemos visto, necesitamos una clase Java auxiliar encargada de ejecutar la implementación del mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSRv2 sobre NS-2. A esta clase auxiliar Java la hemos llamado *SDOLSRv2RoutingAgent*

En la figura 5.8 se muestra cómo la clase *SDOLSRv2RoutingAgent* simula el método *main* de una clase Java. Esta clase se define como una extensión de la clase *AgentJAgent* y como una implementación de la interfaz *AgentJRouter* implementadas en la herramienta AgentJ.

```
public class SDOLSRv2RoutingAgent extends AgentJAgent implements AgentJRouter{}
```

Al tratarse *SDOLSRv2RoutingAgent* de una clase que implementa la interfaz *AgentJRouter*, ésta tiene que proveer la implementación de cada uno de los métodos contenidos dentro de la interfaz *AgentJRouter*: *getNextHop(int)*, *getRoutingPort()* y *command(String, String[])*.

La clase *AgentJRouter* implementada en C++ es la encargada de recibir los paquetes que llegan al agente de encaminamiento, tal como se define en el apartado A.1.3. Cuando se trata de un paquete saliente de datos o control que el nodo

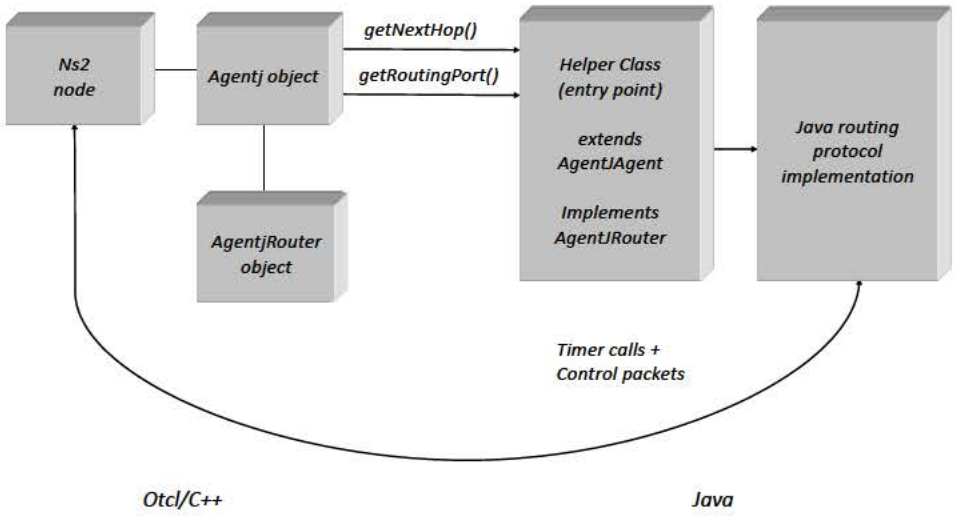

Figura 5.8: Clase Java *SD-OLSRv2RoutingAgent* [Herberg U., 2009]

tiene que encaminar, se comprueba si se trata de un paquete *unicast* o *broadcast*. Si se trata de un paquete *broadcast*, se envía a la capa de enlace. Si se trata de un paquete unicast, se llama al método *getNextHop(int)* del protocolo de encaminamiento Java (Cuadro 5.1). Este método devuelve el siguiente salto al nodo que recibe el paquete o -1 si no encuentra el siguiente salto en la tabla de rutas, lo que se refleja en el simulador NS-2 como un paquete perdido, *DROP*. En la figura 5.9 se muestra gráficamente el proceso.

Los paquetes entrantes los recibe el agente *AgentJ* y los reenvía al agente *AgentJRouter* implementado en C++. Si se trata de un paquete de control, los datos se envían por un socket al protocolo de encaminamiento OLSRv2 desarrollado en Java. El método *getRoutingPort()* devuelve el puerto UDP sobre el que está corriendo el agente de encaminamiento Java para enviar los datos por el socket (Cuadro 5.2). En la figura 5.10 se muestra gráficamente el proceso.

El parámetro *PORT* lo definimos como una constante de la clase *Constants* (Cuadro 5.3).

Al método *command(String, String[])* lo llama el agente AgentJ desde NS-2 y sirve como punto de entrada para poder ejecutar desde el simulador NS-2 comandos implementados en el protocolo de encaminamiento Java. Uno de los comandos obligatorios que deben existir es el de arrancar el protocolo de encaminamiento OLSRv2. Utilizamos el comando *startRoutingProtocol* para tal efecto (Cuadro 5.4).

```
public int getNextHop(int destination) {
    try {
        AgentJVirtualMachine.setCurrentNodeforAgent(this);
        InetAddress inetDest = InetAddress.getByName("0.0.0."  destination);

        if (_node == null || _node.getRoutingSet() == null)
            return -1;

        RoutingTuple tuple = _node.getRoutingSet().getDestination(
        inetDest);

        if (tuple == null){
            return -1;
        }

        int ns2Address = tuple.getNexthop().getAddress()[3];

        return ns2Address;
    } catch (UnknownHostException ex) {
      }
    return -1;
}
```

Cuadro 5.1: Método *getNextHop* del protocolo SD-OLSRv2

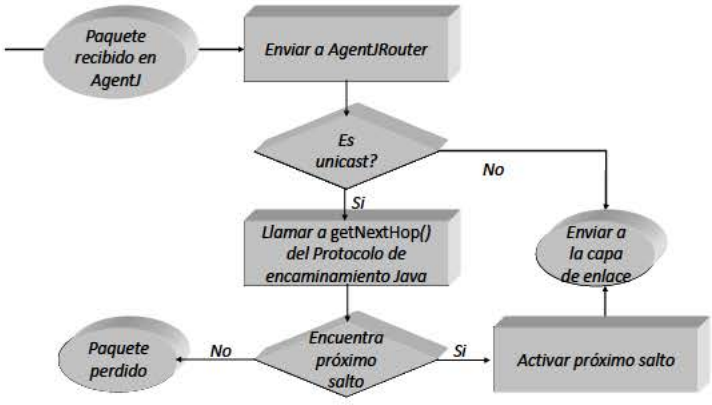

Figura 5.9: Tratamiento de Paquetes salientes [Herberg U., 2009]

```
public int getRoutingPort() {
    return Constants.PORT;
}
```

Cuadro 5.2: Método *getRoutingPort* del protocolo SD-OLSRv2

```
public final class Constants {
    private Constants(){}

    public static final int PORT = 255;

}
```

Cuadro 5.3: Definición del parámetro *PORT* en SD-OLSRv2

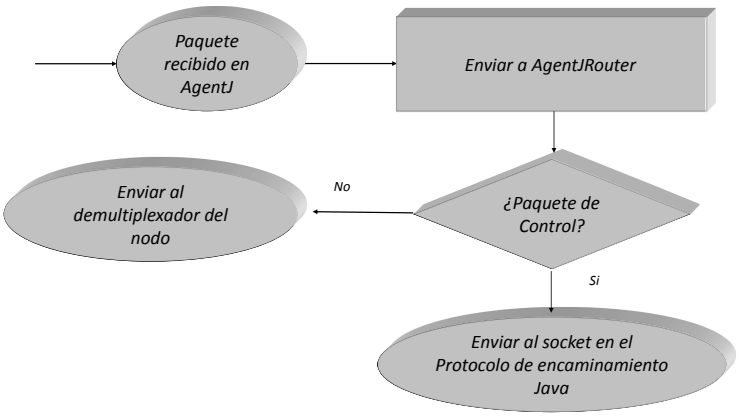

Figura 5.10: Tratamiento de Paquetes entrantes al protocolo de encaminamiento [Herberg U., 2009]

```
public boolean command(String command, String[] args) {
    if (command.equals("startRoutingProtocol")) {
        _logger.info("Agent starting ... ");

        _node = new OLSRv2Router();
        _node.initialize();

        return true;
    } else {
        _logger.warning("UNKNOWN COMMAND: " + command  +"; args: " +  args);
    }

    return false;
}
```

Cuadro 5.4: Método *command* de SD-OLSRv2

```
public static void main(String[] args){
    OLSRv2Router router = new OLSRv2Router();
    router.initialize();
}
```

Cuadro 5.5: Llamada al método *initialize* de SD-OLSRv2

El método *initialize* pertenece a la clase *OLSRv2Router* que es la clase principal de la implementación en Java del protocolo OLSRv2 (Cuadro 5.5).

5.2.3. Envío de mensajes de descubrimiento de servicios

El mecanismo de envío de mensajes de anuncio de servicios en OLSRv2 es similar al mecanismo definido para OLSRv1. La diferencia está en que en la versión 2 del protocolo en lugar de definir un nuevo mensaje de anuncio de servicios, los servicios se anuncian como una estructura TLV que se añade a los mensajes *Hello* y TC del protocolo OLSRv2. Por ello, lo primero que tenemos que hacer es definir dos nuevas estructuras TLV para que un nodo anuncie los servicios que tiene disponibles: SDM_TYPE y SDM_DESCRIPTION, además de las dos estructuras que ya define el protocolo: TLV_VALIDITY_TIME y TLV_INTERVAL_TIME (Cuadro 5.6). Al igual que en SD-OLSRv1, como descripción del servicio enviaremos un valor constante en todos los anuncios.

Como ya se ha explicado, cuando desde el script Tcl se arranca el protocolo

```
public final class Constants{
    private Constants(){}

    public static final int TLV_VALIDITY_TIME = 2000;
    public static final int TLV_INTERVAL_TIME = 3000;
    public static final int TLV_SDM_DESCRIPTION = 1000;
```

Cuadro 5.6: Definición de las estructuras TLV SDM_TYPE y SDM_DESCRIPTION de SD-OLSRv2

OLSRv2 con el comando:

```
$ns_ at 0.0 "[$node_($i) set ragent_] agentj startProtocol"
```

lo que se hace es llamar a la función *initialize* (Cuadro 5.7) de la clase *OLSRv2Router.java* que lo que hace es arrancar el protocolo OLSRv2, empezar a procesar los mensajes que se reciben y enviar periódicamente mensajes *Hello* (Cuadro 5.8) y mensajes *Tc*.

Cuando se genera el mensaje *Hello* hay que incluir, en caso de que el nodo quiera anunciar un servicio o el temporizador haya vencido, el nombre y la descripción de los servicios que haya que anunciar (Cuadro 5.9).

Una vez construido el mensaje *Hello* hay que calcular su tamaño para poder calcular la sobrecarga que se introduce en la red cuando se incluye el mecanismo de descubrimiento de servicios.

```
byte[] bytes = hello.getBytes();
```

Para poder evaluar la viabilidad del mecanismo de descubrimiento de servicios, extraemos del fichero de trazas generado en la simulación de un escenario, los valores que necesitamos para calcular la sobrecarga que el mecanismo de descubrimiento de servicios introduce en la red, la tasa de descubrimiento de servicios, la tasa de descubrimientos falsos y también el tiempo en descubrir un servicio. Para ello tenemos que modificar el fichero de trazas para incluir la información referente a los mensajes *Hello* y mensajes *Tc* que llevan integrado un anuncio de servicios.

Para modificar el fichero de trazas e incluir la información referente al nuevo mecanismo, tenemos que modificar el fichero *cmu-trace.cc* del simulador NS-2 (Cuadro 5.10).

```
/**
 * This method must be called after the router object has been created
 */
public void initialize(){
     _receiver.start(); // start receiving messages from a socket

     try {
         _origAddress = InetAddress.getLocalHost();
     } catch (UnknownHostException e2) {
           e2.printStackTrace();
           System.exit(1);
       }

     _helloTask = new HelloTask(); // this thread will periodically send HELLOs
     _tcTask = new TcTask(); // same for TCs

     _helloTask.start();
     _tcTask.start();
```

Cuadro 5.7: Método *initialize* de SD-OLSRv2

```
private class HelloTask extends Thread {

     public void run() {
         while (true){
             Message newHelloMsg = createHelloMessage();
         }
     }
 }
```

Cuadro 5.8: Mensajes *Hello* de SD-OLSRv2

```
Message hello = new Message(Message.HELLO);

TlvBlock msgTlvBlock = new TlvBlock(message);

Tlv valid = new Tlv (message, VALIDITY_TIME);
valid.setValue = TLV_VALIDITY_TIME;
msgTlvBlock.add(valid);

Tlv valid = new Tlv (message, INTERVAL_TIME);
valid.setValue = TLV_VALIDITY_TIME;
msgTlvBlock.add(valid);

Tlv valid = new Tlv (message, SDM_TYPE);
valid.setValue = service;
msgTlvBlock.add(valid);

Tlv valid = new Tlv (message, SDM_DESCRIPTION);
valid.setValue = TLV_SDM_DESCRIPTION;
msgTlvBlock.add(valid);

hello.setMsgTlvBlock(msgTlvBlock);
```

Cuadro 5.9: Mensaje *Hello* con atributo de anuncio de servicos de SD-OLSRv2

```
void CMUTrace::format(Packet* p, const char *why){
    hdr_cmn *ch = HDR_CMN(p);
    int offset = 0;
    ...
    switch(ch->ptype()) {
        case PT_AGENTJ:
            format_OLSRv2Routing(p, offset);
        break;
```

Cuadro 5.10: Moficiación en el fichero de trazas de SD-OLSRv2

```
DatagramPacket packet = new DatagramPacket(buf, buf.length);
_socket.receive(packet);

Message message = new Message(packet.getData(), packet.getLength());

if (message.getMsgType() == Message.HELLO){
    _router.processHello(message);

}
else if (message.getMsgType() == Message.TC)
    _router.processTc(message);
```

Cuadro 5.11: Comprobar mensaje en OLSRv2

Lo siguiente es añadir el método *format_OLSRv2Routing(Packet*,int)* al fichero *cmu-trace.cc*. Este método es similar al método que se añade en la implementación de OLSRv1 para añadir el mensaje SDM tal como se describió en la sección 4.2.1. Un ejemplo de trazas es el siguiente.

```
s -t 0.004000000 -Hs 39 -Hd -2 -Ni 39 -Nx 1216.36 -Ny 594.63 -Nz
 0.00 -Ne -1.0 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0
 -Is 39.9999 -Id -1.50000 -It AGENTJ -Il 25 -If 0 -Ii 0 -Iv 254
 -P olsrv2 -Pn 1 + -Ps 7996 [-Pt HELLO -Po 1 -Ph 1 -Pms 4104]
```

Estas modificaciones sí que obligan a recompilar el simulador NS-2.

Por último, al igual que ocurre en SD-OLSRv1 las memorias cachés las vamos a simular haciendo uso de ficheros que se ubicarán en el directorio */home*. Su diseño es exactamente igual al definido para OLSRv1.

5.2.4. Procesado de mensajes de descubrimiento de servicios

Cuando un nodo recibe un mensaje lo primero que hace es comprobar de qué tipo es este mensaje, si se trata de un mensaje *Hello* o un mensaje TC (Cuadro 5.11).

La recepción de un mensaje *Hello* que incorpore un anuncio de servicio, trae consigo el almacenamiento, si procede, del servicio en la caché de servicios del nodo que recibe el mensaje (Cuadro 5.12). Si el servicio recibido ya está almacenado y es un servicio que ofrece el propio nodo, no se hará nada con el mensaje. Pero

```
public void processHello(Message msg){
    service = getServiceType(msg);
    description = getServiceDescription(msg);
}

public void getServiceType(Message msg){
    Tlv valid = msg.getMsgTlvBlock().findTlvByType(SDM_TYPE);
}
```

Cuadro 5.12: Procesar mensajes Hello en SD-OLSRv2

si el servicio almacenado no es un servicio que ofrezca el propio nodo, se compararán los tiempos de vida del servicio almacenado con el del servicio recibido y se guardará en la caché de servicios el de mayor tiempo de vida. Por último, en el caso de que el servicio no se encuentre almacenado, directamente se guarda. El procedimiento es similar al del mecanismo SD-OLSRv1.

5.3. Prestaciones de SD-OLSRv2

Las simulaciones realizadas muestran como varían los resultados de la simulación conforme aumenta el número de nodos que integran la red, la velocidad a la que se mueven los nodos y el número de nodos que existe ofreciendo el mismo servicio.

Las prestaciones de SD-OLSRv2 las vamos a evaluar comparando los resultados con la segunda versión de SD-OLSRv1 y con el mecanismo de descubrimiento de servicios integrado en el protocolo reactivo AODV implementado en [Gupta, 2003].

Lo comparamos con SD-OLSRv1 para ver si las mejoras introducidas sobre SD-OLSRv2 y sobre el propio protocolo de encaminamiento OLSRv2 se reflejan en los escenarios simulados.

Lo comparamos con SD-AODV para ver las diferencias que existen entre un mecanismo de descubrimiento de servicios integrado en un protocolo de encaminamiento proactivo y un mecanismo de descubrimiento de servicios integrado en un protocolo de encaminamiento reactivo. Y ver en qué entornos SD-OLSRv2 se comporta mejor que SD-AODV.

Vamos a partir de un escenario base similar al de la sección 4.3.1: los nodos se mueven siguiendo el modelo de movilidad Random Waypoint en un área de

1000 x 1000 metros con un rango de transmisión de 250 metros. El modelo de propagación radio que utilizamos es el modelo Two-Ray Ground. El tiempo de simulación se ha establecido en 400 segundos. Cada simulación la repetimos 10 veces para dar mayor veracidad a los resultados obtenidos. Estos valores permiten obtener un intervalo de confianza del 95 %.

En el caso de SD-OLSRv1, la frecuencia con la que se envían los mensajes SDM la establecemos en 5 segundos. La frecuencia con la que se envían los mensajes *Hello* y TC del protocolo de encaminamiento OLSRv1 la fijamos en 2 y 5 segundos respectivamente. Valores recomendados en la RFC del protocolo de encaminamiento OLSRv1 [Clausen et al, 2003].

En el caso de SD-OLSRv2, también establecemos la frecuencia con la que se envían mensajes *Hello* y TC del protocolo OLSRv2 en 2 y 5 segundos. En SD-OLSRv2 no definimos ningún tipo de mensaje nuevo para que los nodos envíen los anuncios de servicios. En SD-OLSRv2 los nodos anuncian sus servicios dentro de los mensajes *Hello* y TC. Por lo tanto, la frecuencia con la que los nodos anuncian sus servicios en SD-OLSRv2, es la fijada para el envío de los mensajes *Hello* y TC.

En el caso de SD-AODV, cada nodo genera una petición de descubrimiento de servicios SREQ cada 5 segundos. En el intervalo de simulación cada nodo realizará, por lo tanto, 80 peticiones de servicio. En las simulaciones vamos a considerar dos escenarios diferentes para el protocolo SD-AODV. En el primero, los mensajes de petición de servicio se retransmitirán por *broadcast* a toda la red y en el segundo, los mensajes de petición de servicio se retransmitirán en un rango de dos saltos desde el nodo que realiza la petición.

Por simplicidad sólo habrá 5 nodos que ofrezcan servicios en la red. Cada uno de estos 5 nodos únicamente ofrecerá un tipo de servicio. Siempre que no se indique algún parámetro de simulación, se debe considerar el valor indicado en el escenario base. Los parámetros que vamos a evaluar son los siguientes:

- Sobrecarga introducida en la red MANET debido al mecanismo de descubrimiento de servicios SD-OLSRv2, comparada con la sobrecarga introducida por SD-OLSRv1 y SD-AODV.

- Tiempo en descubrir un servicio en SD-OLSRv2 comparado con el tiempo en descubrir un servicio en SD-OLSRv1 y SD-AODV.

- Tasa de descubrimiento de servicios en la red MANET con el mecanismo de descubrimiento de servicios SD-OLSRv2, comparada con la tasa de descubrimiento de servicios con SD-OLSRv1 y SD-AODV.

Parámetros de simulación	Valor
Versión NS-2	2.26 y 2.34
Implementación OLSRv1	UM-OLSR
Implementación OLSRv2	JOLSRv2
Valores Hello y TC	cada 2 y 5 s.
SDM_INTERVAL SD-OLSRv1	5 s.
SDM_INTERVAL SD-OLSRv2	5 s.
Número de servicios	dos por cada nodo
Duración de la simulación	400 s.
Área simulación	1000x1000 m.
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	10, 20, 30, 50

Cuadro 5.13: Parámetros de simulación sobrecarga SD-OLSRv2 vs SD-OLSRv1

- Tasa de descubrimientos falsos en la red MANET con el mecanismo de descubrimiento de servicios SD-OLSRv2, comparada con la tasa de descubrimientos falsos con SD-OLSRv1 y SD-AODV.

5.3.1. Sobrecarga introducida por SD-OLSRv2

SD-OLSRv2 vs SD-OLSRv1

En las simulaciones realizadas para analizar la sobrecarga introducida en la red por SD-OLSRv2 cuando lo comparamos con la sobrecarga introducida por SD-OLSRv1, hemos considerado escenarios con una media de 10, 20, 30 y 50 nodos. Al comienzo de la simulación los nodos anuncian un servicio, diferente para cada nodo. A lo largo de la simulación cada nodo anuncia un segundo servicio.

En la tabla 5.13 se muestra un resumen con los parámetros de la simulación.

Una de las principales diferencias entre SD-OLSRv2 y SD-OLSRv1 es que en SD-OLSRv2 no definimos ningún tipo de mensaje para que los nodos envíen periódicamente anuncios de servicios con lo que la sobrecarga introducida en la red, entendida como número de paquetes que se transmiten en la misma, será menor.

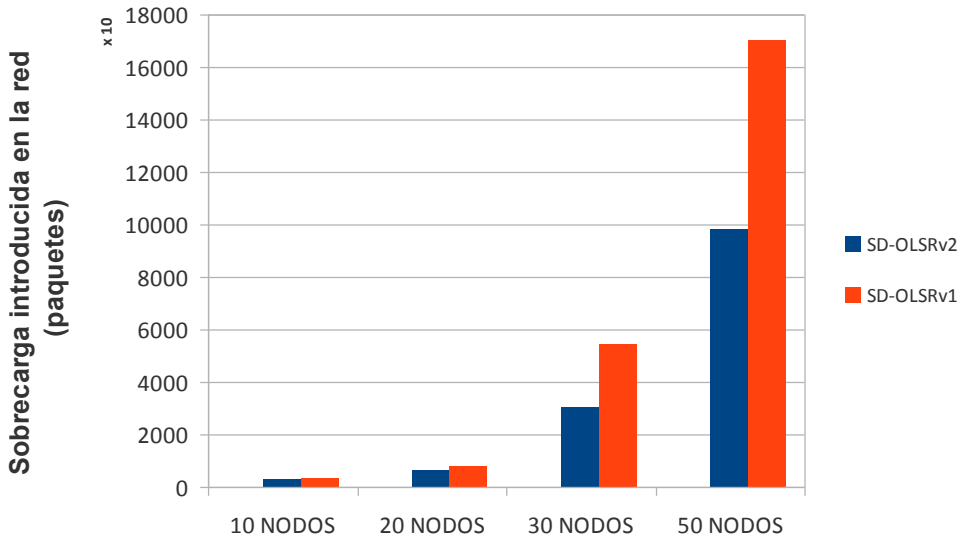

Figura 5.11: Sobrecarga introducida en la red con SD-OLSRv2 y SD-OLSRv1

Efectivamente, tal como vemos en la figura 5.11, independientemente del número de nodos que integren la red, SD-OLSRv2 genera menos sobrecarga que SD-OLSRv1.

Esta diferencia es mayor a medida que la red se hace más grande. En el caso de SD-OLSRv1, cuantos más nodos integren la red más mensajes SDM se envían y más retransmisiones de estos mensajes habrá. En SD-OLSRv2 los nodos anuncian sus servicios como parte de los mensajes *Hello* y TC. No se envía ningún mensaje nuevo para anunciar servicios en SD-OLSRv2.

SD-OLSRv2 vs SD-AODV

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv2 referentes a la sobrecarga introducida en la red, comparada con la sobrecarga introducida en la red por SD-AODV, hemos considerado escenarios con una media de 20, 50 y 100 nodos. Hemos aumentado el número de nodos hasta 100 para ver si existe diferencia en redes muy grandes, entre integrar el mecanismo de descubrimiento de servicios en un protocolo de encaminamiento proactivo e integrarlo en un protocolo de encaminamiento reactivo. Vamos a variar también la velocidad con la que se mueven los nodos en la red desde 2 m/s hasta 10 m/s para simular redes más y menos móviles. Por simplicidad, partimos del escenario base donde

Parámetros de simulación	**Valor**
Versión NS-2	2.34
Implementación OLSRv2	JOLSRv2
Implementación SD-AODV	[Gupta, 2003]
Valores Hello y TC	cada 2 y 5 s.
Peticiones SREQ	cada 5 s.
SDM_INTERVAL SD-OLSRv2	cada 5 s.
Número de servicios	5
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	20, 50, 100

Cuadro 5.14: Parámetros de simulación sobrecarga SD-OLSRv2 vs SD-AODV

sólo cinco nodos anuncian el servicio que ofrecen. En la tabla 5.14 se muestra un resumen con los parámetros de la simulación.

Los resultados que obtenemos de los escenarios simulados se muestran en las figuras 5.12, 5.13 y 5.14.

Tal como observamos en la figura 5.12, en redes pequeñas de 20 nodos SD-OLSRv2 introduce más sobrecarga en la red MANET que SD-AODV. SD-OLSRv2 envía periódicamente mensajes de anuncio de servicios que se retransmiten a toda la red. Independientemente de que existan o no existan cambios en los servicios. A pesar de utilizar los nodos MPR para retransmitir los anuncios de servicios a toda la red, esto genera una sobrecarga mayor a la de un protocolo de encaminamiento reactivo con un mecanismo de descubrimiento de servicios tipo *pull*. SD-AODV tanto cuando las peticiones se envían por *broadcast*, como cuando únicamente se envían a una distancia de dos saltos genera menos sobrecarga que SD-OLSRv2.

A medida que la red aumenta de tamaño y pasa de 20 a 50 y 100 nodos, y los nodos que integran la red se mueven a mayor velocidad la sobrecarga que introduce SD-AODV aumenta, figuras 5.13 y 5.14. Si encima las peticiones de descubrimiento de servicio SREQ de SD-AODV se retransmiten a todos los nodos de la red y no únicamente a los nodos que se encuentran a dos saltos de distancia,

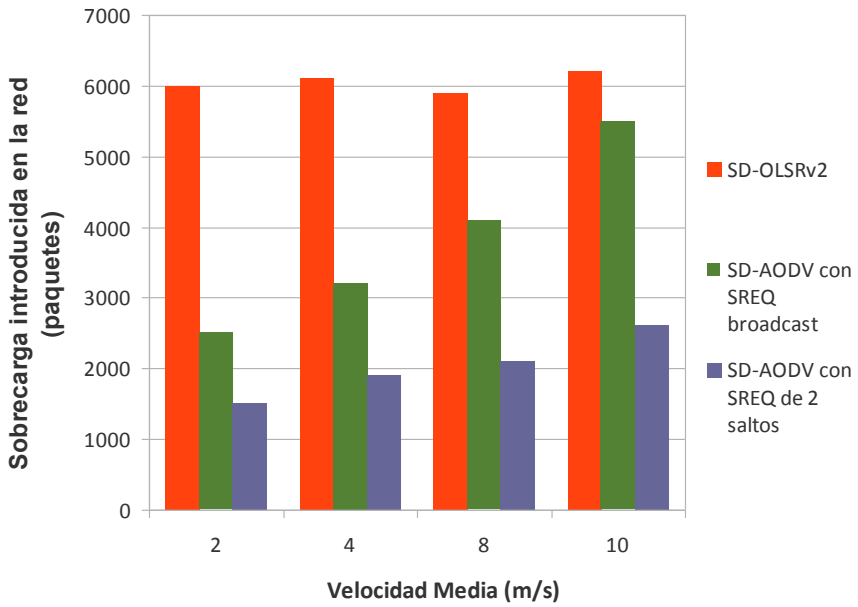

Figura 5.12: Sobrecarga introducida en la red con SD-OLSRv2 comparada con la introducida con SD-AODV en redes con 20 nodos

la sobrecarga en SD-AODV aumenta todavía más. Hasta el punto de que en redes con 100 nodos la sobrecarga introducida por SD-AODV supera a la de SD-OLSRv2.

El protocolo de encaminamiento OLSR tiene un mecanismo para no sobrecargar en exceso la red cuando se retransmiten por *broadcast* mensajes. Los nodos MPR son los únicos encargados de retransmitir estos mensajes. En cambio, en AODV cuando los mensajes se retransmiten a toda la red no existe ningún mecanismo para evitar que todos los nodos de la red retransmitan los mensajes. Por eso, a medida que la red se hace más grande, SD-OLSRv2 se comporta mejor que SD-AODV. Es decir, en redes grandes de 50 y 100 nodos, un mecanismo de descubrimiento de servicios proactivo introduce menos sobrecarga en la red que un mecanismo de descubrimiento de servicios reactivo.

En todas las simulaciones realizadas observamos que cuanto más móvil es la red, SD-AODV introduce más sobrecarga. Con SD-OLSRv2, sin embargo, la sobrecarga se mantiene más o menos estable independientemente de la velocidad a la que se muevan los nodos. Cuando más móvil es la red, existen más enlaces rotos y los nodos necesitan enviar más peticiones de servicio SREQ para encontrar el servicio solicitado. No ocurre lo mismo con SD-OLSRv2 que está enviando

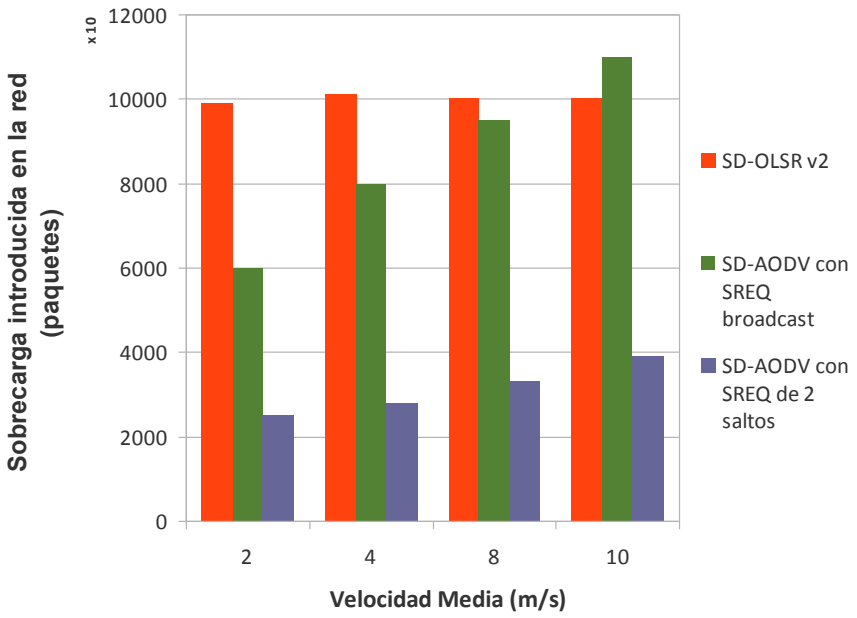

Figura 5.13: Sobrecarga introducida en la red con SD-OLSRv2 comparada con la introducida con SD-AODV en redes con 50 nodos

periódicamente mensajes de anuncio de servicio.

En resumen, SD-OLSRv2 introduce menos sobrecarga en la red MANET que SD-OLSRv1. Esta mejora es debida a que SD-OLSRv2 no define ningún tipo de mensaje nuevo para anunciar servicios. Para anunciar servicios los mensajes *Hello* y TC del protocolo OLSRv2 incorporan un atributo nuevo donde se describe el tipo de servicio que se anuncia. Cuantos más nodos haya anunciando servicios la diferencia en la sobrecarga será mayor.

Si comparamos el mecanismo de descubrimiento de servicios SD-OLSRv2 con el protocolo de descubrimiento de servicios SD-AODV, los resultados que obtenemos están ligados al comportamiento de los protocolos de encaminamiento OLSRv2 y AODV y la diferencia que existe entre protocolos proactivos y protocolos reactivos. En redes pequeñas con poca movilidad SD-AODV se comporta mejor que SD-OLSRv2, pero en redes grandes, por encima de 50 nodos, donde los nodos se mueven rápido, SD-OLSRv2 proporciona mejores resultados.

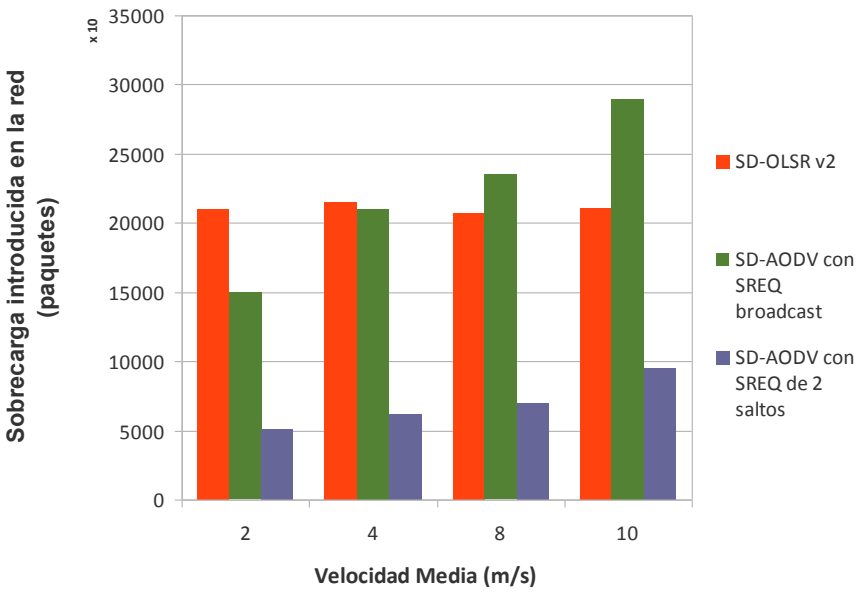

Figura 5.14: Sobrecarga introducida en la red con SD-OLSRv2 comparada con la introducida con SD-AODV en redes con 100 nodos

5.3.2. Tiempo en descubrir un servicio con SD-OLSRv2

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv2 referentes al tiempo que emplea un nodo en descubrir un servicio en la red, hemos considerado escenarios con una media de 50 nodos. Por simplicidad, partimos del escenario base donde sólo cinco nodos anuncian el servicio que ofrecen.

Vamos a simular dos tipos de escenarios en función de los siguientes parámetros:

- la velocidad a la que se mueven los nodos.
- el número de nodos que existe ofreciendo el mismo servicio.

En la tabla 5.15 se muestra un resumen con los parámetros de la simulación.

Vamos a comparar el tiempo empleado por un nodo en descubrir un servicio en SD-OLSRv2 con el tiempo empleado por un nodo en SD-OLSRv1 y SD-AODV. Y veremos si la velocidad y el número de nodos ofreciendo el mismo servicio influyen en el tiempo que un nodo tarda en descubrir un servicio.

Parámetros de simulación	**Valor**
Versión NS-2	2.34 y 2.26
Implementación OLSRv2	JOLSRv2
Implementación OLSRv1	UM-OLSR
Implementación SD-AODV	[Gupta, 2003]
Valores Hello y TC	cada 2 y 5 s.
Peticiones SREQ SD-AODV	cada 5 s.
SDM_INTERVAL SD-OLSRv2	cada 5 s.
SDM_INTERVAL SD-OLSRv1	cada 5 s.
Número de servicios	5
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	50

Cuadro 5.15: Parámetros de simulación Tiempo en descubrir un servicio SD-OLSRv2 vs SD-OLSRv1 y SD-AODV

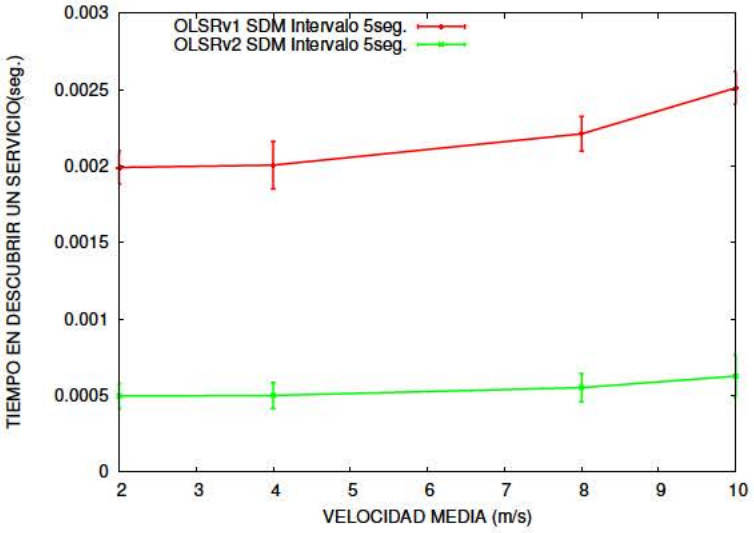

Figura 5.15: Tiempo en descubrir un servicio SD-OLSRv2 vs SD-OLSRv1 en función de la velocidad a la que se mueven los nodos en la red

SD-OLSRv2 vs SD-OLSRv1

En la figura 5.15 se muestran los resultados del tiempo empleado por un nodo en redes MANET con SD-OLSRv1 y con SD-OLSRv2 cuando los nodos que integran la red se mueven a diferentes velocidades. Independientemente de la velocidad a la que se muevan los nodos, observamos que los nodos con SD-OLSRv2 tardan menos tiempo en descubrir un servicio que los nodos con SD-OLSRv1. Estos resultados eran los esperados si tenemos en cuenta que el retardo extremo a extremo en el protocolo de encaminamiento OLSRv2 es menor al de OLSRv1 [Dahiya et al, 2013].

También observamos que en ambos casos, cuanto más móvil es la red más se tarda en descubrir los servicios, aunque esta diferencia es insignificante. La velocidad no es un parámetro clave en la red MANET, con protocolo de encaminamiento OLSR, a la hora de descubrir servicios.

En la figura 5.16 se muestran los resultados del tiempo empleado por un nodo en redes MANET con SD-OLSRv1 y con SD-OLSRv2 en función del número de nodos que hay ofreciendo el mismo servicio. Según se observa, cuantos más nodos hay ofreciendo el mismo servicio más rápido es para un nodo encontrar un servicio. No hay diferencias apreciables entre SD-OLSRv1 y SD-OLSRv2. El mecanismo de descubrimiento de servicios como tal no difiere mucho entre ambos protocolos.

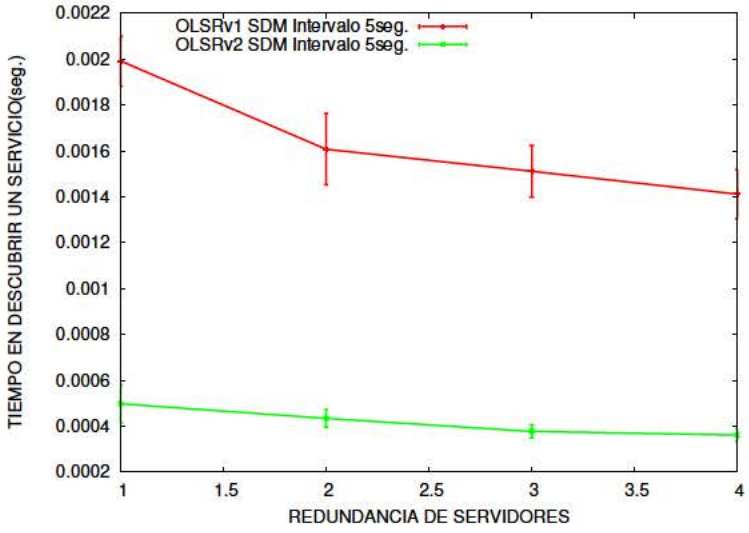

Figura 5.16: Tiempo en descubrir un servicio SD-OLSRv2 vs SD-OLSRv1 en función del número de servidores ofreciendo el mismo servicio en la red

SD-OLSRv2 vs SD-AODV

En la figura 5.17 se muestran los resultados del tiempo empleado por un nodo en descubrir un servicio en redes MANET con SD-OLSRv2 y con SD-AODV cuando los nodos que integran la red se mueven a diferentes velocidades.

Mientras que en SD-OLSRv2 los nodos descubren los servicios casi inmediatamente y la velocidad a la que se mueven los nodos no afecta al tiempo que emplea un nodo en descubrir un servicio, en SD-AODV, el tiempo que un nodo tarda en descubrir un servicio sí que aumenta a medida que los nodos se mueven a una velocidad mayor. Los nodos en SD-OLSRv2 están continuamente anunciando servicios. Los nodos que integran la red guardan en una caché de servicios los servicios anunciados de tal forma que cuando un nodo necesita un servicio sólo necesita buscar en su caché de servicios. Los nodos en SD-AODV piden el servicio cuando lo necesitan. Estas peticiones desencadenan un proceso de búsqueda del servicio entre todos los nodos de la red y su consecuente respuesta que implica más tiempo de búsqueda en la red SD-OLSRv2. Además, en SD-AODV cuanto más móvil sea la red más se tardará en encontrar el servicio, puesto que más enlaces rotos habrá.

En la figura 5.18 se muestran los resultados del tiempo empleado por un nodo en redes MANET con SD-OLSRv2 y con SD-AODV en función del número de nodos que hay ofreciendo el mismo servicio. Tanto para SD-OLSRv2 como

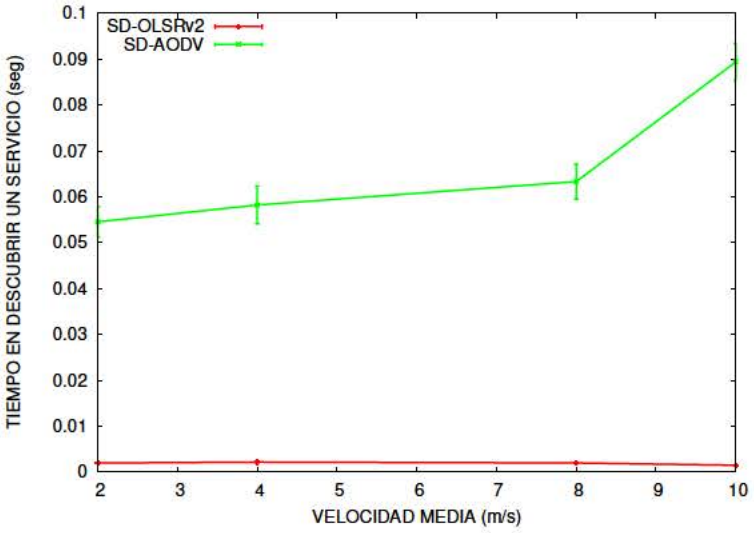

Figura 5.17: Tiempo en descubrir un servicio SD-OLSRv2 vs SD-AODV en función de la velocidad a la que se mueven los nodos en la red

para SD-AODV a medida que aumenta el número de nodos ofreciendo el mismo servicio el tiempo en descubrirlo se reduce. La diferencia es más visual en el caso de SD-AODV puesto que el tiempo en descubrir el servicio es inicialmente mayor.

En resumen, podemos decir que el tiempo en descubrir un servicio en una red MANET proactiva es menor que el tiempo en descubrir un servicio en una red MANET reactiva. En una red MANET proactiva los servicios se están anunciando constantemente y a un nodo le basta con acceder a su caché de servicios para saber quién ofrece el servicio que quiere. En una red reactiva la búsqueda de un servicio implica el envío de un mensaje de petición de servicio y la espera de su correspondiente respuesta.

Dentro de la red MANET con el protocolo de encaminamiento OLSR, el mecanismo de descubrimiento de servicios definido para la versión 2 de OLSR proporciona tiempos ligeramente menores en descubrir los servicios a los del mecanismo definido para la versión 1 de OLSR. La razón reside en las mejoras introducidas en el protocolo OLSRv2 tal como se recoge en [Dahiya et al, 2013].

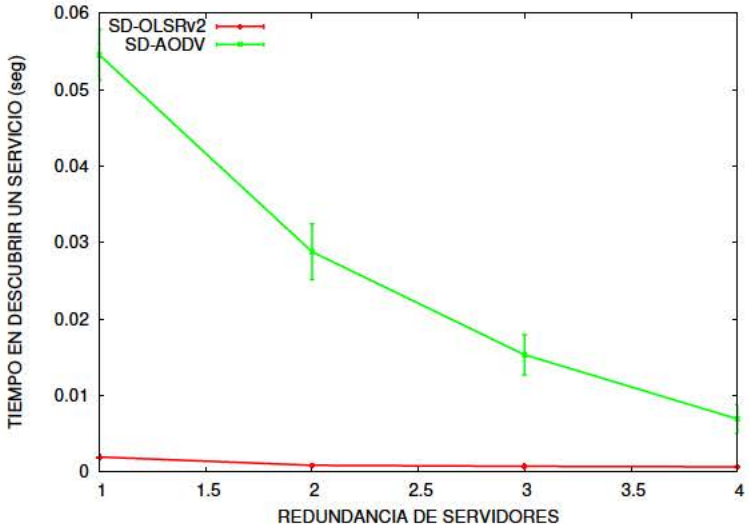

Figura 5.18: Tiempo en descubrir un servicio SD-OLSRv2 vs SD-AODV en función de la velocidad a la que se mueven los nodos en la red

5.3.3. Tasa de descubrimiento de servicios con SD-OLSRv2

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv2 referentes a la tasa de descubrimiento de servicios, hemos considerado escenarios con una media de 50 nodos. Por simplicidad, partimos del escenario base donde sólo cinco nodos anuncian el servicio que ofrecen.

Vamos a medir la tasa de descubrimiento de servicios en SD-OLSRv2 comparada con la de SD-OLSRv1 y SD-AODV en función de los siguientes parámetros:

- la velocidad a la que se mueven los nodos.
- el número de servidores que existe ofreciendo el mismo servicio.

En la tabla 5.16 se muestra un resumen con los parámetros de la simulación.

Vamos a comparar la tasa de descubrimiento de servicios en SD-OLSRv2 con la tasa de descubrimiento de servicios en SD-OLSRv1 y SD-AODV. Y veremos si la velocidad y el número de nodos ofreciendo el mismo servicio influyen en el porcentaje de servicios que se descubren en la red.

Parámetros de simulación	**Valor**
Versión NS-2	2.34 y 2.26
Implementación OLSRv2	JOLSRv2
Implementación OLSRv1	UM-OLSR
Implementación SD-AODV	[Gupta, 2003]
Valores Hello y TC	cada 2 y 5 s.
Peticiones SREQ	cada 5 s.
SDM_INTERVAL SD-OLSRv2	cada 5 s.
SDM_INTERVAL SD-OLSRv1	cada 5 s.
Número de servicios	5
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	50

Cuadro 5.16: Parámetros de simulación Tasa de descubrimiento de servicios SD-OLSRv2 vs SD-OLSRv1 y SD-AODV

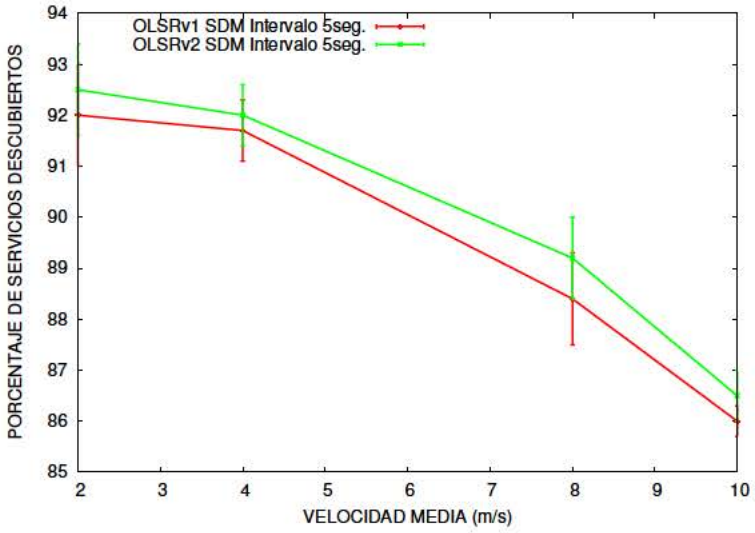

Figura 5.19: Tasa de descubrimiento de servicios SD-OLSRv2 vs SD-OLSRv1 en función de la velocidad a la que se mueven los nodos en la red

SD-OLSRv2 vs SD-OLSRv1

En las figuras 5.19 y 5.20 se muestran los resultados cuando los nodos que integran la red se mueven a diferentes velocidades y cuando existe más de un nodo ofreciendo el mismo servicio respectivamente.

Tal como podemos observar la tasa de descubrimiento de servicios es ligeramente superior en SD-OLSRv2. Las bases del mecanismo de descubrimiento de servicios son similares para OLSRv1 y OLSRv2. El ligero aumento en la tasa de descubrimiento de servicios se debe a las mejoras en el protocolo de encaminamiento OLSRv2 respecto a OLSRv1 [Dahiya et al, 2013].

Del mismo modo, al igual que ocurría en el caso del tiempo que tarda un nodo en descubrir un servicio, cuanto más móvil es la red, menos servicios se descubren. Con la frecuencia de anuncio de servicios definida, cuando los nodos se mueven a velocidades de 10 m/s la tasa de descubrimiento de servicios baja hasta el 87 %. Mientras que cuando los nodos se mueven a velocidades bajas la tasa de descubrimiento de servicios es cercana al 95 %.

A medida que aumenta el número de nodos que ofrecen el mismo servicio, la tasa de descubrimiento de servicios aumenta y para el caso de OLSRv2 llega a ser cercana al 100 %.

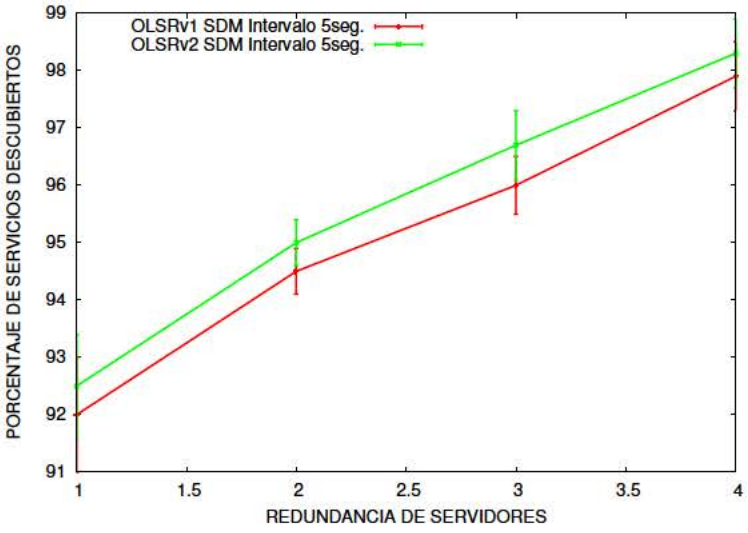

Figura 5.20: Tasa de descubrimiento de servicios SD-OLSRv2 vs SD-OLSRv1 en función del número de nodos ofreciendo el mismo servicio en la red en la red

SD-OLSRv2 vs SD-AODV

En la figura 5.21 se muestran los resultados cuando los nodos que integran la red se mueven a diferentes velocidades. En la figura 5.22 se muestran los resultados en función del número de nodos que hay ofreciendo el mismo servicio.

Tal como podemos observar, la tasa de descubrimiento de servicios en SD-OLSRv2, cuando los nodos se mueven a velocidades bajas, es similar a la de SD-AODV cuando la petición de servicios SREQ se envía por *broadcast* a toda la red. Sin embargo, a medida que la velocidad a la que se mueven los nodos aumenta, SD-AODV tiene una tasa de descubrimiento de servicios peor que SD-OLSRv2. Esto es debido a que a velocidades altas, existen más probabilidades de pérdidas de paquetes.

Cuando las peticiones de servicio en SD-AODV únicamente se envían en un rango de dos saltos desde el nodo origen, la tasa de descubrimiento de servicios cae a menos del 40 %.

Del mismo modo, a medida que aumenta el número de nodos que ofrecen el mismo servicio, la tasa de descubrimiento de servicios aumenta tanto para SD-OLSRv2 como para SD-AODV con un envío de peticiones de servicio *broadcast*. La tasa de descubrimiento de servicios es ligeramente superior en SD-OLSRv2 aunque la diferencia es prácticamente inapreciable. Cuando las peticiones de ser-

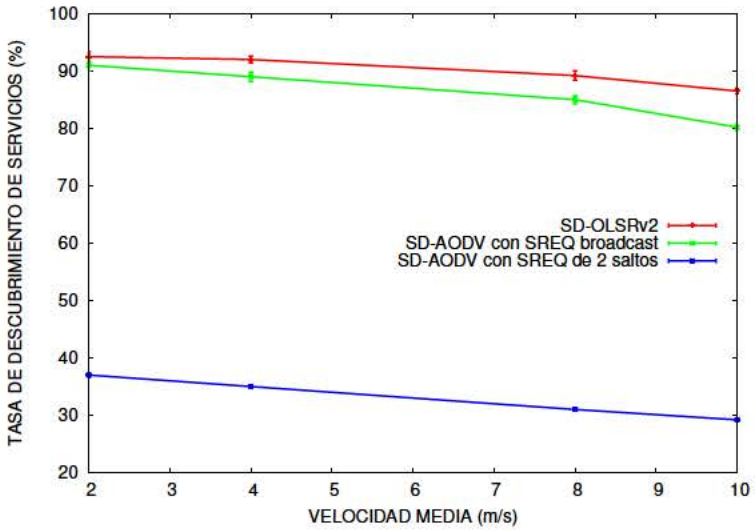

Figura 5.21: Tasa de descubrimiento de servicios SD-OLSRv2 vs SD-AODV en función de la velocidad a la que se mueven los nodos en la red

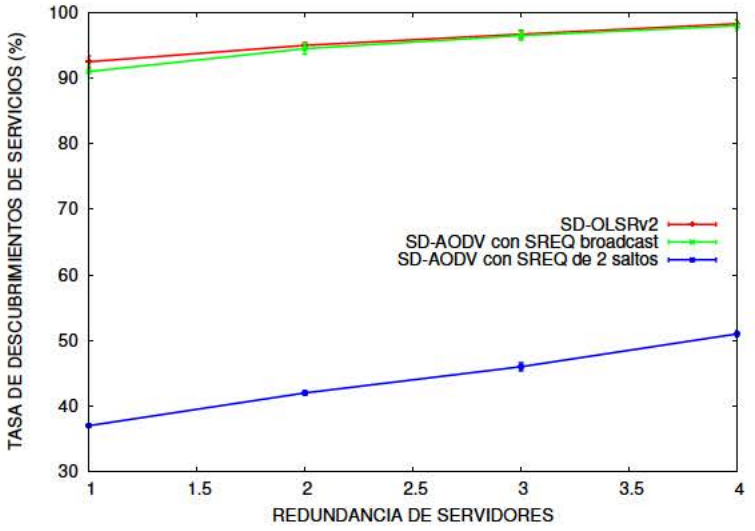

Figura 5.22: Tasa de descubrimiento de servicios SD-OLSRv2 vs SD-AODV en función del número de nodos ofreciendo el mismo servicio en la red

vicio en SD-AODV se reducen a dos saltos desde el nodo que realiza la petición, la tasa de descubrimiento de servicios, aunque aumenta, sigue siendo inferior al 50 %.

En definitiva, SD-OLSRv2 ofrece una tasa de descubrimiento de servicios alta, superior a la que obteníamos con SD-OLSRv1. A medida que aumenta la movilidad de la red, la tasa de descubrimiento de servicios disminuye pero sigue siendo alta, en torno al 90 %. En cualquiera de los casos, la tasa de descubrimiento de servicios es superior a la que se consigue con SD-AODV.

5.3.4. Tasa de descubrimientos falsos con SD-OLSRv2

En las simulaciones realizadas para analizar las prestaciones de SD-OLSRv2 referentes a la tasa de descubrimientos falsos, hemos considerado escenarios con una media de 50 nodos. Por simplicidad, partimos del escenario base donde sólo cinco nodos anuncian el servicio que ofrecen.

Vamos a medir la tasa de descubrimientos falsos en SD-OLSRv2 comparada con la de SD-OLSRv1 y SD-AODV en función de los siguientes parámetros:

- la velocidad a la que se mueven los nodos.
- el número de nodos que existe ofreciendo el mismo servicio.

En la tabla 5.17 se muestra un resumen con los parámetros de la simulación.

Vamos a comparar la tasa de descubrimientos falsos en SD-OLSRv2 con la tasa de descubrimientos falsos en SD-OLSRv1 y SD-AODV. Y veremos si la velocidad y el número de nodos ofreciendo el mismo servicio influyen en el porcentaje de falsos servicios descubiertos.

SD-OLSRv2 vs SD-OLSRv1

En la figura 5.23 se muestran los resultados cuando los nodos que integran la red se mueven a diferentes velocidades. En la figura 5.24 se muestran los resultados en función del número de nodos que hay ofreciendo el mismo servicio.

Una vez más, SD-OLSRv2 ofrece mejores resultados que SD-OLSRv1. En ninguno de los casos, SD-OLSRv2 ofrece una tasa de falsos descubrimientos superior al 0.5 %. La razón de esta mejoría radica en que en SD-OLSRv2 una vez que comprueban en su caché de servicios remotos si algún nodo de la red ofrece el servicio que necesitan, consultan la información que cada nodo almacena en la

Parámetros de simulación	Valor
Versión NS-2	2.34 y 2.26
Implementación OLSRv2	JOLSRv2
Implementación OLSRv1	UM-OLSR
Implementación SD-AODV	[Gupta, 2003]
Valores Hello y TC	cada 2 y 5 s.
Peticiones SREQ	cada 5 s.
SDM_INTERVAL SD-OLSRv2	cada 5 s.
SDM_INTERVAL SD-OLSRv1	cada 5 s.
Número de servicios	5
Duración de la simulación	400 s.
Área simulación	1000x1000
Rango de transmisión	250 m.
Modelo de movilidad	Random Waypoint
Modelo de propagación radio	Two-Ray Ground
Protocolo MAC	IEEE 802.11 DCF
Intervalo de confianza	95 %
Número de nodos	50

Cuadro 5.17: Parámetros de simulación Tasa de descubrimientos falsos SD-OLSRv2 vs SD-OLSRv1 y SD-AODV

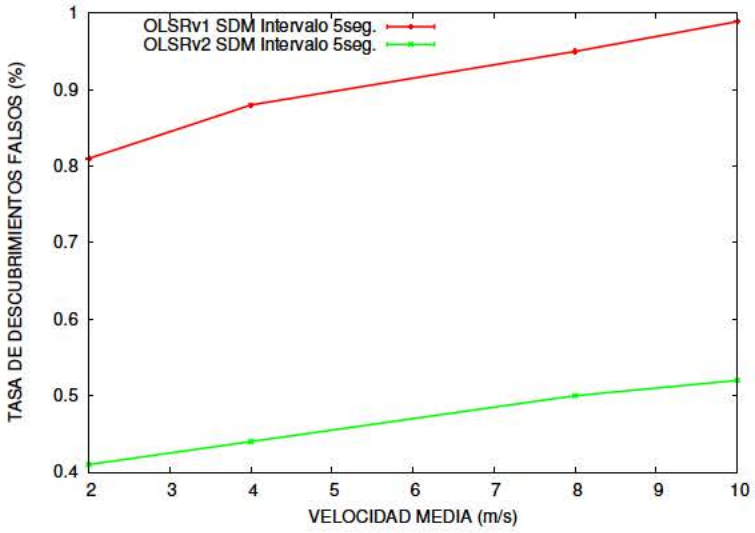

Figura 5.23: Tasa de descubrimientos falsos SD-OLSRv2 vs SD-OLSRv1 en función de la velocidad a la que se mueven los nodos en la red

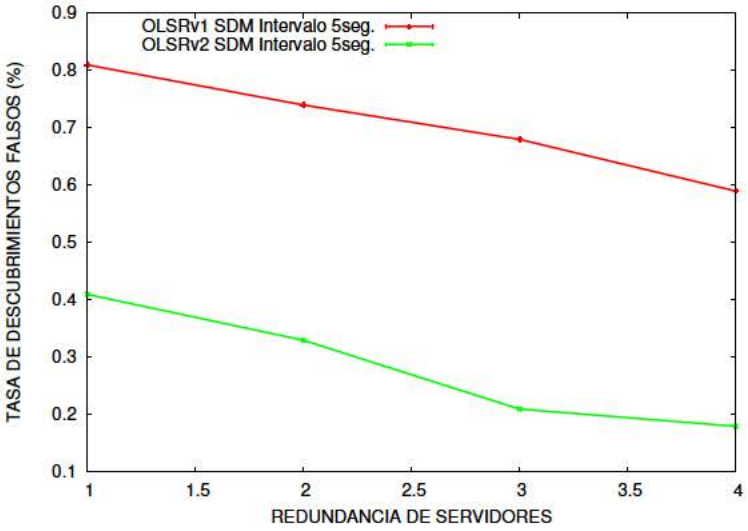

Figura 5.24: Tasa de descubrimientos falsos SD-OLSRv2 vs SD-OLSRv1 en función del número de nodos ofreciendo el mismo servicio en la red

Topology Information Base para ver si el nodo que ofrece el servicio sigue estando accesible en la red.

SD-OLSRv2 vs SD-AODV

En el caso de SD-AODV, la tasa de falsos descubrimientos es similar a la de SD-OLSRv2, independientemente de que la petición de servicio se envíe por *broacast* a toda la red o sólo a los nodos que se encuentran a dos saltos de distancia del nodo origen. En el caso de SD-AODV, cuando un nodo solicita un servicio a través de la petición SREQ, y obtiene una respuesta no hay posibilidad de que el servicio sea erróneo. Sin embargo, cuando un nodo solicita un servicio repetidamente, como es el caso del escenario que estamos simulando, antes de enviar la solicitud de petición de servicio comprueba si ya sabe quién es el nodo que ofrece el servicio que solicita, y en caso afirmativo, ya no envía la petición. En estos casos, sí que se puede dar el caso de falsos descubrimientos. El nodo puede tener almacenado un servicio que ya no esté disponible en la red.

5.4. Conclusiones

En este capítulo hemos descrito de forma razonada el mecanismo de descubrimiento de servicios sobre la versión 2 del protocolo de encaminamiento OLSR y hemos analizado sus prestaciones. Hemos comparado el mecanismo de descubrimiento de servicios integrado en OLSRv2 con el definido para la primera versión del protocolo de encaminamiento OLSR. Además hemos comparado el mecanismo de descubrimiento de servicios integrado sobre el protocolo de encaminamiento proactivo OLSRv2, con el mecanismo de descubrimiento de servicios sobre el protocolo de encaminamiento reactivo AODV.

SD-OLSRv2 mantiene los mismo principios de funcionamiento que SD-OLSRv1 e incluye algunas mejoras. No define ningún tipo de mensaje nuevo para anunciar periódicamente los servicios que cada nodo ofrece a la red MANET. Los anuncios de servicios se definen como una estructura TLV dentro de los mensajes *Hello* y TC del protocolo de encaminamiento OLSRv2. De esta forma la sobrecarga que se introduce en la red MANET en cuanto a paquetes que se transmiten en la misma se reducen considerablemente respecto al mecanismo de descubrimiento de servicios definido para OLSRv1.

Además, con SD-OLSRv2 cuando un nodo comprueba si tienen almacenado en su caché de servicios el servicio del que quiere hacer uso, comprueba también si el nodo que ofrece el servicio sigue estando disponible en la red MANET.

Después de realizar las simulaciones y analizado los resultados, concluimos que SD-OLSRv2 ofrece mejores prestaciones que SD-OLSRv1 para los valores de sobrecarga introducida en la red, tiempo en descubrir un servicio, tasa de descubrimiento de servicios y tasa de descubrimientos falsos analizados. Una de las razones de estos resultados es la mejora en el protocolo de encaminamiento OLSRv2 respecto a OLSRv1: el *throughput* en OLSRv2 es significativamente mayor que en OLSRv1 y el *average end-to-end-delay* es menor en OLSRv2. Además, como hemos dicho, en el caso de SD-OLSRv2, los nodos anuncian sus servicios como un atributo de los mensajes *Hello* y TC del protocolo OLSRv2. No se crea ningún mensaje nuevo para enviar anuncios de servicios como ocurría en SD-OLSRv1. De esta forma, la sobrecarga en cuanto a paquetes transmitidos en la red MANET se reduce considerablemente. Este hecho se hace más evidente a medida que la red es más grande y son más los nodos que ofrecen servicios. Cuantos más nodos integren la red más retransmisiones habrá de los mensajes SDM y cuantos más servicios se anuncien, más mensajes SDM se transmitirán a la red. También la tasa de falsos descubrimientos es menor en SD-OLSRv2 debido a la mejora introducida en el mecanismo de descubrimiento de servicios respecto a la versión definida sobre OLSRv1.

Por otro lado, cuando comparamos SD-OLSRv2 con SD-AODV vemos que cuanto más grande y más móvil es la red, mejor se comporta SD-OLSRv2 respecto a SD-AODV. SD-OLSRv2 es un mecanismo de descubrimiento de servicios integrado en un protocolo de encaminamiento proactivo y como tal, sus prestaciones se ven incrementadas en redes grandes con una alta movilidad. El tiempo en descubrir un servicio es prácticamente inmediato en el caso de SD-OLSRv2 e inferior al de SD-AODV.

Concluimos por lo tanto que, SD-OLSRv2 es un buen mecanismo de descubrimiento de servicios en redes MANET con un número de nodos elevado, y en redes donde sea necesario un tiempo mínimo en descubrir un servicio y un alto porcentaje de éxito en descubrir servicios. Todo ello, sin aumentar la sobrecarga introducida en la red MANET, al no necesitar SD-OLSRv2 enviar ningún mensaje nuevo para que los nodos anuncien sus servicios.

Capítulo 6

Conclusiones y trabajos futuros

El auge que en los últimos años ha tenido el uso de dispositivos móviles, su integración plena en la vida de las personas, así como el desarrollo expansivo de las redes MANET, hace que sea difícil imaginar un mundo sin dispositivos inteligentes que se conecten a redes móviles y ofrezcan a los usuarios aplicaciones, servicios y experiencias. Que las redes MANET sean capaces de soportar descubrimiento de servicios se antoja, por lo tanto, indispensable. Debido a la movilidad de estas redes, en cualquier momento puede cambiar la topología y los servicios y recursos que se ofrecen en la red. Los dispositivos deben poder descubrir de forma automática los servicios que están disponibles en la red, así como los nodos que proporcionan estos servicios.

En esta tesis doctoral se ha contribuido en la definición de un nuevo mecanismo de descubrimiento de servicios en redes MANET. Inicialmente nuestra investigación se centró en definir un mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSR para redes MANET. Mientras mejorábamos la primera versión del mecanismo propuesto, surgieron los primeros *drafts* de la versión 2 del protocolo de encaminamiento OLSR y nuestra investigación se centró en adaptar el descubrimiento de servicios a la versión 2 de OLSR mejorando sus prestaciones en redes MANET.

En este capítulo recopilamos las principales contribuciones de esta tesis, sección 6.1 y exponemos una visión de las posibles direcciones futuras de investigación en la sección 6.2.

6.1. Resumen y principales contribuciones

Las principales contribuciones que se han realizado en esta tesis doctoral son las siguientes:

1. **Estudio de los mecanismos de descubrimiento de servicios más relevantes que se han proporcionado estos últimos años.**

   El problema del descubrimiento de servicios no es algo nuevo, a lo largo de estos últimos años han aparecido diversas propuestas al respecto. En el capítulo 2, describimos estos protocolos de descubrimiento de servicios partiendo de una clasificación que divide los protocolos de descubrimiento de servicios en protocolos basados en directorio y protocolos no basados en directorio. Entre los segundos están los protocolos de descubrimiento de servicios integrados en un protocolo de encaminamiento.

   En el capítulo 2, también realizamos un estudio teórico y detallado del problema de descubrimiento de servicios, planteamos de forma genérica el problema y enumeramos las diferentes soluciones teóricas existentes, denominadas: modo *pull* y modo *push*. Los mecanismos de descubrimiento existentes basan su funcionamiento en alguna o en varias de estas soluciones teóricas.

2. **Estudio de la versión 1 y 2 del protocolo de encaminamiento OLSR.**

   El estudio de los protocolos de descubrimiento de servicios existentes nos abre una nueva línea de investigación que son los protocolos de encaminamiento en redes MANET. Algunos de los protocolos de descubrimiento de servicios se integran en la capa de red y forman parte del protocolo de encaminamiento, en lugar de estar definidos en la capa de aplicación. Técnica que se conoce como soluciones *cross-layer*.

   En el capítulo 3, clasificamos los protocolos de encaminamiento en protocolos uniformes y protocolos no uniformes, dependiendo de las funciones que realizan los nodos respecto al encaminamiento. Dentro de estas dos categorías los protocolos se pueden clasificar nuevamente en protocolos proactivos, protocolos reactivos y protocolos híbridos.

   En el capítulo 3 también describimos en detalle la versión 1 y la versión 2 del protocolo de encaminamiento OLSR, que es el protocolo sobre el que vamos a diseñar el nuevo mecanismo de descubrimiento de servicios.

3. **Propuesta de un nuevo mecanismo de descubrimiento de servicios *cross-layer* basado en OLSR para redes MANET.**

Partiendo de un modo *push* puro hemos definido un nuevo mecanismo de descubrimiento de servicios integrado en el protocolo de encaminamiento OLSR, que hemos denominado *Service Discovery over OLSR* (SD-OLSR). En los capítulos 4 y 5, se ha realizado una descripción formal del mismo para las versiones 1 y 2 del protocolo de encaminamiento OLSR. Para facilitar su implementación en dispositivos reales y garantizar la interoperabilidad entre diferentes implementaciones, hemos realizado una descripción detallada de su implementación para ambas versiones, que se incluye en las secciones 4.2 y 5.2.

4. **Estudio de prestaciones de SD-OLSRv1 y SD-OLSRv2.**

   En esta tesis se ha realizado un estudio de prestaciones de SD-OLSRv1 y SD-OLSRv2 respecto a una serie de parámetros:

   - Número de nodos que integran la red.

     Simulamos escenarios más y menos grandes para ver las prestaciones del mecanismo propuesto. De los resultados obtenidos podemos concluir que ambos mecanismos se comportan bien en redes con muchos nodos. Si bien SD-OLSRv2 tiene un comportamiento mejor, sobre todo en cuanto a sobrecarga de paquetes introducida en la red se refiere. Cuantos más nodos haya en la red más retransmisiones de mensajes SDM habrá en SD-OLSRv1, y cuantos más nodos haya ofreciendo servicios más mensajes SDM se transmitirán en la red.

   - Frecuencia con la que se envían los anuncios de servicios.

     En las simulaciones realizadas en distintos tipos de escenarios, modificamos la frecuencia con la que los nodos de la red MANET anuncian sus servicios. De los análisis realizados mediante simulación, observamos que cuanto más frecuentemente se anuncien los servicios, mayor tasa de servicios descubiertos tendremos pero también generaremos mayor sobrecarga de paquetes transmitidos a la red.

   - Velocidad a la que se mueven los nodos.

     En las simulaciones realizadas en distintos tipos de escenarios, modificamos la velocidad a la que se mueven los nodos para simular entornos más y menos móviles. De los análisis realizados mediante simulación, observamos que la velocidad a la que se mueven los nodos en la red no es un parámetro crítico a la hora de descubrir servicios con el mecanismo SD-OLSR.

   - Número de nodos ofreciendo el mismo servicio.

     En las simulaciones realizadas en distintos tipos de escenarios, modificamos el número de nodos ofreciendo el mismo servicio. De los

análisis realizados mediante simulación, observamos que a medida que el número de nodos ofreciendo el mismo servicio crece la tasa de descubrimiento de servicios aumenta.

Además de este análisis, también hemos realizado un estudio de prestaciones de SD-OLSRv2 comparándolo con el protocolo SD-AODV, que integra el descubrimiento de servicios en el protocolo de encaminamiento reactivo AODV. En este caso los parámetros que se varían son:

- Número de nodos que integran la red.

  Simulamos escenarios más y menos grandes para ver las prestaciones de SD-OLSRv2 respecto a SD-AODV. De los resultados obtenidos podemos concluir, que SD-OLSRv2 se comporta mejor que SD-AODV en redes MANET con un número de nodos grande. SD-OLSR se integra en un protocolo de encaminamiento proactivo que está pensado para aumentar su rendimiento en redes grandes gracias a la técnica MPR, que inunda la red de mensajes reduciendo la sobrecarga de paquetes en la red MANET.

- Velocidad a la que se mueven los nodos.

  En las simulaciones realizadas en distintos tipos de escenarios, modificamos la velocidad a la que se mueven los nodos para simular entornos más y menos móviles. De los análisis realizados mediante simulación, observamos que la velocidad a la que se mueven los nodos en la red no es un parámetro crítico a la hora de descubrir servicios con el mecanismo SD-OLSR. Sí lo es en cambio, para SD-AODV que cuanto más móvil es la red los resultados en cuanto a tasa de descubrimiento de servicios, tiempo en descubrir un servicio y tasa de descubrimientos falsos empeora.

- Número de nodos ofreciendo el mismo servicio.

  En las simulaciones realizadas en distintos tipos de escenarios, modificamos el número de nodos ofreciendo el mismo servicio. De los análisis realizados mediante simulación, observamos que tanto para SD-OLSRv2 como para SD-AODV a medida que aumenta el número de nodos ofreciendo el mismo servicio la tasa de descubrimiento de servicios aumenta.

6.2. Vías de investigación futuras

En esta tesis se ha definido un nuevo mecanismo de descubrimiento de servicios basado en el protocolo de encaminamiento OLSR para redes MANET. En esta misma temática se abren las siguientes líneas de investigación:

1. **Estudio y definición de un nuevo formato de descripción de servicios.**

   El mecanismo de descubrimiento de servicios SD-OLSR no define un formato de descripción de servicios específico. Como línea futura queremos implementar el mecanismo de descubrimiento en entornos reales y diseñar un nuevo formato para describir los servicios de forma más precisa con algoritmos de *matchmarking* [Scioscia et al, 2014] acordes a los actuales servicios que se ofrecen en la *Internet of Things* (IoT) [Atzori et al, 2010]. En la Internet de las cosas, las *cosas* y sus recursos (sensores y actuadores) deben ser capaces de describir sus características de una manera clara y completa con el fin de permitir que otros objetos comunicadores puedan interactuar con ellos. Se tienen que definir formatos de descripción de servicios y lenguajes adecuados, seguramente a nivel semántico.

2. **Estudio y definición de mecanismos para determinar la periodicidad con la que un nodo anuncia sus servicios.**

   Como hemos visto en el estudio de prestaciones del mecanismo de descubrimiento de servicios SD-OLSR, la periodicidad con la que un nodo anuncia sus servicios influye en la tasa de descubrimiento de servicios y también en la sobrecarga que se introduce en la red.

   El parámetro que fija la periodicidad con la que se envían los servicios, SDM_INTERVAL, se preconfigura en el mecanismo de descubrimiento de servicios y es común para todos los anuncios que envíen los nodos. Como línea futura, se podrían estudiar mecanismos que permitiesen adaptar la periodicidad con la que se envían los servicios. Por ejemplo, si se detecta que un 90 % de los nodos que integran la red ya disponen del servicio anunciado, disminuir la periodicidad con la que se envía el servicio.

3. **Estudio para añadir extensiones de selección del servicio en base a la QoS**

   Como hemos visto en la descripción del mecanismo de descubrimiento de servicios SD-OLSR (ver sección 4.1.3), actualmente en la caché de servicios de cada nodo sólo se guarda un servicio de cada tipo. En caso de que existan varios nodos anunciando el mismo servicio, el nodo que recibe los anuncios

de servicio sólo guarda en su caché el servicio del nodo cuyo *Lifetime* sea mayor.

Como línea futura queremos cambiar este mecanismo para que ningún nodo que ofrezca servicios se vea perjudicado porque su servicio lo ofrezca otro nodo cuyo *Lifetime* sea mayor. Además, queremos estudiar que la selección del servicio se realice en función de la QoS que ofrece el nodo que anuncia el servicio.

4. **Estudio de selección del servicio en función de aspectos de seguridad.**

   En la misma línea que la propuesta anterior, y dada la importancia que ha adquirido la seguridad en las redes MANET, queremos estudiar las diferentes técnicas que existen para seleccionar el servicio, del nodo que cumpla unos requisitos de seguridad mayores.

Apéndice A

Metodología de simulación

En esta tesis doctoral hemos estudiado mediante simulación las prestaciones del mecanismo de descubrimiento de servicios definido sobre las dos versiones del protocolo de encaminamiento OLSR estandarizadas.

Durante la simulación, los nodos se mueven siguiendo el modelo Random Waypoint. El número de nodos que integran la red varía entre 10 y 100 dependiendo del escenario simulado. Todas las simulaciones se realizan con la misma duración de 400 segundos y sobre un área de 1000x1000 metros.

A.1. Metodología de simulación en la tesis

A.1.1. Definición de objetivos

A continuación se muestran los parámetros que queremos medir mediante simulación, para comprobar la viabilidad del mecanismo de descubrimiento de servicios propuesto:

- **Sobrecarga introducida en la red**. Es el número de mensajes transmitidos en la red. Este parámetro nos permite conocer la sobrecarga que introduce el mecanismo de descubrimiento de servicios en el protocolo de encaminamiento.
- **Tasa de descubrimiento de servicios**. Es el porcentaje del número de servicios descubiertos, respecto al número total de servicios disponibles en la red. Este valor nos permite comparar entre sí las diferentes versiones del mecanismo de descubrimiento propuesto y también nos permite comparar

este mecanismo con otros protocolos, desde el punto de vista de eficiencia. Lo ideal es que el protocolo nos permita obtener tasas de descubrimiento de servicios del 100 %.

- **Tasa de descubrimientos falsos**. Es el porcentaje del número de servicios descubiertos que en realidad no están disponibles en la red, respecto al número total de servicios descubiertos. Es un valor que nos permite comparar entre sí las diferentes versiones del mecanismo de descubrimiento propuesto y también nos permite comparar este mecanismo con otros protocolos, desde el punto de vista de eficiencia. Lo ideal es que el protocolo nos permita obtener tasas de descubrimiento de servicios falsos del 0 %.

- **Tiempo en descubrir un servicio**. Es el tiempo necesario desde que un nodo quiere hacer uso de un servicio hasta que lo descubre. Idealmente este tiempo debería ser lo más cercano a 0 posible.

A.1.2. El simulador NS-2

El simulador utilizado es el *Network Simulator version 2* (NS-2) [Simulador NS-2]. Se trata de un simulador dirigido por eventos que está escrito en el lenguaje de programación C++ y en el lenguaje de script *Object Tool Command Languaje* (OTcl). OTcl es un intérprete del lenguaje Tcl con extensiones de orientación a objetos. Ambos son lenguajes orientados a objetos, pero C++ es un lenguaje compilado y OTcl es un lenguaje interpretado. Al ser compilado, C++ proporciona gran velocidad en la ejecución, por lo tanto es útil en aquellas partes de la simulación que se ejecutan asiduamente y que no cambian con frecuencia. C++ se utiliza, por lo tanto, para la implementación de protocolos de red cuyo comportamiento es siempre el mismo.

OTcl, en cambio, proporciona más versatilidad debido a que al ser un lenguaje interpretado no es necesario compilarlo por cada cambio realizado, por lo que es idóneo para especificar aquellas partes de la simulación que aunque no se ejecutan con frecuencia, sí cambian muy a menudo. Es decir, OTcl se encarga de generar de forma sencilla los escenarios de simulación y todos los aspectos relacionados con la configuración del escenario, como número de nodos, ancho de banda de los enlaces, ubicación de los nodos, tiempos de ejecución y demás aspectos relevantes de la infraestructura de red.

NS-2 maneja instancias de objetos creados a partir de una jerarquía de clases en C++, lo que se llama jerarquía compilada, y una jerarquía gemela en el intérprete OTcl, que se conoce como jerarquía interpretada. Las dos jerarquías

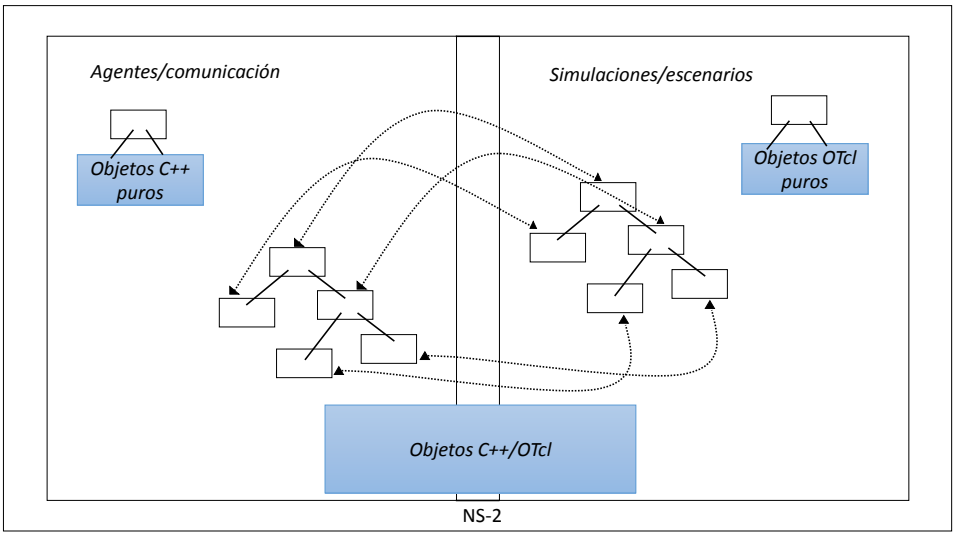

Figura A.1: Correspondencia entre clases C++ y Tcl en NS-2 [Taylor et al, 2010]

están íntimamente relacionadas una con otra. En la figura A.1 se muestra la relación entre ambos lenguajes en el simulador NS-2.

NS-2 utiliza como ficheros de entrada los siguientes ficheros escritos en lenguaje OTcl:

- Fichero de escenario de red que describe el área y la topología de la red MANET.
- Fichero patrón de tráfico de red que describe el patrón de tráfico de la red MANET.
- Fichero de configuración de los nodos que permite especificar los protocolos que se utilizan y sus parámetros de funcionamiento.

A continuación mostramos un ejemplo de los parámetros que se necesitan para configurar un nodo:

```
set val(chan) Channel/WirelessChannel ;# tipo de canal
set val(prop) Propagation/TwoRayGround ;# modelo de propagación radio
set val(netif) Phy/WirelessPhy ;# tipo de interfaz de red
set val(mac) Mac/802_11 ;# tipo de MAC
set val(ifq) Queue/DropTail/PriQueue ;# tipo de cola de la interfaz
set val(ll) LL ;# tipo de capa de enlace
```

```
set val(ant) Antenna/OmniAntenna ;# modelo de antena
set val(ifqlen) 50 ;# máx. Número de paquetes en ifq
```

Para configurar el fichero patrón de tráfico de red, NS-2 utiliza una herramienta que genera de manera eficaz ficheros de movimiento a partir de una serie de parámetros que hay que especificar. La herramienta para definir estos movimientos se llama *setdest* y los parámetros que hay que indicarle para generar el patrón de movimiento son:

```
./setdest [-n num_of_nodes] [-p pausetime] [-M maxspeed] [-t simtime] [-x maxx] \
[-y maxy] > [outdir/movement-file]
```

- Número de nodos que se desean simular.
- Tiempo máximo que se desea que un nodo esté parado.
- Velocidad máxima con la que se desea que se mueva un nodo.
- Duración de la simulación.
- Dimensiones del espacio de simulación.

A continuación se muestra un ejemplo:

```
$ ./setdest -n 50 -p 3.0 -M 20.0 -t 400.0 -x 1000 -y 1000 > scen-50-test
```

De esta forma se genera un fichero llamado `scen-50-test` que contendrá los movimientos de los nodos durante los 400 segundos que dura la simulación, encontrándose dichos nodos siempre en la región 1000 x 1000 metros definida. La velocidad máxima será de 20 m/s y con una pausa máxima de 3 segundos. El fichero generado es de la forma:

```
$ns_ at 10.000000000000 "$node_(0) setdest 319.910607990205
103.452089076870 14.588005844109"
$ns_ at 10.000000000000 "$node_(1) setdest 653.213596366911
532.936544588488 16.195285406108"
$ns_ at 10.000000000000 "$node_(2) setdest 649.128008328524
171.863402430338 9.138711444301"
$ns_ at 10.000000000000 "$node_(3) setdest 244.559690018281
528.949500343645 10.962961204419"
```

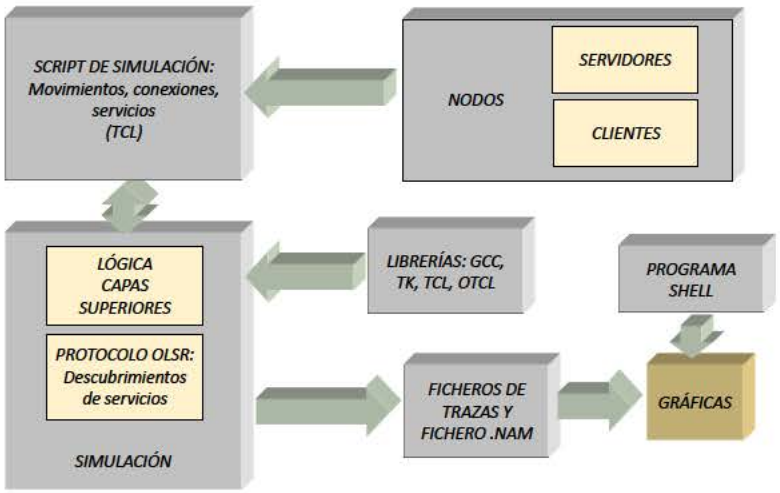

Figura A.2: Esquema global del proceso de simulación

```
$ns_ at 10.000000000000 "$node_(4) setdest 29.852620840386
663.403529580202 7.802536716435"
$ns_ at 10.000000000000 "$node_(5) setdest 611.977759425383
194.837413142943 16.929575763958"
```

La primera línea, por ejemplo, significa que el nodo 0 en el instante 10 segundos se moverá a la posición x = 319.91 y = 103.45 a una velocidad de 14.58 m/s.

Una vez el simulador NS-2 ha simulado el escenario, genera un fichero de trazas con todos los datos relativos a la simulación. Estos datos se pueden extraer fácilmente para obtener los valores de las prestaciones a evaluar. Se utilizan para ello scripts creados por el usuario en *Perl* o *Python*. Opcionalmente, NS-2 también puede generar un fichero de traza para la ejecución de la herramienta visual *Network Animator* (Nam) que permite visualizar de forma gráfica la simulación, de tal forma que se puede observar todo el intercambio de paquetes entre nodos y pérdidas producidos durante la simulación.

En la figura A.2 se muestra un esquema global del proceso de simulación del mecanismo de descubrimiento de servicios.

A.1.3. Nodo móvil NS-2

Debido al carácter inalámbrico de los nodos que conforman los escenarios de simulación, se simularán nodos de tipo *Mobile Node* o nodo móvil. Un nodo móvil

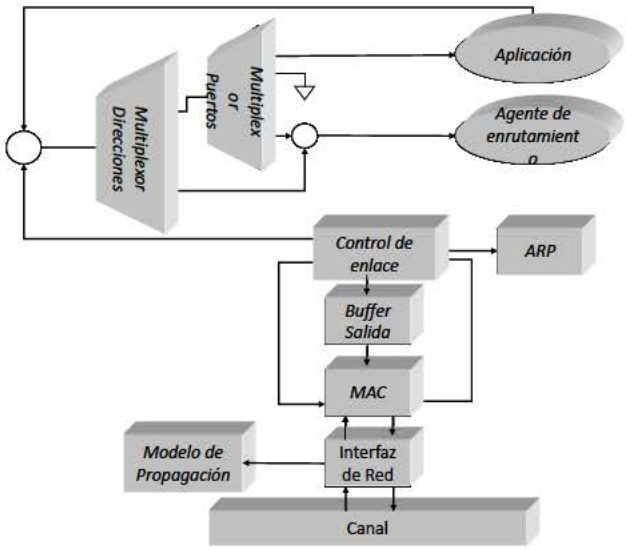

Figura A.3: Estructura de un nodo móvil en NS-2 [Jin et al, 2011]

es una abstracción de cualquier elemento que posea capacidades de comunicación inalámbrica. Esta abstracción interactúa con librerías en las diferentes capas de red a través de las interacciones propias del protocolo IP.

Sin contar con el efecto heterogéneo de los dispositivos, el nodo móvil se convierte en un objeto que asocia agentes dentro del lenguaje de programación, con el fin de permitir al usuario tener la capacidad de generar comportamientos reales dentro del entorno simulado.

Las características dadas al nodo móvil, son esenciales para simular las redes MANET: las condiciones de consumo de energía, el modelo de propagación, el protocolo de encaminamiento, el modelo de la capa MAC *Medium Access Control*, etc.

En la figura A.3 se especifican las partes de las que consta este objeto.

- **Canal**. Representa el medio compartido por todos los nodos, que habilita la conexión entre los diferentes interfaces de red.
- **Interfaz de red (PHY)**. Sirve para emular el interfaz hardware, se encarga de acceder al medio físico y controla los parámetros de potencia de transmisión, energía, tipos de antena...
- **Modelo de propagación**. A partir de la potencia de transmisión, disposición geométrica, y otras características del escenario, determina la potencia

con la que serán recibidas las tramas en el destino. En esta tesis se ha utilizado el modelo de propagación, Two-Ray Ground, para los cálculos de potencia.

- **MAC**. Implementa el protocolo MAC utilizado, principalmente el estándar 802.11.

- **Buffer de salida**. Mantiene los paquetes almacenados hasta que estén listos para enviarse por el medio radio. A través de la clase *PriQueue* se otorga prioridad a los paquetes.

- **Control del enlace**. Conecta el agente de encaminamiento con la tecnología de red subyacente. Tiene asociado un mecanismo de resolución de direcciones ARP (*Address Resolution Protocol*) .

- **ARP**. Se encarga de añadir al paquete la dirección MAC del nodo destino. Si no tuviera esta dirección, el paquete se almacena mientras se envía un paquete *broadcast* para conseguir esta dirección.

- **Multiplexor de direcciones**. Cuando llega un paquete y ha pasado el filtro MAC, en función de la dirección IP del paquete, el multiplexor de direcciones lo procesa y lo envía al segundo multiplexor (si el paquete está dirigido a ese nodo) o al protocolo de encaminamiento para ser retransmitido hacia su próximo salto.

- **Agente de encaminamiento**. Implementa el protocolo de encaminamiento. Mantiene una lista acerca de las rutas para alcanzar todos los destinos. Además gestiona los paquetes de control.

A.1.4. Programación de scripts Tcl para NS-2

Para comenzar un script Tcl, el programa se inicia de la siguiente forma:

```
set ns [new Simulator]
```

La orden *set* inicializa una variable. Esta orden, concretamente, crea una instancia del objeto simulador indicando que se va a realizar una nueva simulación.

Después se abren los ficheros de traza, donde se escribirán los datos obtenidos por el simulador.

```
set tracefd [open traza.tr w]
set namtrace [open traza.nam w]

$ns_trace-all$tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

En este caso, las primeras líneas abren o crean los ficheros de traza *traza.tr* y *traza.nam* donde se escriben y se leen los datos creados en la simulación. Las siguientes líneas guardan en el fichero correspondiente los datos creados por NS.

Es necesario crear en el programa un procedimiento `stop`, que deberá ser definido al principio del programa, el cual cerrará el fichero de trazas y opcionalmente pondrá en marcha la herramienta visual Nam.

```
proc stop{} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit=
}
```

Posteriormente hay que escribir los eventos que queremos que ocurran. Para ello se utiliza la siguiente orden:

```
$ns at <tiempo> <evento>
```

donde *tiempo* será el valor decimal en segundos y *evento* será el procedimiento que queremos que se ejecute en ese instante, por ejemplo con la orden:

```
$ns at 5.0 "stop"
```

estamos indicando al simulador que a los 5 segundos de iniciada la simulación ejecute el procedimiento *stop*.

La configuración de los nodos se realiza, por ejemplo, de la siguiente forma:

```
#configure node
$ns_ node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
```

```
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace OFF \
-energyModel $val(engmodel)\
-rxPower $val(rxPower) \
-txPower $val(txPower) \
-sensePower $val(sensePower) \
-idlePower $val(idlePower) \
-initialEnergy $val(initeng)
```

La última línea del fichero Tcl será la que arranque la simulación:

```
$ns run
```

Envío de comandos a Agentes C++

Los nodos en NS-2 se definen con la instrucción:

```
set <nombre nodo>[$ns node]
```

Siempre que se creen nodos se debe poner el comando *$ns node*.

En NS-2 los datos se envían a través de agentes. Primero hay que crear el agente y luego asignarlo al nodo. Las instrucciones para crear un agente C++ que envíe datos y que posteriormente ese agente se asigne al nodo son las siguientes:

```
#Crear un agente UDP y unirlo al nodo n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
```

El mapeo entre el script Tcl y el agente C++ se hace a través de la función *command*, tal como se explica en la sección B.1.

Envío de comandos a Agentes Java

En el caso de agentes Java, hay que asignar el agente Java al nodo NS-2. Una vez que se hace esta asignación, los agentes Java se comportan de la misma forma

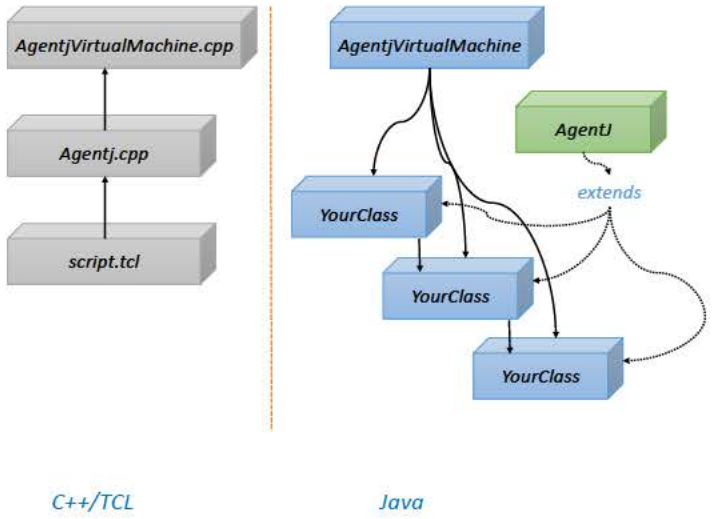

Figura A.4: Interacción entre script Tcl y objetos Java a través del método *command* de la clase *AgentJObject* [Taylor et al, 2010]

que los agentes NS-2 y el mapeo entre el script Tcl y el agente Java también se hace a través de la función *command*, tal como se explica en la sección B.2.

Tal como vemos en la figura A.4 para simular una aplicación Java en NS-2, el usuario únicamente tiene que escribir el script Tcl que simule el entorno que desea simular y programar la clase o clases Java que implementen el funcionamiento del mecanismo o protocolo que quiera simular. El agente C++ NS-2(*Agentj.cpp*) actúa únicamente como intermediario entre el script Tcl y el objeto Java vía la clase C++ *AgentVirtualMachine.cpp* y su correspondiente clase Java *AgentJVirtualMachine.java*.

En el script Tcl esto se traduce en que se crea un nodo NS-2, se crea una agente Java y se asigna este agente al nodo:

```
$set n1 [$ns_node]
$set ragent [New Agent/Agentj]
$ns_ attach-agent $n1 $ragent
```

Gráficamente lo vemos en la figura A.5.

Una vez que se ha declarado el agente, hay que indicarle el objeto Java que tiene que usar el agente:

```
$ns_ at 0.0 "$ragent attach-agentj agentj.olsrv2routing.SDOLSRv2RoutingAgent
```

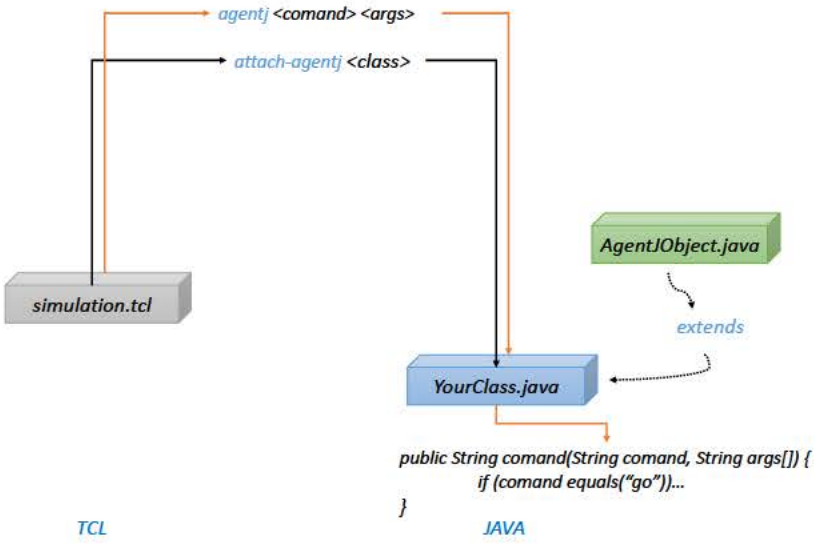

Figura A.5: Interfaz para declarar un agente Java [Taylor et al, 2010]

Una vez que se ha creado el objeto Java, ya se pueden enviar comandos utilizando la instrucción Tcl:

```
agentj <command> <args>
```

Un comando obligatorio que hay que ejecutar cuando se trata de implementaciones de protocolos realizadas en Java es el que arranca el protocolo. En nuestro caso:

```
$ns_ at 0.0 "$ragent agentj startProtocol"
```

Este comando *startProtocol* tiene su correspondiente implementación en la clase Java *SDOLSRv2RoutingAgent* tal como se ha explicado en la sección 5.2.2.

A.2. Cálculo de resultados

Los parámetros de entrada de las simulaciones realizadas son los siguientes:

- **Número medio de nodos que integran la red**. Con este parámetro queremos medir el comportamiento del mecanismo propuesto en redes de diferentes tamaños.

- **Periodicidad con la que se envían los mensajes de anuncio de servicio**. Con este parámetro queremos medir el comportamiento del mecanismo propuesto con diferentes frecuencias para el envío de anuncios de servicios.
- **Número de nodos ofreciendo el mismo servicio**. Con este parámetro queremos medir el comportamiento del mecanismo propuesto cuando en la red aumenta el número de nodos ofreciendo el mismo servicio.
- **Velocidad a la que se mueven los nodos**. Con este parámetro queremos medir el comportamiento del mecanismo propuesto en redes más y menos móviles.

Como resultado de las simulaciones hemos seleccionado las siguientes variables:

- **Número de mensajes transmitidos**. Es el número de mensajes transmitidos en la red comparado con el número de mensajes transmitidos del protocolo de encaminamiento sin llevar integrado el mecanismo de descubrimiento de servicios.
- **Tasa de descubrimiento de servicios**. Es el porcentaje del número de servicios descubiertos, respecto al número total de servicios disponibles en la red.
- **Tasa de descubrimientos falsos**. Es el porcentaje del número de servicios descubiertos que en realidad no están disponibles en la red, respecto al número total de servicios descubiertos.
- **Tiempo en descubrir un servicio**. Es el tiempo necesario desde que un nodo quiere hacer uso de un servicio hasta que lo descubre. Idealmente este tiempo debería ser lo más cercano a 0 posible.

El escenario de simulación base del que partimos en las simulaciones realizadas es el siguiente:

- Número medio de nodos: 50 nodos distribuidos de forma aleatoria en un área de 1000x1000 metros.
- Se fija la duración de las simulaciones en 400 segundos.
- Durante las simulaciones los nodos se mueven según el modelo Random Waypoint.

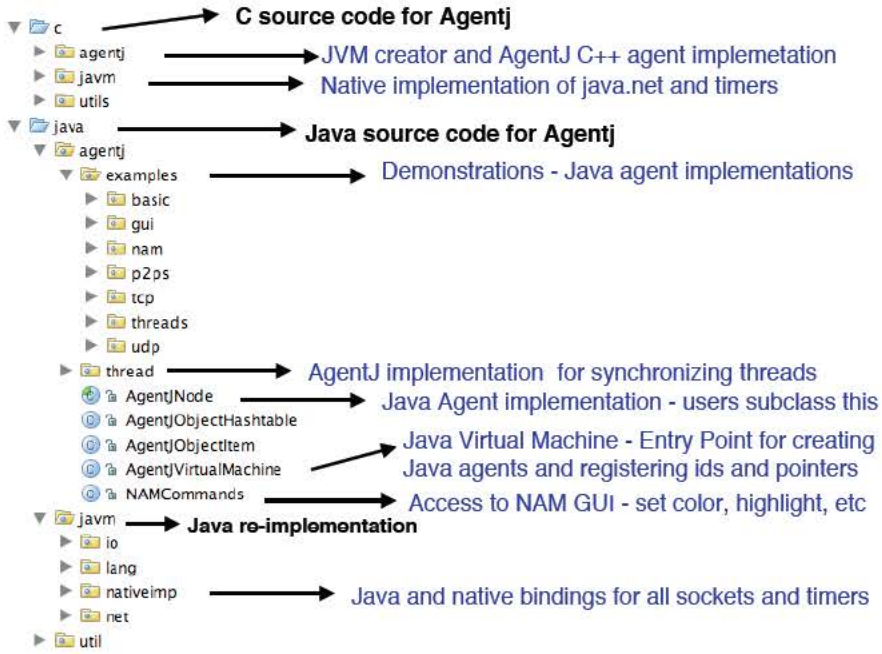

Figura A.6: Estructura Framework Agentj

- Para el modelo de propagación se ha elegido una aproximación del modelo Two-Ray Ground.

Para simular el mecanismo de descubrimiento de servicios sobre OLSRv1 partimos de la implementación en C++ del protocolo OLSR que se encuentra en el siguiente enlace [UM-OLSR].

Sin embargo para simular la versión 2 del protocolo OLSR utilizamos una implementación del protocolo OLSR realizada en Java, [Bross, 2013] y la modificamos para soportar el mecanismo de descubrimiento de servicios. Para poder simular y compilar un código escrito en Java en el simulador NS-2 utilizamos el *framework* AgentJ [Taylor et al, 2006]. Aunque el desarrollo y la integración del *framework* AgentJ en NS-2 es complejo, de cara a los usuarios la simulación de aplicaciones Java en NS-2 sigue las mismas reglas que las simulaciones de aplicaciones en C++. Del mismo modo, el desarrollo de aplicaciones Java que se puedan simular en NS-2 sigue las mismas reglas que cualquier aplicación Java.

El framework AgentJ se compone de una serie de clases escritas en C++ y Java. La estructura del framework está organizada como si se tratara de una aplicación Java. Lo podemos ver en la figura A.6.

El directorio *src* contiene un directorio *c* y otro directorio *java*, que contienen el código fuente C++ y Java referente al framework Agentj. El directorio *c* contiene a su vez tres directorios más. El primero, *agentj*, contiene el código del agente NS-2 (*AgentJ.cpp*). También instancia un objeto *AgentjVirtualMachine* que crea una máquina virtual Java para interactuar con el código Java. El segundo directorio, *javm*, contiene las clases para implementar una interfaz JVM que se enlaza a las clases de la librería Protolib para su integración en NS-2. Protolib (*Protean protocol prototyping library*) es un conjunto de herramientas en forma de clases C++ que facilitan el desarrollo de cualquier software de red independiente de la plataforma.

Por último, el directorio *java*, contiene el correspondiente directorio *agentj* pero en este caso en el lado Java, que implementa un agente Java. También contiene el recíproco a la clase *AgentVirtualMachine* del código C++, que se llama *AgentJVirtualMachine.java* que contiene los métodos con los que interactúa la clase C++ *AgentjVirtualMachine*.

El funcionamiento es el siguiente: en primer lugar, el agente NS-2 instancia y asigna el agente Java al nodo y en segundo lugar, el agente Java accede a las clases de la librería Protolib para poder intercambiar información entre los nodos NS-2. Para instanciar y asignar el agente Java al nodo, el agente NS-2 crea una JVM utilizando las clases *AgentVirtualMachine.cpp* y *AgentVirtualMachine.java* para de esta forma crear y acceder a los objetos Java. Para realizar el mapeo entre las clases Java y las clases de la librería Protolib se utiliza JNI (*Java Native Interface*).

En las figuras A.7 y A.8 se muestra esquemáticamente el funcionamiento descrito.

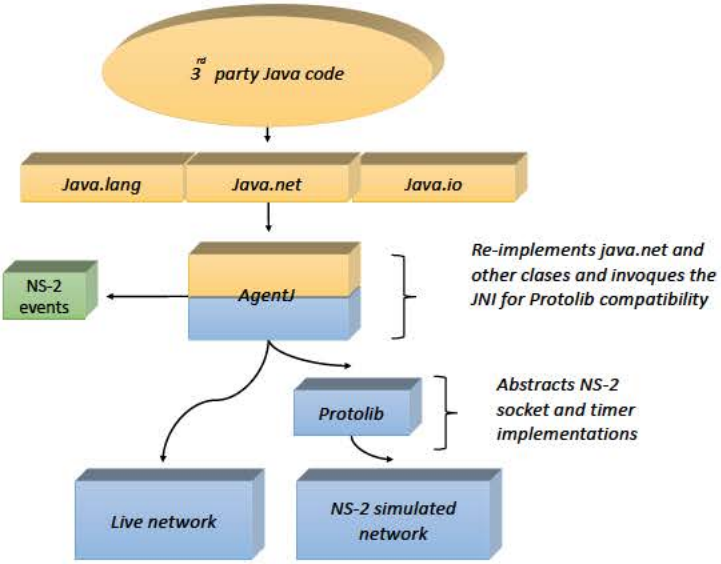

Figura A.7: Framework Agentj [Taylor et al, 2010]

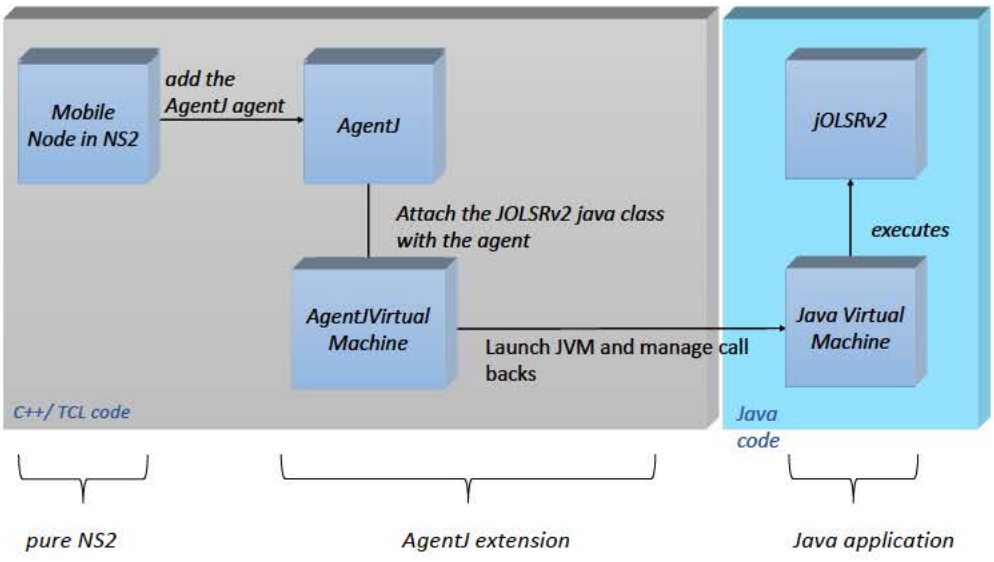

Figura A.8: Framework Agentj con SD-OLSRv2

Apéndice B

Instrucciones Tcl

Para generar un escenario de simulación utilizamos un fichero definido en formato Tcl. En este fichero se definen los movimientos, los protocolos que se utilizan y muchos más parámetros que deben tenerse en cuenta a la hora de simular un escenario determinado.

Para poder simular la implementación del mecanismo de descubrimiento de servicios integrado en OLSRv1 y en OLSRv2 hemos introducido una serie de comandos accesibles desde el fichero Tcl:

- Comando que permite a un nodo solicitar un servicio. Esto traerá consigo el envío de un mensaje SDM de petición del servicio solicitado. Este comando sólo será necesario a la hora de simular el mecanismo de descubrimiento de servicios sobre OLSRv1, cuando en el mismo definíamos peticiones y anuncios de servicios.

- Comando que permite que un nodo ofrezca un nuevo servicio en un instante determinado. Este comando se utiliza tanto en OLSRv1 como en OLSRv2.

- Comando que permite visualizar en el fichero de trazas generado por el simulador NS-2 las cachés de servicios de los nodos. De esta forma podemos ver la evolución de las mismas y no sólo su estado final. Este comando lo utilizamos únicamente cuando simulamos un escenario con SD-OLSRv1.

En los siguientes apartado veremos cómo se han introducido estos cambios.

B.1. Introducción de nuevas instrucciones Tcl en OLSRv1

Para la interacción entre el código Tcl y C++, existe una función *command* definida en el agente OLSR (fichero *OLSR.cc*) que permite hacer el mapeo entre las instrucciones Tcl del escenario de simulación en NS-2 y el código C++ del protocolo OLSRv1.

Esta función *command* recibe dos parámetros:

- *argc*. Entero que indica el número de argumentos.
- *argv*. Array de cadena de caracteres con los argumentos en sí. El primer valor de esta cadena de caracteres es el valor del método invocado. Este valor siempre será *cmd*. Este método llama a la función *command* del agente de encaminamiento. El método *cmd* será capaz de distinguir a qué agente de encaminamiento se llama.

Hacer hincapié en que un agente OLSR es un agente de encaminamiento. Todos los agentes de encaminamiento se encuentran unidos al puerto 255. Si queremos que un comando escrito en el script de simulación Tcl llegue a un agente de encaminamiento, el comando tiene que especificar que hace referencia al puerto 255.

El funcionamiento de la función *cmd()* es muy sencillo. Siempre llamará a la función *TclObject:command()* pasándole como parámetros *argc* y *argv[]*. En el caso que nos ocupa *TclObject* será *OLSR*. Por tanto, la función será:

```
int OLSR::commnad(int argc, const char*const* argv){
```

Tal como hemos dicho, *argv[0]* siempre valdrá *cmd*. Pero, el resto de los parámetros de este array serán variables y serán los que hagan que diferentes acciones se lleven a cabo. En concreto, *argv[1]* llevará el nombre de la acción que se quiera llevar a cabo. Por su parte, los valores *argv[>1]* serán valores que ayuden a cumplir la función especificada en *argv[1]*. A continuación vemos un ejemplo:

```
$ns_ at 28.9 "[$node_(3) agent 255] print_rtable"
```

permite escribir la tabla de encaminamiento del nodo 3 en el fichero de trazas en el instante 28.9. Como se puede observar, se especifica el puerto 255 y el agente

OLSR. Se aprecia que la palabra OLSR no aparece implícitamente. Pero, en el fichero de simulación los nodos son definidos como agentes OLSR, y luego se tratan con su número de nodo, en este caso el 3.

B.2. Introducción de nuevas instrucciones Tcl en OLSRv2

En la clase auxiliar que hemos creado *SDOLSRv2RoutingAgent* definimos el método *command* que es al que se llama desde el agente AgentJ en NS-2. Es decir, para la interacción del script Tcl utilizado para las simulaciones y el agente Java, utilizamos el método *command* creado en la clase auxiliar *SDOLSRv2RoutingAgent*.

```
public boolean command(String command, String[] args) {
```

El primer parámetro hace referencia al comando que se ejecuta desde el simulador y el segundo a los posibles argumentos que puede llevar este comando. Es obligatorio en el caso de protocolos de encaminamiento escritos en Java que desde el simulador, el agente Java arranque el protocolo de encaminamiento.

```
$ns_ at 0.0 "[$node_($i) set ragent_] agentj startProtocol"
```

A continuación se analizarán las nuevas instrucciones creadas para simular el mecanismo de descubrimiento de servicios.

B.3. Envío de una petición de servicio

Esta petición es necesaria cuando en la primera versión del mecanismo de descubrimiento de servicios integrado en OLSRv1 un nodo necesita hacer uso de un servicio, busca en su caché de servicios, no dispone de ese servicio y lanza un mensaje de petición de servicio a la red.

El formato de la instrucción creado es el siguiente:

```
$ns_ at xx.x "[$node_(y) agent 255] queryservice zzz"
```

donde *xx.x* representa el instante de tiempo, *y* representa el nodo al que se hará referencia y *zzz* el servicio que el nodo está solicitando.

Una vez introducida esta instrucción en el código Tcl, se llegará a la función *command* del fichero *OLSR.cc* y se deberá comprobar si las acciones requeridas están implementadas:

```
else if(strcmp(argv[1], "queryservice") == 0)
```

En caso afirmativo se comprobará qué servicio se está buscando y si se encuentra almacenado en la caché de servicios. El proceso es como el que se ha visto en la sección 4.2.3: abrir el fichero, leer línea a línea y compararlo con el valor introducido en el script Tcl.

La existencia o no del servicio en la caché de servicios se controlará mediante una variable denominada *service_present*. Esta variable tendrá un valor igual a cero si el servicio no está almacenado, y, por tanto, hay que mandar un mensaje de petición; y, valdrá uno cuando el servicio sí que se encuentre almacenado, y, debido a ello, no haya que enviar ningún mensaje solicitando el servicio.

B.4. Añadir un servicio a un nodo

En una red MANET un nodo puede adquirir un servicio o dejar de ofrecerlo en cualquier momento. Los servicios en una red MANET no son estáticos ni estacionarios en el tiempo.

Para que en SD-OLSRv1 un nodo pueda publicar su servicio en cualquier momento generamos la siguiente instrucción:

```
$ns_ at xx.x "[$node_(y) agent 255] service zzz www"
```

Para que un nodo publique su servicio en SD-OLSRv2 en cualquier momento generamos la siguiente instrucción:

```
$ns_ at xx.x "[$node_($i) set ragent_] agentj service zzz www"
```

En ambos casos *zzz* describe el servicio que se quiere introducir y *www* el tiempo de vida de dicho servicio.

El proceso es exactamente igual que en el caso de petición de un servicio, B.3. Se comprueba que la sintaxis del comando es correcta y se almacena el contenido en variables para posteriormente escribir el servicio en la caché de servicios.

B.5. Mostrar memorias caché en el fichero de trazas resultante

Esta instrucción únicamente está disponible para el mecanismo de descubrimiento de servicios integrado en OLSRv1. El formato de esta instrucción es el siguiente:

```
$ns_ at xx.x "[$node_(y) agent 255] print_services"
```

Cuando se introduce esta instrucción en el código Tcl, la función *command* del fichero *OLSR.cc*, lo primero que se hace es comprobar la sintaxis, es decir, que la opción introducida, *print_services*, es correcta. A continuación se comprueba que exista un fichero destino donde escribir. En caso de que la variable *logtarget* sea igual a NULL, se muestra un mensaje de error indicando que se debe crear un fichero de trazas.

En caso de que sí que se haya creado el fichero de trazas, se escribirá el título de lo que se va a escribir:

```
P 72.000 _0_ Available services
```

Finalmente se llama a la función *print_services* perteneciente a la clase *OLSR_printer*, pasándole el fichero en el que se debe escribir y el nodo del cual se quiere mostrar el contenido de la caché de servicios (Cuadro B.1).

El funcionamiento de esta función *print_services* es muy sencillo. Una vez que se conoce el nodo del cual se quiere mostrar la caché de servicios se abre el fichero donde se almacenan los servicios en modo lectura y se va leyendo línea a línea. Esa línea se escribe en el fichero de trazas que también se ha recibido como parámetro. A continuación se muestra un ejemplo del resultado que se muestra en el fichero de trazas:

```
P 72.000 _0_ Available services
P owner service TTL
P   000 000 100
P   005 005 255
```

```
 else if (strcasecmp(argv[1], "print_services") == 0) {
   if (logtarget_ != NULL) {
     sprintf(logtarget_->pt_->buffer(), "P %f _%d_ Available services",
                                        CURRENT_TIME,
                                        OLSR::node_id(ra_addr()));
                               logtarget_->pt_->dump();
     OLSR_printer::print_services(logtarget_, OLSR::node_id(ra_addr()));
   }

  else {
    fprintf(stdout, "%f _%d_ If you want to print the services "
   "you must create a trace file in your tcl script",
    CURRENT_TIME,
    OLSR::node_id(ra_addr()));
  }
 return TCL_OK;
}


  else if (strcasecmp(argv[1], "print_services") == 0) {
      if (logtarget_ != NULL) {
          sprintf(logtarget_->pt_->buffer(), "P %f _%d_ Available services",
                                             CURRENT_TIME,
                                             OLSR::node_id(ra_addr()));
          logtarget_->pt_->dump();
          OLSR_printer::print_services(logtarget_, OLSR::node_id(ra_addr()));
      }
      else {
          fprintf(stdout, "%f _%d_ If you want to print the services you must create a
                          trace file in your tcl script", CURRENT_TIME,
                                                          OLSR::node_id(ra_addr()));
  }
  return TCL_OK;
}
```

Cuadro B.1: Mostrar memorias cache en el Fichero de trazas resultante

Glosario de términos

A

ANSN Advertised Neighbour Sequence Number.

AODV Ad-hoc On-Demand Distance Vector.

ARP Address Resolution Protocol

C

CEDAR Core Extraction Distributed Ad Hoc Routing.

D

DARPA Defense Advanced Research Projects Agency.

DAML DARPA Agent Markup Languaje.

DAs Directory Agents.

DSDV Destination-sequenced Distance Vector.

DSR Dynamic Source Routing.

DYMO Dynamic MANET On-demand.

F

FSR Fisheye State Routing.

G

GSD Group-based Service Discovery protocol.

GSR Global State Routing.

HTTP Hypertext Transfer Protocol.

I

IANA Internet Assigned Numbers Authority.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IoT Internet of Things.

IP Internet Protocol.

IPP Internet Printing Protocol.

ISO International Standard Organization.

J

JLS Jini Lookup Services.

JNI Java Native Interface.

JVM Java Virtual Machine.

L

LANMAR Landmark Routing.

LAR Location aided routing.

LPR Low-Cost Packet Radio.

LSP Link State Packet.

M

MAC Medium Access Control

MANET-WG MANET Working Group.

MANET Mobile Ad hoc Networks.

MPR MultiPoint Relay.

MSD Mercury Service Discovery.

N

Nam Network Animator.

NHDP Neighborhood Discovery Protocol.

NS-2 Network Simulator version 2.

O

OLSR Optimized Link State Routing Protocol.

OSI Open Systems Interconnection.

OSPF Open Shortest Path First.

OTcl Object Tool Command Languaje.

P

P2P Peer to Peer.

PDP Pervasive Discovery Protocol.

PRNETs Packet Radio Networks.

PSTN Public Switched Telephone Nerwork.

Q

QoS Quality of Service

R

RFC Request for Comments.

RMI (Java) Remote Method Invocation.

RREQ Routing Request Packets.

RREP Routing Reply.

S

SAs Service Agents.

SD-OLSRv2 Service Discovery over OLSRv2.

SD-OLSR Service Discovery over OLSR.

SDM Service Discovery Message.

SDP Service Discovery Protocol.

SI Swarm Intelligence.

SIP Session Initiation Protocol.

SLE Service Location Extension.

SLMs Salutation Managers.

SLP Service Location Protocol.

SREQ Service Discovery Requests.

SREP Service Discovery Replays.

SRMM Smooth Random Mobility Model.

SSDP Simple Service Discovery Protocol.

SURAN Survivable Radio.

SVRLOC Service Location Protocol Working Group.

T

TBRPF Topology Dissemination Based on Reverse-Path Forwarding.

TC Topology Control.

TLV Type Lengh Value.

TMs Transport Managers.

TORA Temporally-Ordered Routing Algorithm.

TTL Time To Live.

U

UAs User Agents.

UDP User Datagram Protocol.

UPnP Universal Plug and Play.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

W

WRP Wireless Routing Protocol.

X

XML eXtensible Markup Language.

Z

ZRP Zone Routing Protocol.

Bibliografía

[Abramson, 1970] Abramson, N. (1970) THE ALOHA SYSTEM - Another Alternative for Computer Communications. *In Proceedings of the Fall Joint Computer Conference* (pp. 281–286), *37*, Houston, Texas.

[Ahvar et al, 2007] Ahvar, E. and Fathy, M.(2007). Performance Evaluation of Routing Protocols For High Density Ad Hoc Networks based on Energy Consumption by GlomoSim Simulator. *World Academy of Science, Engineering and Technology,5*, 97–100.

[Antonopoulos et al, 2014] Antonopoulos, A., Lalos, A., di Renzo, M. and Verikoukis, C. (2014). Cross-layer Theoretical Analysis of NC-Aided Cooperative ARQ Protocols in Correlated Shadowed Environments. *IEEE Trans. Veh. Technol.*, doi:10.1109/TVT.2014.2361670.

[Athanaileas et al, 2007] Athanaileas, S.E., Ververidis, C. and Polyzos, G.C. (2007). Optimized Service Selection for MANETs using an AODV-based Service Discovery Protocol. *The 6th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Corfu, Greece.

[Atzori et al, 2010] Atzori, L., Iera, A. and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, *54*(15), 2787–2805.

[Apache Software Foundation, 2013] Apache Software Foundation (2013). Apache River - JiniTM Network Technology Specifications, 2.2.2 edn.

[Bellman, 1957] Bellman, R.E. (1957). Dynamic Programming. *Princeton University Press*, Princeton, NJ 1957.

[Bettstetter, 2000] Bettstetter, C. and Renner, C. (2000). A comparison of service discovery protocols and implementation of the service location protocol. *6th EUNICE Open European Summer School: Innovative Internet Applications.*

[Bhumika, 2015] Bhumika, G., Zaveri, M.A. and Kumar, H. (2015). Trust Based Service Discovery in Mobile Ad-Hoc Networks. *Lect. Notes Softw. Eng.*, *3*, 308–313.

[Bross, 2013] Bros, A., Israel, A. and Nazarov, E. (2013). OLSRv2 Simulation (OLSRsim). http://code.google.com/p/olsrsim/source/browse/trunk/OLSRv2/?r=265 (Accessed on 08 November 2015).

[Buvana et al, 2015] Buvana, M.; Suganthi, M. (2015). An Efficient Cluster Based Service Discovery Model for Mobile Ad hoc Network. *KSII Trans. Internet Inf. Syst. (TIIS)*, *9*, 680–699.

[Campo et al, 2006] Campo, C., García-Rubio, C., Marín, A. and Almenárez, F. (2006). PDP: A lightweight discovery protocol for local-scope interactions in wireless ad hoc networks. *Computer Networks,50*, 3264–3283.

[Campo C., 2004] Campo, C. (2004). Tecnologías middleware para el desarrollo de servicios en entornos de computación ubicua. http://e-archivo.uc3m.es/handle/10016/523

[Chakeres et al, 2007] Chakeres, I.D. and Belding-Royer, E.M. (2004). AODV routing protocol implementation design. *The 24th International Conference on Distributed Computing Systems Workshops*(pp. 698–703), Hachioji, Japan.

[Chakeres et al, 2009] Chakeres, I. and Perkins, Ch. E. (2009). Dynamic MANET On-demand (DYMO) Routing. Internet Draft. (Work in progress). http://tools.ietf.org/html/draft-ietf-manet-dymo-17.

[Chakraborty, 2001] Chakraborty, D. (2001). Service Composition in Ad-hoc Environments. Technical Report TR-CS-01-20, Department of Computer Science and Electrical Engineering. University of Maryland, Baltimore Country. Baltimore.

[Chakraborty et al, 2006] Chakraborty, D., Joshi, A., Yesha, Y. and Finin, T. (2006). Toward distributed service discovery in pervasive computing environments. *IEEE Trans. Mob. Comput.*, *5*, 97–112.

[Chen al, 1998] Chen, T. and Gerla, M. (1998). Global State Routing: a New Routing Scheme for AdHoc Wireless Networks. *The IEEE International Conference on Communications*(pp. 171–175), *1*, San Francisco, CA, USA.

[Cheshire et al, 2013] Cheshire, S. and Krochmal, M.(2013). RFC 6763: DNS-based service discovery. (Proposed Standard). https://tools.ietf.org/html/rfc6763 (Accessed on 08 November 2015).

[Clark, 1985] Clark, D.D. (1985). The Structuring of Systems using Upcalls.*10th ACM Symposium on Operating Systems Principels (SOSP 85)*(pp 171–180), Orcas Island, Wash, USA.

[Clark et al, 1990] Clark, D.D. and Tennenhouse, D.L. (1990). Architectural Considerations for a new Generation Protocols. *The 1990 Symposium on Communications Architectures and Protocols (SIGCOMM 90)*(pp. 200–208), ACM Press, New York, NY ,USA.

[Clausen et al, 2014] Clausen, T., Dearlove, C., Jacquet, P. and Herberg, U. (2014). RFC 7181: The Optimized Link State Routing Protocol Version 2. https://tools.ietf.org/html/rfc7181 (Accessed on 08 November 2015).

[Clausen et al, 2003] Clausen, T. and Jacquet, P. (2003).
Optimized Link State Routing Protocol (OLSR).
https://tools.ietf.org/html/rfc3626 (Accessed on 08 November 2015).

[Clausen et al, 2011] Clausen, T.; Dearlove, C. and Dean, J. (2011).
RFC 6130: Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP).
https://tools.ietf.org/html/rfc6130 (Accessed on 08 November 2015).

[Clausen et al, 2015] Clausen, T., Dearlove, C., Dean, J. and Adjih, C. (2009)
RFC 5444: Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format.
https://tools.ietf.org/html/rfc5444. (Accessed on 08 November 2015).

[Clausen et al, 2009] Clausen, T. and Dearlove, C. (2009). RFC 5497: Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs). https://tools.ietf.org/html/rfc5497. (Accessed on 08 November 2015).

[Clausen et al, 2008] Clausen, T., Dearlove, C. and Adamson, B. (2008). RFC 5148: Jitter Considerations in Mobile Ad Hoc Networks (MANETs). https://tools.ietf.org/html/rfc5148. (Accessed on 08 November 2015).

[Cui et al, 2011] Cui, Y., Li, Z. and Jiang, S. (2011). Adaptive deployment of directory for service discovery in MANET. *Procedia Engineering*, *15*(0), 2971–2975.

[Czerwinski et al, 1999] Czerwinski, S. E., Zhao, B. Y., Hodes, T. D., Joseph, A. D., and Katz, R. H. (1999). An architecture for a secure service discovery service. In *Proc. Mobicom'99.*

[Dahiya et al, 2013] Dahiya, V., Sangwan, R., Duhan, M. and Dalal, K. (2013). OLSR v2 Implementation and Performance Evaluation over OLSRv1 in MANET using QualNet 6.1. *International Journal of Scientific & Engineering Research*, *4*(7), 2458–2463.

[Dabrowski et al, 2005] Dabrowski, C.E., Mills, K.L. and Quirolgico, S. (2005). A model-based analysis of first-generation service discovery systems. Tech. Rep. ADA523379.*NIST National Institute of Standards and Technology*, Gaithersburg, MD, USA.

[Engelstad et al, 2005] Engelstad, P. E. and Zheng, Y. (2005). Evaluation of service discovery architectures for mobile ad hoc networks. *The Second Annual Conference on Wireless On-demand Network Systems and Services* (pp. 2–15), IEEE Computer Society.

[Engelstad et al, 2006] Engelstad, P.E., Zheng, Y., Koodli, R. and Perkins, C.E. (2006). Service discovery architectures for on-demand ad hoc networks. *Int. J. Ad Hoc Sens. Wirel. Netw.*, *2*, 27–58.

[Fan et al, 2007] Fan, Z. and Guerreiro, E. (2007).Service Discovery in Ad Hoc Networks: Performance Evaluation and QoS Enhancement. *Wireless Personal Communications*, *40*, 215–231.

[Fifer et al, 1987] Fifer W. and Bruno, F. (1987). The low-cost packet radio. *Proceedings of the IEEE*, *75*(1), 33–42.

[Flathagen et al, 2008] Flathagen, J. and Øvsthus, K. (2008). Service Discovery Using OLSR and Bloom Filters. *The 4th OLSR Interop & Workshop*(pp. 14–16), Ottawa, ON, Canada.

[Ford et al, 1962] Ford L.R. and Fulkerson D.R. (1962). Flows in Networks. Princeton University Press, Princeton, NJ.

[Freebersyser, 2001] Freebersyser, J. A. and Leiner, B. (2001). A DoD perspective on mobile Ad hoc networks. *Ad Hoc Networking* (pp. 29–51), Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[García-Macías et al, 2005] García-Macías, J.A. y Torres, D.A. (2005). Service discovery in mobile ad-hoc networks: Better at the network layer.*the International Conference Workshops on Parallel Processing* (pp. 452–457), Oslo, Norway.

[Gerla et al, 2002] Gerla, M., Hong, X. and Pei, G. (2002). Fisheye State Routing Protocol (FSR) for Ad Hoc Networks. Internet Draft.

http://tools.ietf.org/html/draft-ietf-manet-fsr-03. (Accessed on 08 November 2015)

[Goland et al., 1999] Goland, Y. Y., Cai, T., Leach, P., and Gu, Y. (1999). Simple Service Discovery Protocol/1.0. Internet-Draft (work in progress). draft-cai-ssdp-v1-03.txt.

[Gupta, 2003] Gupta S.(2003). Implementation of Service Discovery in Mobile Ad-Hoc Networks in ns-2.26 (with AODV as Routing Protocol). http://www.geocities.ws/mailtosandy/html/svcdisc.html (Accessed on 08 November 2015)

[Guttman, 1999] Guttman, E. (1999). Service location protocol: automatic discovery of ip network services. *IEEE Internet Computing*, *3*(4), 71–80 .

[Guttman et al, 1999] Guttman, E., Perkins, C., and Kempf, J. (1999). RFC 2608: Service Location Protocol, Version 2. Updated by RFC 3224 https://tools.ietf.org/html/rfc2608 (Accessed on 08 November 2015).

[Guttman et al, 1999] Guttman, E., Perkins, C., Veizades, J. and Day, M. (1999). RFC 2609: Service Templates and Service: Schemes https://tools.ietf.org/html/rfc2609 (Accessed on 08 November 2015).

[Haas et al, 2002] Haas, Z.J.; Pearlman, M. R. and Samar P. (2002). The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet Draft, draft-ietf-manet-zone-zrp-04.txt.

[Halkes et al, 2006] Halkes, G.P., Baggio, A. and Langendoen, K.G. (2006). A Simulation Study of Integrated Service Discovery. *The First European Conference, EuroSSC 2006* (pp. 39–53), Enschede, The Netherlands.

[Herberg U., 2009] Herberg, U. (2009). Integrating Java Support for Routing Protocols in NS2. *Research Report (RR-7075)*, p. 32.

[Jain, 2003] Jain, S. (2003). Energy Aware Communication in Ad - Hoc Networks. Technical Report ,UW-CSE 03-06-03, Dept. of Computer Science and Engineering, University of Washington, Seattle.

[Jayakumar et al, 2007] Jayakumar, G. and Gopinath, G. (2007). Ad Hoc Mobile Wireless Networks Routing Protocols. *A Review. Journal of Computer Science,3*(8), 574–582.

[Jayapal et al, 2011] Jayapal, C. and Vembu, S. (2011). Adaptive Service Discovery Protocol for Mobile Ad Hoc Networks. *European Journal of Scientific Research*, *49*(1), 6–17.

[Jin et al, 2011] Jin, C. and Kunz, T. (2011). Smart Home Networking: Lessons from Combining Wireless and Powerline Networking. *Smart Grid and Renewable Energy*, (2), 136–151.

[Jini, 1999] Jini (1999). Jini Architectural Overview. White Paper. https://river.apache.org/doc/specs/html/jini-spec.html (Accessed on 08 November 2015).

[Jodra et al, 2006] Jodra, J.L., Vara, M., Cabero, J. y Bagazgoitia, J. (2006). Service Discovery Mechanism Over OLSR for Mobile Ad Hoc Networks. *The 20th International Conference on Advanced Information Networking and Applications (AINA 2006)* (pp. 18–20), Vienna, Austria.

[Jonson et al, 2007] Jonson, D., Hu, Y. and Maltz, D. (2007). RFC 4728: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. *Internet Engineering Task Force.*

[Jubin et al, 1987] Jubin J. and Tornow J. D. (1987). The DARPA Packet Radio Network Protocols. *Proc. of the IEEE*, *75*(1), 21–32.

[Ko et al, 1998] Ko, Y.-B. and Vaidya N. H. (1998). Location Aided Routing in Mobile Ad Hoc Networks. *Proc. of the ACM/IEEE Mobicom.*

[Koodli et al, 2002] Koodli, R. and Perkins, C.E. (2002). Service discovery in on-demand ad hoc networks. Internet draft, IETF ManetWorking Group draft-koodli-manet-servicediscovery-00.txt (expired).

[Li et al, 2005] Li, L. and Lamont, L. (2005). A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-Layer Design. *The Third IEEE International Conference on Pervasive Computing and Communications Workshop* (pp. 55–59), Kauai Island, HI, USA.

[Mian et al, 2009] Mian, A., Baldoni, R. and Beraldi, R. (2009). A survey of Service Discovery protocols in multihop mobile ad hoc networks. *IEEE Pervasive Comput. Mag.*, *8*(1), 66–74.

[Miller and Pascoe, 2000] Miller, B. A. and Pascoe, R. A. (2000). Salutation service discovery in pervasive computing environments. Technical report, IBM White Paper.

[Monire et al, 2012] Monire, M.(2012). Cluster Based Cross Layer Intelligent Service Discovery for Mobile Ad-Hoc Networks. *J. Artif. Intell. Electr. Eng.*, *1*, 20–28.

[Moy, 1998] Moy, J. (1998). OSPF Version 2. RFC 2328: Internet Engineering Task Force. https://tools.ietf.org/html/rfc2328 (Accessed on 08 November 2015)

[Murphy et al, 1996] Murphy, S., Garcia-Luna-Aceves J.J., (1996). An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, 183–197.

[Neogy et al, 2012] Neogy, R., Chowdhury, C. and Neogy, S.(2012). A Reliable Service Discovery Protocol Using Mobile Agents in MANET. *The IEEE Reliability and Maintain-ability Symposium (RAMS)* (pp.1–7), Reno, NV, USA.

[Nidd, 2001] Nidd, M. (2001). Service Discovery in DEAPspace. *IEEE Personal Communications*.

[Obaid et al, 2007] Obaid, A., Khir, A. and Mili, H. (2007). A Routing Based Service Discovery Protocol for Ad hoc networks. *Proceedings of the Third International Conference on Networking and Services*, p.108, Athens, Greece.

[Ogier et al, 2004] Ogier, R., Templin F. and Lewis, M. (2004).
RFC 3684: Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). https://tools.ietf.org/html/rfc3684 (Accessed on 08 November 2015)

[Palmieri et al, 2013] Palmieri, F. (2013). Scalable service discovery in ubiquitous and pervasive computing architectures: A percolation-driven approach. *Future Gener. Comput. Syst.*, *29*, 693–703.

[Park et al, 2001] Park, V. and Corson, S. (2001). Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. *Internet Draft.* http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04. (Accessed on 08 November 2015)

[Pascoe, 1999] Pascoe, B. (1999). Salutation-Lite. Find-And-Bind Technologies For Mobile Devices. Technical report, Salutation Consortium.

[Pawlikowski, 1990] Pawlikowski, K. (1990). Steady–state simulation of queueing processes: A survey of problems and solutions. *ACM Computing Surveys*, *22*(2).

[Pei et al, 2000] Pei, G., Gerla M. and Hong X. (2000). LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility. *IEEE/ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA.

[Perkins et al, 1994] Perkins, C.E. and Bhagwat, P.(1994). Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Computer Communi- cations Review*, *24*(4),234–244.

[Radhamani et al, 2011] Radhamani, G.; Vanitha, K.; Sudhamathi, C.(2011). Cross Layer Issues in Service Discovery on Pervasive Computing. *Int. J. Comput. Appl.*, *17*, 1–4.

[Raisinghani et al, 2003] Raisinghani, V. T., Iyer, S.(2003). Cross-Layer Design Optimizations in Wireless Protocol Stacks. *Computer Communications*,*27*, 720–724.

[Ratsimor et al., 2002] Ratsimor, O., Chakraborty, D., Tolia, S., Khushraj, D., Gupta, G., Kunjithapatham, A., Joshi, A. and Finin, T. (2002). Allia: Police-based Alliance Formation for Agents in Ad hoc Environments. *2nd ACM Mobile Commerce Workshop held in conjunction with MobiCom 2002* , Georgia, USA.

[Raychoudhury et al, 2011] Raychoudhury, V., Cao, J.,Wu,W., Lai, Y., Chen, C. and Ma, J. (2011.) K-Directory community: Reliable service discovery in MANET. *Pervasive and Mobile Computing*, *7*(1), 140–158.

[Richard et al, 2000] Richard, G.G.(2000). Service advertisement and discovery: Enabling universal device cooperation. *IEEE Internet Computing*, *4*(5), 18–26.

[Saleem et al, 2013] Abdul Saleem, P.A. and Kumar, N.(2013). Cross Layer Design Approach in Wireless Mobile ADHOC Network Architecture. *Int. J. Adv. Res. Comput. Commun. Eng.*, *2*, 1450–1457.

[Schiele et al, 2004] Schiele, G., Becker, C. and Rothermel, K. (2004). Energy-efficient cluster-based service discovery for ubiquitous computing. *Proceedings of the 11th ACM SIGOPS European Workshop*. ACM, New York, NY, USA.

[Scioscia et al, 2014] Scioscia, F., Ruta, M., Loseto, G., Gramegna, F., Ieva, S., Pinto, A. and di Sciascio, E.(2014). A mobile matchmaker for the Ubiquitous Semantic Web. *International Journal on Semantic Web and Information Systems*, *10*, 77–100.

[Schacham et al, 1987] Schacham, N. and Westcott, J.(1987). Future directions in packet radio architectures and protocols. *Proceedings of the IEEE*, *75*(1), 83–99.

[SDP, 2001] SDP (2001). *Bluetooth Specification v1.1, Part E: Service Discovery Protocol (SDP).* https://www.bluetooth.org/en-us (Accessed on 08 November 2015).

[Siddarth et al, 2013] Siddarth, E.C. and Seetharaman, K.(2013). A Cluster Based QoS-Aware Service Discovery Architecture Using Swarm Intelligence. *Communications and Network*,*5*, 161–168.

[Simulador NS-2] Network Simulator ns (versión 2.34). http://www.isi.edu/nsnam/ns/ (Accessed on 08 November 2015).

[Sivakumar et al, 1999] Sivakumar, R., Sinha, P. and Bharghavan, V.(1999). CEDAR: Core Extraction Distributed Ad Hoc Routing Algorithm. *IEEE Journal on Selected Areas of Communications*, *17*, 1454–1465.

[Srivastava et al, 2005] Srivastava, V. and Motami, N.(2005). Cross-layer Design: A survey and the Road Ahead. *IEEE Communications Magazine*, *43*(12),112–119.

[Taylor et al, 2006] Taylor, I., Adamson, B., Downard, I. and Macker, J.(2006). AgentJ: Enabling Java Ns-2 simulations for large scale distributed multimedia applications. http://www.nrl.navy.mil/itd/ncs/node/267 (Accessed on 08 November 2015).

[Taylor et al, 2010] Taylor, I., Adamson, B., Herberg, U. and Macker, J. (2010). AgentJ Java Network Simulations in NS-2. An Installation and User Manual. http://downloads.pf.itd.nrl.navy.mil/archive/agentj/agentj.pdf (Accessed on 08 November 2015).

[Tian et al, 2005] Tian, Y., Xu, K. and Ansari, N. (2005). TCP in Wireless Environments: Problems and Solutions. *IEEE Commun. Mag.*, *43*, S27–S32.

[Tyagi et al, 2010] Tyagi, S. S. and Chauhan, R. K. (2010). Performance Analysis of Proactive and Reactive Routing Protocols for Ad hoc Networks. *I*nternational Journal of Computer Applications, *1* (14), 0975–8887.

[UM-OLSR] Implemenatción de OLSR para NS-2. https://cloudns2.wordpress.com/um-olsr-patch/ (Accessed on 08 November 2015).

[UpnP, 2015] UpnP Forum (2015). UpnP Device Architecture 2.0. http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf (Accessed on 08 November 2015).

[Vara et al, 2006] Vara, M., Cabero, J. y Jodra, J. (2006). Service Discovery in Proactive Mobile Ad Hoc Networks. *Challenges in Ad Hoc Networking* (pp. 295–299). Boston, MA, USA: Springer.

[Vara et al, 2015] Vara, M. y Campo, C. (2015). Cross Layer Service Discovery Mechanism for OLSRv2 Mobile Ad Hoc Networks. *Sensors*.

[Veizades et al, 1997] Veizades, J., Guttman, E., Perkins, C. and Kaplan, S. (1997). RFC 2165: Service location protocol. Updated by RFCs 2608, 2609. https://tools.ietf.org/html/rfc2165 (Accessed on 08 November 2015).

[Ververidis et al, 2008] Ververidis, C. N. and Polyzos, G. C. (2008). Service discovery for mobile ad hoc networks: A survey of issues and techniques. *IEEE Communications Surveys & Tutorials*, *10*(3), 30–45.

[Weiser, 1991] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*.

[Willians, 2002] Willians, A. (2002). Requirements for Automatic Configuration of IP Hosts. Internet-Draft (work in progress).

[Zhong et al, 2012] Zhong, J., Geng, S., Weng, L. and Li, X. (2012). A Cross-layer Service Discovery Protocol for MANET. *J. Comput. Inf. Syst.* , *8*, 5085–5092.

[Zuo et al, 2014] Zuo, J., Dong, C., Nguyen, H.V., Ng, S.X., Yang, L.-L. and Hanzo, L. (2014). Cross-Layer Aided Energy-Efficient Opportunistic Routing in Ad Hoc Networks. *IEEE Trans. Commun.*, *62*, 522–535.