UNIVERSIDAD DE MURCIA

Departamento de Ingeniería de la Información y las Comunicaciones

Tesis Doctoral

Arquitectura de descubrimiento de servicios en MANET basada en dispositivos de capacidades superiores liderando clusters

Autor: **Miguel Antonio Wister Ovando**
Ingeniero en Informática

Director: **Dr. Juan Antonio Botía Blaya**

2008

Autorizo la defensa de la tesis doctoral *Arquitectura de descubrimiento de servicios en MANET basada en dispositivos de capacidades superiores liderando clusters* cuyo autor es D. Miguel Antonio Wister Ovando

El director de la tesis

Juan Antonio Botía Blaya
Murcia, 28 de mayo de 2008

TESIS DOCTORAL:	Arquitectura de descubrimiento de servicios en MANET basada en dispositivos de capacidades superiores liderando clusters
AUTOR:	D. Miguel Antonio Wister Ovando
DIRECTOR:	Dr. Juan Antonio Botía Blaya

El tribunal nombrado para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

PRESIDENTE:

VOCALES:

SECRETARIO:

acuerda otorgarle la calificación de:

Murcia, 28 de mayo de 2008

El Secretario del Tribunal

D. Luis Daniel Hernández Molinero, Profesor Titular de Universidad del área de Ingeniería Telemática y Director del Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, INFORMA:

Que la Tesis Doctoral titulada "Arquitectura de descubrimiento de servicios en MANET basada en dispositivos de capacidades superiores liderando clusters", ha sido realizada por D. Miguel Antonio Wister Ovando, bajo la inmediata dirección y supervisión de D. Juan A. Botía Blaya, y que el Departamento ha dado su conformidad para que sea presentada ante la Comisión de Doctorado.

En Murcia, a 28 de mayo de 2008

D. Luis Daniel Hernández Molinero

What you know, you know, what you don't know, you don't know. This is knowledge.
Confucius (551 BCE – 478 BCE). Chinese philosopher.

El ignorante afirma, el sabio duda y reflexiona.
Aristóteles (384 AC-322 AC) Filósofo griego.

Let me be ignorant, and in nothing good, but graciously to know I am no better.
William Shakespeare (1564-1616). English writer.

Agradecimientos

Comenzar y terminar esta tesis ha sido un gran reto en mi vida. Al principio no tenía idea del momento exacto en que podría terminar, veía a muchos otros comenzar y parecería que el tiempo se debería detener a esperar que se concluyera lo que se había comenzado. Sin embargo, conforme avanzaba veía más cerca el momento de concluir, cada día de estudio representaba un día menos de esa incertidumbre, lo que mantenía viva la esperanza por alcanzar la culminación de la tesis. La tenacidad y el ahínco del día a día me hacían vislumbrar el final.

Escribir una tesis es como colocar un ladrillo sobre un muro que otros han venido construyendo y que seguramente vendrá otro después de mi a colocar otro ladrillo ....., pero para colocar ese ladrillo es necesario prepararse, trabajar y estudiar asiduamente hasta el cansancio. Esta tesis en mis manos es una prueba de que he hecho todo lo que tenía que hacer, y aun más, estar consciente de que pudo haberse hecho mejor o quedado cosas pendientes de hacer.

Para mí esta tesis representa un desafío a la inteligencia, un desafío a que vengan otros y mejoren lo que he hecho. La realización de esta tesis tiene mucho significado para mí, expresa un triunfo que se manifiesta con un grito proclamando éxito, una voz que anuncia – ¡Lo he logrado!.

Pero para concluir con éxito esta tesis, debo agradecer a algunas personas e instituciones que han colaborado para hacer posible este logro.

En primer lugar deseo dar las gracias a mi director, Juan Botía, por su interés en guiarme, su apoyo prestado, sus brillantes contribuciones y sus invaluables consejos.

En segundo lugar quiero agradecer a la Universidad Juárez Autónoma de Tabasco, mi alma máter, por todo lo que me ha dado y por la beca institucional que hizo posible mi formación.

Quiero agradecer a la Universidad de Murcia, por ser la institución que me acoge y contribuye en mi desarrollo y superación académica.

Quiero agradecer al Programa de Mejoramiento al Profesorado (PROMEP) por haberme otorgado la beca para realizar mis estudios de doctorado.

Quiero agradecer a Dante Arias, por sus aportaciones en varios aspectos de la propuesta.

Esta tesis va dedicada especialmente a mis seres queridos, a Julieta, Jessica y Pamela quienes eran toda mi familia y a Walter que nació durante este tiempo.

Una dedicación muy especial para mis padres que gracias a Dios podrán leer esta dedicatoria e igualmente dedicada para mis hermanos.

Deseo dedicar también este trabajo a mis amigos, a mis compañeros de trabajo y a mi universidad.

Quiero dedicar esta tesis a todos mis compatriotas que estuvieron cerca de mi familia, a todos los que vinimos a Murcia y dejamos nuestro país para venir a estudiar. A Alonso, Andrés, Alan, Mario, Gerardo, Jesús Armando, Karla, Clarisa, José Luis, Tomás, a todos ellos y a sus respectivas familias.

Finalmente, quiero también dedicar esta tesis a todas aquellas muy buenas personas que conocí en Murcia, a Cielo, Isabel, Santiago, Mónica, Raúl y muchos más.

Abstract

The pervasive and ubiquitous computing terms are used to describe a smart space populated by hundreds of intelligent devices that are embedded in their surroundings. Characteristically, ubiquitous computing devices must blend into the background, unobtrusively collaborating to provide value-added services for users. Services are thus essential to the success of this technology and, as a result, both service discovery and service management will play a vital role in generating the revenue stream that is a prerequisite for sustainable ubiquitous deployment. On the one hand, the services provided should be evident by their richness and variety and on the other, the complexity inherent in the environment must be hidden from users [69].

An important aspect in the service discovery field is the dynamic nature of MANET, Mobile ad hoc networks are characterized by their highly dynamic, multi-hop, and infrastructure-less nature, these networks have as main characteristics the frequently changes of topology and the intermittent connectivity of their elements, hence they are forced to assiduously reconfigure their schemes in order to discover available service providers to satisfy the client requests.

In general, a MANET is formed by heterogeneous small devices with limited capabilities (e.g. battery power, memory, processing power, node coverage, etc). Nevertheless, some times are connected high capability devices (HCD) which have suitable capabilities to assume the role of intermediary.

In this thesis is proposed a service discovery architecture in MANET based on HCD who lead clusters. Our proposal combine an approach based on a clustering mechanism with another one based in Cross-layer design, this combination

involves the routing process messages for discovering services. The solution employs existing HCD for leading and supporting the clustering management, while limited devices do not waste their reduced resources.

This proposal attempts to minimize the global communication and computation consumption of the network, this consumption is only assigned into a local ambit, that is, within a cluster where is generated the service requests, in addition to the assignation of workload to the HCD. The flooding of requests of users do not exceed the cluster limits. Therefore, it is achieved to significantly reduce the global consumption of resources of the MANET.

Resumen

Los términos computación pervasiva y computación ubicua se usan para describir un espacio inteligente poblado de cientos de dispositivos inteligentes embebidos en su alrededor. Característicamente, los dispositivos de computación ubicua se mezclan entre el entorno. Los servicios son esenciales para el éxito de esta tecnología y como resultado, el descubrimiento de servicios y la gestión de los servicios juegan un papel importante. Por un lado, los servicios provistos deben ser evidentes por su riqueza y variedad, y por el otro, la complejidad inherente en el entorno debe ser oculta a los usuarios [69].

Uno de los aspectos que llaman la atención del tema de descubrimiento de servicios es la naturaleza dinámica de las MANET, que tiene como característica los frecuentes cambios de topología y la conectividad de sus elementos a veces intermitente, lo cual obliga a reconfigurar asiduamente los medios para descubrir proveedores de servicios disponibles que satisfagan las peticiones de los clientes.

Por lo general una MANET está formada por dispositivos heterogéneos con capacidades limitadas, equipos pobres en recursos como batería, memoria, cómputo, etc. No obstante, en ocasiones se conectan dispositivos que poseen capacidades superiores (HCD) que pueden jugar el papel de intermediario.

En esta tesis se propone una arquitectura de descubrimiento de servicios en MANET basada en HCD que lideran clusters; el diseño combina un enfoque basado en agrupamiento con otro basado en diseño Cross-layer. La combinación involucra los mensajes del proceso de encaminamiento para descubrir servicios. La propuesta emplea los HCD existentes en la red para liderar y dar soporte a

la gestión de grupos, mientras que los dispositivos de capacidades limitadas no desgastan sus reducidos recursos.

Esta propuesta intenta minimizar el consumo de recursos globales de comunicación y de computación de toda la red, dejando únicamente este consumo a un ámbito local, es decir, dentro del grupo en el que se generan las peticiones de servicios; así como la asignación de mayor carga de trabajo a los HCD. Las difusiones lanzadas por un cliente no rebasan los confines de su cluster, logrando reducir significativamente el consumo global de recursos de la MANET.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La computación ubicua conocida por algunos otros como computación pervasiva[1] es un término acuñado por Mark Weiser [156] y es descrita como la posibilidad de conectar a Internet todo lo que nos rodea, con el fin de darnos información acerca de cualquier cosa, en todo momento y en todo lugar. En otras palabras, la computación ubicua es la omnipresencia de pequeños computadores interconectados de forma inalámbrica y embebidos invisiblemente en los objetos a nuestro alrededor. Lo anterior es posible gracias al uso de sensores; estos dispositivos embebidos pueden detectar el entorno que les rodea y poseer capacidades de procesamiento de información y/o de comunicación.

Se estima que en el futuro los ambientes de vida de los humanos estará sustentado con la presencia de diversidad de recursos de información que serán proporcionados por la interconexión de redes de comunicación. Gracias a la movilidad los dispositivos como teléfonos móviles, computadores vestibles y handhelds aumentan sus capacidades de acceso y de procesamiento de información. Por otra parte, aplicaciones cotidianas tradicionales como hornos, cámaras digitales, lavadoras, refrigeradores, aspiradoras y termostatos con capacidades integradas de cómputo y comunicación, aumentan y propician las condiciones para un completo entorno de computación ubicua. Con esto en mente, se deben desarrollar nuevas tecnologías dentro de nuevos paradigmas de computación ubicua que in-

[1]El término computación ubicua tiene una orientación más académica, más centrada en humanos, en cambio, el término computación pervasiva es el nombre acuñado por la industria.

cluyan nuevas arquitecturas, estándares, dispositivos, servicios, herramientas y protocolos [145].

La computación distribuida ha evolucionado hasta llegar a la computación móvil y a la computación ubicua. La computación móvil es una de las tecnologías más importantes que soportan la computación ubicua. En estos últimos años los avances tanto en hardware como en software han propiciado que los nodos móviles y las redes inalámbricas sean más comunes y variadas. Y con ello han surgido nuevas aplicaciones y soluciones pero también nuevos retos y problemas.

La historia de las redes inalámbricas comienza allá por los años 70 con proyectos como ALOHA, que tenía como objetivo conectar las instituciones educativas de las islas de Hawai. Más adelante, en 1973 DARPA comenzó a trabajar en la Packet Radio Networks (PRNETs), como el lector puede suponer esta red tenía fines militares. PRNETs pretendía que las unidades militares pudieran comunicarse a través de radio, con total movilidad y de modo cooperativo.

Desde ese tiempo han sucedido muchos avances y una de las redes inalámbricas más en voga actualmente son las redes sin infraestructura, comúnmente conocidas como redes móviles Ad Hoc. El interés en las redes móviles Ad Hoc ha pasado a los ámbitos académicos y científicos y ha dejado de ser exclusivamente militar y esto es gracias al reciente éxito de las comunicaciones inalámbricas por un lado y a la miniaturización de los dispositivos por el otro. Y gracias al reciente incremento en el uso de dispositivos portátiles y móviles y a la sofisticación de los mismos, el IETF[2] en la mitad de la década de los 90s formó un grupo de trabajo especialmente centrado en el estudio de las redes Ad Hoc, con el objetivo de estandarizar los aspectos más importantes y con el objetivo de que este tipo de redes puedan ser usadas en aplicaciones comerciales.

Comparando las redes móviles Ad Hoc con toda la tecnología que ha existido, éstas presentan configuraciones dinámicas muy peculiares, observan cambios de topología muy frecuentes debido a la alta movilidad de sus nodos. Las

[2]El **IETF** (Internet Engineering Task Force, es el Grupo de Trabajo en Ingeniería de Internet) es una organización internacional abierta de normalización, que tiene como objetivos contribuir al mejoramiento de la ingeniería de Internet.

características de este tipo de red no permiten el empleo de protocolos de encaminamiento, seguridad y descubrimiento de servicios diseñados para las redes convencionales (cableadas, fijas, con administración central) y da origen al desarrollo de soluciones eficientes que superen los problemas inherentes a los propios dispositivos (reducida memoria, batería limitada) y a la topología dinámica.

El concepto de redes Ad Hoc no es novedoso, décadas atrás se pensaba en posibles aplicaciones en el terreno bélico. Principalmente usadas en redes tácticas relacionadas a aplicaciones para mejorar las comunicaciones y supervivencia en campos de batalla [32], muy útil para la comunicación en los campos de combate, situaciones de emergencia y en zonas de desastres. Con las tecnologías actuales que tenemos a día de hoy, es una realidad que con Bluetooth y IEEE 802.11 se logre todo lo que hace tres décadas era difícil llevar a cabo.

1.1. Antecedentes de las redes móviles Ad Hoc

Previamente a la aparición de las MANET[3] (Mobile Ad-Hoc Networks) ya existían otras clases de redes como: LAN, WAN, MAN y WLAN. En este sentido en [96] clasifican a las redes de acuerdo a las siguientes características: *en base a su tamaño*: que van desde redes pequeñas a muy grandes; *en base a su dinámica*: redes que pueden ser estáticas, como redes de área local cableadas o inalámbricas, y en redes móviles como las MANET; y *en base al tipo de dispositivo*: desde dispositivos ricos en recursos, como laptops, hasta dispositivos limitados como las handhelds. De acuerdo a las características anteriores, las redes se pueden agrupar en tres clases: *LAN* - red integrada tanto por dispositivos ricos en capacidades y dispositivos de capacidades reducidas. Aquí la movilidad no es problema dado que los dispositivos no entran y salen de la red muy frecuentemente. *WAN*: red con muchos dispositivos que cubre una amplia

[3]Una red inalámbrica Ad Hoc es una colección de nodos móviles inalámbricos que se autoconfiguran para formar una red sin la ayuda de ninguna infraestructura inherente previamente establecida. Los nodos móviles por si mismos manejan el control necesario y las tareas de enlaces, a través del uso de algoritmos de control distribuidos. Los nodos intermedios mediante conexiones múlti-salto envían paquetes hacia algún destino final [61].

área geográfica. *WLAN*: una red inalámbrica flexible muy utilizada como alternativa a las redes LAN cableadas o como extensión de éstas. Y *MANET*: red sin infraestructura, con topología altamente dinámica en la que sus nodos entran y salen de la red de forma impredecible.

La tecnología MANET usa dispositivos computacionales portátiles como laptops, handhelds y teléfonos móviles, junto con tecnologías de comunicación móvil que permite a los usuarios acceder a Internet y a sus datos, prácticamente desde cualquier sitio. Eventualmente podría ser un grupo de dispositivos inalámbricos con altas capacidades de cómputo que se conecten a redes cableadas o usar redes con infraestructura o algunas veces a redes híbridas. Las MANET tienen características específicas que únicamente una adecuada tecnología puede soportarla.

A diferencia de las redes convencionales basadas en infraestructura: LAN, MAN, WAN, etc, las MANET no poseen infraestructura de red fija subyacente, ni administración centralizada, desaparece el papel del nodo central para dar pie a un entorno en el que los recursos trabajan de modo P2P (Peer-to-Peer[4]). Como ya se ha mencionado, las MANET no cuentan con infraestructura pre-existente, la red se construye entre los mismos nodos. La topología de las MANET cambia dinámicamente de manera arbitraria, súbita e impredeciblemente, la conectividad entre los nodos podría ser variable en el tiempo, debido a que los nodos continuamente entran y salen de la red y también debido a fluctuaciones en la potencia de la señal; por lo tanto, algunos enlaces de comunicación podrían no estar disponibles [40] [96] [88]. Adicionalmente a las características más relevantes ya conocidas (dispositivos con capacidad limitada, cambios de rutas, limitaciones de los enlaces inalámbricos, ausencia de infraestructura, topología variable, descentralización y nodos móviles), se añaden otras peculiaridades como heterogeneidad, autonomía, auto-configuración y alta distribución [147] [89].

[4]Tipo de red en el que cada terminal tiene capacidades y responsabilidades equivalentes. Difiere de la arquitectura cliente/servidor ya que algunos computadores están dedicados a servir a otros. Las redes P2P generalmente son más simples, pero usualmente no ofrecen el mismo rendimiento con el trabajo pesado. En el esquema P2P todos los nodos se tratan de igual a igual

Una MANET está compuesta por nodos, que albergan servicios, y esos nodos pueden jugar tanto el rol de cliente como el de servidor. Un servicio dentro de una red puede ser cualquier entidad de software o hardware que un usuario pueda estar interesado en utilizar [142]. Los servicios son aplicaciones que ofrecen los proveedores para que puedan interactuar con los usuarios. Un protocolo de descubrimiento de servicios permite a los proveedores de servicios anunciar sus servicios y también permite a los usuarios descubrir automáticamente dichos servicios. [149].

Dadas las características de las MANET, es imposible utilizar los protocolos de descubrimiento de servicios creados para las redes convencionales (cableadas, fijas), se requiere desarrollar protocolos de descubrimiento de servicios (**SDP**, Service Discovery Protocol) eficientes que resuelvan los problemas inherentes a entornos Ad Hoc como topología dinámica, recursos de ancho de banda, batería limitada y escasa seguridad.

Los SDP existentes diseñados para redes cableadas en ocasiones no se adaptan a los entornos altamente dinámicos de las MANET, debido entre otras cosas a que los protocolos disponibles emplean técnicas de difusión que inundan de mensajes el reducido ancho de banda de la red. Por consiguiente, es necesario desarrollar nuevos algoritmos y protocolos que superen las limitaciones mencionadas y permitan establecer redes autónomas y descentralizadas. Estos protocolos deben ser flexibles y adaptativos, con capacidad de prever el comportamiento de la red partiendo de información previa, como tasa de errores, nivel de congestión, cambios de rutas utilizadas, etc. Por otra parte, los servicios de la red han de ser localizados y utilizados automáticamente sin la intervención de los usuarios en la configuración del sistema.

Las MANET se han convertido en la solución inalámbrica ideal para conectar tanto terminales móviles como terminales fijas, sin necesidad de utilizar un punto de acceso como sucede con las redes inalámbricas tradicionales, puesto que ese punto de acceso centraliza el control de la red, se convierte en un elemento crítico para la interconexión de los nodos y la ausencia de ese punto de acceso

ocasiona un fallo global en la red. Así mismo es posible formar redes híbridas mediante la combinación con redes locales inalámbricas que se conectan a Internet utilizando puntos de acceso inalámbricos. En este caso las MANET cuando son combinadas con redes inalámbricas tradicionales o con redes cableadas con infraestructura fija encuentran la cooperación de nodos estáticos (sensores, radio tags, smart labels, motes, routers, etc.) que les permite ser totalmente localizables porque continúan formando parte de elementos móviles (PDA, laptops, teléfonos móviles, etc.) [33], de esta forma asumen el papel de nodo con propiedades sensoriales que le permiten localizarse así mismo y localizar a otros nodos.

Respecto al tipo de entorno, existen redes que funcionan en ambientes exteriores (redes públicas) y otras redes que debido a su alcance sólo funcionan en espacios interiores (indoor). Existen MANET que basan su funcionamiento en esquemas como *context-awareness*[5] para facilitar la sensibilidad al contexto [41] y *location-awareness*[6] para ser sensible a la localización [122] [76] de dispositivos y servicios.

1.2. Arquitecturas de descubrimiento de servicios disponibles

Actualmente existen distintos SDP por lo general enfocados a redes con infraestructura estable y poco adaptados a MANET. Las soluciones para MANET están limitadas a redes muy pequeñas en número de nodos y no consideran aspectos como la elección de los nodos servidores más cercanos al cliente, ni la elección de nodos con capacidades superiores (High Capability Devices) que en adelante denominaremos **HCD**.

[5]Sensible al contexto .- que el sistema esté capacitado para emplear cualquier información a fin de que sea posible caracterizar la situación de la entidad, sea ésta una persona, un lugar o un objeto.

[6]Sensible a localización .- que el sistema permite ser localizado en cualquier parte que se ubique y se clasifican en *activos*, que tienen redes sensibles y rastrean la localización de usuarios y en *pasivos* que el sistema no rastrea a los usuarios, sino que posee dispositivos sensibles distribuidos desde donde los usuarios leen su propia información de localización.

De los SDP que podemos encontrar en la literatura, en términos generales se puede decir que existen dos diferentes modelos para relacionar clientes y proveedores de servicios: modelo basado en *interacción directa* y modelo basado en *interacción indirecta*.

- En la *interacción directa*, cuando un nodo cliente quiere descubrir un servicio, difunde mensajes de petición de servicios por toda la red y espera la respuesta de los nodos proveedores de ese servicio. La relación directa se establece también de modo pasivo, es decir, los nodos proveedores anuncian periódicamente por difusión los servicios a toda la red. Ambos modos siguen un esquema P2P, porque los nodos intercambian información directamente entre ellos y no existe la noción de clientes y servidores, sino de igual a igual. Para descubrir un servicio, simplemente interviene el proveedor y el cliente. Ejemplos de arquitecturas con este tipo de relación son: Allia [131], AODV-SD [95], DEAPspace [106], GSD [24], Konark [70], NOM [43], PDP [19], RUBI [69], SPIZ [109], SSD [138] y UPnP [151]. Este modelo de interacción directa es idóneo para MANET, puesto que no se necesitan nodos especiales que actúen como mediadores y además no existe ningún nodo que sea crítico para el funcionamiento de la red; si algún nodo falla, sólo se pierden los servicios que éste ofrece. Sin embargo, este modelo tiene el inconveniente de usar difusión. En el caso de una red Ad Hoc este mecanismo inunda de mensajes la red, con el consiguiente gasto; por lo que es necesario buscar otras soluciones.

- En la *interacción indirecta* se introduce la figura de un mediador o intermediario. En este modelo no es necesario utilizar difusión para descubrir o anunciar servicios. Sin embargo, también existen ventajas e inconvenientes como en el caso de la interacción directa. En el modelo de interacción indirecta se requiere que uno o varios nodos se comporten como servidores. Obviamente, por razones de escalabilidad no se usa un único servidor, en este caso se opta por descentralizar el sistema y se reparte la carga de trabajo con el objetivo de obtener mayor robustez frente a caídas de nodos

que operen como intermediarios; ejemplos de arquitecturas con interacción indirecta son: Bonjour [42], Carmen [99], INS/Twine [6], Jini [77], MARE [144], One ring to rule all [21], Salutation [35], SANDMAN [141], SLP [66], SLPManet [139], Splendor [161], SSDS [39], Superstring [135], UDDI [111], VIA [22], X-Hoc [121] y ZBS [75].

En el primer tipo de interacción están presentes los siguientes problemas: la difusión satura la red de mensajes de petición de servicios lo cual origina altos costos de comunicación, memoria, cómputo y energía. Las estrategias encaminadas a resolver estos inconvenientes se enfocan principalmente en minimizar la cantidad de mensajes enviados para anunciar y descubrir servicios y en proponer mecanismos para generar la menor cantidad de paquetes en el proceso de difusión.

En la interacción indirecta los intermediarios se convierten en nodos críticos para el resto de la red y es difícil llevar a cabo la comunicación entre los nodos debido al número de saltos para alcanzar al nodo destino, además, que la conexión múlti-hop podría llegar a ser crítica por el alcance de radio de los dispositivos. Actualmente los trabajos de investigación proponen replicar los nodos proveedores de servicios [29] [107] y distribuir la carga de trabajo [98] [146].

En la tabla 2.2 del capítulo 2 se presenta un resumen de estos dos modelos de interacción.

En las MANET en cierto momento pudieran estar conectados un gran número de dispositivos heterogéneos, tanto dispositivos de capacidades reducidas y recursos limitados como dispositivos con capacidades superiores y ricos en recursos. Nuestra propuesta de solución supone este hecho y sugiere aprovechar los dispositivos de altas prestaciones para liderar clusters de nodos y proveer información de servicios. La solución que proponemos es híbrida, en cuanto a que interactúa de forma directa pero también lo hace de forma indirecta, ya que difunde mensajes de petición de servicios dentro del cluster, mediante el uso de intermediarios que centralizan las peticiones de servicios. Por esta razón

creemos que es posible disminuir el tráfico en la red minimizando la difusión de mensajes.

1.3. Motivación

Existen líneas de investigación bien definidas en el campo de las MANET como por ejemplo la del *routing* (conocido como: enrutado o enrutamiento o encaminamiento) que se encarga de descubrir rutas válidas entre los nodos. Otra línea de investigación es el estudio de la *Calidad de servicios (QoS)* percibida como una alta tasa de éxitos para acceder a datos multimedia, un aceptable rendimiento para recuperar datos y también por la selección de un buen servicio con alta capacidad de servicio. Una línea de investigación más es la de *seguridad*, por las vulnerabilidades comunes de las conexiones inalámbricas ya que las MANET tienen problemas particulares de seguridad en el momento de enviar datos, por lo tanto es necesario la existencia de diferentes esquemas de gestión y autenticación de claves para proteger la privacidad de las transmisiones. En esta investigación nos interesamos en aspectos relacionados con la topología dinámica de las MANET, específicamente abordamos el tema de *protocolos de descubrimiento de servicios* (SDP) de no menos importancia que la seguridad, encaminamiento y QoS.

Lo que principalmente motiva de este tema de descubrimiento de servicios es la naturaleza dinámica de las MANET, ya que en ellas ocurren frecuentes cambios de topología y la conectividad es muchas veces intermitente, lo cual obliga a reconfigurar constantemente los medios para buscar y encontrar los proveedores de servicios disponibles que satisfagan las peticiones de los clientes.

Existen diversos protocolos, arquitecturas o plataformas que han emergido recientemente como: UPnP, DNS-SD, Jini, Salutation, OSGi, JXTA, UDDI, SLP, Bluetooth SDP, Konark, GSD, Allia, DEAPspace, PDP, etc., encargadas de realizar el descubrimiento y entrega de servicios en redes tanto cableadas como inalámbricas. Sin embargo, no todos los protocolos emplean los algoritmos más convenientes para manejar los complejos mecanismos de infraestructuras

de redes móviles Ad hoc. Mayormente estos mecanismos de descubrimiento de servicios para redes cableadas suponen poseer algún tipo de directorio o servidor de repositorio, generalmente operando bajo un enfoque centralizado [153] [131] [86].

Hemos revisado las principales características de las soluciones actuales mencionadas anteriormente. Muchas de ellas tienen contribuciones y rendimientos interesantes; pero haciendo un análisis objetivo podemos concluir que algunos SDP no logran resolver las complejidades de los entornos móviles. Por ejemplo, algunas soluciones han sido diseñadas para redes fijas sin tener en cuenta que los nodos concurrentemente se conectan o desconectan de la red. En secciones posteriores se enumeran todas estas soluciones a las que nos referimos.

Sólo unos pocos SDP cuentan con un repositorio de servicios local, distribuido, replicado o fragmentado por áreas, grupos o tipos de servicios. Casi todos los protocolos pasan por alto las limitaciones de los dispositivos, usan un repositorio centralizado residente en un PC o un laptop, donde guardan y controlan los servicios disponibles en la red.

1.4. Problema

Podemos decir en términos generales que los protocolos que se basan en el uso de directorios emplean el siguiente mecanismo: Un servicio se anuncia por difusión en la red, se registra en un directorio de servicios donde se almacenan todos los servicios existentes en la red. Los servicios se dan de baja en el registro en cualquier instante, bien de forma intencional o bien de forma no intencional. Normalmente las comunicaciones están dadas sobre redes IP. Las arquitecturas mencionadas son particularmente centralizadas o semi-centralizadas y raras veces se plantean sobre esquemas descentralizados, orientados a registros y la infraestructura de red es algunas veces cableada e incluso estable [24].

El descubrimiento de servicios en escenarios móviles requiere un esquema de diseño descentralizado y que cada nodo sea independiente de los otros nodos para anunciar o registrar sus servicios. Por eso, un dispositivo debe ser autónomo para

que permita que al mismo tiempo otros dispositivos sean capaces de descubrirlo. Pensando en un enfoque descentralizado existen dos soluciones posibles para tratar esto:

- Primera solución.- Para descubrir servicios en una MANET se envía a través de broadcast un *request* de descubrimiento de servicios. Si un nodo contiene el servicio, responde con un *reply* (Modo activo - Pull Method), esta solución es una forma de interacción directa. Normalmente esta solución promete descubrir un servicio, si el servicio se encuentra en la red. Algunas veces esta solución puede generar muchos mensajes, ocasionando una tormenta de broadcast[7] e inundando de mensajes la red [24].

- Segunda solución.- Cuando los servicios se anuncian a sí mismos en la red. Cada nodo interesado en descubrir servicios almacena en caché los anuncios (Modo pasivo - Push Method), esta solución también es una forma de interacción directa. Cuando los anuncios concuerdan con las peticiones de servicios, entonces se produce un resultado. Vemos que la memoria caché aumenta de tamaño con la cantidad de servicios; suponemos que los nodos tienen memoria y energía limitada y son incapaces de almacenar todos los anuncios [25]. Pensamos que esto es ineficiente en términos de mal uso del ancho de banda, porque la mayoría de los nodos son saturados de anuncios cada vez que un nodo anuncia sus servicios.

En redes Ad hoc y en entornos de computación ubicua existe gran movilidad, además los dispositivos tienen capacidades limitadas de almacenamiento, de tal manera que es muy costoso que uno de ellos pueda asumir el rol de servidor. La viabilidad de los escenarios anteriores requiere no sólo la formación de redes Ad hoc, sino también un mecanismo de descubrimiento y entrega idóneo a las necesidades de estos escenarios.

[7]Una tormenta de broadcast es un estado en el que un mensaje que ha sido difundido a través de la red produce muchas respuestas y cada respuesta produce aún más respuestas ocasionando un efecto de bola de nieve. Una tormenta de broadcast grave puede bloquear todo el tráfico de la red.

Es ineficiente el uso de un servidor central que se encargue de manejar redes altamente dinámicas, porque en entornos ubicuos un nodo se puede conectar y desconectar en cualquier momento. En MANET es válido que los nodos puedan actuar como servidor o como cliente al mismo tiempo e incluso los nodos que no son estables en estos escenarios.

1.5. Solución propuesta

Nuestra propuesta mezcla algunas ideas de ambos modelos de interacción y los combina con una solución de descubrimiento de servicios basada en diseño Cross-layer[8] que emplea el protocolo de encaminamiento AODV[9] (Ad hoc On-Demand Distance Vector)[50] ideal para descubrir rutas en redes Ad Hoc. Empleamos el diseño Cross-layer puesto que el proceso de encaminamiento y el proceso de descubrimiento de servicios se realizan al mismo tiempo y se usa la misma cantidad de mensajes. Con ello se optimiza el número de mensajes necesarios en el proceso.

El protocolo de descubrimiento de servicios propuesto tiene el nombre de **LIFT** (**LI**mited **F**looding of requests within a clus**T**er) ya que difunde mensajes únicamente dentro del cluster.

Con esta propuesta buscamos responder a las siguientes preguntas y de este modo justificar la decisión de agrupar los nodos de la red y establecer la política de elegir a los nodos HCD como líderes de los clusters.

- ¿Minimiza nuestro enfoque el consumo de recursos (energía, ancho de banda, memoria) durante el proceso de descubrimiento de servicios?.

[8] Cross-layer es una técnica que consiste en un intercambio de información entre las capas de un modelo de arquitectura por niveles como en OSI o TCP/IP.

[9] AODV es un protocolo de descubrimiento de rutas que trabaja bajo demanda y es basado en encaminamiento por vector de distancia, además de que evita la formación de bucles en virtud de que utiliza números de secuencia por cada destino. AODV únicamente mantiene rutas hacia los nodos con los que tiene comunicaciones activas, en consecuencia introduce menor sobrecarga en la red.

- ¿Cuántos servicios pueden ser descubiertos bajo este enfoque (tasa de éxito)?.
- ¿Es válido este enfoque para MANET de gran escala (100, 150, 200, 300, 400 nodos)?.
- ¿En cuáles escenarios de MANET existen posibilidades de éxito (conferencias, rescate, etc.) y bajo qué restricciones?.
- ¿Cuál es el costo del mantenimiento del clustering?.

1.6. Objetivos

Este trabajo de investigación tiene como propósito proponer soluciones a algunos de los problemas aún por resolver dentro del campo de descubrimiento de servicios en MANET. Concretamente planteamos los siguientes objetivos que esperamos alcanzar para la conclusión de la presente tesis doctoral.

- Formar clusters de nodos con el fin de optimizar el consumo de recursos y disminuir el tráfico en la red minimizando la difusión de mensajes, evitando que los mensajes de petición se propaguen más allá del cluster.
- Diferenciar entre dispositivos de capacidades superiores (HCD) y dispositivos de capacidades limitadas (LCD).
- Establecer el líder del cluster mediante un parámetro que identifique las capacidades superiores de un nodo y las anuncie al vecindario.
- Descubrir servicios a través de los HCD (quienes dan el soporte al interior del cluster).
- Simular mediante una herramienta apropiada el escenario con los nodos móviles altamente dinámicos, heterogéneos, aplicando modelos de movilidad reales (conferencias, grupos, aulas) con patrones de movimientos aproximados al desplazamiento humano.

1.7. Contribuciones

La contribución de esta investigación se centra directamente sobre el problema de consumo de recursos en toda la red. Es decir, esta propuesta de protocolo de descubrimiento de servicios intenta conseguir que los recursos globales de comunicación y de computación de toda la red, se reduzcan únicamente a un consumo local dentro del grupo en el que se generan las peticiones de servicios; así como la asignación de mayor carga de trabajo a los dispositivos que posean capacidades superiores. Las difusiones lanzadas por un nodo peticionario de servicios no rebasan los confines de su cluster. De esta manera se reduce significativamente el consumo global de recursos de la red.

Resultados parciales de esta tesis han sido presentados a la comunidad científica de este campo a través de publicaciones en congresos nacionales e internacionales. En la sección 1.8 se enumera una lista con todas ellas.

1.8. Publicaciones

El trabajo realizado para esta tesis ha generado hasta el momento las siguientes publicaciones. Cada una de ellas han sido revisadas por expertos en la materia:

- Miguel A. Wister, Juan A. Botía, Antonio F. Gómez-Skarmeta *Hacia una arquitectura de agentes descentralizada para la gestión de sistemas asistentes en situaciones de emergencia.* ISBN: 84-9732-454-4 THOMSON. pp. 71 – 77. II Taller en Desarrollo de Sistemas Multiagente. (DESMA 2005). I Congreso Español de Informática. CEDI 2005. 13 de septiembre 2005. Granada.

- Miguel A. Wister, Juan A. Botía, Antonio F. Gómez-Skarmeta *Descubrimiento de servicios en redes móviles Ad-hoc. Una perspectiva global.* ISBN: 84-8138-703-7 UAH. pp. 351 – 363. II Congreso IberoAmericano sobre

Computación Ubicua. (CICU 2006). 7, 8 y 9 de Junio de 2006. Universidad de Alcalá.

- Miguel A. Wister, Juan A. Botía, Antonio F. Gómez-Skarmeta *Evolution and Challenges in Service Discovery Architectures: from fixed networks to Mobile Ad hoc Networks.* 2nd International Symposium on Ubiquitous Computing and Ambient Intelligence – 2007. Zaragoza, Spain. 2007.

- Miguel A. Wister, Juan A. Botía, Antonio F. Gómez-Skarmeta *Arquitectura de descubrimiento de servicios en MANETs basada en dispositivos de capacidades superiores liderando clusters.* 2nd International Symposium on Ubiquitous Computing and Ambient Intelligence – 2007. Zaragoza, Spain. 2007.

1.9. Estructura de la tesis

La presente tesis se divide en seis capítulos. A continuación comentaremos la organización de los cinco capítulos restantes.

En el capítulo 2 se presenta un estado del arte dentro del contexto de este tema. Primeramente se comienza dando una introducción a las redes Ad Hoc, se describe luego lo relativo a las redes móviles, sus orígenes, sus definiciones, sus características, etc,. Se hace una clasificación de los protocolos de descubrimiento de servicios existentes, revisando los principales mecanismos de búsqueda de servicios en MANET. También se repasan las principales arquitecturas de descubrimiento de servicios disponibles a día de hoy; se incluye un resumen de un total de treinta y cinco arquitecturas, no pretende ser una enumeración exhaustiva, pero sí recoge los sistemas más importantes. Se realiza un análisis comparativo entre los principales protocolos de descubrimiento de servicios. Se mencionan los retos más importantes en este campo. Se incluye un breve listado de los protocolos de encaminamiento usados en redes Ad Hoc. Se describen las técnicas de clustering en los protocolos de encaminamiento y el diseño Cross-layer, y final-

mente, se realiza una revisión del tema de los modelos de movilidad empleados en la simulación de estos protocolos.

En el capítulo 3 se describe la parte más importante de la tesis, que es la propuesta que planteamos para solucionar los problemas existentes en las arquitecturas actuales de descubrimiento de servicios. Se continua describiendo los esquemas sobre los que se basa la propuesta (diseño Cross-layer y diseño de clustering) y los trabajos previos que han servido como punto de partida para esta investigación. En este capítulo se describe el diseño general de LIFT, se dan detalles de los formatos de los mensajes de nuestro protocolo. Se describe el funcionamiento analítico de LIFT, se proporciona una descripción de la gestión de grupos. Y por ultimo, se detalla el mecanismo de descubrimiento de servicios de LIFT.

En el capítulo 4 describimos la metodología para validar la propuesta basada en simulación. En ella se lleva a cabo la simulación para validar la viabilidad de LIFT, del mismo modo se describe el método de simulación, se listan los elementos que contiene el escenario de simulación y los parámetros empleados. Se describe la herramienta Ns-2 que es utilizada para simular nuestro modelo, y se cierra el capítulo con los detalles de la simulación realizada.

El capítulo 5 versa sobre el estudio de rendimiento y de los resultados obtenidos en la simulación de LIFT, así como la discusión con respecto a dichos resultados. Primeramente se mencionan los tipos de dispositivos considerados en la simulación, se describen las métricas utilizadas para evaluar el rendimiento de los protocolos de descubrimiento de servicios. Se dedica una sección para analizar las mediciones aplicadas para comprobar la minimización de recursos por parte de LIFT, en otra sección se tratan las métricas relacionadas con la fiabilidad del protocolo. Se realizan pruebas de escalabilidad. Existe una sección específicamente para realizar una comparativa con otro protocolo. Para cerrar, se lleva a cabo un análisis cualitativo en el que se recogen las principales conclusiones del rendimiento de LIFT. Y al final del capítulo, se describen los escenarios idóneos para el protocolo.

Finalmente, en el capítulo 6 presentamos las conclusiones de este trabajo. Hablaremos sobre los resultados obtenidos, restricciones y limitaciones encontradas. Por último, se mencionan las posibles soluciones que podrían ser el comienzo de trabajos futuros que pudieran mejorar los resultados obtenidos con el presente trabajo.

Capítulo 2

Estado del arte

En la primer sección de este capítulo se revisa el tema de descubrimiento de servicios en redes móviles Ad Hoc y se aborda el estado de desarrollo actual de este tópico. Se pone en primer lugar esta temática dado que este es el tema principal que ocupa a esta investigación.

Para dar mayor énfasis a este tema, en la sección 2.2 se incluye una clasificación de las arquitecturas de descubrimiento de servicios en función de la infraestructura de descubrimiento empleada. Dentro de cada arquitectura se incluye una serie de apartados en los que se relata de manera suscinta los principales protocolos de descubrimiento de servicios pertenecientes a esa clasificación. Con esto se da a conocer la variada cantidad de protocolos que se han propuesto.

En la sección 2.3 se presenta un estudio de la evolución de los protocolos de descubrimiento de servicios, el motivo es conocer la cronología de la aparición de dichos protocolos, así como las estrategias y técnicas que proponen.

Se continúa en la sección 2.4 con un análisis comparativo de las técnicas de descubrimiento empleadas en los protocolos, a fin de conocer las diferencias esenciales planteadas en tales protocolos.

En la sección 2.5 se lleva a cabo un análisis del funcionamiento de los protocolos de descubrimiento de servicios. Centrándose principalmente en los más importantes enfoques propuestos.

Se dedica la sección 2.6 exclusivamente para mencionar los retos actuales más importantes en este tema, así como los retos aún pendientes de resolver.

La sección 2.7 trata acerca del tema de los protocolos de encaminamiento en MANET, se revisan las tres categorías de protocolos de encaminamiento existentes, por este motivo se trata detalladamente el protocolo AODV porque en este protocolo se basa nuestra solución.

Más adelante, en la sección 2.8 se lleva a cabo una revisión de las técnicas de clustering, incluyendo los principales y más representativos algoritmos de clustering empleados para la elección de clusterheads; se incluyen estas técnicas en este capítulo porque la solución que proponemos utiliza un mecanismo de clustering para descubrir servicios, de ahí la razón por tratar este tema.

También se reserva la sección 2.9 para abordar el tema de diseño Cross-layer, en virtud de que la solución planteada emplea este tipo de diseño para descubrir servicios durante el proceso de encaminamiento (capa de red). Esta es la razón de incluir este tema en este capítulo.

Para cerrar el capítulo, en la sección 2.10 se repasan los principales modelos de movilidad, puesto que en este tema se describen los patrones de movilidad utilizados para la simulación de este tipo de entornos. Y nuestro proyecto usa la simulación como medio para validar la propuesta de solución planteada.

2.1. Descubrimiento de servicios en MANET

En entornos dinámicos es común ver que los dispositivos que ofrecen varios tipos de servicios se conecten y desconecten de la red en algún momento. El descubrimiento de servicios permite a los dispositivos localizar de forma automática servicios y anunciar sus capacidades a la red [33]. Y para lograr el descubrimiento de servicios se deben considerar ciertos requisitos en el momento del diseño:

1. Permitir el funcionamiento con restricciones de recursos (dispositivos inalámbricos de corto alcance y energía limitada).

2. Permitir el descubrimiento de servicios en grandes redes móviles Ad Hoc (escalabilidad).

3. Permitir el paso de redes móviles Ad Hoc a redes basadas en infraestructura como Intranet y/o Internet (Interoperabilidad).

Dado que los dispositivos cada día son más pequeños y baratos, son más limitadas sus capacidades de memoria, energía y alcance, por lo que es preciso desarrollar arquitecturas que permitan trabajar con estas restricciones físicas. Así mismo los dispositivos móviles se han popularizado tanto que ha aumentado el número de usuarios y por ello se debe pensar en cómo resolver el problema de escalabilidad cuando concurran muchos usuarios en una determinada red Ad Hoc.

A diferencia de las redes cableadas, las redes móviles Ad Hoc presentan cambios de topología muy frecuentes e impredecibles debido a la movilidad de sus nodos. En consecuencia, estas características de movilidad no permiten emplear protocolos convencionales desarrollados para las redes cableadas. Trabajar con dispositivos móviles en una red Ad Hoc implica diseñar sistemas que permitan funcionar óptimamente a la red en su conjunto, a pesar de lo complejo del entorno. Uno de los retos principales en una red móvil Ad Hoc es diseñar algoritmos robustos para el encaminamiento y descubrimiento de servicios que se adapten a los cambios aleatorios y frecuentes de la topología de la red. De ahí que el descubrimiento de servicios sea uno de los temas de interés de los grupos de investigación en este campo.

Con el propósito de estudiar las soluciones disponibles para descubrir servicios en MANET, nos hemos dado a la tarea de clasificar las arquitecturas de descubrimientos de servicios. En la figura 2.1 clasificamos las arquitecturas de descubrimiento de servicios dependiendo de si la arquitectura posee o no, un directorio para almacenar información relacionada con los servicios disponibles en la red. Se ha clasificado de esta forma según la infraestructura de descubrimiento de servicio o por el tipo de directorio que emplea. Esta categorización está basada en clasificaciones previas encontradas en [33] [47] [96] [101] y le hemos añadido

una lista de ejemplos de arquitecturas de descubrimiento de servicios que se corresponden con cada categoría. Existe un número de arquitecturas que caen en determinada categoría, debido a que ha sido bajo esa clasificación en la que se han diseñado más soluciones. No obstante, existen muchos más ejemplos de arquitecturas que no se incluyen en este trabajo.

La taxonomía es la siguiente, inicialmente estas arquitecturas se clasifican en *arquitecturas sin directorio* (definida en el apartado 2.2.1) y *arquitecturas basadas en directorio*, este segundo tipo de arquitecturas se dividen en dos categorías: *arquitecturas de directorio centralizadas* (apartado 2.2.2) y *arquitecturas de directorio distribuidas*, a esta última le corresponde una categorización más: *arquitecturas de directorios distribuidas basadas en infraestructura* (apartado 2.2.3) y *arquitecturas de directorios distribuidas sin infraestructura* (apartado 2.2.4). En caso de ser una *arquitectura de directorio distribuida sin infraestructura*, es la arquitectura ideal para una red Ad Hoc, ya que las MANET tienen como característica que se forman espontáneamente y carecen de una infraestructura previa, básicamente se forman con la aproximación física de dispositivos móviles al momento de descubrirse entre sí y compartir recursos.

Por otra parte, en la tabla 2.1 presentamos 27 protocolos de descubrimiento de servicios; se clasifican en base a la infraestructura de descubrimiento o tipo de directorio empleado (como se presenta en la figura 2.1). En la tabla 2.1 se incluyen las siguientes categorías: El *protocolo* que es el nombre del SDP. El *año*, columna que menciona el año cuando fue propuesto el protocolo. El *tipo de directorio*, básicamente depende si tiene o no directorio y se puede elegir entre basado en directorio, sin directorio, directorio distribuido basado en infraestructura y directorio distribuido sin infraestructura. La *topología* o relación funcional, es un mecanismo usado para manejar información de servicios y consultas de clientes, tales arquitecturas pueden ser: P2P, cliente-servidor, centralizada, red overlay, backbone virtual y área amplia. El *tipo de red*, puede ser desde redes móviles Ad Hoc hasta red overlay. El *almacenamiento de información de servicios*, es la forma en que un protocolo de descubrimiento de servicios mantiene información de

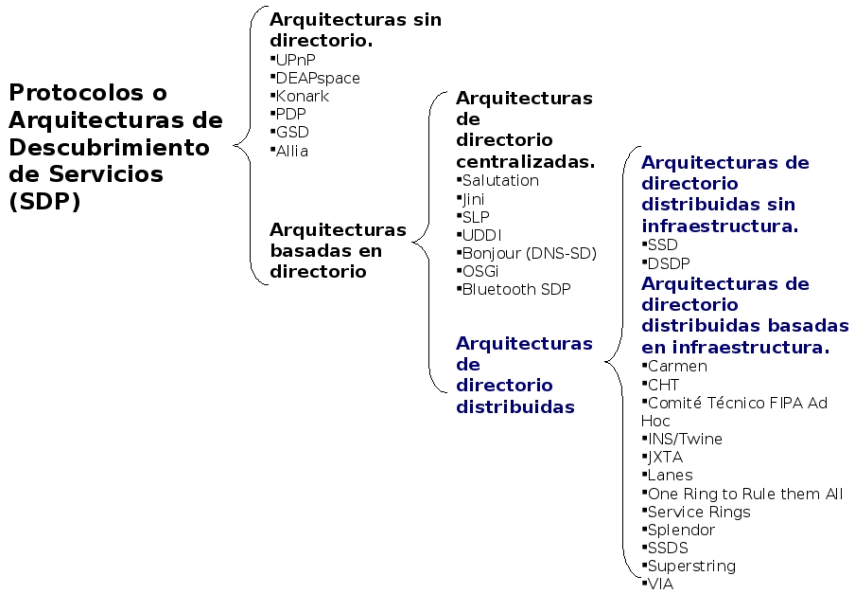


Figura 2.1: Clasificación de las arquitecturas de descubrimiento de servicios.

los servicios disponibles, en este componente cada protocolo de descubrimiento de servicios denomina con un nombre específico a su propio tipo de repositorio. Por último, la *técnica de descubrimiento de servicios*, cada protocolo posee una técnica en la que se basa para descubrir servicios y estas técnicas pueden ser mediante: repositorio centralizado de servicios, inundación de peticiones, encaminamiento semántico y tabla Hash.

Una clasificación muy genérica de los modelos de descubrimiento de servicios se encuentra en [162] y es como sigue: modelo *cliente-servicio*, cuando un cliente necesita algún servicio, envía su petición a través de broadcast o multicast, los servicios que se correspondan con la búsqueda solicitada devuelven un *reply*; en este modelo no existe ningún servidor de directorio para almacenar la información de servicios y este modelo es más apropiado para pequeñas redes. El otro modelo de esta clasificación es *cliente-servicio-directorio*, en este modelo existe un directorio para almacenar toda la información de servicios disponible

Protocolo	**Año**	**Tipo de directorio**	**Topología**	**Tipo de red**	**Almacenamiento de información de servicios**	**Técnica de descubrimiento de servicios**
Salutation	1999	Basado en directorio	Cliente-servidor - P2P	Cualquier red, depende del transporte	Salutation manager	Repositorio centralizado de servicios
SLP	1999	Basado en directorio	Centralizado, P2P	Subred	Dictionary Agent (DA)	Repositorio centralizado de servicios
OSGi	1999	Basado en directorio	Centralizado	WAN, LAN	OSGi service registry	Repositorio centralizado de servicios
BLUETOOTH	1999	Basado en directorio	Cliente-servidor	Mobile Ad Hoc	Service Record	Inundación de peticiones
SSDS	1999	Directorio distribuido con infraestructura	Área amplia	Estructura jerárquica de servidores	Directorios jerárquicos	Inundación de peticiones
UPnP	2000	Sin directorio	P2P	Subred	Punto de control	Inundación de peticiones
DEAPSpace	2001	Sin directorio	P2P	Mobile Ad Hoc, 1-hop	World View	Inundación de peticiones
VIA	2001	Directorio distribuido con infraestructura	P2P	Red overlay	Directorios jerárquicos	Inundación de peticiones
PDP	2002	Sin directorio	P2P	Mobile Ad Hoc	Cache	Inundación de peticiones
GSD	2002	Sin directorio	P2P	Mobile Ad Hoc	Cache	Encaminamiento semántico
ALLIA	2002	Sin directorio	P2P	Mobile Ad Hoc	Cache	Encaminamiento semántico
UDDI	2002	Basado en directorio	Centralizado	Empresarial	Directorios centralizados	Repositorio centralizado de servicios
BONJOUR	2002	Basado en directorio	Centralizado, P2P	Empresarial, Subred	Directorios jerárquicos	Repositorio centralizado de servicios
CARMEN	2002	Directorio distribuido con infraestructura	P2P	Árbol jerárquico, red semi-estática	Carmen proxy	Inundación de peticiones
INS/Twine	2002	Directorio distribuido con infraestructura	Área amplia, P2P	Red Overlay DHT	Resolvedores	Tabla Hash
JXTA	2002	Directorio distribuido con infraestructura	Área amplia, P2P	Red Overlay virtual	Rendevous Peers	Inundación de peticiones
One Ring to Rule them All	2002	Directorio distribuido con infraestructura	P2P de gran escala	Anillo universal	Red Overlay DHT	Inundación de peticiones
KONARK	2003	Sin directorio	P2P	Mobile Ad Hoc multihop	Service Registry	Inundación de peticiones
JINI	2003	Basado en directorio	Centralizado	Subred	Lookup Service	Repositorio centralizado de servicios
FIPA AD HOC	2003	Directorio distribuido con infraestructura	P2P	Mobile Ad Hoc	Directory Facilitator (DF)	Inundación de peticiones
LANES	2003	Directorio distribuido con infraestructura	Red Overlay	Mobile Ad Hoc	Nodos dentro de un carril compartiendo la misma información de servicios	Encaminamiento semántico
SERVICE RINGS	2003	Directorio distribuido con infraestructura	Red overlay de anillos jerárquicos	Mobile Ad Hoc	Access Point de servicios	Encaminamiento semántico
SPLENDOR	2003	Directorio distribuido con infraestructura	Clientes - servicios - directorios - proxies	Mobile Ad Hoc, entornos públicos	Directorios	Inundación de peticiones
SUPERSTRING	2003	Directorio distribuido con infraestructura	Área amplia	Red Overlay DHT	Directorios jerárquicos	Tabla Hash
DSDP	2004	Directorio distribuido sin infraestructura	Red de Backbone virtual	Mobile Ad Hoc	Directorio distribuido en el backbone virtual	Inundación de peticiones
SSD	2005	Directorio distribuido sin infraestructura	Red de Backbone virtual	Mobile Ad Hoc	Cache	Inundación de peticiones
CHT	2005	Directorio distribuido con infraestructura	Área amplia	Mobile Ad Hoc	Cada nodo a través de DHT, contenido céntrico	Tabla Hash, Encaminamiento semántico

Tabla 2.1: Características principales de los protocolos de descubrimiento de servicios.

en la red, de esta forma, un cliente envía sus consultas al servidor de directorio para encontrar y contactar el servicio requerido.

Otra manera de clasificar las soluciones para descubrir servicios en redes móviles Ad Hoc es el que se propone en este trabajo [82], que los categoriza por la técnica de descubrimiento de servicios que emplea y puede ser: basado en repositorio centralizado de servicios, basado en inundación de peticiones, basado en hashing y basado en encaminamiento semántico (estas técnicas se encuentran también contenidas en la última columna de la tabla 2.1).

Finalmente, como se ha explicado en el capítulo 1 respecto a los dos modelos para relacionar clientes y proveedores de servicios, existe un modelo basado en *interacción directa* y otro modelo basado en *interacción indirecta*. En la tabla 2.2 se resumen estos dos modelos de interacción. El diseño basado en una interacción directa entre clientes y proveedores se denomina *Inundación de peticiones*, mientras que el diseño basado en una relación indirecta se denomina *Basado en intermediarios*.

La tabla 2.2 contiene las siguientes categorías: las *características* más importantes del modelo de solución, como puede ser el medio de almacenamiento de la información de servicios, la infraestructura de descubrimiento (centralizado o distribuido), el tipo de conexión, el nivel de escalabilidad, el esquema de agrupamiento, etc. El *modo* de interacción o esquema de relación entre clientes y proveedores de servicios (directo o indirecto). Los *métodos de búsqueda*, si las búsquedas son por difusión de peticiones o mediante el anuncio periódico de servicios. Los *inconvenientes* que traen consigo la decisión de determinada política o estrategia de solución, y su impacto en el gasto de ancho de banda o consumo de recursos y por último, las *propuestas de solución* generales que plantean muchos SDP para resolver los problemas presentes y para atenuar los inconvenientes.

Técnica de descubrimiento de servicio	**Características**	**Modo**	**Método de búsqueda**	**Inconvenientes**	**Propuestas de solución**
Interacción directa (Inundación de petición de servicios)	- Ideal para MANET altamente dinámicas. - Almacena información de servicios en el propio dispositivo (caché local). - No distribuye su descripción de servicios al resto de nodos. - Conexión one-hop	Directo	Activo (pull) los nodos clientes difunden mensajes a toda la red en busca de servicios. - Los nodos proveedores que poseen el servicio solicitado responden por unicast	Satura la red de mensajes de petición de servicios. - Altos costos de comunicación, energía y computación.	- Reducir el número de mensajes de peticiones (control de la periodicidad, selectividad, circunscripción). - Reducir el número de peticiones duplicadas. - Seleccionar inteligentemente los nodos destinos. - Mantener los cachés locales actualizados. - Responder y atender las peticiones de servicios de los nodos más cercanos al nodo solicitante e idear un mecanismo para gestionar las respuestas. - Agrupar nodos por criterios en particular (tipo de servicio, área geográfica, etc.) para reducir el tráfico de mensajes a menos nodos.
			- Pasivo (push) - Los nodos proveedores difunden mensajes anunciando servicios. - Los nodos clientes almacenan en caché local los servicios anunciados.	- Satura la red de mensajes en ocasiones no solicitados. - Altos costos de comunicación, energía y computación.	- Reducir la difusión de mensajes de anuncios de servicios (control de la periodicidad, selectividad, circunscripción). - Enviar mensajes de anuncios a nodos con cachés menos actualizados. - Enviar mensajes de anuncios a nodos próximos. - Discriminar los anuncios de servicios ya registrados en caché local. - Agrupar nodos para enviar peticiones dentro del mismo grupo y evitar el envío a toda la red.
Interacción indirecta (Basado en intermediarios)	- Ideal para MANET medianamente dinámicas. - Es posible alcanzar cierto grado de escalabilidad. - No todos los nodos poseen localmente un directorio de servicios. - Directorios de servicios distribuidos. - Uno o varios nodos se comportan como servidores. - Sistema descentralizado y distribuido para repartir y balancear carga de trabajo. - Búsquedas ejecutadas por intermediarios. - Existen grupos de servicios, alianzas, anillos o círculos, árboles jerárquicos, backbone virtual y redes overlays. - Conexión multi-hop.	Indirecto	- Cada nodo registra sus servicios en el directorio del intermediario. - Se envían peticiones de servicios directamente a los directorios (multicasting). - La gestión de las peticiones de servicios las lleva a cabo el intermediario.	- Los nodos intermediarios son críticos para el resto de la red. - Complejidad para formar grupos y coordinarlos. - Aislamiento de nodos por depender de un intermediario (se desconoce lo qué existe más allá del intermediario). - Dificultad para comunicar nodos miembros de grupos con intermediarios, debido a muchos saltos para llegar del nodo origen al nodo destino. - La conexión multi-hop podría ser crítica por el alcance de radio de los dispositivos.	- Replicar el número de intermediarios para disminuir los fallos. - Implementar mecanismos para reemplazar eficientemente la caída de los nodos críticos. - Establecer políticas para la selección de nodos intermediarios en función de las capacidades de los dispositivos. - Mantener actualizados en todo momento los directorios distribuidos. - Establecer esquemas de interrelación entre intermediarios y entre intermediarios hacia/desde los nodos miembros.

Tabla 2.2: Técnicas de diseños para el descubrimiento de servicios en redes Ad Hoc.

2.2. Arquitecturas de descubrimiento de servicios

A continuación describimos cada una de las arquitecturas clasificadas en la figura 2.1 y debajo de cada arquitectura incluimos una serie de protocolos que caen en esa clasificación. Incluimos los más importantes protocolos de descubrimiento de servicios, damos un breve vistazo a sus principales características, su estado de desarrollo actual y una ligera descripción de sus mecanismos básicos de funcionamiento.

Estos trabajos describen intentos por encontrar una solución al problema de descubrimiento de servicios en MANET y de alguna manera están relacionados directamente con esta investigación.

2.2.1. Protocolos con arquitectura sin directorio

Se trata de aquellas arquitecturas de descubrimiento de servicios que no emplean ningún tipo de repositorio/directorio para almacenar información de los servicios disponibles en la red. Es decir, los proveedores de servicios no distribuyen la descripción de sus servicios en el resto de nodos de la red, en vez de esto los mantienen almacenados en el mismo dispositivo, algunos de ellos emplean un caché local. Para localizar servicios se envían peticiones de servicios a todos los miembros de la red con el objetivo de encontrar la descripción de servicios solicitada.

Por lo general este tipo de mecanismos de difusión no son adecuados para redes móviles Ad Hoc debido a un alto consumo de ancho de banda y energía, que por cierto es muy limitado y reducido en los dispositivos móviles. De modo que el tamaño de la red que puede soportar la arquitectura sin directorio es muy limitado. Sin embargo, en áreas extremadamente dinámicas, la difusión es la única solución [33].

En los siguientes apartados incluimos algunos ejemplos de protocolos con arquitectura sin directorio.

2.2.1.1. UPnP

UPnP(Universal Plug and Play) propuesto por Microsoft. UPnP es una arquitectura para la conectividad de redes ubicuas Peer-to-Peer (P2P), aplicaciones inteligentes, dispositivos inalámbricos y PCs. Dentro de UPnP un dispositivo puede unirse a la red, adquirir una dirección IP y aprender de la presencia y de las capacidades de otros dispositivos. No emplea registro de servicios central. Algo más, un dispositivo puede abandonar la red sin dejar ningún estado indeseado detrás. UPnP se soporta sobre TCP/IP y UDP y en tecnologías web como HTTP y XML; además, controla y transfiere datos entre los dispositivos conectados en el hogar y la oficina. UPnP utiliza Simple Service Discovery Protocol (SSDP) para el descubrimiento de servicios [151].

2.2.1.2. DEAPspace

DEAPspace. Propuesto inicialmente por IBM. El protocolo ha sido diseñado para operar de forma eficiente en redes inalámbricas Ad-hoc de un solo salto. A través del algoritmo DEAPspace, un dispositivo puede detectar la presencia de dispositivos próximos, compartir información de los servicios disponibles, al igual que detectar la indisponibilidad de otros dispositivos. El objetivo principal de este SDP es dar respuesta a los cambios frecuentes que se producen en el entorno, teniendo en cuenta las limitaciones y restricciones de potencia de los dispositivos. El algoritmo se basa en un método push puro, en el que todos los dispositivos mantienen un *world view* que transmiten por difusión cada cierto periodo de tiempo a sus vecinos y que se actualiza cuando recibe del resto de dispositivos los *world view* correspondientes. Una contribución de DEAPspace ha sido la definición del formato de descripción de los servicios y de su mecanismo de codificación para minimizar la cantidad de datos a transmitir. DEAPspace distingue entre el protocolo de comunicación para acceder a un servicio y el protocolo empleado para el descubrimiento [106] [72].

2.2.1.3. Konark

Konark es un SDP diseñado específicamente para redes Ad-hoc dentro de un esquema P2P. En general orientado hacia servicios en dispositivos independientes. Konark usa un mecanismo P2P completamente distribuido que provee en cada dispositivo la habilidad para anunciar y descubrir servicios. La descripción de servicios está basada en XML. Konark provee un sistema para conectar inalámbricamente servicios aislados ofrecidos por dispositivos ubicuos próximos a él. Cada dispositivo participante tiene la capacidad de albergar sus servicios locales, entregar sus propios servicios usando un servidor micro-HTTP residente, consultar la red buscando servicios disponibles ofrecidos por otros y usar los servicios descubiertos. Konark permite a cada dispositivo actuar como servidor y como cliente simultáneamente. Emplea el término cliente para cualquier dispositivo que esté interesado en usar los servicios ofrecidos por otros nodos. Así como define componentes que permiten a los dispositivos asumir el doble rol [70].

2.2.1.4. PDP

PDP (Pervasive Discovery Protocol). PDP es un SDP adaptado a entornos de computación ubicua. PDP es un protocolo de ámbito local, totalmente distribuido, cuyo funcionamiento no depende de directorios. El usuario, a través de sus aplicaciones solicita la búsqueda de un tipo de servicio y si así lo requiere, puede conocer también todos los servicios que ofrecen los dispositivos que le rodean. Es decir, los servicios se descubren bajo demanda. Cada dispositivo almacena en una caché local los servicios que va recibiendo y la comparte con todos los dispositivos que forman la red; la información almacenada en esta caché se consulta cada vez que una aplicación solicita un servicio, esto con la finalidad de minimizar el número de transmisiones necesarias para satisfacer la búsqueda. Esta idea se basa en el hecho de que la unión de la información almacenada en cada una de esas cachés, contiene la misma información que un directorio de servicios, pero en este caso se encuentra distribuida entre todos los dispositivos de la red [20].

2.2.1.5. GSD

GSD. Group-based Service Discovery Protocol es un protocolo basado en el concepto de caché de anuncio de servicios P2P y envío inteligente de peticiones de servicio basado en grupo. GSD no requiere ser registrado en un servidor de registro; no tiene el problema de tormenta de broadcast, optimiza el uso del ancho de banda y garantiza descubrir un servicio si éste se encuentra presente en la MANET. GSD explota las capacidades semánticas ofrecidas por DARPA Agent Markup Language (DAML) para describir en forma efectiva los servicios y recursos. Los servicios se clasificados dentro de diversos grupos 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 modulo de service-matching. Esto incrementa la flexibilidad en el descubrimiento de servicios y es idealmente adecuado para resolver la heterogeneidad de servicios en una MANET [24].

2.2.1.6. Allia

Allia Es una arquitectura distribuida basada en políticas. Emplea una caché de servicios dentro de una red P2P. Maneja elementos como DF (Directory Facilitator) y AMS (Agent Management System). Allia presenta 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 dispositivo y componentes de la plataforma son: Policy Manager: responsable de asegurar que todos los componentes de la plataforma están conforme con las políticas especificadas. Controla las tasas de avisos. Cache Manager: maneja los anuncios de servicios de los dispositivos del vecindario. Advertising Manager: envía broadcast de descripción de servicios registrado al DF local, también controla el diámetro de alianza. Forwarding Manager: recibe anuncios de servicios y peticiones de mensajes de servicios, además, decide basado en la política local, si detiene o propaga el anuncio [131].

2.2.2. Protocolos con arquitectura de directorio centralizadas

Se trata de protocolos de descubrimiento de servicios que utilizan un directorio, repositorio o servidor centralizado para almacenar información relacionada con los servicios disponibles en la red. En cada arquitectura, dicho directorio puede recibir distintos nombres. El directorio centralizado funciona como un servidor o directorio de páginas amarillas, en las que las estaciones o nodos registran sus servicios para ser consultados posteriormente por el resto de nodos existentes en la red. Los clientes y proveedores de servicios descubren los directorios por medio de multicast. Los proveedores de servicios anuncian sus servicios al directorio central a través de mensajes unicast. Para acceder a un servicio, el cliente primero contacta el directorio central para obtener la descripción de servicios y así poder interactuar con el proveedor de servicios. El descubrimiento de servicios centralizado es más adecuado para redes inalámbricas basadas en infraestructura. Sin embargo, este mecanismo implica que el proceso de descubrimiento de servicios dependa en todo momento de la disponibilidad del directorio centralizado.

A continuación hacemos mención de varias arquitecturas de descubrimiento de servicios basadas en arquitecturas de directorio centralizadas.

2.2.2.1. Salutation

Salutation fue desarrollado por Salutation Consortium. Salutation es un SDP compuesto por el Salutation Manager (SLM), que es el núcleo de la arquitectura. SLM cuenta con estos tres elementos: Service Registry: contiene un registro para almacenar información acerca de los servicios. Service Discover: descubre otros SLM para registrar servicios. Descubre servicios remotos que satisfacen los tipos y conjuntos de atributos especificados por el SLM local. Service Availability: aplicación cliente que periódicamente solicita al SLM verificar la disponibilidad de servicios. Este procedimiento es efectuado entre el SLM local y el correspondiente administrador [35].

2.2.2.2. Jini

Jini desarrollado por Sun Microsystems. La tecnología Jini proporciona acceso a servicios para cualquier tipo de red, para cualquier plataforma, sistema operativo y aplicación sin importar la topología de la red, distancia o dispositivo. Jini federa grupos de dispositivos y componentes de software dentro de un sencillo sistema distribuido dinámico. El sistema Jini tiene tres protocolos llamados discovery, join y lookup. La tecnología de conexión Jini consiste básicamente en una infraestructura y en un modelo de programación orientado fundamentalmente a la conectividad de dispositivos, con el propósito de formar una comunidad espontánea [77] [102].

2.2.2.3. SLP

SLP (Service Location Discovery) propuesto por IETF. SLP es un protocolo descentralizado, ligero, escalable y extensible para descubrir servicios dentro de un entorno distribuido. SLP define un URL que incluye el tipo y dirección del servicio. Emplea URLs para la descripción de los servicios, y sobre las cuales los usuarios se basan para conocer qué servicios existen en la red y cómo acceder a ellos. La infraestructura de SLP cuenta con tres tipos de agentes: User Agent (UA), Service Agent (SA) y Directory Agent (DA). El UA es una entidad de software que envía *requests* de descubrimiento de servicios en nombre de una aplicación de usuario. El SA es una entidad que anuncia servicios en nombre de un servicio. Como si fuera un repositorio de información de servicios centralizado, el DA almacena en caché los anuncios de los SAs y luego responde a los *requests* de los UAs [66].

2.2.2.4. UDDI

UDDI (Universal Description, Discovery, and Integration). Funciona como directorio para el registro de descripción de servicios. Es un registro público diseñado para almacenar de forma estructurada, información sobre los servicios que ofrecen las empresas. UDDI es una iniciativa de la industria que pretende

crear plataformas independientes, para describir servicios, descubrir e interactuar con negocios vía Internet. La especificación UDDI maneja información relacionada con los proveedores, las implementaciones y los metadatos de servicios. Un registro de negocios UDDI consiste de tres componentes: Páginas blancas: direcciones, contactos e identificadores conocidos. Páginas amarillas: Categorización industrial basada en taxonomías de estándares. Y Páginas verdes: Información técnica acerca de servicios expuestos por las empresas [150] [111].

2.2.2.5. Bonjour

Bonjour antes DNS-SD (Domain Name Server - Service Discovery) propuesto por ZeroConf, Apple. DNS-SD usa interfaces estándares DNS, servidores y formatos de paquetes para buscar servicios en la red. DNS-SD brinda soporte al descubrimiento de servicios a través de registros de recursos DNS ya existentes. Realiza consultas que permiten a un usuario obtener una lista de instancias de un tipo específico de servicios. DNS-SD es una arquitectura basada en estructura jerárquica de servicios o en modificaciones DNS para la resolución de nombres en redes sin infraestructura, con arquitecturas completamente distribuidas como Multicast DNS o LLMNR (Linklocal Multicast Name Resolution). Dado el tipo de servicio y el dominio que el cliente busca, la convención permite a los clientes descubrir una lista de instancias de nombres del servicio deseado que usan únicamente consultas DNS [42].

2.2.2.6. OSGi

OSGi (Open Services Gateway Initiative) ha sido desarrollado por Sun Microsystems. Fue creado con el objetivo de definir y promover un estándar abierto para hacer posible la conectividad de WAN a LAN o WAN a LON (domóticas). Las especificaciones OSGi definen un entorno de computación estandarizado y orientado a componentes para servicios en red. Los componentes de software pueden ser instalados, actualizados o eliminados sin tener que interrumpir la operación del dispositivo. Los componentes de software son librerías o aplica-

ciones que pueden descubrir dinámicamente y usar otros componentes. La alianza OSGi ha desarrollado muchas interfaces de componentes estándares que se encuentran disponibles con funciones comunes tales como: servidores HTTP, logging, seguridad, administración de usuario, XML y muchos más. El componente central de las especificaciones OSGi es el marco OSGi quien provee un entorno estandarizado para aplicaciones y se divide en cuatro capas: Ejecución, Módulos, Ciclo de vida y Registro de Servicios [1].

2.2.2.7. Bluetooth

Bluetooth es un protocolo de comunicación definido como un conjunto de protocolos a nivel de aplicación que facilitan el desarrollo de aplicaciones denominadas Bluetooth. Bluetooth contiene un SDP para el descubrimiento de servicios. Este protocolo sigue el esquema clásico cliente/servidor cuyas descripciones de servicios están almacenadas como registros de servicios dentro de un dispositivo de SDP. Dado que Bluetooth SDP esta diseñado específicamente para entornos Bluetooth, sí soporta la funcionalidad limitada de los dispositivos. No provee acceso a servicios, intermediación de servicios, anuncio de servicios, ni registro de servicios. No existe notificación de eventos cuando los servicios no se encuentran disponibles [13].

2.2.3. Protocolos con arquitectura de directorio distribuida con infraestructura

Los protocolos de descubrimiento de servicios basados en arquitecturas de directorio distribuidas basadas en infraestructura requieren de la existencia previa de redes cableadas o inalámbricas para que los nodos móviles puedan registrarse y descubrir servicios disponibles en la red. Existen redes móviles Ad Hoc que operan de forma aislada o independiente y cada red móvil Ad Hoc a su vez se encuentra conectada a una red fija con infraestructura que emplea directorios. La mayoría de las redes móviles Ad Hoc que existen actualmente son híbridas y

no 100 % puras; se encuentran conectadas a través de redes fijas, mientras que los directorios se encuentran en redes fijas con infraestructura.

Enseguida mencionamos algunos de los más representativos protocolos con arquitectura de directorio distribuida con infraestructura.

2.2.3.1. Carmen

Carmen. Brinda soporte a una gran cantidad de nodos heterogéneos, tanto dinámicos como estáticos, ofrece y busca servicios e información en redes inalámbricas. Generalmente emplea un mecanismo de descubrimiento basado en el modelo Pull, es eficiente para entornos semi-estáticos y lleva a cabo las entregas a través de proxies. Carmen usa XML para la descripción y anuncio de servicios. En Carmen los clientes acceden a los servidores en Internet a través de un nodo estático o un access point que actúa como gateway para el cliente. Carmen cuenta con tres tipos de nodos: *Carmen cliente*, *Carmen Proxy* y *Carmen service provider*. Los clientes se conectan a los proxies como si fueran hojas del árbol. Anuncian los servicios que tiene disponibles, y/o consultan los servicios que necesitan. Los host finales pueden ser demasiado móviles, conectarse y desconectarse frecuentemente, pero los proxies se localizan principalmente en servidores estáticos por Internet. Para reducir el tráfico y mejorar los resultados de las búsquedas, Carmen usa información contextual para construir el árbol del proxy y aprovechar los dominios de servicios en los proxies [99].

2.2.3.2. CHT

CHT (Contextual Hash Table). Almacena información contextual basada en semántica. Presenta el esquema basado en tabla Hash distribuida (DHT) que usa semántica de información contextual para decidir el sitio en el que será almacenada. Cada dispositivo en el sistema contiene información contextual acerca de si mismo y del entorno (Super Context Provider Peers) que requiere estar replicado a través de la red. CHT logra distribuir información eficientemente en áreas de la red en la que la información es más demandada por los nodos

vecinos. Un mecanismo de replicación garantiza que la información esté siempre disponible a otros usuarios y el esquema de tabla Hash ofrece un balanceo de almacenamiento de la información y reduce el tiempo de acceso tanto para el almacenamiento como para la recuperación de información. Básicamente, CHT consiste en replicar información contextual sobre los nodos participantes [107].

2.2.3.3. FIPA ad-hoc

FIPA ad-hoc. Protocolo propuesto por FIPA. Este modelo descubre agentes desplegados en plataformas de agentes (AP). El Facilitador de Directorio (DF) federa los agentes y funciona sólo si los nodos de la red no se conectan o desconectan muy continuamente. El Servicio de Descubrimiento de Agente (ADS) provee la funcionalidad de descubrimiento para casos cuando los nodos se conectan o desconectan con relativa frecuencia. El modelo de referencia aprovecha varios Middleware de Descubrimiento (DM) dependiendo de la tecnología ad-hoc subyacente. FIPA ad-hoc fundamenta sus trabajos en tres requisitos: 1). Añadir los mínimos cambios a las especificaciones existentes de FIPA. 2). Utilizar los mecanismos de descubrimiento dinámico de servicios existentes en la actualidad (Jini, JXTA, SLP, SSDP, etc.), de manera que la solución propuesta pueda interoperar con cualquiera de ellos. Y 3). Adaptarse a entornos Ad Hoc, pero que la solución propuesta también sea válida para redes con infraestructura [11] [54].

2.2.3.4. INS/Twine

INS/Twine. Emplea resolvedores (clientes, dispositivos) que colaboran como terminales para distribuir información de recursos (servicios) y resolver consultas. INS/Twine mapea recursos para los resolvedores transformando descripciones dentro de claves numéricas de forma que preserve su expresividad, facilite la distribución de datos y permita la resolución eficiente de consultas. Además. INS/Twine logra ser escalable porque realiza una partición de las descripciones de recursos basado en Hash. INS/Twine usa un conjunto de resolvedores que se organizan entre sí dentro de una red overlay que enruta descripciones de recur-

sos de todos los nodos, para almacenar y resolver de manera colaborativa las consultas de clientes. Cada resolvedor tiene conocimiento acerca de un subconjunto de los otros resolvedores en la red. INS/Twine está diseñado para llevar a cabo descubrimiento de recursos escalable en un entorno en el que todos los recursos son igualmente útiles. Para lograr estos objetivos, INS/Twine se basa en un proceso de distribución de tablas Hash (DHT) [6].

2.2.3.5. JXTA

JXTA. Plataforma de cómputo distribuido P2P que permite a cualquier dispositivo (PDA, teléfono móvil, sensor electrónico, etc) comunicarse, colaborar y compartir recursos. Los nodos JXTA crean una red Ad-hoc sobre las redes existentes. En la red JXTA, cualquier nodo puede interactuar sin importar ubicación, tipo de dispositivo o ambiente operativo, incluso cuando algunos puntos están ubicados detrás de firewalls o en diferentes transportes de red. De modo que, el acceso a los recursos en la red no está limitado a las incompatibilidades de la plataforma o a las restricciones de la arquitectura jerárquica cliente-servidor. JXTA se basa en tecnologías probadas y en estándares como HTTP, TCP/IP y XML. La tecnología JXTA no depende de un lenguaje de programación en particular, ni de una plataforma de red o ni de una plataforma de sistema, y puede trabajar con cualquier combinación de estas [62].

2.2.3.6. Lanes

Lanes. Propone un nuevo método para el anuncio y descubrimiento de servicios en MANET. La idea básica consiste en definir una estructura overlay de dos dimensiones, llamados Lanes (carriles). Lanes es similar pero menos estricto que el enfoque presentado en Content Addressable Network (CAN)[130]. Una de las dos dimensiones de este overlay se usa para propagar anuncios de servicios, mientras que la otra dimensión es de uso exclusivo para distribuir peticiones de servicios. Este mecanismo produce una eficiente estructura resistente a fallos, que puede usarse para el descubrimiento de servicios basado en semántica. La

estructura de dos dimensiones de CAN es optimizada por la construcción carriles de nodos ligeramente acoplados. Los anuncios de servicios son propagados a través de un carril en forma de top-bottom. De modo que un nodo arbitrario del carril tiene información completa de los servicios ofrecidos por ese carril. Las peticiones de servicios son propagadas por el otro carril usando encaminamiento anycast [82].

2.2.3.7. One Ring to Rule them All

One Ring to Rule them All. Propone el uso de un anillo universal que ofrece únicamente la funcionalidad de *bootstrap*, al mismo tiempo que cada servicio se ejecuta en una red overlay P2P por separado. El anillo universal provee: un servicio de índice que permite a los usuarios encontrar servicios de interés para las consultas booleanas proporcionadas; un servicio multicast usado para distribuir software que se actualiza y se coordina entre miembros de un servicio overlay; una red de almacenamiento y distribución continua que permite a los usuarios obtener el código necesario para participar en un overlay de servicio; y un servicio para proporcionar a los usuarios un nodo de contacto para unirse a un servicio de overlay. Estos servicios son auto-organizados, tolerante a fallos y escalable a un gran número de nodos. Los nodos en los overlays de servicios específicos forman subconjuntos de nodos del anillo universal. El anillo universal permite a los dispositivos anunciar y descubrir servicios de interés, encontrar el código requerido para participar en un overlay de servicio en particular, y encontrar un nodo de contacto para unirse al overlay de servicio [21].

2.2.3.8. Service Rings

Service Rings. Monta una estructura overlay de anillos jerárquicos en la capa de transporte. Con esta estructura encamina eficientemente anuncios y peticiones de servicios a los destinos correspondientes por medio del análisis de contenido semántico del servicio descrito, mientras es lo suficientemente flexible para adaptarse a los cambios constantes de la topología de red subyacente. Ser-

vice Rings emplea una arquitectura de anillo jerárquico (grupo de dispositivos que están físicamente cercanos unos a otros y que ofrecen servicios similares). Cada dispositivo tiene un punto de acceso de servicio designado (SAP), que almacena información acerca de todos los servicios ofrecidos por el anillo. Los SAPs pueden ser organizados en anillos de alto nivel, los cuales tienen también SAPs que almacenan resúmenes de los servicios que proporcionan. Los anillos permanentemente monitorizan el tráfico en la red y toman decisiones para optimizar su estructura [83].

2.2.3.9. Splendor

Splendor. Es un modelo ideal para soportar usuarios móviles en entornos públicos. Splendor propone un nuevo modelo para soportar movilidad, seguridad y privacidad de usuario, al mismo tiempo que integra location-awareness para el descubrimiento de servicios. Los protocolos de seguridad en Splendor permiten a todas las partes autenticarse mutuamente unos a otros, no importando si los usuarios tienen o no una cuenta en los sistemas de infraestructura de red. Splendor soporta privacidad de usuario. Por otra parte, Splendor integra location-awareness, lo cual no sólo ayuda al descubrimiento de servicios, sino que reduce los requerimientos de infraestructura de la red para el descubrimiento de servicios. Splendor emplea cuatro tipos de componentes: clientes, servicios, directorios y proxies. Al incluir proxies se consigue privacidad de los proveedores de servicios, descargando mucho trabajo computacional de los servicios móviles a los proxies y permitiendo a los servicios móviles hacer la autenticación y autorización más fácilmente. Splendor utiliza tags para conocer la localización de clientes y servicios (location-awareness). Las tags que etiquetan lugares, emiten información de localización y opcionalmente direcciones de directorios y certificados de los directorios. Los clientes usan la información de localización para buscar servicios relevantes. Los servicios móviles usan la información de las tags para determinar que se han movido a nuevos sitios y notificar sus proxies. Los

tags que portan las personas se usan para verificar que los dispositivos están aún en posesión de sus usuarios [161].

2.2.3.10. SSDS

SSDS. Secure Service Discovery Service. Pone énfasis en la seguridad y soporte de un gran número de servicios, conocido como soporte de área amplia. En SSDS los servicios y los clientes utilizan directorios. La autenticación entre clientes y servicios está basada en certificados. Se usa encriptación de claves públicas y de claves simétricas para la confidencialidad y privacidad de la comunicación de los datos. Usa también un Message Authentication Code (MAC) para asegurar la integridad del mensaje y dispone de mecanismos de autenticación y autorización. Para el soporte de área amplia, se cuenta con distintas estructuras de directorio jerárquicos. Para el manejo del tamaño de la red SSDS utiliza métodos de compresión para almacenar la información. SSDS permite la selección múlti-criterios, que produce una colección de nodos que satisfacen más de una condición. SSDS soporta esta característica al implementar un método llamado inundación de búsqueda filtrada (filtered query flooding), en el que resume el contenido de los nodos que usa como filtros para las búsquedas [39].

2.2.3.11. Superstring

Superstring. Diseñado especialmente para entornos extremadamente dinámicos y en el que las consultas se resuelven en el núcleo de la red, dicho núcleo es relativamente estático, compuesto por servidores de gran capacidad y en redes con gran ancho de banda. Superstring combina P2P y las topologías jerárquicas de directorios. Desde el nodo de nivel más alto se crea una jerarquía, que refleja la estructura jerárquica de las descripciones de servicios. Las consultas se resuelven por la jerarquía especializada. Este SDP distribuye la descripción de servicios sobre diversos nodos. De esta forma, cada directorio almacena sólo una pequeña parte de la descripción completa. En Superstring no se procesan las consultas en un único directorio. El costo es compartido entre muchos directorios de tal

manera que se minimice la sobrecarga de memoria. Superstring reconoce que ciertas clases de servicios son más populares que otros. Por lo que ciertas consultas son elaboradas con más frecuencia que otras, esto significa que algunos resolvedores tengan más trabajo que otros. Superstring distribuye la carga extra de trabajo sobre múltiples resolvedores, evitando que un sólo resolvedor cargue con todo el trabajo [135].

2.2.3.12. Via

Via. Verified Information Access, es un protocolo a nivel de aplicación que permite compartir datos entre los dominios del descubrimiento. Cada directorio mantiene una tabla de enlaces o vínculos activos a otros directorios que comparten información relacionada. Un conjunto de directorios vinculados forman un cluster de datos con el fin de poder ser consultados por los dispositivos que buscan información. El cluster de datos está distribuido, auto-organizado y es responsable de la movilidad de los datos. Se usan esquemas de datos definidos por la aplicación, los clusters se organizan entre sí dentro de una jerarquía. En Via el servicio de directorio maestro reside sobre un gateway que conecta un dominio de descubrimiento a un canal principal, permitiendo la comunicación entre gateways. Los gateways actúan como mediadores entre los dispositivos en un dominio de descubrimiento y los recursos disponibles en otros dominios. Un cliente envía todas las consultas al canal principal. Los gateways se auto-organizan dentro de una jerarquía en base a los tipos de consultas en el canal principal, al igual que los elementos de datos actuales que están siendo compartidos en la red [22].

2.2.3.13. LSD

LSD es un protocolo que soporta el modelo *cliente-servicio* y el modelo *cliente-servicio-directorio*. Es un protocolo automático, si existe un servidor de directorio disponible en la red, los clientes le envían peticiones, por el contrario, los clientes utilizan mecanismos de broadcast o unicast. Además, en el servi-

cio de consulta y respuesta. Para coadyuvar en el proceso de descubrimiento de servicios, LSD usa mensajes montados (piggybacking) en el mecanismo del protocolo de encaminamiento proactivo. Un servidor de directorio bien anuncia periódicamente o bien cambia la configuración de la información. Si un servicio de un nodo no escucha algún servidor de directorio en un tiempo determinado, entonces usa un mecanismo de difusión para descubrir directorios y poder registrarse, sin embargo, si no existe un directorio conocido, un nodo replica directamente una petición con el servicio que concuerde con la petición. LSD ofrece escalabilidad para la gestión de la dimensión de las MANET de gran escala [90].

2.2.3.14. SEDIRAN

SEDIRAN (Service Discovery and Interaction with Routing protocols in Ad hoc Networks). Es un protocolo que establece una estrategia de descubrimiento y anuncio de servicios basado en diseño Cross-layer. Usa un mecanismo para minimizar el tráfico generado por el protocolo y adaptarse a los cambios frecuentes de topología. Establece una estrategia para el uso de memoria caché. Otra característica es la interacción con protocolos de encaminamiento. SEDIRAN utiliza dos tipos de servicios: *servicios ordinarios* - servicios conocidos por todos los nodos de la red y se almacenan en una base de datos representada por un árbol jerárquico de tipos de servicios. Para la gestión de servicios ordinarios se envían peticiones y respuestas a través de DREQ (Discovery REQuest) y DREP (Discovery REPly) respectivamente; y los *servicios especiales* - servicios no conocidos por todos los usuarios y son representados por un texto descriptivo y significativo para los usuarios, los servicios especiales no se almacena en ningún sitio; generalmente anuncian sus servicios a través de ADVM (ADVertisment Messages) y los usuarios potenciales deben aprender de su existencia. Existe una memoria caché que gestiona los servicios ordinarios y otra que se encarga de los servicios especiales [112].

2.2.3.15. ReMMoC

ReMMoC (Reflective Middleware to support Mobile Client Interoperability). Es una plataforma de descubrimiento de servicios reflexiva que se adapta dinámicamente tanto al enlace y como al protocolo de descubrimiento para permitir la interoperación con servicios heterogéneos. ReMMoC soporta el desarrollo de aplicaciones móviles y supera las propiedades heterogéneas del entorno móvil. ReMMoC consiste de dos componentes: 1) una estructura de enlace (binding) para la interoperación con los servicios móviles implementados sobre diferentes tipos de middleware y, 2) una estructura de descubrimiento de servicios. La función principal de la estructura de enlace es interoperar con servicios móviles heterogéneos. Mientras que la estructura de descubrimiento de servicios permite que los servicios sean anunciados por diferentes SDP para que pueda ser encontrado [63].

2.2.3.16. ZBS

ZBS (Zone-Based Service). Protocolo que entrega servicios a los usuarios móviles localizados dentro de determinada zona. Una zona puede tener uno o más servicios simultáneamente. Para lograr esto, ZBS sigue la localización de los usuarios móviles y los servicios disponibles en cada zona. La red ZBS está configurada con la mínima infraestructura y mantiene un perfil de cada dispositivo que identifica a cada usuario, al igual que mantiene un perfil de los proveedores de servicios. ZBS se soporta sobre una arquitectura física que consiste de ZSP (Zone Service Portal), IZSS (Infra-Zone Service Station), ZSS (Zone Service Station) y el usuario final. El ZSP y el IZSS son las infraestructuras construidas por los agentes de la red ZBS. El ZSP mantiene al LZD (Local Zone Database) que almacena los perfiles que identifican a cada usuario final y al proveedor de servicios. IZSS hace a la red ZBS más robusta y ayuda a los proveedores de servicios a localizarlos. ZBS supone que la infraestructura es robusta y que los sistemas son estáticos y tolerantes a fallos [75].

2.2.3.17. NOM

NOM es un sistema de localización de recursos (descubrimiento de servicios) completamente descentralizado, basado en una red P2P simple. NOM opera de modo completamente distribuido, en el que cada nodo ejecuta una copia del código NOM, monitoriza su red de tráfico local para detectar peticiones de localización de recursos y usa mensajes estándares para resolver esas consultas y proveer el descubrimiento de servicios al código de nivel aplicación. El algoritmo NOM está compuesto de un bucle que monitoriza los mensajes que llegan de la red y del código del nivel de aplicación, y reacciona de acuerdo al tipo de mensaje recibido, bien reenviando el mensaje recibido si no aplica al nodo actual, creando y reenviando un mensaje apropiado *petición-respuesta* si el nodo respondiera a la petición, o bien creando un mensaje de petición e insertándolo dentro de la red. NOM es una biblioteca que expone un conjunto de funciones que son implementadas por la plataforma subyacente particular (protocolo de encaminamiento y sistema operativo utilizados). Una vez implementado por la plataforma, es posible realizar varias optimizaciones, incluyendo almacenamiento en caché de direcciones físicas de los vecinos (dependiendo de la dinámica de la red) y devolver mensajes de petición directamente a los peticionarios [43].

2.2.3.18. SANDMAN

SANDMAN (Service Advertisement and Discovery for Mobile Ad hoc Networks) es un algoritmo de descubrimiento de servicios basado en agrupamiento. Con el objetivo de ahorrar energía, los nodos del cluster duermen cuando se encuentran en estado inactivo. Mientras que un nodo líder (Cluster Head, CH) permanece siempre activo y responde a las peticiones de descubrimiento de servicio a nombre de los nodos inactivos; esto con el objetivo de reducir las latencias del proceso de descubrimiento. SANDMAN tiene tres objetivos principales: Usar eficientemente la energía al permitir que los nodos duerman, minimizar la latencia de descubrimiento y descentralizar operaciones. En este algoritmo, los nodos del cluster (Cluster Nodes, CN) inactivos duermen periódicamente para ahor-

rar energía. Después de despertar un CN espera por un intervalo de tiempo la llegada de peticiones de clientes. Si no recibe peticiones informa al CH y vuelve de nuevo al modo inactivo. De esta forma, ahorra una importante cantidad de energía. El CH está despierto en todo momento. El CH actúa como *Lookup Service* local, de modo que puede responder instantáneamente a cualquier petición de descubrimiento ya que cuenta una lista con las descripciones de servicios y tiempos de *despertar* de los respectivos CNs [141].

2.2.3.19. SPIZ

SPIZ (Service Ad/D Protocol with Independent Zones) es un protocolo de servicios (`Ad/D`, advertisement and discovery frameworks) en zonas independientes, es un protocolo híbrido adaptativo de descubrimiento de servicios integrado a un eficiente protocolo de encaminamiento, por lo que podemos decir que emplea un diseño Cross-layer. SPIZ permite que los nodos se adapten a sus propias zonas de radios dinámica y automáticamente cambiar sus condiciones, como niveles de movilidad y popularidad. De esta forma, se basa en zonas de radio variables. Además, cada nodo tiene su propia zona de radio que facilita la adaptación automática a los cambios dinámicos en el entorno de red. Mantiene una zona óptima alrededor de cada nodo, que permite a SPIZ proveer un eficiente mecanismo de encaminamiento y procesamiento de peticiones. Este protocolo combina una estrategia de modelo Ad/D pasivo con el modelo Ad/D activo, ya que usa un protocolo de resolución de bordercasting. Integra el protocolo de red como resultado de un esquema ligero y reduce la cantidad de inundación de mensajes innecesarios en la red. SPIZ usa los paquetes de control de la capa de red existente y por tanto, ofrece una implementación ligera de Ad/D de servicios para evitar un excesivo gasto de recursos [109].

2.2.3.20. RUBI

RUBI es un enfoque de descubrimiento de servicios que consiste en encapsular un importante proceso adaptativo para controlar la forma de diseminar y

recuperar información y se basa en vistas locales de la estructura de la red. El diseño de RUBI se fundamenta en la hipótesis de que existen similitudes entre el descubrimiento de servicios y el algoritmo de encaminamiento de una red, ya que ambos se centran en la diseminación de información acerca de la disponibilidad y eficiencia de acceso a los servicios. RUBI usa un mecanismo de descubrimiento de servicios basado en un algoritmo de encaminamiento proactivo, para áreas de relativa baja movilidad y en regiones de alta movilidad descubre los servicios de forma dinámica usando un protocolo reactivo, y de esta forma descubre los servicios sólo cuando es necesario. RUBI permite a los nodos tomar decisiones locales respecto a la elección del enfoque a adoptar en un momento dado. RUBI posee la habilidad de adaptarse; esto significa que es resistente a fallos de una amplia gama de condiciones de red en entornos ubicuos. RUBI está diseñado bajo el enfoque Cross-layer, es decir, opera sobre la capa de red o por encima de la capa de transporte del modelo de referencia OSI. Por lo tanto, RUBI se basa en los protocolos de encaminamiento, puede proveer esta funcionalidad en caso de ser necesario [69].

2.2.4. Protocolos con arquitectura de directorio distribuida sin infraestructura

Este tipo de arquitectura no requieren de la existencia previa de facilidades e instalaciones como requieren las redes fijas, si alguna infraestructura de red fija se incluye en esta arquitectura, se está en contra de la naturaleza de las redes Ad Hoc. En otras palabras, las redes móviles Ad Hoc se construyen espontáneamente, son redes en las que no existe diferencia entre nodos, routers, pasarelas y servidores, sino que cada elemento presente en la red enruta sus propios mensajes. La arquitectura de directorio distribuida sin infraestructura es bastante idónea para el escenario de redes Ad Hoc, ya que los directorios son dinámicamente seleccionados del total nodos móviles que posean las capacidades adecuadas (batería, memoria, cómputo, cobertura, etc.), no requiere una infraestructura predefinida [33].

Aunque esta arquitectura es diseñada principalmente para redes móviles Ad Hoc puras, también puede usarse en redes híbridas usando el concepto de "*gateway*". El *gateway* reside en el límite de una red móvil Ad Hoc y una red fija y actúa como un directorio.

A continuación incluimos ejemplos representativos de protocolos con arquitectura de directorio distribuida sin infraestructura.

2.2.4.1. SSD

SSD. Descubrimiento de servicios escalable para MANET. En SSD los directorios están distribuidos y desplegados dinámicamente con la finalidad de lograr mayor escalabilidad. Se encuentra estructurada como una red virtual, compuesta por un subconjunto de nodos de la MANET que actúan como directorios. Estos directorios representan un backbone de nodos responsables de llevar a cabo el descubrimiento de servicios y están desplegados de tal forma que al menos un directorio sea descubierto por la mayoría de nodos, a través de un número fijo de saltos. Los directorios almacenan en caché las descripciones de servicios disponibles en su vecindad. Un cliente envía una consulta al directorio local, sino existe respuesta en el directorio local, el directorio reenvía selectivamente la consulta a otros directorios y de esta forma ejecuta un descubrimiento global. La selección de directorios a la que son reenviadas las consultas de servicios se basa en el intercambio de perfiles entre directorios. El perfil de directorio ofrece un resumen compacto del contenido del directorio y la capacidad del host. SSD usa filtros Bloom como método de resumen de contenido de directorio [138].

2.2.4.2. DSDP

DSDP. Distributed Service Discovery Protocol, arquitectura que se basa en un backbone virtual para localizar y registrar servicios disponibles dentro de una red con topología dinámica. DSDP consiste de dos partes independientes: fase de gestión del backbone (BBM) y fase de descubrimiento de servicios distribuidos (DSD). La fase BBM selecciona un subconjunto de los nodos de la

red para formar un conjunto dominante relativamente estable, descubre la ruta entre los nodos dominantes y se adapta a los cambios de topología, añadiendo o eliminando nodos de este conjunto dominante. BBM utiliza sólo un mensaje de control de difusión local de 1-salto para formar el backbone, crear enlaces virtuales entre los nodos del backbone y mantener el backbone. Una vez terminada exitosamente la fase BBM, se crea un backbone virtual sobre una estructura de malla con los nodos del backbone y los enlaces virtuales conectados a ellos. La fase DSD se usa para distribuir los mensajes de petición y registro de clientes y servidores a los directorios (nodos backbone). Estos mensajes se presentan en forma de árboles multicast arraigados a los nodos del cliente y del servidor, sobre la parte superior de la malla del backbone [86].

2.3. Perspectiva histórica de los protocolos de descubrimiento de servicios

Esta sección describe un análisis cronológico de los protocolos de descubrimiento de servicios. El análisis comienza desde 1999 (año cuando se propuso el primer protocolo) y termina en el año 2005. El análisis considera para el estudio cronológico cuatro elementos: el tipo de directorio, la arquitectura, el tipo de red y la técnica de descubrimiento de servicios. Estos elementos han sido claves en la evolución de los protocolos de descubrimiento de servicios. Gran parte de la información analizada en esta sección se encuentra resumida en el cuadro 2.1.

El gráfico 2.2 muestra el año cuando se presenta el tipo de directorio. Primero, en 1999 se definieron cinco tipos de directorios: uno *basado en infraestructura* y cuatro *basados en directorio*. En el año 2000 solamente se definió una infraestructura de descubrimiento de servicios y fue una *sin directorio*. En 2001 se propusieron dos tipos de directorios: uno de ellos fue *sin directorio* y otro fue *basado en infraestructura*. El año 2002 ha sido el año más productivo y en este sentido se definieron nueve tipos de directorios: cuatro *basados en infraestructura*, tres *sin directorio* y dos *basados en directorio*. En 2003 tomaron parte siete

tipos de directorio, los *basados en infraestructura* se propusieron cinco veces, *sin directorio* una vez y *basado en directorio* una vez. En 2004 fue propuesta la arquitectura *sin infraestructura* como una única arquitectura. Finalmente, en 2005 se definieron dos tipos de directorios: uno *basado en infraestructura* y uno *sin infraestructura*.

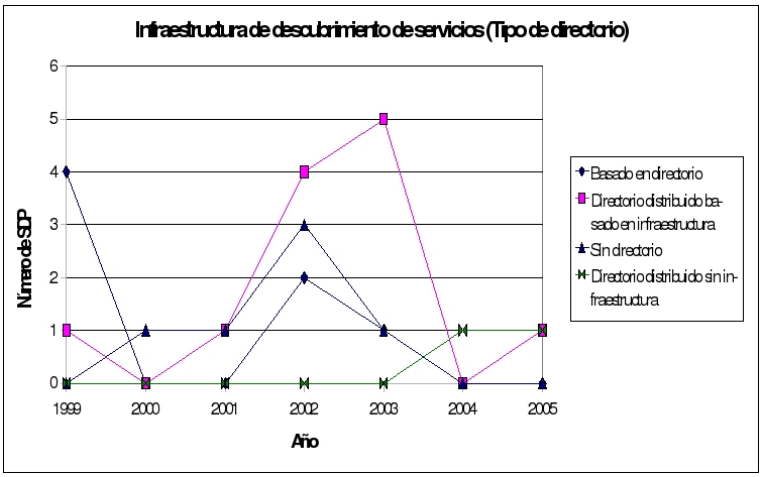


Figura 2.2: Línea de tiempo del tipo de directorio.

El gráfico 2.3 muestra el año cuando aparecieron las arquitecturas de búsqueda de servicios. Primeramente, en 1999 se definieron ocho arquitecturas: dos *cliente-servidor*, dos *centralizadas*, una de *área amplia* y tres de P2P. En el año 2000 se propuso únicamente la arquitectura *P2P*. En 2001 se propusieron dos arquitecturas *P2P*. El año 2002 fue el mejor año, ya que se definieron doce arquitecturas: dos *centralizadas*, ocho *P2P* y dos de *área amplia*. En el 2003 se definieron siete arquitecturas: una *centralizada*, dos *P2P*, una de *área amplia*, dos de *estructura overlay* y una de tipo *cliente-servicio-directorio-proxy*. En el año 2004 se propuso únicamente la arquitectura de *backbone virtual*. Finalmente, en el año 2005 se plantearon dos arquitecturas: una de *área amplia* y otra de *backbone virtual*.

El gráfico 2.4 muestra el año cuando fue propuesto el tipo de red para descubrir servicios. En el año 1999 se definieron seis tipos de redes: una red de tipo *móvil Ad Hoc*, una de tipo *subred*, una de *cualquier tipo*, una de tipo

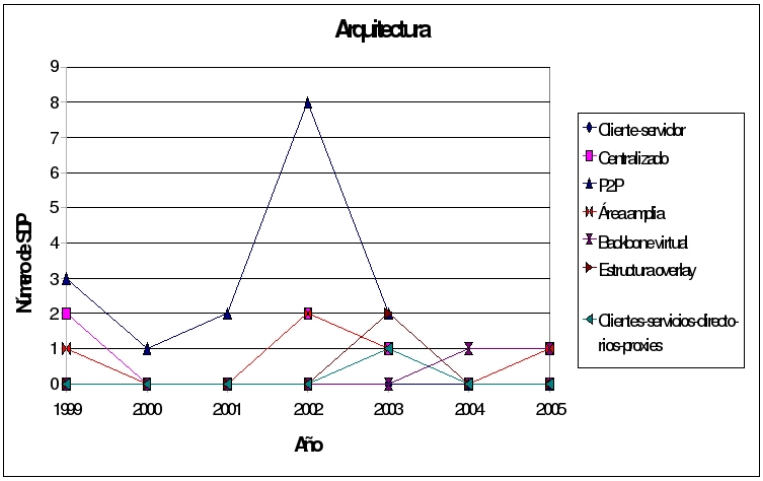


Figura 2.3: Línea de tiempo de la arquitectura.

LAN, una de tipo *WAN* y una red de *estructura jerárquica*. En el 2000 sólo fue propuesta una red de tipo *subred*. En el año 2001 la red de tipo *móvil Ad Hoc* y la *red virtual overlay* fueron las únicas que se propusieron. Después, el año 2002 vuelve a ser el mejor año, en ese año se proponen diez tipos de redes: tres *redes móviles Ad Hoc*, dos *redes virtuales overlay*, una de *anillo universal*, una de tipo *subred*, dos de tipo *empresarial* y una de *estructura de árbol jerárquica*. En el 2003 se proponen siete tipos de redes: cinco *redes móviles Ad Hoc*, una *red virtual overlay* y una de tipo *subred*. Posteriormente, en 2004 se define la *red móvil Ad Hoc* como único tipo de red. Finalmente, en 2005 se proponen solamente dos redes de tipo *móvil Ad Hoc*.

El gráfico 2.5 muestra el año cuando se presentan las técnicas de descubrimiento de servicios como métodos de búsquedas. En el año 1999 se definen cinco técnicas de descubrimiento de servicios: cuatro técnicas de tipo *directorio centralizado* y una de tipo *inundación de peticiones*. La técnica *inundación de peticiones* fue la única propuesta en el año 2000. Después, en el año 2001 se presentan dos técnicas de tipo *inundación de peticiones*. Mientras que 2002 sigue siendo el año más productivo, se definen nueve técnicas de descubrimiento de servicios: cuatro por medio de *inundación de peticiones*, dos por medio de

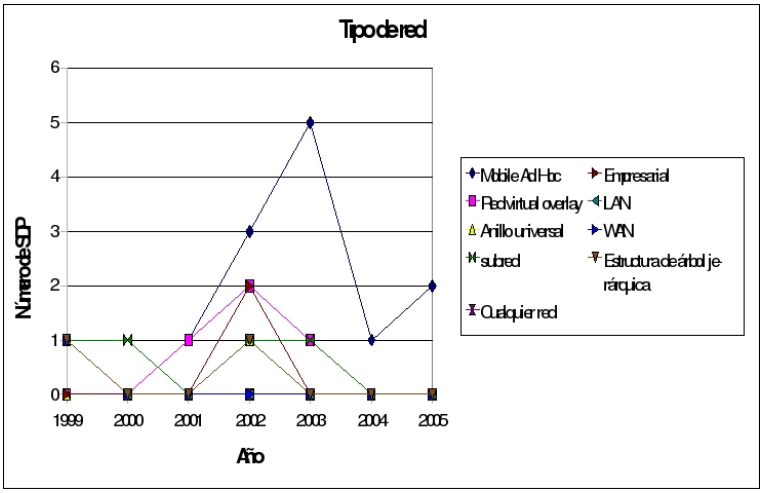


Figura 2.4: Línea de tiempo del tipo de red.

esquemas de *encaminamiento semántico* y uno por medio de *tabla Hash*. En el año 2003 se proponen siete técnicas de descubrimiento de servicios: *directorio centralizado* ha sido el más usado (tres veces), dos veces el *encaminamiento semántico*, una vez la *inundación de peticiones* y una vez la *tabla Hash*. En 2004 *inundación de peticiones* es la única técnica propuesta. Y finalmente, en el año 2005 se definen tres técnicas de descubrimiento de servicios: una vez la *inundación de peticiones*, una vez la *tabla Hash* y una vez el *encaminamiento semántico*.

A la vista de los resultados del análisis cronológico anterior, es de suponer que algunos tipos de directorios, arquitecturas, tipos de red y métodos de búsquedas han sido más populares que otros. Ciertos mecanismos han sido repetidamente aplicados cada año, independientemente que nuevos mecanismos habían sido ya propuestos antes y aún ahora se continúan poniendo en práctica.

Respecto al tipo de directorio, por ejemplo, en el año 2000 UPnP propuso *sin directorio*, años más tarde otros protocolos propusieron nuevamente el antiguo esquema *basado en directorio*. El hecho es que la *infraestructura de directorio distribuida basada en infraestructura* fue propuesta por doce tipos de directorios, casi en todos los años. Luego, el modelo *basado en directorio* fue definido en siete

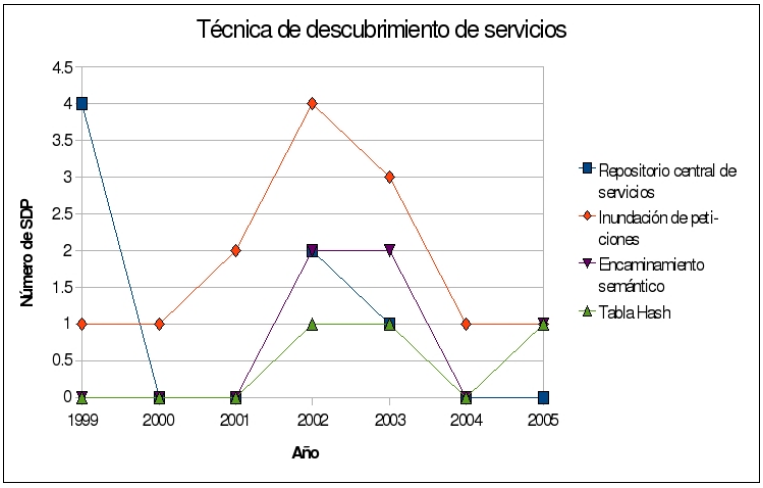


Figura 2.5: Línea de tiempo de la técnica de descubrimiento de servicios.

veces, el modelo *sin directorio* seis veces y el modelo de *directorio distribuido sin infraestructura* fue el más comúnmente presentado. Este último modelo es adecuado para MANET conectadas a redes fijas con infraestructura y podría ser válido para casos cuando un nodo de una MANET requiere localizar un servicio ofrecido por otro nodo de una MANET conectada a través de una red fija. Las redes fijas tienen infraestructura para mantener directorios y mucho de la sobrecarga se soporta por servidores con altas prestaciones conectadas a redes con gran ancho de banda. En redes con infraestructura la sobrecarga de memoria y cómputo es considerablemente grande. Por esta razón, los protocolos de descubrimiento de servicios basados en el modelo de *directorio distribuido con infraestructura* son capaces de soportar más sobrecargas que el modelo de *directorio distribuido sin infraestructura*.

En cuanto a las arquitecturas, *Peer-to-peer* (P2P) se ha puesto en práctica en doce oportunidades, *centralizadas* fue usada por cinco arquitecturas, *área amplia* fue propuesta cinco veces, *cliente-servidor* dos veces, *backbone virtual* dos veces, *estructura overlay* dos veces y *clientes-servicios-directorios-proxies* una vez. Por consiguiente, *P2P* ha tenido más frecuencia de uso en los modelos de arquitecturas.

Es bueno que *P2P* haya sido definido como arquitectura por la mayoría de los protocolos. En el modo P2P los nodos descubren los servicios directamente y no a través de un nodo intermediario. Este hecho tiene la ventaja de que cada nodo puede intercambiar información con otro directamente, en el modo *P2P* no existe la noción de clientes o servidores como en otros modelos, sino de igual a igual, lo cual es idóneo en las MANET.

En el tema de tipo de red, la *red móvil Ad Hoc* se presentó trece veces, *red virtual overlay* cuatro veces, *subred* cuatro veces, *empresarial* dos veces, *estructura de árbol jerárquica* dos veces, *anillo universal* una vez, cualquier red una vez, *LAN* una vez y *WAN* una vez. La red móvil Ad Hoc fue el tipo de red con más popularidad en los entornos móviles Ad Hoc (como era de esperar). Sin embargo, como se ha visto, otros enfoques también fueron propuestos. Si trabajamos con MANET entonces los más apropiado es aplicar una tecnología de red móvil Ad Hoc como tipo de red. Como sabemos, en una MANET cualquier nodo puede actuar como servidor y proveer sus servicios a otros nodos; también es cierto que cualquier nodo puede actuar como cliente y usar un protocolo de descubrimiento de servicios para buscar los servicios disponibles a su alrededor. Por consiguiente, una MANET requiere estrategias y mecanismos adecuados para lograrlo.

Finalmente, en cuanto a las técnicas de descubrimiento de servicios, *inundación de peticiones* fue propuesto por trece protocolos, mientras que el *repositorio central de servicios* fue propuesto siete veces, *encaminamiento semántico* cinco veces y *tabla Hash* tres veces. Como se puede notar, el mecanismo de difusión a través de *inundación de peticiones* ha sido la técnica de descubrimiento de servicios que obtuvo la mayoría. *Inundación de peticiones* no es muy recomendable porque inunda la red redundantemente con sus evidentes resultados de altos costos. Las operaciones de inundación redundante mediante el anuncio y petición de servicios puede imponer serias deficiencias en las MANET, ya que incrementa el consumo de recursos. El descubrimiento de servicios por inundación no es adecuado para MANET debido a su alto consumo de memoria, energía

y ancho de banda, muy limitados en los dispositivos móviles. A no ser que se propongan algoritmos de *inundación de peticiones* que reduzcan el tráfico en la red.

En resumen, el *directorio distribuido con infraestructura*, el modelo *P2P*, la *red móvil Ad Hoc* y el descubrimiento de servicios por *inundación de peticiones* son los enfoques más utilizados en entornos móviles Ad Hoc. Sin embargo, no siempre son los más idóneos para tener éxito en el descubrimiento de servicios en computación ubicua. Primero, el *directorio distribuido con infraestructura* no es muy apropiado para redes móviles Ad Hoc. En cambio, el *directorio distribuido sin infraestructura* debería ser el más apropiado para estos casos. Segundo, el descubrimiento de servicios por *inundación de peticiones* no es recomendable, ya que satura la red enviando gran cantidad de mensajes e incrementa significativamente el uso de recursos en los dispositivos móviles.

2.4. Comparativa entre SDP

Clasificando los protocolos por tipo de infraestructura, podemos constatar que fueron apareciendo cronológicamente; primeramente aparecieron las arquitecturas de directorio centralizadas; luego, las arquitecturas de directorio distribuidas basadas en infraestructura; posteriormente surgieron las arquitecturas sin directorio; y finalmente, hicieron su aparición las arquitecturas de directorio distribuidas sin infraestructura.

Analizando objetivamente nos damos cuenta de que cada arquitectura pretende ser una evolución de la anterior, el objetivo final es lograr una arquitectura capaz de operar en una red móvil Ad Hoc, de naturaleza heterogénea, con gran número de nodos, múlti-salto, altamente dinámica y sin infraestructura. Al parecer, esta última característica es lo que hace deseable que la arquitectura de directorio distribuida sin infraestructura sea la más idónea para escenarios de redes móviles Ad Hoc.

Es fácil inferir que en las arquitecturas de directorio centralizadas no existe garantía de que un servicio se encuentre disponible en la red, aun cuando el

servicio se encuentre registrado en el directorio centralizado o por el contrario, cuando el servicio no se encuentre registrado en el directorio de servicios y éste sí se encuentre disponible en la red. Como se trata de entornos dinámicos en los que los dispositivos móviles entran y salen de la red espontáneamente, bajo estas restricciones es muy difícil localizar servicios sobre un dispositivo centralizado funcionando como gestor de direcciones de servicios, debido a que no existe forma de encontrar otros servicios cuando ese directorio no esté disponible. Por otra parte, la naturaleza móvil de las MANET impiden mantener datos frescos o renovados acerca de los nodos presentes en la red. Un directorio centralizado limita el alcance de los dispositivos dentro de un dominio local de descubrimiento de servicios. Las fronteras de un dominio de descubrimiento de servicios son definidas administrativamente o como resultado de las propiedades físicas o del alcance de la señal de radio de la red inalámbrica.

Las arquitecturas sin directorio emplean un esquema de difusión periódico (DEAPspace) que origina el problema de tormenta de multicast, otras veces se sustituye este continuo reparto innecesario por multicast en períodos de tiempos aleatorios (Konark). Otro protocolo soporta la auto-configuración de la red (UPnP), pero no soporta caché con información de servicios. Otros protocolos mantienen una caché de servicios para reducir el descubrimiento de servicios innecesarios cuando un servicio se encuentra previamente ya anunciado (Konark y PDP). Otros protocolos han propuesto resolver el problema de las limitaciones de los dispositivos, reduciendo el número de mensajes enviados a través de la red, mediante la mezcla de características de soluciones pull y push (PDP). Otros protocolos han adoptado la estrategia de enviar inteligentemente solicitudes de servicios basados en grupos de nodos (ALLIA y GSD) para reducir el tráfico global.

En las arquitecturas de directorio distribuido sin infraestructura, la asignación dinámica representa una carga extra para la red en términos de selección de directorios, así como en informar al resto de la red sobre las identidades de los directorios, lo anterior debido a los cambios de topología. Sin embargo, en

comparación con las arquitecturas sin directorio, existen más ventajas inherentes en usar directorios. Primero que nada, la escalabilidad, que se logra cuando el tamaño de la red aumenta. Segundo, se reduce el tiempo de respuesta para localizar servicios. Y tercero, los directorios aplican técnicas de balanceo para reducir la carga de trabajo en los directorios individuales y mejorar el rendimiento de los servicios.

La arquitectura de directorio distribuido sin infraestructura es idónea para escenarios móviles Ad Hoc. Los directorios son dinámicamente seleccionados de entre nodos móviles con excelentes capacidades (batería, memoria, capacidad de procesamiento, cobertura, etc.), esta arquitectura no requiere una infraestructura predefinida. En cuanto a las arquitecturas de directorio distribuidas basadas en infraestructura, se consideran ciertos costos y consumos. En pequeñas redes inalámbricas el costo de comunicación tiene especial interés, pero no así en redes de áreas amplias, en las que gran cantidad del costo es absorbido por las capacidades de los servidores conectados a redes con gran ancho de banda. En estos entornos de área amplia, los costos de computación y gastos de memoria son a veces más importantes que los costos de comunicación. En consecuencia, los protocolos de descubrimiento de servicios basados en una arquitectura de directorio distribuida con infraestructura, han sido especialmente diseñados para reducir costos computacionales y consumo de memoria.

En VIA, el directorio del nivel superior recibe todas las consultas y sólo delega las consultas más relevantes a sus hijos. Por tanto, para la resolución de consultas se involucra solamente un subconjunto de directorios del sistema. En Superstring se distribuye la descripción simple de servicios sobre varios directorios. De este modo, cada directorio almacena sólo una pequeña parte de toda la descripción y así ningún directorio procesa la consulta por completo.

2.5. Análisis del funcionamiento de los SDP

El descubrimiento de servicios en sistemas distribuidos es diseñado principalmente bajo estos cuatro tipos de enfoques: repositorio centralizado de servi-

cios, inundación de mensajes, basado en hashing y encaminamiento semántico [82]. En la práctica cada protocolo de descubrimiento emplea alguna técnica en particular o a través de un tipo de combinación de técnicas.

En términos generales las redes Ad Hoc se comportan de la siguiente manera. Cada dispositivo actúa como router que debe administrar tablas de encaminamiento. Los retos para las redes Ad Hoc se interesan en la inestabilidad de la red, como son entre otras cosas la movilidad y los fallos de los nodos, los cachés inexistentes y en las restricciones de recursos de energía y memoria de los dispositivos. Algunas soluciones propuestas cuentan con un protocolo de encaminamiento para lograr tanto el reenvío de datos como el descubrimiento de servicios [96]. Habitualmente también realizan un amplio uso de mecanismos de difusión a través de broadcast y multicast.

El descubrimiento de servicios permite a los dispositivos y servicios, descubrir, configurar y comunicarse los unos con los otros. Desafortunadamente se pierde valioso tiempo cuando de forma activa se buscan servicios y se configuran manualmente los dispositivos y programas. Algunas veces se requieren habilidades especiales que nada tienen que ver con las tareas que se desean realizar. Los protocolos han sido diseñados para minimizar la carga administrativa e incrementar la usabilidad. Para facilitar también a los diseñadores de sistemas cuando intentan prever y codificar todas las posibles interacciones y estados entre dispositivos y programas [163].

De todo lo que se puede desear que un entorno ubicuo y una red Ad Hoc puedan tener, es que los dispositivos electrónicos ofrezcan cierta colaboración y facilidades para la interacción humano-máquina. Entre la serie de interfaces, aplicaciones y amigabilidad que los dispositivos puedan tener, incluimos esta lista con las propiedades ideales deseables que los dispositivos deban ser capaces de hacer.

- Habilidad de anunciar su presencia a la red.
- Descubrimiento automático de los dispositivos en la red.

- Habilidad de describir sus capacidades y entender las capacidades del resto de dispositivos.
- Auto-configuración sin la intervención administrativa de nadie.
- Perfecta interoperabilidad con otros dispositivos.

Lo anterior enumera el comportamiento esperado de los dispositivos, pero la oferta actual de protocolos o arquitecturas para el descubrimiento de servicios en redes Ad Hoc dista mucho de lo que se espera de ellos.

El descubrimiento de servicios para sistemas distribuidos por lo regular ha sido diseñado para redes basadas en Internet, de modo que no tienen en cuenta las características de las redes móviles Ad Hoc [82].

2.6. Retos

En el tema de redes móviles Ad Hoc existe una extensa lista de retos a enumerar, principalmente en el tópico de descubrimiento de servicios, además de los retos en materia de seguridad, calidad de servicio y encaminamiento, que son temas aparte. El problema central que enfrentan las redes Ad Hoc es la movilidad. La movilidad a su vez desencadena una serie de problemas como pueden ser la escalabilidad y la tolerancia a fallos. Sin embargo, mencionaremos algunos retos muy puntuales que aún no resuelven por completo la mayoría de los SDP.

- *Límites de la red*, existen protocolos que soportan el descubrimiento de servicios en entornos móviles en términos de topología de red o ubicación. Cada protocolo maneja una mezcla distinta de características, pero la mayoría han sido diseñados para entornos dirigidos hacia hogares o empresas (indoors), por lo que el cómputo móvil en estos casos se circunscribe localmente y no rebasa estos márgenes. Es necesario definir entonces si se extienden o reducen estos límites; extender otorga al entorno un carácter

público y abierto, mientras que reducir equivale a cerrar el entorno a pocos nodos o a un área geográfica muy reducida.

- *Integración*, la heterogeneidad y dinámica de entornos locales de una compañía no es equiparable a los entornos ubicuos. Los servicios en redes empresariales operan dentro de un rango de red protegido por firewalls y manejados por personal encargado de la gestión de sistemas. Pero en un rango o ámbito de red es imposible definir fácilmente los servicios ambientales de ese entorno, ni es posible manejar todo a través de administradores de sistemas [163]. El descubrimiento de servicios en una oficina podría tratarse de un servicio inherente al dominio de la propia empresa, pero podría también ir dirigido hacia servicios de asuntos personales entre empleados. En resumen, es difícil distinguir el tipo de servicios que se trata. Por tanto, se plantean cuestiones relativas a la integración del usuario y su entorno. Un reto verdaderamente serio es la integración de dispositivos de cómputo con el usuario. Aunque muchos protocolos consideran sólo la interacción entre dispositivos y programas que actúan como clientes, servicios y directorios.

- *Inconsistencia e incertidumbre*, dada la naturaleza dinámica de los entornos móviles, la topología de red cambia constantemente en el momento en que los dispositivos se conectan y desconectan de la red, se desplazan hacia otro sitio, o simplemente puede ocurrir un fallo. Por tanto, en los dispositivos también pueden ocurrir algunos cambios y en cualquier momento dado, las consultas de descubrimiento de servicios y de estados de los servicios pueden llegar a ser inconsistentes e inciertos, respectivamente. Naturalmente causando inestabilidad en la red. Se requiere un mecanismo eficiente para monitorizar esta situación. Algunos protocolos de descubrimiento de servicios utilizan esquemas de administración de estados basados en tiempo para controlar la disponibilidad de los servicios (meca-

nismos soft state, hard state y hybrid state). Sin embargo, aun con esto se desconoce lo que pudiera suceder en cualquier momento.

- La *escalabilidad* es un problema que exige ser atendido. Un ejemplo es el entorno público o abierto, compuesto por distintos dispositivos heterogéneos formando espontáneamente una red, con el objetivo de ofrecer y descubrir la gran variedad de servicios disponibles. Por tanto, es apremiante desarrollar nuevos protocolos que soporten entornos con gran número de nodos.

- *Heterogeneidad*, que existe en muchos aspectos: hardware, software, protocolos de redes y proveedores de servicios. Suponemos que a corto plazo se continúe incrementando esta diversidad. Aunque los SDP se adaptan a la heterogeneidad, existen protocolos que tienen varios objetivos y soluciones de diseño. Cada uno tiene sus ventajas y desventajas para diferentes situaciones, parece poco probable que un único protocolo pueda dominar todos los entornos de computación ubicua. En los protocolos actuales, los clientes y servicios sólo se descubren unos a otros a menos que usen un protocolo en común. Se precisa establecer una plataforma compatible que permita la interoperabilidad entre SDP.

- La *interoperabilidad*, entre protocolos de descubrimiento de servicios debe existir soporte para colaborar entre dispositivos, que se permitan descubrir a través de redes heterogéneas que ejecutan distintos protocolos.

- Respecto al *resource awareness*, la delegación de carga de trabajo a dispositivos con capacidades superiores podría no ser una solución cuando estos dispositivos sean escasos. Existe la necesidad de desarrollar protocolos ligeros que sean capaces de ejecutarse en dispositivos limitados de recursos.

- La *seguridad*, tema que pocos protocolos de descubrimiento de servicios han abordado, porque es complejo y existen temas prioritarios a resolver,

como es el propio tema de descubrimiento de servicios. La seguridad debe ser incluida desde el comienzo de la etapa de diseño de cualquier protocolo de descubrimiento de servicios.

Independientemente de los retos planteados anteriormente, en uno de los trabajos más recientes [47] sobre sistemas de descubrimiento de servicios, se plantean las siguientes líneas de investigación:

- *Trabajar con infraestructura sin la infraestructura*, un reto clave es contar con las propiedades deseables de las infraestructuras fijas, como escalabilidad, seguridad, fácil administración, compartir nombres, sin tener una infraestructura en sí.

- *Conectar islas*, para llevar a cabo el descubrimiento de servicios, la tecnología actual presenta una red segmentada múlti-transporte de múltiples islas aisladas, ya que los diferentes sub-estratos de redes imponen necesariamente sus propias semánticas y restricciones, e inherentemente definen de forma diferente el alcance de descubrimiento. Esto obliga a los usuarios a entender el funcionamiento del alcance usado por los diferentes protocolos. Para conseguir una perfecta inter-conectividad se deben encontrar formas de conectar los requerimientos de descubrimiento de transportes individuales, de forma que se requiera poca administración y olvidarse de que el usuario final entienda los conceptos técnicos básicos de redes.

- *Búsqueda apropiada*, es decir, la forma de cómo los clientes buscan los recursos o servicios. Los mecanismos de búsqueda actuales se basan mayormente sobre atributos relativamente estáticos que los recursos publican acerca de ellos mismos. Estos mecanismos de búsquedas son justificados dada la intención de uso de los distintos protocolos de descubrimiento.

- *Aspecto humano*, aun cuando el descubrimiento es un mecanismo técnico, tiene implicaciones para los usuarios. Los protocolos de descubrimiento deben proveer suficiente información y proveer un modelo suficientemente

coherente para que los usuarios puedan entender el proceso de descubrimiento. Aspectos técnicos como nombrar y encontrar recursos tienen grandes ramificaciones sobre cómo los usuarios van a percibir y a trabajar con sistemas basados en descubrimiento. Sin embargo, pocos trabajos han examinado las implicaciones del usuario final de las características técnicas en el descubrimiento. Es necesario entender cómo estos efectos sociales podrían influenciar el diseño de protocolos de descubrimiento.

El descubrimiento es simplemente el primer paso de una interacción de aplicación con algún recurso. Lo siguiente es aún un paso más importante para la aplicación que emplea ese recurso. La mayoría de los protocolos de descubrimiento de servicios ofrecen aplicaciones para el manejo de un recurso y nada más. Pero aunque si dos partes están de acuerdo en una plataforma de descubrimiento, esto no garantiza que sean capaces de trabajar juntos. Deben estar de acuerdo en las operaciones y formato de datos al que darán soporte, así como la semántica a emplear. Esto no es un problema de descubrimiento en sí. Varios proyectos de investigación han comenzado a proponer enfoques más flexibles para la interoperabilidad, entre ellos CoolTown [81], iRoom [79] y Speakeasy [48].

2.7. Protocolos de encaminamiento en MANET

A continuación abordaremos la temática de los protocolos de encaminamiento especialmente diseñados para redes móviles Ad Hoc. Estos protocolos se incluyen en este capítulo para contextualizar el protocolo de encaminamiento reactivo que emplearemos en nuestra propuesta de solución.

Para las redes convencionales (fijas y cableadas) existe un gran número de protocolos de encaminamiento, pero estas técnicas no son de aplicación en las MANET. Los algoritmos usados en las redes cableadas suponen que la topología de la red es estable y poco cambiante, a diferencia de las MANET en las que la topología cambia continuamente debido a que sus nodos observan una alta

movilidad. En las redes convencionales los protocolos de encaminamiento pueden ser de dos tipos: *salto a salto* (hop by hop routing), en este tipo de protocolo cada nodo decide el siguiente punto a donde enviará el paquete, de acuerdo a la información previa que tiene en sus propias tablas; el encaminamiento salto a salto puede ser por *estado del enlace*[1] o por *vector de distancias*[2]. El otro tipo de protocolo es *en origen* (source routing), aquí el responsable del encaminamiento es el nodo que origina el envío, ya que cada paquete almacena la ruta completa por la que ha de pasar hasta llegar al destino.

Los protocolos de encaminamiento orientados a las MANET tienen la principal característica de conservar activa la conectividad entre el nodo origen y el nodo destino durante los desplazamientos y cambios velocidad. De modo que, cuando estos nodos no se encuentren directamente conectados entre sí, la comunicación se mantiene a través del reenvío de paquetes entre los nodos intermedios.

Otro aspecto a considerar, es que los limitados recursos muy característicos de los dispositivos de las MANET muy pronto saturan la red, por un lado debido al denso tráfico de mensajes de control que genera este tipo de algoritmos y por el otro, al creciente tamaño de las tablas de encaminamiento.

Teniendo en cuenta los aspectos antes citados, se han diseñado protocolos de encaminamiento específicos para redes inalámbricas.

Los protocolos de encaminamiento apropiados para redes inalámbricas con topologías tanto estáticas como dinámicas pueden dividirse en dos tipos: protocolos proactivos (PMP, Proactive Manet Protocol) y protocolos reactivos (RMP, Reactive Manet Protocol) [73] y en un tercero como derivación de los primeros dos, los protocolos híbridos.

[1]Estado del enlace (OSPF, Open Shortest Path First). En este protocolo cada nodo conoce el estado general de la red y calcula la ruta óptima para llegar al destino.

[2]Vector de distancias (DV, Distance Vector), en el que cada nodo conoce únicamente el primer salto de la ruta y la distancia que resta para llegar el destino.

2.7.1. Protocolos proactivos

En este primer tipo de protocolos, las rutas a todos los posibles destinos se generan con anticipación y en todo momento se conserva una información suficientemente actualizada de las tablas de encaminamiento, para tal efecto, los protocolos proactivos emplean mensajes periódicos de actualización. En este sentido, los protocolos proactivos suponen una sobrecarga, puesto que aunque no exista transmisión de datos, sí se requiere la señalización. Sin embargo, el beneficio de esta sobrecarga es que se mantiene en cada instante una ruta actualizada hacia cualquier destino. La dinamicidad de la red implica un intercambio de mensajes entre los nodos para actualizar las tablas de encaminamiento y por tanto una sobrecarga, pero por otra parte, se reduce el retardo extremo a extremo desde antes de enviar un paquete, ya que un nodo siempre tiene una ruta disponible hacia un destino. Existe una latencia mínima pero se tienen rutas válidas y vigentes en cada momento.

Algunos de los protocolos proactivos más importantes son:

- DSDV (Destination Sequenced Distance Vector) [123], en este protocolo, los nodos vecinos intercambian periódicamente sus tablas de encaminamiento con el objetivo de calcular la distancia para alcanzar a los demás nodos no vecinos. DSDV obtiene las rutas sin ciclos mediante la introducción de números de secuencia para determinar las rutas más nuevas. DSDV proporciona sólo un camino hacia cada destino, siempre elige el camino más corto basándose en el número de saltos hacia este destino.

  DSDV utiliza dos tipos de mensajes de actualización, uno llamado *full-dump* y otro llamado *incremental*. Los mensajes incrementales se emplean para actualizaciones intermedias entre envíos periódicos full-dump de la tabla de encaminamiento.

- WRP (Wireless Routing Protocol) [103], es un protocolo basado en tablas. En WRP cada nodo es responsable de mantener cuatro tablas: tabla de distancias, tabla de encaminamiento, tabla de coste de ruta y tabla de lista

de mensajes retransmitidos. La tabla de lista de mensajes retransmitidos (MRL) gestiona el envío de los paquetes de actualización de rutas. Cada entrada de la MRL contiene el número de secuencia que identifica el paquete de actualización de rutas, un contador de retransmisiones, un vector de ACK (acuse de recibo) con una entrada por vecino y una lista de las unidades enviadas en el paquete de actualización.

Los cambios en las rutas se informan entre los nodos mediante el envío de paquetes de actualización. Los paquetes de actualización se envían sólo entre vecinos y contienen los elementos que se han de actualizar en las rutas. Los nodos envían estos paquetes cuando procesan las actualizaciones recibidas de otros vecinos o cuando ellos mismos detectan un cambio en el enlace con algún vecino.

Los nodos mantienen activos los enlaces con sus vecinos, siempre y cuando reciban ACKs de ellos. Si un nodo no envía mensajes, entonces debe enviar a sus vecinos un paquete HELLO cada cierto tiempo, a fin de que sus vecinos no supongan que el nodo es inalcanzable. Por lo tanto, si un nodo omite enviar mensajes, entonces ocasiona una ruptura en el enlace.

- CSGR (Cluster Switch Gateway Routing) [31] [137], este protocolo agrupa los nodos de la red en clusters y cada cluster contiene un líder. CSGR usa DSDV como algoritmo base de encaminamiento, pero modificado para utilizar la jerarquía introducida de los líderes y los clusters. Al crear estos clusters se establece una jerarquización en los que cada líder controla su propio cluster.

  El clusterhead (líder) se elige mediante un algoritmo distribuido, y cuando un clusterhead se mueve fuera del grupo, entonces se elige un nuevo clusterhead. Un problema que se puede presentar es que cuando un clusterhead se mueve con demasiada frecuencia, los nodos gastan mucho tiempo y recursos eligiendo al nuevo clusterhead.

Para encaminar un paquete lo primero que hace un nodo es encontrar al clusterhead más cercano y a la ruta hacia el destino conforme a la tabla de miembros del cluster y la tabla de encaminamiento. Posteriormente, debe consultar la tabla de encaminamiento para encontrar el siguiente salto para alcanzar el clusterhead seleccionado anteriormente y finalmente, transmitir el paquete al nodo.

- STAR (Source Tree Adaptive Routing) [58], el protocolo de encaminamiento de árbol en origen adaptativo fue el primer protocolo de encaminamiento proactivo que trabaja con información de estado de enlace y fue el más rápido de los protocolos bajo demanda. STAR no toma el camino más corto para mantener mensajes de control bajos. STAR identifica cada nodo con una dirección fija. Una ventaja de este protocolo es que no requiere de actualizar periódicamente las rutas.

  En STAR, un árbol en origen contiene los enlaces a cada vecino. El siguiente paso es actualizar, STAR envía su propio árbol en origen inmediatamente que se actualizan los otros vecinos. De esta manera, cada nodo puede ser creado con su propio árbol en origen y con los árboles recibidos.

- OLSR (Optimized Link State Routing) [74], es un protocolo de encaminamiento punto a punto que se basa en el algoritmo de estado de enlace. En este algoritmo todos los nodos intercambian mensajes entre sí con el objetivo de formar una visión consistente de toda la red y de esta manera decidir el encaminamiento de paquetes.

  OLSR al igual que DSDV, intercambian un gran número de mensajes periódicos. Además del envío de mensajes HELLO, también se envían mensajes de control TC (Topology Control) que se retransmiten a todos los nodos de la red. Sin embargo, con la incorporación de la técnica de retransmisión multipunto, se ha conseguido una gran optimización en la retransmisión de estos mensajes, con esto, los mensajes sólo se retransmiten por el mínimo número de nodos necesarios para alcanzar a todos

los demás. Estos nodos vecinos seleccionados son los únicos encargados de retransmitir los paquetes de encaminamiento y se les denominan Relays Muntipunto (Retransmisores Multipunto). El resto de nodos vecinos procesan los paquetes de encaminamiento que reciben, pero no los pueden retransmitir.

- TBRPF (Topology dissemination Based on Reverse-Path Forwarding) [113], es un protocolo de encaminamiento de estado de enlace basado en el cálculo de un árbol, en el que su raíz es un nodo origen que genera caminos a todos los nodos que puede alcanzar. En caso de no existir cambios en la red, entonces se minimiza la sobrecarga. El protocolo TBRPF utiliza el encaminamiento proactivo, en el que todas las rutas a todos los posibles destinos se calculan a priori; estas rutas se mantienen actualizadas en todo momento utilizando para ello, mensajes de actualización periódicos. Estos protocolos introducen cierto nivel de sobrecarga, sin embargo, tienen la ventaja de poder seleccionar rutas válidas de forma prácticamente inmediata.

2.7.2. Protocolos reactivos

Los protocolos de encaminamiento reactivos operan bajo demanda. Los protocolos reactivos calculan la ruta óptima hacia un destino específico en el momento que se requiere. Este tipo de protocolo construye las rutas entre un nodo origen y un nodo destino únicamente cuando es necesario enviar un paquete, por eso es bajo demanda. Cuando un nodo quiere enviar un paquete, comienza por descubrir rutas y este proceso termina cuando se encuentra una ruta válida hacia el nodo destino o cuando se hayan examinado todas las posibles alternativas y ninguna de estas descubren el nodo destino. A diferencia de los protocolos proactivos, los protocolos reactivos poseen mayor latencia, puesto que es necesario tener en cuenta el retardo del proceso de descubrimiento de rutas, sin embargo no existe necesidad de estar actualizando en cada momento las tablas de encaminamiento. Los protocolos reactivos reducen la sobrecarga que

se genera cuando se actualizan los mensajes de encaminamiento de los protocolos proactivos. Un problema con los protocolos reactivos ocurre en el momento de buscar una ruta para llegar de un nodo origen cualquiera, a un nodo destino cualquiera y esta ruta se tiene que generar en el acto, lo que implica cierto grado de latencia.

Los protocolos reactivos más importantes a día de hoy son los que se mencionan a continuación, el protocolo AODV (Ad-Hoc On Demand Distance Vector) es tratado a detalle en el apartado 2.7.2.1.

- DSR (Dynamic Source Routing) [80], es un protocolo de encaminamiento que basa su proceso de encaminamiento en el origen (los paquetes de datos llevan en la cabecera la información de los nodos que ha se seguir para llegar al destino). DSR no utiliza ningún tipo de mensajes periódicos, de esta forma reduce la sobrecarga con mensajes de control. Por otra parte, al momento de descubrir rutas, descubre una variedad de caminos alternos hacia el destino. El encaminamiento en origen inserta en la cabecera de cada paquete de datos, la información de la ruta nodo a nodo, que debe seguir el paquete. Cada nodo conserva y mantiene una memoria caché de las rutas en la que se almacena las rutas que obtiene durante el proceso de descubrimiento de rutas o a través del proceso de escuchar en la red. DSR maneja mensajes de solicitud de ruta (ROUTE REQUEST), mensaje de respuesta (REPLY) y mensaje de error (ERROR).

- DYMO (Dynamic Manet On Demand routing) [23], es un protocolo que posee características de los protocolos reactivos AODV y DSR; DYMO al ser también un protocolo reactivo, introduce una latencia en el proceso de descubrimiento de una nueva ruta, sin embargo, utiliza poco tráfico de control. DYMO no envía paquetes de control si no está ejecutando funciones de encaminamiento, transmisión o recepción de información, lo cual es ideal en redes con limitación de recursos.

DYMO emplea dos mecanismos similares a los protocolos reactivos DSR o AODV: Uno, el descubrimiento de ruta, utilizado en los casos cuando no se encuentra en la tabla de encaminamiento la ruta del nodo destino, y dos, el mantenimiento de ruta, que utiliza varias formas para descubrir rutas en caso de ocurrir un ruptura de ruta.

- LQSR (Link Quality Source Routing protocol) [44] [45], es un protocolo de encaminamiento en origen por calidad de enlace. Se basa en el protocolo DSR, LQSR selecciona la ruta en función de métricas relacionadas con la calidad del enlace. Con esta versión modificada de DSR, LQSR mejora el comportamiento en general de DSR, especialmente para soportar métricas basadas en la calidad del enlace de radio.

- TORA (Temporary Ordered Routing Algorithm) [119], es un protocolo de encaminamiento bajo demanda. El funcionamiento de TORA está basado en el mantenimiento de un grafo dirigido y sin ciclos para alcanzar al nodo destino. El objetivo de TORA es minimizar la carga sobre la red. Básicamente, lo que marca la diferencia con otros protocolos, es la imposibilidad de calcular constantemente la distancia hacia el destino o de mantener siempre la ruta más corta. Sin embargo, tiene la ventaja de que es un algoritmo muy eficiente, ya que no satura excesivamente de tráfico la red.

  TORA proporciona varias rutas para cada destino. Aunque esto es benéfico cuando existe demasiada movilidad en la red, podría parecer redundante poseer más de una ruta, ocasionando que las tablas de encaminamiento estén sobrecargadas con información redundante y en algunos casos podría parecer innecesaria.

2.7.2.1. AODV

En este apartado se revisa a detalle el protocolo de encaminamiento AODV, excluido de los protocolos reactivos tratados en el apartado 2.7.2, lo anterior se

debe a dos razones: primero, porque en ese apartado únicamente hemos revisado de manera sucinta los principales protocolos de encaminamiento reactivos; y segundo, porque AODV es el protocolo de encaminamiento subyacente en el que se basa nuestra propuesta de solución, así que en este apartado dedicaremos más espacio a dicho protocolo.

AODV (Ad Hoc on Demand Distance Vector Routing) [124], es un protocolo de encaminamiento para redes Ad Hoc. Es un protocolo reactivo, basado en vector de distancias y opera bajo demanda, es decir, que sólo comienza la búsqueda de rutas cuando es requerido. El descubrimiento de rutas genera tablas de encaminamiento y estas tablas se almacenan localmente en cada nodo y sirven de medio para establecer enlaces con el resto de nodos.

Una ventaja de los protocolos de encaminamiento bajo de demanda, es que pueden eliminar la necesidad de difusiones periódicas de anuncios de rutas. Las rutas se crean únicamente cuando son requeridas, con ello se minimizan difusiones y latencia de transmisión. AODV mejora las características de rendimiento de DSDV en cuanto a creación y mantenimiento de redes Ad Hoc se refiere [124]. Esta es una de las razones por las que hemos tomado este protocolo como base para nuestra propuesta de solución. Otra razón es que AODV es uno de los protocolos reactivos más utilizados y es idóneo para MANET.

El protocolo AODV crea tablas que contienen las rutas de encaminamiento hacia los nodos destinos conocidos. Sin embargo, no mantiene rutas hacia cada uno de los nodos de la red. Este protocolo únicamente crea las rutas hacia los nodos con quienes mantiene una comunicación activa, lo cual genera poca sobrecarga en la red. Las rutas se van descubriendo conforme se van necesitando. AODV soporta tres tipos de transmisión: unicast[3], multicast[4] y broadcast[5]. No obstante, AODV descubre las rutas basándose en un mecanismo de petición de rutas por medio de broadcast y respuestas por medio de unicast [160].

[3]Unicast - se envían paquetes de un nodo a otro.
[4]Multicast - se envían paquetes de un nodo a un grupo de nodos.
[5]Broadcast - se transmiten paquetes de un nodo al resto de nodos de la red.

Inicialmente la tabla de rutas se forma con los nodos vecinos. Para ello, un nodo envía en modo broadcast mensajes de descubrimiento de ruta RREQ (Route Request, los campos de este formato de mensaje se muestran en la figura 2.6). Si un nodo que no es el destino solicitado recibe este paquete, verifica mediante los números de secuencia si lo ha recibido previamente. En caso de que no lo haya recibido, lo reenvía e incrementa el número de saltos y crea la ruta inversa para devolver la respuesta RREP (Route Reply, los campos se muestran en la figura 2.7). La respuesta de establecimiento de ruta RREP es generada por los nodos intermedios que cuentan con una ruta hacia el destino o el mismo destino una vez que el mensaje RREQ le haya llegado. Los nodos envían estas respuestas RREP a los nodos que le habían enviado el mensaje RREQ. La ruta se confirma una vez el RREP llega al nodo que había comenzado el descubrimiento de ruta.

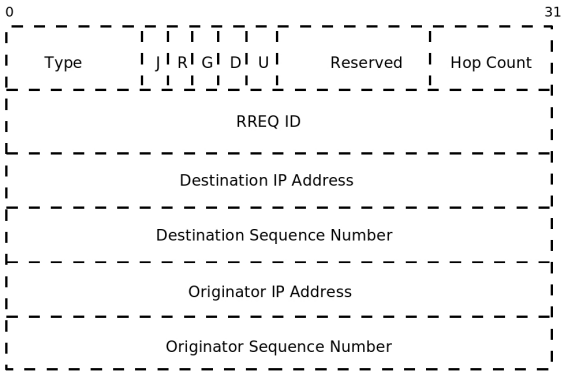


Figura 2.6: Formato del mensaje RREQ del protocolo AODV.

Para mantener el enlace entre los nodos origen-destino después de establecerse una ruta, está se mantiene válida durante un período de tiempo, si los nodos que tienen rutas activas realizan envíos periódicos de mensajes RREP ajustando en 1 el TTL[6], es decir, mediante la difusión local del mensaje a sólo los nodos situados a 1 salto de distancia. Existe otro mecanismo adicional para garantizar la validez de las tablas; consiste en enviar periódicamente mensajes

[6]Time-to-live - variable para controlar el número máximo de saltos.

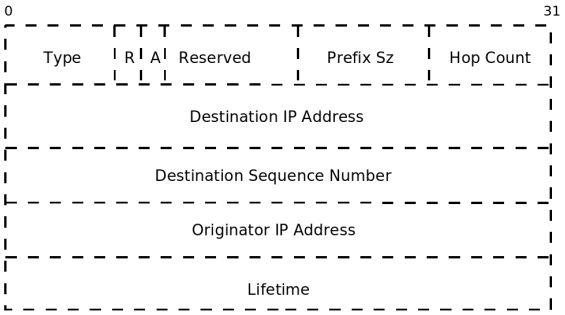


Figura 2.7: Formato del mensaje RREP del protocolo AODV.

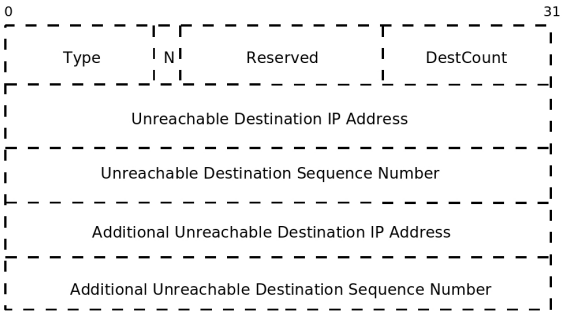


Figura 2.8: Formato del mensaje RERR del protocolo AODV.

HELLO entre los nodos vecinos a fin de determinar la presencia en la red, mantener frescas las tablas de rutas y conservar la conectividad entre ellos, aunque para reducir el volumen de estos mensajes sólo debe permitirse su envío a los nodos que estén transmitiendo datos. Si luego de algún tiempo no se reciben mensajes HELLO desde un nodo vecino, se hace suponer que este dispositivo ya no está disponible, por lo que se desecha la ruta y se borra de la tabla. De esta forma, cuando un nodo detecta la ausencia de un vecino mediante los mensajes HELLO o un error en el envío de un mensaje, se origina un mensaje RERR (Route Error, cuyos campos se muestran en la figura 2.8), que se emplea para invalidar una ruta. De modo que, este mensaje RERR llegará hasta el origen de la ruta para que el nodo comience nuevamente el proceso de descubrimiento.

A continuación ilustramos brevemente a través de las figuras (2.9 2.10, 2.11 y 2.12), un ejemplo del funcionamiento básico del protocolo AODV.

Se trata de una pequeña MANET con 11 nodos móviles, los nodos son representados por pequeños círculos mientras que los enlaces se representan por medio de arcos. Los nodos se identifican con las letras **A**, **B**, **C**, .. **K**. El término RREQ representa un *Route Request* (petición de ruta), el RREP representa un *Route Reply* (respuesta de ruta) y el RERR representa un *Route Error* (error de ruta).

El nodo origen **A** difunde por broadcast un mensaje (RREQ) (figura 2.9) en busca de una ruta válida para alcanzar al nodo **K**. El nodo destino **K** recibe el mensaje RREQ y responde por unicast enviando un mensaje RREP al nodo **A** para informar que ha recibido el mensaje RREQ (figura 2.10); a partir de este momento se ha establecido un enlace entre el nodo origen **A** y el nodo destino **K**. Si después de un tiempo ocurre un corte en el enlace[7] entre el nodo destino **K** y algún nodo intermedio antes de alcanzar el nodo origen **A**, entonces el nodo origen **A** recibe el mensaje RERR (generado en este caso por el nodo **J**) en el que se informa que ha ocurrido una rotura de enlace (figura 2.11). Cuando ocurre un fallo de este tipo, entonces el nodo origen **A** comienza nuevamente el proceso de descubrimiento de ruta para llegar al nodo destino **K**.

En la figura 2.12 se puede visualizar el ejemplo completo, el nodo **A** intenta descubrir una ruta para establecer conexión con el nodo destino **K**. Primero, el nodo **A** difunde mensajes RREQ, los nodos intermedios reenvían el mensaje RREQ, cuando el nodo **K** recibe el mensaje RREQ significa que el mensaje ha llegado al destino final. Segundo, el nodo **K** responde por unicast mensajes RREP hasta el nodo origen **A**, los nodos intermedios reenvían los mensajes RREP con objeto de que estos lleguen al nodo **A**, de esta forma se establece conexión entre el nodo destino **K** y el nodo origen **A**. Y tercero, en caso de ocurrir un fallo en el enlace debido a una desconexión de un nodo o debido a la propia movilidad, entonces se le envía al nodo **A** un mensaje RERR que es generado por el nodo intermedio **J**, los nodos intermedios reenvían el mensaje RERR hasta que llegue al nodo **A**.

[7]Conocido también como enlace roto o rotura de enlace. Esta situación se genera cuando un nodo detecta la caída de un enlace con otro nodo.

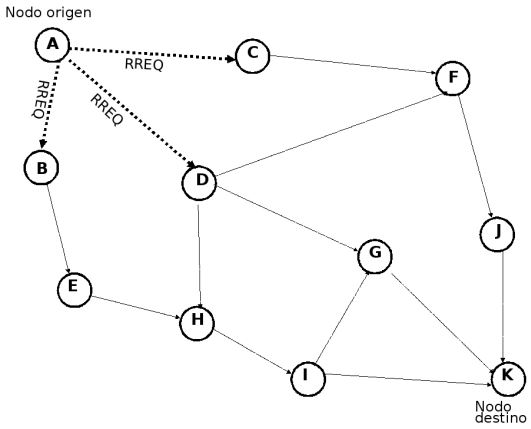


Figura 2.9: El nodo origen **A** envía por broadcast un mensaje RREQ para descubrir una ruta para alcanzar el nodo destino **K**. Los nodos intermedios reenvían el mensaje RREQ.

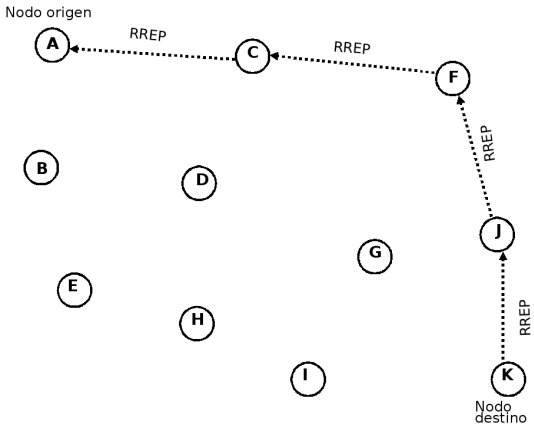


Figura 2.10: El nodo destino **K** devuelve por unicast un mensaje RREP al nodo **A** con la finalidad de informarle que ha recibido el mensaje RREQ. Los nodos intermedios participan en el descubrimiento de ruta reenviando el mensaje RREP.

2.7.3. Protocolos híbridos

Un protocolo de encaminamiento híbrido combina las ventajas de los protocolos proactivos y de los protocolos reactivos. Un ejemplo muy representativo que muestra el uso de las características de los protocolos proactivos y reactivos

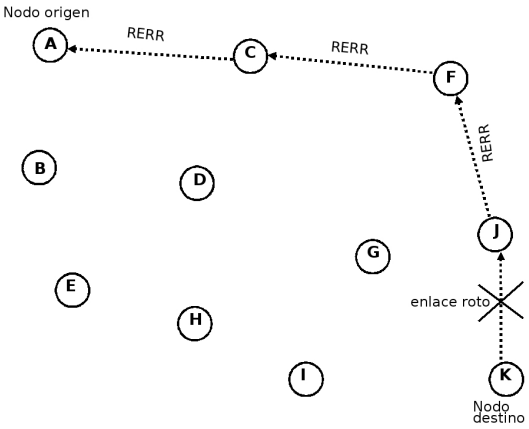


Figura 2.11: Cuando se rompe el enlace entre el nodo **A** y el nodo **K**, el nodo intermedio **J** avisa al nodo **A** que ha ocurrido un fallo en el enlace. El nodo **J** es el que genera el mensaje RERR. Del mismo modo, los nodos intermedios reenvían el mensaje RERR.

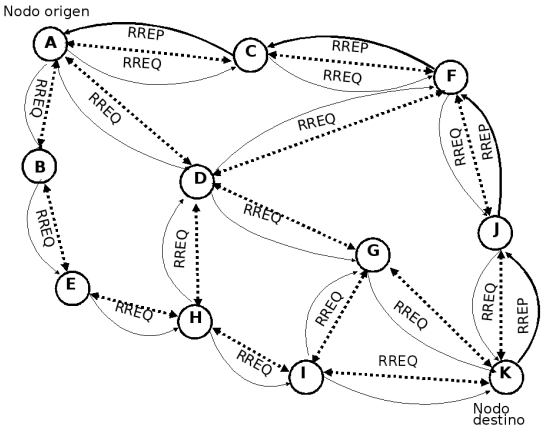


Figura 2.12: El nodo origen **A** encuentra una ruta válida para alcanzar al nodo **K**. Los nodos intermedios se encargan de reenviar los mensajes RREQ, RREP y RERR.

es ZRP. Debido a que nuestra propuesta no involucra el uso de protocolos híbridos, consideramos mencionar solamente un ejemplo de este tipo de protocolo.

ZRP (Zone Routing Protocol) [67] utiliza esquemas de encaminamiento de los protocolos proactivos, con el fin de mantener actualizada la información de la topología; opera dentro de una zona limitada a nodos vecinos a determina-

dos saltos y emplea esquemas de encaminamiento reactivos para nodos que se encuentran fuera de esta zona. Las rutas hacia los nodos dentro de la zona son disponibles de manera inmediata. Para destinos fuera de la zona, ZRP emplea un procedimiento de descubrimiento de ruta, en el que utiliza la información local de las zonas. Para ámbitos locales ZRP opera a la manera de los protocolos proactivos, mientras que para ámbitos globales mantiene la filosofía de los protocolos reactivos. Debido a que las zonas se traslapan, ZRP se clasifica como un protocolo de encaminamiento plano. El comportamiento de ZRP es adaptativo ya que puede detectar rutas óptimas y con ello reducir la congestión de la red. Por lo tanto, el comportamiento depende tanto de la configuración actual de la red como del comportamiento de los usuarios.

2.8. Técnicas de clustering en los protocolos de encaminamiento

En esta sección trataremos el tema de las técnicas de clustering, que consisten en agrupar los nodos de la red con el propósito de obtener algunos beneficios, entre ellos aprovechar su proximidad para reducir significativamente el tráfico global de la red.

Como es sabido, las MANET poseen topologías aleatorias, muy dinámicas y de múltiples saltos. Las MANET presentan por un lado, limitaciones respecto al uso del ancho de banda y al consumo de energía y por otro lado, es de esperar que una MANET pueda soportar un gran número de nodos móviles. En este sentido, es deseable procurar razonables niveles de escalabilidad a fin de garantizar un buen rendimiento.

Para alcanzar un alto grado de escalabilidad en las MANET, se pone en marcha una estrategia que consiste en jerarquizar la topología por medio de un proceso conocido como clustering. La jerarquización de los nodos tiene como objetivo hacer una abstracción de la topología de red.

Los técnicas de clustering[8] mejoran el rendimiento de las MANET en cuanto a que incrementan la rapidez de la conexión, optimizan el proceso de encaminamiento y ayudan en el manejo de la topología [37].

Las técnicas de clustering ofrecen ventajas a la capa de acceso al medio y a la capa de red. El esquema de clustering permite un mejor rendimiento de los protocolos de la capa de control de acceso al medio (MAC) puesto que mejora la reutilización del espacio, el throughput, la escalabilidad y el consumo de energía. Por otro lado, las técnicas de clustering ayudan a mejorar el encaminamiento en la capa de red porque reducen el tamaño de las tablas de encaminamiento y disminuyen la sobrecarga de transmisión, gracias a la actualización de las tablas de encaminamiento después de ocurrir un cambio en la topología [157]. El clustering ayuda a conjuntar la información topológica ya que el número de nodos de un cluster es menor que el número de nodos de toda la red. Por consiguiente, cada nodo sólo requiere almacenar una fracción del total de la información de encaminamiento [51].

En resumen, el clustering es un proceso de jerarquización en el que los nodos de la MANET se auto-organizan en grupos denominados clusters, obteniendo con ello una topología relativamente estable y un alto grado de escalabilidad. La auto-organización de los nodos se consigue a través de la información local que posee cada nodo de sus respectivos vecinos.

En varias de las técnicas de clustering se seleccionan nodos para desempeñar diferentes roles de acuerdo a ciertos criterios. Estos roles son tres:

- Nodos ordinarios .- En los clusters existe un tipo de nodo cuya función se reduce simplemente a ser nodos miembros del cluster, por lo general sólo pertenecen a un mismo cluster [127].

- Nodos gateways .- Este tipo de nodo se encuentra localizado en la periferia de un cluster. Los nodos gateways tienen la capacidad de escuchar transmisiones de un nodo miembro de otro cluster.

[8]El clustering es un técnica que organiza los nodos de la red en grupos, estos grupos denominados clusters deben satisfacer ciertas propiedades.

- Nodos clusterheads .- Muchos de los esquemas de clustering seleccionan un subconjunto de nodos (denominado clusterdhead) para formar un backbone que soporte funciones de control. Cada nodo de la red se encuentra asociado con un clusterhead. Los clusterheads están directamente conectados con otro clusterhead a través de los gateways. El backbone se forma de la unión de los gateways con los clusterheads [7].

En el proceso de agrupamiento de nodos próximos entre sí, se involucra la elección de un líder, que es responsable del tráfico de rutas dentro y fuera del cluster, se trata desde luego del nodo clusterdhead. Uno de los criterios para la elección del nodo líder es tener en cuenta al nodo vecino con mayor grado de capacidad.

El nodo clusterhead (nodo principal dedicado) es el encargado de indicar a los nodos más próximos la pertenencia a cierto cluster. Los clusterheads además de estar informados de la conexión y desconexión de los nodos, también se encargan de interactuar con los nodos gateways entre los diferentes clusters. Generalmente los clusters tienen diferentes capas jerárquicas, en este tipo de esquemas se envía información con mayor frecuencia a los nodos que poseen mayor movilidad o mayor proximidad.

Con los beneficios que se pueden obtener del proceso de clustering en la redes Ad Hoc, es posible utilizar este esquema para descubrir servicios. En la literatura de este campo se documentan propuestas que basan el descubrimiento de servicios en esquemas de clustering, trabajos como [141] y [24] que ya hemos reseñado en el apartado 2.2 y en trabajos como [3] [57] y [91], por citar solo algunos. En todas estas propuestas se plantean una variedad de ideas de agrupamiento de nodos para descubrir servicios. Similarmente también se ha utilizado el diseño clustering en propuestas de solución para redes inalámbricas de sensores (WSN, wireless sensor networks) [97] [159] [98]. Cabe señalar que en las WSN no se tiene en cuenta la movilidad, puesto que la mayoría de los dispositivos que la integran se encuentran fijos, hecho que las hace diferenciar de las MANET en las que la movilidad es un factor crítico.

En el enfoque de descubrimiento de servicios basados en clustering se llevan a cabo búsquedas más inteligentes y se logra una eficiente inundación de mensajes y con ello se reduce la redundancia inherente al inundar de mensajes toda la red. Con las técnicas de clustering en el descubrimiento de servicios, es posible reducir los mensajes redundantes de las búsquedas sobre toda la red y únicamente se descubre servicios en grupos de nodos.

Por otra parte, las técnicas de clustering llevan a cabo una compleja coordinación para mejorar la reconfiguración de la topología, afectada por la movilidad y los fallos de los nodos, lo que ocasiona una seria degradación del rendimiento sobre todo en redes altamente dinámicas. Así, un clusterdhead podría llegar a ser un punto de fallo y un potencial cuello de botella [92] [71].

2.8.1. Algoritmos de clustering

En este apartado daremos una breve introducción a los algoritmos que emplean las técnicas de clustering:

- Algoritmo *Highest degree heuristic* [118], se basa en la conectividad entre un nodo y sus vecinos directos. Cada nodo difunde periódicamente su valor de conectividad a sus vecinos directos. Para que un nodo pueda convertirse en clusterhead o conservar su condición de nodo ordinario, entonces debe comparar el valor de conectividad de sus vecinos con su propio valor. Si algún nodo en su vecindario tiene el valor más alto de conectividad, éste llegará a ser un clusterhead. El algoritmo Highest degree heuristic emplea como criterio de selección de clusterhead, el valor de conectividad más alto de entre sus vecinos directos. El algoritmo consiste básicamente en: difundir una lista de los nodos que puede escuchar, incluido a sí mismo. Un nodo es electo clusterhead, si es el nodo con el valor más alto de todos los nodos vecinos encubiertos, en caso de empate, prevalece el ID más bajo. Un nodo que no haya elegido su clusterhead aún, es un nodo encubierto, de lo contrario es un nodo cubierto. Un nodo que haya elegido a otro nodo como su clusterhead, renuncia a su rol como clusterhead.

- Algoritmo *Lowest-ID* [4] [5], cada nodo de la red tiene un identificador único designado (ID). Los nodos periódicamente difunden su ID a sus vecinos directos. Cada nodo compara los IDs de sus vecinos con su propio ID. El algoritmo Lowest-ID emplea como criterio para la selección de clusterheads, el ID más bajo del nodo de entre los vecinos que se encuentran a 1 salto de distancia. Un nodo puede llegar a ser clusterhead, si tiene el ID más bajo de entre los ID de sus vecinos. El algoritmo Lowest-ID consiste en que a cada nodo se le asigna un ID distinto. El nodo difunde periódicamente la lista de nodos que puede escuchar, incluido a sí mismo. Un nodo es clusterhead si solamente escucha nodos con ID más alto que su propio ID; por lo que el nodo con el ID más bajo que un nodo escuche, es un clusterhead, a menos que el nodo con ID más bajo, especifique renunciar a su rol como clusterhead. Un nodo que escucha dos o más clusterheads es un gateway. De otro modo, se trata de un nodo ordinario.

- Algoritmo *k-CONID* [108], este algoritmo se crea con la combinación de los algoritmos Lowest-ID con Highest-degree. Se llama k-CONID por k-hop connectivity ID. Para seleccionar los clusterheads se considera como primer criterio la conectividad y como segundo criterio el menor ID. Usar como criterio sólo la conectividad de un nodo, causa numerosos vínculos entre los nodos. Por otro lado, usar sólo el criterio de menor ID genera más clusters de los necesarios. El propósito es minimizar el número de clusters formado en la red y de esta forma obtener conjuntos dominantes de tamaños más pequeños. Los clusters en el enfoque k-CONID se forman por un clusterhead y todos los nodos que están a la distancia de al menos k-hops del clusterhead. Este algoritmo comienza cuando un nodo empieza un proceso de inundación enviando al resto de nodos una petición de clustering. En el algoritmo Highest-degree el grado de un nodos sólo mide la conectividad para clusters a 1 salto de distancia. k-CONID generaliza la conectividad para un vecindario k-hop. De esta forma la conectividad para cuando $k = 1$ es la misma que el grado de un nodo.

- Algoritmo *(α, t)* [100], es un algoritmo simple, distribuido y asíncrono que usa un modelo de probabilidad para determinar la disponibilidad de ruta, con el objetivo de asegurar las decisiones de clustering. Este algoritmo organiza de manera dinámica los nodos dentro de los clusters. En el enfoque (α, t) se intenta proveer una topología efectiva que se adapte a la movilidad de los nodos. La disponibilidad de ruta es un proceso aleatorio que está en función de la movilidad de los nodos de una cierta ruta. En el enfoque (α, t) las rutas se evalúan por dos parámetros del sistema: α y t. α establece 1 salto más pequeño en la probabilidad de una ruta dada por el cluster que permanecerá disponible por un tiempo t. α controla la estabilidad del cluster mientras el rol de t es manejado por el tamaño del cluster por un nivel dado de estabilidad.

- Algoritmo *Max-min d-cluster* [2], a diferencia de los algoritmos que eligen sus clusterheads teniendo en cuenta distancias a no más de 1 salto del clusterhead, el algoritmo Max-min d-cluster considera distancias a d-hops o d-saltos. La principal desventaja de los nodos que están a 1 salto de distancia del clusterhead, es la generación de un gran número de clusterheads en la red, lo que ocasiona un problema de congestión. Por esta razón, en la heurística Max-min d-cluster los clusters se forman por nodos que están al menos a d-hops de distancia del clusterhead. Un d-vecindario de un nodo consiste del nodo en sí y el conjunto de todos los nodos localizados dentro de d-hops de distancia del nodo. d se define como el número máximo de saltos de distancia del clusterhead más cercano ($d \geq 1$). Este valor es una entrada a la heurística que le permite controlar el número de clusterheads que serán seleccionados. Los nodos participan en el algoritmo de elección del clusterhead basándose en sus respectivos ID. Cada nodo cuenta con dos arreglos: WINNER y SENDER. El arreglo WINNER se corresponde con el ID del nodo ganador de una serie en particular y el arreglo SENDER se corresponde con el nodo que envía el ID del nodo ganador de una se-

rie en particular. Una vez seleccionado el clusterhead, SENDER ayuda a determinar las rutas más cortas de vuelta al clusterhead.

- Algoritmo *MobDhop* [51], MobDhop particiona una red móvil Ad Hoc en clusters d-hop basándose en una métrica de movilidad. El objetivo de formar clusters d-hop es hacer más flexible el diámetro del cluster. El diámetro del cluster se adapta de acuerdo a la movilidad del nodo. MobDhop asume que cada nodo puede medir su potencia de señal que recibe. De esta manera, un nodo puede determinar la cercanía de sus vecinos [10]. MobDhop emplea como criterio de selección el valor más bajo de estabilidad local entre los nodos. Una fuerte señal de potencia significa cercanía entre dos nodos. El algoritmo MobDhop requiere calcular cinco términos: la distancia estimada entre los nodos, la movilidad relativa entre los nodos, la variación de la distancia estimada en el tiempo, la estabilidad local y la estimación de la distancia media.

- Algoritmo DMAC (distributed mobility-adaptive clustering) [8], es un algoritmo distribuido en el que los clusterheads se seleccionan mediante el criterio basado en peso, en función de los parámetros de movilidad relativa de los nodos. Este algoritmo es apropiado para la gestión de redes móviles altamente dinámicas. DMAC supera una de las desventajas presentes en la mayoría de los algoritmos de clustering. Una suposición común en algunos algoritmos, es creer que durante la aparición o establecimiento de los nodos, estos no se mueven al momento de ser agrupados en clusters. Normalmente, los algoritmos de clustering particionan la red en clusters y sólo después de terminar este paso, suponen que comienza a existir movilidad, antes no. Más tarde, el algoritmo intenta mantener la topología del cluster tan pronto como los nodos comiencen a moverse. Una característica importante de DMAC es que los nodos pueden moverse, incluso durante la formación del clustering. Durante la ejecución del algoritmo DMAC se asume que cada nodo tiene un peso (un número real ≥ 0) y un ID aso-

ciado a él. El peso de un nodo representa los parámetros de movilidad del nodo. Un nodo elige su propio rol (clusterhead u ordinario) basándose en el conocimiento de sus actuales vecinos localizados a 1 salto de distancia. Un nodo llega a ser clusterhead, si tiene el valor más alto entre sus vecinos a 1 salto de distancia, de lo contrario, se une a un clusterhead vecino.

- Algoritmo WCA [27], es un algoritmo distribuido basado en pesos. WCA (weighted clustering algorithm) elige sus clusterheads en base al criterio del valor del peso de cada nodo. El algoritmo WCA selecciona los clusterheads teniendo en cuenta aspectos relacionados al funcionamiento eficiente de los componentes del sistema. Por eso, para optimizar energía, balanceo de carga y funcionalidad de MAC, elige un nodo clusterhead de acuerdo a la capacidad de nodos que puede manejar, a la movilidad, a la potencia de transmisión y a la capacidad de batería. Para evitar la sobrecarga de comunicaciones, el algoritmo no es periódico y el procedimiento de la elección del clusterhead se invoca únicamente en base a la movilidad del nodo y cuando el actual conjunto dominante sea incapaz de cubrir todos los nodos. Para asegurar que no se sobrecarguen los clusterheads, se establece un umbral predefinido, que especifica el número de nodos que cada clusterhead puede idealmente soportar.

Conclusiones

A lo largo de esta sección hemos abordado lo referente a las técnicas de clustering empleadas por los protocolos de encaminamiento, de las que se ha mencionado se obtienen beneficios como reducir el tráfico global de la red, alcanzar cierto grado de escalabilidad, mejorar el rendimiento de la red al incrementar la rapidez de la conexión, optimizar el proceso de encaminamiento y ayudar en el manejo de la topología. Así mismo, se han repasado los algoritmos de clustering más representativos, no obstante existe una gran variedad de heurísticas y algoritmos; estos algoritmos emplean un determinado esquema para la declaración del líder a conveniencia del propio mecanismo de elección, que por lo regular

eligen a sus líderes de clusters en función del número de saltos de distancia entre los nodos o basados en pesos de conectividad o basados en el mayor o menor valor del identificador (ID) o basados en la combinación o extensión de algoritmos previos, entre otras heurísticas.

2.9. Diseño Cross-layer

En esta sección abordaremos el tema de diseño Cross-layer, se mencionan los inconvenientes de emplear el modelo de referencia OSI en las MANET y las ventajas de utilizar el diseño Cross-layer para descubrir servicios en redes Ad Hoc.

La técnica Cross-layer consiste en intercambiar información entre las diferentes capas del modelo OSI, básicamente lo que se busca es la interoperabilidad entre las diferentes capas del sistema, a través del intercambio de información entre ellas. En este sentido, el diseño Cross-layer presenta dentro de la arquitectura de capas, la posibilidad de que los protocolos que pertenezcan a diferentes capas, puedan cooperar entre sí para compartir información del estado de la red, mientras aún mantienen la separación entre las capas en el diseño del protocolo. Un trabajo en el que incluso hasta proponen una pila de protocolos con el esquema Cross-layer es en el proyecto MobileMan[9] [36].

El diseño Cross-layer rompe un poco el paradigma del modelo de referencias OSI, porque éste es inflexible en cuanto a lo que permite comunicar entre capas, no permite la comunicación, ni el paso de parámetros entre capas que no sean adyacentes. El enfoque Cross-layer es más flexible y permite una mayor retroalimentación entre capas [143].

En la figura 2.13 se puede apreciar el modelo OSI con sus 7 capas, modelo rígido en el que la comunicación se lleva a cabo sólo entre capas adyacentes y el

[9]La arquitectura Cross-layer MobileMan promueve la interacción local entre protocolos en un nodo de MANET. EL estado de red maneja uniformemente la interacción Cross-layer y respeta el principio de división de funcionalidades y responsabilidades de las capas. Este enfoque logra optimizar el desempeño global de la red al incrementar la interacción local entre protocolos, reduce las comunicaciones remotas y en consecuencia ahorra ancho de banda.

diseño Cross-layer en el que es posible el intercambio de información entre capas no adyacentes, por ejemplo, en el diseño Cross-layer es posible que parámetros de la capa de aplicación puedan tener comunicación directa con la capa de enlace. Bajo el enfoque Cross-layer, un capa pueden adaptarse a los requerimientos y condiciones particulares de las otras capas [61].

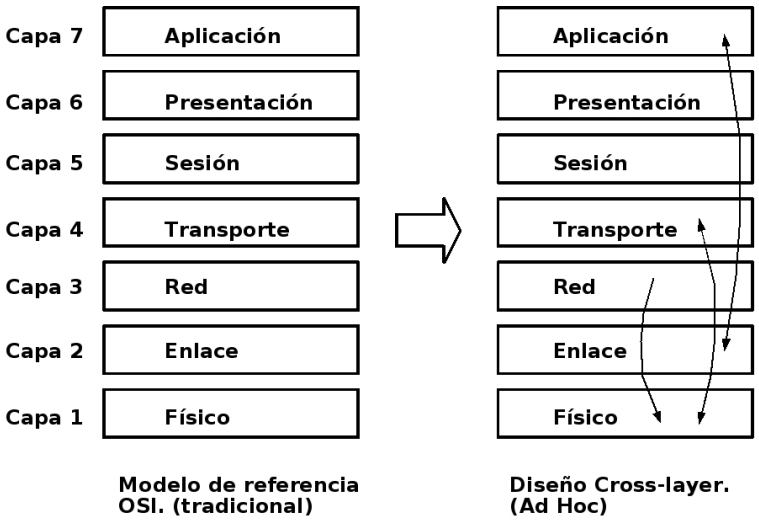


Figura 2.13: Comunicación entre capas adyacentes en el modelo OSI y comunicación entre capas no adyacentes en el diseño Cross-layer.

En muchos trabajos en el área de descubrimiento de servicios se menciona que la integración Cross-layer en la pila de protocolos, mejora la eficiencia del sistema en general, mejora la accesibilidad a los servicios y reduce la sobrecarga de la infraestructura [28].

Por otra parte, la estructura de la pila de protocolos OSI no es lo más conveniente y óptimo para MANET, porque a diferencia de los enlaces convencionales de las redes fijas y cableadas, en las redes móviles el canal de comunicación es altamente variable, los nodos demasiado dinámicos y se utiliza la difusión como medio.

La rigidez y suboptimización del modelo OSI produce un pobre rendimiento en las MANET, especialmente cuando se tienen presente restricciones de energía o cuando la aplicación demanda necesidades de gran ancho de banda o tiene

severas restricciones de retardos. Por eso es necesario un diseño Cross-layer que maneje estos requerimientos y soporte la adaptabilidad y optimización a través de múltiples capas [61] [129].

El modelo OSI es subóptimo para redes MANET porque considera cada capa como una entidad funcional totalmente independiente, mientras que el diseño Cross-layer comparte entre varias capas la información de cada capa. Por lo que el enfoque Cross-layer al utilizar esta información, optimiza el rendimiento general de todo el sistema de manera global.

En la figura 2.14 se muestra el esquema de búsqueda propuesto en [56], en el que con un diseño Cross-layer se descubren los servicios solicitados en la capa de aplicación. En la capa de aplicación se localizan los diferentes tipos de servicios (correo electrónico, servicios web, etc,). La búsqueda de estos servicios se efectúa entre la capa de red (protocolo de encaminamiento) y la capa de enlace. Por lo general, en la capa de aplicación se realizan los registros y las peticiones de servicios, el componente de aplicación tanto envía como recibe mensajes de datos del componente de encaminamiento (AODV) y la capa de red interactúa directamente con la capa de enlace.

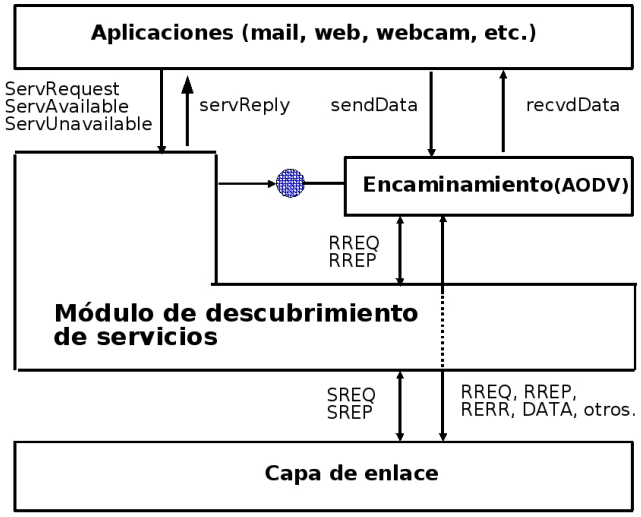


Figura 2.14: Descubrimiento de servicios en la capa de red.

Por otra parte, existen algunos beneficios mencionados en el trabajo [26], producto de la integración del proceso de descubrimiento de servicios y el proceso de descubrimiento de rutas que a continuación resumimos:

1. *Uso de rutas disponibles* - Durante el proceso de descubrimiento de servicios también se descubren múltiples caminos para llegar a un servicio. Es decir, debido a la movilidad de los nodos es ideal tener distintas rutas para alcanzar un servicio; mientras que en las redes fijas no se requiere tener varios caminos, ya que no existe movilidad.

2. *Rutas orientadas a servicios* - En diferentes nodos pueden existir múltiples instancias del mismo servicio. En caso de ser necesario, el enfoque Cross-layer puede usar la información en el descubrimiento para enrutar la petición de servicios a una instancia de servicios en vez de hacerlo a la dirección del nodo. Esto convierte al protocolo en *orientado a servicios* en lugar del enfoque tradicional *orientado a nodos*.

3. *Resistente a fallos servicio-nodo* - Todos los protocolos de encaminamiento son orientados a nodos (enrutan en base a la dirección o IP del nodo) y por tanto, es propenso a los fallos de ese nodo. Un fallo servicio-nodo coloca a un servicio en un estado de indisponibilidad y por consiguiente, a un fallo en el servicio. Lo ideal es que el proceso de descubrimiento de servicios sea inmune a fallos servicio-nodo, ya que múltiples instancias del mismo servicio pudieran estar disponibles en diferentes nodos.

4. *Reducción de la sobrecarga de encaminamiento* - Reutilizar la infraestructura de descubrimiento para invocar rutas produce una reducción en la sobrecarga, ya que ciertas acciones de los protocolos de encaminamiento como descubrimiento de rutas y mantenimiento de caminos pueden potencialmente estar integradas con el proceso de descubrimiento de servicios.

En gran parte de los protocolos de descubrimiento de servicios existentes el proceso de descubrimiento de servicios se lleva a cabo en la capa de aplicación.

Sin embargo, existen otras propuestas como [94] [53] [49] [30] [55] y [68] en las que el descubrimiento de servicios se realiza en la capa de red, es decir, durante el proceso de encaminamiento. El hecho de utilizar técnicas Cross-layer podría traer cierta optimización y ciertas ventajas sobre los esquemas de descubrimiento de servicios que realizan este proceso en la capa de aplicación. En tanto que otros autores [139] opinan que se sacrifica el enfoque tradicional de capas y que no se preserva el enfoque modular de capas del modelo de referencia OSI.

Por otra parte, los protocolos que descubren servicios en la capa aplicación en su esfuerzo por descubrir rutas hacia los servicios, redundan en la transmisión de paquetes, ya que los mensajes de control para descubrir información de servicios requiere tanto de la capa de red como de la capa de aplicación. De ahí la motivación de integrar el proceso de descubrimiento de servicios con el proceso de encaminamiento, para que los nodos encuentren servicios y rutas simultáneamente [152]. Por lo que, dos procesos que producen mensajes deben coexistir: el primero comunica información de servicio entre proveedores y peticionarios de servicios, y el segundo comunica información de encaminamiento entre ellos. Lo que produce que un nodo sea forzado a ejecutar muchas veces la operación y con esto agote la batería debido a la transmisión y recepción de paquetes de control [117].

En nuestra propuesta de solución se plantea la idea de integrar el descubrimiento de servicios con los mecanismos de encaminamiento para redes Ad Hoc (en nuestro caso AODV), de esta forma es posible que los nodos móviles se enteren de los servicios disponibles en la red a través del proceso de creación y actualización de rutas del proceso de encaminamiento. Esencialmente la idea es optimizar el proceso de descubrimiento de servicios, mediante el aprovechamiento de los paquetes de control enviados por el protocolo de encaminamiento gestionados en la capa de red. La integración del descubrimiento de servicio y de encaminamiento nos permite descubrir un servicio y una ruta simultáneamente mediante el uso del mismo conjunto de mensajes. De ahí, que un esquema de integración reduce significativamente el ancho de banda y la latencia general para

descubrir un servicio y una ruta al servicio [117]. En las cabeceras de los paquetes de encaminamiento se puede enviar información de descripción de servicios junto con la propia información de rutas de encaminamiento.

Otras propuestas han combinado el proceso de descubrimiento de servicios con el proceso de encaminamiento empleando distintos protocolos, por ejemplo: [14] emplea DSR, [78] usa OLSR, [87] emplea MAODV, [85] lo combina con ODRMP, y [133] y [104] usan AODV.

2.10. Modelos de movilidad

En esta sección revisaremos los modelos de movilidad existentes para la simulación de redes móviles Ad Hoc. Estos modelos describen el comportamiento que tienen los nodos móviles dentro de los escenarios artificiales utilizados para simular situaciones reales.

Un modelo de movilidad describe el patrón de movimientos que siguen los nodos móviles e indica los cambios que irá teniendo en el tiempo respecto a su localización, velocidad, pausas y aceleración. Dado que los patrones de movimientos pueden jugar un papel significativo para determinar el rendimiento del protocolo, es deseable que los modelos de movilidad emulen de una manera razonable los patrones de movimientos de las aplicaciones del mundo real.

En años recientes, los investigadores han decidido basar la mayoría de sus resultados en el modelo de movilidad *Random Waypoint Mobility Model*. Sin embargo, muchas de las aplicaciones del mundo real requieren la utilización de diferentes modelos de movilidad. Por esta razón, se han desarrollado otros modelos de movilidad especialmente diseñados para modelar situaciones en particular.

Antes de describir los modelos de movilidad existentes, cabe mencionar que en general, existen dos tipos de modelos de movilidad: *modelos simulados* (traces) y *modelos sintéticos* [17] [18]. Los *modelos simulados* emplean patrones de movimientos muy aproximados a las situaciones del mundo real. Este tipo de modelo provee información precisa, especialmente cuando se simula gran número de nodos durante un largo periodo de observación. Sin embargo, los entornos

nuevos como las MANET no son fácilmente modelados cuando aún no se han creado los *modelos simulados*.

Un método intuitivo para crear patrones de movilidad realistas, podría consistir en construir modelos de movilidad basados en trazas, dado que es posible proporcionar la información exacta acerca de las trazas de los nodos. Sin embargo, ya que las MANET no han sido implementadas ni desplegadas en amplia escala, obtener la movilidad real viene a ser un reto mayor. Por consiguiente, en este tipo de situación se usan los *modelos sintéticos* que intentan representar de forma más realista, el comportamiento de los nodos móviles cuando aún no es posible emplear los *modelos simulados*. Es decir, los *modelos simulados* representan situaciones bien conocidas y fácilmente controladas, mientras que los *modelos sintéticos* en ocasiones modelan situaciones hipotéticas.

Esta es una lista de los principales modelos de movilidad sintéticos:

- *Random Walk Mobility Model*: Modelo simple de movilidad basado en direcciones y velocidades aleatorias.

- *Random Waypoint Mobility Model*: Modelo de movilidad ampliamente usado, se basa en que los nodos describen un movimiento aleatorio tanto en velocidad como destino final. Consiste en que cada nodo elije de forma aleatoria un destino y se desplaza hasta ese destino a una velocidad aleatoria entre 0 y [V-max]. Al llegar a su destino el nodo se detiene por un instante de tiempo [Pause] y nuevamente elije un próximo destino y una nueva velocidad.

- *Random Direction Mobility Model*: Modelo que fuerza a los nodos móviles a viajar al límite del área de la simulación, antes de cambiar de dirección y velocidad.

- *Boundless Simulation Area Mobility Model*: Modelo que convierte un área rectangular 2-D a un área de simulación en aro 3-D.

- *Gauss-Markov Mobility Model*: Modelo que usa un parámetro de ajuste para variar el grado de aleatoriedad en el patrón de movilidad.
- *Probabilistic Version of the Random Walk Mobility Model*: Modelo que utiliza un conjunto de probabilidades para determinar la siguiente posición de un nodo móvil.
- *City Section Mobility Model*: Modelo que emplea un área de simulación para representar las calles de una ciudad.

Por otra parte, podemos decir para generalizar que los modelos de movilidad caen en dos grupos principales: Modelos de movilidad de entidad y modelos de movilidad de grupos. Los modelos de movilidad de grupos permiten simular situaciones en las que las decisiones de movimientos de los nodos dependen de los otros nodos del grupo. A continuación mencionamos cinco modelos de movilidad de grupos:

- *Exponential Correlated Random Mobility Model*: Modelo de movilidad de grupo que usa una función para crear movimientos.
- *Column Mobility Model*: Modelo de movilidad de grupo en el que el conjunto de nodos móviles forman una línea y están uniformemente moviéndose hacia una dirección en particular.
- *Nomadic Community Mobility Model*: Modelo de movilidad de grupo en el que el conjunto de nodos móviles se mueven juntos de un lugar a otro.
- *Pursue Mobility Model*: Modelo de movilidad de grupo en el que el conjunto de nodos móviles siguen un determinado destino.
- *Reference Point Group Mobility Model*: Modelo de movilidad de grupo en que los movimientos del grupo están basados en el camino recorrido por un centro lógico. Existe un líder del grupo que se mueve a una velocidad [V-lider] y el resto del grupo se mueven junto con el líder a una velocidad

y dirección similar. La dirección y velocidad del resto del equipo se ajusta periódicamente a las variaciones que ejecuta el líder del grupo. Este modelo se caracteriza por tener una gran dependencia espacial.

El rendimiento de un protocolo de MANET puede variar significativamente con diferentes modelos de movilidad e incluso puede también variar cuando se usa el mismo modelo de movilidad y se emplean diferentes parámetros. De modo que es conveniente evaluar el rendimiento de un protocolo con el modelo de movilidad que más se aproxime al escenario del mundo real [17].

Por otra parte, es necesario tener en cuenta el continuo cambio de posición de los dispositivos, puesto que influye en la aptitud de los protocolos de redes Ad Hoc para que se comporten correctamente en entornos dinámicos. De modo que los modelos de movimientos son un aspecto importante en la simulación. Lo cual incluye entre otras cosas lo siguiente:

- La definición del área simulada en la que los movimientos de los nodos tendrá lugar y las reglas para modelar los nodos que se mueven más allá del área de simulación.
- El número de nodos en el área de simulación y la localización de nodos en el comienzo de la simulación.
- El modelo de movilidad en sí.

2.10.1. Aplicaciones de modelos de movilidad

Independientemente de las peculiaridades de los modelos de movilidad que hemos ya descrito, algunos de estos modelos pueden aplicarse particularmente a representaciones de distintas situaciones de acuerdo al comportamiento que se observa en el mundo real [140]. Estos escenarios o posibles aplicaciones se describen brevemente en la siguiente lista.

- *Event Coverage scenario* (event) .- Son aplicaciones para grupos de reporteros, camarógrafos y periodistas que mantienen distancias y movimientos aleatorios entre sí, mientras siguen un evento o un conferenciante. Este

comportamiento se modela mejor con el *Nomadic Community Mobility model*.

- *Military Combat scenario* (combat) .- Este patrón de movimientos se observa en un escenario de combate militar, se representa por el *Random Walk Mobility model*, ya que los tanques, helicópteros y soldados se mueven inesperadamente con velocidades y direcciones independientes, de ahí que existan paradas y virajes agudos repentinos.

- *Military March scenario* (march) .- En esta aplicación las entidades se mueven todas juntas hacia adelante. Por eso, el *Column Mobility model* es lo que mejor le va a este comportamiento de movilidad de grupo. Además, las entidades se mueven dentro de una pequeña vecindad alrededor de sus puntos de referencia (mientras sigan marchando hacia adelante con el grupo), de modo que se usa la combinación del *Random Waypoint model* con el *Column Mobility model* para representar el movimiento individual de las entidades.

- *Police Pursuit scenario* (pursuit) .- Este modelo de movilidad de persecución se aplica a escenarios como persecución policiaca. Habitualmente, los coches de policía persiguen a un sospechoso o a un criminal; la policía no se mueve aleatoriamente dentro del área de sus puntos de referencia individual debido a que su tarea principal es capturar a su objetivo.

- *Rescue Operation scenario* (rescue) .- Esta aplicación modela los equipos de rescate en acción. Un equipo de rescate usualmente se separa expandiéndose al mismo tiempo para encontrar sus objetivos. Este comportamiento es inducido por el *Random Direction model*.

- *Personal Area Networks scenario* (pan) .- Esta aplicación es una red de área personal, la describe mejor el *Random Waypoint model* ya que los nodos se mueven a destinos aleatorios con velocidades variables, se realizan

pausas para variar la duración y luego comienzan a viajar al siguiente destino. Esta aplicación elige sus propias velocidades y tiempos de pausas.

- *Collaborative Conference scenario* (conf) .- Es una aplicación de conferencias colaborativas que emplea el *Random Waypoint model*, aquí al igual que la aplicación (pan) los nodos se mueven libremente hacia destinos aleatorios a velocidades y pausas particulares de la aplicación.

- *Residential Mesh Networks scenario* (mesh) .- Se trata de una red en malla, la aplicación podría tratarse de una red residencial; el escenario emplea el modelo de movilidad *Random Waypoint model* puesto que los usuarios se mueven a hacia direcciones aleatorias a velocidades variables, también se considera la existencia de pausas.

- *Vehicle-Passenger scenario* (vp) .- Se trata de una aplicación para coches de pasajeros, en este tipo de escenarios los coches se mueven aleatoriamente hacia distintos destinos a velocidades y pausas variables. Lo que mejor describe a esta aplicación es el uso del *Random Waypoint model*.

- *Vehicle-Roadside scenario* (vr) .- Es una aplicación para vehículos que van por la carretera, por las características del escenario, los usuarios transitan libremente hacia direcciones aleatorias a distintas velocidades, es posible la existencia de pausas. Es conveniente aplicar el *Random Waypoint model* como modelo de movilidad.

2.11. Conclusiones

A lo largo de este capítulo hemos realizado un estudio del tema de descubrimiento de servicios en redes móviles Ad Hoc, se ha revisado el tema de los protocolos de encaminamiento para MANET, las técnicas de clustering en los protocolos de encaminamiento, la utilización del diseño Cross-layer en el descubrimiento de servicios en MANET y los modelos de movilidad que se emplean para simular escenarios de redes móviles Ad hoc.

En primer lugar, hemos presentado una clasificación de las arquitecturas de descubrimiento de servicios en función de la infraestructura de descubrimiento o tipo de directorio que emplea. Esta clasificación se resume en un cuadro que incluye las principales características de estos protocolos y en otro cuadro se esbozan los dos modelos de interacción entre clientes y proveedores respecto a una relación directa o indirecta.

Hemos descrito las arquitecturas de descubrimiento de servicios existentes en redes móviles Ad Hoc, a continuación de cada descripción de las arquitecturas, se ha presentado una lista con los principales protocolos de descubrimiento de servicios que caen en esa categoría, esta lista se componen exactamente de 35 protocolos de descubrimiento de servicios. En esa lista se han resumido de manera sucinta las características de los protocolos y su estado de desarrollo actual. Más adelante se ha presentado un estudio de la evolución de los protocolos de descubrimiento de servicios, mostrando gráficamente la cronología de la aparición de dichos protocolos. En la siguiente sección se ha realizado un análisis comparativo resaltando las técnicas de descubrimiento empleadas en los protocolos. Posteriormente, se analiza el funcionamiento general de los protocolos de descubrimiento de servicios en función del mecanismo de descubrimiento.

Una vez estudiado el estado actual del tema de descubrimiento de servicios en MANET, en este mismo contexto se abordan los retos actuales más importantes sobre este particular. Los retos pendientes por resolver y los temas de actualidad en el estudio de este campo.

Dejando a un lado el tema de descubrimiento de servicios y en vista de que nuestra solución se basa en diseño Cross-layer, es imperioso abordar el tema de protocolos de encaminamiento en MANET, para ello se revisan las tres categorías de protocolos de encaminamiento existentes (proactivos, reactivos e híbridos), se revisa a detalle el protocolo de encaminamiento reactivo AODV, por ser éste el protocolo en el que se basa la solución que proponemos. Se describen los formatos de mensajes que emplea AODV para enviar mensajes de petición y respuesta de servicios; y al final se da un ejemplo del funcionamiento de AODV.

Posteriormente se revisan las técnicas de clustering, así como los principales algoritmos que emplean estas técnicas; dado que nuestra solución utiliza un mecanismo de agrupamiento de nodos, mencionamos los beneficios que resultan de agrupar los nodos de una MANET en clusters, con el objetivo de optimizar recursos y alcanzar cierto grado de escalabilidad a la hora de jerarquizar la red.

Dedicamos una sección al diseño Cross-layer, ya que nuestra solución emplea este tipo de diseño para descubrir servicios en la capa de red y no en la capa de aplicación. De ahí la importancia de incluir este tema en este capítulo. Por esta razón resaltamos los beneficios de utilizar diseño Cross-layer en el proceso de descubrimiento de servicios en MANET, también se mencionan los inconvenientes de emplear el modelo de referencia OSI en las redes inalámbricas, según algunos autores.

Y finalmente, se estudia el tema de modelos de movilidad, muy importante para la elección y diseño de nuestro modelo de simulación, puesto que en este tema se describen los patrones de movilidad respecto a la localización, velocidad, pausas y aceleración de los nodos móviles. De igual manera se mencionan las principales aplicaciones de estos modelos y patrones de movimientos en determinadas situaciones del mundo real.

Capítulo 3

Propuesta para descubrir servicios en MANET

En el capítulo 2 hemos descrito las principales características de las MANET, el diseño en el que se basa su mecanismo de descubrimiento de servicios y en general, todo lo relacionado con la clasificación de los protocolos existentes de acuerdo a distintos criterios. Hemos incluido una descripción breve de una serie de protocolos de descubrimiento de servicios en redes Ad Hoc, que aportan distintas soluciones al tema que estamos tratando. Sin embargo, existen problemas importantes aún por resolver que no hemos mencionado.

En este capítulo se revisan los problemas latentes en los protocolos de descubrimiento de servicios. Aportamos nuestra visión de cómo resolver algunos problemas y planteamos nuestra propuesta de solución.

Una vez planteada nuestra solución, mencionamos el trabajo previo que nos han servido como punto de partida para nuestra propuesta. Más adelante definimos a detalle las extensiones a los formatos de mensajes del protocolo de encaminamiento que utiliza nuestro protocolo. Describimos el funcionamiento general del protocolo, el mecanismo de gestión de grupos y el mecanismo de descubrimiento de servicios.

3.1. Problemas en los protocolos de descubrimiento de servicios

En los protocolos de descubrimiento de servicios revisados se han encontrado soluciones interesantes para descubrir servicios en MANET, sin embargo, subyacen problemas que en ocasiones han sido generados en el intento de implementar soluciones. En esta sección revisamos algunos de los problemas actuales sobre este particular y ofrecemos de manera teórica soluciones al respecto.

- *Limitaciones de hardware* .- Por lo general los elementos participantes en las redes móviles Ad Hoc son dispositivos limitados, ligeros, reducidos en capacidades y pobres en recursos, y en algunos casos, dispositivos móviles ricos en recursos (tipo laptop). De modo que, un factor a considerar es la limitación de recursos de los dispositivos móviles, ya que consumen demasiada energía y memoria. Muchos de los protocolos de descubrimiento de servicios no consideran las limitaciones de los dispositivos tanto de batería, memoria, comunicación y cómputo. Por consiguiente, se deben diseñar protocolos de descubrimiento de servicios que se adapten a las restricciones de los protocolos inalámbricos y reduzcan el consumo de batería de los dispositivos limitados. Estos diseños deben ir orientados hacia protocolos de descubrimiento de servicios que operen con las restricciones típicas de los dispositivos y de las redes sin infraestructura.

- *Directorio centralizado* .- En algunos de los protocolos de descubrimiento de servicios revisados existe un dispositivo central funcionando como repositorio o directorio, en el que se lleva el registro de los servicios disponibles en la red (páginas amarillas). Lo anterior genera una fuerte dependencia hacia el servidor de directorios. Evidentemente, para eliminar esta dependencia entre nodos, se requiere un esquema de diseño descentralizado o un sistema de replicación de servidores que garantice en todo momento la disponibilidad de esos registros y de esta manera ser más tolerante a fallos. Se sugiere incluir en cada dispositivo, una plataforma ligera con una

memoria caché local para tener acceso a la lista de servicios disponibles en la red. Es decir, un directorio distribuido y descentralizado con cierto grado de autonomía.

- *Uso excesivo de transmisiones* .- Algunos protocolos de descubrimiento de servicios no emplean directorios para almacenar información de servicios; en cambio, los clientes envían un exceso de transmisiones durante el proceso de búsquedas de servicios y por parte de los proveedores para el anuncio periódico de servicios. Este exceso de transmisiones origina un alto consumo de batería, la energía es un recurso muy crítico en la mayoría de dispositivos limitados. Los protocolos de descubrimiento de servicios que mantienen actualizados oportunamente los registros de los servicios disponibles en la red, consumen demasiados recursos, entre ellos ancho de banda; envían una considerable cantidad de mensajes, ya sea a través de broadcast, multicast o unicast, problema conocido como tormenta de broadcast. Problema originado en parte debido a la movilidad de los dispositivos y a las características inherentes al medio y al entorno ubicuo. Es conveniente encontrar un equilibrio entre el envío de anuncios y peticiones de servicios, la periodicidad de los envíos de mensajes y el tiempo de vida de los anuncios en los registros de servicios.

- *Escalabilidad* .- Esta capacidad puede ser vista como un problema cuando se pretende escalar una red y el rendimiento y la calidad de los servicios se ven considerablemente disminuidos. En general, los protocolos revisados se pueden aplicar tanto ámbitos globales como locales. Algunos protocolos con el objetivo de escalar la red, registran en un directorio de servicios la información de los proveedores de servicios, posteriormente, cuando los clientes requieren algún servicio en particular recurren a consultar este directorio. En MANET no es recomendable el uso de servidores centralizados, debido a que las redes son altamente dinámicas y en algunos casos, los dispositivos miembros de este tipo de redes tienen capacidades de al-

macenamiento muy limitadas, por lo que es muy difícil y costoso que uno de estos dispositivos asuma el rol de proveedor de servicios. Lo anterior es en cuanto a ámbitos globales. Ahora, que tratándose de ámbitos locales, el problema de escalabilidad no es prioritario debido al ámbito local de aplicación.

- *Uso de cachés* .- Como una medida para minimizar el número de transmisiones, a menudo algunas soluciones emplean cachés, de esta forma cada dispositivo almacena localmente en ellas, la información de los servicios que se transmiten por la red, de modo que cuando un cliente busca un servicio, previamente a la transmisión del mensaje de petición, el dispositivo verifica dentro de la caché y si la información del servicio es encontrada, entonces no se transmite ninguna petición de servicios. En ocasiones el problema que conlleva el uso de cachés, consiste en que algunas de las entradas almacenadas se correspondan con servicios falsos, es decir, con servicios que ya no se encuentran disponibles en la red. Este problema se agudiza en redes altamente dinámicas, para ello se deben implementar mecanismos de consistencia de cachés que no aumenten en exceso el tráfico que esto genera. Un mecanismo es el tiempo de vida, que indica el tiempo que un servicio permanece disponible en la red.

- *Heterogeneidad* .- Se deben superar dos tipos de problemas de heterogeneidad para poder proporcionar interoperabilidad: *i.* heterogeneidad de protocolos de descubrimiento de servicios, en este respecto se debe ser capaz de interactuar con una gran variedad de protocolos y *ii.* heterogeneidad en la interacción entre protocolos y servicios, ya que una aplicación implementada sobre un protocolo de descubrimiento de servicios no puede interoperar con servicios desarrollados en otro protocolo de descubrimiento de servicios [15], lo cual hace difícil predecir en el momento del diseño, todos los requerimientos necesarios para ejecutar un programa sobre un entorno de ejecución desconocido [16]. Sin embargo, no importa que SDP esté pre-

sente, los nodos deben descubrir e interactuar con los servicios disponibles en el vecindario. El objetivo es que los protocolos de descubrimiento de servicios permitan a los dispositivos descubrir y usar los servicios presentes en la red, sin la necesidad de conocer previamente su localización específica y permitiendo la interoperación con servicios que provengan de otros protocolos de descubrimiento de servicios.

A continuación enunciamos las distintas alternativas que se plantean para solucionar de manera dinámica los problemas existentes en el descubrimiento de servicios [20]. Estas soluciones centran sus esfuerzos en minimizar el consumo de ancho de banda, en reducir el consumo de recursos (energía, memoria, cómputo), en disminuir el uso excesivo de transmisiones y en la eliminación de la figura del directorio centralizado.

- Solución uno .- Este esquema de solución consiste en la existencia de nodos desempeñando el rol de mediadores (denominados intermediarios) entre clientes y servidores. De esta manera los clientes descubren los servicios ofrecidos por los servidores a través de estos intermediarios. Sin estos intermediarios los clientes no son capaces de descubrir los servicios disponibles en la red. En primera instancia los servidores registran sus servicios en los directorios de uno o varios intermediarios. Los clientes solicitan a uno a varios intermediarios la búsqueda de un servicio específico. Los intermediarios buscan en sus directorios o registros, si alguna de las entradas se corresponde con el servicio solicitado, el intermediario devuelve al cliente la información correspondiente del servicio solicitado. En esta solución tanto clientes como servidores descubren dinámicamente a los intermediarios.

- Solución dos .- El descubrimiento pasivo (denominado push). En esta solución los servidores de servicios anuncian periódicamente los servicios que ofrecen. Los clientes por su parte, almacenan estos anuncios en una caché local, para que posteriormente cuando un cliente solicite un servicio,

sea su caché local el primer sitio en el que compruebe si se trata de uno de los servicios anunciados previamente.

Con el método de descubrimiento push se mantiene información en casi todo momento sobre la presencia en la red de ciertos servicios; pero cada nodo recibe y procesa mensajes no esperados o no requeridos en un momento determinado, lo que ocasiona un alto consumo de energía, memoria y ancho de banda.

En el modo push los servidores envían los anuncios de acuerdo a un tiempo de periodicidad que previamente informan al cliente, cuando el cliente no recibe un anuncio dentro del tiempo previsto, el anuncio es eliminado de la entrada de su caché local. El tiempo de vida de un servicio en el registro local del cliente, se corresponde con el periodo de tiempo de la serie de anuncios que envía el servidor. Durante el envío de la serie de anuncios consecutivos por parte del servidor, pueden ocurrir caídas del propio servidor o ciertas modificaciones, que el cliente no detectará hasta el próximo anuncio. Por consiguiente, una medida es anunciar con mayor frecuencia sus servicios, lo que ocasionará un mayor consumo de ancho de banda con el consecuente problema de escalabilidad del protocolo.

- Solución tres .- En el método de descubrimiento activo (denominado pull), un nodo que demanda un servicio difunde mensajes (bien por broadcast o bien por multicast) a los nodos de la red en busca de solución para su petición y responden por unicast los servidores que ofrecen dicho servicio. Este método tiene problemas de escalabilidad en redes con gran número de nodos debido a un alto consumo de energía, memoria y sobre todo, de ancho de banda.

  El método de descubrimiento pull introduce cachés de servicios remotos en cada cliente, así, un cliente almacena localmente los servicios que va descubriendo. De tal forma que cuando un cliente desee acceder a un servicio, antes de enviar la petición, primero comprueba en su caché local si

tiene almacenada información de ese servicio. Los servidores anuncian sus servicios incluyendo un tiempo de vida asociado al servicio, que sirve para indicar al cliente, el tiempo máximo que el servicio puede estar almacenado en la caché. Lo anterior es para evitar la incertidumbre por saber si el servicio almacenado en la caché continúa estando disponible en la red.

Es importante gestionar adecuadamente las limitaciones de recursos de los dispositivos, por eso existe la preocupación de una óptima gestión de mensajes. Si bien no se puede desatender la actualización periódica de los cachés, tampoco se puede abusar del uso no controlado de tráfico de mensajes. Se propone entonces para cada modo de funcionamiento la siguiente estrategia:

- Que el modo *centralizado* forme clusters con objeto de aumentar la escalabilidad, aun cuando se sabe que el punto débil de este modo de funcionamiento es la dependencia de un único punto de fallo, puesto que la caída de un intermediario hace imposible que los clientes puedan descubrir servidores. Para evitar la situación anterior, es recomendable la presencia de uno o varios líderes en cada cluster.

- Que el modo *push* no sature de anuncios la red y que no permita transcurrir demasiado tiempo sin refrescar los cachés locales; para ello debe implementar una agrupación o segmentación de nodos atendiendo criterios geográficos o a una clasificación de servicios.

- Que el modo *pull* solicite servicios solamente a los nodos miembros de grupos específicos o realice búsquedas más selectivas en relación a la información registrada en su caché local, o bien, si existen grupos o zonas, enviar peticiones a los nodos más cercanos.

3.2. Propuesta de solución

Por definición una MANET está formada por dispositivos con capacidades limitadas, equipos pobres en recursos como batería, memoria, cómputo, etc. Pero

también es cierto que a las MANET se conectan dispositivos heterogéneos y que algunos cuantos de esos dispositivos poseen capacidades superiores (HCD, High Capability Devices).

El presente trabajo de investigación propone una arquitectura de descubrimiento de servicios en redes móviles Ad Hoc, basada en dispositivos con capacidades superiores que lideran clusters, el diseño combina el enfoque basado en cluster con el basado en diseño Cross-layer. Esta combinación tiene como propósito involucrar el descubrimiento de servicios en el proceso de encaminamiento. Empleamos los HCD que existan en la red para liderar y dar soporte a la gestión de grupos, mientras que los dispositivos de capacidades limitadas no desgastan sus reducidos recursos.

El diseño que proponemos se denomina **LIFT** (**LI**mited **F**looding of requests within a clus**T**er) porque intenta difundir mensajes únicamente dentro del cluster, con el objetivo de optimizar el consumo de energía, memoria, procesos de cómputo, mensajes y ancho de banda, gracias al uso eficiente de recursos.

En la figura 3.1 mostramos una red móvil Ad Hoc integrada por 100 nodos, en dicha red existen tanto dispositivos con capacidades superiores como dispositivos con capacidades limitadas. Del mismo modo se muestra que se pueden formar clusters con los nodos y que estos pueden ser liderados por los dispositivos con capacidades superiores.

LIFT se fundamenta en aquellos protocolos de descubrimiento de servicios que emplean intermediarios. A diferencia de estos protocolos, el eje central de la arquitectura propuesta consiste en aprovechar la existencia de algunos dispositivos que poseen capacidades superiores (cómputo, comunicación y energía) con el objetivo de ser elegidos intermediarios en el proceso de descubrimiento de servicios. Por ello es necesario distinguir entre dispositivos con capacidades superiores y dispositivos con capacidades reducidas. Una vez hecha la distinción, es posible una clasificación en base a sus capacidades.

Las capacidades de los dispositivos se obtienen dadas las características de hardware que ellos mismos poseen, es decir, inicialmente un dispositivo conoce

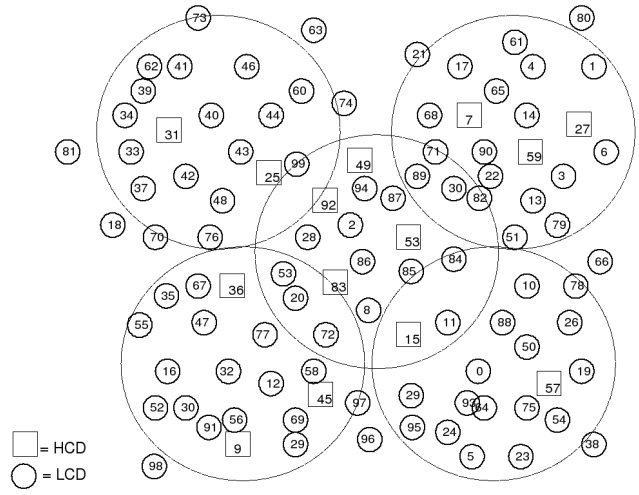


Figura 3.1: Ejemplo de una MANET que contiene HCD y LCD formando clusters.

sus propias características respecto a la cantidad de memoria, energía, capacidad de cómputo, potencia de señal y movilidad. Por lo tanto, la capacidad de un dispositivo se asigna en función del tipo de dispositivo que se trata.

En LIFT no adoptamos un algoritmo de clustering específico para elegir líder o clusterhead del cluster, asumimos un modelo de clustering general, en el que el líder puede seleccionarse en base a algún criterio, como puede ser el del algoritmo Lowest ID [4] o el del Weighted Clustering algorithm [27]. No nos concierne la formación del cluster en sí, sino solamente nos interesa los roles de los líderes, que es lo más importante a la hora formar clusters de nodos y en procurar que esos nodos tengan conectividad. Prácticamente nuestro interés por contar con la figura del clusterhead tiene como objetivo liderar los clusters. En nuestro modelo, el criterio para la elección de líder de clusters se basa en la selección de los dispositivos con capacidades superiores, en este caso la capacidad de un dispositivo se determina por medio de una estrategia sencilla que asigna un identificador binario a cada dispositivo. Es decir, si se trata de un dispositivo con capacidades superiores es un HCD, de lo contrario es un LCD.

Por otra parte, tenemos conocimiento de algunos algoritmos utilizados para la elección de líder en las técnicas de clustering. Estos algoritmos por lo general eligen líderes en función del número de saltos de distancia entre los nodos, otros algoritmos eligen clusterheads basados en pesos de conectividad o en el mayor o menor valor del identificador (ID), entre otras heurísticas. Estos son algunos de esos algoritmos: Highest-Degree heuristic (clustering basado en conectividad) [59] [118], Lowest-ID heuristic (clustering basado en identificador) [4] [5], Node-weight heuristic (clustering basado en el peso del nodo) [9] [8], Weighted Clustering algorithm [27], Distributed Weighted Clustering Algorithm (algoritmo distribuido de clustering basado en peso) [34], entre otros, (en el apartado 2.8.1 se da un breve vistazo a los más representativos).

Como se puede ver, existe una gran variedad de algoritmos para elegir y determinar a los dispositivos que lideran los clusters. La presente investigación no tiene entre sus objetivos proponer un nuevo algoritmo de clustering, ni adoptar un algoritmo de los mencionados en el párrafo anterior, simplemente nos limitamos a suponer que existen dos tipos de dispositivos: uno que tiene capacidades superiores con fuertes posibilidades de liderar clusters y otro que no será considerado para esta tarea.

Para definir la capacidad de un dispositivo LIFT tiene en cuenta las siguientes suposiciones:

- En principio cada dispositivo es consciente del grado de la capacidad que posee.
- El grado de capacidad de un dispositivo puede ser sólo de dos tipos: alta (HCD) o limitada (LCD).
- Conforme transcurre el tiempo, las capacidades de un dispositivo se van degradando como consecuencia del consumo de recursos.

En LIFT los nodos que tienen la característica de ser HCD se pueden declarar líderes de clusters. Esto es, un nodo inicialmente conoce sus prestaciones, por lo que puede ser de tipo HCD o LCD. Un HCD pueden convertirse

en líder de un cluster porque a la hora de crear y difundir las tablas de servicios, llevan en la cabecera de los mensajes SREQ, SREP y CHELLO (estos formatos se ven más adelante en el apartado 3.4.1), el campo *tipo de nodo* que los identifica como nodos HCD. En las entradas de la tabla de servicios de cada nodo queda registrado el campo correspondiente con el valor 1 para HCD y 2 para LCD. Posteriormente, durante las peticiones y respuestas de servicios, los HCD que luego se convierten en CL, se encargan de brindar soporte en este proceso.

En trabajos previos a LIFT se han propuesto una diferenciación entre los dispositivos de una MANET. Enseguida mencionaremos unos ejemplos de trabajos en los que no emplean un algoritmo de clustering en particular para elegir clusterheads, debido a que centran su interés en el proceso posterior al mecanismo de elección del líder de cluster. Y asumen que se puede adoptar algún algoritmo de clustering de los existentes.

Un ejemplo se puede encontrar en el trabajo: *At Home Anywhere Service Discovery Protocol* (SDP@HA) [146], que supone la existencia de dispositivos de tres categorías: dispositivos 3C (3 + cent) aplicaciones muy simples, 3D (3 + dollar) aplicaciones de complejidad media y dispositivos 300D (300 + dollar) aplicaciones con capacidades superiores, a estos últimos se les delega mayor carga de trabajo.

En [146] manejan el grado de capacidad de un dispositivo en función de su precio, mientras que en nuestra propuesta lo hacemos en función de las prestaciones de energía, memoría, comunicación y cómputo. En SDP@HA suponen la existencia de tres tipos de dispositivos: 3C, 3D y 300D, en LIFT sólo suponemos la existencia de dos tipos, HCD y LCD. A los dispositivos de 300D y HCD se les delega mayor carga de trabajo y se excluye al resto de dispositivos para esta tarea. Al igual que en LIFT, en SDP@HA tienen en cuenta el factor de agotamiento de recursos o no disponibilidad de un *Central* (nodo repositorio de servicios, generalmente representado por un dispositivo 300D), para lo cual inician un proceso de elección de un nuevo líder.

El siguiente trabajo es: *Cluster-based service discovery for heterogeneous wireless sensor networks* [98], en esta propuesta consideran que los algoritmos que hemos venido mencionando no son convenientes para su objetivo de descubrir servicios. Primero, elegir clusterheads basándose en información de nodos localizados a múltiples saltos de distancia, ocasiona una alta sobrecarga y una lenta reacción frente a los cambios de topología. Segundo, mantener información completa intra-cluster respecto a las capacidades de los nodos, resulta una tarea muy costosa para los dispositivos pobres en recursos. Y tercero, la complejidad de los algoritmos de clustering multi capa requieren mucho esfuerzo para construir y mantener la estructura deseada. En dicho trabajo se interesan en una solución de clustering más sencilla, una solución que permita reaccionar rápidamente a los cambios topológicos y que requiera poco esfuerzo de construcción y mantenimiento.

El algoritmo de clustering simple del que hablamos, consiste en crear árboles de nodos que se van formando según el grado de capacidad de cada nodo, los nodos van eligiendo como *padre* al nodo vecino con mayor grado de capacidad que otro, y así sucesivamente hasta formar árboles. Y eligen como *raíz* o *clusterhead* al nodo que posea el mayor grado de capacidad de todos los nodos. Para ello ocupan un mensaje denominado *SetRoot*, que se usa para propagar la dirección del nodo *raíz* a todos los miembros de los clusters. La descripción formal de la construcción de los clusters y la fase de inicialización se resume a continuación.

- Los nodos que tienen el grado de capacidad más alto de entre sus vecinos, se declaran a sí mismos *clusterheads* y difunden un mensaje *SetRoot* anunciando sus roles.

- Los nodos restantes eligen como *padre*, al vecino con el grado de capacidad más alto.

- Cuando un nodo recibe un mensaje *SetRoot* de su *padre*, se entera de que es miembro del cluster y re-difunde el mensaje *SetRoot*.

Otro trabajo es: *A Cluster Based Service Discovery Model for Mobile Ad hoc Networks* [3], en el que no utilizan un algoritmo de clustering específico de los algoritmos formales mencionados en el apartado 2.8.1, lo mismo pueden adoptar cualquiera de ellos. Se centran especificamente en el rol del clusterhead dentro del esquema de clustering y por la definición del tamaño del cluster. Suponen que el tamaño del cluster está en función del número de saltos h que se encuentra un nodo de distancia de su clusterhead. Es decir, que un nodo perteneciente a un cluster debe estar a más de h saltos de distancia de su clusterhead. En cambio, adoptan la restricción de que dos clusterheads no deben estar a menos de dos saltos de distancias el uno al otro.

Hemos visto ejemplos en los que al igual que LIFT, el mecanismo de elección de líder no se basa en los algoritmos formales para este propósito. Sino que emplean otros medios más sencillos para declarar sus líderes de clusters. Comparando el criterio de LIFT para determinar líder de cluster con el criterio de los algoritmos formales para elegir clusterheads, creemos que los HCD de LIFT se corresponden con los ID, pesos o grados de esos algoritmos.

Volviendo a LIFT, en este diseño todos los nodos de la red ofrecen servicios, pero los HCD son los encargados de brindar soporte para descubrir servicios. Todos los nodos están al mismo nivel, es decir, todos los nodos ofrecen servicios y no existen nodos sin proveer servicios, no se depende de ningún nodo para el funcionamiento del resto. Si un nodo no puede proveer el servicio solicitado siempre existe otro nodo que lo puede hacer, por tanto no existen nodos imprescindibles para la red.

Como es bien sabido, por varias razones ampliamente expuestas en la literatura básica de este tema, en las MANET no deben existir nodos críticos que jueguen el papel de servidor central. Habitualmente se pueden conectar a la red nodos con capacidades restringidas y otros con capacidades superiores; de esta manera, es posible aprovechar la existencia de este último tipo de nodos, para asignar más carga de trabajo con el propósito de minimizar el consumo de memoria, energía, procesos, búsquedas y ancho de banda. También es im-

portante tener en cuenta que nadie puede garantizar la fiabilidad de los nodos con capacidades superiores en un momento determinado, ya que estos pueden desconectarse, moverse o agotar sus recursos.

3.2.1. Patrón de interacción de la propuesta

Presentamos en la figura 3.2 el patrón de interacción que describe de manera gráfica el proceso de descubrimiento de servicios de nuestra propuesta. Dicho patrón de interacción está integrado por dos únicos objetos y una serie de eventos y mensajes. Uno de los objetos que intervienen es el nodo clusterizado (**CN**, Cluster Node) o *nodo ordinario*, que es un dispositivo con capacidades limitadas (LCD), este tipo de nodo anuncia sus servicios por broadcast y estos servicios son almacenados en tablas de servicios. El otro objeto presente en el patrón de interacción es el Cluster Leader (**CL**) o *clusterhead*, que es esencialmente un dispositivo con capacidades superiores (HCD). Tanto el objeto CN como el objeto CL son mostrados en las cajas superiores. El objeto que comienza teniendo el control es el CN, que es el que envía el primer mensaje con la descripción del evento: *ANUNCIO de servicio (broadcast)*. Este mensaje es recibido por el objeto CL, quien ejecuta el evento *Almacena el ANUNCIO en la tabla de servicios de LIFT*; las tablas de servicios son los repositorios en los que los CL registran y comprueban la concordancia de patrones de la descripción de servicios tanto de los servicios que proveen los CL como de los servicios que ofrecen los CN. Una vez registrado el servicio en las tablas, el objeto CL tiene el control e invoca el evento *Concordancia de descripción de servicio* y el evento *HCD contiene información de la concordancia de descripción de servicio*. Posteriormente, el objeto CN tiene el control y genera el evento para enviar el mensaje *Envía PETICIÓN de servicio (multicast)* al objeto CL solicitando una petición de servicios. Finalmente, el objeto CL responde con el mensaje *Envía RESPUESTA de servicio (unicast)* devolviendo por unicast la respuesta de servicio.

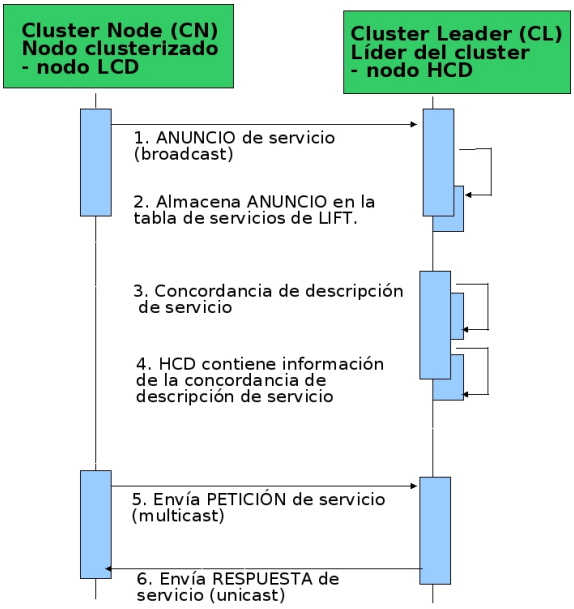


Figura 3.2: Patrón de interacción de la propuesta

3.3. Punto de partida

LIFT mezcla las ideas del modelo de interacción directa con el modelo de interacción indirecta y lo combina con una solución de descubrimiento de servicios que emplea un diseño Cross-layer basándose en AODV (Ad Hoc On-Demand Distance Vector) [50] como protocolo de encaminamiento; AODV es ideal para descubrir rutas en redes Ad Hoc. La decisión de emplear Cross-layer como técnica para descubrir servicios responde a la siguiente razón: durante el proceso de encaminamiento se van descubriendo los servicios disponibles en el vecindario; optimizando con ello el número de mensajes necesarios en el proceso.

El diseño del protocolo de descubrimiento de servicios que proponemos, toma como punto de partida la idea de AODV-SD[1] [95] [148], que es una versión

[1]AODV-SD es un protocolo de descubrimiento de servicios propuesto por García y Arias en el trabajo titulado: Service Discovery in Mobile Ad Hoc Networks: Better at the Network Layer?, presentado en el International Conference on Parallel Processing Workshops en 2005. AODV-SD descubre servicios basándose en un esquema Cross-layer empleando AODV como protocolo de encaminamiento.

extendida del protocolo AODV, a este planteamiento LIFT le añade una solución de interacción indirecta basada en clustering.

La característica principal de LIFT radica en la combinación de un *diseño Cross-layer* con un *diseño de clustering*. Dado que LIFT supone dos hechos: primero, que una MANET se integra por dispositivos heterogéneos y segundo, que se pueden diferenciar a los dispositivos que tienen capacidades superiores de los que tienen capacidades reducidas, por lo que selecciona y elige a los dispositivos con capacidades superiores, con el fin de establecerlos líderes de los clusters. La función de los líderes es dar soporte durante el proceso de descubrimiento de servicios.

Volviendo al tema de AODV-SD, es un diseño Cross-layer que descubre servicios en una MANET usando el protocolo de encaminamiento AODV [84]. AODV-SD descubre servicios en la capa de red y no en la capa de aplicación en la que tradicionalmente se efectúa; AODV-SD demuestra que los servicios se pueden descubrir con mayor rapidez y con menor tráfico de control.

AODV-SD modifica las cabeceras de los formatos de mensajes RREQ, RREP, Service URL Request, HELLO y tabla de encaminamiento de AODV. La modificación consiste en añadir un campo denominado *service-string* utilizado para registrar la información de los servicios que ofrecen algunos nodos. Por consiguiente, también se modifican las acciones efectuadas por los nodos.

El protocolo AODV-SD aumenta la eficiencia de descubrimiento de servicios gracias a que monta la información de servicios en los mensajes de control de la capa de red, es decir, lleva a cabo un piggybacking con la descripción de servicios al aprovechar los mensajes de control. De modo que cuando un nodo quiere descubrir servicios, envía un mensaje de petición de servicios RREQ con los campos adicionales para obtener la dirección IP del proveedor de servicios así como la ruta de ese nodo.

AODV-SD emplea tablas de encaminamiento que además de la información de las rutas, contienen información relacionada con los servicios que proveen los dispositivos; de esta manera, cuando un nodo requiere un servicio entonces debe

realizar una búsqueda en su tabla local. Cada entrada en la tabla contiene la dirección IP, el identificador de servicios, el tiempo de vida y el servicio ofrecido.

Ahora bien, es preciso aclarar que AODV-SD descubre servicios, pero no distingue tipos de nodos, ni forma clusters, ni desde luego emplea líderes. AODV-SD gestiona todos los nodos por igual, da un trato igualitario a todos los nodos; AODV-SD sigue el modelo de interacción directa, pertenece al tipo de SDP que inundan la red de mensajes de anuncios y peticiones de servicios, generando un alto consumo de memoria, energía, ancho de banda y procesos. Sin embargo, lo interesante de AODV-SD es que se basa en un diseño Cross-layer. De ahí nuestro interés por su estrategia de descubrimiento de servicios.

Hasta ahora hemos descrito todo lo que LIFT toma de AODV-SD, su diseño Cross-layer para descubrir servicios en la capa de red, los formatos de mensajes modificados y la tabla de servicios. A partir de este punto, LIFT añade el campo *Node Type*[2] a los formatos de mensajes previamente modificados por AODV-SD, por lo que se puede afirmar que hasta aquí termina la influencia de AODV-SD sobre LIFT. Este es el punto donde termina AODV-SD y comienza LIFT. Nuestra solución propone la estrategia de diferenciar los tipos de dispositivos presentes en la red, con el fin de formar clusters que han de ser liderados por los dispositivos que posean capacidades superiores. Por lo tanto, esto último es exclusivo de LIFT.

Con lo tomado de AODV-SD, con la distinción de tipos de dispositivos y el mecanismo de clustering que vamos a implementar, suponemos que se optimiza el proceso de descubrimiento de servicios y se obtiene un máximo rendimiento.

3.4. Diseño de LIFT

En esta sección se describen todos los formatos de mensajes utilizados por LIFT (apartado 3.4.1), que como hemos estado diciendo, son extensiones a los formatos de mensajes de AODV y de AODV-SD. En el apartado 3.4.2 se descri-

[2]El objetivo del campo *Node Type* es registrar el tipo de dispositivo y así diferenciar si se trata de un dispositivo con capacidades superiores o con capacidades reducidas.

ben las tablas de servicios que emplea LIFT en las que los dispositivos almacenan la información de servicios. Posteriormente en el apartado 3.4.3 se proporciona una explicación detallada acerca del funcionamiento de LIFT, en el apartado 3.4.3.1 se describe la forma en que LIFT realiza la gestión de grupos, y por último, en el apartado 3.4.3.2 se incluye la explicación del mecanismo de descubrimiento de servicios.

3.4.1. Formatos de mensajes

Las extensiones que presentan los formatos de mensajes de LIFT usan el formato Type-Length-Value (TLV) de tipos de 8-bits y son los mismos que presenta AODV en su Internet-Draft [124] [84]. Es decir, las extensiones de los campos deben definirse primero por un *Type* que es el tipo de dato. *Length* que es la longitud de la extensión y *Value* que es el valor del campo.

Como ya habíamos mencionado, nos interesa clasificar los dispositivos de acuerdo a sus prestaciones y designarlos líderes de los clusters. Para tal efecto, LIFT añade un campo a las extensiones hechas por AODV-SD. Añadimos el campo *Node Type* a los formatos de mensajes SREQ (figura 3.3), SREP (figura 3.4), Service URL Request (figura 3.5) y HELLO, con ello, la tabla de servicios ahora también contiene el tipo de dispositivo que se trata; además del identificador de servicios, la dirección IP, el tiempo de vida y la lista de los servicios ofrecidos. El campo *Node Type* va acompañado de los correspondientes campos *Type* y *Length* siguiendo las recomendaciones del Internet-Draft de AODV.

El formato del mensaje **Service Request** (SREQ) que se observa en la figura 3.3 tiene como propósito llevar la información de una *petición de servicios* al nodo destino que provee finalmente el servicio solicitado, el paquete SREQ puede pasar por varios nodos intermedios antes de alcanzar el nodo destino. En el paquete SREQ se transportan datos de control útiles para la retransmisión de mensajes por parte de los nodos intermedios y para el envío de vuelta de la respuesta del nodo destino. Este formato de mensaje contiene los siguientes campos de cabecera: *Type* es el tipo de dato. *J* es la Join flag, reservado para

multicast. *R* es la Repair flag, reservado para multicast. *G* es el Gratuitous RREP flag, indica si se debe enviar un gratuitous RREP en modo unicast al nodo indicado en el campo dirección IP destino. *D* es la bandera de destino, se indica sólo cuando el destino puede responder a este SREQ. *U* es un flag que indica que el número de secuencia es desconocido. *Reserved* se envía como 0, ignorado en recepción. *Hop Count* es el número de saltos desde la dirección IP origen al nodo que gestiona la petición. *RREQ ID* es un número de secuencia que identifica únicamente el SREQ cuando es tomado en conjunto con la dirección IP del nodo origen. *Destination IP Address* es la dirección IP del destino para la ruta deseada. *Destination Sequence Number* es el último número de secuencia recibido en el pasado por el origen para cualquier ruta hacia el destino. *Originator IP Address* es la dirección IP del nodo que origina el Service Request. *Originator Sequence Number* es el número actual de secuencia del origen. *Type* es el tipo de dato de Node Type. *Length* es la longitud de Node Type y *Node Type* es el campo para el tipo de nodo en el que se registra el tipo de nodo que se trata y con el que se identifica el tipo de nodo.

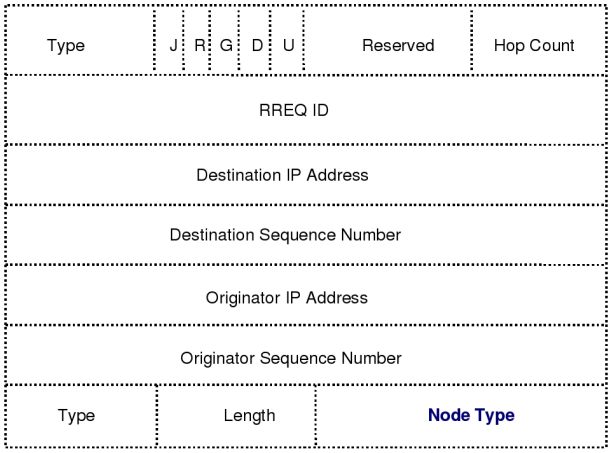


Figura 3.3: Formato del mensaje SREQ

El formato de mensaje **Service Reply** (SREP) ilustrado en la figura 3.4 tiene como propósito llevar la información referente a la *respuesta de servicio*, este paquete es generado por el nodo destino proveedor del servicio solicitado

por el nodo peticionario. El mensaje SREP lleva de vuelta los datos de control que le serán útiles a los nodos intermedios y al nodo solicitante. El paquete SREP contiene los siguientes campos de cabecera: *Type* es el tipo de dato. *R* es la Repair flag, usado para multicast. *A* cuando se requiere un mensaje de Acknowledgment (reconocimiento de peticiones) y se activa cuando se quiere comprobar si un enlace es unidireccional. *Reserved* si se envía 0, se ignora cuando es recibido. *Prefix Size* si no es cero, este prefijo especifica que el siguiente salto puede ser usado por cualquier nodo con el mismo prefijo de encaminamiento. *Hop Count* es el número de saltos desde la dirección IP del origen hasta la dirección IP del destino. *Destination IP Address* es la dirección IP del destino. *Destination Sequence Number* es el número de secuencia del destino asociado a la ruta. *Originator IP Address* es la dirección IP del nodo que ha originado el SREQ. *Lifetime* es el tiempo en milisegundos de los que los nodos reciben el SREP considerando que la ruta es válida. *Type* es el tipo de dato de Node Type. *Length* es la longitud de Node Type y *Node Type* es el campo para el tipo de nodo en el que se registra el tipo de nodo que se trata y con el que se identifica el tipo de nodo.

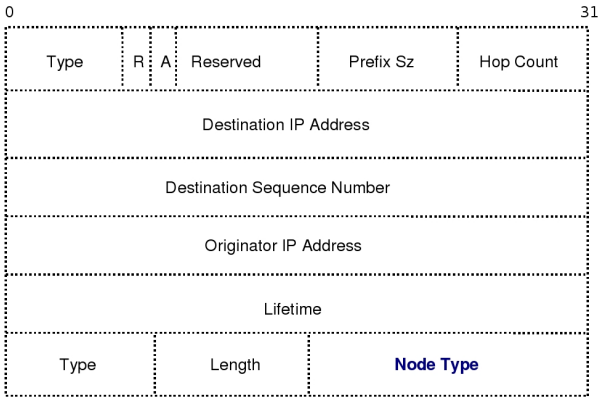


Figura 3.4: Formato del mensaje SREP

El formato de mensaje de **Service URL Request** que se muestra en la figura 3.5 tiene como finalidad describir el tipo de servicio, la descripción de servicio y el tipo de nodo que posee un dispositivo en caso de ser proveedor de

servicios. Este formato de mensaje contiene la siguiente cabecera: el *Type* que es el tipo de dato. *Length* es la longitud de la extensión. *service-type length* es la longitud de la cadena de service-type. *service-type* es el tipo de servicio. *Service Request Predicate* es la cadena con la descripción de servicios. *Type* es el tipo de dato de Node Type. *Length* es la longitud de Node Type y *Node Type* es el campo para el tipo de nodo en el que se registra el tipo de nodo que se trata y con el que se identifica el tipo de nodo.

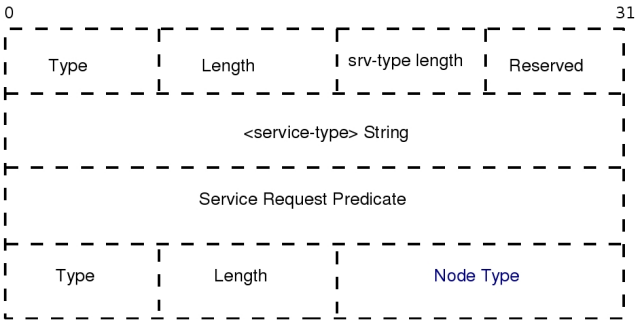


Figura 3.5: Formato del mensaje Service URL Request

El formato de extensión de **Service Port Request** se muestra en la figura 3.6. Donde *Type* es el tipo de dato. *Length* es la longitud de la cadena port y *# Port* es el número de puerto de TCP o UDP en el que reside la aplicación del servicio.

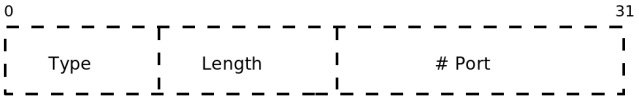


Figura 3.6: Formato del mensaje Service Port Request

Formato de mensaje **Route Reply Acknowledgment** (RREP-ACK) en la figura 3.7 se debe enviar en respuesta a un mensaje SREP con el bit *A*, se hace típicamente cuando existe peligro de enlaces unidireccionales para prevenir que se complete el ciclo de descubrimiento de servicios, se activa cuando se quiere comprobar si un enlace es unidireccional. *Type* es el tipo de dato y *Reserved* es cero, se ignora cuando se recibe.

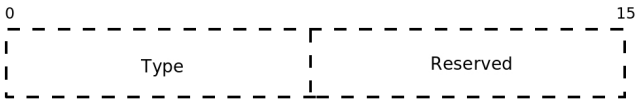


Figura 3.7: Formato del mensaje Route Reply Acknowledgment

El formato de mensaje **Route Error (RERR)** mostrado en la figura 3.8 es otro de los mensajes usados por LIFT, pero este paquete se utiliza tal cual es propuesto por AODV, es decir, no le hemos añadido ningún campo, por lo que se conserva y se utiliza igual como lo emplea AODV. El formato de la cabecera de RERR tiene como propósito invalidar una ruta. El paquete RERR llega hasta el origen de la ruta con el fin de que se comience de nuevo el proceso de descubrimiento de servicios. El mensaje RERR se envía cuando algún enlace roto ocasiona que algunos destinos lleguen a ser inalcanzables por algunos de sus nodos vecinos. El paquete RERR contiene los siguientes campos de cabecera: *Type* es el tipo de dato. *N* es una bandera de no eliminación, se coloca cuando un nodo ha ejecutado una reparación local de un enlace y los nodos siguientes no deben eliminar la ruta. *Reserved* se envía 0, es ignorado en recepción. *DestCount* es el número de destino inalcanzable incluido en el mensaje, debe tener como valor al menos un 1. *Unreachable Destination IP Address* es la dirección IP del destino que ha llegado a ser inalcanzable debido a un enlace roto. *Unreachable Destination Sequence Number* es el número de secuencia en la entrada de la tabla de rutas para el destino listado en el campo de la dirección IP del anterior destino inalcanzable.

3.4.2. Tabla de servicios

En LIFT cada dispositivo de la red posee una tabla de servicios (figura 3.9), dicha tabla posee la información necesaria para conocer los servicios que proveen los dispositivos de la red; de esta forma, si un nodo solicita algún servicio, entonces en primer lugar realiza una búsqueda en su tabla local.

Como hemos mencionado, LIFT añade el campo *Node Type* a los formatos de mensajes de AODV-SD y por tanto también añade el mismo campo a la tabla

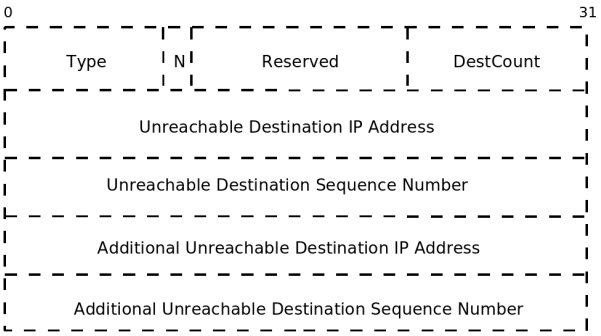


Figura 3.8: Formato del mensaje RERR

de servicios que maneja AODV-SD. Estas tablas de servicios mantienen exclusivamente entradas con información de las rutas hacia sus nodos vecinos, de tal manera que en cada tabla de servicios se conserva información de los dispositivos más próximos y no de toda la red. Las tablas de servicios son actualizadas periódicamente en función de la dinámica de la red.

32	8	4	32	16	32	8
Service ID	**Node Type**	**Protocol**	**IP Address**	**Lifetime**	**Attribute list**	**URL path**
Service ID$_0$	*Node Type*$_0$	*Protocol*$_0$	*IP Address*$_0$	*Lifetime*$_0$	*Attribute list*$_0$	*URL path*$_0$
Service ID$_1$	*Node Type*$_1$	*Protocol*$_1$	*IP Address*$_1$	*Lifetime*$_1$	*Attribute list*$_1$	*URL path*$_1$
Service ID$_2$	*Node Type*$_2$	*Protocol*$_2$	*IP Address*$_2$	*Lifetime*$_2$	*Attribute list*$_2$	*URL path*$_2$
Service ID$_3$	*Node Type*$_3$	*Protocol*$_3$	*IP Address*$_3$	*Lifetime*$_3$	*Attribute list*$_3$	*URL path*$_3$
Service ID$_4$	*Node Type*$_4$	*Protocol*$_4$	*IP Address*$_4$	*Lifetime*$_4$	*Attribute list*$_4$	*URL path*$_4$

Figura 3.9: Tabla de servicios

La tabla de servicios (figura 3.9) contiene una lista de los servicios que han sido anunciados por otros dispositivos proveedores de servicios. Cada entrada de esta tabla de servicios está compuesta por siete campos útiles para la localización de los servicios disponibles en la red: *Identificador del servicio (Service ID)* es una variable numérica para identificar un servicio, *Tipo de nodo (Node Type)* define el tipo de nodo, bien sea un HCD o un LCD, *Protocolo (Protocol)* el protocolo usado, *Dirección IP (IP Address)* una vez localizado el servicio,

se invoca directamente a través de esta dirección IP, que es la dirección que identifica al nodo que contiene el servicio. *Tiempo de expiración (*lifetime*)* es el tiempo estimado que el servicio permanecerá disponible en la red, *Lista de atributos (Attribute list)* describe el nombre del servicio en sí y varía de acuerdo al tipo de servicio que ofrece, y *Ruta URL (URL path)* es la ruta que se ha de seguir para alcanzar el nodo que posee el servicio solicitado.

La tabla de servicios emplea el campo *lifetime* que tiene como objetivo mantener un soft-state[3] del nodo, puesto que es muy importante mantener la información actualizada. El *lifetime* es el tiempo que un servicio permanece disponible en la red. Una vez finalizado el *lifetime* de un dispositivo, se elimina la fila de la tabla de servicios. El espacio en la tabla de servicios para almacenar nuevas filas es reducido y acotado a pocas entradas, por lo que se eliminan todas las filas que posean un tiempo próximo a expirar.

3.4.3. Funcionamiento de los distintos procedimientos de LIFT

En LIFT existe una tabla de servicios que se emplea como repositorio para almacenar la información de los servicios que se ofrecen en la MANET. Cada nodo participante en la red maneja su propia tabla de servicios, que contiene información de los servicios que ese nodo provee, así como la información referente a los servicios que proveen otros nodos. De esta manera, cuando un nodo requiere un servicio, ejecuta una búsqueda en su tabla de servicios local. La información de los servicios ofrecidos por un nodo se obtiene cuando este nodo se conecta a la red, mientras que la información de los servicios ofrecidos por otros nodos es adquirida cuando el nodo actual participa en el proceso de descubrimiento de servicios.

El funcionamiento de LIFT es como sigue:

[3]Es el estado actual de los dispositivos en una red, lo más fresco o reciente que sea posible. Los servicios deben ser periódicamente registrados de nuevo en la tabla de servicios para reiterar su disponibilidad, si dentro de cierto tiempo no se recibe confirmación del servicio, se eliminan las entradas en la tabla.

Nodo desempeñando el rol de peticionario de servicios

1. Un nodo envía un mensaje SREQ por dos razones. Bien porque requiere tener una ruta válida hacia un destino que no está registrado en su tabla de servicios o bien porque contiene una ruta no válida. Y simplemente comienza el proceso de descubrimiento de servicio.

    a) Revisa su tabla de servicios para verificar si él, contiene el servicio solicitado. En caso contrario, contacta al proveedor del servicio que tenga una ruta válida en su tabla de servicios, si el *lifetime* es aún válido y siempre que éste sea un HCD y se encuentre próximo.

    b) Si no cuenta con información del proveedor del servicio, entonces comienza el proceso de descubrimiento de servicios (inunda el cluster para contactar a otro nodo) mediante el envío de un mensaje SREQ. El mensaje SREQ lleva como valor del número de secuencia de destino el último número conocido de este destino. Y en el valor del número de secuencia de origen lleva el número de secuencia del nodo que se ha incrementado previamente.

2. El nodo que envía el mensaje SREQ indica un valor máximo para el TTL y fija un timeout determinado para esperar respuesta. Lo anterior es con la finalidad de evitar que los mensajes deambulen por tiempo indefinido por la red y ocasionen una congestión.

Nodo desempeñando el rol de proveedor de servicios

1. Cuando un nodo recibe el mensaje SREQ ejecuta lo siguiente.

    a) Determina una asociación válida (nombre de servicio, dirección IP y tipo de nodo) para saber si el servicio solicitado se encuentra en su tabla. Es decir, si provee el servicio o si conoce una ruta válida hacia un nodo que pueda proveer ese servicio.

b) Si encuentra esa información en su tabla de servicios, entonces el nodo emite por unicast un mensaje SREP al nodo solicitante.

2. Se genera un mensaje SREP cuando el nodo que envía el SREQ es el destino o cuando tiene una ruta válida hacia el destino y el número de secuencia de la entrada de la tabla de servicios es mayor que el valor de número de secuencia del mensaje SREQ. El nodo descarta el mensaje SREQ después de generar el mensaje SREP.

Nodo desempeñando el rol de nodo intermedio

Un nodo reenvía un mensaje cuando:

1. Contiene una asociación (nombre de servicio, dirección IP y tipo de nodo) a un servicio, pero no tiene una ruta válida a ese servicio, entonces asigna su dirección IP como la dirección que emite el SREQ. Cualquier nodo que reciba el mensaje SREQ con una dirección de destino válida, envía un mensaje SREP si tiene una ruta hacia el nodo destino o si conoce una ruta equivalente al servicio solicitado. En caso contrario, si no tiene información del servicio solicitado o una ruta hacia el nodo destino, únicamente reenvía el mensaje SREQ.

2. Recibe un mensaje SREQ, entonces crea o actualiza una ruta hacia el salto anterior. Posteriormente verifica que no haya recibido antes un mensaje con el mismo ID y número de secuencia de origen. En caso afirmativo, descarta el mensaje actual. En caso contrario: Incrementa en 1 el valor del contador de saltos. El nodo busca una ruta hacia la dirección IP origen del mensaje. En caso de no existir una ruta, entonces se crea una nueva ruta de vuelta. Posteriormente se realiza lo siguiente:

   a) Se compara el valor del número de secuencia de origen con el valor del número de secuencia del destino que se tiene en la tabla de servicios, y en caso de ser mayor entonces se sustituye el valor.

b) Se valida el campo de número de secuencia.

c) El nodo desde donde ha llegado el mensaje, se convierte en el siguiente salto en la tabla de servicios.

d) Se actualiza el valor del número de saltos en la tabla de servicios.

3. Recibe un mensaje SREP y ya tiene información acerca del servicio solicitado, compara el *lifetime* del mensaje SREP con el *lifetime* que tiene en su tabla de servicios, y si la información en su tabla es más reciente, entonces descarta el mensaje recibido y emite un SREP con esa información. De lo contrario, simplemente envía el mensaje a su destino.

4. No se trata del nodo destino el que ha generado el mensaje SREP, entonces escribe su propio número de secuencia de destino en el campo de número de secuencia del mensaje SREP. De esta forma, el nodo intermedio ha actualizado la ruta de retransmisión porque se ha puesto él como el último nodo en la lista de predecesores.

Hasta aquí terminamos con la descripción del funcionamiento analítico de LIFT. Básicamente LIFT consta de dos procesos: una gestión de grupos que clasifica los nodos de la red de acuerdo a sus capacidades y un mecanismo que usa estos clusters de red para descubrir servicios. Estos procesos se explican a continuación.

3.4.3.1. Gestión de grupos

En los protocolos de descubrimiento de servicios basados en clustering [3] [24] [57] [98] [97] [141] se construyen clusters de nodos estableciendo un líder denominado por algunos autores como *Clusterhead* (CH) o *Cluster Leader* (CL) y una cantidad arbitraria de nodos clusterizados (CN) o nodos ordinarios (ON). El líder del cluster actúa como un representante de los nodos clusterizados. Sin embargo, exceptuando a [97] (solución para redes de sensores inalámbricos y no de una MANET), estos enfoques tampoco distinguen tipos de nodos y dan

un trato igualitario al conjunto total de dispositivos, indistintamente de sus capacidades. En nuestra solución el nodo con capacidades superiores asume el rol de líder del cluster. Cabe resaltar que pueden coexistir más de un líder en cada cluster, esto con el fin de dar robustez al cluster ante eventuales caídas de algún líder.

Al igual que los formatos de mensajes a los que se les ha añadido el campo *Node Type* (descritos en el apartado 3.4.1), para la gestión de grupos es necesario utilizar ese campo en el formato de mensaje HELLO. El campo *Node Type* permite a cada nodo conocer las capacidades de sus vecinos. Esta información nos es útil para la creación de los clusters, en el que cada HCD desempeña un rol de CL (cluster leader) y a su vez guarda información acerca de todos los CN (clustered node) de su vecindario. Con este mecanismo es posible crear y mantener clusters, ya que cuando la topología de red cambie, los nodos pueden adaptar su información del cluster y su rol en la red, bien de CN o bien de CL. Es posible que existan varios líderes en el vecindario, todos ellos desempeñan el rol de CL. El propósito de esta política es evitar que se agoten los recursos de un mismo nodo.

Una vez añadido el campo *Node Type* al formato de mensajes HELLO del protocolo AODV-SD, es posible crear los clusters. El formato de mensaje HELLO (figura 3.10) es esencialmente un mensaje SREP, del que ya hemos descrito el contenido de su cabecera en el apartado 3.4.1. La única diferencia es que el formato de mensaje HELLO tiene en el campo TTL el valor de 1 salto, esto con el objetivo de que el mensaje sólo se difunda en la cercanía del vecindario. El nuevo formato de mensaje de LIFT lo denominaremos en adelante *CHELLO* (Clustering HELLO).

El mensaje CHELLO se envía periódicamente para obtener información relacionada con el vecindario y para mantener las rutas de red. La modificación del mensaje CHELLO obedece a que este mecanismo de descubrimiento de vecindario se usa para crear y mantener los clusters.

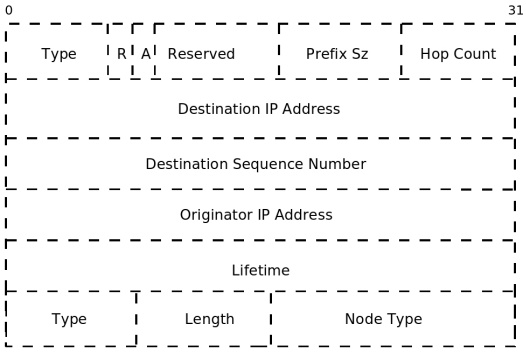


Figura 3.10: El formato del mensaje HELLO ahora tiene añadido el campo Node Type y ha sido renombrado como CHELLO

El mecanismo de mantenimiento de los clusters, consiste específicamente en el envío periódico de mensajes CHELLO anunciando la presencia de un nodo en el vecindario y contiene información para diferenciar el tipo de dispositivo. De esta manera, los nodos en el cluster se enteran de la identidad de los CL.

Con la extensión del campo *Node Type* en el mensaje CHELLO, la cantidad de mensajes enviados en la red no se incrementa y no modifica para nada su propósito original de crear y de actualizar las filas en las tablas de servicios que tienen como destino el nodo que ha originado el mensaje y de determinar la conectividad entre vecinos.

El mensaje CHELLO sirve para controlar el estado (conectado o desconectado), ya que si algún nodo vecino recibe un mensaje CHELLO del nodo líder y después de un tiempo de espera, no recibe un nuevo mensaje de ese mismo nodo emisor, entonces se puede suponer que se ha perdido el enlace debido a varias causas: que no está activo o que se ha desplazado lejos del alcance o que se le han agotado sus recursos, etc.

Un CL anuncia que su rol ha cambiado, por medio de un mensaje CHELLO, es decir, los nodos en la proximidad se enteran de qué nodos son HCD, mediante el envío periódico de mensajes CHELLO (el mensaje CHELLO lleva un campo Node Type con información del tipo de nodo). De esta forma, si los CN no perciben la presencia de un CL, suponen que no se encuentra disponible en ese

momento, y dado que en un cluster existe más de un HCD, entonces al no estar disponible un CL, el resto de HCD asumen en ese momento el papel de CL y continúan atendiendo las peticiones de servicios de los CN. Para esto, suponemos que todos los HCD están dispuestos a cooperar con la red. De esta forma, un HCD no rehúsa asumir el papel de CL.

3.4.3.2. Ejemplo de funcionamiento

Para comenzar el proceso de descubrimiento de un servicio, se debe crear un mensaje SREQ llevando información del propio CN origen y del CL destino. El mensaje SREQ posee un identificador propio (campo RREQ ID). El identificador se incrementa cada vez que se genera un nuevo SREQ, el identificador es de utilidad para los nodos intermedios, ya que ellos verán si reenvían el mensaje o lo descartan si es que ya lo han retransmitido anteriormente. El mensaje SREQ se envía por broadcast (figura 3.11) al vecindario dentro del cluster.

Aunque los nodos intermedios no sean los destinatarios del mensaje SREQ, mantienen una entrada para ese destino en su tabla de servicios, contestan al CN origen para evitar la propagación innecesaria del SREQ por toda la red. Aun cuando tengan alguna entrada activa, es necesario que se cumpla que esa ruta sea más actual que la última ruta recibida por el CN que ha enviado el mensaje SREQ. El mensaje SREQ lleva la dirección IP del CN origen, la dirección IP del CL destino, el campo Node Type que tiene como propósito distinguir si se trata de un nodo HCD o LCD. Los nodos intermedios registran este dato en su tabla de servicios y de esta forma saben que el mensaje va dirigido hacia un nodo destino que tiene como característica ser CL y por tanto proveedor de algunos servicios. Es posible que el nodo destino sea un CN, sólo en caso que el CL no sea el proveedor del servicio.

En el momento que un nodo intermedio reenvía un SREQ (figuras 3.12, 3.13), añade una ruta inversa en su tabla, esta ruta apunta hacia el CN origen emisor del SREQ. Cuando el mensaje SREQ alcanza el CL destino, el desti-

natario devuelve un mensaje SREP al CN origen (figura 3.14), a través de la ruta inversa por donde le ha llegado el mensaje SREQ.

Una vez establecida una ruta entre ambos nodos (CN origen - CL destino) (figura 3.16), la ruta es válida sólo durante un periodo de tiempo; ese tiempo se encuentra registrado en el campo *lifetime*, se maneja un tiempo de expiración debido a la movilidad de la red, ya que más adelante esa ruta puede no ser válida.

El mantenimiento de las rutas sirve precisamente para resolver la situación antes descrita. Si el CN origen que ha enviado el mensaje se mueve y como consecuencia modifica la topología, entonces debe reiniciar un nuevo proceso de descubrimiento de ruta. Por otro lado, si el nodo que se mueve es un nodo intermedio o se trata del CL destino (figura 3.17 en este momento el CL destino se ha movido ligeramente) y existe algún mensaje dirigido a él, entonces el nodo que detecta el fallo genera un mensaje de error (denominado RERR y tiene como propósito invalidar la ruta) y lo envía al nodo que ha originado el envío. (figura 3.18).

Todos los nodos intermedios por los que pasa el mensaje RERR (figura 3.18), cancelan las rutas hacia el nodo que se ha vuelto inalcanzable. Una vez que el mensaje RERR llega a su destino (figura 3.19 en este caso el CN origen), el CN origen decide dar por terminado el envío o enviar de nuevo un mensaje SREQ para establecer de nueva cuenta los enlaces hacia el nodo que provee el servicio.

Cada determinado tiempo es necesario mantener actualizada la información referente a los nodos del vecindario. Siempre que un nodo recibe un mensaje de algún vecino, actualiza la entrada en la tabla de servicios del vecino que ha enviado el mensaje. Cada lapso determinado se envían mensajes CHELLO a los vecinos, con objeto de avisar que el nodo que ha enviado el mensaje aún se encuentra activo. Con la llegada de este mensaje, los nodos vecinos actualizan los *lifetime* asociados a ese nodo o en caso contrario, eliminan la entrada de la tabla de servicios de los nodos que no respondan.

Con el mensaje CHELLO los CN vecinos saben que la ruta hacia el CL se encuentra aún válida y en caso de no recibir el mensaje CHELLO, se hace suponer que la ruta hacia CL se ha roto. Después de este evento, cualquier CN podría comenzar de nuevo el proceso de descubrimiento de servicios; en caso de que el último CL no se encuentre disponible ya en el cluster, entonces otro HCD asumirá el rol de CL.

Ahora bien, para ilustrar un ejemplo más explicitamente, a continuación se resume gráficamente la secuencia de pasos que LIFT sigue para descubrir un servicio.

El ejemplo muestra 10 dispositivos móviles heterogéneos dentro de un fragmento de la red, para fines de mostrar el ejemplo hemos omitido desplegar la red en su conjunto, por tanto, lo que únicamente mostramos en las figuras es un grupo de nodos o cluster.

En la figura 3.11 se muestra que el CN origen con la dirección IP 192.168.1.4 inicia el proceso de descubrimiento de servicio a través del envío del mensaje SREQ. El mensaje SREQ difundido es escuchado por los nodos intermedios con dirección IP: 192.168.1.3, 192.168.1.9 y 192.168.1.1.

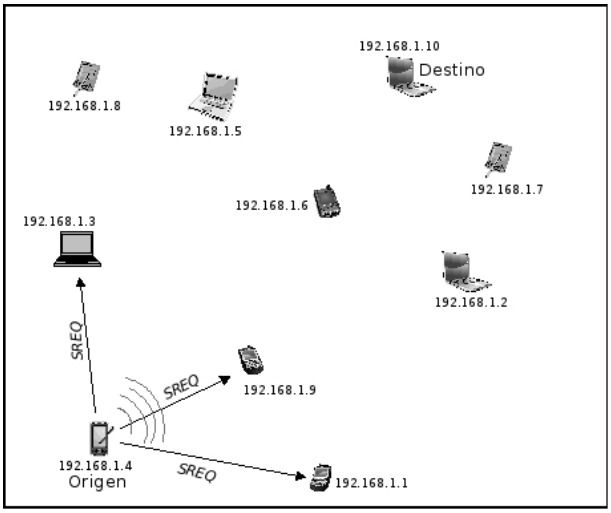


Figura 3.11: CN origen 192.168.1.4 envía el mensaje SREQ para descubrir un servicio

La figura 3.12 muestra como el nodo intermedio con dirección IP 192.168.1.9 analiza el mensaje, al enterarse que no es él el destinatario, actualiza los campos correspondientes del mensaje y lo reenvía al nodo intermedio cuya dirección IP es 192.168.1.6.

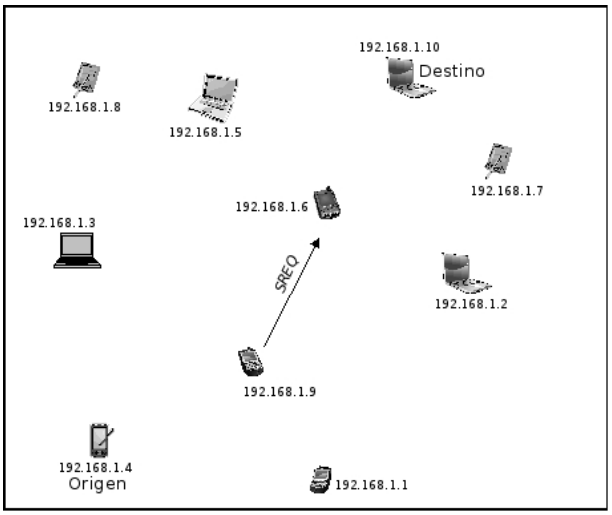


Figura 3.12: El nodo intermedio 192.168.1.9 recibe el mensaje SREQ, lo analiza y lo reenvía al nodo intermedio 192.168.1.6.

El nodo intermedio cuya dirección IP es 192.168.1.6 recibe el mensaje, lo analiza, lo procesa y lo reenvía al destinatario que lleva el mensaje. El mensaje SREQ finalmente alcanza al CN destino (dirección IP: 192.168.1.10), en la figura 3.13 se ilustra este paso.

Una vez que el mensaje SREQ ha llegado al CL 192.168.1.10, éste devuelve por unicast un mensaje SREP a través de la ruta inversa por la que ha llegado el mensaje SREQ (figura 3.14). El mensaje SREP comienza el camino de vuelta hacia el CN emisor de la petición de servicios. EL mensaje SREP lo genera el CL 192.168.1.10 y lo envía al último nodo intermedio de la ruta. El mensaje lo recibe el nodo intermedio 192.168.1.6.

En la figura 3.15 se muestra el paso en el que el nodo intermedio 192.168.1.6 recibe el mensaje SREP generado por el CL destino 192.168.1.10. Este nodo

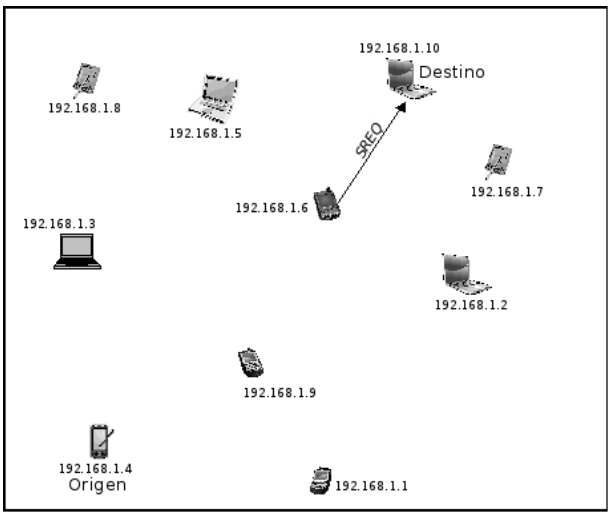


Figura 3.13: El nodo intermedio 192.168.1.6 recibe el mensaje SREQ, lo analiza y lo reenvía al CL destino 192.168.1.10.

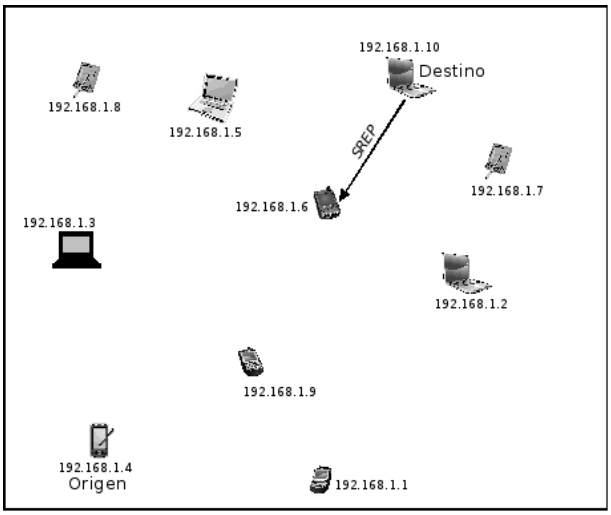


Figura 3.14: El CL destino 192.168.1.10 genera el mensaje SREP y lo envía al CN origen 192.168.1.4, atravesando por algunos nodos intermedios.

intermedio, analiza el mensaje y lo reenvía al CN origen, no sin antes atravesar por el nodo intermedio 192.168.1.9.

En la figura 3.16 se observa que el mensaje SREP proveniente del nodo intermedio 192.168.1.9 llega al CN origen cuya dirección IP es 192.168.1.4. A

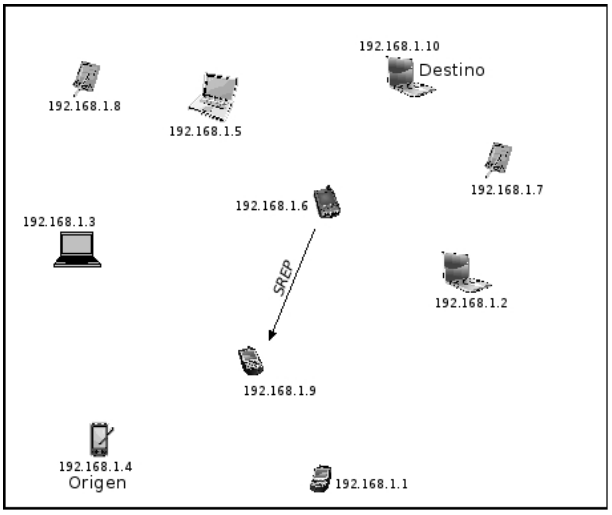


Figura 3.15: El nodo intermedio 192.168.1.6 recibe el mensaje SREP y lo reenvía al nodo intermedio 192.168.1.9.

partir de este momento se establece comunicación y por tanto, una ruta válida para llegar del CN origen 192.168.1.4 al CL destino 192.168.1.10

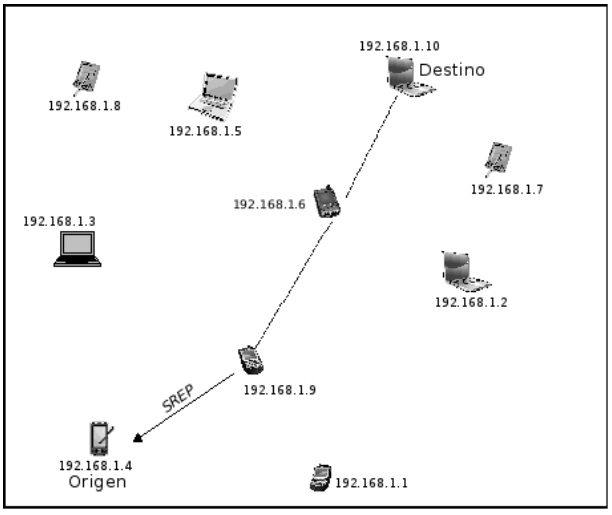


Figura 3.16: El nodo intermedio 192.168.1.9 recibe el mensaje SREP y lo reenvía al CN destino 192.168.1.4. Se estable una ruta válida entre CN origen 192.168.1.4 y el CL destino 192.168.1.10.

Debido a la movilidad natural de la redes móviles Ad Hoc, algunos nodos se han desplazado de su localización anterior, ocasionándose una rotura de enlace entre el CL destino 192.168.1.10 y el nodo intermedio 192.168.1.6., la ilustración del fallo se muestra en la figura 3.17.

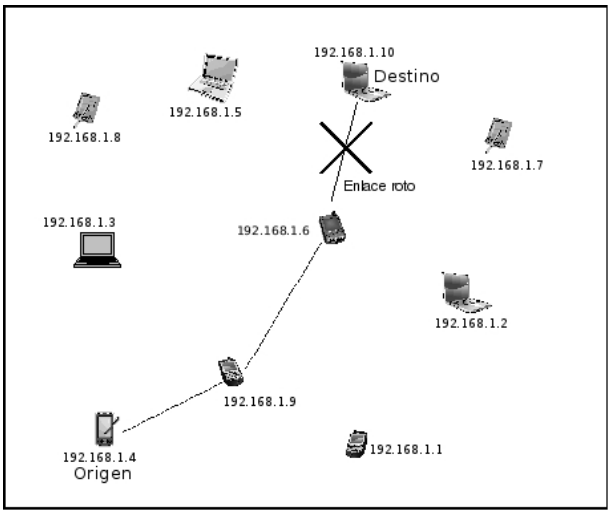


Figura 3.17: El nodo intermedio 192.168.1.6 detecta el enlace roto entre él y el CL destino 192.168.1.10.

Por ser el nodo intermedio 192.168.1.6 el que advierte el fallo, entonces será él mismo quien genere el mensaje RERR (error de ruta) y lo envíe al nodo intermedio 192.168.1.9, dirigiéndose hacia el CN origen (figura 3.18).

En la figura 3.19 se muestra el momento cuando el mensaje RERR alcanza al CN origen 192.168.1.4 proveniente del nodo intermedio 192.168.1.9. De esta manera, el CN origen es informado de que ha ocurrido un fallo en el enlace entre el nodo intermedio 192.168.1.6 y el CL destino 192.168.1.10. El CN origen decide si comienza nuevamente el proceso de descubrimiento de servicios, puesto que la ruta válida entre el CN origen 192.168.1.4 y el CL destino 192.168.1.10 ha sido cancelada por completo, cada nodo intermedio en el momento de reenviar el mensaje RERR cancelaba la ruta hacia el CL destino que se había vuelto inaccesible.

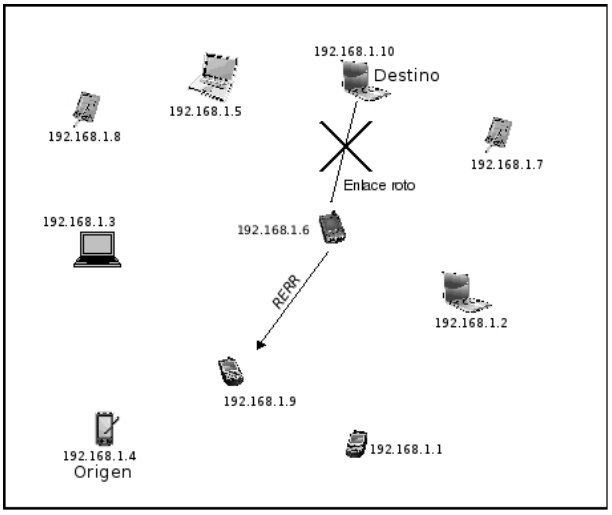


Figura 3.18: Nodo intermedio 192.168.1.9 origina y envía el mensaje RERR hacia el nodo intermedio 192.168.1.9.

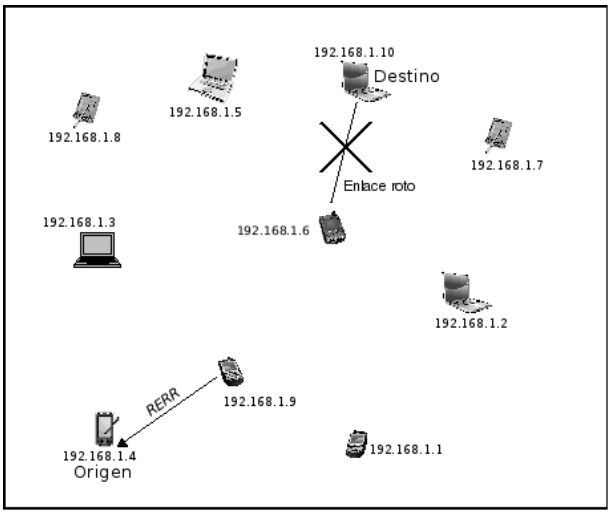


Figura 3.19: CN origen 192.168.1.4 recibe el mensaje RERR con el aviso de que ha ocurrido una rotura de enlace entre el nodo intermedio 192.168.1.6 y el CL destino 192.168.1.10.

Al perderse el enlace activo entre el CN origen y el CL destino, el CN origen puede iniciar nuevamente el proceso de descubrimiento de servicios; este paso se ha mostrado ya en la figura 3.11, en dicha figura se ve como el CN origen lanza una petición de servicios (mediante el envío del mensaje SREQ) a sus nodos

vecinos más próximos. A partir de este paso se repite todo el proceso descrito anteriormente. Es de esta manera como LIFT descubre servicios

3.4.4. Características de LIFT

Para cerrar el capítulo describiremos las características de LIFT respecto a las diferentes dimensiones de clasificación del capítulo 2 Estado del arte y poder así clasificarlo de acuerdo a la taxonomía mostrada en la figura 2.1.

LIFT es un protocolo de descubrimiento de servicios que pertenece a los esquemas que caen en la categoría de *arquitectura de directorio distribuida con infraestructura*. Esto es, arquitecturas que recurren al uso directorios locales, en este caso LIFT emplea tablas de servicios distribuidas para registrar de forma local, la información de los servicios disponibles en los nodos del vecindario.

Una peculiaridad distintiva de LIFT es que combina un *diseño Cross-layer* con un *esquema de clustering*. Además, tiene presente la suposición de que una MANET está compuesta por dispositivos heterogéneos y que dentro de esta variedad de tipos de dispositivos, se pueden distinguir a los dispositivos que tienen capacidades superiores, con el objetivo de llevar a cabo una selección y así poder ser elegidos líderes de los clusters.

Se puede decir que LIFT es una solución que utiliza diseño Cross-layer, porque descubre servicios en la capa de red y no en la capa de aplicación como muchas otras soluciones. LIFT emplea el protocolo de encaminamiento reactivo AODV como protocolo de red subyacente. Para ello, durante el proceso de encaminamiento inserta un campo dentro de la cabecera de algunos mensajes de AODV, este campo contiene información del tipo de nodo. El campo tiene como fin distinguir el tipo de dispositivo que se trata, bien sea un dispositivo con capacidades superiores (HCD) o bien sea un dispositivo con capacidades limitadas (LCD); de esta forma, LIFT aprovecha llevar información de descubrimiento de servicios durante el proceso de descubrimiento de rutas.

Y respecto a que emplea técnicas de clustering, porque agrupa los nodos de un área determinada y les establece líderes de cluster para el soporte del proceso

de descubrimiento de servicios. El criterio para elegir líder consiste esencialmente en diferenciar y seleccionar los dispositivos que poseen capacidades superiores de aquellos dispositivos limitados en recursos, una vez seleccionados los dispositivos con capacidades superiores, estos podrán ser elegidos líderes de clusters.

Capítulo 4

Metodología para validación de la propuesta basada en simulación

Una vez realizada la propuesta de solución se debe validar de alguna forma. Existen dos formas de validar la propuesta. La primera es modelar matemáticamente el protocolo, mediante teoría de probabilidades u otras herramientas de modelado formal, y mediante teoremas matemáticos, para validar la propuesta se deben mostrar orientaciones de cotas inferiores y superiores de los parámetros a medir. La segunda forma es mediante simulación, reproduciendo en computadora los procedimientos del protocolo y ajustando valores a los parámetros. Ambas formas no son excluyentes. Es posible también utilizar ambos enfoques. En nuestro caso, validaremos la propuesta mediante simulación.

Antes que nada, presentamos el entorno de simulación para el protocolo de descubrimiento de servicios propuesto. Hemos elegido el simulador de redes Ns-2[1] como herramienta de simulación; primero, porque cumple con nuestros requerimientos funcionales; segundo, porque es un software de distribución gratuita y de código abierto; y tercero, porque además es la herramienta más extendida y en la que la mayoría de los investigadores generan sus resultados. Por consiguiente,

[1]Ns-2 (Network Simulator) es un simulador de redes basado en eventos discretos, ampliamente usado por la comunidad de investigadores en redes.

la mayor parte del código se puede reutilizar y de esta forma realizar comparaciones de rendimiento contra otros proyectos que utilizan la misma herramienta de simulación.

Posteriormente, se presenta alguna información acerca de la elección y diseño del escenario de simulación, seguido de la descripción de la selección de todos los parámetros de simulación utilizados.

Antes de entrar en los detalles de las características de la simulación para probar el rendimiento de LIFT, describiremos la herramienta de simulación utilizada. No obstante que existen otros simuladores de redes (OPNET [116], OMNET++[115], QualNet [128], GloMoSim [60], Parsec [120] y GTNetS [64]) utilizados para simular algunos de los protocolos de descubrimiento de servicios expuestos en el apartado 2.2, nosotros nos hemos decantado por utilizar Ns-2 [110]. Aunque el resto de herramientas de simulación disponibles son ampliamente usadas dentro de la comunidad de investigadores de redes. Cabe mencionar que cada una de estas herramientas tiene sus fortalezas y debilidades, y ningún único entorno de simulación es adecuado para todos los posibles requerimientos de simulación [134].

4.1. Herramienta de simulación

Como hemos mencionado al principio de este capítulo, la herramienta que utilizamos para simular nuestra propuesta es el simulador de redes Ns-2, enseguida abordaremos los aspectos generales del simulador.

La razón principal de usar este simulador es que los protocolos básicos sobre los que se sustenta LIFT se encuentran ya programados en Ns-2. Por tanto, es adecuado reutilizar estos procedimientos, dado que se ha contrastado suficientemente dicho código y se asume como válido, plantear una simulación en ese entorno. Para comparar, sobre todo, con AODV-SD.

Ns-2 es un simulador ampliamente aceptado y utilizado en la comunidad académica para el análisis de redes, es una herramienta desarrollada por el

proyecto VINT(Virtual InterNetwork Testbed)[2] en University of California at Berkeley.

El código fuente de Ns-2 está disponible bajo diversas licencias, lo que permite que en el simulador haya aportaciones de multitud de centros de investigación de todo el mundo. Ns-2 se ha construido por diferentes personas por lo que la confiabilidad de los módulos puede variar.

Inicialmente Ns-2 fue ideado para redes fijas, sin embargo, el grupo Monarch de CMU desarrolló una ampliación para el análisis de redes inalámbricas que incluye las principales propuestas de redes Ad Hoc así como de redes WLAN (Wireless Local Area Networks).

Ns-2 es un simulador de redes de eventos discretos que simula diversos entornos, gran variedad de protocolos de transporte de red (transporte, multicast, routing), topologías simples o complejas, agentes, distintas fuentes de generación de tráfico, simulación de aplicaciones, diversas políticas de gestión de colas, modelado de errores, redes de área local, redes inalámbricas, etc. Ns-2 tiene implementados protocolos como TCP y UPD. Genera comportamientos de tráfico como FTP, Telnet, Web y CBR. Tiene la capacidad de simular mecanismos de gestión de colas en routers como: Drop Tail, RED y CBQ, y soporta diversos algoritmos de encaminamiento como DSR, AODV, DSDV, TORA, etc.

Ns-2 está totalmente escrito en C++ y usa dos lenguajes de programación, C++ y OTcl. La razón principal de usar dos lenguajes se debe a que Ns-2 lleva a cabo dos diferentes tareas [52]. En la primer tarea implementa todos los detalles de los protocolos de comunicaciones, para ello requiere un lenguaje que manipule bits, bytes, defina cabeceras de paquetes e implemente algoritmos de manera eficiente y con gran facilidad, en esta tarea es importante la velocidad de procesamiento, para esta primer tarea Ns-2 utiliza el lenguaje C++. Mientras que la segunda tarea consiste en desarrollar propiamente la simulación, para ello se requiere un lenguaje que permita crear la configuración y la variación de los

[2]El proyecto VINT es la unión de esfuerzos de gentes de UC Berkeley, USC/ISI, LBL y Xerox PARC. El proyecto es soportado por la Defense Advanced Research Projects Agency (DARPA).

parámetros para poder explorar los resultados de la simulación; este lenguaje debe ser capaz de realizar fácilmente cambios en los parámetros y no requerir recompilar nuevamente la simulación, para esta segunda tarea Ns-2 utiliza el lenguaje OTcl (Object Tool command language, lenguaje orientado a objetos de tipo intérprete) que funciona como interfaz de usuario.

Ns-2 incluye igualmente la capacidad de crear aplicaciones y protocolos personalizados. El Ns-2 dispone de varios protocolos implementados. Sin embargo, a veces resulta conveniente ampliarlos o implementar unos nuevos a partir de ciertas especificaciones. En este caso, la utilización de Ns-2 para este tipo de tareas requiere la utilización de C++.

Una característica de Ns-2 es la capacidad de modificar parámetros en distintas capas. Otra característica de Ns-2 es la capacidad de soportar fuentes de tráfico como: web, FTP y CBR. El patrón de tráfico CBR (tasa de bits constante) se genera recurriendo a la importación de tráfico de un archivo externo, en cambio, para realizar los experimentos de FTP, el simulador emplea la fuente de tráfico propietaria de Ns-2 [93].

Ns-2 produce resultados de los que se pueden obtener datos para todo tipo de mediciones sobre la simulación o bien, trazas específicas para visualizarlas en la herramienta *nam* que reproduce una animación de la simulación.

Se puede observar en la figura 4.1 que el núcleo de Ns-2 está basado principalmente en el lenguaje C++, en tanto que el escenario se especifica en el lenguaje OTcl. Este escenario se nutre de la herramienta de generación de tráfico *cbrgen* y de la herramienta *setdest*, útil para simular el comportamiento de movilidad de los dispositivos, de acuerdo con los modelos de movilidad más utilizados para modelar el comportamiento de los nodos móviles. Finalmente, al concluir la simulación se produce un resultado que se registra en trazas, que pueden posteriormente ser procesadas con el lenguaje *perl* o *awk* y visualizadas con la herramienta gráfica *nam* o *gnuplot*.

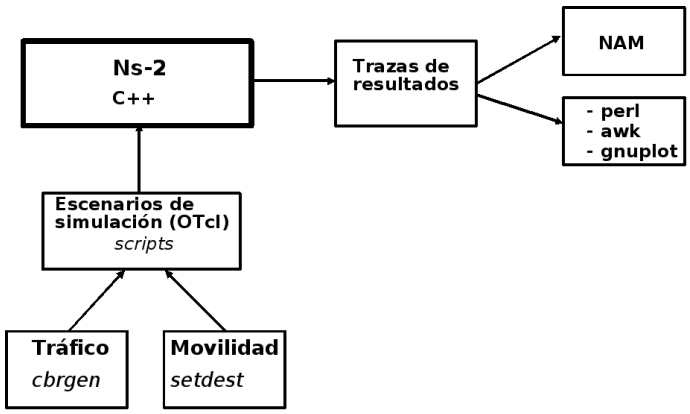


Figura 4.1: Esquema de módulos de Ns-2.

La figura 4.2 ofrece una vision simplificada de Ns-2 desde el punto de vista del usuario, que percibe el simulador como un intérprete de OTcl y que cuenta con:

- Un planificador de eventos.
- Librerías de objetos con componentes de red: nodos, aplicaciones y protocolos, todos ellos productores y consumidores de paquetes.
- Librerías de configuración de red que permiten crear enlaces entre los componentes de red (denominado fontanería).

Como se puede observar en la figura 4.2, se comienza con un script escrito en Tcl orientado a objetos, es decir, OTcl y este script es exactamente lo que el usuario codifica para simular y es una de las entradas que el usuario proporciona al programa. El script tiene diversos componentes internos que se muestran en el cuadro central de la misma figura; entre ellos, iniciar el planificador de eventos, configurar la topología de la red usando las librerías de componentes de red y configuración de red, e indicar a las fuentes de tráfico cuándo comenzar y detener la transmisión de paquetes a través del planificador de eventos. Se puede crear un objeto de red a partir de cero o como la composición e interconexión de otros objetos, en el que un objeto apuntaría a la dirección del objeto que sea

su vecino, estableciendo la trayectoria seguida por los datos en la red. En estos componentes se configura la topología de la red, se calendariza los eventos, se definen las funciones necesarias para la simulación, entre otras cosas.

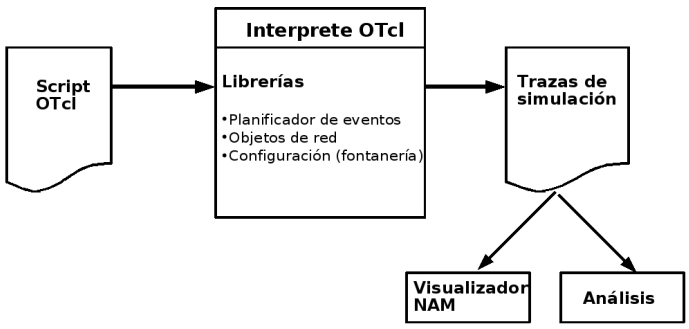


Figura 4.2: Simulador de red Ns-2.

4.1.1. Análisis de los archivos de trazado

Después de explicar el funcionamiento de la herramienta de simulación Ns-2, daremos una breve explicación de su uso básico.

El uso del simulador Ns-2 es bastante simple, primero se crea un archivo de configuración que describe los parámetros de la simulación que deseamos (nodos, topología, movimientos, tráfico, energía, etc.) este código de configuración es un script escrito en OTcl, posteriormente este código se pasa como parámetro a Ns-2, el simulador genera una traza de todos los eventos que se han generado durante la simulación. La salida de la simulación son dos archivos de texto (uno con extensión *.tr* y otro con extensión *.nam*), el archivo *.tr* puede visualizarse en el visor gráfico del programa *xgraph*.

Por ejemplo, cuando se ejecuta el comando: *ns nombrearchivo.tcl* se generan como habíamos dicho, dos archivos, uno *.nam* y otro *.tr*, aunque contienen la misma información, tienen distintos formatos. Al observar estos archivos se distinguen todos los eventos realizados durante la simulación línea por línea. El archivo *.nam* se puede visualizar con la aplicación de ns-2 conocida como **nam**, programa que tiene como función interpretar estos valores y simularlos en

una interfaz gráfica bastante amigable. Mientras que el archivo *.tr* es necesario procesarlo con un lenguaje de scripts con reconocimiento de patrones y manejo de expresiones regulares, a fin de que pueda ser de utilidad en el momento de filtrar ciertos eventos. Eso se puede lograr desde la línea de comandos en Linux, específicamente con lenguajes como *awk* o *Perl*.

4.1.1.1. Formato de las trazas generadas por el simulador

Existe un formato básico de salida del archivo *.tr*, sin embargo, puede haber información adicional que depende del protocolo de encaminamiento utilizado, es decir, se tiene un formato específico para AODV, DSDV, DSR, TORA, etc. Cada evento puede comenzar con uno de los cuatro caracteres (s, r, f, d) seguido por un flag, en el que el primer y segundo carácter del flag determinan su tipo:

- **N** Propiedad del nodo.
- **I** Información a nivel de paquete IP.
- **H** Información del próximo HOP.
- **M** Información del paquete a nivel de MAC.
- **P** Información específica del paquete.

Nos interesa la salida de trazas del protocolo AODV, entonces estudiaremos exclusivamente las salidas relacionadas a ese protocolo. Una salida típica de un archivo *.tr* puede ser como sigue, aunque esto es sólo una línea, el archivo completo puede contener miles de estas líneas, una para cada evento ocurrido durante la simulación.

El formato del archivo *.tr* que produce la simulación posee la siguiente estructura:

```
s -t 7.000000000 -Hs 49 -Hd 2 -Ni 49 -Nx 627.42 -Ny 137.61 -Nz 0.00
-Ne 10.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 49.255
-Id -1.255 -It AODV -Il 116 -If 0 -Ii 0 -Iv 30 -P aodv -Pt 0x2 -Ph 1
-Pb 1 -Pd -1 -Pds 0 -Ps 49 -Pss 4 -Pc REQUEST
```

donde:

s = send, r = received, f = forward, d = drop

-t = time
-Hs = source node (for this node)
-Hd = destination node (id for next hop towards the destination.)
-Ni = source node
-Nx, -Ny, -Nz = x, y, z coordinates
-Ne = energy level
-Nl = trace level, such as AGT, RTR, MAC
-Nw = node event (reason for the event)
-Ma = MAC level information (duration)
-Md = destination Ethernet address
-Ms = source Ethernet address
-Mt = Ethernet type
-Ii = IP packet level information
-Is = source address.source port number
-Id = destination address.destination port number
-It = packet type
-Il = packet size
-If = flow id
-Iv = ttl value
-P = Packet info at Application level (ARP, TCP)
-Pd = destination address
-Ps = source address
-Pl = reply length
-Pc = report to whom

4.1.1.2. Ejemplo de script de OTcl

A continuación presentamos un pequeño y sencillo ejemplo de script en OTcl. No se pretende que la sección sea un tutorial de programación de Ns-2 sino el ofrecer una idea de la potencia del lenguaje de especificación de scripts.

En el código inicialmente se definen 4 nodos móviles extendidos sobre un área plana de 500 x 500 metros y el tiempo total de simulación es de 5 segundos. A grandes rasgos, las acciones que realiza el script consisten en que 4 nodos aparecen en el escenario y comienzan a enviarse paquetes de datos entre ellos. Enseguida describimos brevemente dichas acciones.

- A los 0.05 segundos, los nodos 0 y 1 aparecen en el escenario en determinados puntos del área de simulación, se establece comunicación entre ellos y en todo momento los nodos se mueven libremente. En el tiempo 0.3 segs. el nodo 0 comienza a mandar paquetes FTP al nodo 1.
- En el tiempo 0.6 segundos, simultáneamente aparecen en el escenario los nodos 2 y 3, al igual que los nodos 0 y 1, estos se desplazan libremente en todo momento; el nodo 0 establece conexión con el nodo 2 y en el tiempo 0.6 segs., el nodo 0 comienza a mandar paquetes FTP al nodo 2.
- En el tiempo 0.9 segs., el nodo 0 establece conexión con el nodo 3 y en ese mismo instante el nodo 0 comienza a enviar paquetes FTP al nodo 3.
- En el tiempo 1.2 segs., el nodo 1 establece conexión con el nodo 3 y al mismo tiempo el nodo 1 comienza a enviar paquetes FTP al nodo 3.

Y así prosigue la simulación hasta transcurrir los 5 segundos definidos como período de observación.

El script completo en OTcl que ejecuta las acciones descritas aparece a continuación. Para ir explicando parte por parte el código, hemos añadido líneas con comentarios que tienen como propósito explicar brevemente la acción que realiza cada sentencia. Las líneas que comiencen con el carácter # al igual que las sentencias seguidas de ;# se tratan de un comentario explicativo por parte del programador.

```
# ========================================================
#  Nombre de archivo : wireless4nodes.tcl
# ========================================================

# Se definen todas las opciones de configuración

set val(chan)   Channel/WirelessChannel  ;# tipo de canal
set val(prop)   Propagation/TwoRayGround ;# modelo de propagation de radio
set val(netif)  Phy/WirelessPhy          ;# tipo de interface de red
set val(mac)    Mac/802_11               ;# tipo de MAC
set val(ifq)    Queue/DropTail/PriQueue  ;# tipo de interfaz de cola
set val(ll)     LL                       ;# tipo de capa de enlace
set val(ant)    Antenna/OmniAntenna      ;# modelo de antena
set val(ifqlen) 50                       ;# paquete máximo en ifq
set val(nn)     4                        ;# número de nodos móviles
set val(rp)     AODV                     ;# protocolo de encaminamiento
set val(x)      500                      ;# dimension de la topografía en x
set val(y)      500                      ;# dimension de la topografía en y


# ========================================================
# Programa principal
# ========================================================

# Inicializa las variables globales, crea las instancias del simulador.
set ns_ [new Simulator]

# Crea los archivos de trazas para ns (.tr) y nam (.nam)
# Se abre un archivo para escritura (w) wireless_4_nodes.tr
set tracefd [open wireless_4_nodes.tr w]   ;# con el fin de enviar el trazado
                                            # de la simulación al archivo .tr

# Se abre un archivo para escritura (w) wireless_4_nodes.nam
set namtrace [open wireless_4_nodes.nam w] ;# con el fin de enviar el trazado
                                            # de la simulación al archivo .nam

$ns_ trace-all $tracefd  ;# el trazado se envía al archivo
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# ========================================================

set topo       [new Topography] ;# Colocación del objeto topográfico

$topo load_flatgrid $val(x) $val(y)  ;# Se define el tamaño del escenario.

# En este punto se configuran las variables que usarán los nodos.
# Estas variables han sido declaradas al principio de este archivo.
```

```
create-god $val(nn)
   $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

# =============================================================


# Se crean los 4 nodos que intervienen en la simulación.

for {set i 0} {$i < $val(nn) } {incr i} {
set node_($i) [$ns_ node]
$node_($i) random-motion 0;
}

# =============================================================

# Se colocan los nodos 0 y 1 en su posición
# inicial dentro del escenario.
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 50.0
$node_(1) set Y_ 50.0
$node_(1) set Z_ 0.0

# =============================================================

# Se indica a los nodos 0 y 1 se muevan hacia
# a esas coordenadas en el tiempo .05 segs.
$ns_ at .05 "$node_(0) setdest 50.1 50.0 150.0"
$ns_ at .05 "$node_(1) setdest 5.1 5.0 150.0"


# Creación de agentes, aplicación FTP sobre TCP
set tcp [new Agent/TCP]           ;# Configuración del agente TCP.
```

```
$tcp set class_ 1
$ns_ attach-agent $node_(0) $tcp  ;# agregar el agente al nodo 0
set sink [new Agent/TCPSink]      ;# Configuración del agente SINK
$ns_ attach-agent $node_(1) $sink ;# agregar el agente al nodo 1
$ns_ connect $tcp $sink           ;# conexión para los agentes tcp y sink
# Configurar que agente TCP sea una aplicación FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 0.05 "$ftp start"         ;# se indica que comience al 0.05 segs.

# Se indica a los nodos 0 y 1 se muevan
hacia esta nueva posición.
$ns_ at 1.0 "$node_(0) setdest 5.0 5.0 150.0"
$ns_ at 1.0 "$node_(1) setdest 50.0 50.0 150.0"

# Se colocan los nodos 2 y 3 en su posición
# inicial dentro del escenario.
$node_(2) set X_ 5.0
$node_(2) set Y_ 50.0
$node_(2) set Z_ 0.0
$node_(3) set X_ 50.0
$node_(3) set Y_ 5.0
$node_(3) set Z_ 0.0

# Se indica a los nodos 2 y 3 se muevan hacia
# a esas coordenadas en el tiempo .5 segs.
$ns_ at .5 "$node_(2) setdest 50.1 5.0 150.0"
$ns_ at .5 "$node_(3) setdest 5.1 50.0 150.0"

# ============================================================

# Creación de agentes, aplicación FTP sobre TCP
set tcp [new Agent/TCP]           ;# Configuración del agente TCP.
$tcp set class_ 2
$ns_ attach-agent $node_(0) $tcp  ;# agregar el agente al nodo 0
set sink [new Agent/TCPSink]      ;# Configuración del agente SINK.
$ns_ attach-agent $node_(2) $sink ;# agregar el agente al nodo 2
$ns_ connect $tcp $sink           ;# conexión para los agentes tcp y sink
# Configurar que agente TCP sea una aplicación FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 0.6 "$ftp start"          ;# se indica que comience al 0.6 segs.

# Se indica al nodo 2 se mueva
hacia esta nueva posición.
$ns_ at .8 "$node_(2) setdest 25.0 5.0 150.0"
$ns_ at 1.8 "$node_(2) setdest 50.0 5.0 150.0"
```

```
# Creación de agentes, aplicación FTP sobre TCP
set tcp [new Agent/TCP]           ;# Configuración del agente TCP.
$tcp set class_ 3
$ns_ attach-agent $node_(0) $tcp  ;# agregar el agente al nodo 0
set sink [new Agent/TCPSink]      ;# Configuración del agente SINK.
$ns_ attach-agent $node_(3) $sink ;# agregar el agente al nodo 3
$ns_ connect $tcp $sink           ;# conexión para los agentes tcp y sink
# Configurar que agente TCP sea una aplicación FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 0.9 "$ftp start"           ;# se indica que comience al 0.9 segs.

# Creación de agentes, aplicación FTP sobre TCP
set tcp [new Agent/TCP]           ;# Configuración del agente TCP.
$tcp set class_ 4
$ns_ attach-agent $node_(1) $tcp  ;# agregar el agente al nodo 1
set sink [new Agent/TCPSink]      ;# Configuración del agente SINK.
$ns_ attach-agent $node_(3) $sink ;# agregar el agente al nodo 3
$ns_ connect $tcp $sink           ;# conexión para los agentes tcp y sink
# Configurar que agente TCP sea una aplicación FTP
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 1.2 "$ftp start"          ;# se indica que comience al 1.2 segs.

# ============================================================

# Se le indica a los nodos cuando termina la simulación.
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 5.0 "stop"
$ns_ at 5.01 "puts \"SALIENDO DE NS ...\" ; $ns_ halt"

# Definición del procedimiento 'stop', que se llama cuando
# finaliza la simulación.
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

# ============================================================
# Primer mensaje que aparece a la hora de la simulación.
puts "Comienza la simulación  "
$ns_ run
# ============== F I N ======================================
```

4.2. Modelo de simulación y metodología

Esta sección aborda las consideraciones generales de diseño que se han de tener en cuenta para simular LIFT. En primer término se describe el escenario general de simulación, después particularizamos en una descripción del entorno de simulación diseñado especialmente para LIFT. Posteriormente incluimos una tabla que lista todos los parámetros y rangos de valores utilizados en la simulación. Se describe también el modelo de movilidad empleado. Antes de entrar en los detalles de la simulación, se incluye la metodología que hemos seguido para simular y evaluar la propuesta. Finalmente, se describe propiamente la simulación realizada a LIFT.

4.2.1. Consideraciones del diseño de la simulación

Probar la vialidad de un protocolo con las características de LIFT (que descubre servicios en MANET basándose en un diseño Cross-layer, que agrupa los nodos de la red en clusters, que diferencia los dispositivos con capacidades superiores de los dispositivos ligeros, y que establece líderes de clusters a los dispositivos con capacidades superiores), implica tener en cuenta los siguientes criterios a la hora de realizar la simulación.

Primero, es necesario generar escenarios con nodos móviles heterogéneos altamente dinámicos, aplicar modelos de movilidad más reales (rescate, combate, conferencias y grupos) con patrones de movimientos aproximados al desplazamiento humano tanto en velocidad como aceleración. Segundo, tener presente que hipotéticamente en una MANET pueden existir HCD y LCD. Y Tercero, identificar y diferenciar los nodos con capacidades superiores; una vez identificados los HCD, elegir y establecer los HCD como proveedores de servicios.[3].

[3]Este tipo de nodo provee varios servicios, ya que se aprovechan sus capacidades superiores, de esta manera se evita que el resto de nodos desgasten sus recursos innecesariamente proporcionando tantos servicios.

4.2.2. Descripción y características del escenario

Los parámetros de número de nodos, número de clusters, número de líderes y movilidad pueden corresponder a escenarios reales como salas de conferencias, salas de reuniones, aulas de campus, etc., por lo general serían espacios cerrados, con gran número de usuarios llevando dispositivos móviles heterogéneos ofreciendo y demandando servicios, estos usuarios están agrupados temporalmente en espacios físicos y eventualmente son liderados por algún conferenciante, monitor, profesor, directivo, etc. Por lo que en el escenario de red utilizado para simular LIFT, existen c clusters integrados por los n nodos que forman la MANET, cada dispositivo ofrece s servicios que se almacenan en una tabla que está disponible en el entorno durante t tiempo. Este tiempo t se denomina tiempo de disponibilidad (*lifetime*) y es variable en cada dispositivo en función de su particular movilidad y actividad de envío y recepción de paquetes.

En las subsecciones posteriores se abordan los siguientes temas: En la subsección 4.2.2.1 Entorno de simulación, se describen las condiciones particulares de nuestro modelo, los parámetros aplicados a LIFT durante la simulación y una tabla en la que se recogen todos los parámetros y rangos de valores utilizados en la simulación. Y en la subsección 4.2.2.2 Modelo de movilidad, se describe el modelo de movimientos aplicado en la simulación de LIFT.

4.2.2.1. Entorno de simulación

Tomando en consideración que nuestra propuesta de solución se reduce a un dominio de aplicación circunscrito al ámbito de ambientes interiores, como aulas de conferencias, salas de reuniones, etc., hemos de tener en cuenta que en los escenarios para simular LIFT se utiliza una MANET con alcance dentro de espacios interiores, sin ninguna restricción de seguridad para anunciar y descubrir servicios en cualquier dispositivo participante; ningún evento se ejecuta sobre entornos desconocidos y se descarta la posibilidad de ejecutarse sobre redes públicas que trabajen en ambientes exteriores. Se considera la posible existencia de dispositivos con capacidades superiores y dispositivos con limitación de

recursos; que algunos HCD asumirán el rol de CL y que por la naturaleza del escenario que recién hemos descrito (espacios cerrados, salas de conferencias), los dispositivos participantes tienden a mantenerse más o menos dentro de un mismo emplazamiento, facilitando de esta forma la estrategia de agrupamiento y el establecimiento de líderes.

Para simular los escenarios con las características anteriores, hemos pensado en los siguientes parámetros y valores para la simulación de LIFT: nodos móviles altamente dinámicos (tanto de tipo HCD como de tipo LCD)[4]. Los nodos son colocados aleatoriamente sobre un área rectangular, con una duración de observación máxima de 600 segundos, con distintos valores de velocidad (2, 5, 10, 15 y 20 Km/h), con un rango de comunicación de 100 metros, usa IEEE 802.11b como tecnología de transmisión inalámbrica, 20 Julios de energía inicial, un total de 300 peticiones de servicios, 15 diferentes tipos de servicios y empleando el modelo de movilidad Random Waypoint (RWP, explicado en el apartado 2.10). En las simulaciones de LIFT hemos empleado básicamente el modelo de movilidad RWP pero modificando algunos parámetros, este modelo lo llamaremos Clustering Random Waypoint (CRWP)[5]. Es decir, hemos introducido la figura de un líder de grupo que se desplaza a una velocidad, aceleración y dirección aleatoria. La aceleración y velocidad del resto del grupo se ajusta periódicamente a las variaciones que ejecuta el líder de grupo y se han eliminado las pausas características del modelo RWP con la intención de reflejar un poco más la realidad de los escenarios que deseamos representar.

Otros parámetros más específicos son la topografía del escenario, que contempla un área total de 750 x 750 metros, fraccionada en áreas internas de 200 x 200 metros distribuidas uniformemente. El número de subáreas y de nodos en cada área total es variable dependiendo del experimento. Es decir, tenemos números de áreas ajustadas a escenarios para 100, 200, 300, 400 y 500 nodos. La razón de tener ese número de nodos se debe principalmente a que deseamos

[4]Para efectos de la simulación hemos generado estos nodos de manera aleatoria.

[5]Eligiendo el modelo RWP por ser un modelo muy aproximado a la realidad y combinado con conceptos de clustering por tratarse de grupos de nodos.

simular escenarios con clusters de 3, 4, 5, 6 y 7 CL, por lo que para conseguir clusters y escenarios con números enteros, hemos puesto clusters con 20, 33, 43, 50 y 56 nodos. Para unos experimentos especiales tenemos escenarios de 750, 1000, 1250 y 1500 nodos, dado que LIFT emplea técnicas de clustering debe soportar estos números de nodos y con ello comprobar cómo de escalable es LIFT.

En los escenarios con 100 nodos existen 5 áreas, cada área está integrada por 20 nodos, constituida por 3 CL y 17 CN. Para los escenarios con 200 nodos tenemos 6 áreas, cada área se compone de 33 nodos (4 CL y 29 CN). En escenarios con 300 nodos tenemos 7 áreas, cada área se compone de 43 nodos (5 CL y 38 CN). En escenarios con 400 nodos existen 8 áreas, en cada área hay 50 nodos (6 CL y 44 CN). Por último, en los escenarios con 500 nodos, tenemos 9 áreas compuestas por 56 nodos, cada área es liderada por 7 CL que dan soporte a los 49 CN restantes.

En la tabla 4.1 se resumen parcialmente los parámetros y valores referente a los clusters, líderes y miembros de clusters. Mientras que en la tabla 4.2 se recogen los parámetros y valores de aplicación general para todas las simulaciones, algunos de estos valores son constantes y algunos otros son rangos de valores.

No. de nodos	**No. de clusters**	**Nodos/cluster**	**HCD**	**LCD**	**CL**	**CN**
100	5	20	15	85	3	17
200	6	33	24	176	4	29
300	7	43	35	265	5	38
400	8	50	48	352	6	44
500	9	56	63	437	7	49

Tabla 4.1: Números de nodos, clusters y líderes de clusters

Respecto a la distribución de los CL en las áreas mencionadas anteriormente, podemos decir que inicialmente los CL se localizan en la zona central de cada área, luego comienzan a desplazarse hacia un punto fijo dentro del área, el desplazamiento es libre y aleatorio antes de llegar al punto predefinido, una vez que los CL alcanzan el punto meta, de nuevo vuelven a su punto de origen y

Parámetro	**Rango de valores**
Número de dispositivos	100, 200, 300, 400, 500
Número de dispositivos de tipo HCD	15, 24, 35, 48, 63
Número de dispositivos de tipo LCD	85, 176, 265, 352, 437
Número de clusters	5, 6, 7, 8, 9
Número de dispositivos en cada cluster	20, 33, 43, 50, 56
Número de CL en cada cluster	3, 4, 5, 6, 7
Número de tipos de servicios	15
Número de redundancia de servicios	4, 8, 12, 16, 21
Número total de proveedores de servicios	30, 48, 70, 96, 126
Número de proveedores de servicios tipo HCD	15, 24, 35, 48, 63
Número de proveedores de servicios tipo LCD	15, 24, 35, 48, 63
Tiempo de simulación	600 segundos
Área total de simulación	750 x 750 mts.
Número de subáreas de simulación	5, 6, 7, 8, 9
Subáreas de simulación	200 x 200 mts.
Modelo de movilidad	CRWP
Protocolo Ad Hoc	AODV
Protocolo MAC	IEEE 802.11b
Rango de transmisión (Tx)	100 mts.
Velocidad	2, 5, 10, 15, 20 Km/h.
Pausas	0 segs.
Energía inicial	20 Julios

Tabla 4.2: Parámetros para la simulación de LIFT

así sucesivamente hasta agotar el tiempo fijado en la simulación. Cabe resaltar que siempre que los CL llegan a un punto en específico, no se detienen en ningún momento, puesto que el modelo de movilidad empleado no contempla pausas.

Para ilustrar las posiciones iniciales de las áreas y de los CL, hemos puesto en la figura 4.3 los clusters que son liderados por 3 CL, estos CL han sido colocados por coordenadas en la parte central del cluster, mientras que los CN son colocados aleatoriamente (los CN no se visualizan en la figura debido al elevado número de nodos que son y al poco espacio disponible en la figura). De la misma forma se muestra en la figura 4.4 los grupos liderados por 4 CL, en la figura 4.5 los grupos liderados por 5 CL, en la figura 4.6 los grupos liderados

por 6 CL, y en la figura 4.7 los grupos liderados por 7 CL. En estas figuras no se han incluido los nodos de capacidades limitadas, ya que constituyen la mayoría de nodos en el escenario, además que su localización inicial dentro de las áreas es completamente aleatoria.

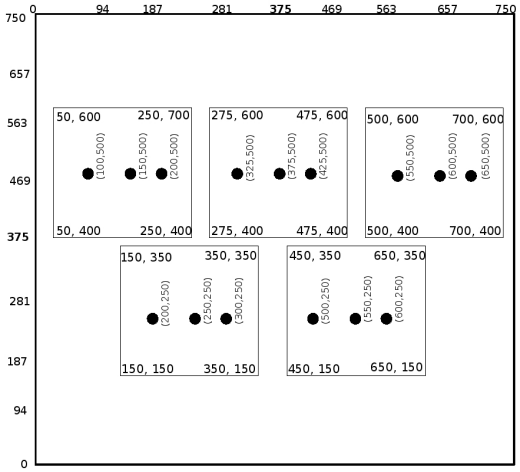


Figura 4.3: Localización inicial de los CL en escenarios con 100 nodos

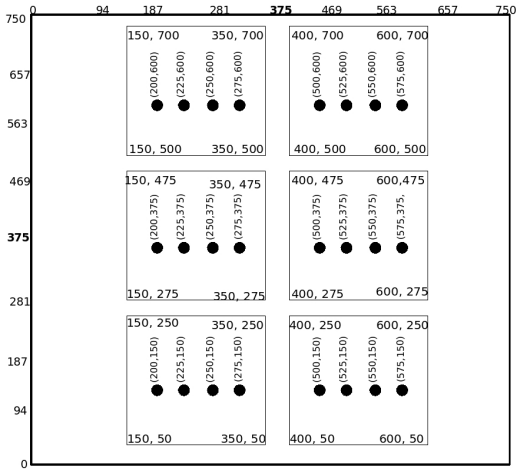


Figura 4.4: Localización inicial de los CL en escenarios con 200 nodos

En las figuras que hemos citado anteriormente aparecen una serie de puntos negros que simbolizan a los CL y unos cuadros que simbolizan las áreas de los

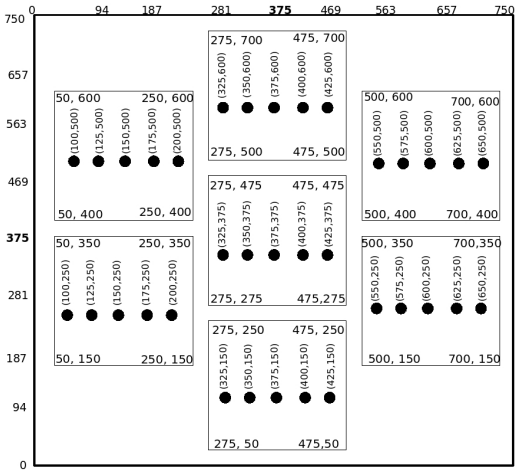


Figura 4.5: Localización inicial de los CL en escenarios con 300 nodos

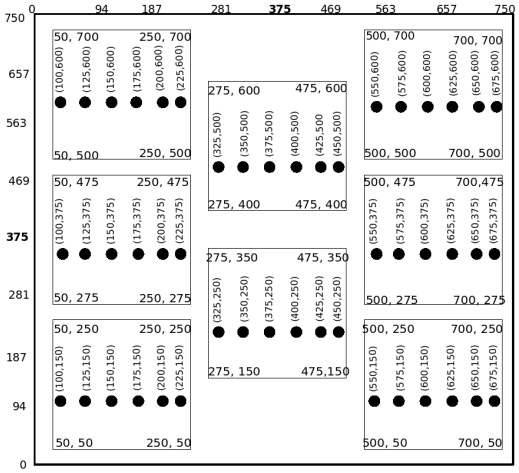


Figura 4.6: Localización inicial de los CL en escenarios con 400 nodos

clusters. Junto a estos puntos y cuadros existen unos valores numéricos que representan las coordenadas cartesianas de la localización inicial. El primer valor que aparece se corresponde con el eje x y el segundo valor con el eje y. Del mismo modo, los valores numéricos que aparecen en las esquinas de los cuadros se corresponden con las coordenadas x y y. Estos cuadros demarcan las áreas geográficas por donde se desplazarán los nodos móviles. Existe además un cuadro

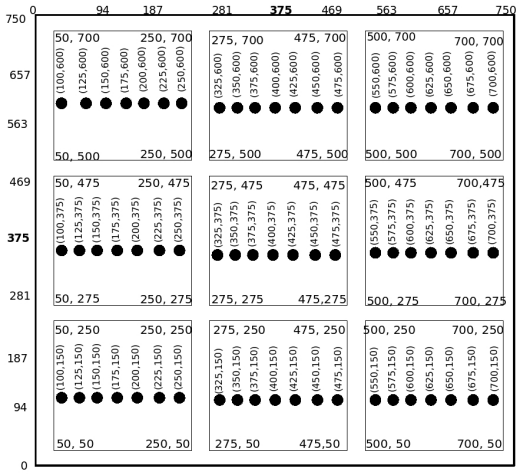


Figura 4.7: Localización inicial de los CL en escenarios con 500 nodos

más grande que encierra a todos los cuadros, este cuadro se corresponde con el área total del escenario de simulación.

4.2.2.2. Modelo de movilidad

Al final del capítulo 2 apartado 2.10 se repasan los principales modelos de movilidad existentes en la simulación de redes móviles Ad Hoc, los patrones de movimientos y los entornos en los que se pueden aplicar dichos modelos. En este apartado se describen las características del modelo de movilidad empleado en LIFT.

Para simular minuciosamente un nuevo protocolo de redes móviles Ad Hoc, se requiere el uso de modelos de movilidad que representen de forma precisa y realista los nodos móviles que utilizará el protocolo. De esta forma es posible darnos cuenta que el protocolo propuesto tendrá la utilidad esperada cuando sea implementado.

Debido a que LIFT tiene como estrategia agrupar nodos liderados por dispositivos con capacidades superiores, se ha elegido CRWP (Clustering Random Waypoint Model) como modelo de movilidad.

El modelo CRWP (modelo RWP adaptado a clusters) maneja la figura de líder de cluster, este líder se desplaza a una velocidad y aceleración aleatoria junto con el resto de miembros del cluster. La aceleración y velocidad del resto de miembros del cluster se ajusta periódicamente a las variaciones que ejecuta el líder del cluster. En el modelo de movilidad CRWP hemos eliminado las pausas que realiza el modelo RWP.

El modelo de movilidad CRWP además de ser modificado y adaptado para clusters, le hemos añadido la variante de eliminar las pausas y considerar que el conjunto de miembros del cluster es liderado por varios CL. Por otra parte, contrastando CRWP con el modelo RPGM, habitualmente en cada cluster existe un líder que es seguido por el resto de miembros del cluster. En el modelo CRWP el resto de miembros del cluster no persiguen o no se mueven alrededor del líder; por otro lado, en el escenario que hemos preparado para LIFT, hemos puesto más de un CL en cada cluster para asegurar y garantizar un liderazgo en todo momento en caso de pérdida de un único CL.

En este patrón de movilidad el escenario es una superficie plana y rectangular libre de obstáculos, en el que cada nodo:

- Se mueve en línea recta hacia un punto, con velocidad constante elegida de forma aleatoria según cierta distribución.
- Comienza de nuevo a moverse hacia el punto inicial, describiendo una trayectoria en zigzag.
- Los nodos nunca se detienen (el movimiento de los nodos es continuo), a diferencia del modelo Random Waypoint puro en el que los nodos se detienen durante un intervalo de tiempo variable.

4.2.3. Metodología y evaluación

Con el fin de alcanzar los objetivos planteados en el apartado 1.6, se han tenido que realizar las siguientes acciones:

1. Estudiar minuciosamente el simulador de redes Ns-2, analizando el uso y mecanismo de interacción entre C++ y OTcl, etc.

2. Estudiar y seleccionar el protocolo de encaminamiento más adecuado para redes Ad Hoc, en este caso hemos analizado la implementación del protocolo de encaminamiento AODV sobre Ns-2.

3. Estudiar y analizar un protocolo de descubrimiento de servicios con diseño Cross-layer que descubre servicios en la capa de red (AODV-SD, ver apartado 3.3).

4. Modificar parte del código[6] de la implementación AODV-SD y extender los formatos de mensajes con el fin de adecuarla a nuestra propuesta de solución.

5. Crear el código de los scripts en OTcl que envían los comandos al Ns-2 y éste a su correspondiente parte en C++.

6. Estudiar los distintos escenarios y modelos de movilidad que nos puedan ser útiles para simular LIFT.

7. Seleccionar el modelo de movilidad para diseñar los escenarios de la simulación.

8. Diseñar diferentes soluciones con el escenario de simulación seleccionado y elegir el escenario más próximo al logro de los objetivos impuestos.

9. Ejecutar una serie de simulaciones con parámetros que aumenten su complejidad: con 100 nodos, 200 nodos, 300 nodos, etc. con 5 servicios, 10 servicios, 15 servicios, etc. con 10 proveedores de servicios, con 15 proveedores, etc.

[6]El código original del protocolo AODV para Ns-2 fue diseñado de manera inicial por el grupo MONARCH de Carnegie Mellon University, posteriormente optimizado y añadido extensiones para Wireless por investigadores de University of Cincinnati, y posteriormente modificado por [95].

10. Observar los resultados de las simulaciones después de escalar algunos parámetros.

11. Analizar los resultados obtenidos sobre diferentes métricas: tasa de descubrimiento de servicios, sobrecarga de paquetes de control, consumo de energía, tasa de entrega de paquetes, throughput, promedio de saltos, carga de encaminamiento normalizada y tiempo de adquisición de servicios.

12. Comparar el rendimiento de LIFT con los resultados de AODV-SD.

13. Poner a disposición de los interesados el código fuente, a fin de que el trabajo pueda ser analizado, criticado, mejorado, modificado o incluido en trabajos futuros.

14. Divulgar mediante publicaciones los logros obtenidos en esta investigación.

4.3. Simulación de LIFT

En esta sección describimos las simulaciones que hemos realizado para estudiar el comportamiento del protocolo LIFT. Proporcionamos una explicación teórica detallada de cada una de las situaciones, justificando mediante un análisis cualitativo.

Los dispositivos se conectan a la red al comienzo de la simulación, solicitan y ofrecen servicios de forma aleatoria y no se desconectan de la red hasta finalizado el tiempo de simulación o una vez que se les haya agotado su energía. El número de dispositivos que existen en la red puede variar con el tiempo dependiendo la cantidad de energía disponible en cada dispositivo.

Los tiempos aleatorios de las peticiones de servicios, del dispositivo que realiza la petición de servicios y de la asignación de los servicios que ofrecen los dispositivos, siguen una distribución uniforme. Para tratar de ser lo más real posible, se considera que cada dispositivo puede ofrecer cualquier cantidad de servicios y algunos cuantos dispositivos no ofrecen ningún servicio.

Las peticiones de servicios a través de broadcasts se difunden dentro de cada cluster, de esta forma, las peticiones no inundan toda la red, con ello se reduce el tráfico global. No obstante, existen peticiones de servicios de dispositivos que se encuentran en la frontera de dos o más áreas; entonces esas peticiones de servicios traspasan los confines del cluster, difundiéndose estas peticiones a los clusters vecinos.

El tiempo de disponibilidad de los servicios en la red está ajustado al tiempo medio que permanecen los dispositivos en la red, para todos los experimentos la duración es de 600 segundos.

Los resultados de las simulaciones pueden tener variaciones si se modifican algunos de sus parámetros. En los distintos experimentos que se llevan a cabo se indica detalladamente el rango de valores utilizados para caso en particular y cuando no se indique algún parámetro de simulación distinto, entonces se trata del valor indicado en el escenario descrito en la tabla 4.2.

A manera de ejemplo para mejor ilustración de lo descrito anteriormente, explicaremos una simulación con 100 nodos, 5 clusters, 20 miembros por cluster y 3 CL en cada cluster.

- Los 5 clusters que integran el área de simulación general los denominaremos cluster *A*, *B*, *C*, *D* y *E*. Estas áreas dentro del área general están localizadas adyacentemente, 2 en la parte superior, 1 en el centro y 2 en la parte inferior (figura 4.8(a)).

- Los nodos de cada cluster se mueven libremente dentro de su área, nunca cruzan el perímetro imaginario de su propia área.

- La mayoría de las veces los broadcasts sólo se difunden dentro de la proximidad del nodo emisor y dentro del propio cluster. De tal manera que se reduce el tráfico en toda la red (figura 4.8(b)).

- Cuando un nodo desea descubrir un servicio, difunde por broadcast una petición de servicios, el mensaje de petición llega hasta donde existe cobertura del mensaje enviado.

- Los casos típicos deseables consisten en que las difusiones de peticiones sólo inunden el cluster en el que se origina el broadcast. En el ejemplo mostrado en la figura 4.8(c) observamos que distintos nodos en los clusters B, C y D lanzan secuencialmente peticiones de servicios, las peticiones inundan exclusivamente las regiones de los respectivos clusters. En consecuencia, el tráfico es local y se evita inundar toda la red y consumir un gran número de recursos.

- Pocas veces ocurre que la señal de petición de servicios de un nodo alcance a un nodo perteneciente a otro cluster (aunque el servicio solicitado se haya descubierto ya dentro del mismo cluster) y la petición inunde más allá del cluster en el que se ha originado la petición.

- En la figura 4.8(d) ilustramos el siguiente ejemplo: el nodo a del cluster B envía una petición de servicios, la señal se difunde dentro del cluster B, la señal llega hasta el nodo d que también es miembro del cluster B; el nodo d reenvía la señal y esta señal alcanza al nodo i que pertenece al cluster C, y el nodo i reenvía la petición y ésta alcanza al nodo m que también pertenece al cluster C; el nodo m reenvía la señal y alcanza al nodo p que igualmente está dentro del cluster C, y así sucesivamente.

- Lo anterior ha ocurrido debido a que el nodo i se localiza o se ha desplazado a un área de intersección entre los clusters B y C, la petición originada por el nodo a ha sido recibida por el nodo i por estar en el radio de alcance del nodo emisor a y porque la difusión es múlti-salto, y el nodo i ha reenviado la petición hacia el cluster C. Esta situación es ideal cuando algún servicio solicitado no se encuentra en el cluster B, pero es indeseable para el caso cuando el servicio sí se encuentra disponible en el cluster `B`.

- En relación al consumo de recursos, se observa en la figura 4.8(d) que los nodos que se ubican dentro del cluster C, sufren mayor desgaste de recursos a causa del trabajo adicional añadido al trabajo habitual de descubrir servicios dentro de su propio cluster.

- En este análisis no existen incidencias de casos de colisiones de mensajes, puesto que el tráfico de petición de servicios no se realiza de manera simultánea. Es decir, hasta que un evento descubre o no un servicio solicitado, entonces no se genera otra petición.

- En general, creemos que existen muy pocas peticiones de servicios fallidas o muy pocos fracasos en el descubrimiento de servicios. De acuerdo a los resultados de la simulación, la tasa de servicios que no han sido descubiertos respecto al número total de peticiones de servicios es muy baja y cercana al 6 %.

- Por lo tanto, al existir pocas peticiones de servicios no logradas, la tasa de descubrimiento de servicios es muy alta.

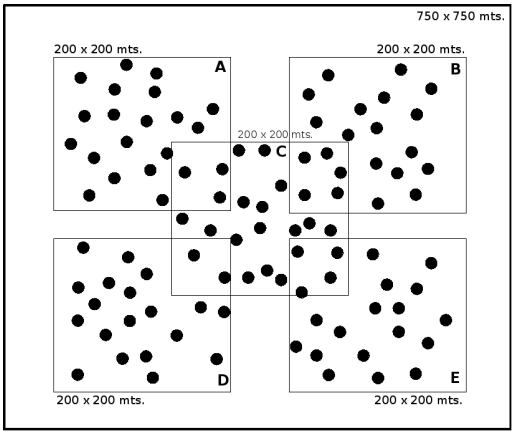
- Para reforzar lo mencionado anteriormente, en el gráfico 4.9 mostramos la tasa de servicios descubiertos obtenida a partir de una serie de simulaciones.

La tasa de descubrimiento de servicios nos permite percibir a nivel de usuario, el éxito que tiene un SDP para descubrir servicios, es una métrica para estimar la efectividad. Esta tasa se obtiene a partir del total de servicios descubiertos dividido por el número de peticiones de servicios enviadas.

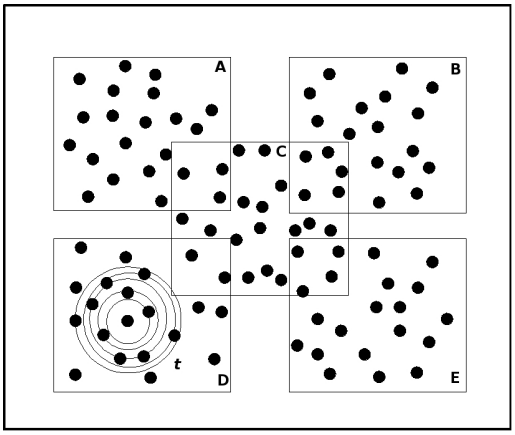
En las simulaciones realizadas a LIFT esta tasa llega a ser de 92,87 % en su valor más alto; por lo que el porcentaje obtenido es muy razonable.

Gráficamente se puede apreciar la trayectoria ascendente y progresiva que trazan las curvas en relación al eje x. Se puede observar que cuanto menor es el número de peticiones de servicios y mayor es el número de nodos, ligeramente mayor es la tasa. Por lo que es clara la influencia que el número de nodos ejerce sobre esta tasa, mientras que la influencia que presenta el número de peticiones de servicios no es muy significativa.

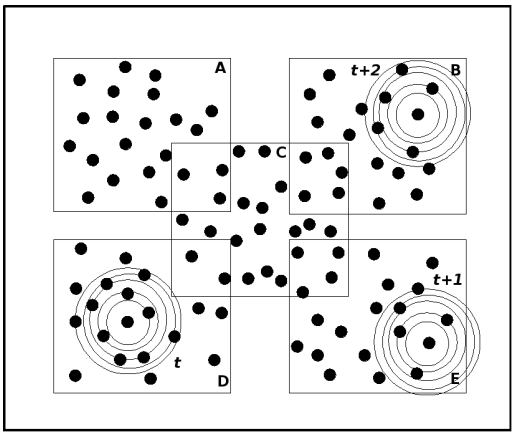
Esto es todo en cuanto a la metodología para validar nuestra propuesta mediante simulación. Hemos dado los pormenores acerca de la metodología de



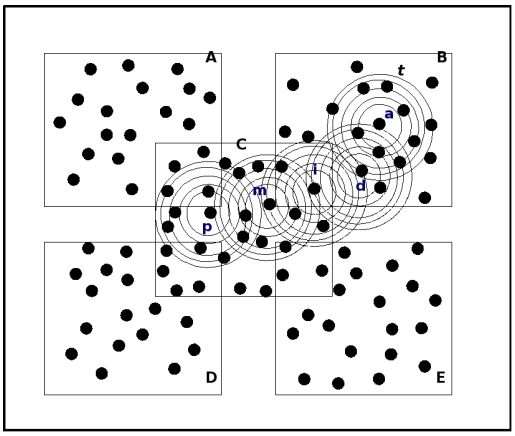
(a) MANET con 100 nodos extendidos sobre 5 áreas



(b) Broadcast dentro de un cluster



(c) Broadcasts realizados dentro de cada cluster



(d) Broadcast que traspasa el área de un cluster

Figura 4.8: Broadcasts en los clusters

simulación, la herramienta de simulación (Ns-2), la descripción de los escenarios de simulación, el modelo de movilidad y los parámetros y valores que se usarán en la simulación.

En el capítulo 5 se lleva a cabo un estudio más profundo sobre el rendimiento de LIFT. Evaluamos más métricas como: sobrecarga de paquetes de

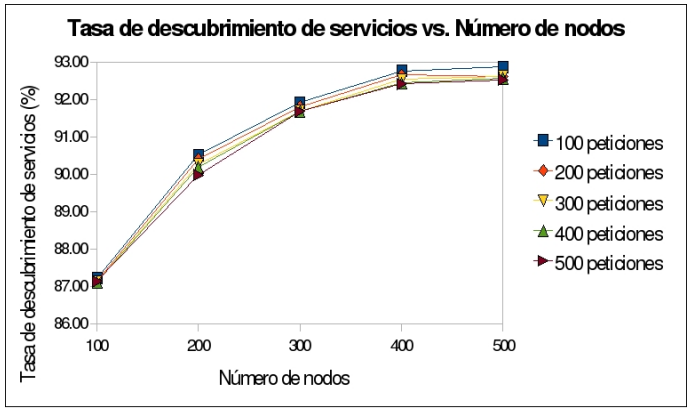


Figura 4.9: Tasa de descubrimiento de servicios en función del número de nodos y del número de peticiones de servicios. Experimentos realizados con los parámetros de la tabla 4.2 con un valor constante de movilidad de 10 Km/h.

control, consumo de energía, tasa de entrega de paquetes, throughput, promedio de saltos, carga de encaminamiento normalizada y tiempo de adquisición de servicios.

Capítulo 5

Estudios de rendimiento

En el capítulo 4 hemos descrito la metodología para validar la propuesta basada en simulación. Antes de nada, se ha presentado la herramienta de simulación Ns-2 utilizada para simular LIFT. Posteriormente se ha descrito el método de simulación y se han listado los elementos que contiene el escenario de simulación, los parámetros y valores empleados. Todo lo realizado en el capítulo 4 es la preparación para la simulación del protocolo y es ahora cuando vamos a simular el protocolos en su totalidad.

A lo largo del presente capítulo hablaremos sobre todo lo relacionado con la evaluación del rendimiento de LIFT respecto a diferentes parámetros; en concreto, veremos cómo se comporta dependiendo del número de dispositivos, la movilidad, la redundancia de servicios y el número de peticiones. Y en un apartado final contrastaremos el rendimiento de LIFT con otro protocolo de descubrimiento de servicios.

Con las métricas elegidas buscamos comprobar en términos generales, si LIFT permite reducir considerablemente el número de mensajes transmitidos por búsqueda de servicio, si el promedio de saltos para alcanzar el destino es aceptable, si el número de mensajes de control que emplea para descubrir servicios no es tan elevado, si sus tiempos de respuestas son tolerables, si sus tiempos de latencias no son muy prolongados, si la cantidad de nodos que soporta es suficiente para creer que es escalable y si posee altas tasas de servicios descu-

biertos y por último, si el número de dispositivos que descubren servicios en su mayoría son de tipo HCD. Con todo lo anterior, habremos cumplido en parte los objetivos impuestos en esta investigación.

Debido a que LIFT es una optimización, únicamente deseamos ver sus mejoras y beneficios, no es propiamente un nuevo algoritmo de descubrimiento de servicios. Para realizar una comparativa se requiere disponer de un protocolo que tenga características similares a LIFT, es decir, encontrar un protocolo que descubra servicios en redes móviles Ad Hoc basándose en un diseño Cross-layer, que opere con un protocolo de encaminamiento reactivo como AODV, que forme clusters de nodos, que distinga HCD de LCD y que los HCD lideren los clusters de la red para mediar en el descubrimiento de servicios. De los protocolos de descubrimiento de servicios revisados, ninguno de ellos tiene todas las características de LIFT, por lo que no es posible realizar ninguna comparativa al respecto. Por otra parte, creemos que lo valioso del modelo propuesto, es la combinación del diseño Cross-layer con el esquema de liderar clusters por dispositivos de altas prestaciones.

Este capítulo está estructurado de la siguiente manera: Primeramente, en la sección 5.1 damos a conocer los dos tipos de dispositivos presentes en los escenarios de simulación de nuestro modelo. Seguido de la sección 5.2 en la que enumeramos las métricas empleadas para estimar el rendimiento de los protocolos. A continuación, en la sección 5.3 abordamos el tema de la minimización de recursos relacionándolo con el ancho de banda y el gasto energético. En la sección 5.4 describimos los detalles de la obtención de las siguientes métricas: tasa de entrega de paquetes, promedio de saltos, throughput, carga de encaminamiento normalizada y tiempos de respuestas. Posteriormente, en la sección 5.5 detallamos la medición realizada para conocer las posibilidades de escalabilidad de LIFT. En la sección 5.6 nos centramos en la comparativa entre LIFT y AODV-SD. En la sección 5.7 presentamos un análisis cualitativo de los resultados obtenidos y las conclusiones de todas las mediciones realizadas. Finalizando

con la sección 5.8 en la que describimos los escenarios idóneos en donde LIFT tiene oportunidades de aplicación.

5.1. Tipos de dispositivos

En este apartado mencionaremos los tipos de dispositivos presentes en los escenarios de simulación. Hemos definido anteriormente que LIFT está diseñado para trabajar con dispositivos heterogéneos y supone la existencia de dos tipos diferentes de dispositivos: HCD.- dispositivos con capacidades superiores y ricos en recursos, y LCD.- dispositivos limitados en recursos y con prestaciones reducidas.

LIFT aprovecha la existencia de los HCD a fin de ser elegidos intermediarios en el proceso de descubrimiento de servicios. La distinción del tipo de dispositivo se obtiene mediante una clasificación en base a las capacidades que posee el propio dispositivo, una vez realizada esta diferenciación, los que resultan ser de tipo HCD son elegidos líderes de clusters y de esta forman se encargan de mediar en el descubrimiento de servicios dentro de su respectivo cluster.

La clasificación de HCD la obtiene un nodo, a partir de la capacidad que éste tenga respecto a las prestaciones de hardware (memoria, batería, potencia de señal, procesador). Suponemos que un dispositivo es consciente de las prestaciones que posee. De esta forma, a un nodo se le asigna un identificador del nodo propiamente y un identificador binario (1 cuando se trata de un HCD ó 2 cuando se trata de un LCD), este identificador define el tipo de nodo. Y es así como se determina cuándo un nodo es HCD.

En [98] por ejemplo, emplean un mecanismo similar a LIFT para determinar la capacidad de un nodo. El mecanismo consiste en asignar un identificador único denominado *dirección del nodo* y un peso denominado *grado de capacidad*, que representa un cálculo de la dinámica del nodo y la disponibilidad de recursos. A mayor grado de capacidad mayor aptitud del nodo para desempeñar el rol de líder.

Para fines de simular la presencia de HCD en el escenario de simulación, en el script de OTcl (Apéndice A) asignamos los s servicios a los p proveedores y también asignamos arbitrariamente el tipo de dispositivo a cada uno de los n nodos que existen en el escenario. Es decir, con ello determinamos de qué tipo es cada dispositivo. A continuación mostramos un trozo del código en donde se asignan servicios a proveedores. Podemos ver que se asignan cinco distintos servicios al nodo 0, al nodo 1, al nodo 2, y así sucesivamente hasta el nodo 14.

```
.
.
# Se asignan 15 servicios a los HCD 0 .. 14
for {set j 0} {$j < 15} {incr j} {
set cliente($j) [$node_($j) set ragent_]
}
$ns_ at 1.0 "$cliente(0) service-bind 1 DISK"
$ns_ at 1.0 "$cliente(0) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(0) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(0) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(0) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(1) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(1) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(1) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(1) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(1) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(2) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(2) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(2) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(2) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(2) service-bind 1 WEB"
.
.
```

El gráfico 5.1 muestra el número de HCD y de LCD que hemos utilizado en la mayoría de los experimentos. Para información más detallada acerca de la distribución exacta en cada simulación se puede consultar la tabla 4.1.

Dado que en los escenarios simulados existen dos tipos distintos de dispositivos y ambos tienen la misma opción de proveer servicios, entonces consideramos oportuno mostrar en el gráfico 5.2 el tipo de dispositivo que más servicios descubre durante la simulación. Por ejemplo, en el experimento realizado en un escenario compuesto por 100 nodos, el 93 % de los nodos que descubren servicios

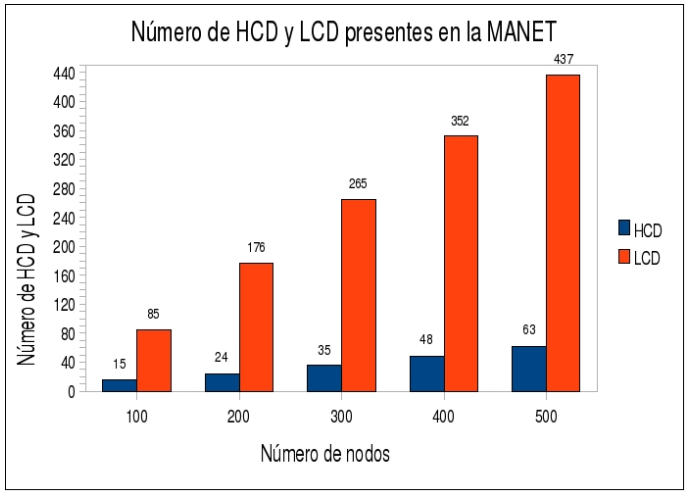


Figura 5.1: Número de tipo de nodos que integran el escenario.

son de tipo HCD, mientras que sólo el 7 % son nodos de tipo LCD. No obstante, el número de proveedores de servicios se ha asignado equitativamente. En todos los escenarios con distintos números de nodos, el porcentaje de proveedores representa el 50 % para cada tipo de dispositivo.

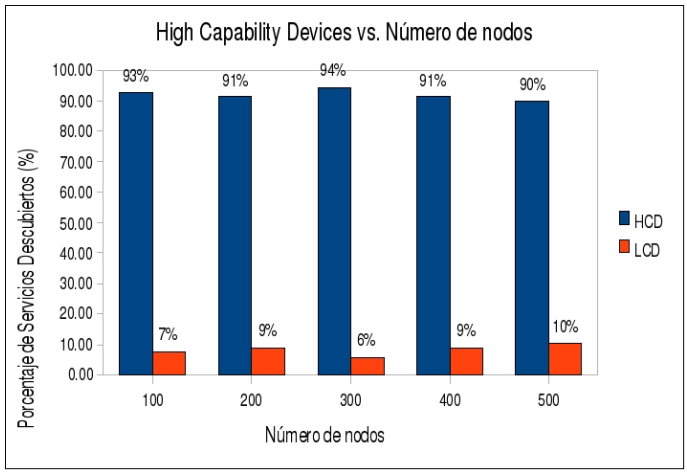


Figura 5.2: Porcentaje de nodos tipo HCD que descubren servicios.

5.2. Medidas particulares sobre el protocolo

El grupo de trabajo de MANET de la IETF (evaluación de protocolos routing/multicasting) [38] [132] sugiere una lista de métricas para evaluar el rendimiento de un protocolo. Algunas de estas métricas se han utilizado en trabajos como [12] [158] [114] [155] [26] por mencionar sólo unos cuantos.

Veremos a continuación una descripción de las medidas utilizadas en la serie de experimentos realizados a LIFT, con objeto de evaluar su comportamiento y medir su rendimiento.

1. *Tasa de entrega de paquetes* (Packet Delivery Ratio) .- También conocida como *probabilidad de éxito*. Se define como el número de paquetes de datos recibidos por el destino final como una fracción del número de paquetes de datos transmitidos por el origen. En otras palabras, es la tasa del número de paquetes de datos entregados a los destinos vs. el número de paquetes de datos recibidos. Este número representa la efectividad del protocolo.

2. *Throughput* .- Se define como la velocidad de transmisión de los datos. Es el total de todos los bits o paquetes exitosamente entregados a destinos individuales. Al final lo que realmente interesa es el número de paquetes entregados exitosamente. Por consiguiente, es una medida de efectividad de un protocolo.

3. *Promedio de longitud de ruta* (Average Number of Hops).- El número promedio de saltos se refiere a la cantidad de saltos que tiene que realizar un *request* de un nodo origen para alcanzar el servicio solicitado. Un *request* lanzado por un nodo pasa a través de n nodos para alcanzar el nodo destino.

4. *Retardo extremo a extremo* (End-to-End Delay) .- Se define como la diferencia de tiempo transcurrida entre el instante que se genera el paquete y el momento que éste llega a su destino. Es el promedio de retardo que un paquete toma para llegar del nodo origen al nodo destino.

5. *Sobrecarga de paquetes de control* (Control Message Overhead) .- El número de mensajes generados por el protocolo de descubrimiento de servicios y por el protocolo de encaminamiento. Esta métrica nos permite observar la eficiencia de un protocolo de descubrimiento de servicios con respecto al uso de ancho de banda disponible.

6. *Consumo energético* .- Tiene como propósito estimar la energía consumida por los dispositivos en el escenario. Se calcula como la diferencia de la energía inicial asignada a todos los dispositivos menos la energía que queda en los dispositivos terminada la simulación.

7. *Carga de encaminamiento normalizada* (Normalized Routing Load) .- Se refiere al número de paquetes de encaminamiento transmitidos por paquete de datos entregados en el destino. Mide la sobrecarga introducida por el protocolo de encaminamiento de la MANET y se calcula como el porcentaje de paquetes de control sobre el porcentaje de paquetes de datos. El óptimo ideal es una carga de encaminamiento normalizada cercana a 1, significa que para entregar un paquete de datos sólo es necesario transmitir dicho paquete, esto es, que no son necesarios los mensajes de control.

8. *Tiempo de adquisición de servicios* (Service Acquisition Time) .- El tiempo que transcurre desde el envío del primer *request* hasta la recepción del primer *reply* válido. Es decir, el tiempo de duración de un *reply* para llegar al origen, después de que éste ha enviado un *request*. Esta métrica nos permite estimar cómo de rápido descubre servicios un SDP.

9. *Tasa de descubrimiento de servicios* .- Representa el porcentaje del número de servicios descubiertos respecto al número total de servicios solicitados. Esta medida se obtiene a partir de la división del número de servicios descubiertos entre el número de peticiones de servicios lanzadas. Esta métrica nos da una idea de la eficacia del protocolo.

La tabla 5.1 lista las nueve medidas anteriores y describe el propósito que tiene cada medida, en la serie de experimentos aplicados a LIFT.

El número de nodos móviles, el comportamiento de esos nodos (movilidad) y la calidad de los enlaces, suelen ser los principales parámetros que afectan el rendimiento de una MANET [18]. Por consiguiente, la razón de representar algunas de estas métricas en la serie de simulaciones que haremos, se debe principalmente a que el número de nodos y la movilidad ejercen influencia sobre estos parámetros. De este forma, creemos que el número de nodos y la movilidad son factores determinantes para medir el rendimiento de LIFT.

Para evaluar el comportamiento de la MANET bajo el SDP propuesto, hemos optado cuantificar estas medidas mediante las métricas descritas anteriormente. Para la optimización de recursos mediremos la *sobrecarga de paquetes de control* en la sección 5.3.1 y el *consumo energético* en la sección 5.3.2. Y para conocer el comportamiento de LIFT con respecto a las métricas, estimaremos la *tasa de éxitos* en la sección 5.4.1, el *promedio de saltos* en la sección 5.4.2, el *throughput* en la sección 5.4.3, la *carga de encaminamiento normalizada* en la sección 5.4.4 y *tiempo de adquisición de servicios* en la sección 5.4.5.

5.3. Minimización de recursos

Los dispositivos de una MANET son muy limitados en cuanto a energía, memoria y comunicación. Debido a la movilidad de los propios nodos se demanda gran cantidad de recursos. Por tal razón es importante minimizar tales recursos.

Se consumen recursos en cada mensaje que pasa por un nodo, bien un paquete de control o bien un paquete de datos. Prácticamente, cada petición de servicio enviada requiere transmitir mensajes llevando información de encaminamiento y de descubrimiento de servicios, y a su vez se genera una respuesta a dicha petición, que también genera paquetes. Además, una vez establecida la comunicación se necesita estar permanentemente conectado y esto también ocasiona consumir gran número de recursos.

Medida	**Objeto de medición**
Sobrecarga de paquetes de control	Mide el número de mensajes de descubrimiento de servicios y de encaminamiento generados. Mide la eficiencia del protocolo respecto al ancho de banda.
Gasto energético	Mide la media de la energía consumida por los dispositivos.
Tasa de entrega de paquetes	Mide el número de paquetes de datos recibidos en relación al número de paquetes enviados. Mide efectividad del protocolo.
Promedio de saltos	Mide el número promedio de saltos necesarios para descubrir un servicio.
Throughput	Mide la velocidad de transmisión de datos o la tasa de bits. Mide la efectividad del protocolo.
Carga de encaminamiento normalizada	Mide la sobrecarga de encaminamiento introducida por el protocolo.
Tiempo de adquisición de servicios	Mide cómo de rápido descubre servicios el protocolo.
Tasa de descubrimiento de servicios	Mide el porcentaje de servicios descubiertos en relación al número de petición de servicios. Mide el éxito del protocolo para descubrir servicios.

Tabla 5.1: Resumen de las mediciones aplicadas a LIFT

Uno de los recursos que más se consumen es el ancho de banda, y éste, a su vez influye directamente sobre el consumo de energía y memoria, porque al existir demasiado tráfico en la red, la sobrecarga de paquetes de control es muy elevada, en consecuencia, los dispositivos tendrán mayor carga de trabajo y mayor consumo de energía y cómputo.

Por su parte, el consumo de energía es un recurso muy importante a tener en cuenta en el ámbito de las MANET. Este consumo energético llega a ser considerablemente elevado cuando existe demasiado tráfico en la red. Es obvio, que los dispositivos cuando transmiten mensajes tienen mayor gasto energético

que cuando están recibiendo mensajes. Consecuentemente, con una reducción en el número de mensajes por petición y en la sobrecarga de paquetes, podemos ahorrar energía.

El comportamiento de la MANET con respecto a la minimización de recursos (ancho de banda y energía) se ha cuantificado mediante el empleo de las siguientes medidas: *sobrecarga de paquetes de control* para medir el consumo de ancho de banda y *consumo de energía* para conocer el gasto energético.

5.3.1. Ancho de banda

Para averiguar la eficiencia del protocolo de descubrimiento de servicios respecto al ancho de banda disponible, es preciso conocer la cantidad de mensajes generados por el propio SDP y por el protocolo de encaminamiento.

La cantidad de mensajes generados por el protocolo tiene un impacto en el ancho de banda. Por otra parte, el descubrimiento de servicios es un proceso que se desarrolla con cierta periodicidad, por ello es deseable que este proceso ocupe la menor cantidad de mensajes.

En este apartado emplearemos la métrica *sobrecarga de paquetes de control* (**Control Message Overhead**) para evaluar el funcionamiento interno del protocolo. La sobrecarga de paquetes de control expresa el número de mensajes generados por el protocolo de descubrimiento de servicios y por el protocolo de encaminamiento.

Con el objetivo de comprobar qué parámetro impacta más en la sobrecarga de paquetes de control, si la movilidad o el número de peticiones de servicios, hemos optado por realizar dos experimentos. Uno para medir la *sobrecarga de paquetes de control vs. número de nodos* con distinta movilidad y otro para cuantificar la *sobrecarga de paquetes de control vs. número de peticiones de servicios.*

En el gráfico 5.3(a) se muestran los valores promedios obtenidos de la métrica *sobrecarga de paquetes de control vs. número de nodos* con diferentes velocidades de movilidad. Los valores usados en esta medida se indican en la tabla 4.2. Este experimento se lleva a cabo con un rango de valores de velocidad

de 2, 5, 10, 15 y 20 Km/h. y enviándose 300 peticiones de servicios para todos los casos.

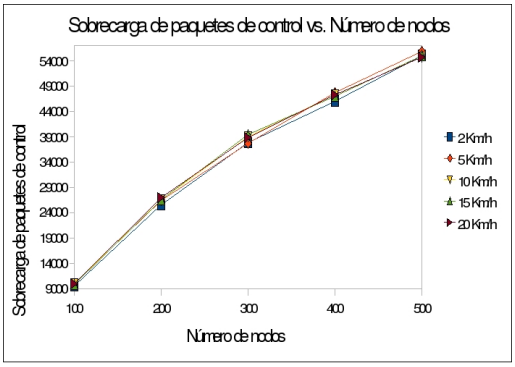
Podemos apreciar en el gráfico 5.3(a) que las distintas curvas que representan la sobrecarga de paquetes de control, no tiene una diferencia significativa a medida que cambia el comportamiento de los nodos. Casi es nula la diferencia entre las curvas. Sin embargo, esas curvas aumentan progresivamente de forma pronunciada respecto al número de nodos. En este sentido, el crecimiento que se observa es lineal, aunque las 5 curvas crecen de manera similar y se solapan en algunos puntos, casi se obtiene el mismo patrón de crecimiento conforme aumenta el número de nodos.

En el gráfico 5.3(b) mostramos los resultados del estudio respecto a la *sobrecarga de paquetes de control vs. número de peticiones de servicios*, tales experimentos se han basado en el escenario y variables que se describen en la tabla 4.2, y hemos hecho las pruebas con un rango de valores de peticiones de servicios de 100, 200, 300, 400 y 500 requests, y una velocidad media de desplazamiento de 10 Km/h.

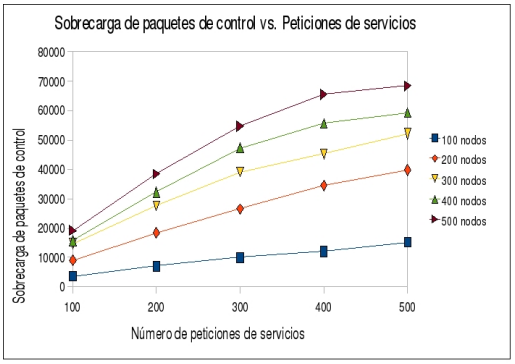
Se observa en el gráfico 5.3(b), que a medida que aumenta el número de nodos en la red, es notable la diferencia entre las curvas. Parece claro que a medida que aumenta tanto el número de dispositivos como el número de peticiones de servicios, crece considerablemente la sobrecarga de paquetes de control.

A la vista de los resultados obtenidos podemos concluir que el comportamiento de los nodos, no afecta demasiado al crecimiento de la sobrecarga de paquetes de control como afecta el número de peticiones de servicios, puesto que las curvas del gráfico 5.3(a) no muestran diferencias al aumentar la movilidad. Mientras que las curvas del gráfico 5.3(b) muestran una clara diferencia entre las curvas a medida que aumenta el número de peticiones de servicios.

En ambas series de experimentos, el número de nodos juega un papel decisivo en la sobrecarga de paquetes de control y hasta cierto punto es normal. Sin embargo, con los otros parámetros se concluye lo siguiente: La sobrecarga de paquetes de control se ve poco influenciada por el comportamiento de los



(a) Sobrecarga de paquetes de control vs número de nodos



(b) Sobrecarga de paquetes de control vs. peticiones de servicios

Figura 5.3: Sobrecarga de paquetes de control, con 300 peticiones de servicios y a 10 Km/h como media de velocidad.

nodos, por lo que la movilidad no es un factor decisivo. Por su parte, el número de peticiones de servicios sí ejerce influencia sobre la sobrecarga de paquetes de control, en este caso, las peticiones de servicios sí son un factor determinante.

Está claro que una mayor cantidad de nodos y un mayor número de peticiones de servicios, introduce mayor sobrecarga de paquetes de control. La explicación es que al aumentar las peticiones de servicios, es normal que aumente el tráfico en la red y por tanto aumente también el consumo del ancho de banda. En cambio, cuando el número de nodos es menor, la eficiencia de la red aumenta (existe menor tráfico de control y por consiguiente menor congestión) permitiendo optimizar el ancho de banda de la red.

Como era de esperar, un SDP basado en diseño Cross-layer que aprovecha los mensajes de control del protocolo de encaminamiento para enviar sus propios mensajes de control, sin duda, reduce la sobrecarga de paquetes de control; gracias a que desaparecen por completo los mensajes de control de descubrimiento de servicios de la capa de aplicación. En este caso, se puede decir que LIFT realiza un consumo adecuado del ancho de banda. Más adelante en el apartado 5.6, comprobaremos que lo hace mejor que AODV-SD.

Debemos tener en cuenta que si la sobrecarga de paquetes de control es alta, el ancho de banda disponible para los datos disminuye. Por lo tanto, es deseable que un SDP introduzca menor cantidad de mensajes de control para descubrir servicios. La presencia de un valor de sobrecarga de paquetes de control elevado es un problema, ya que congestiona la red y satura el ancho de banda.

En el apartado 5.6 hacemos una comparación entre LIFT y AODV-SD con objeto de demostrar que nuestra propuesta minimiza el uso de ancho de banda.

5.3.2. Gasto energético

En una MANET el consumo de energía es uno de los fenómenos más complicados y son muchos los factores que contribuyen a este consumo energético. Estos factores pueden estar relacionados a diferentes actividades de la red como: transmisión y recepción de mensajes, a factores relacionados con la manipulación de los protocolos de red (el protocolo MAC es de los más relevantes) y a factores dinámicos relacionados con la carga de tráfico [65].

En este estudio hemos evaluado el consumo de energía con el objetivo de conocer en qué medida LIFT minimiza el consumo de este recurso. Para tal efecto, hemos hecho dos mediciones. La primer medida toma la media de la energía consumida en función del número de nodos y la movilidad, mientras que la segunda medida toma la media de la energía consumida en función del número de peticiones de servicios y el número de nodos.

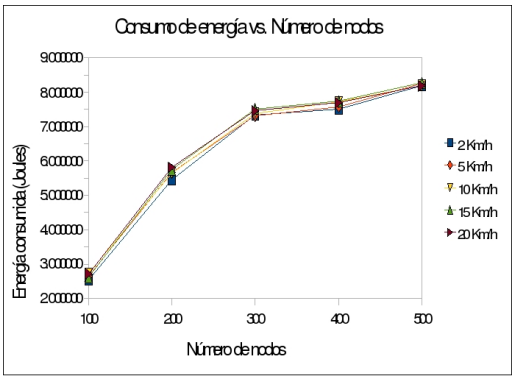
La tabla 4.2 indica los valores empleados para estos experimentos. Inicialmente hemos ajustado el valor de energía en 20 Julios[1], este valor realmente no tiene ninguna importancia, solamente nos es de utilidad para calcular la diferencia entre el valor de energía inicial y el valor de energía restante una vez terminada la simulación.

El gráfico 5.4(a) resume los valores promedios de *energía consumida vs. número de nodos* con distinta movilidad. Se ha realizado este experimento con

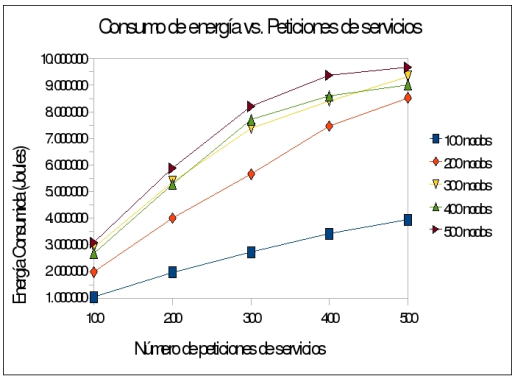
[1]Un Julio (J) o Joule es el trabajo producido por una fuerza de 1 newton, cuyo punto de aplicación se desplaza 1 metro en la dirección de la fuerza. El Julio es igual a 1 watts (vatio) x segundo (J = W x s).

los rangos de velocidad de desplazamiento 2, 5, 10, 15 y 20 Km/h y un valor de petición de servicios de 300 requests.

Se observa que el consumo energético crece conforme aumenta la movilidad de los nodos, es decir, aunque entre las curvas de las distintas velocidades de desplazamiento no es muy significativo el gasto energético, este gasto como es lógico, sí es más significativo en función del número de nodos. Las diferencias obtenidas en las diferentes curvas no son muy significativas. Podemos ver que las 5 curvas ascienden desde un consumo mínimo de 2,5 Julios hasta un consumo máximo de casi 8,3 Julios.



(a) Energía consumida en función del número de nodos



(b) Energía consumida vs. peticiones de servicios

Figura 5.4: Energía consumida, con 300 peticiones de servicios y a una movilidad constante de 10 Km/h.

La otra medición para conocer la energía consumida por LIFT aparece en el gráfico 5.4(b), que muestra los valores promedios de *energía consumida vs. número de peticiones de servicios*. En este experimento se han utilizado un rango de valores de petición de servicios que van desde 100 hasta 500 requests, a una velocidad media de desplazamiento de 10 Km/h.

Parece claro que el número de peticiones de servicios en combinación con el número de nodos, sí influye significativamente en el consumo de energía.

Se observa en el gráfico 5.4(b) lo siguiente: a medida que aumenta el número de peticiones y el número de nodos, las curvas ascienden separademente entre sí, desde un consumo inferior de 1,04 Julios hasta un valor máximo de 9,66 Julios. Se observa también que a medida que el número de nodos es mayor, la diferencia entre las curvas contiguas es menor.

Una vez realizadas las dos mediciones para conocer cómo se consume más energía, - si a mayor número de nodos y movilidad, mayor consumo energético o - si a mayor número de peticiones de servicios, mayor energía consumida, se concluye lo siguiente:

En las curvas del gráfico 5.4(a) que se corresponden con el número de nodos con distinta movilidad, el aumento de la movilidad no representan gran crecimiento en el consumo energético, debido a que el comportamiento de los nodos no introduce mayor tráfico en la red. Por el contrario, la trayectoria ascendente de las curvas se debe principalmente al número de nodos. A medida que el número de nodos aumenta, las curvas tienen un crecimiento similar.

En tanto que las curvas del gráfico 5.4(b) que se corresponden con el número de peticiones de servicios, sí representan diferencias en la energía consumida, ya que las peticiones de servicios introducen más tráfico en la red, por consiguiente, aumenta la cantidad de mensajes y la sobrecarga de paquetes.

Como se había mencionado al inicio de este apartado, el envío y recepción de mensajes y la carga de tráfico afectan el consumo de energía en las MANET. Por tanto, si se desea reducir el consumo de energía es conveniente minimizar el número de transmisiones de mensajes.

5.3.3. Conclusiones

De las mediciones realizadas para comprobar el grado de optimización de recursos tenemos lo siguiente.

Respecto a la *sobrecarga de paquetes de control* que hemos utilizado como parámetro para medir el ancho de banda, se ha visto que el comportamiento de los nodos no influye excesivamente en el aumento de la sobrecarga, en cambio,

el número de peticiones de servicios introduce una notoria sobrecarga. Esto es debido a que el número de peticiones de servicios, como es lógico, genera mayor cantidad de mensajes que se ven reflejados en una mayor carga de tráfico. En las dos mediciones realizadas (sobrecarga de paquetes de control en función del número de nodos con distinta movilidad y sobrecarga de paquetes de control en función del número de peticiones de servicios), sin duda el número de nodos es un factor clave en la sobrecarga de paquetes de control. Por el contrario, la movilidad no es un factor determinante en la sobrecarga.

En cuanto al *gasto energético*, éste crece conforme aumenta el número de nodos. La combinación del número de nodos con el número de peticiones de servicios influye considerablemente en el consumo de energía. La razón es que al momento de ser lanzada una petición de servicio, se transmiten mensajes de encaminamiento que a su vez generan más mensajes y más tráfico, y todas estas transmisiones elevan el consumo de energía. Por otra parte, los distintos valores de movilidad aplicados en la simulación, no introducen demasiada carga de tráfico, por lo que la movilidad no incrementa el consumo energético. Sin embargo, es de esperar que los nodos al desplazarse con mayor velocidad consuman más energía; pero las simulaciones arrojan resultados contrarios y muestran que la movilidad no influye tan significativamente sobre el consumo energético.

Con los experimentos simulados, se ha visto que tanto para la sobrecarga de paquetes de control como para el consumo energético, el número de peticiones de servicios juega un papel más decisivo que la propia movilidad.

5.4. Fiabilidad del protocolo

Hasta ahora hemos evaluado el ancho de banda y el gasto energético, a continuación mediremos una serie de parámetros que nos darán cuenta del nivel de fiabilidad de LIFT.

La fiabilidad es la proporción de actividades esperadas y recibidas desde un enlace. La fiabilidad es la tasa del número de nodos que reciben la transmisión del origen, del total de nodos en la red. Si la proporción es alta, la línea es fiable.

La fiabilidad se calcula dividiendo el número total de mensajes recibidos en sus respectivos destinos entre el número total de mensajes generados. Esta métrica mide el éxito general del algoritmo para proporcionar una comunicación fiable [105] [154].

No obstante, existen problemas relacionados con la medición de la fiabilidad debido al rango limitado de transmisión inalámbrica, la naturaleza de la difusión de los medios inalámbricos, la pérdida de paquetes a causa de la movilidad y errores en la transmisión de datos.

Para conocer la fiabilidad de LIFT y la efectividad en su conjunto, emplearemos la *tasa de entrega de paquetes* principalmente, acompañada de otra serie de métricas como el throughput o velocidad de transmisión de datos, el promedio de saltos para alcanzar el nodo destino, la carga de encaminamiento normalizada y el tiempo de adquisición de servicios.

5.4.1. Tasa de entrega de paquetes

La *tasa de entrega de paquetes* (**Packet Delivery Ratio**, **PDR**) representa el valor porcentual de paquetes exitosamente recibidos por el nodo destino. El porcentaje de paquetes entregados se calcula dividiendo el número de paquetes recibidos entre el número de paquetes generados.

En las simulaciones realizadas para analizar el rendimiento de LIFT respecto al *porcentaje de paquetes entregados vs. número de nodos* con distinta movilidad, hemos partido del escenario descrito en la tabla 4.2. En esa tabla se puede ver que el número de nodos varía entre 100 y 500, el número de peticiones se ajusta a una media de 300 requests, la velocidad de desplazamiento varía entre 2 y 20 Km/h., todo ello tiene el objetivo de ver en qué medida impactan estos parámetros al PDR.

En el gráfico 5.5 trazamos el promedio de la tasa de entrega de paquetes en función del número de nodos y la movilidad. Los resultados muestran que LIFT tiene un PDR de 92,72 % y de 87,12 % en el mejor y peor de los casos, respectivamente.

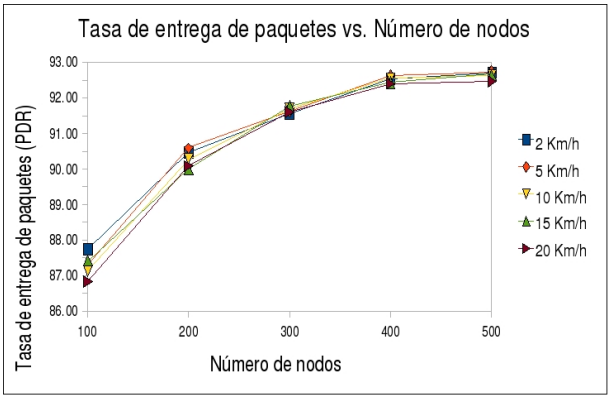


Figura 5.5: Tasa de entrega de paquetes (PDR) en función del número de nodos y la movilidad. Experimento realizado con 300 peticiones de servicios.

Se observa claramente en el gráfico 5.5 la influencia que la movilidad presenta sobre la tasa de entrega de paquetes. Conforme aumenta el número de nodos y aumentan los valores de la movilidad, también el valor de la tasa de entrega de paquetes se ve afectado significativamente. Aunque las curvas se solapan en algunos puntos y aumentan progresivamente de forma lineal hasta conseguir el valor máximo, la poca diferencia entre las curvas nos muestran que conforme se incrementa la movilidad y el número de nodos, desacelerando a su vez el crecimiento de las curvas, el PDR se ve afectado. La explicación es la siguiente, cuando la velocidad de los nodos es más rápida, existe mayor posibilidad de que los enlaces puedan romperse, ocasionando un mantenimiento de rutas más frecuente.

Por el contrario, este hecho nos indica que a mayor número de nodos y menor velocidad, mayor tasa de entrega de paquetes, lo cual es deseable. Fácilmente se deduce que el comportamiento de los nodos hace que aumente o disminuya el PDR. Por lo que la movilidad sí es un factor determinante para el PDR.

Lo anterior se puede explicar debido a dos motivos: Primero, que la baja movilidad ocasiona que se conserven los enlaces entre los nodos, mientras que la alta movilidad provoca más roturas de enlaces; por consiguiente, en el caso de baja movilidad, las tablas de rutas del protocolo de encaminamiento conser-

van información fresca del vecindario, ya que no existe necesidad de recalcular frecuentemente las rutas. Y segundo, que los protocolos reactivos[2] tienen mejor capacidad para reaccionar ante la movilidad, ya que ante cualquier cambio en la red, recalculan las rutas para establecer nuevos enlaces.

5.4.2. Promedio de saltos

El promedio de saltos (**Average Hop Count**) es la media de la longitud de la ruta que un paquete toma para llegar desde el nodo origen, hasta el nodo destino. Cuando un nodo genera un mensaje de descubrimiento de servicios, éste ha de pasar por muchos nodos intermedios para alcanzar el nodo que contiene el servicio solicitado. Con esta métrica conoceremos la media de saltos que un protocolo emplea para descubrir servicios. El promedio de saltos generalmente es medido en número de saltos.

En el gráfico 5.6 se presentan los resultados de la evaluación realizada a LIFT para conocer el promedio de saltos. Esta medida tiene como objetivo conocer el número de saltos que sigue el protocolo para descubrir servicios y conocer al mismo tiempo si emplea una ruta óptima para alcanzar el nodo destino, ya que cuanto menor sea el número de saltos menor latencia. Y en este caso en particular, nuestra intención es comprobar si el comportamiento de los nodos afecta de alguna manera al promedio de saltos. Por consiguiente, en esta medición cuantificamos el *promedio de saltos* en función del *número de nodos* con distinta movilidad. Al igual que en la evaluación anterior, para realizar este experimento se ha utilizado el escenario descrito en la tabla 4.2, llevándose a cabo con 300 peticiones de servicios.

El estudio produce los siguientes resultados: LIFT descubre servicios en 25 saltos en el peor caso y en el mejor caso en 10 saltos. Se observa en el gráfico 5.6 que las curvas describen una trayectoria completamente parabólica y descendente, y en algunos puntos de la pendiente, las curvas se solapan. Es

[2]LIFT tiene como protocolo de encaminamiento subyacente a AODV, que es un protocolo de encaminamiento reactivo.

muy notorio que el número de saltos disminuye conforme aumenta el número de nodos y cuando disminuye ligeramente la velocidad de desplazamiento de los nodos.

Podemos afirmar, que el promedio de saltos se ve muy poco influenciado por la movilidad. Es muy ligera la diferencia entre las curvas, ya que siguen una tendencia similar; sin embargo, esta diferencia es en promedio de al menos un número.

Por todo ello se concluye, que a mayor número de nodos, menor número de saltos, lo cual es lo óptimo ideal. Esto se explica debido a que cuanto más número de nodos haya, más número de HCD proporcionan soporte para el descubrimiento de servicios y por tanto, más número de servidores disponibles en los clusters; este hecho provoca que los servicios se descubran en las cercanías, evitando con esto hacer más saltos para descubrir servicios.

Mientras que a menor cantidad de nodos, menor número de HCD que soporten la búsqueda de los servicios solicitados, de modo que se han de realizar más saltos hasta descubrir el servicio solicitado.

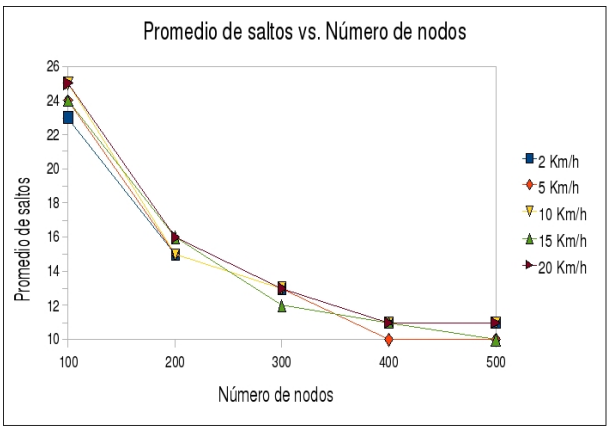


Figura 5.6: Promedio de número de saltos en función del número de nodos y la movilidad. El número de peticiones de servicios se ajusta a un valor medio de 300 requests.

5.4.3. Throughput

El *throughput* mide la velocidad de transmisión de los datos, esta métrica nos proporciona la media de la tasa de bits que llegan al nodo destino y representa la velocidad máxima de enlace. El throughput es una métrica de efectividad del protocolo.

Hemos decidido emplear esta medición con la intención de comprobar si la movilidad representa alguna influencia sobre la velocidad de transmisión de los datos y ver en qué medida el comportamiento de los nodos afecta al rendimiento general del protocolo propuesto.

En las simulaciones realizadas para analizar el *throughput* en función del *número de nodos* con distinta movilidad, partimos del escenario descrito en la tabla 4.2, ajustándonos a una gama de valores de 100, 200, 300, 400 y 500 nodos, enviándose 300 peticiones de servicios y una velocidad de desplazamiento en un rango de 2, 5, 10, 15 y 20 Km/h.

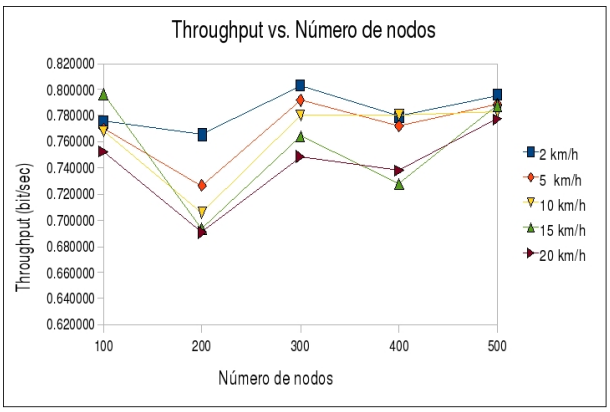


Figura 5.7: Throughput del protocolo en función del número de nodos y la movilidad. Experimento realizado con 300 peticiones de servicios.

El estudio realizado señala que LIFT tiene una media de throughput de 0.743286 bits/segs. El gráfico 5.7 muestra que las curvas tienen considerables diferencias entre sí y se aprecia que las curvas no presentan una evolución ascendente como en las otras mediciones realizadas a LIFT. Se observa que el throughput sí se ve influenciado por el comportamiento de los nodos. Conforme aumenta

la velocidad de desplazamiento de los nodos disminuye la tasa de throughput. En cada experimento con distinto número de nodos se observa el mismo efecto respecto a las diferentes variables de movilidad. Es decir, a mayor movilidad menor tasa de throughput.

La razón por la que disminuye la velocidad de transmisión de datos cuando existe mayor movilidad, se debe sencillamente a que existe mayor tráfico de mensajes de control, debido probablemente a la rotura de los enlaces entre los nodos, por tanto, la actualización de las tablas de rutas generan mayor tráfico de control y mayor congestión en la red. De ahí que el protocolo genere mayor tráfico de señalización que ocasiona que se sature el ancho de banda y disminuya la velocidad de transmisión.

En tanto que una menor movilidad provoca que el throughput aumente y en consecuencia, se permita más espacio para el tráfico útil gracias a una menor congestión en la red.

Todo parece indicar que la velocidad de los nodos es un factor que influye en la tasa de throughput. En cambio el número de nodos no influye significativamente en la velocidad de transmisión de los datos.

5.4.4. Carga de encaminamiento normalizada

La carga de encaminamiento normalizada (**Normalized Routing Load**, NRL) se utiliza para estimar el número de mensajes de encaminamiento que emplea el protocolo. Equivale a la cantidad de paquetes de control que deben ser generados respecto a la cantidad de paquetes de datos recibidos.

La carga de encaminamiento normalizada expresa la tasa de bytes transmitidos, para asegurar que un byte de datos se entrega con éxito en el destino. Para el cómputo de esta métrica se tiene en cuenta cada retransmisión de un paquete de control, considerándose como un nuevo paquete de control.

Conforme esta medida se aproxima a la unidad, mejor es la carga introducida, puesto que para entregar un paquete de datos sería preciso transmitir dicho paquete y no serían necesarios los mensajes de control. Lo anterior implica una

óptima eficiencia en la entrega de paquetes de datos. Una cualidad deseable en un protocolo, es que genere poco tráfico durante el proceso de encaminamiento y así mantener más espacio disponible para el tráfico de paquetes de datos.

En el gráfico 5.8 se muestran los resultados de la medición de la *NRL* en función del *número de nodos* con distinta movilidad, con el fin de ver en qué medida la NRL se ve influenciada por la movilidad. Al igual que en los otros estudios, hemos utilizado el escenario descrito en la tabla 4.2, realizado con un valor de peticiones de servicios de 300 requests.

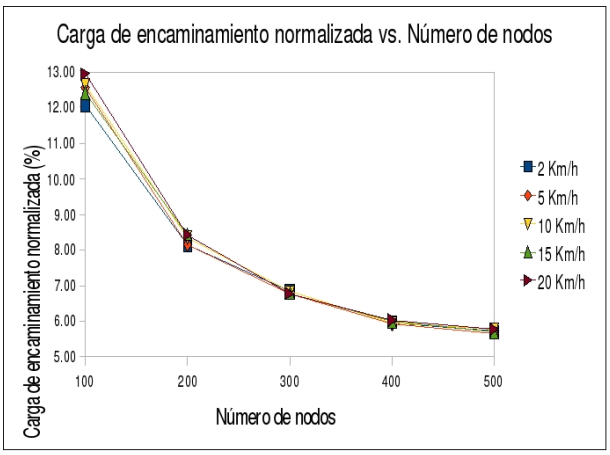


Figura 5.8: Carga de encaminamiento normalizada en función del número de nodos y la movilidad. El número de peticiones de servicios es de 300 requests.

Se observa en el gráfico 5.8 que inicialmente los valores de NRL son muy elevados y van disminuyendo exponencialmente hasta alcanzar los valores más inferiores cercanos a 1. Se aprecia que las curvas presentan un comportamiento lineal y descendente; al principio de la pendiente las curvas presentan diferencias mínimas y conforme van descendiendo comienzan a solaparse entre sí. Cuando las curvas llegan a un punto del eje x que representa el valor de 300 nodos, se encuentran completamente traslapadas y así continúan descendiendo traslapadamente sobre el mismo eje, hasta alcanzar el valor final de 500 nodos. Nos damos cuenta claramente que conforme aumenta el número de nodos, disminuye el valor de la NRL. Además, es menos significativa la movilidad.

Gráficamente puede verse que la movilidad introduce muy poca carga de encaminamiento normalizada, puesto que las tendencias de las curvas indican que no existen diferencias entre sí. Por otro lado, lo que sí podemos deducir es que a mayor número de nodos, menor NRL.

La explicación de que la velocidad de movimiento de los nodos introduce poco valor de NRL, se debe principalmente al comportamiento del propio protocolo de encaminamiento subyacente en el que se cimenta LIFT. Otra razón para suponer que un mayor número de nodos introduce poca NRL, es que el número de peticiones de servicios (300 requests) es el mismo para cada valor del número de nodos.

Parece claro que en un elevado número de nodos existe un mayor número de clusters y por tanto más proveedores de servicios, por un lado evitando enviar gran número de paquetes de encaminamiento y por otro, que los servicios se encuentren en las proximidades de los nodos que envían las peticiones.

En resumen, no se observan cambios en la carga de encaminamiento normalizada en función de la movilidad, sobretodo cuando la movilidad tiende a decrecer y cada vez que aumenta el número de nodos. En realidad la movilidad influye muy poco en la rotura de los enlaces de las rutas para que sea necesario recalcular dichas rutas, lo que supone que las rutas se mantienen frescas a pesar de la movilidad. Esto se debe, como ya habíamos mencionado, a las características del protocolo reactivo sobre el que se basa LIFT.

De modo que, a menor carga de encaminamiento normalizada, mayor disponibilidad para ocupar el medio para otro tipo de tráfico.

5.4.5. Tiempo de adquisición de servicio

En esta sección cuantificamos el *tiempo de adquisición de servicios*. Esta medida nos es útil para conocer los tiempos que emplea LIFT para descubrir servicios. Esta métrica nos permite medir el tiempo transcurrido desde el envío de la petición de servicios hasta recibir la respuesta de petición.

El tiempo de adquisición de servicio (**Service Acquisition Time**) mide el tiempo que transcurre desde el envío del primer *request* hasta la recepción del primer *reply* válido. Es decir, el tiempo que toma una respuesta de servicio para volver al origen después de que se ha enviado una petición de servicio.

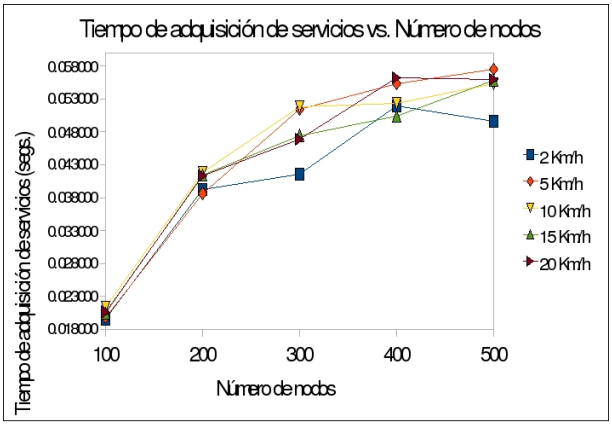


Figura 5.9: Tiempo de adquisición de servicios en función del número de nodos y la movilidad. Experimento realizado con 300 peticiones de servicios.

En el gráfico 5.9 se observan los resultados de la evaluación realizada a LIFT. Se mide el *tiempo de adquisición de servicio* en función del *número de nodos* con distinta movilidad. En este experimento se han utilizado los valores descritos en la tabla 4.2, llevándose a cabo con 300 peticiones de servicios. Lo anterior tiene como objetivo comprobar si el número de nodos y el comportamiento de los nodos tiene influencia sobre el tiempo de adquisición de servicios.

Considerando todos los experimentos, el estudio computa una media de 0.043363 segundos de tiempo para adquirir un servicio. En general se observa que las curvas presentan un comportamiento logarítmico, ascienden pronunciadamente conforme se incrementa el número de nodos. Las curvas en algunos puntos se solapan entre sí. Sin embargo, existe diferencia entre ellas, por tanto podemos afirmar que la movilidad refleja un ligero efecto sobre el tiempo de adquisición de servicios.

Se observa que a mayor número de nodos y mayor movilidad, aumenta el tiempo de adquisición de servicios. Sin embargo, se espera un efecto contrario mientras decrece el número de nodos y disminuye la velocidad.

Podemos concluir que a menor número de nodos y menor valor de movilidad, menor es el tiempo de adquisición de servicios, lo cual es deseable.

La razón que explica lo anterior, se debe a que cuando existe menor movilidad y menos dispositivos en la red, se conservan por más tiempo los enlaces de las rutas, existe menor tráfico de señalización, menor sobrecarga de paquetes de control y es muy probable que un servicio sea encontrado en menor tiempo.

En el *tiempo de adquisición de servicios* los valores inferiores son lo óptimo ideal, ya que se cuantifica un tiempo de respuesta.

Una vez analizado el tiempo de adquisición de servicios, podemos darnos cuenta que aunque se obtienen altas tasas de entrega de paquetes y por tanto, altas tasas de descubrimiento de servicios, se sacrifica un poco el tiempo de adquisición de servicios. Por un lado LIFT es más eficaz y por el otro es menos eficiente.

5.4.6. Conclusiones

En este apartado resumimos brevemente las principales conclusiones del comportamiento de LIFT en relación a la fiabilidad.

Dado que hemos evaluado el número de nodos y la movilidad como elementos claves en el comportamiento de nuestro protocolo, veremos que en algunos casos la movilidad o ambos resultan ser factores determinantes en las mediciones.

En cuanto a la *tasa de entrega de paquetes* (PDR) se refiere, ésta es considerablemente influenciada por el número y el comportamiento de los nodos. Conforme aumenta la velocidad de desplazamiento de los nodos y aumenta la cantidad de nodos, se ve afectado significativamente el valor de la tasa de entrega de paquetes. Por lo tanto, la combinación de número de nodos con movilidad, son un factor determinante para el PDR. Esto al menos es positivo, porque si la MANET tiene muchos nodos la tasa de éxitos es alta.

Respecto al *promedio de saltos*, este parámetro se ve muy poco influenciado por la movilidad. Se puede decir que el número de saltos disminuye conforme aumenta el número de nodos y disminuye la velocidad de la movilidad. En cambio, el número de nodos sí afecta en mayor proporción a la media de número de saltos, ya que a mayor número de nodos, menor número de saltos. Esto también es ideal cuando tenemos una MANET con gran cantidad de nodos y movilidad media/alta.

En cambio, cuando se trata del *throughput*, la movilidad juega un papel más importante que el número de nodos. Mayores valores de velocidad implican menor tasa de *throughput*; mientras que el número de nodos casi no refleja variaciones importantes para la tasa de throughput. Por consiguiente, en esta métrica la velocidad de los nodos es un factor decisivo en la tasa de throughput.

En la métrica de *carga de encaminamiento normalizada*, la movilidad no ejerce demasiada influencia sobre la NRL. Se ha visto que conforme aumenta el número de nodos, disminuye el valor de la NRL. Los resultados de los estudios muestran que la movilidad y el número de nodos introducen muy poca carga de encaminamiento normalizada. Estos resultados son de atractivo interés si tenemos una MANET con gran cantidad de nodos y alta movilidad.

Respecto al *tiempo de adquisición de servicios*, se puede decir que la movilidad presenta una ligera influencia sobre este parámetro, ya que a medida que aumenta el número de nodos y la velocidad de la movilidad, entonces aumenta el tiempo de adquisición de servicios. Por el contrario, se concluye que cuando disminuye el número de nodos y la velocidad, entonces disminuye también el tiempo de adquisición de servicios.

Como se había supuesto al principio, el número de nodos y la movilidad resultan ser los parámetros que más influencia ejercen sobre los aspectos evaluados en la sección 5.4 y que además, sean los factores más determinantes en el rendimiento general de LIFT. El motivo que explica esto se debe principalmente a que cuando los nodos tienden a moverse con mayor velocidad y estos además son demasiados, entonces los enlaces existentes se rompen con mayor frecuencia

y la información de las tablas de servicios deja de ser actual, por tanto, entra en marcha más a menudo el mantenimiento de rutas y el proceso de descubrimiento de servicios. Esto trae consigo mayor cantidad de intercambio de paquetes de control, mayor tráfico en la red y mayores tiempos de latencias.

En general LIFT satisface con éxito las peticiones de servicios cuando existe un elevado número de nodos y movilidad media, obtiene un PDR aceptable; sin embargo, se ha visto que ofrece peores tiempos de respuestas cuando tiene escenarios con alta movilidad. Por consiguiente, descubre servicios con más eficacia pero con menor eficiencia.

5.5. Escalabilidad

En términos generales, la escalabilidad es la capacidad de una aplicación informática, sistema o red para continuar funcionando adecuadamente después de aumentar su tamaño, sin mermar la calidad de los servicios. Esto es, que el sistema sea capaz de cambiar su volumen o configuración para adaptarse a las circunstancias cambiantes o a las necesidades de los usuarios conservando el nivel de calidad de los servicios.

La escalabilidad es una característica que le confiere a las MANET la posibilidad de soportar cientos e incluso miles de usuarios móviles. Para conseguir un nivel razonable de escalabilidad se sugiere establecer mecanismos que fraccionen la MANET en grupos, este proceso es conocido como clustering y tiene como fin mantener una topología relativamente estable.

En las MANET con elevado número de dispositivos y con excesivo consumo de ancho de banda, se reduce considerablemente el rendimiento general del protocolo empleado. Una de las soluciones al problema antes descrito, es introducir clusters con el fin de segmentar el tráfico y establecer líderes de clusters que operen una mayor carga de trabajo.

Se espera que LIFT alcance niveles satisfactorios de escalabilidad, ya que es un SDP que opera con clusters. Bajo este esquema se tiene conocimiento del vecindario y por tanto, es posible descubrir servicios con mayor eficacia dentro

de una porción del total de número de nodos, gracias a una disminución del tráfico global.

Uno de los principales obstáculos para lograr escalabilidad en las MANET es la abundante cantidad de sobrecarga de control causada por topologías de gran dimensión, alta movilidad y al frecuente mantenimiento de las rutas [160]. Cuando tenemos gran número de nodos, alta movilidad y gran densidad en un área específica, es recomendable una solución como la nuestra, ya que es un protocolo de descubrimiento de servicios con diseño Cross-layer basado a su vez en un protocolo de encaminamiento reactivo, este tipo de protocolos teóricamente tienen mejor rendimiento respecto a la sobrecarga de paquetes de control que los protocolos proactivos [136].

En los experimentos para evaluar el rendimiento de LIFT (apartado 5.4), se ha simulado con un número máximo de 500 nodos, hasta ese nivel se ha visto que LIFT tiene razonables tiempos de respuesta.

Para probar la escalabilidad de LIFT hemos preparado experimentos con 500, 750, 1000, 1250 y 1500 nodos extendidos sobre 9, 10, 11, 12 y 13 clusters, el resto de parámetros en particular se describen en la tabla 5.2. Consideramos que para probar la escalabilidad de LIFT es suficiente con 1500 nodos, ya que para estimar la escalabilidad con un número mayor de nodos, se podrían extrapolar los siguientes valores en función del comportamiento conocido hasta 1500 nodos.

En este experimento se ha ajustado el nivel de redundancia de servicios a los siguientes valores: 5, 10, 15 y 20. La razón de elegir esos valores se debe a que deseamos comprobar si el nivel de redundancia de servicios influye sobre el tiempo de adquisición de servicios, por lo que con esos cuatro valores es suficiente para notar la diferencia.

En el gráfico 5.10 se muestran los resultados de dichas mediciones. Podemos apreciar gráficamente en las curvas un comportamiento logarítmico y sobre todo las diferencias entre las curvas que siguen una trayectoria ascendente y progresiva. Conforme aumenta el número de nodos, aumenta el tiempo de adquisición de servicios, en cambio, la diferencia entre las curvas se mantiene casi constan-

Parámetro	**Rango de valores**
Número de dispositivos	500, 750, 1000, 1250, 1500
Número de dispositivos de tipo HCD	63, 80, 100, 120, 156
Número de dispositivos de tipo LCD	437, 670, 900, 1130, 1344
Número de clusters	9, 10, 11, 12, 13
Número de dispositivos en cada cluster	56, 75, 90, 105, 116
Número de CL en cada cluster	7, 8, 9, 10, 12
Número de tipos de servicios	15
Número de redundancia de servicios	5, 10, 15, 20
Número total de proveedores de servicios	126, 160, 200, 240, 312
Número de proveedores de servicios tipo HCD	63, 80, 100, 120, 156
Número de proveedores de servicios tipo LCD	63, 80, 100, 120, 156
Tiempo de simulación	600 segundos

Tabla 5.2: Parámetros empleados para comprobar la escalabilidad

te, lo cual indica que el elevado número de nodos y el nivel de redundancia de servicios influyen sobre el tiempo requerido para descubrir un servicio.

En cuanto al nivel de redundancia de servicios, la curva de 20 réplicas se mantiene por debajo de las otras curvas, como es de suponer, a mayor número de réplicas de servicios disponibles, menor tiempo para adquirir un servicio. Con este resultado se puede apreciar la influencia de la redundancia de servicios sobre el tiempo de adquisición de servicios.

En general el tiempo de adquisición de servicios en 500, 750, 1000, 1250 y 1500 nodos es considerablemente elevado. Este comportamiento hasta cierto punto es normal, ya que a mayor número de nodos, mayor es el tiempo de adquisición de servicios. Por consiguiente, es posible deducir que de simularse con 2000 nodos, 3000, etc. los tiempos de adquisición de servicios irremediablemente aumentarán, haciendo intolerables dichos tiempos.

Por otra parte, un elevado número de nodos además de incrementar el tiempo de adquisición de servicios, también introduce mayor sobrecarga de paquetes de control, como consecuencia del continuo mantenimiento de las rutas, consumiendo el reducido ancho de banda disponible y por tanto afectando al throughput.

Al parecer LIFT observa un incremento normal en el tiempo de adquisición de servicios al momento de incrementar la cantidad de nodos, pero gracias al agrupamiento puede manejar este fenómeno y puede delimitar el tráfico a pequeñas áreas.

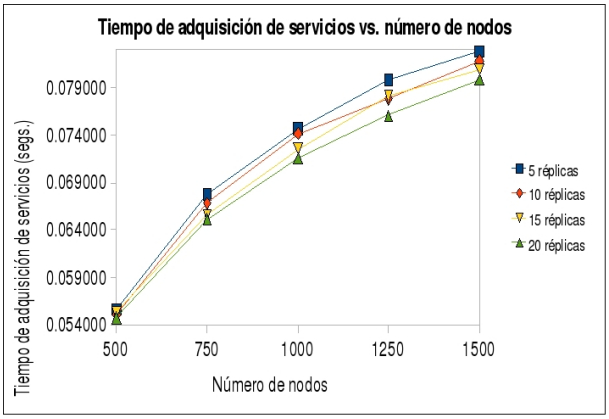


Figura 5.10: Tiempo de adquisición de servicios vs. número de nodos. La velocidad constante de movilidad es de 10 Km/h., niveles de redundancia de 5, 10, 15 y 20 réplicas de servicios y un total de 300 peticiones de servicios.

5.6. Comparativa con AODV-SD

En este apartado abordaremos el estudio comparativo del rendimiento de LIFT con AODV-SD. Las simulaciones se llevan a cabo sólo en los casos cuando sea aplicable realizar la comparación, puesto que LIFT es un protocolo basado en diseño de clustering que funciona agrupando nodos y estableciendo líderes de grupos, mientras que AODV-SD no es un protocolo que gestione grupos.

El motivo de comparar dos protocolos hasta cierto punto diferentes, obedece a la siguiente razón. Comprobar las diferencias de rendimiento de un protocolo basado en clusters y liderado por dispositivos con capacidades superiores (LIFT), con un protocolo que no los agrupa y que considera que todos los dispositivos son homogéneos (AODV-SD). Esta idea se basa en el hecho de que LIFT, al agrupar los nodos de la red supone una optimización de recursos. Independientemente si

previamente se hayan comparado estos dos esquemas de diseño, nuestro interés por esta comparativa es conocer particularmente el nivel de rendimiento de ambos protocolos.

Debido a la desemejanza de escenarios de aplicación entre LIFT y AODV-SD, se esperan diferentes comportamientos en los resultados de las simulaciones. Por esta razón, es conveniente que se simule bajo las mismas condiciones de topología, tiempos, dimensiones de los escenarios y movilidad. No obstante, existen parámetros que son únicos en LIFT y que no se pueden aplicar a AODV-SD; por ejemplo, el tema de los dispositivos de capacidades superiores, la formación de clusters y la figura de líderes de clusters.

Hemos optado por comparar los mismos parámetros de la sección 5.3 y de la sección 5.4 para ambos protocolos. A continuación explicamos la serie de parámetros usados en las simulaciones.

El número total de dispositivos colocados en los escenarios caen en un rango de 50, 100, 150 y 200 nodos en ambos protocolos[3], en el caso de LIFT compuestos de la siguiente manera: 9, 12, 15 y 24 de tipo HCD y 41, 88, 135 y 176 de tipo LCD, en AODV-SD no hay distinción de tipo de nodos. En LIFT se forman 3, 4, 5 y 6 clusters, en AODV-SD no se usan clusters. El número de nodos en cada cluster es de 17, 25, 30 y 33 nodos. En LIFT hay 3 y 4 líderes por cluster, en AODV-SD no se utilizan líderes de clusters. Para ambos protocolos existen disponibles 15 tipos diferentes de servicios, ofrecidos por 18, 24, 30 y 48 nodos proveedores de servicios, con réplicas de los 15 tipos de servicios en un orden de redundancia de 4, 8, 12 y 16 veces el mismo servicio. Para balancear y hacer más justa la distribución de proveedores de servicios, en LIFT tenemos 9, 12, 15 y 24 nodos de tipo HCD y la misma proporción de nodos de tipo LCD. La duración de la simulación es de 600 segundos. El área rectangular en la que se realiza la simulación es de 750 x 750 mts. LIFT por su parte tiene 3, 4, 5 y 6 subáreas de simulación, cada área de 200 x 200 mts. El modelo de movilidad utilizado en LIFT es CRWP, porque este modelo tiene en cuenta la figura de un

[3]Los experimentos se han simulado con este rango de valores, en virtud de que AODV-SD no soportaba valores superiores a 200 nodos.

líder de cluster y es el modelo de movilidad que usamos en nuestra propuesta, mientras que en AODV-SD se utiliza el modelo tradicional Random WayPoint. El protocolo de encaminamiento para redes Ad Hoc en ambos protocolos es AODV y el protocolo MAC es el IEEE 802.11b. Los dos protocolos simulados tienen un rango de transmisión de 100 mts., con una velocidad media de 10 Km/h. No existen pausas. Y por último, la energía inicial que tiene cada nodo es de 20 Julios.

La tabla 5.3 resume los parámetros descritos en el párrafo anterior. Hasta donde sea posible pretendemos encontrar una equivalencia para que los resultados no sean sesgados. Muchos de estos valores son de aplicación general, tanto para LIFT como para AODV-SD. Algunos de estos valores son constantes y la gran mayoría son rangos de valores para variar los resultados.

La primer métrica que comparamos es la *tasa de descubrimiento de servicios*; por ser una métrica externa nos es fácil percibir la eficacia de ambos protocolos, de esta manera cuantificamos el porcentaje de servicios descubiertos respecto a las peticiones de servicios enviadas y se comprueba el éxito de un SDP para descubrir servicios.

El gráfico 5.11 detalla los resultados de esta métrica. Como podemos ver en el gráfico, LIFT presenta valores superiores a AODV-SD, no es muy amplia la diferencia entre las curvas, la curva de LIFT tiene una trayectoria logarítmica y la curva de AODV-SD tiene una trayectoria lineal y pronunciada. Sin embargo, la curva perteneciente a LIFT tiene en su punto más inferior un valor de 80,33 % mientras que en AODV-SD su valor mínimo es de 63,00 %; por otra parte, en la cúspide de las curvas la diferencia es mínima, pero en todo momento, LIFT lleva la delantera en este sentido. Por ello, podemos decir que LIFT tiene mayor éxito para descubrir servicios que AODV-SD. La razón de que LIFT tenga más éxito para descubrir servicios es gracias a los beneficios del esquema de clustering, ya que optimiza el proceso de encaminamiento y de descubrimiento de servicios.

En el gráfico 5.12 representamos la métrica *sobrecarga de paquetes de control* en función del *número de nodos*. Esta métrica interna la utilizamos para cuan-

Parámetros	**Rangos para LIFT**	**Rangos para AODV-SD**
No. de nodos	50, 100, 150, 200	50, 100, 150, 200
No. de nodos de tipo HCD	9, 12, 15, 24	No aplica
No. de nodos de tipo LCD	41, 88, 135, 176	No aplica
No. de clusters	3, 4, 5, 6	No aplica
No. de nodos en cada cluster	17, 25, 30, 33	No aplica
No. de CL en cada cluster	3, 4	No aplica
No. de tipos de servicios	15	15
No. de redundancia de servicios	4, 8, 12, 16	4, 8, 12, 16
No. de proveedores de servicios	18, 24, 30, 48	18, 24, 30, 48
No. de proveedores de tipo HCD	9, 12, 15, 24	No aplica
No. de proveedores de tipo LCD	9, 12, 15, 24	No aplica
Tiempo de simulación	600 segundos	600 segundos
Área total de simulación	750 x 750 mts.	750 x 750 mts.
No. de subáreas de simulación	3, 4, 5, 6	No aplica
Subáreas de simulación	200 x 200 mts.	No aplica
Modelo de movilidad	CRWP	RWP
Protocolo Ad Hoc	AODV	AODV
Protocolo MAC	IEEE 802.11b	IEEE 802.11b
Rango de transmisión	100 mts.	100 mts.
Velocidad	10 (Km/h.)	10 (Km/h.)
Pausas	0 segs.	0 segs.
Energía inicial	20 Julios	20 Julios

Tabla 5.3: Parámetros para la simulación de LIFT y AODV-SD

tificar el ancho de banda consumido, y por tanto, para conocer en qué medida cada protocolo minimiza este recurso. Se puede observar en dicho gráfico, que ambas curvas presentan una trayectoria ascendente, progresiva y pronunciada, no existe diferencia significativa, la diferencia entre sí es ligera y mínima; sin embargo, la curva que se corresponde con AODV-SD se encuentra por encima de la curva de LIFT, excepto cuando el número de nodos es igual a 50, que es cuando la curva de LIFT se encuentra ligeramente por encima de la curva de AODV-SD. Se puede deducir que LIFT envía menos mensajes de control que AODV-SD independientemente de la cantidad de nodos presentes en el escenario

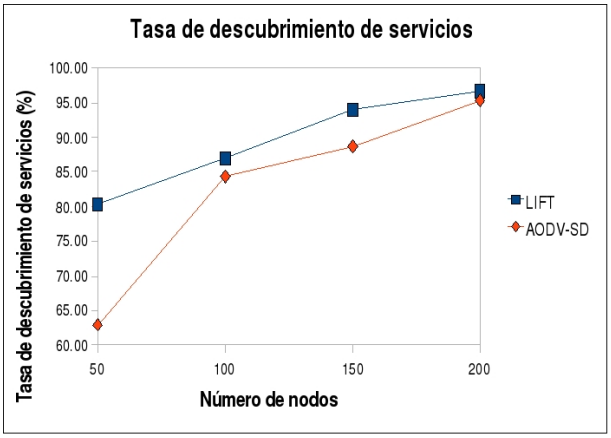


Figura 5.11: Tasa de descubrimiento de servicios de LIFT y AODV-SD

y a pesar del envío de mensajes de control para mantener los clusters. Con estos resultados se puede decir que LIFT introduce menor sobrecarga de paquetes de control, de este modo, AODV-SD ocupa ligeramente mayor ancho de banda que LIFT en este respecto.

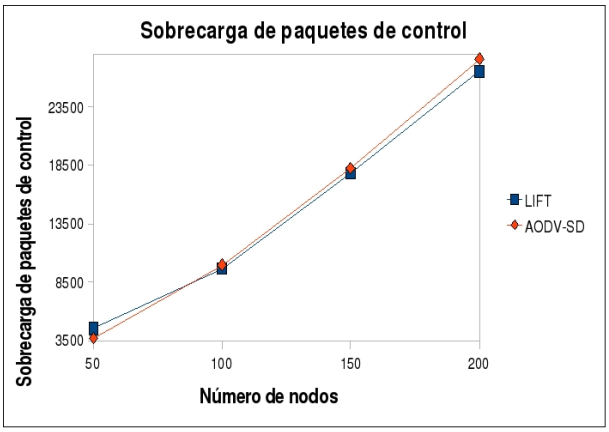


Figura 5.12: Sobrecarga de paquetes de control de LIFT y AODV-SD

Se muestra en el gráfico 5.13 los detalles de la medición correspondiente a *consumo energético* vs. *número de nodos*. Utilizamos esta métrica para cuantificar el consumo de energía y de esta forma, conocer cuál de los dos protocolos optimiza mejor este recurso. Se puede apreciar en este gráfico que las curvas

ascienden de forma lineal, pero existe una clara diferencia entre ellas. La curva correspondiente a LIFT asciende por debajo de la curva de AODV-SD, lo que indica que LIFT consume menos energía que AODV-SD, salvo en el caso cuando el número de nodos es menor, en el que la curva de LIFT asciende por encima de la curva de AODV-SD. El hecho de que LIFT tenga menor gasto energético que AODV-SD, es producto de la estrategia de agrupamiento de los nodos, dado que uno de los beneficios que se esperan al formar clusters, es la reducción del consumo de energía [157].

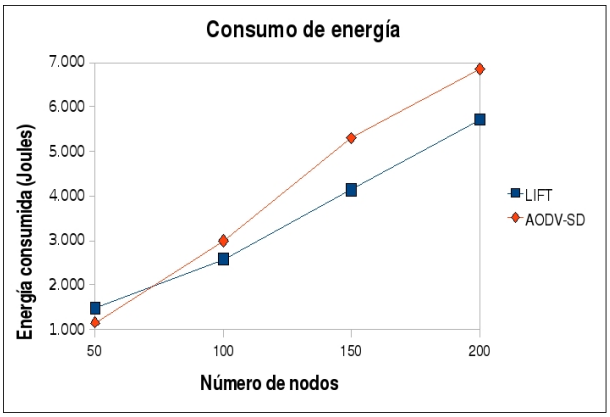


Figura 5.13: Consumo energético de LIFT y AODV-SD

En el gráfico 5.14 mostramos los resultados de la comparación de ambos protocolos en la métrica *tasa de entrega de paquetes* vs. *número de nodos*. En este gráfico podemos ver que las curvas presentan diferencias significativas. Si bien las dos curvas tienen tendencia a describir una trayectoria ascendente, la curva que se corresponde con AODV-SD es la que asciende linealmente, ya que comienza en un mínimo de 75,41 % como valor de PDR, mientras LIFT su valor mínimo se ubica por encima de 86 %, por lo que esta curva no asciende de forma similar a la curva de AODV-SD, ya que la primera no muestra efectos considerables respecto al incremento del número de nodos. Podemos concluir que LIFT tiene mayor PDR que AODV-SD.

Los resultados de la métrica *throughput* en función del *número de nodos* nos dan una idea de la velocidad de transmisión de datos de cada protocolo.

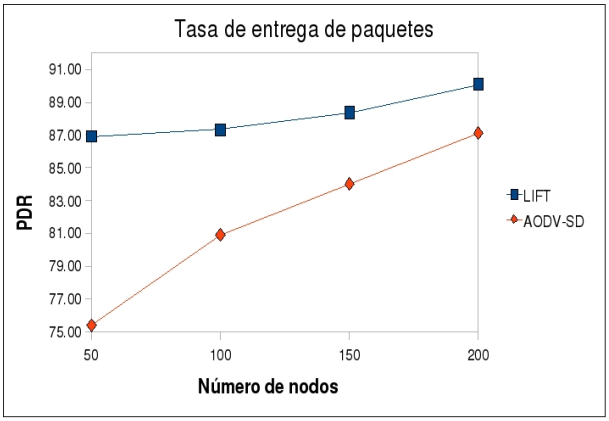


Figura 5.14: Tasa de entrega de paquetes de LIFT y AODV-SD

Estos resultados se dan a conocer en el gráfico 5.15. Se pueden apreciar diferencias significativas en las curvas, sobre todo en la curva que se corresponde con LIFT, puesto que desciende más pronunciadamente que la curva correspondiente a AODV-SD. La velocidad de transmisión de AODV-SD se mantiene casi constante respecto al número de nodos, mientras que la tasa de transmisión de datos de LIFT va descendiendo a medida que aumenta el número de nodos. A pesar de esto LIFT transmite datos a velocidades superiores a las que transmite AODV-SD. El aumento en la tasa de throughput es una de las ventajas que el diseño de clustering de LIFT tiene sobre la capa de acceso al medio, este tipo de diseño ayuda a incrementar la rapidez de la conexión y por tanto, a conseguir mayor tasa de transmisión de datos [37].

En el gráfico 5.16 aparecen los resultados de la medición *promedio de saltos* vs. *número de nodos*. Con esta métrica se computa la cantidad de saltos que emplea cada protocolo para alcanzar el nodo destino. Podemos comprobar gráficamente que las curvas presentan diferencias muy significativas y tienden a reducir el número de saltos conforme se incrementa el número de nodos. La curva correspondiente a AODV-SD desciende por encima de la curva de LIFT, lo que significa que LIFT emplea en promedio menos saltos para descubrir servicios que AODV-SD. Esto gracias a la abstracción de la topología de red (clustering) ya que las peticiones de servicios son atendidas por los HCD dentro del cluster

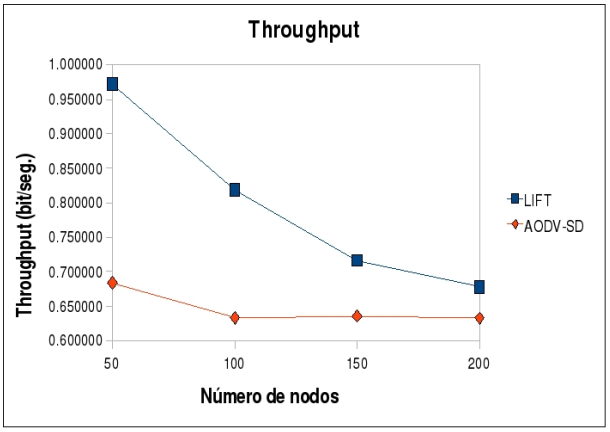


Figura 5.15: Throughput de LIFT y AODV-SD

donde ha sido lanzada la petición y el servicio es descubierto en una ruta más corta, mientras que en AODV-SD las peticiones son enviadas a toda la red, por lo que la media de saltos para descubrir servicios es mayor.

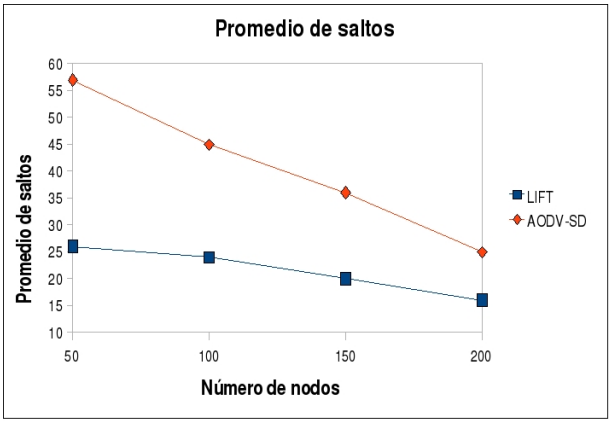


Figura 5.16: Promedio de saltos de LIFT y AODV-SD

El gráfico 5.17 traza los resultados de la métrica *carga de encaminamiento normalizada* en función del *número de nodos*. La métrica NRL nos dimensiona el número de mensajes de encaminamiento necesarios para descubrir servicios. Podemos apreciar en este gráfico que las curvas descienden regresivamente conforme aumenta el número de nodos, pero siempre la curva de LIFT se mantiene por debajo de la curva de AODV-SD. Con esto podemos concluir que LIFT

introduce menor carga de encaminamiento normalizada que la introducida por AODV-SD. Principalmente debido al esquema de clustering de LIFT, este tipo de diseño contribuye a mejorar el proceso de encaminamiento mediante la reducción del tamaño y la actualización de las tablas de encaminamiento inmediatamente después de ocurrir cambios en la topología [157].

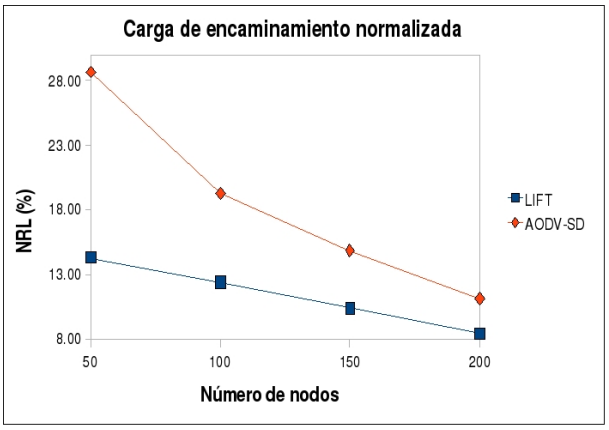


Figura 5.17: Carga de encaminamiento normalizada de LIFT y AODV-SD

El gráfico 5.18 detalla los resultados de la medida del parámetro *retardo extremo a extremo* en función del *número de nodos*. Con esta métrica podemos estimar la duración de un paquete desde el momento que parte del nodo origen hasta que llega al nodo destino. En el gráfico se puede observar que las curvas presentan diferencias notorias entre sí. La curva correspondiente a LIFT desciende paulatinamente conforme aumenta el número de nodos, mientras que la curva perteneciente a AODV-SD no tiene el mismo comportamiento, esta curva tiende a mantenerse horizontalmente, excepto en el tramo de 50 a 100 nodos, en donde sí influye el número de nodos sobre el retardo extremo a extremo, mientras que para el tramo de 100, 150 y 200 nodos, el número de nodos no influye sobre el retardo extremo a extremo, al parecer AODV-SD en este último tramo de la curva es insensible al número de nodos. Se puede decir que en esta métrica AODV-SD emplea menor retardo extremo a extremo que LIFT.

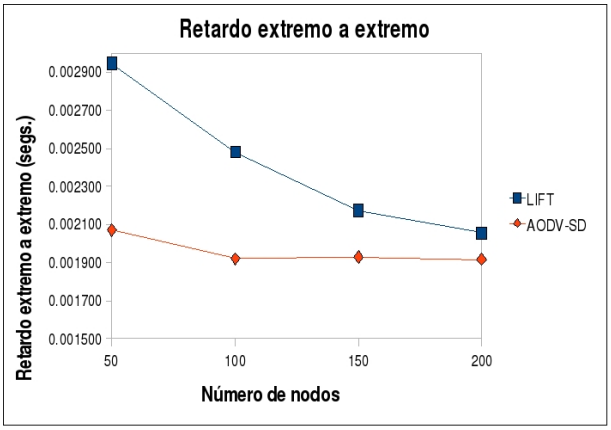


Figura 5.18: Retardo extremo a extremo de LIFT y AODV-SD

Por último, se muestra en el gráfico 5.19 las mediciones correspondientes al *tiempo de adquisición de servicios* vs. *número de nodos*. Esta métrica nos permite estimar cuál de los dos protocolos es más rápido para descubrir servicios. Se observa en el gráfico que las diferencias obtenidas en las curvas describen una trayectoria ascendente en zigzag que se cruzan en algunos puntos, es decir, de forma alternada en los cuatro puntos del eje x, en dos de ellos AODV-SD se localiza por encima de LIFT, mientras que en los dos puntos restantes LIFT se encuentra por encima de AODV-SD. Esto indica que para el tiempo de adquisición de servicios LIFT y AODV-SD obtienen casi los mismos valores.

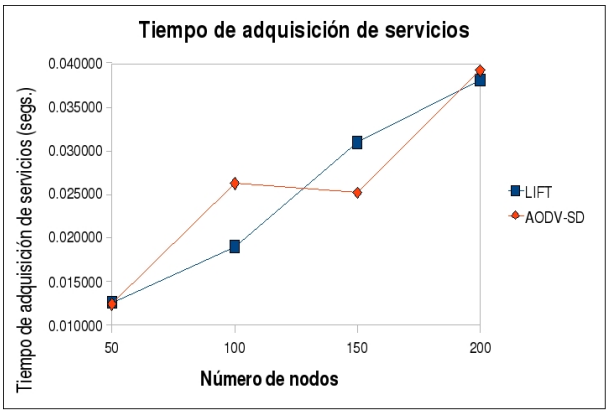


Figura 5.19: Tiempo de adquisición de servicios de LIFT y AODV-SD

5.6.1. Conclusiones

En este apartado resumiremos las principales conclusiones que resultan de la comparativa de LIFT con AODV-SD.

En cuanto a la *tasa de descubrimiento de servicios*, hemos visto que LIFT descubre servicios con una tasa superior al 95 %, de esta forma podemos decir que LIFT descubre con más éxito servicios que AODV-SD.

Respecto a la métrica *sobrecarga de paquetes de control*, también hemos visto que AODV-SD consume mayor ancho de banda, ya que LIFT introduce menor sobrecarga en el proceso de descubrimiento de servicios. En este sentido LIFT optimiza mejor el ancho de banda que AODV-SD. Minimizar en este tema es uno de los objetivos impuestos en este trabajo.

LIFT minimiza el *consumo energético* ya que consume menos energía que AODV-SD. Al igual que en la métrica anterior, también es importante para nosotros saber que LIFT minimiza este recurso.

La *tasa de entrega de paquetes* mide de alguna forma la efectividad de un protocolo, en especial en esta métrica LIFT supera significativamente a AODV-SD.

LIFT transmite datos a velocidades superiores a las que transmite AODV-SD, existen diferencias significativas entre ambas propuestas. Podemos afirmar que LIFT tiene ligeramente mejor *throughput* que AODV-SD.

Por lo que respecta al *promedio de saltos*, hemos comprobado en la comparativa, que LIFT emplea en promedio menos saltos para descubrir servicios que AODV-SD. Y esta diferencia es muy clara, LIFT generalmente descubre servicios dentro de mismo clusters, de ahí que realice menor número de saltos, mientras que AODV-SD busca por toda la red.

En relación a la métrica *carga de encaminamiento normalizada*, LIFT introduce menor carga de encaminamiento normalizada que AODV-SD. Debido principalmente a que en LIFT no se rompen los enlaces de rutas tan frecuentemente, porque la densidad de nodos y la movilidad se circunscriben a áreas determinadas.

En cuanto al *retardo extremo a extremo*, se ha visto que AODV-SD emplea menor retardo extremo a extremo que LIFT. Se puede decir que en este parámetro AODV-SD tiene ventaja sobre LIFT. Esta ventaja se acentúa más cuando existe menor número de nodos y conforme se va incrementando este número, se va reduciendo la diferencia.

Finalmente, en lo que respecta al *tiempo de adquisición de servicios*. Se ha visto en la comparativa que las trayectorias de las curvas de ambos SDP, describen líneas ascendentes en zigzag que se cruzan en algunos puntos, por lo que no se observa una clara diferencia. No se puede afirmar que un SDP tengan mejores tiempos de adquisición de servicios que el otro. En cambio, podemos decir que los dos protocolos descubren servicios con tiempos de adquisición casi similares.

A la vista de los resultados se clarifican ligeras diferencias entre ambos protocolos, el resultado de la comparativa es muy favorable para LIFT, en general LIFT tiene mejor rendimiento que AODV-SD. Una razón sin duda alguna, es que LIFT no inunda de mensajes toda la red, sino que lanza los mensajes de petición en las cercanías del cluster, y por lo regular, dentro del mismo cluster encuentra el servicio solicitado, por tanto optimiza ancho de banda, energía, paquetes de control, carga de encaminamiento, velocidad de transmisión, número de saltos y tiempo de respuesta.

5.7. Análisis cualitativo

Discutiremos a continuación los resultados proporcionados por las simulaciones y daremos una explicación lógica a todo lo observado. Además, intentaremos responder las preguntas formuladas inicialmente en el apartado 1.5.

- ¿Minimiza nuestro enfoque el consumo de recursos (energía, ancho de banda, memoria) durante el proceso de descubrimiento de servicios?.
- ¿Cuántos servicios pueden ser descubiertos bajo este enfoque (tasa de éxito)?.

- ¿Es válido este enfoque para MANET de gran escala (100, 150, 200, 300, 400 nodos)?.
- ¿En cuáles escenarios de MANET existen posibilidades de éxito (conferencias, rescate, etc.) y bajo qué restricciones?.
- ¿Cuál es el costo del mantenimiento del clustering?.

Concluida la simulación se puede afirmar que el rendimiento de LIFT se ve afectado por diversos factores externos, tales como el grado de movilidad de los nodos, la distancia en términos de saltos entre los nodos, el protocolo de encaminamiento (AODV) utilizado dentro de la MANET, el grado de particiones de la red (clusters), el modelo de movilidad, entre otros factores.

Una vez conocido el rendimiento de nuestra propuesta, podemos decir que LIFT promete interesantes beneficios cuando se trata de un elevado número de nodos, ya que que maneja clusters y con ello el tráfico en la red sólo se intensifica en áreas delimitadas; por el contrario, cuando se maneja gran cantidad de nodos (no agrupados) el tráfico de mensajes inunda toda la red. En consecuencia, este efecto es perjudicial para el óptimo rendimiento de la MANET. Esto ha quedado demostrado en la comparativa con AODV-SD (sección 5.6).

Según los resultados de la simulación, LIFT optimiza ancho de banda y consumo de energía (también demostrado en la sección 5.6), esta minimización de recursos se produce gracias a que las peticiones de servicios se lanzan en el vecindario del cluster y en la mayoría de las ocasiones las búsquedas no trascienden más allá del área geográfica correspondiente, mientras que en otros enfoques las búsquedas se difunden por toda la red, incrementando por supuesto, el tráfico global.

Parece claro que LIFT minimiza recursos, sin embargo, introduce ciertos retardos para encontrar un servicio. Esto es muy característico de los protocolos reactivos (LIFT se basa en AODV que es un protocolo de encaminamiento reactivo) que tienen altos valores de latencias, puesto que los protocolos reactivos tienen en cuenta el retardo del proceso de encaminamiento. Por otro lado,

no se precisa actualizar frecuentemente las tablas de servicios. No obstante, los protocolos reactivos reducen en mayor medida la sobrecarga de paquetes de control que los protocolos proactivos. Se sabe que los enfoques basados en diseño Cross-layer logran reducir los mesajes de control de descubrimiento de servicios de la capa de aplicación, así como el costo de grandes paquetes del protocolo de encaminamiento y también de grandes tablas de encaminamiento [3].

En cuanto a la tasa de descubrimiento de servicios, LIFT tiene una media de descubrimiento de servicios de 95,39 %, es una tasa razonablemente alta. En el peor caso esta tasa ha sido de 85,67 % y en el mejor caso de 99,33 %. El número de nodos ha sido lo que determinantemente ha influido para conseguir altas tasas; a mayor número de nodos, mayor número de clusters, y buscar servicios dentro de un cluster siempre garantiza mayores expectativas de éxito.

En lo que respecta al número de nodos que puede soportar LIFT, se ha demostrado que descubre servicios con razonables tiempos de espera. Los tiempos de respuesta que presenta LIFT son tolerables. Se demuestra que sí es posible bajo este enfoque tener una MANET con gran número de nodos, hemos simulado 500 nodos con tiempos de respuestas más o menos aceptables. La razón de tener tiempos de retardos altos, se debe principalmente al protocolo de encaminamiento empleado, puesto que a este tipo de protocolo hay que añadir el propio retardo del proceso de encaminamiento, debido principalmente a que opera bajo demanda y que necesita actualizar sus tablas de rutas para mantener fresca la información de los enlaces, por lo que se introduce determinada latencia.

Respecto a los escenarios en donde tendría éxito LIFT y en los que se ha simulado, se ha pensado para espacios geográficamente delimitados, en áreas específicas en las que se sigue a nodos ricos en recursos, en este caso considerados líderes de clusters. Podemos suponer que se trata de salas de conferencias o salas de reuniones, formando una red espontánea, sin infraestructura fija, entre los dispositivos personales de los usuarios presentes. Se ha utilizado un patrón de movimientos orientado a grupos con movilidad aleatoria y sin pausas.

Las restricciones de los escenarios de LIFT comprenden áreas cerradas con gran número de nodos fraccionados en grupos. Una restricción importante de mencionar es que la movilidad debe ser aproximada al desplazamiento humano. No se ha considerado que funcione para escenarios abiertos con velocidades similares a la de los automóviles.

De acuerdo a las simulaciones y a la comparación con AODV-SD, el costo de mantener el cluster penaliza un poco el tiempo de retardo extremo a extremo; en este sentido, LIFT presenta altas tasas de descubrimiento de servicios y de PDR a cambio de tiempos de latencias relativamente elevados. El hecho de mantener actualizadas las tablas de servicios de clusters con alta movilidad, implica mayor sobrecarga de paquetes de control, más retardo y mayor cantidad de saltos. Este es el costo de mantener actualizado un cluster. Pero la recompensa es tener tasas de descubrimiento de servicios y tasas de entrega de paquetes por encima del 95 % y del 90 % respectivamente.

LIFT maneja diferentes tipos de servicios, pero al aumentar el número de servicios que LIFT puede ofrecer, también aumenta el número de proveedores de servicios y conforme aumenta el número de servidores, entonces disminuye el PDR, lo anterior como resultado de la congestión de la red. Esto implica una excesiva competencia por acceder al medio, en consecuencia, existe mucha pérdida de paquetes. Sin embargo, al introducir más número de nodos en el escenario, es posible aumentar el PDR, ya que más servidores ofrecerían esos servicios y se competiría menos por alcanzar a dichos proveedores de servicios.

Hemos visto en las simulaciones que la movilidad introduce un poco de carga de encaminamiento y un ligero incremento en el promedio de saltos; esto se debe principalmente a que a mayor velocidad de desplazamiento, mayor cantidad de enlaces rotos, ocasionando un aumento del tráfico en el esfuerzo por dar mantenimiento a las tablas de servicios.

Dado que una de las debilidades de LIFT ha sido su media de tiempo de respuestas, es conveniente la existencia de más proveedores de servicios con el fin de introducir más redundancia de servicios y con ello disminuir el tiempo

de respuesta de las peticiones de servicios; presumiblemente cuanto mayor sea el número de servicios disponibles y menor sea el número de nodos en la red, menor será el tiempo de respuesta para descubrir un servicio.

Se puede esperar un rendimiento razonable de un protocolo de descubrimiento de servicios que sea capaz de maximizar la tasa de descubrimiento de servicios y la tasa de entrega de paquetes, minimizar la sobrecarga de paquetes de control y el consumo de energía, minimizar el número de saltos, maximizar el throughput, minimizar la carga de encaminamiento normalizada y minimizar el tiempo de adquisición de servicios. Utilizar un protocolo de descubrimiento de servicios con diseño Cross-layer basado en un protocolo de encaminamiento como AODV puede ser lo idóneo, y sobre todo cuando tenemos alta movilidad y gran cantidad de nodos.

5.8. Escenarios idóneos

Una vez revisado el rendimiento de LIFT respecto a distintas métricas y vistos sus resultados, a continuación se mencionan los posibles escenarios para su aplicación.

Los escenarios idóneos para LIFT son dentro de espacios cerrados, en áreas físicas claramente delimitadas geográficamente; en MANET conformadas por dispositivos heterogéneos con presencia de dispositivos tanto ricos como limitados en recursos, aglomerados en clusters encabezados por líderes que aporten la mayor carga de trabajo, que asuman el papel de intermediarios y que den soporten al proceso de descubrimiento de servicios. Y que sean este tipo de dispositivos los que señalen la pauta de movilidad al resto de miembros del cluster; además, como medida de prevención existe más de un líder en caso de fallos o desconexión de la red.

En cuanto al perfil de los dispositivos líderes para esos escenarios, se consideran características de capacidades superiores (comunicación, cómputo, energía y memoria) con el objetivo de asumir el rol de nodo intermediario entre el resto de dispositivos del cluster.

Una posible aplicación es el servicio de emergencia en desastres naturales como erupciones volcánicas, huracanes, inundaciones y terremotos, puesto que en este tipo de situaciones no es posible garantizar el correcto funcionamiento de una red cableada o una red con infraestructura fija. En este sentido LIFT podría ser útil para salas de conferencias, aulas en campus, rescate en edificios, etc., para espacios públicos pero a la vez cerrados, en la que la movilidad de los nodos sea aproximada el movimiento humano en relación a la aleatoriedad de dirección, velocidad y aceleración.

Aplicable a entornos con gran número de usuarios móviles que ofrezcan y en igual medida demanden variados y distintos tipos de servicios; donde un mismo dispositivo pueda ofrecer más de un servicio a la vez; donde algunos dispositivos no ofrezcan ningún tipo de servicio o donde un dispositivo pueda solicitar y ofrecer servicios simultáneamente.

Esta sería la aplicabilidad para nuestro modelo propuesto, escenarios con gran número de nodos móviles, agrupados y liderados por nodos ricos en recursos y capacidades.

Los escenarios descritos se corresponden con las condiciones que artificialmente hemos reproducido en las simulaciones y de las que ya hemos comprobado el rendimiento. De modo que, si hipotéticamente estos escenarios fueran reales, el rendimiento sería el que ya se conoce.

Proyecto POPEYE

POPEYE es una iniciativa que tiene como objetivo desarrollar aplicaciones para entornos de trabajo colaborativo (CWE, Collaborative Working Environment) libre de las rígidas restricciones y severas limitaciones. El proyecto POPEYE (Professional Peer Environment Beyond Edge Computing) [126] se centra esencialmente en brindar soporte a entornos de grupos de trabajo colaborativo dinámicos y espontáneos con coordinación autónoma y soporte a la gestión del conocimiento. El proyecto POPEYE está dirigido a redes P2P móviles y a grupos Ad Hoc, en dichos entornos no se requiere infraestructura física previa, las comu-

nidades virtuales emergen espontáneamente y comparten datos con una apropiada calidad de servicios. POPEYE es un proyecto de investigación financiado por la UE (número de contrato IST-2006-034241, http://srvweb01.softeco.it/IST-Popeye/) en el que participa la Universidad de Murcia.

La definición de la arquitectura de software de POPEYE apunta que se pretende que el sistema sea fácil de usar por parte del usuario. Cuando un usuario de POPEYE quiera colaborar en un *Espacio de trabajo*[4] (WS) sólo tiene que poner en marcha el entorno POPEYE en su dispositivo inalámbrico y conectarse a la red POPEYE [46]. El dispositivo se conecta a una red overlay P2P construida sobre una red móvil Ad Hoc. Una vez conectado, un usuario puede buscar y unirse al WS existente o crear un nuevo WS e invitar a otros usuarios a unírsele para colaborar. Los usuarios que se unen al WS forman un *Grupo*. El *Espacio compartido* asociado a cada WS, da soporte a los datos compartidos entre los miembros del *Grupo*. Las aplicaciones que los usuarios emplean en el trabajo colaborativo (calendario del grupo, archivos compartidos, pizarra electrónica, etc.) están cargadas en tiempo de ejecución al entorno local de los usuarios de POPEYE. En la terminología de las MANET, las aplicaciones de los usuarios de POPEYE se corresponden con los servicios.

A propósito de escenarios de aplicación, LIFT se ha pensado dentro del marco de desarrollo de POPEYE y es en este contexto en el que se ha aplicado. Sobre todo, orientando el escenario de aplicación de LIFT, hacia el descubrimiento de servicios en esos espacios de trabajo y grupos de colaboración reunidos espontáneamente en salas de conferencias, en congresos, etc.

En cuanto a la capa de red, POPEYE propone un modelo basado en cluster operando sobre la capa de red. Este modelo se beneficia de la ubicación de los nodos para reducir la sobrecarga de comunicación del grupo. POPEYE usa DYMO como protocolo de encaminamiento (considerado como un sustituto del protocolo AODV); DYMO [23] está diseñado específicamente para trabajar con dispositivos heterogéneos en términos de poder de capacidad y cómputo.

[4]POPEYE llama *Workspace* (Espacio de trabajo) a un grupo de usuarios, datos y aplicaciones que comparten recursos entre sí.

Además, DYMO escala muy bien cientos de nodos introduciendo baja sobrecarga de control. Cabe resaltar que DYMO opera solamente a nivel de encaminamiento, por lo que no descubre servicios. De acuerdo a lo anterior, LIFT se puede aplicar en POPEYE para realizar el trabajo de descubrimiento de servicios.

El software de las aplicaciones MANET de POPEYE usa una arquitectura orientada a servicios, agrupa los nodos en clusters con el propósito de conseguir altos niveles de escalabilidad. POPEYE distingue un tipo de nodo llamado *super-peer* (SP) como organizador y responsable del cluster y es el encargado de enviar mensajes intra-cluster e inter-cluster.

La descripción funcional de POPEYE se ilustra de forma gráfica en una MANET compuesta por tres clusters. La topología de red de la figura 5.20 involucra tres tipos de nodos: tres *super-peers*, ocho *peers estándares* y seis *participantes M1 multicast*. La idea principal de este diseño es minimizar la sobrecarga de comunicación multicast mediante el envío de los mensajes multicast a través del *super-peer* ubicado dentro de cada cluster. Por un lado, el hecho de considerar los nodos ubicados físicamente cerca (dentro del mismo cluster) significa que la mayor parte de los mensajes se intercambien evitando que el cluster ejecute la mayoría de las tareas colaborativas. Por otro lado, los diferentes peers que estén o no, localizados en el mismo espacio físico, conforman los grupos de colaboración. Por eso, cuando los peers localizados en diferentes clusters quieran colaborar, intercambiarán mensajes multicast a través de los *super-peers*. El *super-peer* une todos los grupos multicast existentes en el cluster. Por ejemplo, cuando un *super-peer* intercepta un mensaje dirigido al grupo multicast M1, este mensaje se reenvía a todos los *super-peers* de la red. Entonces, cada *super-peer* entrega el mensaje al grupo multicast M1 del cluster local, en caso de existir nodos registrados a este grupo en el cluster [125].

Cabe destacar que el conjunto de *super-peers* no está predefinido y puede cambiar dinámicamente para ajustarse a la topología actual de la MANET. El mecanismo de selección de *super-peer* es local y se auto-ajusta, y puede tener en cuenta el perfil del dispositivo, proporcionado por los servicios de contexto,

así como información de la topología de red. El perfil del dispositivo incluye información acerca del tipo de dispositivo, ancho de banda y nivel de batería, entre otras características. La información de la topología de la red consiste básicamente del número de dispositivos que están a su alcance (generalmente vecinos) y de la información de nodos a 2 saltos de distancia.

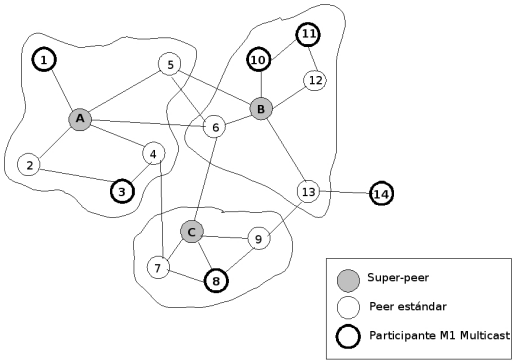


Figura 5.20: Topología de la red

Capítulo 6

Conclusiones y trabajos futuros

Hasta ahora esta tesis doctoral ha tratado todo lo relacionado con las tecnologías que hacen posible descubrir servicios en redes móviles Ad Hoc, ha repasado las distintas arquitecturas disponibles a día de hoy, ha identificado los problemas latentes en la mayoría de ellas, ha planteado un esquema de solución que contribuye en alguna medida a resolver dichos problemas, ha llevado a cabo la simulación de la propuesta y ha analizado los resultados de la simulación.

En este capítulo se recogen las contribuciones más importantes extraídas de esta investigación. En la sección 6.1 se exponen las principales conclusiones obtenidas al término del estudio. Mientras que en la sección 6.2 se mencionan las líneas de investigación abiertas para su estudio posterior.

Principales contribuciones

- **Estudio de las principales arquitecturas de descubrimiento de servicios**. En el capítulo 2 de esta tesis realizamos una revisión referente al tema de las MANET y al tema de descubrimiento de servicios; se estudian las diferentes soluciones existentes, se explora una larga lista de arquitecturas disponibles, los mecanismos y técnicas en las que se basan, así como los problemas aún por resolver.

- **Clasificación de las arquitecturas de descubrimiento de servicios**. Dentro del mismo capítulo 2 realizamos una clasificación de las arquitec-

turas de descubrimiento de servicios en función de si la arquitectura tiene o no directorio de servicios. También elaboramos un cuadro que resume las principales características de las arquitecturas respecto al tipo de directorio, topología, tipo de red, almacenamiento de información de servicios y técnica de descubrimiento de servicios.

- **Propuesta de un protocolo de descubrimiento de servicios**. Se ha diseñado una propuesta que combina una solución para descubrir servicios basado en diseño Cross-layer con un mecanismo que agrupa los dispositivos presentes en la red y los organiza en clusters, selecciona y declara líderes de clusters a los HCD para desempeñen el rol de intermediarios y se encarguen de dar soporte al descubrimiento de servicios.

- **Modificación de los formatos de mensajes de AODV**. El diseño Cross-layer empleado se basa en el protocolo de encaminamiento AODV. Para tal efecto, realizamos una extensión a una versión posterior a AODV conocida como AODV-SD. A dicha versión le hemos añadido un campo adicional (denominado *Node_Type*) a las cabeceras de algunos mensajes que utilizaremos para diferenciar el tipo de dispositivo y que nos es útil para la gestión de agrupamiento y para el mecanismo de descubrimiento de servicios.

- **Simulación de LIFT en una herramienta de simulación**. Para probar la viabilidad y el rendimiento de LIFT hemos utilizado la herramienta de simulación Ns-2, se han diseñado los respectivos scripts en OTcl, los archivos fuentes del patrón de movilidad y el programa en lenguaje AWK para el filtrado y cómputo de las variables que nos interesa medir.

- **Diseño de un escenario con el patrón de movilidad**. Para simular nuestra propuesta con la herramienta Ns-2, ha sido necesario crear el escenario apropiado en el que se desplieguen los dispositivos móviles, se definan sus posiciones iniciales y su patrón de movimientos a seguir. Del mismo modo, se definan las áreas de los clusters, la posición inicial de los

líderes y el patrón de movilidad de los líderes. En ese escenario se define el modelo de movilidad que deben observar todos los nodos de la red. Para tal efecto hemos modificado el modelo de movilidad Random Waypoint y lo hemos adaptado a la formación de clusters, este modelo de movilidad modificado lo hemos denominado CRWP (Clustering Random Waypoint).

- **Evaluación del rendimiento de LIFT con varias métricas**. Una de las más importantes contribuciones en el desarrollo de esta tesis, ha sido sin duda, la inclusión de distintas métricas utilizadas para evaluar el rendimiento de LIFT. Estas métricas son: tasa de descubrimiento de servicios, sobrecarga de paquetes de control, consumo energético, tasa de entrega de paquetes, promedio de saltos, throughput, carga de encaminamiento normalizada y tiempo de adquisición de servicios. Cabe destacar que en otros trabajos sólo se han tenido en cuenta dos o tres de estas métricas.

6.1. Conclusiones

Se han repasado las más importantes arquitecturas de descubrimiento de servicios y se han identificado los problemas subsistentes que presentan debido esencialmente al enfoque que plantean. Se han visto arquitecturas, bien basadas en el modelo de interacción directa que inunda de mensajes la red o bien basadas en el modelo de interacción indirecta que emplea intermediarios pero que no inunda la red con mensajes; y se han analizado los problemas que implica cada modelo.

Una vez analizadas las ventajas del modelo de interacción directa y del modelo de interacción indirecta, hemos planteado una propuesta de solución que combina algunas características de ambos modelos con una solución para descubrir servicios basado en diseño Cross-layer, empleando el protocolo de encaminamiento AODV como mecanismo de red subyacente. El diseño Cross-layer lleva a cabo el proceso de encaminamiento y el proceso de descubrimiento de servicios simultáneamente, aprovechando los mensajes de control con el fin de

optimizar el número de mensajes necesarios en este proceso. Nuestra propuesta **LIFT** organiza los nodos de la red en clusters y elige líderes de los clusters a los dispositivos con capacidades superiores para que asuman el rol de intermediarios y tengan por tanto, mayor carga de trabajo.

Hemos visto mediante las simulaciones llevadas a cabo, que LIFT logra cumplir con los siguientes puntos:

- **Optimizar el consumo de recursos (ancho de banda y energía)**.

  Para comprobar si LIFT optimiza ancho de banda y energía, hemos llevado a cabo dos mediciones para cada recurso. Un primer grupo de mediciones primero evalúa la sobrecarga de paquetes de control en función del número de nodos y la movilidad y después evalúa la sobrecarga de paquetes de control en función del número de peticiones de servicios. Otro segundo grupo de mediciones evalúa en primer término la energía consumida en función del número de nodos y la movilidad y en segundo término evalúa la energía consumida en función del número de peticiones de servicios. En ambas series de pruebas se observa que el número de nodos es un factor clave en la sobrecarga de paquetes de control y la energía consumida. El número de peticiones de servicios impacta sobre el ancho de banda y la energía consumida, mientras que la movilidad en sí, no es un factor determinante tanto para el ancho de banda como para el consumo energético.

  Otra manera de corroborar el rendimiento de LIFT en este particular es la comparativa realizada, en la que se puede observar que LIFT consume menos ancho de banda y energía que el protocolo con el que se ha comparado. Con los experimentos simulados, se ha visto que tanto para la sobrecarga de paquetes de control como para el consumo energético, LIFT consume menos recursos que AODV-SD.

  La optimización de recursos se debe principalmente a que las peticiones de servicios se lanzan dentro del vecindario del propio cluster y la mayoría de las veces la búsquedas no rebasan los límites del cluster; en este caso

el tráfico se limita a un ámbito local. Mientras que otras arquitecturas no orientadas a grupos difunden las peticiones de servicios por toda la red.

- **Obtener tasas de descubrimiento de servicios con alto grado de éxito**.

  De acuerdo a los resultados de las simulaciones se ha obtenido una media de la tasa de descubrimiento de servicios del 95,39 %, de los que un buen porcentaje de los servicios descubiertos es realizado por HCD y sólo un mínimo porcentaje es descubierto por LCD. Cabe mencionar, que la distribución de los proveedores de servicios es del 50 % para cada tipo de dispositivo, lo cual demuestra que los HCD tienen mayor carga de trabajo y son responsables de descubrir la mayor cantidad de servicios.

  En base a las simulaciones realizadas en la comparativa con AODV-SD, podemos afirmar que LIFT alcanza tasas más altas de descubrimiento de servicios que AODV-SD.

  El estudio de los resultados de las simulaciones indican que el número de nodos influye determinantemente en altas tasas de descubrimientos de servicios, debido principalmente a que un elevado número de nodos propicia que haya más clusters y de esta forma, existe mayor garantía de éxito al realizar búsquedas dentro de clusters.

- **Obtener niveles de escalabilidad razonables**.

  Para probar el nivel de escalabilidad que alcanza LIFT, hemos llevado a cabo cinco series de experimentos, con 500, 750, 1000, 1250 y 1500 nodos con redundancias de 5, 10, 15 y 20 réplicas de servicios a una velocidad media de 10 Km/h. Los resultados obtenidos indican que a medida que aumenta el número de nodos y disminuye el nivel de redundancia de servicios, aumenta el tiempo de adquisición de servicios. Este comportamiento es normal. Un ejemplo que da cuenta de este crecimiento es el siguiente: tomando resultados de experimentos previos, tenemos que el tiempo de adquisición de servicios en 100 nodos es de 0.020641 segs, en 200 nodos

0.041361 segs, en 300 nodos 0.046867 segs, en 400 nodos 0.056285 segs, en 500 nodos 0.055634 segs, en 750 nodos 0.067769 segs, en 1000 nodos 0.074629 segs, en 1250 nodos 0.079635 y en 1500 nodos 0.082321 segs. Se puede observar que el crecimiento es casi lineal y no exponencial. Hasta donde decidimos simular, se puede decir que el tiempo de adquisición de servicios de LIFT es elevado. Este tiempo aumenta conforme aumenta el tamaño de la red; los tiempos de adquisición de servicios son significativos pero no son tan intolerables.

Cabe añadir que el tipo de protocolo de encaminamiento reactivo sobre el que se basa LIFT, imprime cierto grado de ralentización al operar bajo demanda y por tanto tiene en cuenta el propio retardo del proceso de encaminamiento. Sin embargo, el protocolo de encaminamiento reactivo optimiza ancho de banda, puesto que no actualiza frecuentemente las tablas de servicios, reduciendo con ello la sobrecarga de paquetes de control.

- **Definir los escenarios de aplicación idóneos para nuestro modelo**.

  LIFT se ha simulado sobre un escenario especialmente diseñado, podemos decir que el escenario de aplicación de LIFT podría ser en entornos indoor, en donde en ocasiones no es posible garantizar el funcionamiento deseable de una red cableada o disponer de una infraestructura fija debido a la movilidad de los usuarios participantes. Dentro de este contexto LIFT pudiera ser de gran utilidad en aulas en campus, salas de reuniones, salas de conferencias y en general dentro de edificios; siempre teniendo en cuenta estas dos restricciones: primera, que sea dentro de espacios cerrados y segunda, con una movilidad similar al desplazamiento que tienen las personas al andar (aleatoriedad de dirección, velocidad y aceleración).

  En cuanto al espacio, se deben considerar áreas específicas geográficamente delimitadas, en las que se formen redes espontáneamente con la sola presencia de los dispositivos de los asistentes. Esta red se forma sin la previa existencia de infraestructura fija.

En cuanto a los dispositivos, suponemos la conexión a la red de dispositivos heterogéneos, especialmente esperando la presencia de dispositivos ricos en recursos (memoria, batería, procesamiento, alcance) este tipo de dispositivo será tenido en cuenta para liderar grupos con el objetivo de brindar soporte en el descubrimiento de servicios.

En cuanto a los servicios, aplicable a escenarios en los que exista un gran número de dispositivos móviles que ofrezcan y demanden distintos tipos de servicios, que un solo dispositivo ofrezca muchos servicios, que algunos dispositivos no ofrezcan ningún servicio y que un dispositivo demande y ofrezca al mismo tiempo servicios.

- **Conocer el costo del mantenimiento del agrupamiento**.

  Utilizando como base los resultados de las simulaciones tanto para medir el propio rendimiento de LIFT como para comparar su actuación frente a AODV-SD, podemos afirmar que el tiempo de retardo extremo a extremo penaliza al mantenimiento del mecanismo de agrupamiento. Sabemos que LIFT presenta elevados tiempos de latencia, esos tiempos de latencias es debido al propio protocolo de encaminamiento, que opera bajo demanda y que requiere actualizar sus tablas de rutas para mantener fresca la información de los enlaces. El costo de mantenimiento del clustering se agudiza más cuando la red tiene mayor movilidad, en este caso requiere refrescar con mayor frecuencia las rutas hacia los destinos conocidos. Y es sabido que los protocolos reactivos no intercambian información con tanta periodicidad y que únicamente mantienen rutas hacia los nodos activos.

  En conclusión, el costo de mantenimiento de un cluster con alta movilidad exige mantener actualizadas las tablas de servicios, lo que ocasiona introducir mayor sobrecarga de paquetes de control y mayor tasa de retardo extremo a extremo. A cambio de este costo se obtienen atractivas tasas de descubrimiento de servicios.

6.2. Trabajos futuros

Los resultados obtenidos en esta tesis suponen un avance en el tema de descubrimiento de servicios en MANET, sin embargo es un tema aún abierto al que se le pueden añadir propuestas que abran nuevas líneas de investigación. En este sentido y continuando con la misma línea de recurrir a un diseño Cross-layer para descubrir servicios durante el proceso de encaminamiento, mediante la formación de clusters liderados por dispositivos de capacidades superiores, se proponen los siguientes trabajos:

- **Implementar un mecanismo de elección de líder de clusters**.

  En nuestro diseño hemos simulado experimentalmente escenarios en los que el tipo de dispositivo es asignado previamente por nosotros. Es decir, los nodos que han de ser de tipo HCD son designados en el script en OTcl que emplea la herramienta de simulación Ns-2. El motivo principal es simplemente demostrar que los HCD puedan asumir el papel de CL. Con el propósito de hacer más sólida y estricta la estrategia para distinguir y elegir al líder del mecanismo de clustering, es preciso emplear algoritmos ampliamente probados (Highest-Degree heuristic, Lowest-ID heuristic, Node-weight heuristic, Weighted Clustering algorithm y Distributed Weighted Clustering Algorithm). De esta forma, en los escenarios de aplicación reales la elección de líder se hará en base a cualquier algoritmo de los mencionados.

  Estos algoritmos de clustering habitualmente eligen sus líderes de clusters en atención a cierto criterio de selección, bien sea por medio del valor del peso de conectividad o bien de acuerdo al mayor o menor valor del identificador (ID) o bien en función del número de saltos de distancia que existe entre los nodos.

  Como línea de investigación futura planteamos implementar un mecanismo que automáticamente identifique el tipo de dispositivo y si tiene las

características esperadas, luego le elija líder del cluster, para ello sugerimos utilizar el WCA [27] (Weighted Clustering Algorithm), algoritmo distribuido basado en pesos, que elige a sus líderes en base al criterio del valor del peso que posee cada nodo, teniendo en cuenta aspectos relacionados a las prestaciones del dispositivo. Lo anterior es lo más aproximado al esquema que se propone en LIFT, en el que el dispositivo que lidera el cluster debe tener características de capacidades superiores en varios aspectos (memoria, cómputo, batería, comunicación).

- **Probar LIFT con otro mecanismo de encaminamiento**.

  A la vista de los resultados obtenidos en la simulaciones y a las características de los protocolos de encaminamientos para redes Ad Hoc, se puede decir que los protocolos reactivos introducen poca sobrecarga de paquetes de control, como resultado se optimiza el ancho de banda y se obtienen altas tasas de entrega de paquetes y altas tasas de descubrimiento de servicios, pero en cambio, se obtienen tiempos de latencias muy elevados.

  En nuestro escenario de simulación la movilidad es baja, sólo hemos simulado condiciones de movilidad parecidas a las humanas. Mientras que en los resultados de algunas mediciones, la movilidad no ha impactado en el parámetro de la métrica. Por tal razón resultaría de utilidad simular LIFT con un protocolo de encaminamiento proactivo con el fin de ver en qué medida mejoran los tiempos de latencias y de adquisición de servicios, que dicho sea de paso son los puntos débiles de LIFT.

  Los protocolos proactivos mantienen actualizada de manera consistente la información de encaminamiento desde cada nodo hacia todos los nodos de la red. Para ello utilizan tablas para almacenar la información de las rutas. Existen varios protocolos de encaminamiento proactivos (apartado 2.7.1) y la diferencia entre ellos radica en el número de tablas que maneja y el método para difundir los cambios.

Existe un protocolo de encaminamiento proactivo especialmente orientado a clusters, conocido como: CGSR [31] (Clusterhead Gateway Switch Routing). En principio, CGSR se basa en el protocolo de encaminamiento proactivo DSDV y le añade el mecanismo de elección de líder. De modo que, proponemos como línea de investigación futura, realizar una implementación del protocolo CGSR en un protocolo de descubrimiento de servicios con las características de LIFT, del que ya hemos descrito, utiliza como mecanismo de encaminamiento un protocolo reactivo.

- **Ceder temporalmente el liderazgo a otros CL**.

  Según los resultados de las simulaciones realizadas a LIFT, nos hemos percatado de que los CL consumen recursos en forma relativamente innecesaria. Creemos que es posible alternar el control del cluster al resto de los HCD del mismo cluster, dado que también se ha podido comprobar que la tasa de descubrimiento de servicios es alta, gracias a que existen varios CL activos en un mismo cluster, por lo que se puede probar la opción de alternar el liderazgo para evitar que los CL de un cluster agoten sus recursos simultáneamente.

  De acuerdo a lo anterior, como línea de trabajo futuro sugerimos implementar mecanismos de intercambio periódico de liderazgo y de períodos de descanso, mediante el establecimiento de políticas en las que los CL permanezcan activos por un tiempo aleatorio, por ejemplo: 20 segs, 30 segs, etc., posteriormente el CL actual anunciará a los otros HCD del cluster, que entrará en un período de descanso y el liderazgo será cedido temporalmente a otro HCD convirtiéndose éste en un CL por un espacio de tiempo determinado, y así sucesivamente transferir el control del cluster a otro HCD. De este modo se optimizan los recursos de los HCD, evitando que se desgasten los recursos simultáneamente y de esta forma se garantiza que el control de los clusters se mantenga activo por más tiempo.

Apéndice A

Script en OTcl

En este apéndice se incluye un script escrito en OTcl utilizado para interactuar con Ns-2. Este código envía a Ns-2 los parámetros iniciales, así como la serie de acciones a realizar en la simulación. En este archivo se incluyen el nombre del archivo donde está descrito el patrón de movimientos y comportamiento de los nodos móviles, el número de nodos, el número de proveedores de servicios, la lista de los 15 diferentes tipos de servicios, el tipo de dispositivo, la lista de nodos que proveen los servicios, el tiempo total de simulación, los tiempos aleatorios cuando se lanzan las peticiones de servicios, etc.

```
#  ===== ESCENARIO DE CINCO CLUSTERS y TRES CL x cluster ============================
#  100 nodos
#  15 tipos de servicios (desde nodo 30 .. 99 son peticionarios de servicios)
#  15 nodos proveedores de servicios - Tipo de nodo 1 (HCDs)
#  15 nodos proveedores de servicios - Tipo de nodo 2 (LCDs)

set opt(chan)   Channel/WirelessChannel       ;# Tipo de canal
set opt(prop)   Propagation/TwoRayGround      ;# Modelo de radio-propagation
set opt(netif)  Phy/WirelessPhy               ;# Tipo de interface de red
set opt(mac)    Mac/802_11                    ;# Tipo de MAC
set opt(ifq)    Queue/DropTail/PriQueue       ;# Tipo de interface queue (prioridad)
set opt(ll)     LL                            ;# Tipo de link layer
set opt(ant)    Antenna/OmniAntenna           ;# Modelo de antena
set opt(x)      750                           ;# X Dimension  750
set opt(y)      750                           ;# Y Dimension  750
set opt(ifqlen) 50                            ;# Máximo de paquetes en ifq
set opt(seed)   9999                          ;# semilla
set opt(tr)     nodocluster100nodes-2Acc.tr   ;# archivo trace
set opt(nam)    nodocluster100nodes-2Acc.nam  ;# archivo nam
set opt(adhocRouting)   AODV                  ;# protocolo de encaminamiento
set opt(nn)     100                           ;# Cantidad de nodos en la simulación
set opt(nm)     "scen-disaster-100Nodes-2Acc" ;# CRWP
set opt(stop)   600                           ;# Tiempo de simulación
set opt(nsvc)   15                            ;# Número de servicios disponibles
```

```
set opt(nreq)   300                          ;# Número de peticiones de servicios
LL set mindelay_  50us
LL set delay_     25us
LL set bandwidth_ 0                          ;# no es usado
Mac/802_11 set dataRate_ 11Mb
Phy/WirelessPhy set Pt_  7.214e-3
# --------- Opciones de energía ---------
set val(engmodel) EnergyModel
set val(txPower) 1.4                         ;# transmitting power in mW
set val(rxPower) 0.9                         ;# recving power in mW
set val(sensePower) 0.00000175               ;# sensing power in mW
set val(idlePower) 0.0                       ;# idle power in mW
set val(initeng) 20.0                        ;# Initial energy in Joules
# ================================================
# PROGRAMA PRINCIPAL
# ================================================
set run 1

# Inicializa las variables globales, crea las instacias del simulador
set ns_ [new Simulator]

# Establece el canal wireless, radio-model y los objetos topograficos
set wtopo [new Topography]

# Crea los archivos de trazas para ns (.tr) y nam (.nam)
set tracefd [open $opt(tr) w]
set namtrace    [open $opt(nam) w]
$ns_ use-newtrace
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Define la topologíaset opt(cp)
$wtopo load_flatgrid $opt(x) $opt(y)
#$wprop topography $wtopo

# Create God
set god_ [create-god $opt(nn)]

# Define la creación de los nodos
$ns_ node-config -adhocRouting $opt(adhocRouting) \
     -llType $opt(ll) \
     -macType $opt(mac) \
     -ifqType $opt(ifq) \
     -ifqLen $opt(ifqlen) \
     -antType $opt(ant) \
     -propInstance [new $opt(prop)] \
     -phyType $opt(netif) \
     -channel [new $opt(chan)] \
     -topoInstance $wtopo \
     -agentTrace ON \
     -routerTrace ON \
     -movementTrace OFF \
     -macTrace ON \
     -energyModel $val(engmodel) \
     -txPower $val(txPower) \
     -rxPower $val(rxPower) \
     -sensePower $val(sensePower) \set opt(cp)
     -idlePower $val(idlePower) \
     -initialEnergy $val(initeng)
```

```
# Crea un número especifico de nodos [$opt(nn)]
for {set i 0} {$i < $opt(nn) } {incr i} {
set node_($i) [$ns_ node]
$node_($i) random-motion 0
  }

# Define el modelo de movimientos del nodo
puts "Modelo de movilidad (CRWP) --> 100 nodos en movimiento ...."
source $opt(nm)

# Define la cantidad de provedores de servicios y clientes que
# existen en el escenario.
# Siembra la semilla por default RNG

global defaultRNG
$defaultRNG seed 9999            ;# 0 para (hora - día actual y un contador)

# crea los RNGs y los pone en el substream correspondiente
set id_nodoRNG              [new RNG]
set tiempo_peticionRNG      [new RNG]
set servicioRNG             [new RNG]
set tipo_nodoRNG            [new RNG]

for {set j 1} {$j < $run} {incr j} {
        $id_nodoRNG               next-substream
        $tiempo_peticionRNG       next-substream
        $servicioRNG              next-substream
        $tipo_nodoRNG             next-substream
}

# id_nodo_ y tiempo_peticion son variables aleatorias uniformes
set nodes [expr $opt(nn)-1]                ;# En todos los nodos
puts [format "%d" $nodes ]

set id_nodo_ [new RandomVariable/Uniform]    ;# Nodos que lanzan requests
$id_nodo_ set min_ 30                        ;# Desde los nodos que NO contienen
$id_nodo_ set max_ $nodes                    ;# ellos mismos los servicios hasta $nodes
$id_nodo_ use-rng  $id_nodoRNG

set tiempo_peticion_ [new RandomVariable/Uniform]     ;# Desde el segundo 2 ... 600
$tiempo_peticion_ set min_  2
$tiempo_peticion_ set max_ $opt(stop)
$tiempo_peticion_ use-rng  $tiempo_peticionRNG

set tipo_nodo_ [new RandomVariable/Uniform]    ;# Por tipo de nodo (1) HCD ó (2) LCD
$tipo_nodo_ set min_ 1                         ;# HCD
$tipo_nodo_ set max_ 2                         ;# LCD
$tipo_nodo_ use-rng  $tipo_nodoRNG

# Se asignan aleatoriamente los servicios
set servicio_ [new RandomVariable/Uniform]      ;# Asigna desde 1 ... 15
$servicio_ set min_ 1
$servicio_ set max_ 15                          ;# 15 diferentes tipos de servicios
$servicio_ use-rng  $servicioRNG

# Se asignan 15 servicios a los HCD 0 .. 14
for {set j 0} {$j < 15} {incr j} {
set cliente($j) [$node_($j) set ragent_]
}
$ns_ at 1.0 "$cliente(0) service-bind 1 DISK"
```

```
$ns_ at 1.0 "$cliente(0) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(0) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(0) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(0) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(1) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(1) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(1) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(1) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(1) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(2) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(2) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(2) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(2) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(2) service-bind 1 WEB"
$ns_ at 1.0 "$cliente(3) service-bind 1 DISK"
$ns_ at 1.0 "$cliente(3) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(3) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(3) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(3) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(4) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(4) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(4) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(4) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(4) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(5) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(5) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(5) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(5) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(5) service-bind 1 WEB"
$ns_ at 1.0 "$cliente(6) service-bind 1 DISK"
$ns_ at 1.0 "$cliente(6) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(6) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(6) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(6) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(7) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(7) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(7) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(7) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(7) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(8) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(8) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(8) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(8) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(8) service-bind 1 WEB"
$ns_ at 1.0 "$cliente(9) service-bind 1 DISK"
$ns_ at 1.0 "$cliente(9) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(9) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(9) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(9) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(10) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(10) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(10) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(10) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(10) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(11) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(11) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(11) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(11) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(11) service-bind 1 WEB"
$ns_ at 1.0 "$cliente(12) service-bind 1 DISK"
```

```
$ns_ at 1.0 "$cliente(12) service-bind 1 FAX"
$ns_ at 1.0 "$cliente(12) service-bind 1 FTP"
$ns_ at 1.0 "$cliente(12) service-bind 1 GPS"
$ns_ at 1.0 "$cliente(12) service-bind 1 MAIL"
$ns_ at 1.0 "$cliente(13) service-bind 1 MP3"
$ns_ at 1.0 "$cliente(13) service-bind 1 PHONE"
$ns_ at 1.0 "$cliente(13) service-bind 1 PRINTER"
$ns_ at 1.0 "$cliente(13) service-bind 1 RADAR"
$ns_ at 1.0 "$cliente(13) service-bind 1 SCANNER"
$ns_ at 1.0 "$cliente(14) service-bind 1 TELNET"
$ns_ at 1.0 "$cliente(14) service-bind 1 CAMERA"
$ns_ at 1.0 "$cliente(14) service-bind 1 TRANSLATOR"
$ns_ at 1.0 "$cliente(14) service-bind 1 VOICE"
$ns_ at 1.0 "$cliente(14) service-bind 1 WEB"
# Se asignan 15 servicios a los LCD 15 .. 30
for {set j 15} {$j < 30} {incr j} {
set cliente($j) [$node_($j) set ragent_]
}
$ns_ at 1.0 "$cliente(15) service-bind 2 DISK"
$ns_ at 1.0 "$cliente(15) service-bind 2 FAX"
$ns_ at 1.0 "$cliente(15) service-bind 2 FTP"
$ns_ at 1.0 "$cliente(15) service-bind 2 GPS"
$ns_ at 1.0 "$cliente(15) service-bind 2 MAIL"
$ns_ at 1.0 "$cliente(16) service-bind 2 MP3"
$ns_ at 1.0 "$cliente(16) service-bind 2 PHONE"
$ns_ at 1.0 "$cliente(16) service-bind 2 PRINTER"
$ns_ at 1.0 "$cliente(16) service-bind 2 RADAR"
$ns_ at 1.0 "$cliente(16) service-bind 2 SCANNER"
$ns_ at 1.0 "$cliente(17) service-bind 2 TELNET"
$ns_ at 1.0 "$cliente(17) service-bind 2 CAMERA"
$ns_ at 1.0 "$cliente(17) service-bind 2 TRANSLATOR"
$ns_ at 1.0 "$cliente(17) service-bind 2 VOICE"
$ns_ at 1.0 "$cliente(17) service-bind 2 WEB"
$ns_ at 1.0 "$cliente(18) service-bind 2 DISK"
$ns_ at 1.0 "$cliente(18) service-bind 2 FAX"
$ns_ at 1.0 "$cliente(18) service-bind 2 FTP"
$ns_ at 1.0 "$cliente(18) service-bind 2 GPS"
$ns_ at 1.0 "$cliente(18) service-bind 2 MAIL"
$ns_ at 1.0 "$cliente(19) service-bind 2 MP3"
$ns_ at 1.0 "$cliente(19) service-bind 2 PHONE"
$ns_ at 1.0 "$cliente(19) service-bind 2 PRINTER"
$ns_ at 1.0 "$cliente(19) service-bind 2 RADAR"
$ns_ at 1.0 "$cliente(19) service-bind 2 SCANNER"
$ns_ at 1.0 "$cliente(20) service-bind 2 TELNET"
$ns_ at 1.0 "$cliente(20) service-bind 2 CAMERA"
$ns_ at 1.0 "$cliente(20) service-bind 2 TRANSLATOR"
$ns_ at 1.0 "$cliente(20) service-bind 2 VOICE"
$ns_ at 1.0 "$cliente(20) service-bind 2 WEB"
$ns_ at 1.0 "$cliente(21) service-bind 2 DISK"
$ns_ at 1.0 "$cliente(21) service-bind 2 FAX"
$ns_ at 1.0 "$cliente(21) service-bind 2 FTP"
$ns_ at 1.0 "$cliente(21) service-bind 2 GPS"
$ns_ at 1.0 "$cliente(21) service-bind 2 MAIL"
$ns_ at 1.0 "$cliente(22) service-bind 2 MP3"
$ns_ at 1.0 "$cliente(22) service-bind 2 PHONE"
$ns_ at 1.0 "$cliente(22) service-bind 2 PRINTER"
$ns_ at 1.0 "$cliente(22) service-bind 2 RADAR"
$ns_ at 1.0 "$cliente(22) service-bind 2 SCANNER"
$ns_ at 1.0 "$cliente(23) service-bind 2 TELNET"
$ns_ at 1.0 "$cliente(23) service-bind 2 CAMERA"
```

```
$ns_ at 1.0 "$cliente(23) service-bind 2 TRANSLATOR"
$ns_ at 1.0 "$cliente(23) service-bind 2 VOICE"
$ns_ at 1.0 "$cliente(23) service-bind 2 WEB"
$ns_ at 1.0 "$cliente(24) service-bind 2 DISK"
$ns_ at 1.0 "$cliente(24) service-bind 2 FAX"
$ns_ at 1.0 "$cliente(24) service-bind 2 FTP"
$ns_ at 1.0 "$cliente(24) service-bind 2 GPS"
$ns_ at 1.0 "$cliente(24) service-bind 2 MAIL"
$ns_ at 1.0 "$cliente(25) service-bind 2 MP3"
$ns_ at 1.0 "$cliente(25) service-bind 2 PHONE"
$ns_ at 1.0 "$cliente(25) service-bind 2 PRINTER"
$ns_ at 1.0 "$cliente(25) service-bind 2 RADAR"
$ns_ at 1.0 "$cliente(25) service-bind 2 SCANNER"
$ns_ at 1.0 "$cliente(26) service-bind 2 TELNET"
$ns_ at 1.0 "$cliente(26) service-bind 2 CAMERA"
$ns_ at 1.0 "$cliente(26) service-bind 2 TRANSLATOR"
$ns_ at 1.0 "$cliente(26) service-bind 2 VOICE"
$ns_ at 1.0 "$cliente(26) service-bind 2 WEB"
$ns_ at 1.0 "$cliente(27) service-bind 2 DISK"
$ns_ at 1.0 "$cliente(27) service-bind 2 FAX"
$ns_ at 1.0 "$cliente(27) service-bind 2 FTP"
$ns_ at 1.0 "$cliente(27) service-bind 2 GPS"
$ns_ at 1.0 "$cliente(27) service-bind 2 MAIL"
$ns_ at 1.0 "$cliente(28) service-bind 2 PHONE"
$ns_ at 1.0 "$cliente(28) service-bind 2 PRINTER"
$ns_ at 1.0 "$cliente(28) service-bind 2 RADAR"
$ns_ at 1.0 "$cliente(28) service-bind 2 SCANNER"
$ns_ at 1.0 "$cliente(28) service-bind 2 TELNET"
$ns_ at 1.0 "$cliente(29) service-bind 2 CAMERA"
$ns_ at 1.0 "$cliente(29) service-bind 2 TRANSLATOR"
$ns_ at 1.0 "$cliente(29) service-bind 2 VOICE"
$ns_ at 1.0 "$cliente(29) service-bind 2 WEB"
$ns_ at 1.0 "$cliente(29) service-bind 2 DISK"

# --- Numero de request de servicios
for {set j 0} {$j <300} {incr j} {
# ---  Nodo que hace el request de servicio
  set cliente($j) [$node_([expr round([$id_nodo_ value])]) set ragent_]
# --- servicio solicitado en el request
  set service_ [expr round([$servicio_ value])]
# --- tiempo cuando se lleva a cabo el request
  set time_req [expr round([$tiempo_peticion_ value])]
# --- tipo de nodo que se trate
  set nodetype [expr round([$tipo_nodo_ value])]

   if {$service_ == 1} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion PRINTER"
      }
   if {$service_ == 2} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion WEB"
      }
   if {$service_ == 3} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion MAIL"
      }
   if {$service_ == 4} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion FTP"
      }
   if {$service_ == 5} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion TELNET"
      }
```

```
   if {$service_ == 6} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion DISK"
      }
   if {$service_ == 7} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion FAX"
      }
   if {$service_ == 8} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion CAMERA"
      }
   if {$service_ == 9} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion RADAR"
      }
   if {$service_ == 10} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion GPS"
      }
   if {$service_ == 11} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion TRANSLATOR"
      }
   if {$service_ == 12} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion SCANNER"
      }
   if {$service_ == 13} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion MP3"
      }
   if {$service_ == 14} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion PHONE"
      }
   if {$service_ == 15} {
         $ns_ at $time_req "$cliente($j) servicio-descripcion VOICE"
      }
 }

# Define la posicion inicial del nodo en el nam
for {set i 0} {$i < $opt(nn)} {incr i} {
   $ns_ initial_node_pos $node_($i) 20
}

# Indica a los nodos cuando termina la simulación
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
proc finish {} {
        global ns_ tracefd namtrace f0 f1 l0 l1
        $ns_ flush-trace
    $ns_ halt
        close $tracefd
    close $namtrace
    exec nam nodocluster5_100requests.nam  &
            exec xgraph nodocluster5.tr  -geometry 800x400 &
        exit 0
}
# Indica a nam el tiempo de duración de la simulacion
$ns_ at  $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
$ns_ at  $opt(stop).000000001 "puts \"NS EXITING...\" ; finish"
puts "Starting Simulation ..."
puts "-------------------------------------------------------------------------------"
puts "Time (secs)     Node      Service requested and node where have been found      "
puts "-------------------------------------------------------------------------------"
$ns_ run
```

Apéndice B

Visualización de la simulación

A continuación incluimos una salida típica de la simulación en Ns-2, esta salida se obtiene a partir de la ejecución del script en OTcl incluido en el apéndice A.

En este listado se muestra por ejemplo: el primer evento ocurre a los 13 segundos, cuando el nodo 44 lanza una petición de servicios solicitando el servicio WEB, una vez transcurrido 0.054512 segundos, justo a los 13.054512 segundos, se anuncia que se ha descubierto el servicio WEB (solicitado por el nodo 44) en el dispositivo 11 que es de tipo HCD; y así sucesivamente para el resto de eventos hasta cumplir 600 segundos y solicitadas las 300 peticiones de servicios.

```
num_nodes is set 100
Modelo de movilidad (CRWP) --> 100 nodos en movimiento ....
99
Starting Simulation ...
----------------------------------------------------------------------------
Time (secs)     Node      Service requested and node where have been found
----------------------------------------------------------------------------
At 13.000000    node 44 - Started to discover the WEB service
At 13.054512    node 44 - Discovered the WEB service on device 11 type: 1 in 0.054512 secs.
At 17.000000    node 92 - Started to discover the VOICE service
At 17.000000    node 93 - Started to discover the VOICE service
At 17.079108    node 92 - Discovered the VOICE service on device 11 type: 1 in 0.079108 secs.
At 17.081066    node 93 - Discovered the VOICE service on device 5 type: 1 in 0.081066 secs.
At 18.000000    node 53 - Started to discover the PRINTER service
At 18.059042    node 53 - Discovered the PRINTER service on device 13 type: 1 in 0.059042 secs.
At 19.000000    node 41 - Started to discover the WEB service
At 19.000000    node 31 - Started to discover the PRINTER service
At 19.000000    node 32 - Started to discover the FAX service
At 19.006446    node 41 - Discovered the WEB service on device 11 type: 1 in 0.006446 secs.
At 19.042652    node 31 - Discovered the PRINTER service on device 25 type: 2 in 0.042652 secs.
At 19.097041    node 32 - Discovered the FAX service on device 9 type: 1 in 0.097041 secs.
At 20.000000    node 74 - Started to discover the TRANSLATOR service
```

```
At 20.000000    node 84 - Started to discover the DISK service
At 20.005277    node 74 - Discovered the TRANSLATOR service on device 2 type: 1 in 0.005277 secs.
At 20.039413    node 84 - Discovered the DISK service on device 6 type: 1 in 0.039413 secs.
At 21.000000    node 79 - Started to discover the CAMERA service
At 21.011902    node 79 - Discovered the CAMERA service on device 2 type: 1 in 0.011902 secs.
At 22.000000    node 41 - Started to discover the FTP service
At 22.051698    node 41 - Discovered the FTP service on device 12 type: 1 in 0.051698 secs.
At 25.000000    node 93 - Started to discover the PRINTER service
At 25.000000    node 47 - Started to discover the DISK service
At 25.027902    node 93 - Discovered the PRINTER service on device 4 type: 1 in 0.027902 secs.
At 25.065134    node 47 - Discovered the DISK service on device 12 type: 1 in 0.065134 secs.
At 34.000000    node 81 - Started to discover the TELNET service
At 34.011133    node 81 - Discovered the TELNET service on device 2 type: 1 in 0.011133 secs.
At 37.000000    node 40 - Started to discover the CAMERA service
At 37.051866    node 40 - Discovered the CAMERA service on device 11 type: 1 in 0.051866 secs.
At 39.000000    node 49 - Started to discover the DISK service
At 39.057808    node 49 - Discovered the DISK service on device 6 type: 1 in 0.057808 secs.
At 43.000000    node 85 - Started to discover the FTP service
At 43.034401    node 85 - Discovered the FTP service on device 6 type: 1 in 0.034401 secs.
At 48.000000    node 95 - Started to discover the CAMERA service
At 48.005181    node 95 - Discovered the CAMERA service on device 5 type: 1 in 0.005181 secs.
At 54.000000    node 54 - Started to discover the MP3 service
At 54.000000    node 62 - Started to discover the RADAR service
At 54.057887    node 54 - Discovered the MP3 service on device 13 type: 1 in 0.057887 secs.
At 58.000000    node 45 - Started to discover the SCANNER service
At 58.003373    node 45 - Discovered the SCANNER service on device 10 type: 1 in 0.003373 secs.
At 60.000000    node 62 - service RADAR couldn't be discovered on any device
At 61.000000    node 79 - Started to discover the FAX service
At 61.039089    node 79 - Discovered the FAX service on device 3 type: 1 in 0.039089 secs.
At 64.000000    node 79 - Started to discover the CAMERA service
At 64.037321    node 79 - Discovered the CAMERA service on device 2 type: 1 in 0.037321 secs.
At 65.000000    node 88 - Started to discover the TRANSLATOR service
At 65.000000    node 85 - Started to discover the PHONE service
At 65.059912    node 85 - Discovered the PHONE service on device 4 type: 1 in 0.059912 secs.
At 65.067640    node 88 - Discovered the TRANSLATOR service on device 5 type: 1 in 0.067640 secs.
At 66.000000    node 61 - Started to discover the SCANNER service
At 66.003333    node 61 - Discovered the SCANNER service on device 13 type: 1 in 0.003333 secs.
At 68.000000    node 55 - Started to discover the CAMERA service
At 68.064463    node 55 - Discovered the CAMERA service on device 14 type: 1 in 0.064463 secs.
At 69.000000    node 97 - Started to discover the PHONE service
At 69.024882    node 97 - Discovered the PHONE service on device 4 type: 1 in 0.024882 secs.
At 72.000000    node 62 - Started to discover the TRANSLATOR service
At 72.037964    node 62 - Discovered the TRANSLATOR service on device 14 type: 1 in 0.037964 secs.
At 75.000000    node 90 - Started to discover the TRANSLATOR service
At 75.000000    node 46 - Started to discover the PHONE service
At 75.003613    node 46 - Discovered the PHONE service on device 10 type: 1 in 0.003613 secs.
At 75.014967    node 90 - Discovered the TRANSLATOR service on device 5 type: 1 in 0.014967 secs.
At 76.000000    node 66 - Started to discover the MP3 service
At 76.000000    node 54 - Started to discover the TELNET service
At 76.034225    node 66 - Discovered the MP3 service on device 1 type: 1 in 0.034225 secs.
At 76.041040    node 54 - Discovered the TELNET service on device 14 type: 1 in 0.041040 secs.
At 77.000000    node 69 - Started to discover the WEB service
At 77.029426    node 69 - Discovered the WEB service on device 2 type: 1 in 0.029426 secs.
At 79.000000    node 43 - Started to discover the FTP service
At 79.000000    node 80 - Started to discover the PHONE service
At 79.016761    node 43 - Discovered the FTP service on device 27 type: 2 in 0.016761 secs.
At 83.000000    node 78 - Started to discover the GPS service
At 83.041583    node 78 - Discovered the GPS service on device 3 type: 1 in 0.041583 secs.
At 84.000000    node 56 - Started to discover the FAX service
At 84.005456    node 56 - Discovered the FAX service on device 12 type: 1 in 0.005456 secs.
```

```
At 85.000000    node 85 - Started to discover the TELNET service
At 85.003233    node 85 - Discovered the TELNET service on device 5 type: 1 in 0.003233 secs.
At 87.000000    node 80 - service PHONE couldn't be discovered on any device
At 90.000000    node 38 - Started to discover the CAMERA service
At 90.039088    node 38 - Discovered the CAMERA service on device 26 type: 2 in 0.039088 secs.
At 91.000000    node 41 - Started to discover the VOICE service
At 91.025045    node 41 - Discovered the VOICE service on device 11 type: 1 in 0.025045 secs.
At 92.000000    node 63 - Started to discover the FAX service
At 92.006792    node 63 - Discovered the FAX service on device 9 type: 1 in 0.006792 secs.
At 96.000000    node 92 - Started to discover the VOICE service
At 96.000000    node 95 - Started to discover the WEB service
At 96.003293    node 95 - Discovered the WEB service on device 5 type: 1 in 0.003293 secs.
At 96.057985    node 92 - Discovered the VOICE service on device 5 type: 1 in 0.057985 secs.
At 99.000000    node 69 - Started to discover the RADAR service
At 99.041164    node 69 - Discovered the RADAR service on device 1 type: 1 in 0.041164 secs.
At 106.000000   node 98 - Started to discover the DISK service
At 107.000000   node 66 - Started to discover the MAIL service
At 107.035507   node 66 - Discovered the MAIL service on device 3 type: 1 in 0.035507 secs.
At 112.000000   node 32 - Started to discover the DISK service
At 112.005437   node 32 - Discovered the DISK service on device 9 type: 1 in 0.005437 secs.
At 114.000000   node 98 - service DISK couldn't be discovered on any device
At 119.000000   node 71 - Started to discover the TRANSLATOR service
At 119.002933   node 71 - Discovered the TRANSLATOR service on device 2 type: 1 in 0.002933 secs.
At 121.000000   node 85 - Started to discover the SCANNER service
At 121.000000   node 63 - Started to discover the RADAR service
At 121.003033   node 63 - Discovered the RADAR service on device 13 type: 1 in 0.003033 secs.
At 121.006971   node 85 - Discovered the SCANNER service on device 4 type: 1 in 0.006971 secs.
At 122.000000   node 62 - Started to discover the PRINTER service
At 122.005537   node 62 - Discovered the PRINTER service on device 13 type: 1 in 0.005537 secs.
At 123.000000   node 60 - Started to discover the MAIL service
At 123.055860   node 60 - Discovered the MAIL service on device 12 type: 1 in 0.055860 secs.
At 129.000000   node 45 - Started to discover the TRANSLATOR service
At 129.003273   node 45 - Discovered the TRANSLATOR service on device 11 type: 1 in 0.003273 secs.
At 131.000000   node 91 - Started to discover the GPS service
At 131.003393   node 91 - Discovered the GPS service on device 6 type: 1 in 0.003393 secs.
At 133.000000   node 53 - Started to discover the WEB service
At 133.064910   node 53 - Discovered the WEB service on device 11 type: 1 in 0.064910 secs.
At 135.000000   node 85 - Started to discover the MAIL service
At 139.000000   node 50 - Started to discover the FTP service
At 139.031209   node 50 - Discovered the FTP service on device 12 type: 1 in 0.031209 secs.
At 140.000000   node 78 - Started to discover the WEB service
At 140.000000   node 33 - Started to discover the FTP service
At 140.005963   node 78 - Discovered the WEB service on device 2 type: 1 in 0.005963 secs.
At 140.057650   node 33 - Discovered the FTP service on device 27 type: 2 in 0.057650 secs.
At 141.000000   node 81 - Started to discover the TELNET service
At 141.000000   node 98 - Started to discover the TELNET service
At 141.000000   node 85 - service MAIL couldn't be discovered on any device
At 141.003093   node 81 - Discovered the TELNET service on device 2 type: 1 in 0.003093 secs.
At 141.005577   node 98 - Discovered the TELNET service on device 5 type: 1 in 0.005577 secs.
At 148.000000   node 41 - Started to discover the FAX service
At 148.060639   node 41 - Discovered the FAX service on device 9 type: 1 in 0.060639 secs.
At 151.000000   node 58 - Started to discover the PRINTER service
At 151.007739   node 58 - Discovered the PRINTER service on device 13 type: 1 in 0.007739 secs.
At 153.000000   node 79 - Started to discover the DISK service
At 153.028732   node 79 - Discovered the DISK service on device 3 type: 1 in 0.028732 secs.
At 154.000000   node 48 - Started to discover the MP3 service
At 154.000000   node 62 - Started to discover the MP3 service
At 154.003233   node 62 - Discovered the MP3 service on device 13 type: 1 in 0.003233 secs.
At 154.003654   node 48 - Discovered the MP3 service on device 10 type: 1 in 0.003654 secs.
At 160.000000   node 41 - Started to discover the RADAR service
```

```
At 165.000000   node 66 - Started to discover the TELNET service
At 165.005336   node 66 - Discovered the TELNET service on device 2 type: 1 in 0.005336 secs.
At 168.000000   node 41 - service RADAR couldn't be discovered on any device
At 170.000000   node 49 - Started to discover the WEB service
At 170.000000   node 61 - Started to discover the MP3 service
At 170.038134   node 49 - Discovered the WEB service on device 14 type: 1 in 0.038134 secs.
At 170.086883   node 61 - Discovered the MP3 service on device 4 type: 1 in 0.086883 secs.
At 173.000000   node 86 - Started to discover the DISK service
At 173.062075   node 86 - Discovered the DISK service on device 6 type: 1 in 0.062075 secs.
At 175.000000   node 37 - Started to discover the PRINTER service
At 175.006651   node 37 - Discovered the PRINTER service on device 10 type: 1 in 0.006651 secs.
At 176.000000   node 50 - Started to discover the RADAR service
At 176.034726   node 50 - Discovered the RADAR service on device 13 type: 1 in 0.034726 secs.
At 177.000000   node 68 - Started to discover the CAMERA service
At 177.003373   node 68 - Discovered the CAMERA service on device 2 type: 1 in 0.003373 secs.
At 178.000000   node 96 - Started to discover the SCANNER service
At 178.006222   node 96 - Discovered the SCANNER service on device 4 type: 1 in 0.006222 secs.
At 180.000000   node 98 - Started to discover the PHONE service
At 180.000000   node 72 - Started to discover the PHONE service
At 180.000000   node 59 - Started to discover the PHONE service
At 180.037011   node 98 - Discovered the PHONE service on device 4 type: 1 in 0.037011 secs.
At 180.044524   node 59 - Discovered the PHONE service on device 13 type: 1 in 0.044524 secs.
At 181.000000   node 66 - Started to discover the MAIL service
At 181.000000   node 36 - Started to discover the MP3 service
At 181.003714   node 36 - Discovered the MP3 service on device 10 type: 1 in 0.003714 secs.
At 181.076602   node 66 - Discovered the MAIL service on device 3 type: 1 in 0.076602 secs.
At 182.000000   node 97 - Started to discover the CAMERA service
At 182.015791   node 97 - Discovered the CAMERA service on device 5 type: 1 in 0.015791 secs.
At 185.000000   node 47 - Started to discover the FAX service
At 185.003233   node 47 - Discovered the FAX service on device 9 type: 1 in 0.003233 secs.
At 186.000000   node 72 - service PHONE couldn't be discovered on any device
At 189.000000   node 30 - Started to discover the DISK service
At 189.000000   node 90 - Started to discover the WEB service
At 189.003293   node 30 - Discovered the DISK service on device 24 type: 2 in 0.003293 secs.
At 189.026007   node 90 - Discovered the WEB service on device 5 type: 1 in 0.026007 secs.
At 191.000000   node 87 - Started to discover the TRANSLATOR service
At 191.042342   node 87 - Discovered the TRANSLATOR service on device 5 type: 1 in 0.042342 secs.
At 192.000000   node 51 - Started to discover the MP3 service
At 192.000000   node 69 - Started to discover the RADAR service
At 192.003293   node 51 - Discovered the MP3 service on device 13 type: 1 in 0.003293 secs.
At 192.014957   node 69 - Discovered the RADAR service on device 19 type: 2 in 0.014957 secs.
At 195.000000   node 87 - Started to discover the GPS service
At 195.003293   node 87 - Discovered the GPS service on device 6 type: 1 in 0.003293 secs.
At 196.000000   node 73 - Started to discover the PHONE service
At 196.032963   node 73 - Discovered the PHONE service on device 19 type: 2 in 0.032963 secs.
At 197.000000   node 69 - Started to discover the FTP service
At 197.032505   node 69 - Discovered the FTP service on device 3 type: 1 in 0.032505 secs.
At 200.000000   node 66 - Started to discover the FAX service
At 201.000000   node 52 - Started to discover the GPS service
At 203.000000   node 40 - Started to discover the FTP service
At 203.000000   node 66 - Started to discover the RADAR service
At 203.059734   node 40 - Discovered the FTP service on device 24 type: 2 in 0.059734 secs.
At 204.000000   node 38 - Started to discover the TRANSLATOR service
At 204.037078   node 38 - Discovered the TRANSLATOR service on device 8 type: 1 in 0.037078 secs.
At 207.000000   node 76 - Started to discover the GPS service
At 207.000000   node 52 - service GPS couldn't be discovered on any device
At 207.000000   node 66 - service FAX couldn't be discovered on any device
At 207.029416   node 76 - Discovered the GPS service on device 3 type: 1 in 0.029416 secs.
At 210.000000   node 81 - Started to discover the WEB service
At 210.000000   node 93 - Started to discover the MP3 service
```

```
At 210.003213   node 93 - Discovered the MP3 service on device 4 type: 1 in 0.003213 secs.
At 210.040003   node 81 - Discovered the WEB service on device 2 type: 1 in 0.040003 secs.
At 212.000000   node 67 - Started to discover the RADAR service
At 212.038968   node 67 - Discovered the RADAR service on device 1 type: 1 in 0.038968 secs.
At 214.000000   node 38 - Started to discover the CAMERA service
At 214.000000   node 44 - Started to discover the TELNET service
At 214.009913   node 44 - Discovered the TELNET service on device 11 type: 1 in 0.009913 secs.
At 214.028856   node 38 - Discovered the CAMERA service on device 11 type: 1 in 0.028856 secs.
At 215.000000   node 43 - Started to discover the GPS service
At 215.003094   node 43 - Discovered the GPS service on device 12 type: 1 in 0.003094 secs.
At 216.000000   node 53 - Started to discover the GPS service
At 216.003613   node 53 - Discovered the GPS service on device 12 type: 1 in 0.003613 secs.
At 218.000000   node 40 - Started to discover the FAX service
At 218.000000   node 49 - Started to discover the SCANNER service
At 218.003153   node 49 - Discovered the SCANNER service on device 13 type: 1 in 0.003153 secs.
At 218.003654   node 40 - Discovered the FAX service on device 9 type: 1 in 0.003654 secs.
At 225.000000   node 91 - Started to discover the RADAR service
At 225.005736   node 91 - Discovered the RADAR service on device 4 type: 1 in 0.005736 secs.
At 226.000000   node 88 - Started to discover the RADAR service
At 226.048696   node 88 - Discovered the RADAR service on device 13 type: 1 in 0.048696 secs.
At 227.000000   node 85 - Started to discover the SCANNER service
At 227.003653   node 85 - Discovered the SCANNER service on device 4 type: 1 in 0.003653 secs.
At 234.000000   node 59 - Started to discover the WEB service
At 234.021936   node 59 - Discovered the WEB service on device 14 type: 1 in 0.021936 secs.
At 235.000000   node 84 - Started to discover the VOICE service
At 235.003193   node 84 - Discovered the VOICE service on device 5 type: 1 in 0.003193 secs.
At 236.000000   node 98 - Started to discover the SCANNER service
At 236.000000   node 41 - Started to discover the WEB service
At 236.003213   node 41 - Discovered the WEB service on device 11 type: 1 in 0.003213 secs.
At 236.005542   node 98 - Discovered the SCANNER service on device 4 type: 1 in 0.005542 secs.
At 239.000000   node 75 - Started to discover the WEB service
At 239.032401   node 75 - Discovered the WEB service on device 2 type: 1 in 0.032401 secs.
At 241.000000   node 65 - Started to discover the FAX service
At 241.000000   node 54 - Started to discover the CAMERA service
At 241.010883   node 54 - Discovered the CAMERA service on device 14 type: 1 in 0.010883 secs.
At 241.058375   node 65 - Discovered the FAX service on device 12 type: 1 in 0.058375 secs.
At 242.000000   node 88 - Started to discover the CAMERA service
At 242.038423   node 88 - Discovered the CAMERA service on device 5 type: 1 in 0.038423 secs.
At 245.000000   node 41 - Started to discover the FTP service
At 245.000000   node 83 - Started to discover the MP3 service
At 245.003253   node 41 - Discovered the FTP service on device 9 type: 1 in 0.003253 secs.
At 252.000000   node 58 - Started to discover the CAMERA service
At 252.000000   node 83 - service MP3 couldn't be discovered on any device
At 252.038354   node 58 - Discovered the CAMERA service on device 11 type: 1 in 0.038354 secs.
At 254.000000   node 30 - Started to discover the GPS service
At 254.003073   node 30 - Discovered the GPS service on device 15 type: 2 in 0.003073 secs.
At 257.000000   node 81 - Started to discover the TRANSLATOR service
At 257.003233   node 81 - Discovered the TRANSLATOR service on device 2 type: 1 in 0.003233 secs.
At 262.000000   node 75 - Started to discover the WEB service
At 262.002794   node 75 - Discovered the WEB service on device 2 type: 1 in 0.002794 secs.
At 263.000000   node 48 - Started to discover the TELNET service
At 263.000000   node 82 - Started to discover the WEB service
At 263.002993   node 48 - Discovered the TELNET service on device 11 type: 1 in 0.002993 secs.
At 263.039494   node 82 - Discovered the WEB service on device 2 type: 1 in 0.039494 secs.
At 266.000000   node 43 - Started to discover the RADAR service
At 266.000000   node 82 - Started to discover the CAMERA service
At 266.003434   node 43 - Discovered the RADAR service on device 10 type: 1 in 0.003434 secs.
At 266.032181   node 82 - Discovered the CAMERA service on device 2 type: 1 in 0.032181 secs.
At 269.000000   node 40 - Started to discover the PHONE service
At 269.034948   node 40 - Discovered the PHONE service on device 10 type: 1 in 0.034948 secs.
```

```
At 270.000000   node 56 - Started to discover the TELNET service
At 270.023875   node 56 - Discovered the TELNET service on device 14 type: 1 in 0.023875 secs.
At 273.000000   node 65 - Started to discover the CAMERA service
At 273.005237   node 65 - Discovered the CAMERA service on device 14 type: 1 in 0.005237 secs.
At 274.000000   node 66 - Started to discover the TELNET service
At 274.003233   node 66 - Discovered the TELNET service on device 2 type: 1 in 0.003233 secs.
At 275.000000   node 39 - Started to discover the WEB service
At 275.056431   node 39 - Discovered the WEB service on device 17 type: 2 in 0.056431 secs.
At 279.000000   node 81 - Started to discover the TRANSLATOR service
At 284.000000   node 64 - Started to discover the MAIL service
At 284.051196   node 64 - Discovered the MAIL service on device 12 type: 1 in 0.051196 secs.
At 285.000000   node 54 - Started to discover the TELNET service
At 285.000000   node 81 - service TRANSLATOR couldn't be discovered on any device
At 285.035073   node 54 - Discovered the TELNET service on device 11 type: 1 in 0.035073 secs.
At 287.000000   node 76 - Started to discover the WEB service
At 287.000000   node 89 - Started to discover the CAMERA service
At 287.002913   node 76 - Discovered the WEB service on device 2 type: 1 in 0.002913 secs.
At 287.002993   node 89 - Discovered the CAMERA service on device 5 type: 1 in 0.002993 secs.
At 292.000000   node 35 - Started to discover the TRANSLATOR service
At 292.003313   node 35 - Discovered the TRANSLATOR service on device 11 type: 1 in 0.003313 secs.
At 295.000000   node 58 - Started to discover the GPS service
At 295.003413   node 58 - Discovered the GPS service on device 12 type: 1 in 0.003413 secs.
At 297.000000   node 31 - Started to discover the PHONE service
At 297.002833   node 31 - Discovered the PHONE service on device 22 type: 2 in 0.002833 secs.
At 298.000000   node 84 - Started to discover the PRINTER service
At 298.002993   node 84 - Discovered the PRINTER service on device 13 type: 1 in 0.002993 secs.
At 299.000000   node 54 - Started to discover the SCANNER service
At 304.000000   node 46 - Started to discover the SCANNER service
At 304.003713   node 46 - Discovered the SCANNER service on device 10 type: 1 in 0.003713 secs.
At 305.000000   node 41 - Started to discover the SCANNER service
At 305.000000   node 57 - Started to discover the TELNET service
At 305.003294   node 41 - Discovered the SCANNER service on device 10 type: 1 in 0.003294 secs.
At 305.020522   node 57 - Discovered the TELNET service on device 14 type: 1 in 0.020522 secs.
At 306.000000   node 75 - Started to discover the PHONE service
At 306.000000   node 54 - service SCANNER couldn't be discovered on any device
At 306.033506   node 75 - Discovered the PHONE service on device 1 type: 1 in 0.033506 secs.
At 310.000000   node 73 - Started to discover the TELNET service
At 310.000000   node 73 - Started to discover the FTP service
At 310.007396   node 73 - Discovered the TELNET service on device 2 type: 1 in 0.007396 secs.
At 311.000000   node 87 - Started to discover the FTP service
At 312.000000   node 92 - Started to discover the WEB service
At 312.002753   node 92 - Discovered the WEB service on device 5 type: 1 in 0.002753 secs.
At 314.000000   node 55 - Started to discover the CAMERA service
At 314.036609   node 55 - Discovered the CAMERA service on device 14 type: 1 in 0.036609 secs.
At 315.000000   node 35 - Started to discover the MP3 service
At 315.006398   node 35 - Discovered the MP3 service on device 10 type: 1 in 0.006398 secs.
At 318.000000   node 74 - Started to discover the CAMERA service
At 318.000000   node 72 - Started to discover the CAMERA service
At 318.000000   node 87 - service FTP couldn't be discovered on any device
At 318.024635   node 74 - Discovered the CAMERA service on device 2 type: 1 in 0.024635 secs.
At 318.039459   node 72 - Discovered the CAMERA service on device 2 type: 1 in 0.039459 secs.
At 319.000000   node 51 - Started to discover the FAX service
At 319.000000   node 34 - Started to discover the RADAR service
At 319.000000   node 31 - Started to discover the FTP service
At 319.003473   node 31 - Discovered the FTP service on device 18 type: 2 in 0.003473 secs.
At 319.006562   node 51 - Discovered the FAX service on device 12 type: 1 in 0.006562 secs.
At 319.099568   node 34 - Discovered the RADAR service on device 10 type: 1 in 0.099568 secs.
At 321.000000   node 80 - Started to discover the SCANNER service
At 321.032376   node 80 - Discovered the SCANNER service on device 1 type: 1 in 0.032376 secs.
At 326.000000   node 76 - Started to discover the CAMERA service
```

```
At 326.002993   node 76 - Discovered the CAMERA service on device 2 type: 1 in 0.002993 secs.
At 329.000000   node 50 - Started to discover the GPS service
At 329.002793   node 50 - Discovered the GPS service on device 12 type: 1 in 0.002793 secs.
At 330.000000   node 49 - Started to discover the DISK service
At 330.000000   node 32 - Started to discover the DISK service
At 330.003114   node 49 - Discovered the DISK service on device 12 type: 1 in 0.003114 secs.
At 330.019190   node 32 - Discovered the DISK service on device 9 type: 1 in 0.019190 secs.
At 332.000000   node 73 - Started to discover the WEB service
At 332.003813   node 73 - Discovered the WEB service on device 2 type: 1 in 0.003813 secs.
At 336.000000   node 85 - Started to discover the TRANSLATOR service
At 336.003753   node 85 - Discovered the TRANSLATOR service on device 5 type: 1 in 0.003753 secs.
At 337.000000   node 79 - Started to discover the GPS service
At 337.000000   node 67 - Started to discover the WEB service
At 337.002953   node 67 - Discovered the WEB service on device 11 type: 1 in 0.002953 secs.
At 339.000000   node 55 - Started to discover the RADAR service
At 339.003133   node 55 - Discovered the RADAR service on device 13 type: 1 in 0.003133 secs.
At 340.000000   node 69 - Started to discover the CAMERA service
At 340.000000   node 98 - Started to discover the TRANSLATOR service
At 342.000000   node 56 - Started to discover the WEB service
At 342.046302   node 56 - Discovered the WEB service on device 8 type: 1 in 0.046302 secs.
At 343.000000   node 92 - Started to discover the PHONE service
At 343.000000   node 83 - Started to discover the RADAR service
At 343.053819   node 92 - Discovered the PHONE service on device 4 type: 1 in 0.053819 secs.
At 344.000000   node 97 - Started to discover the PHONE service
At 344.002993   node 97 - Discovered the PHONE service on device 4 type: 1 in 0.002993 secs.
At 345.000000   node 50 - Started to discover the WEB service
At 345.000000   node 79 - service GPS couldn't be discovered on any device
At 345.004568   node 50 - Discovered the WEB service on device 11 type: 1 in 0.004568 secs.
At 346.000000   node 99 - Started to discover the MP3 service
At 346.000000   node 95 - Started to discover the PRINTER service
At 346.003553   node 99 - Discovered the MP3 service on device 4 type: 1 in 0.003553 secs.
At 346.060378   node 95 - Discovered the PRINTER service on device 4 type: 1 in 0.060378 secs.
At 348.000000   node 33 - Started to discover the TELNET service
At 348.000000   node 69 - service CAMERA couldn't be discovered on any device
At 348.000000   node 98 - service TRANSLATOR couldn't be discovered on any device
At 348.012542   node 33 - Discovered the TELNET service on device 11 type: 1 in 0.012542 secs.
At 349.000000   node 79 - Started to discover the SCANNER service
At 349.003113   node 79 - Discovered the SCANNER service on device 1 type: 1 in 0.003113 secs.
At 351.000000   node 59 - Started to discover the MAIL service
At 351.000000   node 83 - service RADAR couldn't be discovered on any device
At 351.043222   node 59 - Discovered the MAIL service on device 12 type: 1 in 0.043222 secs.
At 353.000000   node 31 - Started to discover the DISK service
At 353.002813   node 31 - Discovered the DISK service on device 18 type: 2 in 0.002813 secs.
At 354.000000   node 62 - Started to discover the TELNET service
At 354.000000   node 49 - Started to discover the WEB service
At 360.000000   node 49 - service WEB couldn't be discovered on any device
At 360.000000   node 62 - service TELNET couldn't be discovered on any device
At 361.000000   node 58 - Started to discover the TELNET service
At 361.000000   node 87 - Started to discover the FTP service
At 361.002894   node 58 - Discovered the TELNET service on device 11 type: 1 in 0.002894 secs.
At 361.028220   node 87 - Discovered the FTP service on device 6 type: 1 in 0.028220 secs.
At 363.000000   node 69 - Started to discover the SCANNER service
At 363.030051   node 69 - Discovered the SCANNER service on device 1 type: 1 in 0.030051 secs.
At 364.000000   node 76 - Started to discover the GPS service
At 364.030472   node 76 - Discovered the GPS service on device 3 type: 1 in 0.030472 secs.
At 365.000000   node 90 - Started to discover the CAMERA service
At 365.013350   node 90 - Discovered the CAMERA service on device 5 type: 1 in 0.013350 secs.
At 368.000000   node 49 - Started to discover the TRANSLATOR service
At 368.000000   node 85 - Started to discover the WEB service
At 368.000000   node 81 - Started to discover the MAIL service
```

```
At 368.003033   node 85 - Discovered the WEB service on device 5 type: 1 in 0.003033 secs.
At 368.021622   node 49 - Discovered the TRANSLATOR service on device 11 type: 1 in 0.021622 secs.
At 368.042413   node 81 - Discovered the MAIL service on device 3 type: 1 in 0.042413 secs.
At 370.000000   node 98 - Started to discover the WEB service
At 370.000000   node 42 - Started to discover the PHONE service
At 370.000000   node 71 - Started to discover the MP3 service
At 370.002953   node 98 - Discovered the WEB service on device 5 type: 1 in 0.002953 secs.
At 370.003193   node 42 - Discovered the PHONE service on device 10 type: 1 in 0.003193 secs.
At 370.031718   node 71 - Discovered the MP3 service on device 1 type: 1 in 0.031718 secs.
At 371.000000   node 46 - Started to discover the SCANNER service
At 371.000000   node 64 - Started to discover the FTP service
At 371.003774   node 46 - Discovered the SCANNER service on device 10 type: 1 in 0.003774 secs.
At 371.080590   node 64 - Discovered the FTP service on device 12 type: 1 in 0.080590 secs.
At 374.000000   node 58 - Started to discover the MAIL service
At 374.000000   node 52 - Started to discover the FAX service
At 374.032589   node 52 - Discovered the FAX service on device 12 type: 1 in 0.032589 secs.
At 377.000000   node 57 - Started to discover the RADAR service
At 377.026467   node 57 - Discovered the RADAR service on device 13 type: 1 in 0.026467 secs.
At 381.000000   node 77 - Started to discover the RADAR service
At 381.000000   node 58 - service MAIL couldn't be discovered on any device
At 381.003713   node 77 - Discovered the RADAR service on device 1 type: 1 in 0.003713 secs.
At 382.000000   node 53 - Started to discover the PHONE service
At 382.000000   node 78 - Started to discover the WEB service
At 382.003253   node 78 - Discovered the WEB service on device 2 type: 1 in 0.003253 secs.
At 382.034269   node 53 - Discovered the PHONE service on device 13 type: 1 in 0.034269 secs.
At 384.000000   node 92 - Started to discover the PRINTER service
At 390.000000   node 75 - Started to discover the CAMERA service
At 390.000000   node 92 - service PRINTER couldn't be discovered on any device
At 390.003473   node 75 - Discovered the CAMERA service on device 2 type: 1 in 0.003473 secs.
At 391.000000   node 67 - Started to discover the SCANNER service
At 391.006991   node 67 - Discovered the SCANNER service on device 1 type: 1 in 0.006991 secs.
At 392.000000   node 34 - Started to discover the SCANNER service
At 392.000000   node 36 - Started to discover the PHONE service
At 392.015940   node 34 - Discovered the SCANNER service on device 10 type: 1 in 0.015940 secs.
At 392.026826   node 36 - Discovered the PHONE service on device 10 type: 1 in 0.026826 secs.
At 394.000000   node 52 - Started to discover the WEB service
At 394.003173   node 52 - Discovered the WEB service on device 14 type: 1 in 0.003173 secs.
At 395.000000   node 76 - Started to discover the TELNET service
At 397.000000   node 92 - Started to discover the SCANNER service
At 397.007277   node 92 - Discovered the SCANNER service on device 4 type: 1 in 0.007277 secs.
At 402.000000   node 76 - service TELNET couldn't be discovered on any device
At 403.000000   node 65 - Started to discover the WEB service
At 403.000000   node 58 - Started to discover the TELNET service
At 403.052946   node 58 - Discovered the TELNET service on device 5 type: 1 in 0.052946 secs.
At 403.070133   node 65 - Discovered the WEB service on device 14 type: 1 in 0.070133 secs.
At 404.000000   node 34 - Started to discover the FAX service
At 404.072100   node 34 - Discovered the FAX service on device 0 type: 1 in 0.072100 secs.
At 407.000000   node 39 - Started to discover the DISK service
At 407.020033   node 39 - Discovered the DISK service on device 12 type: 1 in 0.020033 secs.
At 408.000000   node 53 - Started to discover the PRINTER service
At 408.000000   node 87 - Started to discover the SCANNER service
At 412.000000   node 91 - Started to discover the FAX service
At 412.003373   node 91 - Discovered the FAX service on device 6 type: 1 in 0.003373 secs.
At 414.000000   node 53 - service PRINTER couldn't be discovered on any device
At 414.000000   node 87 - service SCANNER couldn't be discovered on any device
At 415.000000   node 94 - Started to discover the FAX service
At 415.000000   node 57 - Started to discover the CAMERA service
At 415.003173   node 94 - Discovered the FAX service on device 6 type: 1 in 0.003173 secs.
At 415.036889   node 57 - Discovered the CAMERA service on device 14 type: 1 in 0.036889 secs.
At 416.000000   node 53 - Started to discover the SCANNER service
```

```
At 418.000000   node 47 - Started to discover the DISK service
At 418.003073   node 47 - Discovered the DISK service on device 9 type: 1 in 0.003073 secs.
At 419.000000   node 76 - Started to discover the DISK service
At 419.009109   node 76 - Discovered the DISK service on device 3 type: 1 in 0.009109 secs.
At 423.000000   node 53 - service SCANNER couldn't be discovered on any device
At 426.000000   node 47 - Started to discover the TRANSLATOR service
At 426.011411   node 47 - Discovered the TRANSLATOR service on device 11 type: 1 in 0.011411 secs.
At 431.000000   node 43 - Started to discover the FAX service
At 431.048151   node 43 - Discovered the FAX service on device 12 type: 1 in 0.048151 secs.
At 434.000000   node 54 - Started to discover the FAX service
At 434.024458   node 54 - Discovered the FAX service on device 12 type: 1 in 0.024458 secs.
At 435.000000   node 45 - Started to discover the FTP service
At 435.037558   node 45 - Discovered the FTP service on device 9 type: 1 in 0.037558 secs.
At 437.000000   node 36 - Started to discover the MAIL service
At 437.003033   node 36 - Discovered the MAIL service on device 9 type: 1 in 0.003033 secs.
At 441.000000   node 57 - Started to discover the FAX service
At 441.003354   node 57 - Discovered the FAX service on device 12 type: 1 in 0.003354 secs.
At 445.000000   node 56 - Started to discover the MAIL service
At 445.000000   node 85 - Started to discover the FAX service
At 445.006406   node 56 - Discovered the MAIL service on device 12 type: 1 in 0.006406 secs.
At 445.013252   node 85 - Discovered the FAX service on device 6 type: 1 in 0.013252 secs.
At 446.000000   node 94 - Started to discover the VOICE service
At 446.035200   node 94 - Discovered the VOICE service on device 5 type: 1 in 0.035200 secs.
At 448.000000   node 77 - Started to discover the SCANNER service
At 448.000000   node 38 - Started to discover the MP3 service
At 448.003254   node 77 - Discovered the SCANNER service on device 1 type: 1 in 0.003254 secs.
At 448.005355   node 38 - Discovered the MP3 service on device 10 type: 1 in 0.005355 secs.
At 449.000000   node 35 - Started to discover the TRANSLATOR service
At 449.006334   node 35 - Discovered the TRANSLATOR service on device 11 type: 1 in 0.006334 secs.
At 454.000000   node 34 - Started to discover the WEB service
At 454.002913   node 34 - Discovered the WEB service on device 11 type: 1 in 0.002913 secs.
At 456.000000   node 47 - Started to discover the DISK service
At 459.000000   node 35 - Started to discover the TELNET service
At 459.000000   node 95 - Started to discover the FAX service
At 459.003313   node 35 - Discovered the TELNET service on device 11 type: 1 in 0.003313 secs.
At 459.035005   node 95 - Discovered the FAX service on device 6 type: 1 in 0.035005 secs.
At 460.000000   node 59 - Started to discover the TELNET service
At 460.000000   node 93 - Started to discover the MP3 service
At 460.000000   node 68 - Started to discover the FTP service
At 460.000000   node 75 - Started to discover the PHONE service
At 460.003448   node 93 - Discovered the MP3 service on device 4 type: 1 in 0.003448 secs.
At 460.016996   node 59 - Discovered the TELNET service on device 14 type: 1 in 0.016996 secs.
At 462.000000   node 47 - service DISK couldn't be discovered on any device
At 464.000000   node 40 - Started to discover the RADAR service
At 468.000000   node 69 - Started to discover the MAIL service
At 468.000000   node 68 - service FTP couldn't be discovered on any device
At 468.000000   node 75 - service PHONE couldn't be discovered on any device
At 468.032976   node 69 - Discovered the MAIL service on device 3 type: 1 in 0.032976 secs.
At 469.000000   node 61 - Started to discover the DISK service
At 469.000000   node 58 - Started to discover the MAIL service
At 469.083432   node 61 - Discovered the DISK service on device 6 type: 1 in 0.083432 secs.
At 469.085146   node 58 - Discovered the MAIL service on device 6 type: 1 in 0.085146 secs.
At 471.000000   node 40 - service RADAR couldn't be discovered on any device
At 475.000000   node 62 - Started to discover the TRANSLATOR service
At 475.003593   node 62 - Discovered the TRANSLATOR service on device 14 type: 1 in 0.003593 secs.
At 476.000000   node 96 - Started to discover the TELNET service
At 476.000000   node 59 - Started to discover the PHONE service
At 476.003133   node 96 - Discovered the TELNET service on device 5 type: 1 in 0.003133 secs.
At 478.000000   node 53 - Started to discover the SCANNER service
At 478.003173   node 53 - Discovered the SCANNER service on device 13 type: 1 in 0.003173 secs.
```

```
At 479.000000   node 33 - Started to discover the MP3 service
At 479.003173   node 33 - Discovered the MP3 service on device 10 type: 1 in 0.003173 secs.
At 482.000000   node 38 - Started to discover the PRINTER service
At 483.000000   node 59 - service PHONE couldn't be discovered on any device
At 485.000000   node 42 - Started to discover the VOICE service
At 485.003394   node 42 - Discovered the VOICE service on device 8 type: 1 in 0.003394 secs.
At 486.000000   node 60 - Started to discover the RADAR service
At 486.003133   node 60 - Discovered the RADAR service on device 13 type: 1 in 0.003133 secs.
At 487.000000   node 79 - Started to discover the PRINTER service
At 487.031492   node 79 - Discovered the PRINTER service on device 1 type: 1 in 0.031492 secs.
At 489.000000   node 38 - service PRINTER couldn't be discovered on any device
At 490.000000   node 55 - Started to discover the MAIL service
At 490.062439   node 55 - Discovered the MAIL service on device 3 type: 1 in 0.062439 secs.
At 492.000000   node 61 - Started to discover the WEB service
At 492.003153   node 61 - Discovered the WEB service on device 14 type: 1 in 0.003153 secs.
At 504.000000   node 63 - Started to discover the MAIL service
At 504.003034   node 63 - Discovered the MAIL service on device 12 type: 1 in 0.003034 secs.
At 505.000000   node 36 - Started to discover the TELNET service
At 505.003513   node 36 - Discovered the TELNET service on device 11 type: 1 in 0.003513 secs.
At 506.000000   node 42 - Started to discover the DISK service
At 506.003514   node 42 - Discovered the DISK service on device 9 type: 1 in 0.003514 secs.
At 507.000000   node 68 - Started to discover the RADAR service
At 507.003473   node 68 - Discovered the RADAR service on device 1 type: 1 in 0.003473 secs.
At 513.000000   node 80 - Started to discover the TELNET service
At 513.004744   node 80 - Discovered the TELNET service on device 2 type: 1 in 0.004744 secs.
At 523.000000   node 93 - Started to discover the RADAR service
At 524.000000   node 67 - Started to discover the RADAR service
At 524.038944   node 67 - Discovered the RADAR service on device 1 type: 1 in 0.038944 secs.
At 525.000000   node 80 - Started to discover the MAIL service
At 525.034113   node 80 - Discovered the MAIL service on device 3 type: 1 in 0.034113 secs.
At 527.000000   node 30 - Started to discover the FTP service
At 527.002853   node 30 - Discovered the FTP service on device 27 type: 2 in 0.002853 secs.
At 528.000000   node 59 - Started to discover the MAIL service
At 528.029580   node 59 - Discovered the MAIL service on device 12 type: 1 in 0.029580 secs.
At 529.000000   node 34 - Started to discover the CAMERA service
At 531.000000   node 93 - service RADAR couldn't be discovered on any device
At 533.000000   node 82 - Started to discover the MAIL service
At 533.003273   node 82 - Discovered the MAIL service on device 3 type: 1 in 0.003273 secs.
At 534.000000   node 84 - Started to discover the SCANNER service
At 534.008604   node 84 - Discovered the SCANNER service on device 4 type: 1 in 0.008604 secs.
At 537.000000   node 34 - service CAMERA couldn't be discovered on any device
At 542.000000   node 97 - Started to discover the CAMERA service
At 542.039755   node 97 - Discovered the CAMERA service on device 5 type: 1 in 0.039755 secs.
At 544.000000   node 43 - Started to discover the SCANNER service
At 545.000000   node 59 - Started to discover the MP3 service
At 545.002853   node 59 - Discovered the MP3 service on device 13 type: 1 in 0.002853 secs.
At 546.000000   node 47 - Started to discover the TRANSLATOR service
At 546.003033   node 47 - Discovered the TRANSLATOR service on device 8 type: 1 in 0.003033 secs.
At 547.000000   node 59 - Started to discover the MAIL service
At 547.030890   node 59 - Discovered the MAIL service on device 12 type: 1 in 0.030890 secs.
At 549.000000   node 83 - Started to discover the TRANSLATOR service
At 549.000000   node 46 - Started to discover the SCANNER service
At 549.020024   node 46 - Discovered the SCANNER service on device 10 type: 1 in 0.020024 secs.
At 549.023758   node 83 - Discovered the TRANSLATOR service on device 5 type: 1 in 0.023758 secs.
At 552.000000   node 43 - service SCANNER couldn't be discovered on any device
At 559.000000   node 73 - Started to discover the PHONE service
At 559.033760   node 73 - Discovered the PHONE service on device 1 type: 1 in 0.033760 secs.
At 561.000000   node 43 - Started to discover the MP3 service
At 561.000000   node 75 - Started to discover the FAX service
At 561.003293   node 43 - Discovered the MP3 service on device 10 type: 1 in 0.003293 secs.
```

```
At 561.015861   node 75 - Discovered the FAX service on device 3 type: 1 in 0.015861 secs.
At 568.000000   node 95 - Started to discover the WEB service
At 570.000000   node 70 - Started to discover the FAX service
At 570.003373   node 70 - Discovered the FAX service on device 3 type: 1 in 0.003373 secs.
At 571.000000   node 79 - Started to discover the SCANNER service
At 571.005080   node 79 - Discovered the SCANNER service on device 1 type: 1 in 0.005080 secs.
At 572.000000   node 57 - Started to discover the TELNET service
At 572.000000   node 97 - Started to discover the RADAR service
At 574.000000   node 49 - Started to discover the FAX service
At 574.000000   node 96 - Started to discover the WEB service
At 574.000000   node 75 - Started to discover the VOICE service
At 574.002853   node 49 - Discovered the FAX service on device 12 type: 1 in 0.002853 secs.
At 574.003393   node 96 - Discovered the WEB service on device 5 type: 1 in 0.003393 secs.
At 574.003454   node 75 - Discovered the VOICE service on device 2 type: 1 in 0.003454 secs.
At 576.000000   node 95 - service WEB couldn't be discovered on any device
At 577.000000   node 79 - Started to discover the FTP service
At 577.003094   node 79 - Discovered the FTP service on device 3 type: 1 in 0.003094 secs.
At 578.000000   node 91 - Started to discover the FAX service
At 578.000000   node 77 - Started to discover the PHONE service
At 578.003553   node 77 - Discovered the PHONE service on device 1 type: 1 in 0.003553 secs.
At 578.051556   node 91 - Discovered the FAX service on device 6 type: 1 in 0.051556 secs.
At 579.000000   node 57 - service TELNET couldn't be discovered on any device
At 579.000000   node 97 - service RADAR couldn't be discovered on any device
At 580.000000   node 89 - Started to discover the SCANNER service
At 580.048120   node 89 - Discovered the SCANNER service on device 4 type: 1 in 0.048120 secs.
At 583.000000   node 38 - Started to discover the TELNET service
At 583.032276   node 38 - Discovered the TELNET service on device 11 type: 1 in 0.032276 secs.
At 584.000000   node 58 - Started to discover the RADAR service
At 584.003113   node 58 - Discovered the RADAR service on device 13 type: 1 in 0.003113 secs.
At 585.000000   node 60 - Started to discover the PHONE service
At 591.000000   node 45 - Started to discover the PRINTER service
At 591.000000   node 60 - service PHONE couldn't be discovered on any device
At 591.002913   node 45 - Discovered the PRINTER service on device 10 type: 1 in 0.002913 secs.
At 594.000000   node 41 - Started to discover the TELNET service
At 594.007770   node 41 - Discovered the TELNET service on device 11 type: 1 in 0.007770 secs.
At 596.000000   node 67 - Started to discover the FAX service
At 597.000000   node 56 - Started to discover the PRINTER service
At 598.000000   node 58 - Started to discover the FTP service
At 598.003053   node 58 - Discovered the FTP service on device 12 type: 1 in 0.003053 secs.
At 600.000000   node 62 - Started to discover the SCANNER service
NS EXITING...
```

Apéndice C

Código en AWK

A continuación añadimos al presente documento, el código escrito en el lenguaje AWK que hemos usado para obtener los resultados descritos en el capítulo 5 y mostrados en el apéndice D. Este código computa los valores que nos interesa filtrar a partir de las trazas del archivo *.tr* generado por el simulador Ns-2. El archivo *.tr* se genera simultáneamente a la salida mostrada en el apéndice B.

Este código se encarga de procesar mediante reconocimiento de patrones y manejo de expresiones regulares, los eventos que consideramos nos puedan ser de utilidad para nuestro análisis.

```
# ------------------------------------------------------------------------
# awk -f metricasLIFT.awk
# Programa en awk que resume información de archivos .tr de Ns-2
# November 23 2007. Mike Wister
# ------------------------------------------------------------------------

BEGIN {
   highest_packet_id = 0; counter1= 0; counter2=0;
   highest_flow_id = 0; energia = 0;
}
# ------------------------------------------------------------------------
# --------- Packet Delivery Ratio ----------------------------------------
# ------------------------------------------------------------------------
//                {total++ }
/AGT/||/RTR/||/MAC/|| /IFQ/ || /ARP/ {totalNL++}  # total packets (Network trace Level)

/RTR/||/MAC/     {energia += $17}  # total packets (Network trace Level)

/RTR/                    {rtr++}      # total packets RTR
/MAC/                    {mac++}      # total packets MAC
/^s/               {sent++}     # total de paquetes enviados
/^s/ && /RTR/  && /-It AODV/  {rtr_sent++} # paquetes enviados tipo RTR
```

```
/^r/               {rec++}      # total de paquetes recibidos
/^r/ && /RTR/                 {rtr_rec++}  # paquetes recibidos tipo RTR
/RTR/ && /-It AODV/           {data_rtr_sent++} # total paquetes de datos
/^r/ && /RTR/ && /-It AODV/   {data_rtr_rec++} # paquetes de datos recibidos
/^f/ && /RTR/ && /-It AODV/   {data_rtr_forw++} # paquetes de datos reenviados
/^d/ && /MAC/ && /-It AODV/   {data_rtr_drop++} # paquetes de datos caidos
/^d/         {drop++}        # total de paquetes caidos (dropped)
/^d/ && /RTR/           {rtr_drop++}    # paquetes caidos tipo RTR
/^d/ && /IFQ/           {ifq_drop++}    # ROP_IFQ_QFULL i.e no buffer space in IFQ.
/^f/                    {forw++}        # total de paquetes reenviados
/^f/ && /RTR/           {rtr_forw++}    # paquetes reenviados tipo RTR
/^s/ && /RTR/ && /-It AODV/   {cp_sent++}  # Paqs. de ctrl enviados
/^r/ && /RTR/ && /-It AODV/   {cp_rec++}  # Paqs. de ctrl recibidos
/^f/ && /RTR/ && /-It AODV/   {cp_forw++}  # Paqs. de ctrl reenviado
/^d/ && /RTR/ && /-It AODV/   {cp_drop++}  # Paqs. de ctrl caídos
/^s/ && /RTR/ && /-It AODV/ && /-Pc REQUEST/ {rreq++}  # Route Requests sent -Pq
/^s/ && /RTR/ && /-It AODV/ && /-Pc REPLY/   {rrep++}  # Route Reply sent    -Pp
/^s/ && /RTR/ && /-It AODV/ && /-Pc 1/       {rerr++}  # Route Error sent    -Pw
/^r/ && /RTR/ && /-It AODV/ && /-Pc REQUEST/ {rreqrec++} # Route Requests sent -Pq
/^r/ && /RTR/ && /-It AODV/ && /-Pc REPLY/   {rreprec++} # Route Reply sent    -Pp
/^r/ && /RTR/ && /13a/ && /REPLY/ && /800/ {found_RTR++} # Servs. encontrados -Ma 0
/^r/ && /MAC/ && /13a/ && /REPLY/ && /800/ {found_MAC++} # Servs. encontrados -Ma 13a
/^s/ && /RTR/ && /-Ma 0/ && /REPLY/ {found_RTR_s++}      # Servs. encontrados -Ma 0
/^s/ && /MAC/ && /13a/ && /REPLY/   {found_MAC_s++}      # Servs. encontrados -Ma 13a
/^f/ && /RTR/ && /13a/ && /REPLY/   {found3++}          # Servs. encontrados -Ma 13a
$1~/s/ && /MAC/  {counter1++ }
$1~/r/ && /RTR/  {counter2++ }

# ------------------------------------------------------------------
# --------- Throughput analyse -------------------------------------
# ------------------------------------------------------------------
{  action = $1; # $1
   time = $3; # $3
   packet_size = $37; # $37
   packet_id = $47; # $47
   flow_id = $41; # $41

   if ( packet_id > highest_packet_id ) highest_packet_id = packet_id;
   if ( flow_id > highest_flow_id ) highest_flow_id = flow_id;

   if ( ($19 == "MAC") && (start_time[flow_id] == 0) ){    # $19
     start_time[flow_id] = time;
     size[flow_id] = packet_size;
   }

   if ( ($1 == "r") && ($19 == "MAC")) {
         end_time[flow_id] = time;
   }
   if ($1 == "d")
      end_time[flow_id] = -1;
}
END {

# ------------------------------------------------------------------------
# --------- Packet Delivery Ratio ----------------------------------------
# ------------------------------------------------------------------------

 cp_sent = rreq + rrep + rerr;      # Controll Packets sent
 cp_lost = cp_sent - cp_rec;        # Controll Packets lost
 if (data_rtr_sent > 0)
```

```
    PDR = (data_rtr_rec/data_rtr_sent)*100;  # PDR in percentage
 if (data_rtr_rec > 0)
    nrl = ((cp_sent + cp_forw)/data_rtr_rec)*100;       # NRL
 lost = sent - rec;                 # Packets lost

 data_rtr_lost = data_rtr_sent - data_rtr_rec;  # Data Packets lost
 rtr_lost = rtr_sent - rtr_rec;      # Packets lost by Agent
 rtr_lost = rtr_lost + cp_lost;      # Packets lost by Devices

# ---------------------------------------------------------------------
# --------- Throughput ------------------------------------------------
# ---------------------------------------------------------------------
for ( flow_id = 0; flow_id <= highest_flow_id; flow_id++ ) {
   start = start_time[flow_id];
   end = end_time[flow_id];
   if ( start < end ){
     packet_duration = end - start;      # single distance
     duration_total += packet_duration;  # total duration
     flow_number++;                  # flow number
     bits_total += $37;                  # Bits Total
    }
}
 thrgputPackets = flow_number / duration_total;     # End-to-End Delay
 thrgputBits    = bits_total / duration_total;      # Throughput
 system(clear);
 printf ("\t------------------- ANALISIS DE DATOS ----------------------\n");
 printf ("\tPacket Delivery Ratio (PDR) :  %3.2f\n", PDR);       # Tasa de exitos
 printf ("\tThroughput                  :  %f\n",    thrgputBits);    # Avg duration of bits/sec.
 printf ("\tAverage Hop Counts          :  %2d\n",   counter1/counter2 * 100); # Path length
 printf ("\tEnd-to-End Delay            :  %f\n",    thrgputPackets); # Avg duration of packets/sec
 printf ("\tRouting Overhead (NRL)      :  %3.2f\n", nrl);            # Porcentaje
 printf ("\tControl Message Overhead    :  %2d\n",   cp_sent);
 printf ("\tEnergia promedio : %f gasto :  %f\n", energia/totalNL, 20- energia/totalNL);
 printf ("\t-------------------------------------------------------------\n\n\n");
}
```

Apéndice D

Resumen de los resultados de las simulaciones

En este apéndice incluimos un resumen de los resultados de la serie de simulaciones. Estas salidas son obtenidas a partir de los distintos parámetros utilizados para medir el rendimiento de LIFT. Estos resultados los produce el código incluido en el apéndice C. Dichos resultados son extraídos de experimentos que van desde 100 nodos hasta 500 nodos y con velocidades de movilidad desde 2 Km/h hasta 20 Km/h.

```
Resultados de la simulación para conocer el efecto
respecto a la movilidad

Comienza el análisis de 100 nodos --------

Analizando 100 nodos 2 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  87.73
        Throughput                  :  0.775498
        Average Hop Counts          :  23
        Routing Overhead (NRL)      :  12.05
        Control Message Overhead    :  9681
        Energía promedio : 17.448617 gasto :  2.551383
        ---------------------------------------------------

Analizando 100 nodos 5 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  87.28
        Throughput                  :  0.769913
        Average Hop Counts          :  24
        Routing Overhead (NRL)      :  12.54
        Control Message Overhead    :  10088
        Energía promedio : 17.350055 gasto :  2.649945
        ---------------------------------------------------
```

```
Analizando 100 nodos 10 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  87.12
        Throughput                  :  0.768047
        Average Hop Counts          :  25
        Routing Overhead (NRL)      :  12.62
        Control Message Overhead    :  10054
        Energía promedio : 17.270213 gasto :  2.729787
        --------------------------------------------------

Analizando 100 nodos 15 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  87.40
        Throughput                  :  0.796448
        Average Hop Counts          :  24
        Routing Overhead (NRL)      :  12.40
        Control Message Overhead    :  9867
        Energía promedio : 17.380553 gasto :  2.619447
        --------------------------------------------------

Analizando 100 nodos 20 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  86.82
        Throughput                  :  0.752830
        Average Hop Counts          :  25
        Routing Overhead (NRL)      :  12.93
        Control Message Overhead    :  10054
        Energía promedio : 17.280119 gasto :  2.719881
        --------------------------------------------------

Comienza el análisis de 200 nodos --------

Analizando 200 nodos 2 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  90.46
        Throughput                  :  0.765356
        Average Hop Counts          :  15
        Routing Overhead (NRL)      :  8.15
        Control Message Overhead    :  25623
        Energía promedio : 14.546276 gasto :  5.453724
        --------------------------------------------------

Analizando 200 nodos 5 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  90.58
        Throughput                  :  0.726578
        Average Hop Counts          :  15
        Routing Overhead (NRL)      :  8.14
        Control Message Overhead    :  26527
        Energía promedio : 14.340491 gasto :  5.659509
        --------------------------------------------------

Analizando 200 nodos 10 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  90.27
        Throughput                  :  0.705749
        Average Hop Counts          :  15
        Routing Overhead (NRL)      :  8.33
        Control Message Overhead    :  26582
```

```
        Energía promedio : 14.345265 gasto :  5.654735
        ------------------------------------------------

Analizando 200 nodos 15 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  89.99
        Throughput                  :  0.693586
        Average Hop Counts          :  16
        Routing Overhead (NRL)      :  8.43
        Control Message Overhead    :  26560
        Energía promedio : 14.252559 gasto :  5.747441
        ------------------------------------------------

Analizando 200 nodos 20 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  90.08
        Throughput                  :  0.690758
        Average Hop Counts          :  16
        Routing Overhead (NRL)      :  8.43
        Control Message Overhead    :  26985
        Energía promedio : 14.186436 gasto :  5.813564
        ------------------------------------------------

Comienza el análisis de 300 nodos --------

Analizando 300 nodos 2 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  91.55
        Throughput                  :  0.803303
        Average Hop Counts          :  13
        Routing Overhead (NRL)      :  6.84
        Control Message Overhead    :  37876
        Energía promedio : 12.660046 gasto :  7.339954
        ------------------------------------------------

Analizando 300 nodos 5 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  91.62
        Throughput                  :  0.791573
        Average Hop Counts          :  13
        Routing Overhead (NRL)      :  6.78
        Control Message Overhead    :  37806
        Energía promedio : 12.694697 gasto :  7.305303
        ------------------------------------------------

Analizando 300 nodos 10 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  91.69
        Throughput                  :  0.780042
        Average Hop Counts          :  13
        Routing Overhead (NRL)      :  6.83
        Control Message Overhead    :  39078
        Energía promedio : 12.615607 gasto :  7.384393
        ------------------------------------------------

Analizando 300 nodos 15 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  91.75
        Throughput                  :  0.764342
        Average Hop Counts          :  12
```

```
        Routing Overhead (NRL)      :  6.78
        Control Message Overhead    :  39523
        Energía promedio : 12.488657 gasto :  7.511343
        ------------------------------------------------

Analizando 300 nodos 20 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  91.59
        Throughput                  :  0.748588
        Average Hop Counts          :  13
        Routing Overhead (NRL)      :  6.78
        Control Message Overhead    :  38888
        Energía promedio : 12.544560 gasto :  7.455440
        ------------------------------------------------

Comienza el análisis de 400 nodos --------

Analizando 400 nodos 2 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.51
        Throughput                  :  0.779127
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.97
        Control Message Overhead    :  46101
        Energía promedio : 12.484075 gasto :  7.515925
        ------------------------------------------------

Analizando 400 nodos 5 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.63
        Throughput                  :  0.772164
        Average Hop Counts          :  10
        Routing Overhead (NRL)      :  5.91
        Control Message Overhead    :  47697
        Energía promedio : 12.428257 gasto :  7.571743
        ------------------------------------------------

Analizando 400 nodos 10 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.54
        Throughput                  :  0.780359
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.95
        Control Message Overhead    :  47343
        Energía promedio : 12.289614 gasto :  7.710386
        ------------------------------------------------

Analizando 400 nodos 15 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.42
        Throughput                  :  0.727718
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.98
        Control Message Overhead    :  47060
        Energía promedio : 12.266473 gasto :  7.733527
        ------------------------------------------------

Analizando 400 nodos 20 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.40
```

```
        Throughput                  :  0.738163
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  6.03
        Control Message Overhead    :  47377
        Energía promedio : 12.306537 gasto :  7.693463
        -------------------------------------------------

Comienza el análisis de 500 nodos --------

Analizando 500 nodos 2 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.70
        Throughput                  :  0.795814
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.72
        Control Message Overhead    :  55115
        Energía promedio : 11.804068 gasto :  8.195932
        -------------------------------------------------

Analizando 500 nodos 5 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.72
        Throughput                  :  0.788995
        Average Hop Counts          :  10
        Routing Overhead (NRL)      :  5.67
        Control Message Overhead    :  55950
        Energía promedio : 11.754565 gasto :  8.245435
        -------------------------------------------------

Analizando 500 nodos 10 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.62
        Throughput                  :  0.783387
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.76
        Control Message Overhead    :  54718
        Energía promedio : 11.802030 gasto :  8.197970
        -------------------------------------------------

Analizando 500 nodos 15 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.64
        Throughput                  :  0.787950
        Average Hop Counts          :  10
        Routing Overhead (NRL)      :  5.68
        Control Message Overhead    :  55110
        Energía promedio : 11.730261 gasto :  8.269739
        -------------------------------------------------

Analizando 500 nodos 20 Acc . . . .
        ------------------- ANALISIS DE DATOS -------------
        Packet Delivery Ratio (PDR) :  92.46
        Throughput                  :  0.777279
        Average Hop Counts          :  11
        Routing Overhead (NRL)      :  5.78
        Control Message Overhead    :  54722
        Energía promedio : 11.811619 gasto :  8.188381
        -------------------------------------------------
```

Referencias

[1] Alliance OSGi. About the osgi service platform technical whitepaper.

[2] Amis, A. D., Prakash, R., Huynh, D., and Vuong, T. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2000*.

[3] Artail, H., Safa, H., Hamze, H., and Mershad, K. A Cluster Based Service Discovery Model for Mobile Ad Hoc Networks. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)* (October 2007), pp. 57–64.

[4] Baker, D., and Ephremides, A. A distributed algorithm for organizing mobile radio telecommunication networks. In *Proceedings of the 2nd International Conference on Distributed Computer Systems* (April 1981), IEEE Transactions on Communications, pp. 476–483.

[5] Baker, D., and Ephremides, A. The architectural organization of a mobile radio network via a distributed algorithm. In *Proceedings of the 2nd International Conference on Distributed Computer Systems* (November 1981), IEEE Transactions on Communications, pp. 1694–1701.

[6] Balazinska, M., Balakrishnan, H., and Karger, D. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *Proceedings of the First International Conference on Pervasive Computing* (London, UK, 2002), Springer-Verlag, pp. 195–210.

[7] Bao, L., and Garcia-Luna-Aceves, J. J. Topology Management in Ad Hoc Networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 03)* (New York, NY, USA, 2003), ACM, pp. 129–140.

[8] Basagni, S. Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks. In *Proceedings of Vehicular*

Technology Conference, VTC 1999-Fall (Washington, DC, USA, 1999), vol. 2, IEEE Computer Society, pp. 889–893.

[9] Basagni, S. Distributed Clustering for Ad Hoc Networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99)* (Washington, DC, USA, 1999), IEEE Computer Society, pp. 310–315.

[10] Basu, P., Khan, N., and Little, T. D. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems. (ICDCSW '01)* (Washington, DC, USA, April 2001), IEEE Computer Society, pp. 413–432.

[11] Berger, M., Watzke, M., and Helin, H. Towards a FIPA Approach for Mobile Ad hoc Environments. In *Siemens AG, Corporate Technology, Intelligent Autonomous Systems Group, Munich, GERMANY, Sonera Corporation, Emerging Technologies and Innovations, Sonera, FINLAND.*

[12] Blanco, O. A. Evaluación de una red de sensores con protocolo AODV y tecnología radio IEEE 802.15.4. Trabajo final de carrera, Escola Politecnica Superior de Castelldefels, Universitat Politecnica de Catalunya, Junio 2005.

[13] Bluetooth. The official bluetooth website.

[14] Boyaci, C., and Jung, E. Combined Approach for Service and Route Discovery. In *Department of Computer and Information Science and Engineering, University of Florida*, Gainesville, FL, USA.

[15] Bromberg, Y.-D., and Issarny, V. INDISS: Interoperable Discovery System for Networked Services. In *INRIA-Rocquencourt.*

[16] Bromberg, Y.-D., and Issarny, V. Service Discovery Protocol Interoperability in the Mobile Environment. In *INRIA-Rocquencourt. Domaine de Voluceau, 78153 Le Chesnay, France.*

[17] Camp, T., Boleng, J., and Davies, V. A Survey of Mobility Models for Ad Hoc Network Research. In *Wireless Communications and Mobile Computing (WCMC). Special issue on Mobile Ad Hoc Networking. Research, Trends and Applications*, vol. 2, pp. 483–502.

[18] Camp, T., Boleng, J., Williams, B., Navidi, W., and Wilcox, L. Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM 2002)* (2002), vol. 3, pp. 1678–1687.

[19] Campo, C. Directory Facilitator and Service Discovery Agent. *Dept. Telematic Engineering. Universidad Carlos III de Madrid, Spain*.

[20] Campo, C. *Tecnologías middleware para el desarrollo de servicios en entornos de computación ubicua*. PhD thesis, 2004.

[21] Castro, M., Druschel, P., Kermarrec, A.-M., and Rowstron, A. One Ring to Rule Them All: Service Discovery and Binding in Structured Peer-to-Peer Overlay Networks. In *Proceedings of the 10th Workshop on ACM SIGOPS European workshop: beyond the PC* (New York, NY, USA, 2002), ACM Press, pp. 140–145.

[22] Castro, P., Greenstein, B., Muntz, R., Kermani, P., Bisdikian, C., and Papadopouli, M. Locating Application Data Across Service Discovery Domains. In *Proceedings of the 7th annual international conference on Mobile Computing and Networking* (2001), ACM Press, pp. 28–42.

[23] Chakeres, I. D., and Perkins, C. E. Dynamic MANET On-demand (DYMO) Routing. Internet-Draft. draft-ietf-manet-dymo-11. http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-11.txt. Mobile Ad hoc Networks Working Group, November 2007.

[24] Chakraborty, D., and Anupam, J. GSD: A Novel Group-based Service Discovery Protocol for MANETS. In *4th IEEE Conference on Mobile and Wireless Communications Networks* (Stockholm. Sweden, September 2002).

[25] Chakraborty, D., and Finin, T. Toward Distributed Service Discovery in Pervasive Computing Environments. *IEEE Transactions on Mobile Computing 5*, 2 (2006), 97–112.

[26] Chakraborty, D., Joshi, A., and Yesha, Y. Integrating Service Discovery with Routing and Session Management for Ad hoc Networks. In *Ad Hoc Networks* (March 2006), vol. 4, pp. 204–224.

[27] Chatterjee, M., Das, S. K., and Turgut, D. WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks. In *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)* (April 2002), vol. 5, pp. 193–204.

[28] Chen, K., Shah, S. H., and Nahrstedt, K. Cross-Layer Design for Data Accessibility in Mobile Ad Hoc Networks. In *Wireless Personal Communication.* (2002), vol. 21, Kluwer Academic Publishers, pp. 49–76.

[29] Chen, Y. P., Liestman, A., and Liu, J. Clustering Algorithms for Ad Hoc Wireless Networks. In *Ad Hoc and Sensor Networks* (2004).

[30] Cheng, L. Service Advertisement and Discovery in Mobile Ad hoc Networks. *Department of Computer Science and Engineering, Lehigh University*.

[31] Chiang, C.-C., Wu, H.-K., Liu, W., and Gerla, M. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *IEEE Singapore International Conference on Networks, SICON'97, Singapore* (April 1997), IEEE, pp. 197–211.

[32] Chlamtac, I., Conti, M., and Liu, J. J.-N. Mobile Ad Hoc Networking: Imperatives and Challenges. In *Ad Hoc Networks* (2003), vol. 1, pp. 13–64.

[33] Cho, C., and Lee, D. Survey of Service Discovery Architectures for Mobile Ad hoc Networks. *Computer and Information Science and Engineering Department, University of Florida, Gainesville, FL-32611, USA*.

[34] Choi, W., and Woo, M. A Distributed Weighted Clustering Algorithm for Mobile Ad Hoc Networks. In *AICT-ICIW '06: Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 73–78.

[35] Consortium, T. S. Salutation architecture specification version 2.0c.

[36] Conti, M., Maselli, G., Turi, G., and Giordano, S. Cross-Layering in Mobile Ad Hoc Network Design. *Computer 37*, 2 (2004), 48–51.

[37] Correa, B. A., Ospina, L., and Hincapie, R. C. Survey of clustering techniques for mobile ad hoc networks. In *Rev.fac.ing.univ. Antioquia* (2007), pp. 145–161.

[38] Corson, M. S., and Macker, J. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Request For Comments 2501, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2501.txt, January 1999.

[39] Czerwinski, S. E., Zhao, B. Y., Hodes, T. D., Joseph, A. D., and Katz, R. H. An architecture for a secure service discovery service. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (1999), no. 5, ACM Press, pp. 24–35.

[40] Denko, M. K. The use of Mobile Agents for Clustering in Mobile Ad Hoc Networks. In *Proceedings of SAICSIT. University of Guelph, Canada* (2003), pp. 241–247.

[41] Dey, A., Abowd, G., and Salber, D. A Context-based Infrastructure for Smart Environments. In *Proceedings of the 1st International Workshop on*

Managing Interactions in Smart Environments (MANSE '99) (1999), pp. 114–128.

[42] DNS-SD. DNS service discovery.

[43] Doval, D., and O'Mahony, D. NOM: Resource Location and Discovery for Ad Hoc Mobile Networks. *Proceedings of the 1st Annual Mediterranean Ad Hoc Networking Workshop* (2002).

[44] Draves, R., Padhye, J., and Zill, B. Comparison of routing metrics for static multi-hop wireless networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2004), ACM, pp. 133–144.

[45] Draves, R., Padhye, J., and Zill, B. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking* (New York, NY, USA, 2004), ACM, pp. 114–128.

[46] Duong, H. D. H., Melchiorre, C., Meyer, E. M., Nieto, I., París, G., Pelliccione, P., and Tastet-Cherel, F. A software architecture for simple and reliable computing environments for collaborative work. In *Proceedings of the 5th International Workshop on Distributed and Mobile Collaboration (DMC 2007)* (Paris, France, June 2007), IEEE WETICE.

[47] Edwards, W. K. Discovery Systems in Ubiquitous Computing. In *IEEE Pervasive Computing* (2006), IEEE Educational Activities Department, pp. 70–77.

[48] Edwards, W. K., Newman, M. W., Sedivy, J., and Izadi, S. Challenge: Recombinant Computing and the Speakeasy Approach. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking* (New York, NY, USA, 2002), ACM Press, pp. 279–286.

[49] Engelstad, P. E., and Zheng, Y. Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks. In *Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 2–15.

[50] Engelstad, P. E., Zheng, Y., Koodli, R., and Perkins, C. E. Service Discovery Architectures for On-Demand Ad Hoc Networks. In *Ad Hoc and Sensor Wireless Networks* (2005), University of Oslo, Norway.

[51] ER, I. I., and Seah, W. K. Mobility-based d-hop clustering algorithm for mobile ad hoc networks. In *Wireless Communications and Networking*

Conference 2004. WCNC (March 2004), Dept. of Comput. Sci., Nat. Univ. of Singapore, Singapore, pp. 2359–2364.

[52] Fall, K. The ns Manual "formerly ns Notes and Documentation". *The VINT Project*.

[53] Fan, Z., and Ho, E. G. Service Discovery in Mobile Ad Hoc Networks. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 05)* (2005), vol. 01, IEEE Computer Society, pp. 457–459.

[54] FIPA. FIPA Agent Discovery Service Specification. In *PC00095A* (November 2003).

[55] Flores-Cortés, C. A., Blair, G. S., and Grace, P. A Multi-protocol Framework for Ad hoc Service Discovery. In *MPAC '06: Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing* (2006).

[56] Frank, C., and Karl, H. Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems* (2004), ACM Press, pp. 105–114.

[57] Gao, Z., Wang, L., Yang, M., and Yang, X. CNPGSDP: An Efficient Group-based Service Discovery Protocol for MANETs. In *Comput. Networks* (2006), vol. 50, pp. 3165–3182.

[58] Garcia-Luna-Aceves, J. Source Tree Adaptive Routing (STAR) Protocol. Internet Draft. http://cabernet.levkowetz.com/html/draft-ietf-manet-star-00. IETF MANET Working Group., October 1999.

[59] Gerla, M., and Tsai, J. T.-C. Multicluster, mobile, multimedia radio network. *Journal of Wireless Networks 1*, 3 (1995), 255–265.

[60] GloMoSim. GloMoSim - Global Mobile Infortion Systems Simulation Library. http://pcl.cs.ucla.edu/projects/glomosim/.

[61] Goldsmith, A. J., and Wicker, S. B. Design challenges for energy-constrained ad hoc wireless networks. In *IEEE Wireless Communications* (Aug 2002), vol. 9, pp. 8–27.

[62] Gong, L. Project JXTA: A Technology Overview. In *Sun Microsystems. Palo Alto, CA. 94303 USA*.

[63] Grace, P., Blair, G. S., and Samuel, S. A Reflective Middleware to support Mobile Client Interoperability. In *Distributed Multimedia Research Group, Computing Department, Lancaster. Global Wireless Systems Research, Bell Laboratories, Lucent Technologies.*

[64] GTNetS. GTNetS - Georgia Tech Network Simulator. http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[65] Guo, Z., and Malakooti, B. Energy Aware Proactive MANET Routing with Prediction on Energy Consumption. In *International Conference on Wireless Algorithms,Systems and Applications (WASA 2007)* (Washington, DC, USA, August 2007), IEEE Computer Society, pp. 287–293.

[66] Guttman, E., Perkins, C., Veizades, J., and Day, M. Service Location Protocol. Version 2. RFC Editor, Request for Comments: 2608. June 1999.

[67] Haas, Z. J., Pearlman, M. R., and Samar, P. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt, July 2002.

[68] Halkes, G. P., Baggio, A., and Langendoer, K. G. A Simulation Study of Integrated Service Discovery. In *EUROSSC 2006* (2006), Lectures Notes in Computer Sciences. Springer-Verlag, pp. 39–53.

[69] Harbird, R., Hailes, S., and Mascolo, C. Adaptive Resource Discovery for Ubiquitous Computing. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing* (Toronto, Canada, 2004), ACM Press, pp. 155–160.

[70] Helal, S., Desai, N., Verma, V., and Lee, C. KONARK: A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *Proceedings of the Third IEEE Conference on Wireless Communication Networks* (March 2003).

[71] Helmy, A., Garg, S., Pamu, P., and Nahata, N. CARD: A Contact-based Architecture for Resource Discovery in Ad Hoc Networks. In *Mobile Networks and Applications* (2005), vol. 10, pp. 99–113.

[72] Hermann, R., Husemann, D., Moser, M., Nidd, M., Rohner, C., and Schade, A. DEAPspace: Transient Ad Hoc Networking of Pervasive Devices. In *Comput. Networks* (2001), vol. 35, Elsevier North-Holland, Inc., pp. 411–428.

[73] IETF WG. Mobile Ad-hoc Networks (manet). Internet Engineering Task Force. http://www.ietf.org/html.charters/manet-charter.html, September 2007.

[74] Jacquet, P., Mühlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001)* (2001).

[75] Jeong, I. Zone-Based Service Architecture for Wireless Ad Hoc Networks. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies* (2006), vol. 0, IEEE Computer Society, p. 69.

[76] Jiménez, A. R., Seco, F., Prieto, C., and Roa, J. Tecnologías sensoriales de localización para entornos inteligentes. In *I Congreso Español de Informática. Simposium UCAmI 05 Computación Ubicua e Inteligencia Ambiental)* (Instituto de Automática Industrial (CSIC), September 2005), pp. 75–86.

[77] Jini. The Community Resource for Jini Technology.

[78] Jodra, J. L., Vara, M., Cabero, J. M., Bagazgoitia, J., and Jodra, J. L. Service Discovery Mechanism Over OLSR for Mobile Ad-hoc Networks. In *20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06)* (2006), vol. 02, IEEE Computer Society, pp. 534–542.

[79] Johanson, B., Fox, A., and Winograd, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing 01*, 2 (2002), 67–74.

[80] Johnson, D. B., Maltz, D. A., and Broch, J. DSR: the Dynamic Source Routing Protocol for Multihop Wreless Ad Hoc Networks. *Addison-Wesley Longman Publishing Co., Inc.* (2001), 139–172.

[81] Kindberg, T., and Barton, J. A Web-Based Nomadic Computing System. In *Computer Networks.* (2001), vol. 35, pp. 443–456.

[82] Klein, M., König-Ries, B., and Obreiter, P. LANES - A Lightweight Overlay for Service Discovery in Mobile Ad Hoc Networks, 2003.

[83] Klein, M., König-Ries, B., and Obreiter, P. Service Rings: A Semantic Overlay for Service Discovery in Ad hoc Networks. *dexa 00* (2003), 180.

[84] Koodli, R., and Perkins, C. E. Service Discovery in On-Demand Ad Hoc Networks. In *INTERNET DRAFT* (October 2002).

[85] Kozat, U. C., and Tassiulas, L. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Department of Electrical and*

Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA. IEEE INFOCOM Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE (2003), pp. 1965–1975.

[86] Kozat, U. C., and Tassiulas, L. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. In *Department of Electrical and Computer Engineering and Institute for Systems Research* (University of Maryland, College Park, MD 20742, USA., June 2003).

[87] Lee, C., Yoon, S., Kim, E., and Helal, A. S. An Efficient Service Propagation Scheme for Large-Scale MANETs. In *Proceedings of the 4th international workshop on Middleware for Pervasive and Ad-Hoc Computing* (2006), ACM Press, p. 9.

[88] Lee, S.-J., Su, W., Hsu, J., Gerla, M., and Bagrodia, R. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *IEEE INFOCOM 2000* (2000).

[89] Lenders, V., May, M., and Plattner, B. Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach. In *WOWMOM* (2005), IEEE Computer Society, pp. 120–130.

[90] Li, L., and Lamont, L. A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-Layer Design. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 55–59.

[91] Liang, J.-C., Chen, J.-C., and Zhang, T. Mobile Service Discovery Protocol (MSDP) for Mobile Ad-Hoc Networks. In *Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)* (2007), vol. 00, IEEE Computer Society, pp. 352–362.

[92] Lin, C. R., and Gerla, M. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal of Selected Areas in Communications 15*, 7 (1997), 1265–1275.

[93] Lucio, G. F., Farrera, M. P., Jammeh, E., Fleury, M., Reed, M. J., and Ghanbari, M. Packet Analysis in Contemporary Network Simulators. In *IEEE Latin America Transactions* (University of Essex, Wivenhoe Park, Colchester, United Kingdom, June 2006), vol. 4, IEEE, pp. 297–305.

[94] Luo, H., and Barbeau, M. Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks. In *Second Annual Conference on*

Communication Networks and Services Research (2004), vol. 00, IEEE Computer Society, pp. 61–68.

[95] Macías, J. A. G., and Torres, D. A. Service Discovery in Mobile Ad Hoc Networks: Better at the Network Layer? In *2005 International Conference on Parallel Processing Workshops* (2005), vol. 00.

[96] Marin-Perianu, R. S., Hartel, P. H., and Scholten, H. A Classification of Service Discovery Protocols. Tech. rep., Centre for Telematics and Information Technology, University of Twente, The Netherlands, June 2005.

[97] Marin-Perianu, R. S., Scholten, H., Havinga, P., and Hartel, P. Performance Evaluation of a Cluster-Based Service Discovery Protocol for Heterogeneous Wireless Sensor Networks. University of Twente.

[98] Marin-Perianu, R. S., Scholten, H., Havinga, P., and Hartel, P. H. Cluster-based service discovery for heterogeneous wireless sensor networks. In *International Journal of Parallel, Emergent and Distributed Systems* (August 2008), vol. 23, pp. 325–346.

[99] Marti, S., and Krishnan, V. CARMEN: A Dynamic Service Discovery Architecture. In *Mobile and Media Systems Laboratory.* (HP Laboratories Palo Alto, September 2002).

[100] McDonald, A. B., and Znati, T. F. Design and Simulation of a Distributed Dynamic Clustering Algorithm for Multimode Routing in Wireless Ad Hoc Networks. In *SIMULATION* (2002), vol. 78, pp. 408–422.

[101] Mian, A., Beraldi, R., and Baldoni, R. Survey of Service Discovery Protocols in Mobile Ad Hoc Networks. Technical Report 4/06. Dipartimento di Informatica e Sistemistica "Antonio Ruberti". Università degli Studi di Roma "La Sapienza". Rome, Italy, 2006.

[102] Microsystems, S. Jini Network Technology. *http://wwws.sun.com/software/jini/* .

[103] Murthy, S., and Garcia-Luna-Aceves, J. J. An Efficient Routing Protocol for Wireless Networks. *Mobile Networks and Applications 1*, 2 (1996), 183–197.

[104] Nedos, A., Singh, K., Cunningham, R., and Clarke, S. A Gossip Protocol to Support Service Discovery with Heterogeneous Ontologies in MANETs. In *Technical Report, TCD-CS-2006-34. Distributed Systems Group, Trinity College Dublin, Ireland.*

[105] Newport, C. Simulating mobile ad hoc networks: a quantitative evaluation of common MANET simulation models. Tech. Rep. TR2004-504, Dartmouth College, Computer Science, Hanover, NH, June 2004.

[106] Nidd, M. Service discovery in DEAPspace. In *Personal Communications, IEEE* (2001), vol. 8, pp. 39–45.

[107] Nieto, I., Botía, J. A., Ruiz, P. M., and Gómez-Skarmeta, A. F. Distributed Contextual Information Storage using Content-Centric Hash Tables. In *University of Murcia*.

[108] Nocetti, F. G., Gonzalez, J. S., and Stojmenovic, I. Connectivity Based k-Hop Clustering in Wireless Networks. *Telecommunication Systems 22*, 1–4 (2003), 205–220.

[109] Noh, D., and Shin, H. SPIZ: An Effective Service Discovery Protocol for Mobile Ad Hoc Networks. *EURASIP Journal on Wireless Communications and Networking 7* (2007), 13 pages.

[110] Ns-2. Ns-2 - The Network Simulator. http://www.isi.edu/nsnam/ns/.

[111] OASIS UDDI. UDDI Technical White Paper. http://uddi.xml.org/uddi-org, 2004.

[112] Obaid, A., Khir, A., and Mili, H. A Routing Based Service Discovery Protocol for Ad hoc Networks. In *International Conference on Networking and Services (ICNS '07)* (2007), vol. 0, IEEE Computer Society, p. 108.

[113] Ogier, R. G., Templin, F. L., and Lewis, M. G. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). Request for Comments: 3684. http://www.ietf.org/rfc/rfc3684.txt, February 2004.

[114] Oliveira, L. B., Siqueira, I. G., Macedo, D. F., Loureiro, A. A. F., Wong, H. C., and Nogueira, J. M. Evaluation of Peer-to-Peer Network Content Discovery Techniques over Mobile Ad Hoc Networks. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks* (2005), vol. 01, IEEE Computer Society, pp. 51–56.

[115] OMNeT++ Community Site. OMNeT++ - Discrete Event Simulation System. http://www.omnetpp.org/.

[116] OPNET. OPNET Technologies - Making Networks and Applications Perform. http://www.opnet.com/.

[117] Outay, F., Vèque, V., and Bouallègue, R. Survey of Service Discovery Protocols and Benefits of Combining Service and Route Discovery.

International Journal of Computer Science and Network Security 7, 11 (November 2007), 85–92.

[118] Parekh, A. K. Selecting routers in ad-hoc wireless networks. In *Proceedings of the SBT/IEEE International Telecommunications Symposium* (August 1994).

[119] Park, V. D., and Corson, M. S. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 1405–1413.

[120] Parsec. Parsec - PARallel Simulation Environment for Complex systems. http://pcl.cs.ucla.edu/projects/parsec/.

[121] Patini, M., Beraldi, R., Marchetti, C., and Baldoni, R. A Middleware Architecture for Inter Ad-Hoc Networks Communication. In *Fourth International Conference on Web Information Systems Engineering Workshops* (2003), vol. 00, IEEE Computer Society, pp. 201–208.

[122] Patterson, C. A., Muntz, R. R., and Pancake, C. M. Challenges in Location-Aware Computing. *IEEE Pervasive Computing 2*, 2 (2003), 80–89.

[123] Perkins, C., and Bhagwat, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM, SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications* (1994), pp. 234–244.

[124] Perkins, C. E., Belding-Royer, E. M., and Das, S. R. Ad Hoc On-Demand distance vector (AODV) Routing. In *Request for Comments 3561. MANET Working Group. http://www.ietf.org/rfc/rfc3561.txt* (July 2003).

[125] POPEYE Consortium Partners. D4.2 - POPEYE lower level Architecture Definition. POPEYE Deliverable D4.2., October 2006.

[126] POPEYE Consortium Partners. POPEYE Project. Whitepaper 1.0. http://srvweb01.softeco.it/IST-Popeye/, December 2007.

[127] Purtoosi, R., Taheri, H., Mohammadi, A., and Foroozan, F. A Light-Weight Contention-Based Clustering Algorithm for Wireless Ad Hoc Networks. *The Fourth International Conference on Computer and Information Technology (CIT'04) 00* (2004), 627–632.

[128] QualNet. QualNet - Scalable Network Technologies. http://www.scalable-networks.com/.

[129] Radunovic, B. *A Cross-Layer Design of Wireless Ad-Hoc Networks*. PhD thesis, June 2005.

[130] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. A Scalable Content-Addressable Network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), ACM, pp. 161–172.

[131] Ratsimor, O., Chakraborty, D., Joshi, A., and Finin, T. ALLIA: Alliance-based Service Discovery for Ad Hoc Environments. In *Proceedings of the 2nd International Workshop on Mobile Commerce* (2002), ACM Press, pp. 1–9.

[132] Reddy, P. C., and Reddy, P. C. Performance Analysis of Ad Hoc Network Routing Protocols. Academic Open Internet Journal. http://www.acadjournal.com/2006/V17/part6/p3/, 2006.

[133] Ren, J., Zhang, X., and Liu, Y. Services Driven Cross-Layer Routing Protocol for Mobile Ad Hoc Networks. *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07) 2* (2007), 770–775.

[134] Riley, G. F. Simulation of large scale networks II: large-scale network simulations with GTNetS. In *WSC '03: Proceedings of the 35th conference on Winter simulation* (2003), Winter Simulation Conference, pp. 676–684.

[135] Robinson, R., and Indulska, J. Superstring: A Scalable Service Discovery Protocol for the Wide-Area Pervasive Environment. In *School of Information Technology and Electrical Engineering* (The University of Queensland. Networks, 2003), The 11th IEEE International Conference on, pp. 699–704.

[136] Ros, F. J. Evaluación de Propuestas de Interconexión a Internet para Redes Móviles Ad Hoc Híbridas. Trabajo final de carrera, Facultad de Informática. Depto. de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, 2004.

[137] Royer, E. M., and Toh, C.-K. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. IEEE Personal Communications, April 1999.

[138] Sailhan, F., and Issarny, V. Scalable Service Discovery for MANET. In *Third IEEE International Conference on Pervasive Computing and Communications* (2005), IEEE Computer Society, pp. 235–244.

[139] Saoud, M. A. E., Kunz, T., and Mahmoud, S. SLPManet: service location protocol for MANET. In *Proceeding of the 2006 international conference on Communications and mobile computing* (2006), ACM Press, pp. 701–706.

[140] Saoud, M. M. A. E. MANET Reference Configurations and Evaluation of Service Location Protocol for MANET. Thesis of master of applied science, Faculty of Engineering. Department of Systems and Computer Engineering. Carleton University, Ottawa, Ontario, Canada, April 2005.

[141] Schiele, G., Becker, C., and Rothermel, K. Energy-efficient cluster-based service discovery for Ubiquitous Computing. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC* (New York, NY, USA, 2004), ACM Press, p. 14.

[142] Seno, S. A. H., Budiarto, R., and Wan, T.-C. Survey and new Approach in Service Discovery and Advertisement for Mobile Ad hoc Networks. In *International Journal of Computer Science and Network Security* (February 2007), vol. 7, Network Research Group, School of Computer Sciences, Universiti Sains Malaysia, pp. 275–284.

[143] Shakkottai, S., Rappaport, T. S., and Karlsson, P. C. Cross-layer design for wireless networks. In *Communications Magazine, IEEE* (October 2003), vol. 41, pp. 74–80.

[144] Storey, M., Blair, G., and Friday, A. MARE: Resource Discovery and Configuration in Ad Hoc Networks. In *Mobile Networks and Applications* (2002), Kluwer Academic Publishers, pp. 377–387.

[145] Sun, J.-Z. Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing. In *Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on* (2001), vol. 3, pp. 316–321.

[146] Sundramoorthy, V., Scholten, H., Jansen, P., and Hartel, P. Service Discovery At Home. In *4th Int. Conference on Information, Communications and Signal Processing and 4th IEEE Pacific-Rim Conference On Multimedia (ICICS/PCM)* (Singapore, September 2003), vol. III, IEEE Computer Society Press, pp. 1929–1933.

[147] Toh, C.-K. *Ad Hoc Mobile Wireless Networks : Protocols and Systems.* Prentice-Hall. Upper Saddle River, New Jersey, 2002.

[148] Torres, D. A., and Macías, J. A. G. Performance Analysis of Two Approaches to Service Discovery in Mobile Ad Hoc Networks. Lectures Notes in Computer Sciences. Springer-Verlag, pp. 368–376.

[149] Tyan, J., and Mahmoud, Q. H. A Comprehensive Service Discovery Solution for Mobile Ad Hoc Networks. In *Mobile Networks and Applications* (2005), vol. 10, Kluwer Academic Publishers, pp. 423–434.

[150] UDDI. OASIS UDDI Specification TC. http://www.oasis-open.org/who/, 2005.

[151] UPnP forum. *White Paper, http://www.upnp.org (2000).*

[152] Ververidis, C., and Polyzos, G. Extended ZRP: a Routing Layer Based Service Discovery Protocol for Mobile Ad Hoc Networks. In *MOBIQUITOUS* (2005), vol. 0, IEEE Computer Society, pp. 65–72.

[153] Ververidis, C., and Polyzos, G. Routing Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops* (2005), vol. 00, IEEE Computer Society, pp. 258–262.

[154] Viswanath, K., and Obraczka, K. Modeling the performance of flooding in wireless multi-hop ad hoc networks. Computer Communications, May 2006.

[155] Wang, N.-C., and Wang, S.-M. An Efficient Location-Aided Routing Protocol for Mobile Ad Hoc Networks. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 335–341.

[156] Weiser, M. The Computer for the 21st Century. *ACM SIGMOBILE Mobile Computing and Communications Review 3*, 3 (1999), 3–11.

[157] Wu, J., and Cao, J. Connected k-Hop Clustering in Ad Hoc Networks. In *ICPP '05: Proceedings of the 2005 International Conference on Parallel Processing* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 373–380.

[158] Yadav, N. S., and Yadav, R. P. Performance Comparison and Analysis of Table-Driven and On-Demand Routing Protocols for Mobile Ad-hoc Networks. *International Journal of Information Technology 4*, 2 (2007).

[159] Younis, O., and Fahmy, S. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)* (Hong Kong, 2004).

[160] Zhang, X., and Riley, G. F. Performance of routing protocols in very Large-Scale Mobile Wireless Ad Hoc Networks. In *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (2005), IEEE Computer Society, pp. 115–122.

[161] Zhu, F., Mutka, M. W., and Ni, L. Splendor: A Secure, Private, and Location-Aware Service Discovery Protocol Supporting Mobile Services. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications* (2003), IEEE Computer Society, p. 235.

[162] Zhu, F., Mutka, M. W., and Ni, L. PrudentExposure: A Private and User-centric Service Discovery Protocol. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PERCOM'04)* (Los Alamitos, CA, USA, 2004), vol. 00, IEEE Computer Society, p. 329.

[163] Zhu, F., Mutka, M. W., and Ni, L. M. Service Discovery in Pervasive Computing Environments. In *IEEE Pervasive Computing* (2005), vol. 4, IEEE Educational Activities Department, pp. 81–90.