Sistemas Informáticos
Curso 2005-06

Sistema de Autoconfiguración para Redes Ad Hoc

Miguel Ángel Tolosa Diosdado
Adam Ameziane

Dirigido por:
Profª. Marta López Fernández
Dpto. Sistemas Informáticos y Programación
Grupo de Análisis, Seguridad y Sistemas (GASS)

Facultad de Informática
Universidad Complutense de Madrid

AGRADECIMIENTOS:

Queremos agradecer la dedicación de la profesora Marta López Fernández, Directora del presente Proyecto de Sistemas Informáticos, y del resto de integrantes del Grupo de Análisis, Seguridad y Sistemas (GASS) del Departamento de Sistemas Informáticos y Programación de la Universidad Complutense de Madrid, y de forma muy especial a Fabio Mesquita Buiati y a Javier García Villalba, Miembro y Director del citado Grupo, respectivamente, por el asesoramiento y las facilidades proporcionadas para el buen término de este Proyecto.

Índice

RESUMEN

Este trabajo tiene como objetivo principal el desarrollo de un protocolo de autoconfiguración para redes móviles Ad hoc (Manet). Ese servicio realiza la distribución de las informaciones de red tales como direcciones IP, de forma distribuida.

En el proyecto se trata de un modo detallado el proceso de autoconfiguración. Que consiste en que la configuración de un nuevo nodo entrante en la red se realice automáticamente. También trataremos el proceso de salida de nodos, en todas sus modalidades, y la consecuente actualización de información por parte de los nodos participantes de la red. Tanto en la entrada como en la salida. A este proceso lo conocemos como sincronización y se realizará mediante el intercambio de mensajes.

El protocolo de autoconfiguración especificado en ese trabajo se basa en Dynamic Configuration and Distribution Protocol (DCDP) que utiliza el mecanismo de división binaria para suministrar conjuntos distinguidos de direcciones IP a los nodos asegurando que una misma dirección IP no sea usada por dos o más nodos.

Además de la descripción y de la especificación del esquema de autoconfiguración, el trabajo realiza la implementación del protocolo para facilitar la comprensión y verificar la eficiencia del mismo en un ambiente real utilizando sistema operativo Linux y equipamientos de red inalámbrica de acuerdo con el patrón IEEE 802.11.b/g.

ABSTRACT

The main goal for this work is the development of a autoconfiguration protocol for mobile ad hoc networks (Manet). This service deals with the delivering of network information, such as addresses IP, in a distributed and collaborative way.

The project deals with the process of auto-configuration in a detailed way. It consists of the configuration of a new node entering into a network, a process which will be carried out automatically. Also, we will discuss the process of nodes leaving the network, which includes each manner of leaving, and the subsequent updating of information on the part of the participating nodes in the network.

The same issues will be addressed in relation to the entrance as well as the exit of nodes. We know this process as synchronization and it will be carried out by the exchange of messages.

The autoconfiguration protocol specified in this work is based on the Dynamic Configuration and Distribution Protocol (DCDP), which uses a mechanism of binary division to supply distinct sets of IP addresses to the nodes, assuring that the same IP address is not used for two or more nodes of the network.

Besides description and specification of a secure autoconfiguration scheme, an implementation for the secure protocol is also provided to facilitate the understanding as well as to verify its efficiency on a real environment. An actual test bed is built using Linux nodes with IEEE 802.11b/g wireless interfaces.

PALABRAS CLAVE:

Ad-hoc, autoconfiguración, sincronización, redes, DCDP, MANET, enrutamiento, IP, buddy.

1-INTRODUCCIÓN

El rápido desarrollo que las redes inalámbricas han experimentado en los últimos años ofrece a los usuarios diferentes soluciones que nos aproximan hacia las comunicaciones posibles en cualquier momento y desde cualquier lugar. Algunas de las opciones más destacadas en este campo son las redes inalámbricas de área local (WLANs), que aportan alta velocidad en la transmisión de datos, y las redes inalámbricas de área extendida (WWANs), que permiten una mayor movilidad para los usuarios.

En algunas situaciones, como en entornos de aplicación militar o en operaciones de emergencia, la necesidad de establecer comunicaciones dinámicas sin contar con ningún tipo de infraestructura se convierte en imprescindible. Entonces, la facilidad de rápido despliegue que las redes “Ad hoc”, herederas naturales de las redes inalámbricas de conmutación de paquetes, proporcionan, resulta de gran utilidad. De hecho, hoy en día, como en los años 70 cuando comenzó el interés por las redes inalámbricas de conmutación de paquetes, son los proyectos militares los que lideran la actividad investigadora en este campo, dado que las aplicaciones que pueden obtener mayor provecho de la flexibilidad y dinamismo de las redes Ad hoc son militares por excelencia.

Las redes Ad hoc están formadas por dispositivos móviles que cooperan los unos con los otros de manera distribuida para llevar a cabo la transmisión de los paquetes por los enlaces inalámbricos que forman la red, el encaminamiento de dichos paquetes y la gestión y el mantenimiento de la red misma. Sus peculiares características y limitaciones condicionan sobremanera el diseño en varios de los niveles OSI de red, de forma que parámetros como el ancho de banda o el consumo energético, que tienden a ser críticos en un diseño multinivel como el que parece apropiado para las redes Ad hoc, deben ser tenidos en cuenta de manera especialmente cuidadosa.

Este proyecto, con el objetivo de identificar cuestiones abiertas y problemas de investigación en este campo, es un estudio acerca de las redes Ad hoc que se centra en temas críticos relativos a la autoconfiguración.

1.1- MOTIVACIÓN

Las redes Ad hoc se caracterizan por no tener un punto de acceso y control que permita organizar la red y mantener su consistencia. Los equipos o nodos que forman parte de ella se organizan por sí mismos para ayudarse los unos a los otros en el proceso de transportar paquetes de datos entre un origen y un destino. Por tanto las MANET dan un paso más en cuanto a movilidad (todos los nodos de la red pueden ser móviles) y flexibilidad (ni siquiera hay inversión en infraestructura, y además se minimiza la gestión de la red pues se auto-organiza ella misma).

Ésta auto-organización necesita de un protocolo de autoconfiguración de direcciones IP eficiente, escalable y seguro, que mantenga en todo momento la consistencia de la red.

El desarrollo de éste protocolo de autoconfiguración es el objetivo de nuestro estudio en nuestro proyecto.

1.2 – OBJETIVO

El objetivo de este trabajo es la creación e implementación de un protocolo de bajo coste que haga la autoconfiguración de direcciones IP en redes Ad hoc en un ambiente real, tratar la entrada de nodos en una red, la salida de los mismos en sus diferentes modos (salida brusca o salida fácil) que atienda a todos los requisitos y características pertinentes a las redes Ad hoc, y la sincronización que mantenga la consistencia de la red en cada momento. Esta hecha una implementación en ambiente real del protocolo de autoconfiguración a fin de evaluar su funcionamiento y su escalabilidad. Para que la implementación de ese protocolo en un ambiente real sea realmente de gran valía para el ambiente académico, las herramientas utilizadas son: el sistema operativo Linux y el lenguaje de programación C, las cuales son herramientas públicas en Internet.

Para que tal objetivo fuese alcanzado, se hicieron estudios e investigaciones que presentan resultados a través de un ambiente de red propuesto, en las áreas de autoconfiguración de direcciones IP, auto-organización de los nodos participantes, gerencia de movilidad, y por fin la aplicabilidad de la solución.

1.3- ESTRUCTURA DE LA MEMORIA

Se tratan en esta memoria los asuntos y características relevantes para la comprensión el tema propuesto así como una exploración del ambiente utilizado para la implementación, pruebas y resultados del protocolo desarrollado. El trabajo que se sigue está estructurado en cinco capítulos, a fin de facilitar la comprensión del tema en estudio.

En el capítulo 2 se explican los conceptos de autoconfiguración en redes Ad hoc y los protocolos ya existentes revelando su vital importancia.

En el capítulo 3 se presenta una especificación del funcionamiento y la arquitectura del protocolo de autoconfiguración propuesto en éste trabajo.

El capítulo 4 explica la implementación del protocolo, los escenarios utilizados para la evaluación del protocolo, así como las pruebas y los resultados del trabajo realizado.

En el capítulo 5 presenta la conclusión del trabajo, evaluando los beneficios del estudio hecho para el mundo académico y sugerencias para investigaciones futuras.

2 -AUTOCONFIGURACIÓN EN REDES AD HOC

2.1- HISTORIA

El punto de partida para el estudio de las redes de transmisión radio de paquetes [1] coincidió con el interés de organizaciones militares tales como proyectos de investigación avanzados de defensa (DARPA) en los mediados de los años 60 y el establecimiento consiguiente de ARPANET en 1969. Iniciado en 1970, el ALOHANET, basado en la universidad de Hawaii, era el primer proyecto grande de la transmisión de paquetes por radio.

El PRNET, o la red experimental, es un proyecto que fue propuesto por SRI internacional del parque de Menlo y financiado por DARPA empezando en 1979 y funcionando por cuatro años y medio (la descripción de la tecnología del sistema se puede encontrar en [2] y [3]). Esto fue seguido por el desarrollo aficionado del grupo de la comunicación de Vancouver Digital (VADCG) del regulador terminal del nodo (TNC) en los años 80, que divide automáticamente los mensajes en los paquetes, afina el transmisor, y después envía los paquetes, así como las operaciones análogas como receptor. Posteriormente, la red aficionada de transmisión de paquetes por radio (AMPRNET) fue desarrollada y utilizada como onda corta por aficionados de radio. Antes de 1985, cerca de 30.000 aficionados alrededor del mundo tenían un equipo capaz de transmitir datos con seguridad. La limitación principal de estas redes consistió en el hecho de que cada nodo no podía entrar en contacto con varias otras estaciones simultáneamente. Esto fue superado con la construcción de la primera Metropolitan Area Network (MAN), que tenía una topología de estrella, donde cada estación era capaz de entrar en contacto con el nodo del cubo, pero lo no podía detectar. El paso siguiente, en 1991, vino con la utilización del acceso múltiple de sentido de portador (CSMA) que se realiza a través de un repetidor de la MAN con dos diversos canales de radio, dedicados respectivamente a la transmisión y a la recepción. Otros sistemas de transmisión de paquetes por radio, tales como datos celulares, han sido usados, pero las limitaciones impuestas por el medio sin cables (propagación multidireccional, alta tasa de fallos de los datos, ancho de banda limitado) no han permitido la existencia de las altas tasas de datos. Recientemente, la aparición de productos nuevos de licencia libre para LANs sin cables - basados en técnicas separadas del espectro, ha aumentado considerablemente las tasas de datos. Estos resultados son justos los que están siendo utilizados por la comunidad de redes Ad hoc para implementar la solución apropiada para sus propósitos, debido a la identidad conceptual entre las redes de transmisión de paquetes por radio que se han desarrollado hacia LANs sin cables y redes Ad hoc. Esta comunidad de redes Ad hoc está liderada por el grupo de trabajo MANET [4], creado adentro 1999 por el IETF para proporcionar un marco para el estudio de redes móviles Ad hoc.

2.2- VISIÓN GENERAL DE LAS REDES AD HOC

Una red móvil Ad hoc (Mobile Ad hoc NETwork o MANET) consiste en un conjunto de nodos móviles que se auto-organizan para poder comunicarse entre ellos sin necesitar que intervenga ningún tipo de infraestructura previa desplegada (como pudiera ser una antena de una red de telefonía o un punto de acceso de una WLAN). Este tipo de redes pueden surgir de forma espontánea y por sus características el medio inalámbrico es el que usan de forma natural para comunicarse.

Los nodos de la manet son simultáneamente hosts y routers, ya que pueden tanto ejecutar aplicaciones que hacen uso de la red, como participar en el encaminamiento de los paquetes.

El hecho de usar transmisión inalámbrica influye decididamente en el comportamiento de las manet. Las comunicaciones inalámbricas tienen un rango de transmisión en el que el receptor es capaz de recibir e interpretar correctamente la señal que envió el emisor. Si se encuentra fuera de este rango el receptor no podrá interpretar adecuadamente los paquetes que fueron destinados a él. Por eso en las manet los nodos colaboran para enviarse los paquetes de datos enrutándolos salto a salto. En la figura 2.1 vemos cómo el nodo B colabora para que el tráfico destinado a C llegue a buen puerto.

Las manet pueden concebirse como redes aisladas o como extensiones de redes fijas a las que están conectadas. Este último caso es lo que se conoce como redes Ad hoc híbridas y necesitan de uno o varios gateways que ejerzan de pasarela entre ambas redes. Hay consenso en que las manet no serán redes de transito que conecten otras redes, sino que el tráfico estará originado/dirigido por/hacia los nodos internos [5].

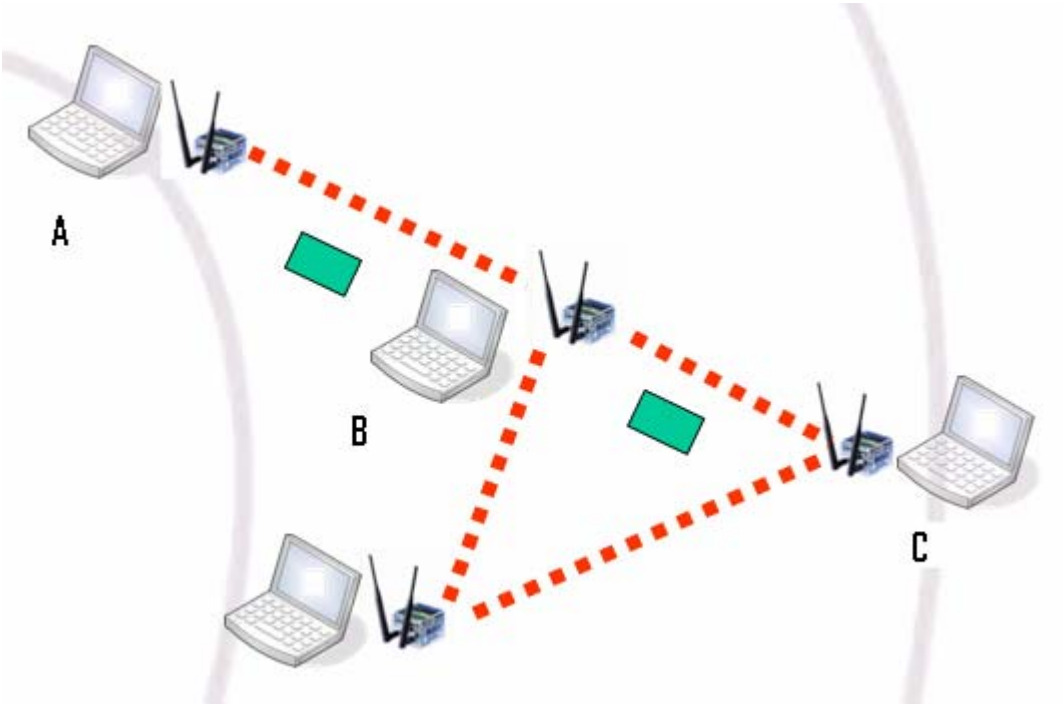


Figura 2.1: Retransmisión de un paquete por parte de un nodo intermedio en una MANET.

2.2.1- CARACTERÍSTICAS

Una red Ad hoc consiste en un grupo de terminales móviles que forman una red temporal en acoplamientos inalámbricos sin la ayuda de cualquier administración centralizada o servicio de ayuda estándar regularmente disponible en la red.

Las características más relevantes de las redes Ad hoc son las siguientes:

1. Topologías dinámicas: los nodos pueden moverse libremente en direcciones arbitrarias y con velocidad caprichosa. Por lo tanto, la red debe adaptarse a los cambios imprevisibles en su topología, que es típicamente multihop.

2. El ancho de banda restringido: las restricciones impuestas por el canal inalámbrico (wireless), tal como acceso múltiple, interferencia multidireccional, ruido, disponibilidad limitada del espectro, junto con los problemas inherentes que el protocolo de control de acceso al medio tiene que tratar , hace que el rendimiento de procesamiento para cada nodo sea mucho menos que la tasa máxima de transmisión radio de los datos.

3. Energía limitada: los dispositivos que forman la parte de las rede Ad hoc pueden ser limitados en energía debido a las circunstancias de su funcionamiento (como en las redes de sensores, por ejemplo, donde maximizando el promedio de vida de la red es un criterio de diseño), por lo tanto, los algoritmos de la encaminamiento deben manejar correctamente esta edición, que puede ser complicada si el modo somnolencia se acepta para los terminales.

4. Seguridad física limitada: las redes móviles inalámbricas son susceptibles de tener carencia de seguridad y pueden ser atacados fácilmente. Técnicas existentes de seguridad se aplican en la capa del acoplamiento de la fecha para reducir el riesgo, pero algunos mecanismos se pueden también introducir en la capa de red. Por otra parte, el hecho de ser una red descentralizada proporciona robustez adicional contra el solo punto de fallos.

Además, algunas redes (e.g. redes militares móviles) pueden ser relativamente grandes (e.g. decenas o centenares de nodos por área de encaminamiento), aunque la necesidad de escalabilidad no es única para las redes Ad hoc. Sin embargo, debido a las características precedentes, los mecanismos requeridos para alcanzar escalabilidad son probablemente más complicados. Realmente, la mayor parte de los protocolos existentes fallan para las redes grandes.

Estas características crean un conjunto de preocupaciones de funcionamiento por el diseño del protocolo que extienden más allá de ésas que dirigen el diseño de encaminar dentro de la alta velocidad, topología semi-estática de otras redes de paquetes tales como Internet fijo.

2.2.2- CLASIFICACIÓN

No hay ningún consenso existente para clasificar las redes Ad hoc, sin embargo podemos clasificarlas de acuerdo con su forma de comunicación.

Único salto (s*ingle-hop*)

Los nodos se comunican directamente con los otros nodos, pues están dentro del área de transmisión. A continuación en la Figura 2.2 podemos ver una red *Ad hoc* single-hop.

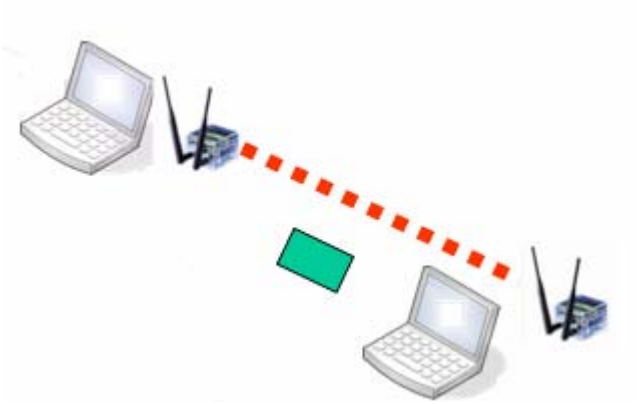


Figura 2.2 – Una red *Ad hoc single-hop.*

Múltiples saltos (*multi-hop*)

Algunos nodos no pueden comunicarse directamente con los otros nodos. Así, los datos deben ser encaminados por nodos intermediarios, también llamados de nodos encaminadores. A continuación en la Figura 2.3 podemos ver una red *Ad hoc* multi-hop.

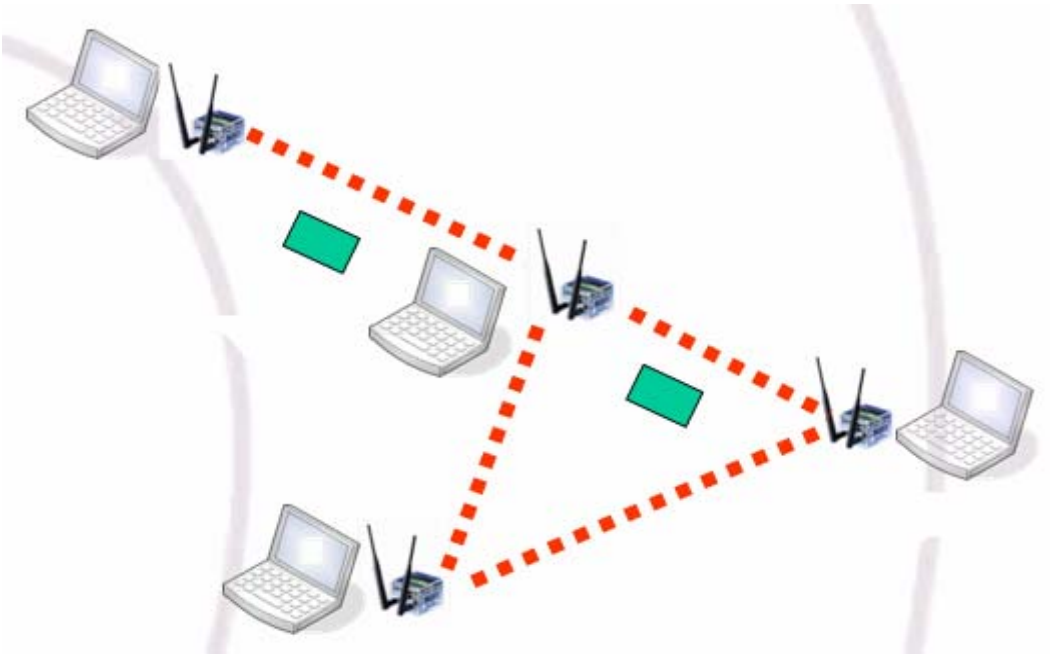


Figura 2.3 – Una red *Ad hoc multi-hop.*

2.2.3- APLICACIONES

Los ejemplos del uso práctico de las redes Ad hoc son limitados solamente por la imaginación. Así pues, el conjunto de aplicaciones para las redes Ad hoc es diverso, desde las redes pequeñas, estáticas que son limitadas por las fuentes de energía, hasta grandes, móviles, redes altamente dinámicas. Las aplicaciones típicas son aquellas en las cuales las comunicaciones eficientes y dinámicas deben ser establecidas. Algunos ejemplos son:

- Conferencias y reuniones para un grupo de gente con ordenadores portátiles que pueden desear intercambiar archivos y datos sin la mediación de infraestructura alguna adicional entre ellos.

- La comunicación entre los aparatos electrodomésticos inteligentes en un domicilio puede ser sostenida por una red Ad hoc entre los diversos dispositivos, que pueden compartir información de control para su correcto funcionamiento. Podemos pensar en una red Ad hoc formada por nuestros aparatos electrodomésticos en la cocina, la televisión, las ventanas y las puertas, el sistema de aire acondicionado, todo junto comunicado para realizar automáticamente las funciones imaginativas relacionadas con su control y su relación con el usuario.

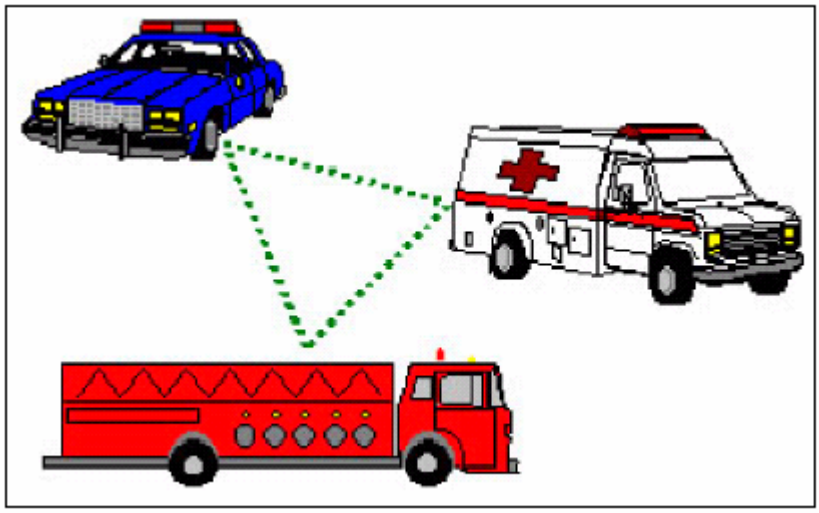


Figura 2.4: Las redes Ad hoc pueden ser útiles en emergencias y operaciones de rescate.

- Las operaciones de búsqueda y de rescate en emergencias requieren rápidas y dinámicas comunicaciones, y con la ayuda de las redes Ad hoc podrían ser desarrolladas en áreas inhospitalarias.

- El uso de las redes Ad hoc es conveniente en las áreas donde el terremoto u otros desastres naturales han destruido las infraestructuras de la comunicación. En esto caso, el despliegue rápido es un parámetro fundamental, así como la carencia de necesidad de cualquier infraestructura fija.

- Las redes Ad hoc satisfacen perfectamente las necesidades militares como supervivencia en el campo de batalla, la operación sin infraestructura pre-colocada y la conectividad. De hecho, la investigación sobre la red de transmisión de paquetes por radio comenzó en un contexto militar y hoy en día, el concepto del campo de batalla de Digital [10] es un asunto ardiente.

- Para los propósitos de supervisión y medición, una gran cantidad de dispositivos computadores pequeños se podrían extender por un área geográfica (la

dispersión de los sensores) para formar una red Ad hoc auto-sostenida. En este caso uno de los más importantes criterios de diseño deben obtener el promedio de vida más alto para la red, basado en los requisitos del consumo de energía, así como la necesidad de escalabilidad, a condición de que el número de sensores sea muy alto. Algunos ejemplos de esta clase de usos podrían ser:

- Redes de sensores militares para detectar los movimientos enemigos, la presencia de material peligroso (tal como gases venenosos o radiación), explosiones, etc.
- Redes de sensores ambientales para detectar y para supervisar cambios ambientales.
- Redes inalámbricas de sensores de tráfico para supervisar el tráfico de vehículos en a carretera o en una parte congestionada de una ciudad.
- Redes inalámbricas de sensores de vigilancia para proporcionar seguridad en los supermercados y centros comerciales, en el garaje del estacionamiento, u otra facilidad.
- Redes inalámbricas de sensores de estacionamiento para determinar qué puntos están ocupados y que puntos están libres.

2.2.4- VENTAJAS

Las redes Ad hoc son realmente una alternativa a las redes fijas en algunas situaciones operacionales, pero un análisis de sus ventajas y desventajas, nos ayudará a conocer los usos y los contextos en los que pueden ser útiles. Debajo, presentamos algunos de ésos. Entre las ventajas de las redes Ad hoc está:

Instalación rápida: el nivel de la flexibilidad para la creación de la redes Ad hoc es alto, puesto que no requieren ninguna instalación o infraestructura previa y, así, pueden ser montadas y desmontadas en poco tiempo.

Topologías dinámicas: los nodos pueden moverse arbitrariamente alrededor de la red y pueden desaparecer temporalmente de la red, por lo que el grafo de la topología de la red puede cambiar continuamente a una velocidad indeterminada.

Tolerancia a fallos: debido a las limitaciones de las interfaces de radio y de la topología dinámica, las redes Ad hoc soportan fallos de conexión, porque los protocolos de control de transmisión y encaminamiento están diseñados para manejar estas situaciones.

Conectividad: el uso de puntos centralizados o pasarelas (gateways) no es necesario para la comunicación dentro de una red Ad hoc, debido a la colaboración entre los nodos en la tarea de entregar los paquetes.

Movilidad: los nodos móviles inalámbricos pueden moverse al mismo tiempo en distintas direcciones. Aunque los algoritmos de encaminamiento tratan esto, las simulaciones del funcionamiento demuestran que hay un límite de la movilidad del nodo donde la operación del protocolo comienza a fallar.

Coste: Las redes Ad hoc podrían ser más económicas en algunos casos, eliminando costes de la infraestructura fija y reduciendo el consumo de energía en los

nodos móviles. Posibilidad de la reutilización del espectro: debido a los enlaces de comunicaciones cortos (nodo a nodo, en vez del nodo a una estación central base), los niveles de emisión radio se podrían mantener a bajo nivel. Esto aumenta posibilidad de la reutilización del espectro o la posibilidad de usar bandas sin licencia.

2.2.5- DESVENTAJAS

Algunos de los problemas que las redes Ad hoc tienen son:

Ancho de banda restringido: según lo comentado arriba, la capacidad de los enlaces inalámbricos es siempre mucho más baja que en los cableados. De hecho, varios Gbps está disponible para la LAN cableada, mientras que, hoy en día, las aplicaciones comerciales para LANs inalámbricas trabajan típicamente alrededor de 2 Mbps.

Capacidad de proceso: la mayor parte de los nodos son dispositivos sin una potente CPU. Además, las tareas de la red tales como el encaminamiento y la transmisión de datos no pueden consumir los recursos de energía de los dispositivos, previstos para desempeñar cualquier otro papel, tal como detección de funciones.

Restricción de energía: la energía de las baterías es limitada en todos los dispositivos, y no permite un funcionamiento infinito en el tiempo para los nodos. Por lo tanto, la energía no debe ser desperdiciada y por esto han sido implementados algunos algoritmos de conservación de energía (COMPOW [6] , PARO [7] y MBCR [8] son algunos ejemplos).

Latencia alta: cuando un diseño de conservación de energía se ha aplicado, eso significa que los nodos están durmiendo cuando no tienen que transmitir ningún dato. Cuando el intercambio de datos entre dos nodos pasa por nodos que están durmiendo, el retraso puede ser más alto si el algoritmo de encaminamiento decide que estos nodos se deben despertar.

Errores de transmisión: la atenuación e interferencias son otros efectos de los enlaces inalámbricos que aumentan la tasa de error.

Seguridad: [9] Resume las vulnerabilidades y los ataques que las redes Ad hoc pueden sufrir. Los autores dividen los ataques posibles en pasivos, cuando el atacante procura solamente descubrir la información valiosa escuchando el tráfico; y ataques activos, que ocurren cuando el atacante inyecta arbitrariamente paquetes en la red con el propósito de inhabilitar la red.

Localización: el direccionamiento es el otro problema para la capa de red en las redes Ad hoc, la información sobre la localización de la dirección IP utilizada en redes fijas ofrece algunas facilidades para la encaminamiento que no pueden aplicarse en las redes Ad hoc. El modo de direccionamiento en las redes Ad hoc, por supuesto, no tiene nada que hacer con la posición del nodo.

Roaming: los cambios continuos en el grafo de conectividad de la red implican que los algoritmos de roaming de la red fija no son aplicables en las redes Ad hoc, porque se basan en la existencia de rutas garantizadas a algunos destinos.

Comercialmente inasequible: las redes Ad hoc está todavía lejos de ser desplegadas en una base comercial de grande escala.

2.2.6 - ENCAMINAMIENTO

La tarea de encaminar en las redes Ad hoc se logra en la capa de red de la OSI, aunque hay una gran dependencia entre la MAC y la capa de red en las redes Ad hoc y por consiguiente, aspectos como el rango de transmisión y la fuerza de control afectan a los algoritmos de encaminamiento en gran parte.

El primer aspecto que debemos considerar al analizar la elección del protocolo de encaminamiento en las redes Ad hoc es porqué los algoritmos convencionales de encaminamiento no serían convenientes para ellas. Están bien probados y absolutamente familiares para la comunidad de las comunicaciones. El problema es que fueron diseñados para las redes estáticas, y tienen problemas para converger a un estado estable en una red Ad hoc con una topología de frecuentes cambios.

Otra característica para los protocolos convencionales como link-state y distance-vector es que asumen enlaces bidireccionales (si un nodo A puede oír otro nodo B, entonces B puede también oír A), que, al desigual que en las redes cableadas, no es siempre verdad en un ambiente de radio inalámbrica.

El problema de contar-a-infinito del algoritmo distribuido clásico de Bellman-Ford era el primer asunto a solucionar y algunos algoritmos distribuidos Shortest-Path(ruta más corta) ([11], [12], [13], [14], [15]) fueron propuestos para eliminar este problema utilizando la información que tiene en cuenta la longitud y al salto segundo-a-último (predecesor) de la ruta más corta a cada destino.

2.2.6.1- CARACTERÍSTICAS ESPERADAS

Luego, una plétora de algoritmos se ha propuesto para satisfacer algunas de las características que un algoritmo ideal para las redes Ad hoc[16] debe tener:

- Tener ejecución descentralizada.
- Proporcionar rutas libres.
- Responder rápidamente a los cambios de la topología.
- Adaptarse al patrón de tráfico en base a la demanda
- Escalar tanto como crece el tamaño de la red.
- Minimizar el retardo.
- Presentar múltiples rutas para evitar la congestión.
- Eficiente en ancho de banda (minimizar los gastos indirectos del encaminamiento).
- Utilizar enlaces tanto unidireccionales como bidireccionales.
- Conservar la energía y permitir la operación de somnolencia.
- Seguridad de garantía en la red.
- Calidad de soporte del servicio y manejo de las prioridades del mensaje.

Además de estas características que los protocolos de encaminamiento en las redes Ad hoc deben cumplir, hay algunas características relevantes a considerar en el diseño de éstos algoritmos.

Por ejemplo, la asignación de las direcciones concierne la configuración de los nodos, pero si no se maneja correctamente - y en las redes Ad hoc esto implica muchas particularidades -los protocolos de encaminamiento no pueden funcionar.

Además, el conocimiento de las localizaciones del nodo, a través de un sistema como el GPS (o, en algunos años, GALILEO [17], el futuro sistema europeo de navegación basado en satélite), podría ayudar en algunas ocasiones en la tarea del encaminamiento. También, sería interesante descubrir si una red Ad hoc podría soportar servicios de voz en tiempo real y video, y los requisitos que este hecho introduciría en el sistema en general y en el protocolo de encaminamiento particularmente.

2.2.6.2 - MÉTRICAS POSIBLES

Existen varias métricas que se puede utilizar para determinar el funcionamiento de los protocolos de encaminamiento:

- Rendimiento de procesamiento de datos fin-a-fin: tasa acertada media de la transmisión.
- Retardo fin-a-fin: el tiempo medio que tarda un paquete en ir desde su origen a su destino.
- Tiempo de adquisición de la ruta.
- Porcentaje de error en la entrega.
- Eficiencia.
- Diversidad de ruta: deseable debido a las restricciones de la ancho de banda y de energía.
- Optimalidad de la ruta.

2.2.6.3 - ESQUEMAS BÁSICOS DEL PROTOCOLO DE ENCAMINAMIENTO.

Como los algoritmos sugeridos en la literatura tienen un protocolo tradicional de encaminamiento, es necesario entender la operación básica para los protocolos convencionales como el Distance-vector, link-state y source-routing:

- **Link-State**: en el encaminamiento link-state [18] cada router primero obtiene una vista de la topología completa de la red con un coste para cada enlace y entonces calcula la ruta más corta a cada otro router usando el algoritmo de Dijkstra.

- **Distance-Vector**: en el Distance-Vector [18], cada nodo supervisa solamente el coste de sus enlaces salientes y difunde periódicamente una valoración de la distancia más corta a cada otro nodo en la red. Los nodos de recepción entonces utilizan esta información para recalcular las tablas de encaminamiento.

- **Source-Routing**: en el Source-Routing, cada paquete lleva la trayectoria completa que tiene que seguir a través de la red, cosa que requiere grandes gastos indirectos si la ruta tiene muchos saltos. Dado que la decisión del encaminamiento está tomada en la fuente, es fácil evitar bucles de encaminamiento.

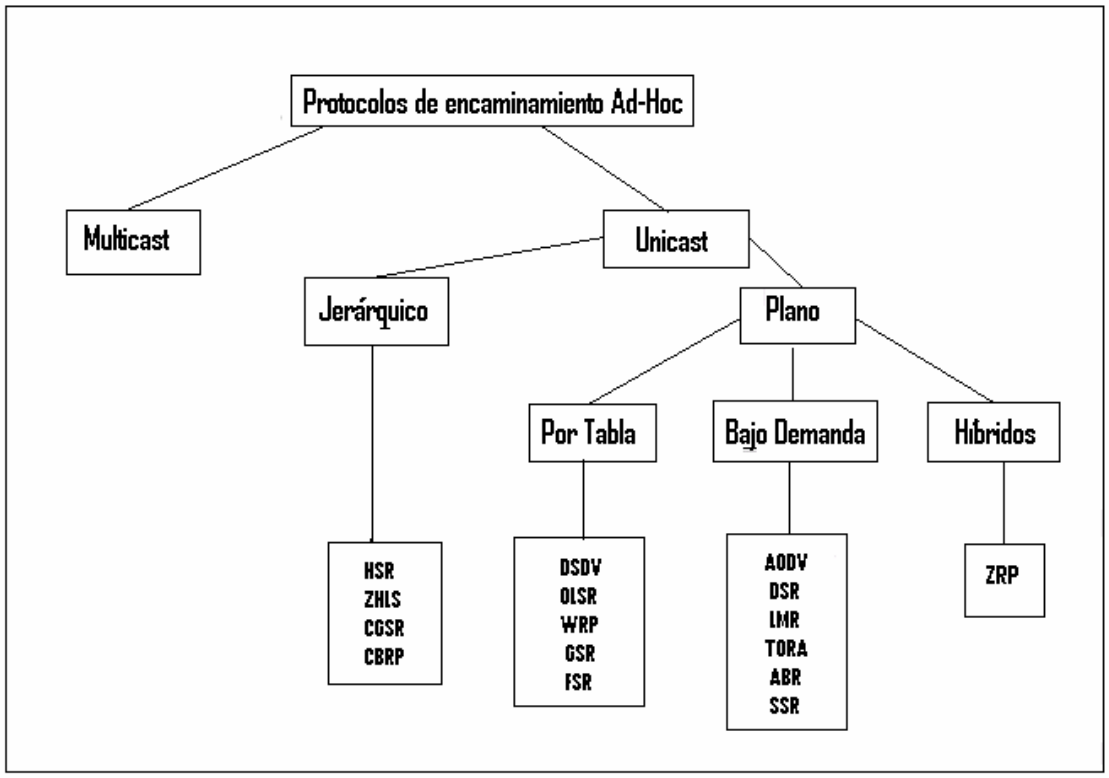


Figura 2.5: Clasificación de los protocolos de encaminamiento en las redes Ad hoc

2.2.6.4 CLASIFICACIÓN DE LOS ALGORITMOS DE ENCAMINAMIENTO.

Dependiendo de los criterios que consideremos, podemos tener varias clasificaciones de los algoritmos de encaminamiento para las redes Ad hoc.

Primero, podemos distinguir claramente entre los protocolos de encaminamiento unicast y multicast. Las comunicaciones simultáneas entre una fuente y un grupo de destinos en una red Ad hoc se pueden llevar acabo por el empleo de los protocolos de encaminamiento multicast, que tienen problemas particulares de diseño (explicados en [19], como problemas generales derivados del contexto Ad hoc. Algunos ejemplos de protocolos de encaminamiento multicast se pueden ver en [20]. Podemos clasificar los protocolos así [21]:

A) Estructura:

- Protocolos uniformes: todos los nodos tienen los mismos papeles en los esquemas del encaminamiento, así hay una estructura plana de encaminamiento.
- Protocolos no uniformes: para limitar la complejidad del encaminamiento disminuyendo el número de los nodos implicados en un cómputo de la ruta.

B) Información del estado:

- Protocolos basados en la topología: los nodos mantienen información de la topología a gran escala. Los protocolos basados en el estado del enlace(link-state) son representativos entre estos protocolos de encaminamiento.

- Protocolos basados en el destino: los nodos guardan una cierta información local de la topología, como los protocolos basados en el vector de distancia(Distance-Vector) lo hacen almacenando la distancia y vector (salto siguiente) al destino.

C) Scheduling:

- Los algoritmos (proactivos) guiados por tabla almacenan la información necesaria para los propósitos del encaminamiento en las tablas, que se ponen al día en varias ocasiones a través de los paquetes de control que son enviados por cada nodo. Las actualizaciones pueden también responder a los cambios topológicos de la red.

- Los protocolos (reactivos) bajo demanda, en contraste con los protocolos guiados por tabla, calculan la ruta a un destino específico solamente cuando se necesita, así que una tabla de encaminamiento que contiene todos los nodos como entradas no tiene que ser mantenida en cada nodo. Cuando una fuente desea enviar a un destino, invoca un mecanismo del descubrimiento de la ruta para encontrar la trayectoria al destino. La ruta sigue siendo válida hasta que el destino sea alcanzado o hasta que la ruta se no necesite más.

- También están los esquemas híbridos .

La figura 2.5 representa una clasificación general posible de los protocolos de encaminamiento Ad hoc.

2.2.6.5- DESCRIPCIÓN DE ALGUNOS PROTOCOLOS USADOS.

DSDV

El protocolo Destination-Sequenced Distance-Vector es un protocolo de encaminamiento basado en el vector de distancia donde en su tabla de encaminamiento, cada nodo mantiene, para cada destino disponible, además del número de los saltos y del siguiente-salto en la ruta para alcanzarlo, el número de secuencia para cada ruta para garantizar la libertad del lazo (véase Tabla 2.1). El número de secuencia demuestra la frescura de una ruta, puesto que aumentan cuando el nodo de la fuente detecta que una ruta a algún destino ha sido rota (en ese momento el número de los saltos para esa ruta se fija a infinito). Por lo tanto, números de secuencia más altos resultan ser una característica favorable para que una ruta sea elegida. Si dos rutas tienen los mismos números de secuencia, entonces el criterio es el número del saltos hacia el destino.

El procedimiento que este protocolo sigue para cuidar los cambios de la topología se basa en dos clases de puesta al día: actualizaciones conducidas por el tiempo, que son una transmisión periódica de la tabla de encaminamiento de los nodos; y actualizaciones conducidas por evento, que reaccionan a la fallos en los enlaces. Cuando sucede esto el valor métrico de la actualización es uno y se aumenta mientras se propaga salto-por-salto en la red.

Para reducir la cantidad de información en la difusión de las tablas de encaminamiento, dos tipos distintos de mensajes de actualización son definidos: descarga completa e incremental. Una descarga completa envía la tabla de encaminamiento llena a los vecinos y puede extender muchos paquetes mientras que en una actualización incremental solamente esas entradas de la tabla de encaminamiento que tenían un cambio métrico desde la última actualización son enviadas y debe ser un solo paquete. En redes estables, las actualizaciones incrementales se envían para evitar tráfico adicional y las descargas completas son absolutamente esporádicas, mientras que en redes de cambio rápido , las descargas completas llegan a ser más eficaces y por eso, son más frecuentes.

El problema principal que tiene este protocolo se presenta a partir del tiempo que necesita para converger, porque una ruta no puede ser utilizada después de que una cierta hora transcurra desde la difusión periódica. Esto puede ser inaceptable en una red Ad hoc móvil, donde se espera que la topología sea muy dinámica. Por otra parte, las difusiones periódicas también agregan una gran cantidad de gastos indirectos en la red, un revés común para todos los protocolos guiados por tablas.

Tabla 2.1: Una entrada de la tabla de encaminamiento de DSDV

Destino	Siguiente Salto	Número de Saltos	Número Secuencia

AODV

El encaminamiento Ad hoc On-Demand Distance Vector (AODV) [22] es uno de los algoritmos más usados para las simulaciones de las redes Ad hoc y, como está considerado por el IETF un trabajo en marcha, se está modificando continuamente. Cada terminal móvil funciona como router especializado, y las rutas se obtienen según lo necesitado (es decir, bajo demanda) con poco o nada de confianza en los avisos periódicos. AODV proporciona rutas sin bucles incluso mientras se están reparando enlaces rotos. Porque el protocolo no requiere avisos periódicos globales del encaminamiento, la demanda en el ancho de banda total disponible para los nodos móviles es substancialmente menos que en esos protocolos que necesitan tales avisos. Sin embargo aún puede mantener la mayor parte de las ventajas de los mecanismos del encaminamiento basado en el vector de distancias. El algoritmo escala también a las poblaciones grandes de nodos móviles que desean formar redes Ad hoc.

Los nodos que no mienten en las rutas activas, ni mantienen información alguna de encaminamiento, ni participan en ningún intercambio periódico de las tablas de encaminamiento. Además, un nodo no tiene que descubrir y mantener una ruta a otro nodo hasta que los dos necesiten comunicarse, a menos que el nodo anterior se esté comportando como estación intermedia para mantener una conexión entre dos otros nodos.

Cuando la conectividad local del nodo móvil es de interés, cada nodo móvil puede ser informado de los otros nodos en su vecindad por el uso de varias técnicas, incluyendo las difusiones locales conocidas como mensajes “HELLO”.

Las tablas de encaminamiento de los nodos dentro de la vecindad se organizan para optimizar el tiempo de respuesta y para proporcionar un tiempo de respuesta rápido a las peticiones de establecimiento de rutas nuevas.

Los objetivos primarios del algoritmo son:

- Difundir los paquetes de descubrimiento solamente cuando es necesario.
- Distinguir entre la gestión local de la conectividad (detección de la vecindad) y el mantenimiento general de la topología.
- Diseminar información sobre cambios en la conectividad local a esos nodos móviles vecinos que probablemente necesiten la información.

AODV utiliza un mecanismo de descubrimiento de la ruta de difusión, tal como se utiliza (con modificaciones) en el algoritmo DSR. En vez del encaminamiento en el origen, AODV confía el establecimiento dinámico de las entradas de las tablas de rutas a los nodos intermedios. Esta diferencia se paga en redes con muchos nodos, donde los gastos indirectos más grandes son incurridos llevando las rutas del origen en cada paquete de datos. Para mantener la información de encaminamiento más reciente entre los nodos, el concepto de los números de secuencia del destino se presta del DSDV. Al

desigual que en DSDV, cada nodo Ad hoc mantiene un contador monótono ascendente del número de secuencia, que se utiliza para sustituir las viejas rutas depositadas. La combinación de estas técnicas da un algoritmo que utiliza el ancho de banda eficientemente (minimizando la carga de la red para el tráfico de control y de datos), responde a los cambios en la topología, y asegura un encaminamiento libre de bucles.

DSR

Dinamic Source Routing (DSR) [23] es un protocolo de encaminamiento diseñado específicamente para su uso en redes Ad hoc móviles. El protocolo permite que los nodos descubran dinámicamente una ruta del origen a través de saltos múltiples a cualquier destino en la red Ad hoc. Al usar el encaminamiento en el origen, cada paquete que se encaminará lleva en su cabecera una lista completa y ordenada de los nodos a través de los cuales el paquete debe pasar.

Una ventaja clave del encaminamiento en el origen es que los saltos intermedios no necesitan mantener la información de encaminamiento para encaminar los paquetes que reciben, puesto que los paquetes ellos mismos contienen ya toda la información de encaminamiento necesaria. Esto, junto a la naturaleza dinámica y bajo demanda del descubrimiento de rutas en el DSR, elimina totalmente la necesidad de avisos periódicos del router y de paquetes del estado del enlace, reduciendo perceptiblemente los gastos indirectos de DSR, especialmente durante los períodos en que la topología de la red es estable y estos paquetes sirven solamente como avisos de mantenimiento en vida.

Para enviar un paquete a otro terminal, el remitente construye una ruta del origen en la cabecera del paquete, dando la dirección de cada terminal en la red a través del cual el paquete se debe remitir para alcanzar el terminal destino.

El remitente entonces transmite el paquete sobre su interfaz inalámbrica de la red al primer salto identificado en la ruta del origen. Cuando un terminal recibe un paquete, si este terminal no es el destino final del paquete, transmite simplemente el paquete al salto siguiente identificado en la ruta del origen en la cabecera del paquete, así como a las rutas hacia todos los nodos en sentido descendente.

Una vez que el paquete alcance su destino final, se entrega al software de la capa de red en ese terminal. Cada terminal móvil que participa en la red Ad hoc mantiene una caché de rutas en la cual deposita las rutas del origen que ha aprendido. Cuando un terminal envía un paquete a otro terminal, el remitente chequea primero su caché de rutas para encontrar una ruta del origen hacia el destino. Si se encuentra una ruta, el remitente la utiliza para transmitir el paquete. Si no se encuentra ninguna ruta, el remitente puede procurar descubrir una usando el protocolo de descubrimiento de ruta.

El proceso de descubrimiento de ruta comienza con un mensaje de petición de ruta difundido por el origen. Cada nodo que lo recibe agrega su número de identificación único en el registro de la ruta hasta que el paquete alcanza el destino o un nodo intermedio que contenga en su caché de rutas una ruta válida al destino. Entonces, un paquete de contestación de ruta se envía de nuevo al nodo que inició el descubrimiento de la ruta. Para hacer eso, el nodo que responde puede utilizar una ruta no vencida posible que tiene en su caché de rutas. Si no, si los enlaces bidireccionales están garantizados, la ruta se puede utilizar de nuevo hacia el origen. Si los enlaces simétricos no son soportados, el nodo puede comenzar su propio proceso de descubrimiento de la ruta y devolver el paquete de contestación de la ruta en la nueva petición de ruta.

Mientras se espera la finalización del descubrimiento de la ruta, el terminal puede continuar el proceso normal y puede enviar y recibir los paquetes con otros terminales. Este puede proteger el paquete original para transmitirlo una vez que la ruta

se ha aprendido al descubrirse, o puede desechar el paquete, confiando en el software del protocolo de la capa superior para retransmitir el paquete si éste es requerido. A cada entrada en la caché de ruta se asocia un período de expiración, después del cual la entrada se suprime de la caché. Mientras que un terminal está utilizando cualquier ruta del origen, él supervisa la operación correcta continuada de esa ruta. Por ejemplo, si el remitente, el destino, o uno de los otros terminales nombrados como saltos a lo largo de una ruta se mueve fuera del rango de transmisión inalámbrica del salto siguiente o anterior a lo largo de la ruta, la ruta no se puede utilizar más para alcanzar el destino. Una ruta tampoco funcionará más si los terminales a lo largo de la ruta fallan o se apagan. Esta supervisión de la correcta operación de una ruta en uso se puede llamar mantenimiento de la ruta. Cuando el mantenimiento de la ruta detecta un problema con una ruta en uso, el descubrimiento de la ruta se puede utilizar otra vez para descubrir una nueva y correcta al destino.

ZRP

El protocolo de la encaminamiento Zone Routing (ZRP) [24] es un protocolo proactivo y reactivo híbrido para redes Ad hoc que intenta combinar las ventajas de ambas estrategias. Divide la red en varias zonas de encaminamiento y especifica dos protocolos separados que funcionan dentro y entre las zonas de encaminamiento. Dentro de su zona de encaminamiento, que esta definida por el conjunto de nodos alrededor de un nodo dado que se puede alcanzar con un radio indicado por un número de saltos k, cada nodo emplea el protocolo de encaminamiento del interior de la zona (IARP). Este protocolo trabaja de una manera proactiva, para poder encontrar rutas muy rápidamente dentro de la zona de encaminamiento. Sin embargo, el protocolo particular que tiene que ser utilizado no se especifica, así que cualquier protocolo guiado por tabla es válido (AODV, OSPF, encaminamiento link-state). Esto permite que las diferentes zonas de encaminamiento utilicen varios protocolos, lo que puede ser un problema en algunas situaciones.

Cuando un origen no tiene una ruta de IARP a su destinación, invoca un protocolo reactivo de encaminamiento entre-zonas (Interzone Routing Protocol) (IERP), que se encarga de encontrar rutas entre distintas zonas de encaminamiento. El protocolo emplea un servicio llamado “bordercasting”, que dirige la petición de ruta (Route Request) desde un nodo a sus nodos fronterizos (ésos que están en el límite de su zona de encaminamiento, es decir, a una distancia de saltos k) por multicasting. Cuando el nodo fronterizo recibe la petición, puede consultar su información de encaminamiento IARP. Si el destino no está en la zona de la frontera del nodo, agrega su número de identificación a la petición y lo difunde otra vez. Finalmente, la petición de ruta alcanza la zona de encaminamiento que contiene el destino, el encaminamiento en origen se utiliza, en el sentido que cada nodo mencionado (en el paquete de petición de ruta) tiene una ruta IARP hacia el elemento siguiente y anterior en la ruta del origen. Las zonas de encaminamiento también ayudan a mejorar la calidad y la supervivencia de las rutas descubiertas, haciéndolas más robustas hacia los cambios en topología de la red. Las zonas de encaminando ofrecen un mantenimiento de la ruta realzado y en tiempo real, una vez han sido descubiertas. Rutas de multisalto dentro de la zona de encaminamiento pueden puentear fallos de enlace. Semejantemente, los segmentos suboptimales de la ruta pueden ser identificados y el tráfico se puede reencaminar a lo largo de rutas más cortas.

El problema principal que este protocolo tiene es la naturaleza estática que la elección del radio de las zonas tiene. El parámetro k se puede ajustar a las condiciones

operacionales de la red actual, pero esto no se puede hacer dinámicamente, lo que puede ser un problema con alta movilidad.

CBRP

El mecanismo básico CBRP [25] se usa para dividir los nodos de una red Ad hoc en clusters disjuntos de una manera distribuida. Un nodo se selecciona como cabeza del cluster para cada cluster y mantiene la información de los miembros para el cluster. Las rutas dentro de un cluster son descubiertas dinámicamente usando la información de los miembros. Dividiendo los nodos en grupos, el protocolo minimiza eficientemente el tráfico que inunda durante el descubrimiento de ruta y acelera este proceso también. Además, el protocolo considera la existencia de enlaces unidireccionales y utiliza estos enlaces tanto para el encaminamiento intra-cluster como el encaminamiento inter-cluster.

CBRP tiene las características siguientes:

- Operación completamente distribuida.
- Menos tráfico inundante durante el proceso de descubrimiento dinámico de ruta.
- Explotación explícita de los enlaces unidireccionales que serían en otro caso no usados.
- Las rutas rotas se podrían reparar localmente sin redescubrimiento.
- Las rutas suboptimales podrían ser acortadas tal como han sido usadas.

Las operaciones de CBRP se distribuyen completamente. Los componentes principales del protocolo son: Formación del cluster, descubrimiento y encaminamiento del cluster adyacente.

La meta de la formación del cluster es imponer una cierta clase de estructura de la jerarquía en una red Ad hoc totalmente desorganizada de otra manera. El algoritmo es una variación del algoritmo de clustering de identificación simple más baja en el cual el nodo con el ID más bajo entre sus vecinos se elige como la cabeza del cluster.

El objetivo del descubrimiento del cluster adyacente es que el cluster descubra todos sus clusters adyacentes con enlaces bidireccionales. Para este propósito, cada nodo mantiene una tabla de la adyacencia del cluster (CAT) que registra información de todos los cabezas de sus clusters vecinos.

El encaminamiento en CBRP se basa en el encaminamiento en el origen. Puede ser visto como consistente en dos fases: descubrimiento de la ruta y el encaminamiento real de los paquetes. La estructura del cluster se explota para minimizar el tráfico que inunda durante fase de descubrimiento de la ruta. Por otra parte, ciertos enlaces unidireccionales han sido descubiertos y usados, aumentando así la conectividad de la red.

Otras Soluciones

Varios protocolos han sido propuestos específicamente para redes Ad hoc. Además de los protocolos descritos arriba, que son algunos de los mas comunes, otras soluciones aparecen en libros.

Así, entre los protocolos de encaminamiento planos guiados por tabla, tenemos el protocolo Optimized Link State Routing(OLSR) [26], el protocolo Wireless Routing (WRP) [27], el protocolo Global State Routing (GSR) [28] y el protocolo Fisheye State Routing (FSR) [29]. También algunos protocolos jerárquicos, tales como el protocolo Hierarchical Routing (HSR) [30], el protocolo Zone-based Hierarchical Link State Routing (ZHLS) [31] y el protocolo Clusterhead Gateway Switch Routing (CGSR) [32], son guiados por tabla.

Por otra parte, el mecanismo bajo demanda para la construcción de las rutas es empleado por Lightweight Mobile Routing (LMR) [33], el Temporary Ordered Routing Algorithm (TORA) [34], el protocolo Associativity Based Routing (ABR) [35] y el protocolo Signal Stability Routing (SSR) [36], entre otros.

2.2.6.6- COMPARACIÓN ENTRE PROTOCOLOS

En la tabla 2.2, los cinco protocolos que hemos descrito arriba son comparados considerando una ciertas características importantes de los protocolos de encaminamiento Ad hoc, de modo que demuestre cómo funcionan estos protocolos, en lo referente a estas características.

Como se precisa en la tabla 2.2, la conservación de la energía o la calidad del servicio no son soportadas por estos protocolos. De todos modos, hay otros protocolos diseñados específicamente para soportar estas características. También, estas características serán incorporadas probablemente a algunas de las soluciones que definen el trabajo en marcha realizado en el grupo MANET de IETF.

Todos los protocolos funcionan de una manera distribuida, así que pueden adaptarse fácilmente a los cambios de la topología y los fallos puntuales en la red (como cuando un enlace o un nodo viene abajo) no llegan a ser críticos.

Tabla 2.2: Comparación entre algunos protocolos de encaminamiento Ad hoc significativos.

	DSDV	AODV	DSR	ZRP	CBRP
Libre de bucles	Sí	Sí	Sí	Sí	Sí
Rutas múltiples	No	No	Sí	No	Sí
Distribuido	Sí	Sí	Sí	Sí	Sí
Reactivo	No	Sí	Sí	Parcial	Sí
Enlace Unidireccional	No	No	Sí	No	Sí
QoS	No	No	No	No	No
Multicast	No	Sí	No	No	No
Seguridad	No	No	No	No	No
Conservación de Energía	No	No	No	No	No
Difusiones Periódicas	Sí	Sí	No	Sí	Sí
Requiere datos confiables	No	No	No	No	No

DSDV es el único protocolo guiado por tabla en esta comparación. viene de los protocolos usados en redes cableadas, pero incluye números de secuencia para garantizar las rutas sin bucles. El problema principal que tiene es el tiempo de la convergencia, de modo que tarda un tiempo considerable en reaccionar a los cambios de la topología, que pueden ser frecuentes en un contexto de la movilidad alto. Eso es

porque AODV, la versión reactiva de DSDV, fue diseñado teniendo en cuenta estas consideraciones. Por otra parte, los autores incluyeron las capacidades del multicast, útiles cuando las comunicaciones llegan a ser punto-a-múltiples puntos. AODV y DSR emplean mecanismos de descubrimiento de ruta cuando una ruta nueva tiene que ser encontrada, puesto que están utilizando una solución reactiva del encaminamiento. La diferencia es que AODV utiliza la información en las tablas de encaminamiento de los nodos intermedios para encaminar los paquetes, mientras que DSR se basa en el encaminamiento en el origen, así aprenderá más rutas que AODV, aunque añade gastos indirectos importantes a cada paquete. DSR también tiene la ventaja que soporta enlaces unidireccionales.

ZRP y CBRP son propuestas muy interesantes en términos de escalabilidad de la red, puesto que dividen la red en varias particiones. Dentro de las zonas/clusters utilizan un esquema proactivo y estando entre las zonas/clusters funcionan con una aproximación bajo demanda, como AODV y DSR. La manera en que se divide la red establece la diferencia principal entre ambas soluciones.

Ninguno de los protocolos presentados tienen en cuenta la carga del tráfico en el encaminamiento, así que ellos no realizan ningún mecanismo posible para adaptarse a los requisitos del tráfico. Los criterios empleados para definir las métricas del encaminamiento consideran el número más corto de saltos y el tiempo de respuesta más rápido una petición.

Más allá de este análisis comparativo, podemos concluir que las continuas mejoras en los protocolos actuales y las nuevas propuestas dentro de la comunidad que investiga las redes Ad hoc está conduciendo para alcanzar protocolos cada vez más sofisticados, así como tareas básicas de encaminamiento para redes Ad hoc se están entendiendo bien. Sin embargo, no está claro todavía si un protocolo puede resolver las necesidades diversas de todos los contextos de la red Ad hoc. En cuanto a la dirección para alcanzar ése objetivo, la solución debe emplear probablemente las aproximaciones de encaminamiento híbridos, que parecen poder adaptarse mejor a diversas situaciones. Por supuesto, cada posibilidad tiene que ser evaluada en términos de escalabilidad también. De todas formas, el desarrollo se está haciendo (y debería seguir haciéndose aún más) para realzar el funcionamiento, la escalabilidad y algunas partes particulares e importantes tales como seguridad, gestión de la energía y calidad del servicio.

2.3 – DESCRIPCIÓN DEL PROBLEMA DE AUTOCONFIGURACIÓN

Para que un nodo pueda comunicarse en cualquier red, deberá poseer un identificador único, que usualmente es la dirección IP. En redes cableadas, la dirección IP puede ser configurada manualmente por un administrador de la red o puede ser asignada a través del protocolo DHCP. Sin embargo, la utilización del protocolo DHCP requiere un servidor centralizado para la distribución de informaciones como dirección IP, máscara de red, gateway y otras informaciones adicionales de red. En redes *Ad hoc* es difícil garantizar el acceso a un servidor, ya que los nodos pueden moverse libremente alterando frecuentemente la topología de la red. Así pues, es deseable realizar la tarea de configuración de los nodos de forma dinámica, automática y preferentemente sin ninguna intervención humana.

2.4 – NECESIDADES DEL PROTOCOLO DE AUTOCONFIGURACIÓN

Tabla 2.3 – Necesidades en un protocolo de configuración.

Necesidades	Descripción
Unicidad de las direcciones IP	Asegurar que dos o más nodos no obtengan la misma dirección IP.
Correcto funcionamiento	Una dirección IP está asociada a un nodo solamente por el tiempo que permanece en la red. Cuando un nodo deja la red, su dirección IP debe quedarse disponible para ser asociada a otros nodos.
Solucionar problemas debido a perdida de mensajes	En caso de que algún nodo falle u ocurra perdida de mensajes, el protocolo debe actuar suficientemente rápido como para que no ocurra que dos, o más nodos posean la misma dirección IP.
Permitir el encaminamiento multi-hop	Un nodo no será configurado con una dirección IP cuando no hubiera ninguna dirección IP disponible en toda la red. Siendo así, si cualquier nodo de la red posee una dirección IP libre, esta dirección debe ser asociada al nodo que está solicitando una dirección IP, aunque esté la dos saltos o más de distancia.
Minimizar el tráfico de paquetes adicionales en la red	El protocolo debe minimizar el número de paquetes intercambiados entre los nodos en el proceso de autoconfiguración de modo que no perjudique el tráfico de paquetes de datos afectando así el rendimiento de la red Ad hoc.
Verificar la ocurrencia de solicitudes concurrentes de dirección IP	Cuando dos nodos solicitan una dirección IP en el mismo instante de tiempo, el protocolo debe realizar el tratamiento para que no sea suministrada la misma dirección IP para los dos nodos.
Ser flexible al particionamento y la fusión de redes Ad hoc	El protocolo debe manipular la fusión de dos redes Ad hoc distintas así como el particionamento en dos o menores redes.
Realizar el proceso de sincronización.	El protocolo debe adaptar a los rápidos cambios de la topología de las redes inalámbricas debido a la movilidad frecuente de los nodos. La sincronización se realiza periódicamente para mantener una configuración actualizada de la topología de la red.

Algunos autores e investigadores argumentan que la dirección Medía Access Control (MAC) puede ser usado como identificador único de un nodo en la red. Sin embargo, la unicidad de la dirección MAC ni siempre es garantizada ya que es posible cambiarlo a través de comandos propietarios del Linux como el ifconfig. Siendo así, en ese trabajo, se propone un protocolo de configuración dinámica de los nodos atendiendo todas las necesidades arriba citadas, garantizando la unicidad de la identificación de un nodo en la red Ad hoc. La identificación única se garantiza a través del uso de certificados digitales distribuidos, lo que hace el protocolo totalmente seguro y confiable, necesitando que los nodos se autentifiquen antes de que se hagan un miembro de la red Ad hoc.

2.5 – CLASIFICACIÓN DE LOS PROTOCOLOS

Varios protocolos fueron propuestos para atender las necesidades de la autoconfiguración. Cabe Destacar que ninguno de los protocolos existentes aún ha sido patentado. Todos están en formato de draft (IETF) o como artículos presentados en los congresos más importantes del área.

Los protocolos de autoconfiguración pueden ser clasificados de dos formas. Una forma es con relación al proceso de autoconfiguración y la segunda forma respecto a los mecanismos utilizados para la detección de direcciones duplicadas, fijándose en cómo y cuando esas direcciones duplicadas se detectan. A continuación se presentan las dos clasificaciones.

2.5.1 – EN CUANTO AL PROCESO DE AUTOCONFIGURACIÓN

Independente (*Stateless*)

Autoconfiguración Independiente permite que el nodo construya su propia dirección IP, basado o en el identificador del hardware o en un número aleatorio. Ese proceso no depende de una segunda entidad para hacer la autoconfiguración. Después de la construcción de la dirección IP, necesitamos un mecanismo de detección de direcciones duplicadas para asegurar la unicidad de la dirección generada.

Dependiente (*Stateful*)

Autoconfiguración Dependiente requiere que cada nodo en la red mantenga un conjunto de direcciones IP. Eso implica en la necesidad de participación de una segunda entidad en el proceso de asociación de una nueva dirección IP. Además de eso, el mantenimiento de una estructura común y distribuida entre todos los nodos de la red, requiere consumo de anchura de banda principalmente en la presencia frecuente de fusión y separación de redes *Ad hoc*. Este el modelo con el que nosotros trabajaremos.

2.5.2 – EN CUANTO AL PROCESO DE DETECCIÓN DE DIRECCIONES DUPLICADAS

Asignación para Detección de Conflictos (*Conflict-Detection Allocation*)

El mecanismo de detección de conflictos adopta la política “trial and error”. O sea, el nodo escoge una dirección IP por tentativa y hace un requisito esperando por la aprobación de todos los nodos de la red Ad hoc. Si algún nodo en la red responde negativamente, significa que esa dirección IP ya está siendo usada. Siendo así el proceso es repetido hasta que el nodo encuentre una dirección IP que no esté siendo usada por ningún nodo de la red.

Asignación Libre de Conflictos (*Conflict-Free Allocation*)

Ese método de asignación de direcciones IP usa el concepto de división binaria, que significa que cada nodo posee conjuntos de direcciones IP distinguidos. Cada nodo puede asociar una dirección IP sin la necesidad de consultar otros nodos para obtenerse

aprobación. Así, todos los nodos de la red son responsables por el proceso de asociación de una dirección IP. Ese mecanismo posee la ventaja de no necesitar de difusión para asociar una dirección IP, economizando el ancho de banda de la red.

Asignación de Mejor Esfuerzo (*Best-Effort Allocation*)

En este protocolo, los nodos de la red son responsables de la asociación de direcciones IP para los nuevos nodos, intentando asociar una dirección libre que no esté siendo usado por ningún nodo en la red. Todos los nodos de la red mantienen una tabla de las direcciones IP que están en uso o libres en la red. Así, cuando un nuevo nodo llega a la red, su vecino más próximo va a escoger una dirección IP libre para asociar a él. El único problema es que dos o más nodos pueden llegar simultáneamente y los nodos pueden ofrecer la misma dirección IP. La ventaja de ese protocolo es que funciona muy bien con protocolos de encaminamiento pro-activo, pues los nodos frecuentemente realizan difusión con las informaciones de las direcciones ya usadas en la red.

Algunos autores clasifican los mecanismos de detección de direcciones duplicadas de una otra manera. En [37] los protocolos son clasificados como activos y pasivos. Los protocolos activos son aquellos que distribuyen información adicional en la red, siendo necesarios paquetes adicionales de control para que el protocolo funcione perfectamente. Ya que los protocolos pasivos son aquellos que detectan direcciones duplicadas, pero sin la necesidad de diseminar paquetes adicionales de control en la red. Todo se hace sólo monitorizando el tráfico del protocolo de encaminamiento. Sin embargo, ese último método tiene un periodo de tiempo en que puede haber entrega de paquetes para el destino errado, conocido como periodo de vulnerabilidad.

2.6 – TRABAJOS RELACIONADOS

Se describen abajo, los protocolos ya existentes citando los mecanismos utilizados para la autoconfiguración de direcciones, los procedimientos necesarios para el tratamiento de partición y fusión de redes *Ad hoc* citando sus ventajas y desventajas. Existen más de diez protocolos existentes, sin embargo, se citan aquí solamente los más importantes y que están relacionados al encaminamiento del nodo utilizando direcciones IPv4.

2.6.1 - IP ADDRESS AUTOCONFIGURATION FOR AD HOC NETWORKS

Funcionamiento:

Un nodo realizando el proceso de autoconfiguración utiliza dos direcciones IP. La primera dirección es una dirección aleatoria y temporal asignada utilizando la franja de direcciones 169.254/16 entre las direcciones 0 y 2047. Este es la dirección que servirá temporalmente para la comunicación con otros nodos de la red. El nodo entonces selecciona una segunda dirección en la franja entre 2048 y 65534 que es la dirección deseada para ser usada en la red.

Después de la obtención de los dos direcciones, el nodo envía un mensaje Address Request (AREQ) para chequear si algún nodo ya configurado en la red posee la dirección que él está deseando. Si algún nodo posee la dirección solicitada, envía un mensaje Address Reply (AREP) informando de que esa dirección ya está siendo utilizada en la red. En ese caso, el nodo selecciona otra dirección aleatoriamente y realiza el proceso nuevamente. Si el nodo que desea asociarse a la red no recibe ninguna respuesta, significa que ningún nodo de la red posee la dirección IP, siendo así, asocia esa dirección IP a su interfaz.

Consideraciones sobre el protocolo

El mecanismo de Duplicate Address Detection (DAD) se realiza a través de inundación (flooding) para encontrar la dirección deseada, no siendo adecuado para redes de ancha escala o cuando hay la fusión de dos o más redes Ad hoc.

El protocolo utiliza tiempos constates para el proceso de detección de direcciones duplicadas y ese tiempo puede no ser el suficiente para redes Ad hoc multi-hop cuando los nodos están muy lejos uno del otro. Se debe usar un tiempo límite en función del diámetro de la red Ad hoc, utilizando O(n) donde n es el número de nodos.

Puede ocurrir que dos o más nodos seleccionen la misma dirección IP en la franja de direcciones de 1-2047 y eso está fuera del objetivo del protocolo, alegando que el tiempo que los nodos se quedan con esas direcciones temporales es limitado.

2.6.2 – AD HOC IP ADDRESS AUTOCONFIGURATION

Funcionamiento:

Ese protocolo trabaja con dos mecanismos de detección de direcciones duplicadas. El primer mecanismo es llamado de “Fuerte detección de direcciones duplicadas” (Strong Duplicate Address Detection) y el segundo mecanismo es llamado

de “Débil detección de direcciones duplicadas” (Weak Duplicate Address Detection) que utiliza los mensajes del protocolo de enrutado para chequear la unicidad de las direcciones.

El algoritmo Strong DAD esta basado en el mismo esquema utilizado en [38], asociando tiempos limitados para el cambio de mensajes. Utilizar tiempos determinados y limitados en el cambio de mensajes puede ser impracticable en redes de ancha escala. Ya el algoritmo Weak DAD utiliza una forma diferente de hacer la detección de direcciones duplicadas. Es utilizado una dirección virtual que es la combinación de una dirección IP con una llave única pre-asociada. Esta llave debe ser única dentro de la red y puede ser la dirección MAC de la interfaz, así como puede ser una derivación del nombre del fabricante juntamente con el número de serie, u otra forma cualquiera desde que sea único. Dada esa llave única, una única dirección IP puede ser creada simplemente combinando la llave con la dirección IP. Entonces, es con el uso de esas llaves que este protocolo hace la detección de las direcciones duplicadas. No fueron hechos cambios en la cabecera IP, solamente en los protocolos de enrutado utilizados, acordando que las decisiones de encaminamiento de paquetes están hechas basadas solamente en la dirección IP de destino de la cabecera de los paquetes IP.

Sea un nodo recibiendo un paquete que incluye (IP_x, key_x). Entonces ese nodo chequea su tabla de enrutado para ver si contiene la dirección IP_x. Asumiendo que hay una entrada en la tabla para esa dirección, verifica si la key_x del paquete recibido es la misma que él guarda en su tabla de enrutado. Si no fuera así, él concluye que hay dos nodos con la misma dirección IP en la red *Ad hoc*, sin embargo con dos llaves distinguidas. En ese punto el nodo invalida el paquete recibido y propaga esa información para los nodos vecinos para que todos sepan que hay dos nodos utilizando la misma dirección IP.

Consideraciones sobre el protocolo

El protocolo tiene el objetivo la disminución del número de mensajes intercambiados entre los nodos, por estar integrado con el protocolo de enrutado.

Debido a la necesidad de adicional anchura de banda para la distribución de las llaves, puede ocurrir que dos nodos escojan la misma dirección y también la misma llave, visto que la llave es generada solamente una vez en cada nodo.

2.6.3 - IP ADDRESS ASSIGNMENT IN A MOBILE AD HOC NETWORK

Funcionamiento:

En ese protocolo de autoconfiguración, cada nodo posee un conjunto de direcciones IP que son usadas para configurar nuevos nodos que llegan a la red sin necesitar consultar cualquiera otro nodo ya configurado de la red. Dentro de una misma red *Ad hoc*, esos conjuntos son distinguidos. Un nodo que desea asociarse a una red (nodo cliente), hace una petición de una dirección IP. Uno de los nodos ya configurados en la red (nodo servidor) responde la solicitud del nodo cliente y se queda responsable de la autoconfiguración de la dirección IP. Así, el nodo servidor divide su conjunto de direcciones IP en dos mitades. Entonces, él envía una mitad para el nodo cliente y mantiene la otra mitad consigo para atender futuros requisitos. Cuando el cliente recibe ese conjunto de direcciones, asocia el primero a sí mismo y mantiene el resto como un

conjunto de direcciones disponibles. Hecho eso, el nodo cliente envía un mensaje confirmando el éxito de la operación.

Consideraciones sobre el protocolo

El protocolo trabaja muy bien con fusión y separación de red *Ad hoc* por mantener conjuntos distinguidos de direcciones IP en los nodos configurados de la red.

En el protocolo, se tratan las pérdidas de mensajes así como el fallo de los nodos evitando que dos nodos tengan la misma dirección IP en un determinado momento.

Ningún nivel de seguridad se menciona en el protocolo asumiendo que todos los nodos son confiables.

El proceso de sincronización depende de un proceso de difusión (broadcast) confiable. Eso no siempre es alcanzable debido al ambiente distribuido existente en las redes móviles *Ad hoc*.

2.6.4 – MANETconf: CONFIGURATION OF HOST IN A MANET

Funcionamiento:

En ese protocolo, un nodo que solicita una dirección IP envía un mensaje a sus vecinos que ya forman parte de la red. Uno de esos vecinos responde la solicitud e inicia el proceso de asociación de la dirección IP. Ese nodo ya configurado en la red tiene que obtener una aprobación de todos los otros nodos de la red para que la asociación de esa dirección IP se concluya. Si el nodo configurado reciba algún mensaje negando la configuración, el proceso no se concluirá alegando que esa dirección ya está siendo usada por un nodo. Cada nodo en la red mantiene una estructura de datos que contiene las direcciones IP asociadas y las direcciones IP que aún están pendientes por estar en proceso de autoconfiguración.

Consideraciones sobre el protocolo

El protocolo trabaja muy bien con fusión y separación de redes *Ad hoc*.

Una de sus mayores desventajas es que para cada proceso de asociación de una dirección IP, el protocolo requiere el envío de mensajes de difusión, acarreando problemas de escalabilidad.

2.6.5 – PASSIVE DUPLICATE ADDRESS DETECTION IN MOBILE AD HOC NETWORKS

Funcionamiento:

Se usa en ese protocolo, un mecanismo de detección de direcciones duplicadas, diferente a todos los protocolos citados anteriormente. La detección de direcciones duplicadas se hace de una forma pasiva, realizada solamente por el seguimiento del tráfico del protocolo de enrutado. Basándose en protocolos de enrutado pro-activo, se emplean tres formas diferentes. Una forma es utilizar el número de secuencia usado por los protocolos de enrutado. La segunda forma esta basada en el principio de la

localización, debido al hecho que en los protocolos de enrutado pro-activos, los nodos se mueven con velocidad limitada. Ya en la tercera forma, la vecindad es la característica a ser explorada visto que un nodo conoce toda su vecindad y también la vecindad del origen del paquete de enrutado.

Consideraciones sobre el protocolo

Para trabajar con fusión y separación de redes *Ad hoc*, el protocolo necesita mecanismos adicionales.

La elección adecuada de un tiempo para que el mensaje viaje por toda la red es una tarea difícil visto que si la elección fuese hecha erróneamente, pueden ocurrir direcciones duplicadas o el protocolo puede avisar que existen direcciones duplicadas sin existir.

Sin embargo, es evidente que ningún protocolo satisface todos los requisitos citados en la sección (2.4). La seguridad es el mayor problema al que se enfrenta en todos los protocolos existentes, ya que en ningún protocolo se menciona ningún nivel de seguridad.

2.7 – MÉTRICAS PARA LA EVALUACIÓN DEL RENDIMIENTO

En éste apartado, se citan las métricas más importantes para analizar el rendimiento de un protocolo de autoconfiguración en redes *Ad hoc* y un comparativo mostrando cuáles tienen mayor influencia en la eficacia de un protocolo de autoconfiguración.

Operación Distribuida

Un nodo en una red Ad hoc no es tan confiable como un servidor DHCP debido a la movilidad, área de transmisión limitada, caída de los nodos, suministro de energía limitado y otros factores. El fallo de algunos nodos no debe impedir al servicio de autoconfiguración funcionar. Siendo así, el algoritmo de configuración debe ser diseñado para funcionar de forma distribuida.

Exactitud

Dos o más nodos no pueden poseer la misma dirección IP, y aunque la posean debe ser por el menor periodo de tiempo posible, o sea, el algoritmo de detección de direcciones duplicadas debe ser ejecutado lo más rápido posible, generando el menor procesamiento en la red.

Complejidad

Llevando en cuenta la cantidad limitada de memoria y un bajo poder computacional de los nodos móviles, la solución debe ser la más sencilla posible, pudiendo consistir en módulos de asignación de direcciones, detección de direcciones duplicadas y mantenimiento de las tablas de estado de los nodos. La complejidad de cada módulo debe ser cuidadosamente pensada. Es viable una división de tareas para que los módulos interaccionen sin que haya un adicional procesamiento en la red.

Comunicación excesiva

La difusión es extremadamente consumidora de banda y debe ser evitada todo lo posible. Una solución viable es la de comunicarse solamente entre nodos vecinos. Por lo tanto, los protocolos que no necesitan de una segunda entidad para hacer la autoconfiguración son más eficaces, pues cada nodo puede hacer el proceso de autoconfiguración aisladamente.

Igualdad

La distribución de direcciones IP debe ser igual, o sea, debe ser justa para evitar que la duplicidad de direcciones ocurra. Así, si el protocolo de autoconfiguración distribuye de forma injusta las direcciones, existirá la necesidad de iniciar varias veces el mecanismo de detección de direcciones duplicadas, que llevará a un creciente número de paquetes de control en la red. Si el protocolo funciona perfectamente y distribuye uniformemente las direcciones, la probabilidad de conflictos es baja, resultando un menor procesamiento en la red.

Latencia

Latencia es el tiempo entre el inicio del proceso de autoconfiguración hasta el final, cuando se le asocia una dirección IP libre a un nuevo nodo. Cuanto menor es el número de mensajes en difusión, menor la latencia, así, la comunicación local entre vecinos es la deseable.

Escalabilidad

La anchura de banda consumida por los mensajes de difusión está relacionada al número de nodos en la red Ad hoc. La latencia es proporcional al tamaño de la red, pues cuanto mayor es la red, mayor será el tiempo para asociar una dirección IP a un nuevo nodo. Por lo tanto, si se necesita difusión para la autoconfiguración, podemos decir que el protocolo posee una baja escalabilidad. Si las comunicaciones ocurren entre nodos vecinos y localmente, el protocolo tiene alta escalabilidad.

Todas esas métricas están íntimamente conectadas. Mientras más uniforme la distribución de las direcciones y menor el cambio de mensajes en la red, menor será la latencia y mayor será la escalabilidad. Así, la uniformidad en la distribución de las direcciones, la sobrecarga en la red a través del cambio de mensajes y la latencia son las métricas más importantes en la evaluación de un protocolo de autoconfiguración.

2.7.1 – COMPARACIÓN DE RENDIMIENTO

La Tabla 2.4 presenta un cuadro comparativo de las métricas para la clasificación de los protocolos descritos. Se han de considerar las siguientes variables para comparar el rendimiento de los protocolos de autoconfiguración:

n – número de nodos
l – número de enlaces
t – media de tiempo de transmisión entre dos nodos adyacentes
d – diámetro de la red
k – tiempo de respuesta

Tabla 2.4 – Características importantes de las redes *Ad hoc*.

Característica	Detección de conflictos	Libre de conflictos	Mejor esfuerzo
Organización de la red	Jerárquica	Plana	Jerárquica / Plana
Autoconfiguración	Independiente	Dependiente	Dependiente
Conflicto de direcciones	Si	No	Si
Recuperación de direcciones	Innecesario	Necesario	Necesario
Complejidad	Baja	Media	Alta
Procesamiento	O((n+l)k)	O(2 l/n)	O((n+l)k)
Uniformidad	Si	No	Si
Latencia	O(2tdk)	O(2t)	O(2tdk)
Escalabilidad	Pequeña	Media	Pequeña

Asignación para Detección de Conflictos es el método más simple. Ningún estado de los nodos se almacena. No se necesita ningún mensaje de difusión para la recuperación de direcciones inutilizadas. Sin embargo, difusión se adopta para realizar la detección de direcciones duplicadas, generando excesiva comunicación, alta latencia y baja escalabilidad.

Asignación Libre de Conflictos utiliza un simple mecanismo de autoconfiguración. Sin embargo, su problema reside en la gerencia del conjunto de direcciones. Si un nodo notifica su salida antes de dejar la red, él libera su dirección IP y su conjunto de direcciones IP, y avisa a todos los nodos en la red. Sin embargo si el nodo deja la red abruptamente y no envía ningún mensaje a los nodos vecinos, llevará consigo el conjunto de direcciones IP que momentáneamente no podrá ser usado por otros nodos que desean asociarse a la red. Así, se necesita un mecanismo para controlar el conjunto de direcciones y se hace más difícil y complicado que el propio mecanismo de autoconfiguración. Como el cambio de informaciones es solamente entre nodos vecinos, el protocolo posee poco procesamiento, baja latencia y media escalabilidad.

Asignación del Mejor Esfuerzo, se espera que su rendimiento sea semejante al del algoritmo “Asignación para Detección de Conflictos”: procesamiento excesivo en la red, distribución uniforme de las direcciones IP, alta latencia y baja escalabilidad. Sin embargo, debido al mantenimiento de los estados de los nodos, la complejidad es mayor causada por el exceso de mensajes necesarios para el mantenimiento y sincronización de los estados de los nodos.

Con base en el análisis arriba, se llega la conclusión de que un óptimo algoritmo de autoconfiguración debe satisfacer la dos propiedades para encontrar baja latencia y alta escalabilidad:

- Comunicación local. La comunicación debe ser realizada entre nodos vecinos por generar menos cambio de mensajes en la red.

- Asociación aleatoria. Lleva la una distribución más uniforme de las direcciones en la red.

Conforme se fue observado en este capítulo, hay algunas necesidades que los protocolos de autoconfiguración deben atender. Además de eso, para obtener un protocolo eficaz, se debe estar atento a las métricas citadas a fin de encontrar baja latencia y una alta escalabilidad.

3 - ESPECIFICACION DEL PROTOCOLO

En este capítulo, mostraremos todas las funciones del protocolo de autoconfiguración en redes móviles ad hoc. Las funcionalidades se explican detalladamente a fin de hacer más sencilla su implementación. Inicialmente, se hace referencia el objetivo principal del protocolo. A continuación mostraremos los escenarios que se pueden producir debido a la constante movilidad en la topología de las redes móviles ad hoc. Se citan también las estructuras de datos utilizadas así como los mensajes intercambiados por los nodos para el proceso de autoconfiguración de direcciones IP. Los temporizadores utilizados en el intercambio de mensajes entre los nodos también es de suma importancia para asegurar el correcto funcionamiento del protocolo. Al final del capitulo se explica detalladamente el funcionamiento del protocolo con un ejemplo tanto para la entrada como para la salida de nodos con su correspondiente sincronización.

3.1 - OBJETIVOS

El protocolo especificado utiliza los mecanismos de división binaria y considera cuestiones como partición y unión de redes ad hoc, baja de nodos y pérdida de mensajes atendiendo a los requisitos citados en la Sección 2.4. El objetivo principal del protocolo diseñado es realizar este proceso de un modo efectivo y teniendo en cuenta la sincronización de nodos.

El protocolo de autoconfiguración propuesto y basado en el trabajo hecho en [41] con las mejoras en [39]. Se tratan todos los casos de entrada y salida de nodos que modifican la topología de la red. De tal modo que se pueda actualizar toda la información de los nodos pertenecientes a la red. Y para así tener en cada instante datos actualizados de la situación actual de la red.

3.2 – FUNCIONAMIENTO

Inicialmente, en el caso de la entrada de un nodo, se considera que el nodo que desea asociarse a una red y obtener una dirección IP se llama “nodo cliente”, y el nodo que hace el proceso asociativo de direcciones y responde las peticiones se llama “nodo servidor”.

Cada nodo perteneciente a la red posee una dirección IP identificando su interface y un conjunto de direcciones libres que llamaremos de free_ip_blocks para servir a los nodos clientes que desean asociarse a la red. Dentro de una misma red ad hoc, los conjuntos de direcciones IP libres (free_ip_blocks) de los nodos deben ser disjuntos, garantizando así que dos o más nodos servidores no suministren la misma dirección IP para los nodos clientes. También, cada nodo asociado poseerá un TimeStamp-TS para identificarlo unívocamente en caso de unión de redes, y para asignarle una prioridad en el momento de asumir direcciones libres liberadas por un nodo saliente. Además de eso, cada partición lógica, o sea, cada red ad hoc posee un identificador único llamado Partition_id-PID. Así, todos los nodos que poseen la misma PID son parte de la misma red ad hoc, este identificador facilita la detección de unión y separación de redes ad hoc. En cada intercambio de mensajes de nodos configurados se debe enviar el TS del nodo y el PID de la red para distinguir unívocamente los nodos y evitar problemas de separación y unión de redes como veremos en el apartado 3.2.6.

El protocolo DCDP (Dynamic Configuration Distribution Protocol) [41] es un protocolo para la distribución de configuraciones de red como: dirección IP, máscara de red y gateway patrón basado en el modelo llamada “buddy system” propuesto en [40], que utiliza el mecanismo de división binaria para suministrar conjuntos disjuntos de direcciones IP a los nodos de la red. La figura 3.1 muestra el proceso de división binaria para el suministro de dirección IP a nuevos nodos:

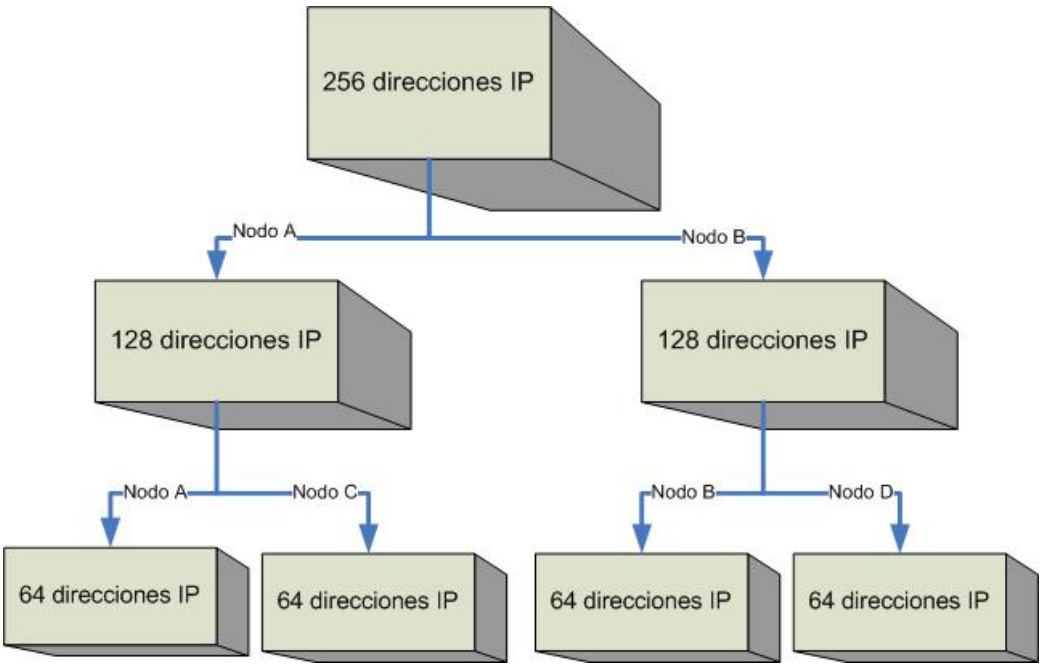


Figura 3.1 – Funcionamiento del modelo de división binaria.

Inicialmente, el nodo A posee todo el rango de direcciones IP conteniendo 256 direcciones IP, teniendo que una está asociada a su interfaz de red, restando 255

direcciones libres(free_ip_block) para asociar los otros nodos. Aplicando los métodos de división binaria, el nodo A responde una solicitación de configuración del nodo B y divide su conjunto de direcciones IP libres en dos mitades, suministrando la mitad para el nodo B y quedándose con la otra mitad. Después de esta división, el nodo A y el nodo B se quedan cada una con 128 direcciones, siendo una dirección IP para su interface de red y 127 para servir a otras peticiones. A partir del mismo proceso, el nodo A divide su conjunto de direcciones IP libres en dos mitades nuevamente suministrando la mitad para el nodo C y quedándose con la otra mitad. El mismo procedimiento ocurre con el nodo B que para atender una petición suministra la mitad de sus direcciones IP para el nodo D y se queda con la otra mitad. Este mismo proceso se usa para atender todas las nuevas peticiones hasta que todas las direcciones IP libres estén siendo usadas.

Este mecanismo de división binaria asegura que todos los nodos de la red posean conjuntos disjuntos de direcciones IP, evitando que una misma dirección IP pueda ser usada por dos o más nodos, no posibilitando el conflicto de direcciones IP y aún cuando haya una unión de dos redes ad hoc.

Con base en este modelo de distribución de direcciones y asumiendo tales suposiciones, se muestran a continuación todos los escenarios que pueden ocurrir debido a la topología dinámica de las redes ad hoc.

3.2.1 – INICIALIZACIÓN DE LA RED AD HOC

Al iniciar un nodo, este intenta conectarse a una red. Para lo cual envía mensajes address_request solicitando una dirección IP y un bloque de direcciones libres, esperando la respuesta de alguna red. Si en un número finito y definido de intentos no recibe respuesta alguna de ninguna red, creará la suya propia. Este será el comienzo de una nueva red a la que le asignará un Partition_ID en base al timestamp. Y este nodo que se convertirá en el nodo líder poseerá todas las direcciones libres de la red. Para su interfaz reservará la primera dirección libre y configurable.

3.2.2 – ASOCIACIÓN DE UNA DIRECCIÓN IP A UN NUEVO NODO

El proceso de asociación de una dirección IP se da de la siguiente manera:

Cuando un nodo desea asociarse a la red ad hoc para la obtención de una dirección IP, envía un mensaje addr_req en difusión utilizando su dirección física de la placa como dirección de origen. Cualquier nodo servidor perteneciente a la red responde a la petición del nodo cliente enviando un mensaje addr_rep. Este mensaje contiene el free_ip_block con la mayor cantidad de direcciones IP libres, pues el mismo nodo puede poseer dos o más free_ip_block con cantidades de direcciones IP diferentes. El nodo cliente puede recibir más de un mensaje proveniente de diferentes nodos servidores, entretanto, él selecciona el nodo servidor con el mayor free_ip_block enviando un mensaje server_poll específicamente para el servidor escogido, descartando los mensajes de otros servidores. De este modo el nodo servidor recibe el mensaje server_poll del nodo cliente confirmando la intención de obtener una dirección IP, este divide su free_ip_block por la mitad, suministrando una mitad al nodo cliente y quedándose la otra mitad para atender futuras peticiones.

Al recibir las direcciones IP contenidas en free_ip_block a través del mensajes ip_assigned, el nodo cliente asocia el free_ip_block recibido a su estructura

free_ip_block. En esta estructura, la primera dirección IP se asocia a su interface y las direcciones IP restantes son utilizadas para servir a otros nodos. Por motivos de seguridad y de facilidad de implementación, es necesario que el nodo marque cual free_ip_block almacena su propia dirección IP, caso el nodo posea más de dos registros en la tabla free_ip_block. Para que confirmar que la configuración fue realizada con éxito, el nodo cliente envía un mensaje ip_assignment_ok hacia el nodo servidor. A continuación informará al resto de nodos de la red su presencia en la misma. Para lo cual les enviará un mensaje Node_up las veces que fuera necesario, hasta recibir sendos Node_up_reply con información del propio nodo.

Termina aquí el proceso de asociación de una dirección IP para un nuevo nodo. Los formatos de los mensajes, así como los tiempos utilizados son descritos en las secciones siguientes. El proceso de asociación de una dirección IP a un nuevo nodo está ilustrado a través de la Figura 3.2.

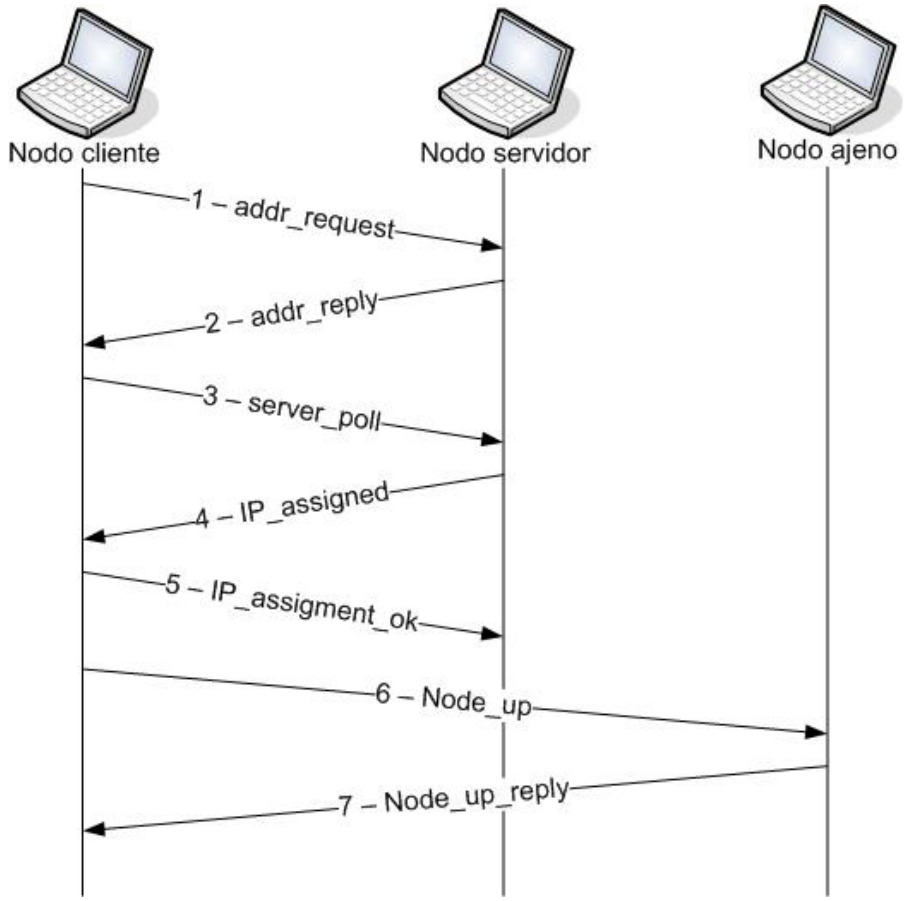


Figura 3.2 – Asociación de una dirección IP a un nuevo nodo.

Donde:

1. El cliente envía un mensaje broadcast addr_req.
2. Al recibir una petición, el nodo servidor responde enviando un mensaje addr_rep. Es posible que más nodos servidores respondan al mensaje de petición.
3. El nodo cliente selecciona sólo el nodo servidor con mayor cantidad de direcciones IP libres y envía de vuelta a este nodo servidor un mensaje server_poll, ignorando la respuesta de los otros nodos servidores.

4. Al recibir el mensaje server_poll, el nodo servidor está listo para asociar una dirección IP al nodo cliente. Divide su free_ip_block en dos mitades, enviando una mitad para el nodo cliente y guardando la otra mitad.
5. Cuando el nodo cliente recibe el mensaje ip_assigned, este aloja toda la estructura en la tabla local free_ip_block, donde la primera dirección IP se usará para la configuración de su interface y las restantes direcciones IP se utilizan para servir a otros nodos. Después de esto, el nodo cliente envía un mensaje ip_assignment_ok hacia el nodo servidor indicando que la configuración se realizó con éxito.
6. El cliente envía al resto de nodos de la red un Node_up para informar de su presencia en la red. De este modo estos podrán actualizar su información de la topología de la red.
7. Al recibir todos los mensajes Node_up_reply de confirmación, termina el proceso de autoconfiguración.

Si el nodo servidor no recibe ip_assignment_ok del nodo cliente que es un mensaje de confirmación del proceso, enviará un PING al nodo cliente para verificar si el nodo cliente fue definitivamente configurado. Si recibe alguna respuesta, implica que el proceso de autoconfiguración fue realizado con éxito, y que lo que sucedió es que el mensaje de confirmación se perdió.

Por otro lado, si el nodo cliente no recibe algún mensaje Node_up_reply, comprobará que los nodos siguen aun en la red mediante PING, y si es así reenviará el mensaje Node_up, hasta recibir respuesta.

Al recibir el free_ip_block que contiene su dirección IP, el nodo cliente y el nodo servidor pasan a ser llamados “nodos amigos”(buddy). Esta connotación dada a los nodos facilita el proceso de recuperación de direcciones IP cuando hay fallo en algún nodo o cuando ocurre pérdida de mensajes. Así, todos los nodos tienen registrado el nodo que le suministro las direcciones.

Es posible que un nodo servidor no tenga ninguna dirección IP disponible en su free_ip_block al recibir un mensaje de petición de dirección IP. La solución es encaminar la petición a los nodos vecinos en la red. Para mantener una distribución uniforme de las direcciones IP, el nodo servidor investiga en su tabla Allocated donde almacena toda la información de la red y donde puede buscar cual es el nodo que posee mayor numero de direcciones libres. De modo que repartir las direcciones del modo mas equilibrado posible, lo cual mejora el rendimiento notablemente.

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

3.2.3 – SALIDA DE NODOS

Debido a las constantes variaciones en la topología de la red como consecuencia de la movilidad de los nodos, habrá situaciones en que los nodos saldrán de la franja de comunicación de sus vecinos, este proceso será tratado como si el nodo hubiese abandonado la red ad hoc. El proceso de salida o fallo de un nodo debe ser cuidadosamente analizado con el fin de evitar problemas de distribución de direcciones IP para nuevos nodos. El protocolo esta proyectado para minimizar el trabajo del nodo cliente, permitiendo que él mismo deje la red e intercambie el menor número de mensajes posibles con sus nodos vecinos. Esto es porque el nodo que desea salir de la

red no tiene tiempo o energía suficiente para realizar el intercambio de mensajes con sus nodos vecinos.

Por tanto, habrá dos situaciones en que se considera que un nodo deje la red. La primera situación es cuando el nodo desea salir de la red y avisa a sus vecinos de su intención. La segunda forma de detectar la salida de un nodo es cuando ocurre un fallo en el nodo o el nodo deja la red sin avisar a sus vecinos. Abajo, los dos procesos son explicados detalladamente.

3.2.3.1 – SALIDA FÁCIL

Cuando un nodo desea abandonar la red lo debe notificar. Esto facilita la tarea de recuperar las direcciones que dejará libre el nodo saliente. Para ello verifica que la información de sus tablas es correcta comprobando que el padre actual, el nodo en principio responsable de sus direcciones, se encuentra en la red. Si su padre responde al PING entonces le envía un mensaje Graceful_dep y abandona la red.

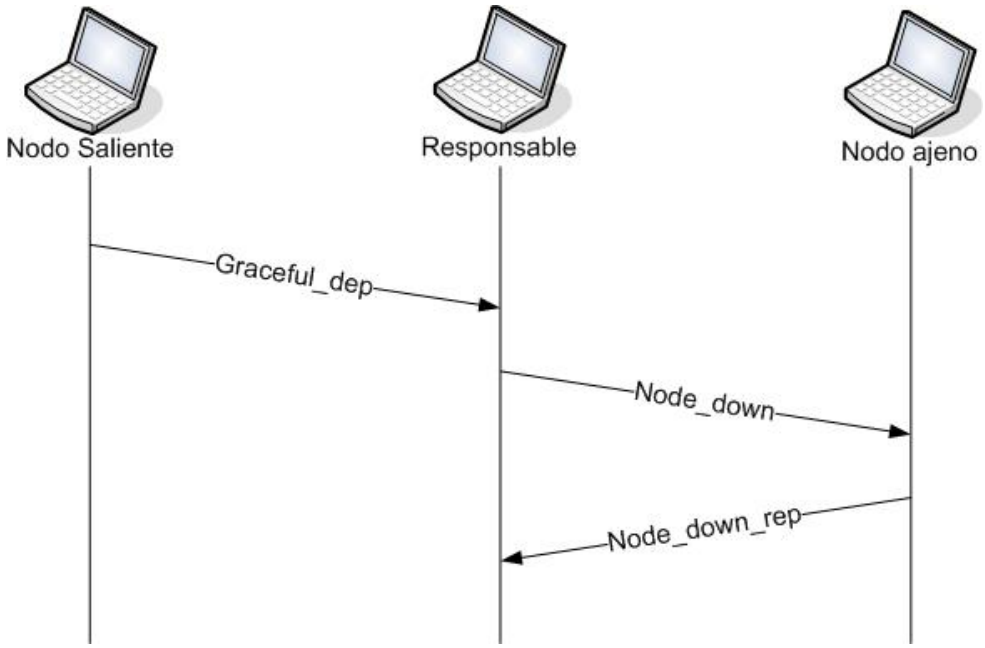


Figura 3.3 - Proceso de salida fácil de un nodo.

Si su padre no se encuentra en la red, buscará otro nodo responsable de sus direcciones, el nodo con mayor antigüedad, es decir, con mayor prioridad de la red. Al cual le deberá entregar las direcciones liberadas. Esto lo indicará con el envío del mensaje Graceful_dep. El nodo responsable al recibirlo, coge las direcciones IP y actualiza su tabla con los nuevos cambios. Después manda un mensaje Node_down a todos los nodos de la red avisando que un nodo ha dejado la red. De este modo todos actualizarán las tablas y se asegura una sincronización que da lugar a una coherencia de datos. Si algún nodo no responde a ése mensaje con Node_down_rep, se reenvía Node_down. Hasta que todos los nodos notifican ser conscientes del cambio.

Donde:

1 – El nodo que desea salir y que previamente ha comprobado la presencia del nodo destino, envía un Graceful_departure con la información de sus direcciones.

2 – Tras realizar los cambios pertinentes en las tablas, el nodo responsable notifica al resto de los nodos con mensajes Node_down.

3 – El resto de nodos confirman los cambios mediante Node_down_rep al nodo responsable, y una vez recibidos finaliza este proceso.

Acordando que la pérdida de cualquiera de los mensajes citados arriba, implica que el nodo deja la red sin avisar, hubo algún fallo o salió bruscamente de la red. Esta cuestión se trata abajo.

3.2.3.2 – SALIDA BRUSCA

Cuando el nodo que abandona la red lo hace de modo abrupto y sin avisar de ello. Puede ser porque el nodo ha caído, que porque quedó sin batería, se desconecto de un modo inesperado o simplemente porque el nodo se ha movido y ha perdido conexión con la red. Para evitar la perdida de direcciones IP libres los nodos tienen que comprobar periódicamente que su nodo padre y sus nodos hijos siguen en la red. Mediante un PING a nivel IP. Si los nodos hijos han abandonado la red el nodo recupera las direcciones que le asignó. Si su nodo padre ha abandonado la red se encargará de sus direcciones solo si es el nodo más prioritario que queda en la red. Si no lo es, comprueba que un nodo más prioritario esté en ella, para asegurarse de que hay alguien que va a asumir esas direcciones. Si se hace responsable de recoger alguna dirección liberada notificará mediante mensajes Node_down la caída del nodo, para que el resto de integrantes de la red tenga una tabla de la situación de la red actualizada. Si algún nodo no responde a este Node_down con un Node_down_rep, se reenvía Node_down.

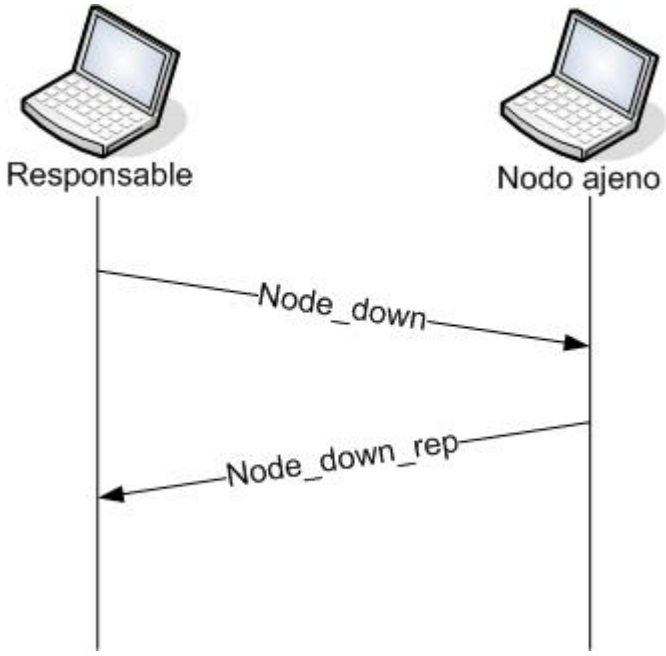


Figura 3.4 – Intercambio de mensajes en la salida brusca de un nodo.

3.2.4 – PROCESO DE SINCRONIZACIÓN

En redes Ad hoc la información de los nodos y de las direcciones libres no puede estar centralizada. Por tanto, todos los nodos tienen que tener una información actualizada de la estructura de la red. Para lo cual, cada vez que se produzca un cambio, la nueva situación ha de ser detectada y notificada al resto de nodos. De este modo todos podrán actualizar sus tablas y mantener una información veraz.

Disponemos de una tabla Allocated, que contiene información de los nodos de la red, los padres de cada uno, sus direcciones libres, y su TimeStamp. De modo que podamos reconstruir la topología de la red con cada tabla Allocated. Esta tabla ha de ser igual en todos los nodos. En caso de diferir algún valor debe de ser durante el periodo de tiempo más breve posible. Para ello disponemos de mensajes de sincronización tanto en la entrada como en la salida de nodos.

Después de la entrada de un nodo y de la asignación de direcciones se debe de notificar al resto con mensajes Node_up, Node_up_reply.

En la salida de nodo fácil se informará de la salida mediante el mensaje Graceful_departure. De este modo el nodo responsable sabrá que tiene que asumir esas direcciones liberadas, modificar sus tablas e indicar al resto como modificar sus tablas mediante el mensaje Node_down.

Para la salida brusca disponemos de un temporizador de sincronización que cada cierto tiempo expira para que se realice una comprobación de los nodos. Cada nodo se responsabiliza por defecto de los nodos a los que él ha entregado direcciones libres. De modo que sea lógico que él vuelva a recuperar esas direcciones y no otro. En el caso de salida brusca de un nodo líder, es decir, el nodo más antiguo de la red. Surge un caso a parte. Se tendrá que responsabilizar el nodo más prioritario de la red. Por tanto, a la hora de realizar la sincronización y la comprobación de nodos hijos, también se realizará la comprobación de nodos padres en este caso concreto. Si se detecta la caída de nodos se notificaría mediante mensajes como en la salida fácil.

3.2.5 – PERDIDA DE MENSAJES

El protocolo de autoconfiguración utiliza mensajes para los procesos de asociación, salida y sincronización, se utilizan mensajes UDP/IP al ser un protocolo no orientado a conexión es muy importante controlar la perdida de paquetes para evitar que existan nodos con IP duplicada.

Para tratar la pérdida de mensajes, evitando problemas en el funcionamiento del protocolo, se usan mensajes de confirmación y temporizadores apropiados para el cambio de mensajes, garantizando así que cada servicio sea realizado dentro de un tiempo finito. En la sección siguiente se describe cada tiempo utilizado en la comunicación entre los nodos.

Una ejemplificación de la pérdida de mensajes se muestra en el siguiente escenario: un nodo desea obtener una dirección IP enviando un mensaje addr_req. Siguiendo el proceso de configuración, los nodos servidores responden a la solicitud. Entonces, el nodo cliente escoge nodo servidor que por su parte suministra un free_ip_block y asocia una dirección IP. Por fin el nodo cliente envía el último mensaje confirmando la autoconfiguración. Suponiendo que el último mensaje (ip_assignment_ok), el resto no darían problemas, se pierde y no llegue hasta el nodo servidor confirmando la asociación de la dirección IP al nodo cliente. En este momento, el nodo servidor ya hizo el envío del free_ip_block. Sin embargo, como el mensaje de

confirmación no llegó, el nodo servidor no sabe si fue su mensaje ip_assigned que se perdió por el camino o fue el mensaje del nodo cliente que no llegó de vuelta. Al final, el nodo cliente tiene una dirección IP asociada, posee un free_ip_block, pero no consta como hijo del nodo servidor. La pérdida de ese mensaje descrito en el escenario anteriormente puede acarrear en un mal funcionamiento del protocolo, lo que es resuelto con la especificación del temporizador para cada mensaje intercambiado en el protocolo.

Otro caso de perdida de mensajes puede darse en la recepción del mensaje node_down_rep o en el caso homologo node_up_rep. Para los cuales se establece un sistema de detección de pérdidas de mensajes y sus correspondientes temporizadores. Si un mensaje de los anteriormente citados no llega a su destino se comprueba mediante PING que el nodo sigue aun en la red. Si sigue en la red quiere decir que uno de los mensajes se ha perdido y se reenvía.

3.2.6- PARTICIÓN Y FUSIÓN DE REDES

Partición

Cuando uno o varios nodos pierden la conexión con la red formarán su propia red. Según el protocolo definido, esta situación no es mayor problema. Ya que se actuará como si se tratase de una salida brusca de uno o varios nodos. En el caso de ser varios nodos se recuperará progresivamente las direcciones liberadas, a medida que salten los temporizadores de sincronización. Los nodos que han perdido la conexión formarán una red independiente. Con el mismo identificador de red (partition_id). Con lo cual el problema podría darse al volver a fusionar estas dos redes si alguna dirección IP común ya a sido entregada. Pero este inconveniente se soluciona comparando los TimeStamp existentes en la tabla Allocated, con aquellos recibidos de otros nodos. Ya que solo se considerarán nodos de la misma red si estos dos últimos datos coinciden. Los nodos de la nueva red al detectar este conflicto cambiarán su partition_id.

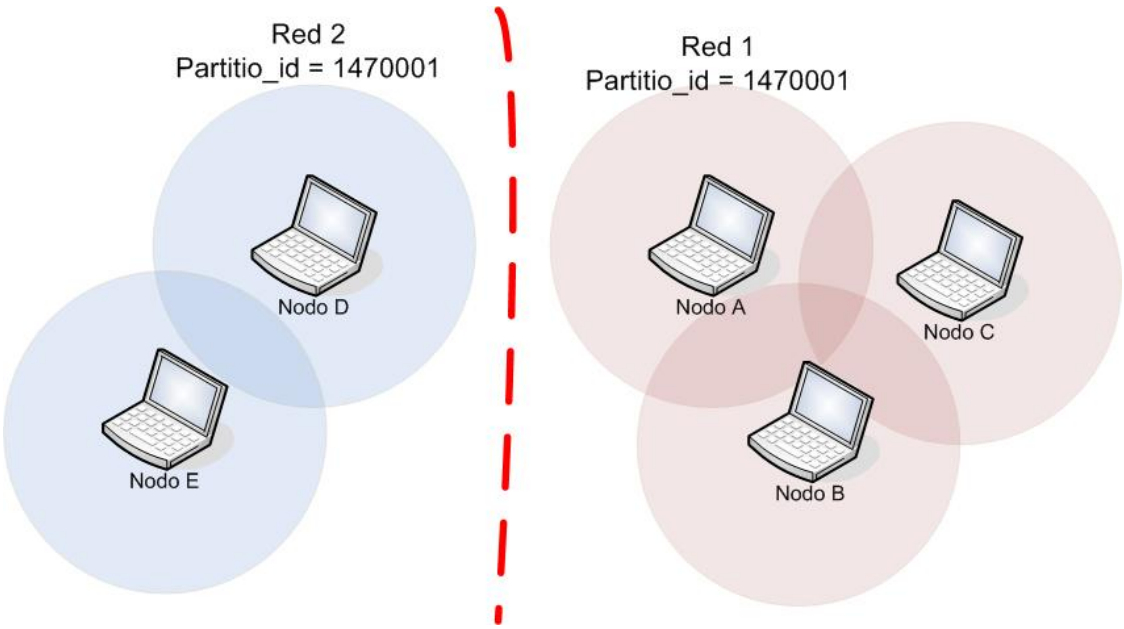


Figura 3.5 – Partición de una red en dos con el mismo partition_id

Fusión

Cuando dos redes con diferente partition_id entran en contacto se tendrá en cuenta el dato de la densidad de cada red enviado en cada mensaje. Si una red se encuentra con otra igual o más grande solicitará la entrada en la misma enviando un addr_req. Y realizará el proceso de autoconfiguración en caso de ser aceptado por la otra red. El resto de los nodos se unirán a esta red de mayor tamaño progresivamente.

La detección de una unión de dos o más redes Manet se da a través de un proceso muy simple. Aprovechando que los nodos intercambian mensajes Hello periódicamente, es posible detectar de forma clara y eficiente cuando dos o más redes se juntaron. Esos mensajes Hello contienen la dirección IP y el partition_id del nodo. Cuando un nodo recibe un mensaje Hello contiendo un partition_id diferente de la suya, él detecta que hubo una unión de dos redes ad-hoc.

Cuando se produce la unión de dos redes ad-hoc, la cantidad de nodos puede superar las 256 direcciones IP disponibles del direccionamiento. En ese caso, la solución a tomar es el cambio de la clase de direccionamiento de la red IP que en ese ejemplo, sería el cambio del direccionamiento utilizando la clase C que comporta a lo sumo 256 nodos para el direccionamiento utilizando la clase B que soporta redes con hasta 65536 nodos.

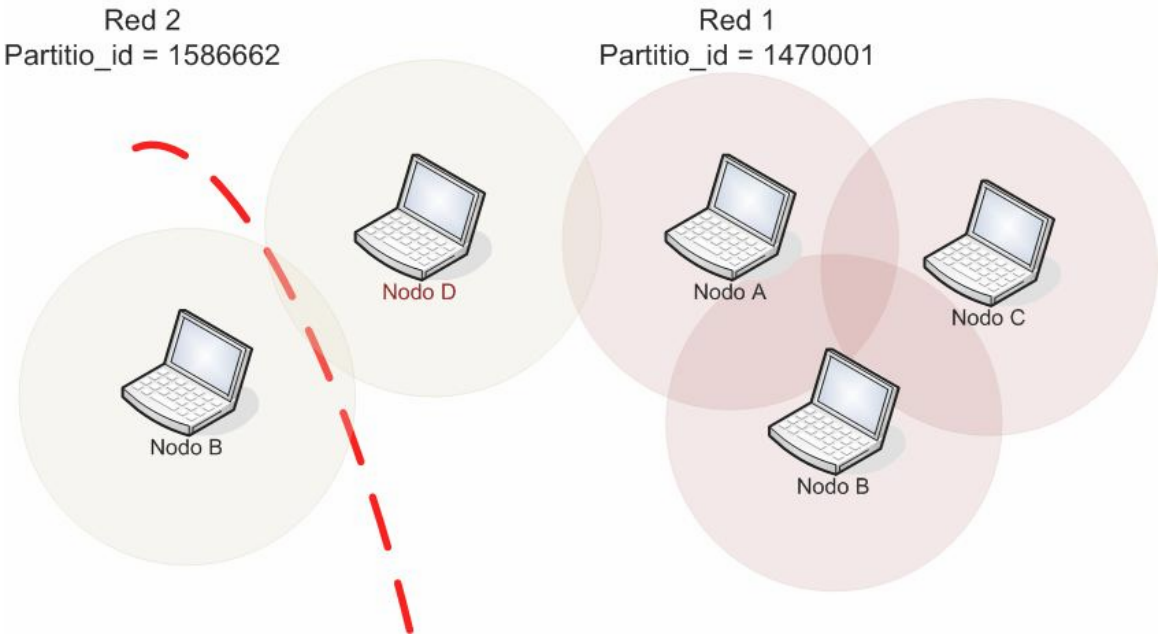


Figura 3.6 – Fusión de dos redes

3.3 - MENSAJES Y ESTRUCTURAS DE DATOS DEL PROTOCOLO

El protocolo de autoconfiguración es eficiente por poseer un número finito de mensajes para cada operación. Todas las operaciones descritas y especificadas en el protocolo como, la asociación de una dirección IP, la salida de nodos, la sincronización de la red y la unión y la separación de redes poseen un tiempo finito para el cambio de mensajes garantizando así un óptimo funcionamiento y una baja latencia.

En esta sección, se describen todas las estructuras de datos y tablas que se utilizan en el cambio de mensajes del protocolo. También, se tiene el formato de cada mensaje utilizado en la comunicación entre los nodos, además de la especificación del tiempo utilizado en cada operación.

3.3.1- ESTRUCTURA DE DATOS Y TABLAS UTILIZADAS

Las siguientes estructuras de datos se necesitan para mantener actualizadas las informaciones de la red. Cada tabla especificada abajo debe ser almacenada localmente en todos los nodos. Las tablas son actualizadas en intervalos de tiempo determinados en todos los nodos como esta descrito en el proceso de sincronización. Considerando un nodo, tenemos:

Free_IP_blocks: tabla que contiene las franjas de direcciones IP libres para atender los nuevos nodos, es decir, son las direcciones que están libres para que sean asociadas a los nuevos nodos que llegan a la red. Para esa tabla, se tiene una característica importante, que es que todas las direcciones de cada bloque son disjuntas de cualquier otro Free_IP_block de la red. En esta tabla se indicará en que bloque se encuentra la dirección IP del propio nodo. Para tenerlo en cuenta a la hora de distribuir direcciones.

Allocated_IP_blocks: tabla que contiene todos los datos necesarios de la topología de la red. Cada entrada de la tabla corresponde a la información de cada nodo que se encuentre en la red. Con los siguientes datos:

- Su dirección de red asignada para su interfaz.
- Sus Free_IP_blocks. Indicando en cual de esos bloques se encuentra su propia dirección de red.
- Su Timestamp, que identifica al nodo, y le otorga una prioridad, útil a la hora de recuperar direcciones de red.
- La dirección IP del padre que hizo de servidor en su entrada en la red y le otorgo su bloque de direcciones libres.

Para cada red ad-hoc, tenemos un identificador único (partition_id) que identifica unívocamente una red. El primer nodo de la red es el responsable de la generación del partition_id utilizando su dirección de hardware y su Timestamp. También se llevara cuenta de la densidad de la red.

3.3.2 – FORMATO DE LOS MENSAJES

En este apartado veremos como están formados los mensajes de control que intercambian los nodos para realizar la autoconfiguración, controlar la salida de nodos, tanto fácil como brusca, y la sincronización de nodos.

Los mensajes del protocolo de configuración que son utilizados en la comunicación entre los nodos están descritos en la Tabla 3.1:

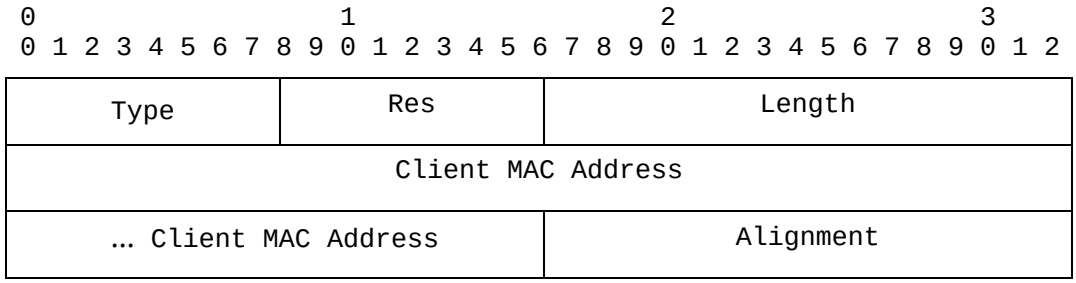
Tabla 3.1 – Mensajes del protocolo de autoconfiguración.

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

Los siete primeros mensajes son los mensajes necesarios para la asociación de una IP a un nuevo nodo de acuerdo con funcionamiento del protocolo descrito en la sección 3.2.

3.3.2.1 - ADDR_REQ

Mensaje que el nodo cliente envía para la obtención de una dirección IP. La dirección origen es la dirección MAC de su interfaz de red y la dirección destino es la dirección de broadcast, ya que deberá solicitar a todos los nodos.



El formato del mensaje addr_req está ilustrado arriba y contiene 12 bytes con los siguientes campos:

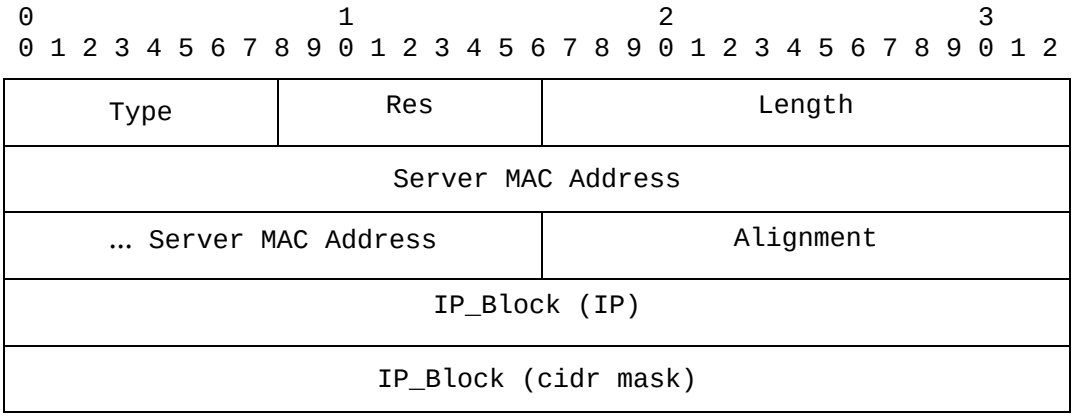
Tabla 3.2 – Campos del mensaje *addr_req*.

Campo	Tamaño (bits)	Valor
Type	8	Addr_req
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Client MAC Address	48	Dirección de hardware del nodo cliente
Alignment	16	Enviado como 0 e ignorado en la recepción.

Tiempo: el nodo cliente inicia el temporizador después de enviar el mensaje de requisito y espera un mensaje de respuesta de algún nodo servidor de la red. Cuando el tiempo expira y no se recibe ningún mensaje de los nodos vecinos, envía un nuevo mensaje de petición T veces, donde T es la cantidad máxima de tentativas de obtener una dirección IP. Una vez fallada la tentativa de conectarse el nodo iniciará su propia red.

3.3.2.2 - ADDR_REP

Mensaje que el nodo servidor envía para el nodo cliente cuando recibe el mensaje addr_req. El nodo cliente puede recibir este mensaje de varios nodos servidores, cosechando las informaciones de cada respuesta, sin embargo, escogerá solo el nodo servidor con el mayor free_ip_block.



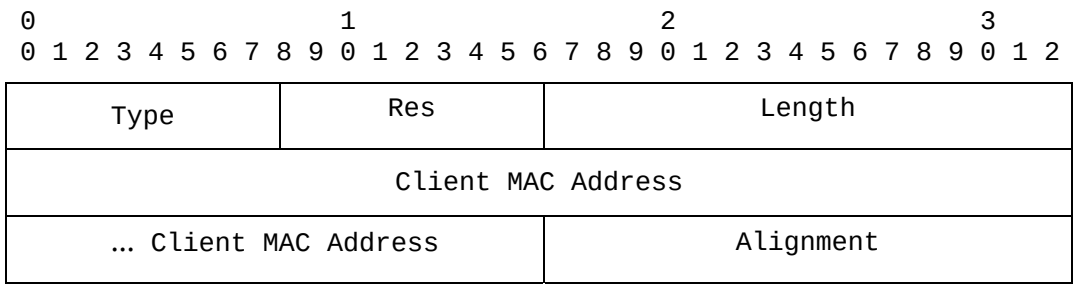
El formato del mensaje addr_rep está ilustrado arriba y contiene 20 bytes con los siguientes campos:

Tabla 3.3 – Campos del mensaje *addr_rep*.

Campo	Tamaño (bits)	Valor
Type	8	Addr_rep
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Server Mac Address	48	Dirección hardware del nodo servidor
Alignment	16	Enviado como 0 e ignorado en la recepción.
IP_Block	64	IP address + CIDR mask

3.3.2.3 - SERVER_POLL

Mensaje del nodo cliente enviada únicamente para el nodo servidor diciendo que lo escogió entre todos los mensajes addr_rep recibidos por contener el mayor free_ip_block. Ese mensaje contiene sólo la dirección hardware del nodo cliente. Después de la recepción de ese mensaje por el nodo servidor, comienza el proceso de autoconfiguración del nodo cliente.



El formato del mensaje server_poll está ilustrado arriba y contiene 12 bytes con los siguientes campos:

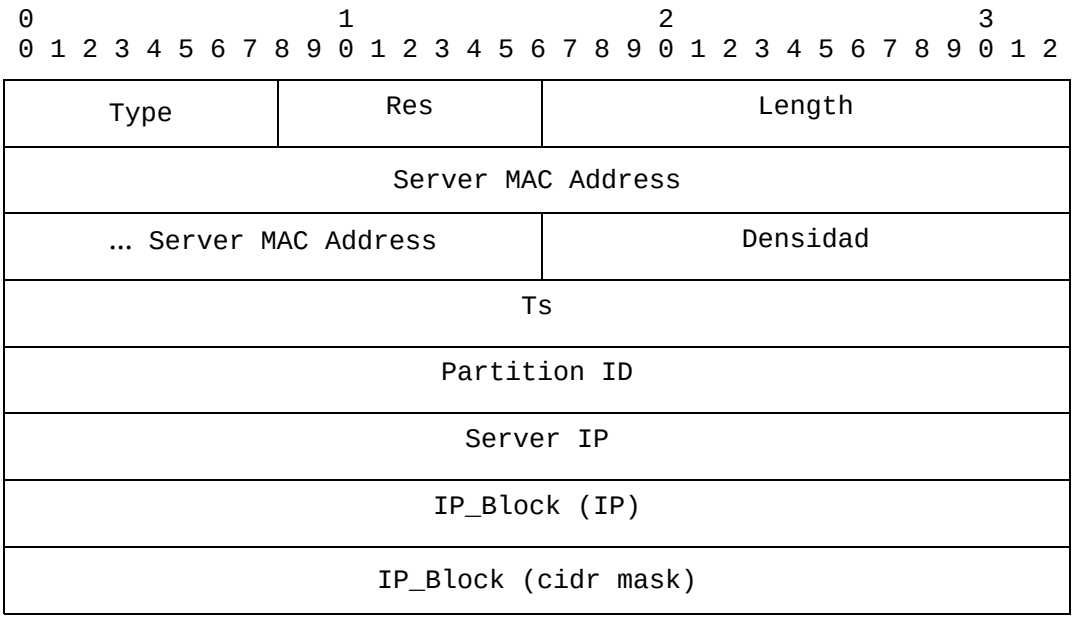
Tabla 3.4 – Campos del mensaje *server_poll*.

Campo	Tamaño (bits)	Valor
Type	8	Server_poll
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Client MAC Address	48	Dirección hardware del nodo cliente
Alignment	16	Enviado como 0 e ignorado en la recepción.

Timer: el nodo cliente inicia ese tiempo después del envío del mensaje server_poll y espera un mensaje ip_assigned del nodo servidor. Cuando el tiempo expira, el nodo cliente asume que el nodo servidor dejó la red antes de completar el proceso de autoconfiguración enviando un nuevo mensaje server_poll por T veces, donde T es la cantidad máxima de tentativas de comunicar con el servidor escogido para suministrar el free_ip_block.

3.3.2.4 - IP_ASSIGNED

Mensaje enviado por el nodo servidor al nodo cliente asociando la dirección IP al nodo cliente. Conjuntamente con ese mensaje sigue el identificador único de la partición, el Partition_id. Por lo tanto, este mensaje contiene las siguientes informaciones: Partition_id de la red, densidad de la red, el timestamp del nodo servidor, la IP del nodo servidor y la free_ip_block. La primera dirección del free_ip_block es asociado a la interfaz del nodo cliente y el restante de las direcciones se queda almacenado en la estructura free_ip_block del nodo cliente y sirve para que él pueda atender el requisito de otros nodos, convirtiéndose en un nodo servidor.



El formato del mensaje ip_assigned está ilustrado arriba y contiene 32 bytes con los siguientes campos:

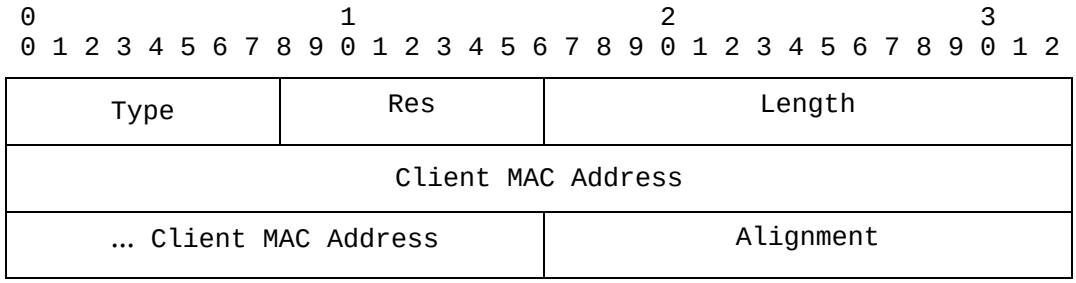
Tabla 3.5 – Campos del mensaje ip_assigned.

Campo	Tamaño (bits)	Valor
Type	8	IP_assigned
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Server Mac Address	48	Dirección hardware del nodo servidor
Densidad	16	Densidad de la red
Ts	32	Time Stamp
Partition_id	32	Identificador único de la partición
Server IP	32	IP del servidor
IP_Block	64	IP address + CIDR mask

Timer: el nodo servidor inicia ese tiempo después de enviar un mensaje ip_assigned y espera el mensaje ip_assignment_ok confirmando que el proceso de autoconfiguración se ha concluido con éxito.

3.3.2.5 - IP_ASSIGMENT_OK

Mensaje del nodo cliente enviada únicamente para el nodo servidor afirmando que la dirección IP fue asociado con éxito.



El formato del mensaje ip_assignment_ok está ilustrado arriba y contiene 12 bytes con los siguientes campos:

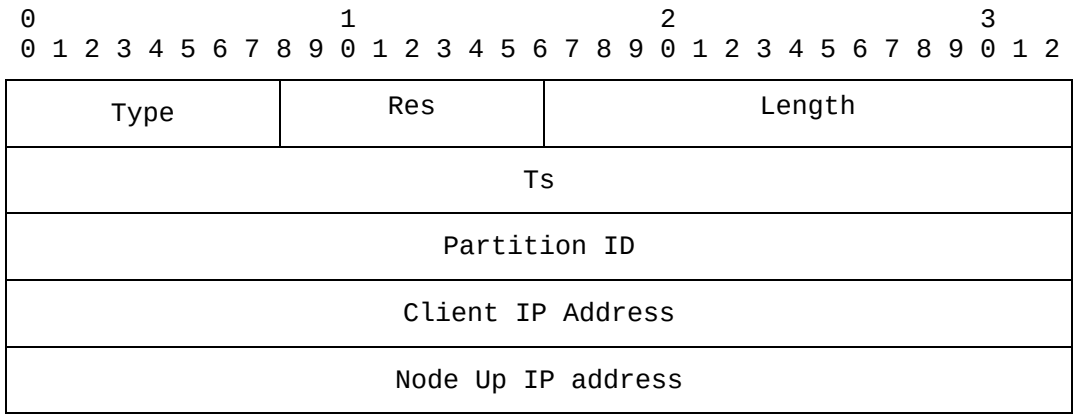
Tabla 3.6 – Campos del mensaje ip_assignment_ok.

Campo	tamaño (bits)	Valor
Type	8	IP_assignment_ok
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Client Mac Address	48	Dirección hardware del nodo cliente
Alignment	16	Enviado como 0 e ignorado en la recepción.

Si este mensaje no llega el nodo servidor comprobará mediante mensajes PING la asociación correcta de direcciones del nodo cliente. Si contesta asume que el IP_assigned_ok se ha perdido. Si no responde al PING rectificará los cambios que haya podido realizar en sus tablas.

3.3.2.6 - NODE_UP

Mensaje que el nodo entrante en la red manda al resto para que actualicen sus tablas. Realiza un primer envió mediante broadcast. Y los sucesivos envíos si fuesen necesarios los hace directamente a los nodos que no respondan a la primera petición.



Mensaje: 20 bytes

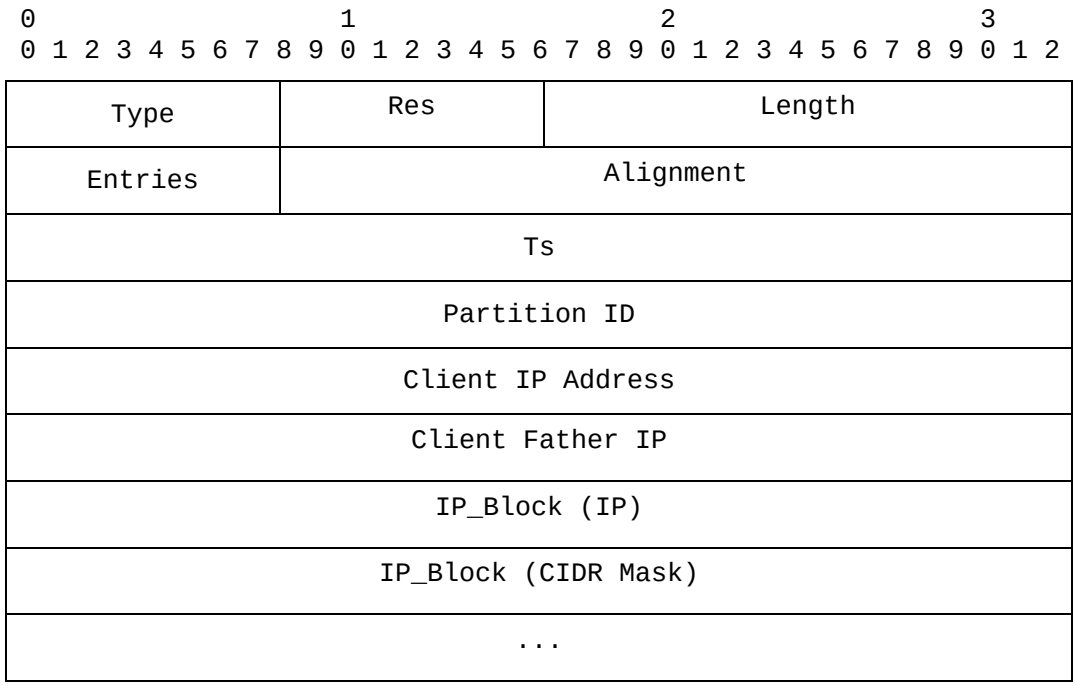
Tabla 3.7 – Campos del mensaje node_up.

Campo	Tamaño (bits)	Valor
Type	8	Node_up
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Ts	32	Timestamp
Partition ID	32	Partition ID
Client IP Address	32	Dirección IP del nodo que le ha dado las direcciones al nodo entrante
Node up IP Address	32	Dirección de IP del nodo entrante

El nodo receptor del mensaje realizará los cambios pertinentes en sus tablas para contener una información actualizada de la situación de la red. Dividir las direcciones de “Client IP” para “Node up”, asignarle un timestamp y un padre. Y ajustar la densidad de la red.

3.3.2.7 - NODE_UP_REP

Mensaje de respuesta que mandan los nodos que han recibido un Node_up una vez que ha actualizado las tablas. Sirve para notificar el cambio y contiene la información del nodo que responde, de este modo el nodo entrante podrá construir su tabla allocated con esta información.



Mensaje: 32 bytes mínimo = (24 + entradas*8)

Tabla 3.8 – Campos del mensaje node_up_rep.

Campo	Tamaño (bits)	Valor
Type	8	Node_up_rep
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Alingment	24	Enviado como 0 e ignorado en la recepción
Entries	8	Numero de entradas de la tabla free_ip
Ts	32	Timestamp
Partition ID	32	Partition ID
Client IP Address	32	Dirección IP del nodo cliente
Client Father IP	32	Dirección IP del padre
IP_Block	64 * Entries	IP + CIDR mask (El primer IP_block es el que contiene la IP del nodo)

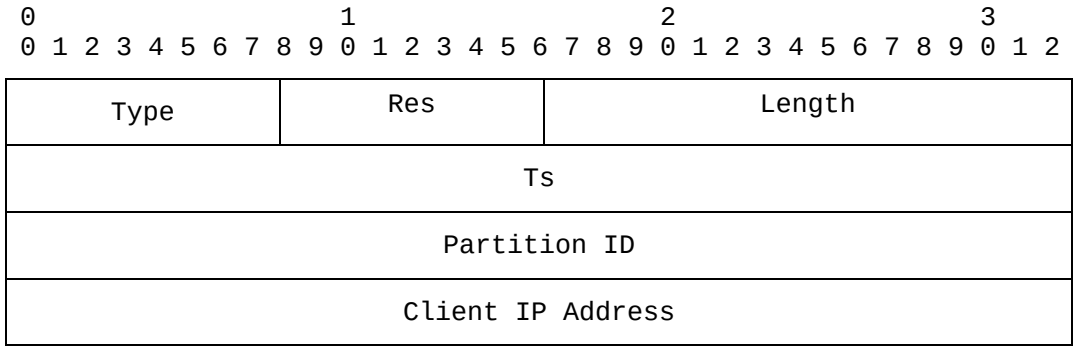
El tamaño de este mensaje es variable. Y dependerá del numero de entradas que tenga cada nodo en su tabla free_ip_blocks. En función de esto mandará un mensaje en el modo descrito anteriormente. El nodo entrante lleva control de los nodos que le responden para asegurarse que todos han recibido su Node_up, así en caso de salto de temporizador reenviará el Node_up si eso fuese necesario. Con la recepción de este mensaje se concluye la autoconfiguración.

Se hace notar que los mensajes intercambiados entre los nodos poseen una cabecera compuesta de un campo con el tipo del paquete, un campo que especifica el tamaño del paquete y por fin un campo reservado para uso futuro. Los mensajes poseen tamaño y tiempo limitados, garantizando la eficiencia del protocolo así como su robustez ante la pérdida de mensajes.

Ahora veremos los tres mensajes de control utilizados en el proceso de salida de nodos.

3.3.2.8 - GRACEFUL_DEPARTURE

Mensaje que el nodo cliente envía para el nodo responsable de sus direcciones indicando que él está definitivamente dejando la red. En principio será el nodo padre. Si no se encontrara disponible en ese momento lo seria el nodo disponible más prioritario (con menor Ts). El nodo cliente solo lo envía si previamente se ha cerciorado que el nodo al que desea enviarlo se encuentra realmente en la red mediante PING.



Mensaje: 16 bytes

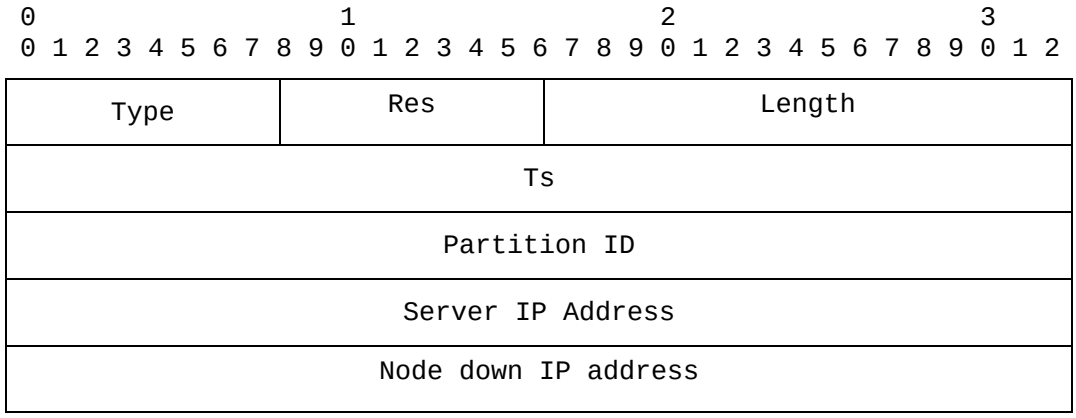
Tabla 3.9 – Campos del mensaje graceful_departure.

Campo	Tamaño (bits)	Valor
Type	8	Graceful_departure
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Ts	32	Timestamp
Partition ID	32	Partition ID
Client IP Address	32	Dirección IP del nodo que deja la red

A partir de ese momento se asume que el nodo cliente dejó la red con éxito y todas las direcciones IP forman parte del free_ip_block del nodo servidor con el cual él estaba comunicándose. Al igual que los buddy del nodo que abandona la red pasan a ser nodos buddy del receptor del mensaje. Es decir, que todos los que tenían como padre al nodo caído pasan a tener como padre al nodo responsable. De este modo, no es necesario el proceso de recuperación de direcciones IP, que ocurre cuando un nodo deja la red abruptamente.

3.3.2.9 - NODE_DOWN

Mensaje que el nodo que asume las direcciones de un nodo caído manda al resto. Para que actualicen sus tablas. Realiza un primer envío mediante broadcast. El resto de envío, si los tuviese que realizar, sería individualmente a cada nodo que no haya respondido correctamente a esta notificación.



Mensaje: 20 bytes

Tabla 3.10 – Campos del mensaje node_down.

Campo	Tamaño (bits)	Valor
Type	8	Node_down
Length	16	Longitud total de mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Ts	32	Timestamp
Partition ID	32	Partition ID
Server IP Address	32	Direccion IP del nodo que asume las direcciones
Node down IP Address	32	Direccion de IP del nodo caído

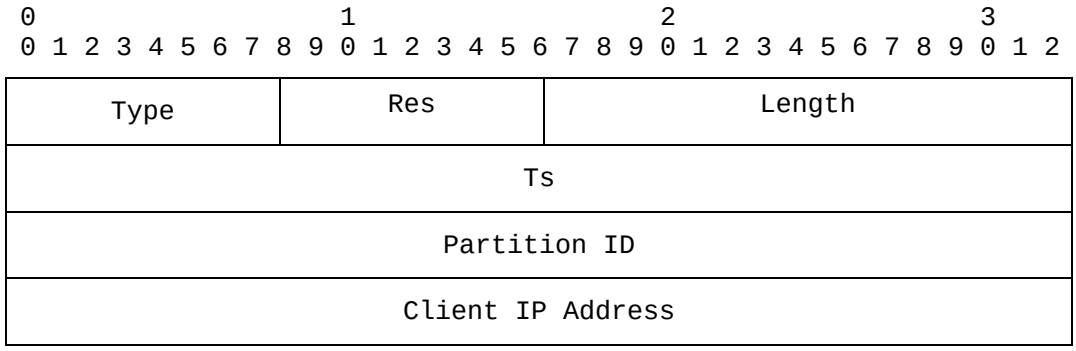
El nodo que abandona la red lo llamaremos A. El que asume sus direcciones será B.
Para actualizar las tablas allocated de los nodos receptores del mensaje se realizaran los siguientes cambios en la tabla Allocated:

- Las Free_IP de A pasan a las Free_IP de B.
- Los que tenían como padre a A, pasan a tener como padre a B.
- Si el padre de A no es B quiere decir que el nodo caído era un nodo líder(nodo más prioritario). Así que B pone como padre NULL.

El nodo que envía el mensaje activará el temporizador node_down_timer. De modo que si no recibe las confirmaciones esperadas antes del salto del temporizador reenviará el mensaje a todos los nodos que no contesten y todavía se encuentren en la red.

3.3.2.10 - NODE_DOWN_REP

Mensaje de respuesta que mandan los nodos que han recibido un Node_down una vez que ha actualizado las tablas.



Mensaje: 16 bytes

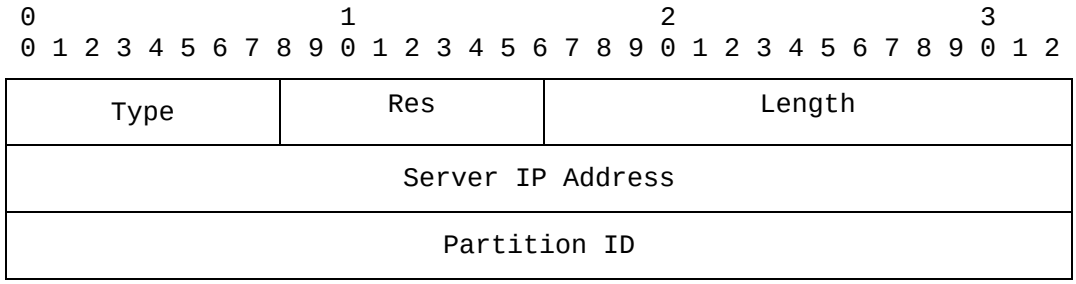
Tabla 3.11 – Campos del mensaje node_down_rep.

Campo	Tamaño (bits)	Valor
Type	8	Node_down-rep
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Ts	32	Timestamp
Partition ID	32	Partition ID
Client IP Address	32	Dirección IP del nodo que envía el mensaje

Después de haber recibido un Node_down los nodos lo envían cuando han actualizado su tabla allocated para notificar al nodo que envió ese mensaje que los cambios han sido realizados.

3.3.2.11 - HELLO

Mensaje utilizado para difusiones locales. Intercambiado entre todos los nodos contiendo sólo la identificación de la partición para que pueda ser detectada la unión de particiones, lo que ocurre cuando un nodo recibe un mensaje Hello con una identificación de partición diferente de la suya.



El formato del mensaje Hello está ilustrado arriba y contiene los siguientes campos:

Tabla 3.12 – campos del mensaje hello.

Campo	Tamaño (bits)	Valor
Type	8	IP_assigned
Length	16	Longitud total del mensaje
Res	8	Reservado - enviado como 0 e ignorado en la recepción
Server IP	32	Dirección IP del nodo
Partition ID	32	Identificador único de partición

3.3.3 TEMPORIZADORES UTILIZADOS

La utilización de temporizadores para el cambio de mensajes en el protocolo asegura que su funcionamiento sea correcto aún con eventuales pérdidas de mensajes o fallo de los nodos. La especificación de cada temporizador utilizado se configura en la implementación, donde se hacen pruebas y simulaciones con temporizadores varios para evaluarse el comportamiento del protocolo.

La especificación de los temporizadores se hace considerando varios factores externos como: la densidad de la red, la cantidad de nodos y la tasa de comunicación entre ellos. Así, cuanto mayor la cantidad de nodos, mayor debe ser el tiempo para que un mensaje llegue a su destino. Abajo, se describe detalladamente cada temporizador utilizado en la asociación de una dirección IP a un nuevo nodo y la salida de nodos:

- **Reply_timer:** el nodo cliente inicia ese temporizador después de enviar un mensaje de petición de dirección IP addr_req y espera el mensaje addr_rep del nodo servidor. Si el temporizador expira y no ha recibido algún mensaje, el nodo cliente envía un nuevo mensaje de petición e inicia el temporizador. El nodo cliente repite el envío del mensaje addr_req T veces donde T es la cantidad máxima de tentativas que el nodo cliente hace para obtener una dirección IP.

- **IP_assigned_timer**: el nodo cliente inicia ese temporizador después de enviar un mensaje Server_poll y espera el mensaje IP_assigned que contiene el free_ip_block. Si el temporizador expira y no ha recibido ningún mensaje, el nodo cliente asume que el nodo servidor dejó la red antes de completar el proceso de autoconfiguración. El nodo cliente repite el envío del mensaje Server_poll por T veces donde T es la cantidad máxima de tentativas de escoger el servidor para suministrar el free_ip_block.

- **Confirm_timer**: el nodo servidor inicia ese temporizador después de enviar un mensaje ip_asigned a un nodo cliente y espera el mensaje ip_asigment_ok del nodo cliente confirmando que el proceso de autoconfiguración fue concluido con éxito. Si el temporizador expira y no ha recibido el mensaje de confirmación, el nodo servidor asume que el nodo cliente dejó la red antes de completar el proceso de autoconfiguración. En caso contrario, el nodo servidor actualiza su tabla de direcciones.

- **Node_up_timer**: Un nodo entrante en la red notificará de su llegada mediante un envío generalizado de mensajes node_up. En este momento se activa este temporizador. Se aguarda la llegada de confirmaciones, en forma de node_up_req, como respuesta de los nodos. Si expira el temporizador y no han llegado todas las confirmaciones se repite el envío de mensajes node_up a todos los nodos que no respondan y se haya previamente comprobado que aun sigan en la red.

- **Node_down_timer**: El nodo encargado de recoger las direcciones liberadas por un nodo saliente activará este temporizador una vez enviado un mensaje node_down a todos los nodos involucrados. Si expira el temporizador y todavía no se han recibido todas las confirmaciones en forma de node_down_rep. Se

reenviará un mensaje node_down solo a los nodos que no hayan respondido y que continúen en la red, previa comprobación de esto último mediante mensajes PING.

- **Synchronize_timer**: Temporizador que nos permite realizar la sincronización. Cuando tenemos un nodo configurado en la red activamos este temporizador. De modo que cada vez que salte realizaremos un proceso de comprobación de cambios en la red y se tomarán las acciones pertinentes al respecto. Una vez concluido volvemos a reiniciar el valor del temporizador. Cuando expira el temporizador tenemos que comprobar si todos los nodos de los que somos responsables están aun en la red. Es decir, todos nuestros nodos hijos y nuestro nodo padre, si este último es nodo líder de la red. Si alguno ha caído nos responsabilizaremos de sus direcciones libres y se realizaran los cambios y notificaciones pertinentes como se ve en el apartado de salida brusca 3.2.3.1.

3.4 – EJEMPLO ILUSTRATIVO DEL PROTOCOLO

En este apartado mediante un sencillo ejemplo explicaremos las principales situaciones a las que ha tenido que enfrentarse el protocolo diseñado. Partiremos de una situación básica en la existe una red con cuatro nodos. Veremos la entrada de un nuevo nodo a esa red y posteriormente los diferentes tipos de salidas que pueden experimentar los nodos, la salida controlada de un nodo y el proceso de sincronización para la recuperación de direcciones IP de los nodos que fallaron o partieron abruptamente.

Para ejemplificarlo usaremos una red de clase C con 256 direcciones.

3.4.1 – INICIALIZACIÓN DE LA RED CON 3 NODOS

Partiremos de una situación en la que ya se encuentran 3 nodos activos en la red. Los cuales han formado la red de la siguiente manera: En el instante de tiempo 50 entró el nodo A nombrándose como nodo líder de la red y poseyendo 256 direcciones. Su dirección IP sería 192.168.0.1, nótese que las direcciones .0 y .255 son direcciones reservadas.

Posteriormente B pedirá direcciones libres a A en el instante 100. Por tanto A dará la mitad de sus direcciones a B quedando cada uno con 128 direcciones libres. B se configurará con la dirección .128

En el instante 200 entra C y pide direcciones. El nodo A divide sus direcciones de nuevo y le entrega 64 direcciones. Quedando configurada la red en ese instante del siguiente modo.

Allocated

	IP	*free_IP*	*father*	*Ts*
A	*.1*	*.2 - .63*	*-*	*50*
B	*.128*	*.129 - .254*	*A*	*100*
C	*.64*	*.65 - .127*	*A*	*200*

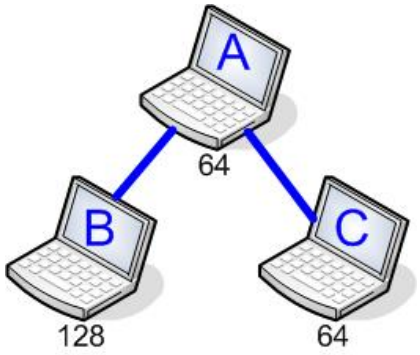


Figura 3.7 – Estado inicial de la red con 3 nodos

3.4.2 – ASIGNACIÓN DE UNA DIRECCIÓN IP A UN NODO ENTRANTE

Partiendo de la situación de la figura 3.5 suponemos la entrada de un nuevo nodo D en el instante de tiempo 500. El cual solicita formar parte de la red.

El nodo D enviará un mensaje addr_request en modo broadcast para solicitar direcciones libres. Suponemos que a este mensaje responden todos los nodos de la red con sendos mensajes addr_reply. D examinará estas respuestas y seleccionará como nodo más apropiado para su configuración aquel que posee mayor número de direcciones libres. En este caso será elegido B ya que posee 128 respecto a las 64 de A y C. En este momento D envía un mensaje Server_poll para hacer saber a B que ha sido el

nodo elegido para la configuración. B dividirá sus 128 direcciones y entregará 64 a D mediante el mensaje IP_assignment. El siguiente paso en la configuración de D será que este nodo asuma esas direcciones y configure su dirección IP. En este caso su IP será .192 que es la primera dirección del grupo de sus direcciones libres asignadas. Ahora D manda un IP_assignment_OK para hacer saber al nodo servidor B que la configuración del nuevo nodo ha sido realizada. Y a continuación D mandará un mensaje Node_up en modo broadcast para notificar al resto de nodos su entrada en la red.

Solo falta la confirmación por parte de A y C en forma de mensajes Node_up_req. Que cuando son recibidos sendos mensajes por D se da por concluida la autoconfiguración del nuevo nodo.

Allocated

	IP	*free_IP*	*father*	*Ts*
A	*.1*	*.2 - .63*	*-*	*50*
B	*.128*	***.129 - .191***	*A*	*100*
C	*.64*	*.65 - .127*	*A*	*200*
D	***.192***	***.193 - .254***	***B***	***500***

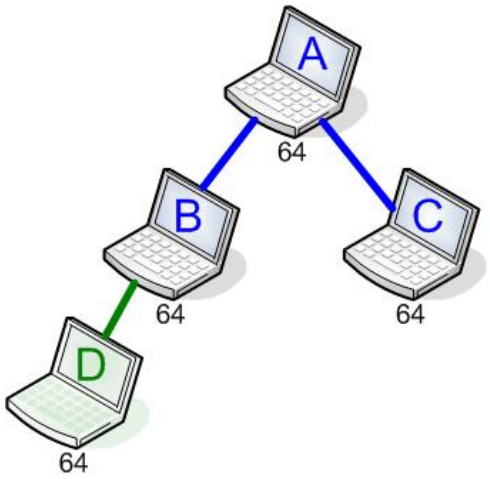


Figura 3.8 – Estado de la red tras la entrada de un cuarto nodo.

3.4.3 – SALIDA FACIL DE UN NODO

Partimos de la situación de la figura 3.6 con cuatro nodos configurados. Vamos a ver como se produciría la salida de un nodo que notifica su salida de la red. Para lo cual seleccionaremos al nodo B para realizarla. Primeramente el nodo B tendrá que buscar al nodo que sea responsable de sus direcciones cuando las libere. En principio debería ser su padre si todavía se encuentra en la red. Por lo tanto comprueba si su padre está o no en la red mediante un PING. Se dan dos posibles situaciones:

- Ante una respuesta positiva considerará como nodo responsable al nodo A.
- Si no responde tiene que buscar por orden de prioridad al nodo que este aun en la red. Cuando lo encuentre lo hace nodo responsable. En este caso si A no responde haría un PING a C que tiene timestamp a 200 y si no respondiese lo haría a D con timestamp 500.

Suponemos que A tal y como indica la tabla allocated se encuentra en la red. Por tanto le entrega las direcciones de su free_ip_blocks en un mensaje graceful_departure a este nodo padre A, y a continuación el nodo B se desconecta.

Ahora A realiza una serie de cambios en sus tablas:

- Añade la (IP + free_IP) de B a su tabla Free_IP_Blocks.
- Busca quien tenía como padre B. En Allocated, todos los que tenían como padre a B pasan a tener como padre A.

- Cancela la entrada de B en la Allocated.

Una vez realizados, A envía un Node_down a todos los nodos restantes, C y D para que realicen los mismos cambios en sus tablas allocated. A esperará también de C y D un mensaje Node_down_rep. Una vez hecho esto la situación es la siguiente:

Allocated

	IP	*free_IP*	*father*	*Ts*
A	*.1*	*.2 - .63* **.128 - .191**	-	*0*
~~*B*~~	~~*.128*~~	~~*.129 - .191*~~	~~*A*~~	~~*100*~~
C	*.64*	*.65 - .127*	*A*	*200*
D	*.192*	*.193 - .254*	***A***	*500*

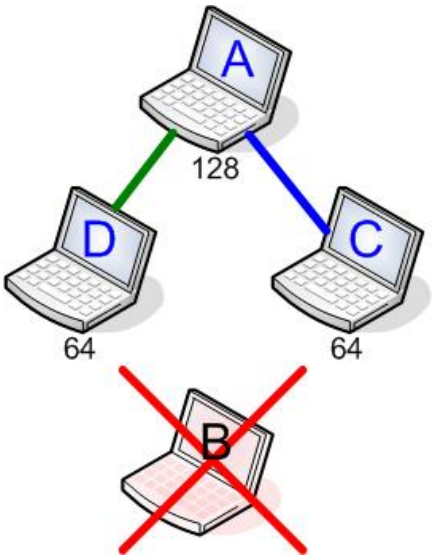


Figura 3.9 – Estado de la red tras la salida del nodo B.

3.4.4 – FALLO DE UN NODO Y RECUPERACIÓN DE DIRECCIONES IP

Volvemos a partir de la situación de la figura 3.6. Suponemos ahora que el nodo B abandonó la red sin notificar. Por lo tanto hasta que salte el temporizador de sincronización las tablas no contendrán información veraz. Una vez que esto ocurra D comprobará que su padre no se encuentra en la red mediante PING. Tras esto deberá comprobar que no exista un nodo más prioritario que el que sea el responsable. Así que comprueba que A esta en la red. A responde y es un nodo más prioritario que él, por tanto no debe hacer nada.

Cuando salte el temporizador de sincronización de A comprobará que sus hijos siguen en la red. C responderá al PING de comprobación, pero B no lo hará ya que abandonó la red. En este momento A deberá realizar los cambios oportunos en sus tablas que serán los mismos que en la salida fácil.

Como último paso A notifica a C y D de la caída de B. Lo hará mediante node_down de modo análogo a la salida fácil. La situación final como es lógico será la misma a la de la figura 3.7.

4 – IMPLEMENTACIÓN DEL PROTOCOLO

Una vez especificado y detallado el funcionamiento del protocolo, se describe en este capítulo los pasos necesarios para su implementación incluyendo las partes más importantes del código fuente, las herramientas utilizadas para proyectarse el protocolo, el ambiente físico, lógico y la arquitectura utilizada. El objetivo final de la implementación es evaluar la funcionalidad del protocolo en una red ad-hoc.

De este modo, se implementa el protocolo de autoconfiguración de direcciones IP en redes móviles ad-hoc, a fin de automatizar y optimizar la tarea de la distribución de las direcciones IP a los nodos de la red. El sistema necesita tener características de adaptabilidad y ajustarse a las situaciones especiales, debido a la movilidad constante e imprevisible de los nodos, que pueden entrar y salir de la red necesitando a cada instante de configuración de la dirección IP.

Todos los mensajes intercambiados en el protocolo utilizan los protocolos UDP/IP para la comunicación.

4.1 – CONFIGURACIÓN DEL ENTORNO

Se detalla aquí la configuración física y lógica del modelo, así como los entornos de hardware y software utilizados para el desarrollo del protocolo de autoconfiguración de direcciones IP en redes móviles ad-hoc.

4.1.1 – CONFIGURACION FÍSICA

Compaq nx7010 Intel Centrino 512 RAM
Compaq nx7010 Intel Centrino 512 RAM
Acer Aspire 3614LMi Intel Celeron 512 RAM
AcerPower 4400 Intel Pentium III 600 MHz 256 RAM

Todos los equipos portátiles tienen redes inalámbricas de comunicación Wi-Fi b/g integradas. El resto de equipos se han utilizado con tarjetas inalambricas Wi-Fi USB.

El sistema operativo en todos los casos será la distribución de Linux Ubuntu 2.12.1.

El entorno está configurado para operar en modo Ad hoc, eso implica que la comunicación entre las máquinas es hecha de forma directa, no necesitando de ningún punto común para centralizar el tráfico y las informaciones.

4.1.2 – CONFIGURACIÓN LÓGICA

El software utilizado para realizar la implementación del protocolo lo hemos llamado Autoconf. Es necesario tenerlo instalado en todas las máquinas participantes.

Módulo Autoconf: en ese módulo están todas las funcionalidades del servicio de autoconfiguración de los nodos. La solicitud y la distribución de las configuraciones de red las hace el mismo software. En otras palabras, el software actúa tanto de cliente como de servidor.

4.2 - HERRAMIENTAS UTILIZADAS

La implementación del entorno en cuestión fue realizada con herramientas de código abierto, o de uso libre, que son distribuidas gratuitamente en Internet, además del uso del sistema operativo Linux, tanto para los nodos que actúan como clientes cuanto para los nodos que actúan como servidores dentro de la red.

El uso de esas herramientas permitió la modificación y adaptación del código necesario para crear el protocolo de autoconfiguración y sincronización. Los siguientes puntos hablan de las herramientas utilizadas durante la implementación.

4.2.1 – LENGUAJE DE PROGRAMACIÓN C

El protocolo fue desarrollado utilizando el lenguaje de programación C con la ayuda de herramientas propietarias del sistema operativo Linux, que fueron de suma importancia en la implementación del protocolo en términos de la comunicación entre los nodos de la red. Se utilizó la herramienta “Wireless Tools” [42].

El leguaje C fue escogido porque presenta características propicias para el desarrollo de software en red. Dentro de las características que hacen de C un lenguaje de éxito para la programación de una red, destacan:

Simplicidad: C utiliza características sintácticas y semánticas que son utilizadas por millones de programadores en todo el mundo, desde programadores noveles hasta expertos de sistemas operativos.

Independencia de plataforma: el lenguaje C posee un mecanismo que facilita tanto la ejecución de los programas en máquinas diferentes, como en sistemas operativos diferentes, sin a necesidad de recompilación. Esa característica es fundamental en redes *ad hoc*, en las que es común el uso de máquinas de diferentes plataformas formando un ambiente heterogéneo.

Programación de red: el lenguaje C tiene ventajas cuando se trata de comunicación entre programas remotamente distribuidos en la red. Esa comunicación acontece por medio del uso de *sockets* que representan un punto de conexión lógica para una red TCP/IP. Cuando dos computadores quieren comunicarse, se puede crear una conexión vía *socket* especificando la dirección IP y la puerta lógica de cada máquina. En el caso de programación de *sockets,* un computador actúa como servidor, esperando conexiones, y otro, como cliente que irá a conectarse con el servidor para intercambiar información.

4.2.2 - LINUX

La parte principal del sistema operativo Linux es el Kernel que va evolucionando constantemente a través de iniciativas de OpenSource respaldadas por todo el mundo. En este trabajo fue utilizada la última versión de ese momento 2.12.1, concretamente en Linux Ubuntu, una distribución de Linux basada en Debian, por poseer la última versión de los compiladores, la mayoría de los archivos de configuración de las placas de red, las bibliotecas de programación para redes y la herramienta Wireless Tools.

El kernel del Linux no fue modificado en ningún nodo, facilitando así la implantación del protocolo en cualquier nodo que tenga instalado el sistema operativo.

4.2.3 - DHCP

El código fuente del protocolo DHCP fue bastante útil durante toda la fase de implementación. Se trata del protocolo padre para la distribución de las configuraciones de red para nodos en redes cableadas, su código fuente es muy extenso y de difícil comprensión.

La implementación hecha en lenguaje C de DHCP es mantenida por programadores del mundo entero posee dos archivos principales, uno que es el archivo cliente y otro que actúa como archivo servidor y que fija los requisitos dentro de la red. El código fuente posee también una gran cantidad de bibliotecas de archivos adicionales para el perfecto funcionamiento del protocolo. Entretanto, en el protocolo de autoconfiguración propuesto en este trabajo, la cantidad de archivos y bibliotecas necesarias fue minimizada, a fin de facilitar su entendimiento y funcionamiento.

4.2.4 - LIBNET

En la implementación del protocolo, son necesarias dos herramientas. Una que envié paquetes por la red y otra que los capture.

LibNet [43] es una biblioteca escrita en lenguaje C que permite que los programadores construyan paquetes para ser enviados por la red a través de cualquier plataforma. La construcción de la biblioteca tiene dos razones principales. De entrada, la biblioteca fue escrita para establecer un interface simple entre programadores y la programación de bajo nivel para redes TCP/IP. La segunda razón fue disminuir las dificultades que los programadores tenían debido a la falta de estándares para los programas en redes TCP/IP.

Hasta la implementación de este trabajo, la biblioteca estaba en su versión 0.10.11.

La elección de esta herramienta es debida a la facilidad en el montaje de paquetes y la inserción de los mismos en la red. El único problema de esta herramienta es que no trabaja muy bien con timeouts en algunas plataformas. Este fallo se describe por los desarrolladores de la herramienta y fue constatado durante las pruebas del protocolo.

4.2.5 - LIBPCAP

Pcap es un interfaz de una aplicación de programación para captura de paquetes. La implementación del pcap para sistemas basados en Unix se conoce como libpcap. El libpcap puede ser utilizado por un programa para capturar los paquetes que viajan por toda la red y, en las versiones más recientes, para transmitir los paquetes en la capa de enlace de una red, así como para conseguir una lista de los interfaces de red que se pueden utilizar con el libpcap.

El libpcap es la captura del paquete y el motor de filtración de muchas herramientas de código abierto y comerciales de la red, incluyendo analizadores de

protocolo, monitores de la red, sistemas de detección de intrusos en la red , programas de captura de las tramas de red (packet sniffers) , generadores de tráfico y puesta a punto de la red. Algunas de estas herramientas, tales como tcpdump, Ethereal, Nmap , Cain y Abel , y Snort son conocidas y utilizadas a través de la red y de una comunidad de seguridad internacional.

Libpcap [44] es una biblioteca escrita en el lenguaje C que permite que los programadores capturen y recuperen paquetes de la red con la utilización de filtros BPF (*Berkeley Packet Filter*). La biblioteca Pcap fue desarrollada por Van Jacobson, Craig Leres y Steven McCanne, todos de la Universidad de California, y que ya desarrollaron innumerables proyectos conocidos como el *tcpdump*, *snort*, *arpwatch*, *fragrouter*, entre otros. Este poderoso y simple mecanismo es ideal para monitorear el tráfico y puede constituir un peligroso *sniffe*r de señal para varias aplicaciones sin criptografía.

La utilización de esta herramienta está especificada en el análisis de paquetes que están llegando a la capa de enlace. Claramente, es necesario especificar una interface para la captura de paquetes y libpcap puede hacer esta tarea automáticamente para el programador. En la programación de libpcap, el programador puede escoger entre utilizar el modo promiscuo o el modo no promiscuo, escogiendo el modo que mejor se adapte a sus objetivos.

Las dos técnicas son muy diferentes. El modo de captura no promiscuo solamente captura el tráfico con destino a su propia dirección de hardware. El modo promiscuo funciona como un “agujero negro”, capturando todo el tráfico de broadcast de la red. Su ventaja es poder capturar un mayor número de paquetes, sin embargo esta estrategia debe ser definida para un uso que no ocupe muchos recursos del sistema. Este puede ser hecho aplicando filtros BPF.

La libpcap tiene algunas funciones para recepción y procesamiento de cada paquete. El control del procesamiento de paquetes puede ser ajustado fijándose un límite o configurarlo para procesar infinitamente. Cada paquete puede ser analizado a través de una función llamada *callback*, en la cual son reconstituidos los formatos de cabecera y datos para una visión humana.

Para el funcionamiento de esta herramienta se utiliza el modo promiscuo para la captura de paquetes de la red, visto que el protocolo necesita de comunicación broadcast en algunos intercambios de mensajes.

4.2.6 – WIRELESS EXTENSIONS FOR LINUX

Esta herramienta desarrollada por Jean Tourriles [42] es una interfaz para la programación de Aplicaciones (API) de redes no cableadas para el sistema operativo Linux. Las extensiones para redes no cableadas son divididas en tres partes complementarias.

La primera parte consiste en la interfaz con el usuario, que es un conjunto de herramientas para manipular estas extensiones. La segunda parte es una modificación del Kernel para soportar y definir todas estas extensiones. Ya la tercera y última parte trata de la interna de hardware y sus archivos de configuración (drivers).

Para la configuración del entorno de implementación, se usa una herramienta para configurar todos los parámetros del hardware. Se trata del comando *iwconfig* que es una copia del comando *ifconfig*, propietario del Linux. Los siguientes parámetros están disponibles:

Tabla 4.1 – Parámetros de configuración del comando iwconfig.

Parámetro	Descripción
ESSID	Identificador único de la red. Ha de ser el mismo para establecer comunicación entre nodos de la red.
Mode	Modo de operación de la red. Puede operar en modo Administrador o en modo Ad-Hoc. Usaremos este último para el caso que nos ocupa.
Frequency	Frecuencia utilizada por la red ad hoc
Bit Rate	Tasa de transmisión
Sens	Nivel de señal de la red, comúnmente llamado de sensitividad.
Encryption	Clave criptográfica

Sin ningún argumento, la ejecución del comando *iwconfig* muestra las siguientes configuraciones de una interface de red no cableada:

```
[root@localhost root]# iwconfig
lo        no wireless extensions.

wlan0     IEEE 802.11-DS  ESSID:"LabRedes"  Nickname:"HERMES I"
          Mode:Ad-Hoc  Frequency:2.422GHz  Access Point: 00:02:2D:4B:3B:69
          Bit Rate:11Mb/s   Tx-Power=15 dBm   Sensitivity:1/3
          Retry limit:4   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:53/92  Signal level:-43 dBm  Noise level:-97 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:1
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

4.3 – DEFINICIÓN Y ESPECIFICACIÓN DE LOS ESTADOS DEL PROTOCOLO

Para la implementación del protocolo se realizaron una serie de estados por los que cada nodo tiene que ir moviéndose para realizar las diferentes funcionalidades del protocolo.

Cada diagrama consiste en varios estados que modularizan el protocolo, volviéndolo más flexible y portable para futuras aplicaciones. Esos estados son explicados detalladamente en los siguientes apartados.

4.3.1 – ESTADOS DE ENTRADA DE NODOS

A continuación se describen los estados relacionados con la entrada de nodos en la red. Tanto para el nodo cliente, o entrante, el nodo servidor, y el resto de nodos activos de la red.

4.3.1.1 – ESTADOS DEL NODO CLIENTE.

Se explican los diferentes estados por los que pasa un nodo que desea entrar en la red.

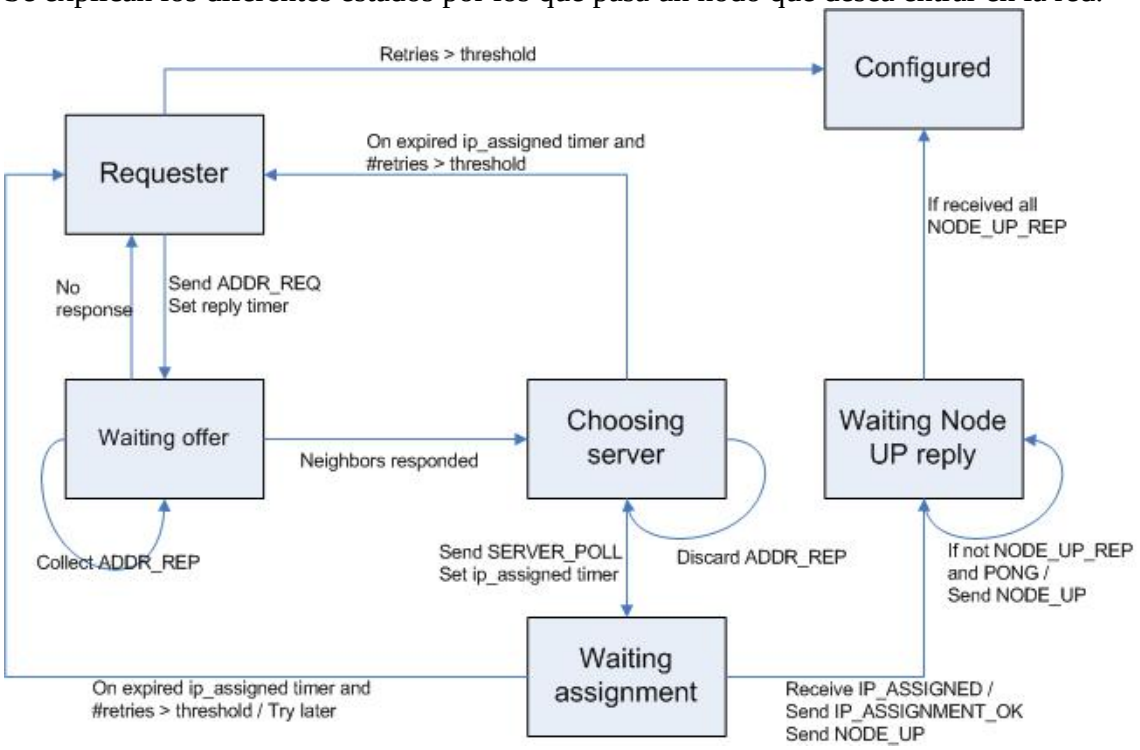


Figura 4.1 – Estados del nodo cliente en la entrada de nodos.

Estado Requester.

En este estado el nodo entrante envía un mensaje addr_req para que sea oído por algún miembro de la red. Pasa a esperar una respuesta al estado Waiting Offer. Si tras varios intentos no se recibe respuesta alguna, pasará al estado Configured configurándose como nodo líder de una red nueva.

Estado Waiting Offer.

Para este estado el nodo recogerá todas las respuestas en forma de addr_rep que lleguen. Activará el temporizador para delimitar la espera de recepción de mensajes. Pasará al estado Choosing Server.

Estado Choosing Server.

De todas las respuestas recibidas se quedará con aquella que ofrezca mayor número de direcciones libres. A este nodo le responderá con un mensaje Server_poll.

Estado Waiting Assignment.

Tras haber seleccionado un servidor para que gestione su entrada en la red deberá esperar en este estado la recepción del mensaje ip_assigned. Una vez recibido utilizará la información contenida en el mensaje para configurarse en la red y asignarse una dirección. Si salta el temporizador habrá fracasado el intento de configuración en la red. Sino notificará al resto de nodos de la red su entrada en la misma mediante node_up y pasará al estado Waiting Node UP Reply.

Estado Waiting Node UP Reply.

Aquí el nodo deberá esperar la respuesta de todos los nodos integrantes de la red. Para lo cual se establecerán temporizadores. Si en un determinado tiempo algún nodo no responde a los node_up enviados, se comprobará mediante PING su estado activo en la red y en caso afirmativo se reenviará de nuevo un node_up. Con la recepción de todos los mensajes node_up_rep el nodo entrante podrá construir con información actualizada la tabla Allocated y tener información veraz de la topología de la red en la que se ha configurado.

Estado Configured.

Una vez superado el proceso de configuración el nodo pasará a un estado configurado. A partir de este punto el podrá convertirse en servidor de direcciones a otros nodos entrantes.

4.3.1.2 – ESTADOS DEL NODO SERVIDOR.

El nodo que recibe una petición de otro que desea entrar en la red pasa por los siguientes estados.

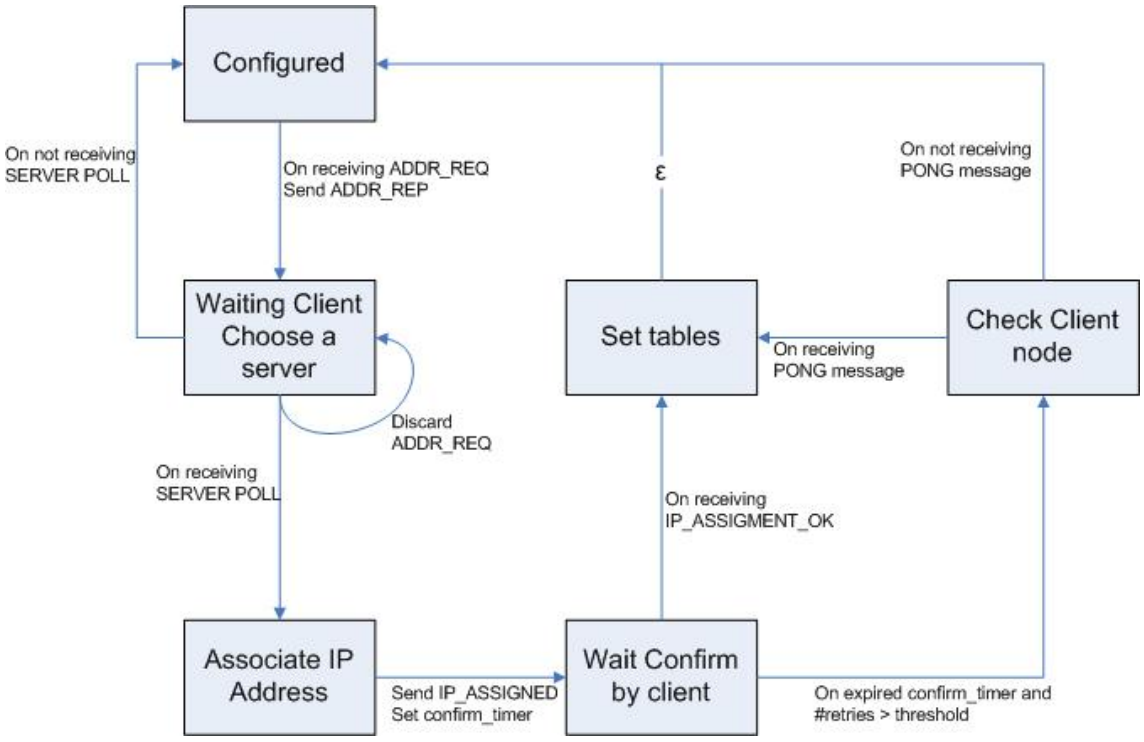


Figura 4.2 – Estados del nodo servidor en la entrada de nodos.

Estado Waiting Client Choose A Server.

El nodo configurado recibe un addr_req de un nodo entrante solicitando ser configurado. El nodo servidor ha de responder con un addr_rep indicando las direcciones libres que ofrece al nodo cliente. Una vez hecho esto el nodo servidor esperará un tiempo determinado la llegada de la respuesta de confirmación por parte del nodo cliente en forma de server_poll. Solo si esta llega pasará al estado Associate IP Address.

Estado Associate IP Address.

Llegado a este estado el nodo servidor se considera elegido para servir direcciones al nodo entrante. Enviará un mensaje ip_assigned con información para el nodo cliente. Pasa entonces al estado Wait Confirm By Client.

Estado Wait Confirm By Client.

Como dice su nombre, este estado espera la confirmación del cliente indicando que ha recibido satisfactoriamente ip_assigned y ha asumido esas direcciones libres. Entonces en este caso el nodo servidor pasará a adecuar sus tablas a la nueva situación.

Estado Check Client Node.

Si el temporizador activado para la espera del mensaje ip_assigned_ok salta, comprobará mediante PING su estado activo en la red. Si responde se interpretará que el mensaje ip_assigned_ok se perdió.

Estado Set Tables.

Actualiza las tablas del nodo servidor para tener los datos actualizados de la situación de la red. Tras esto pasa a estar de nuevo en estado Configured.

4.3.1.3 – ESTADOS DEL RESTO DE NODOS.

Se define el cambio de estado que sufre un nodo presente y configurado en la red al recibir la notificación de que un nuevo nodo ha entrado en la red.

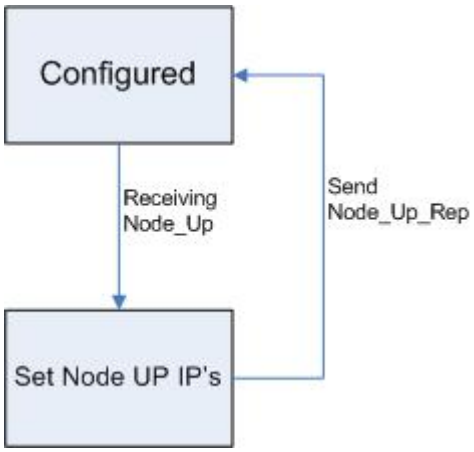


Figura 4.3 – Estados de otro nodo en la entrada de nodos.

Estado Set Node UP IP's.

El único estado que nos encontramos es cuando un nodo configurado en la red recibe un node_up. Tras lo cual actualiza las tablas en consecuencia de los cambio producidos en la red y envía un mensaje de confirmación node_up_rep.

4.3.2 – ESTADOS DE SALIDA DE NODOS

Podemos definir dos tipos de salida. Salida fácil, donde el nodo advierte de su intención de abandonar la red y entrega sus direcciones. Y salida brusca donde un nodo sale abruptamente de la red y el resto de nodos se percata a posteriori de la caída de otro.

4.3.2.1 – SALIDA FACIL

Aquí se presentan los estados para el proceso de abandono de la red de un nodo en modo de salida fácil.

4.3.2.1.1 – ESTADOS DEL NODO CLIENTE.

El nodo que desea salir avisa al nodo responsable de sus direcciones su intención y le entrega las direcciones libres.

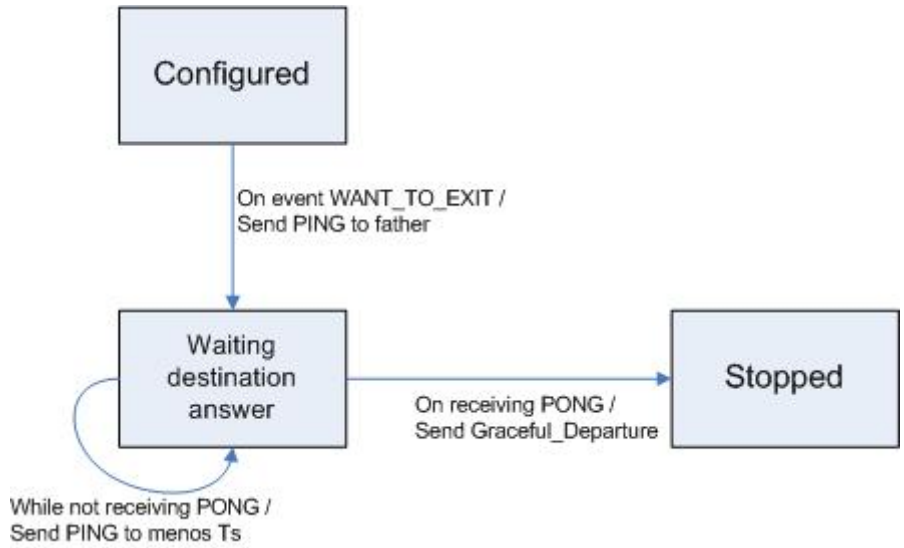


Figura 4.4 – Estados del nodo cliente en la salida fácil.

Estado Waiting destination answer.

Un nodo configurado que desea salir deberá entregar sus direcciones al nodo responsable en la red. Este será en principio el nodo que tenga en su tabla allocated como padre si todavía se encuentra activo. Sino será el nodo más prioritario de la red. Para lo cual comprueba que su padre está aun en la red mediante PING. Sino buscará el nodo más prioritario de la red y le enviará un mensaje graceful_departure entregándole las direcciones liberadas.

Estado Stopped.

El nodo se desconecta.

4.3.2.1.2 – ESTADOS DEL NODO SERVIDOR.

En este caso entendemos como nodo servidor como aquel responsable de las direcciones del nodo que abandona la red. A continuación los estados que atraviesa.

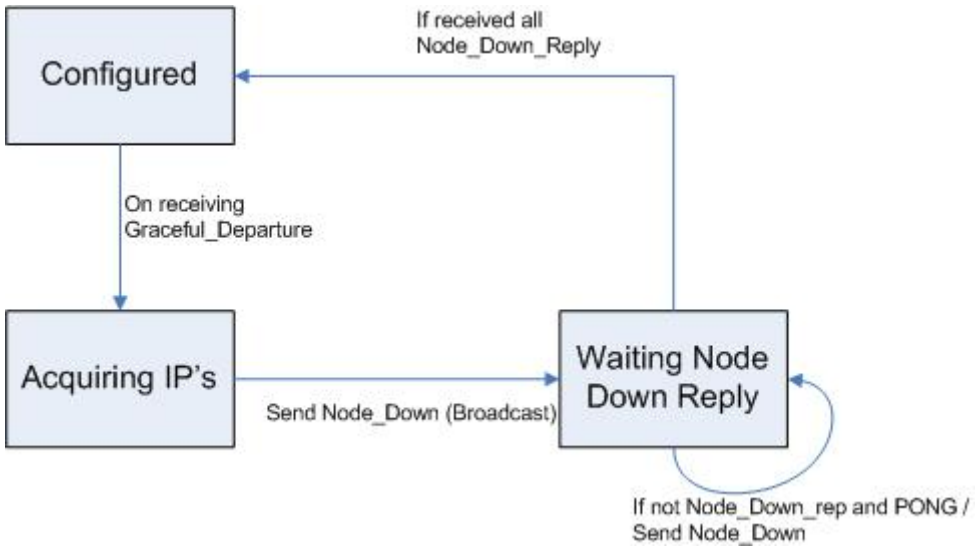


Figura 4.5 – Estados del nodo servidor en la salida fácil.

Estado Acquiring IP's.

Un nodo configurado en la red recibe un mensaje graceful_departure con una serie de direcciones de las cuales tiene que hacerse responsable. Una vez actualizadas sus tablas notificará al resto de nodos la caída de un nodo mediante node_down. Pasará entonces al estado Waiting Node Down Reply.

Estado Waiting Node Down Reply.

Aquí el nodo deberá esperar la respuesta de todos los nodos integrantes de la red a los mensajes node_down anteriormente enviados. Para lo cual se establecerán temporizadores. Si en un determinado tiempo algún nodo no responde se comprobará mediante PING su estado activo en la red y en caso afirmativo se le reenviará de nuevo un node_down. Una vez recibida la confirmación de todos los nodos pasará a estado Configured.

4.3.2.1.3 – ESTADOS DEL RESTO DE NODOS.

Los nodos ajenos a este proceso de salida también deben ser informados de la caída de un nodo y tienen un proceso de cambio de estado.

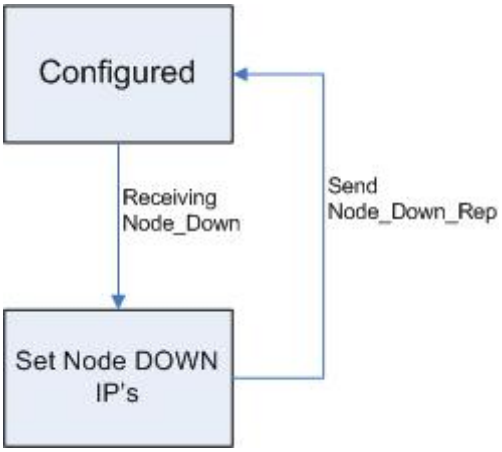


Figura 4.6 – Estados de otro nodo ajeno en la salida fácil.

Estado Set Node Down IP's.

Si un nodo en estado configurado recibe un node_down ha de responder con el mensaje node_down_rep. Con la información de node_down actualiza las tablas en consecuencia de los cambio producidos en la red.

4.3.2.2 – SALIDA BRUSCA

Esta salida en la que se produce un fallo de un nodo y algún nodo ha de hacerse responsable de la recuperación de las direcciones libres perdidas.

4.3.2.2.1 – ESTADOS DE NODO SERVIDOR.

En un determinado momento cada nodo controla el estado de los nodos de los que él es responsable. Para lo cual se sigue este diagrama de estados.

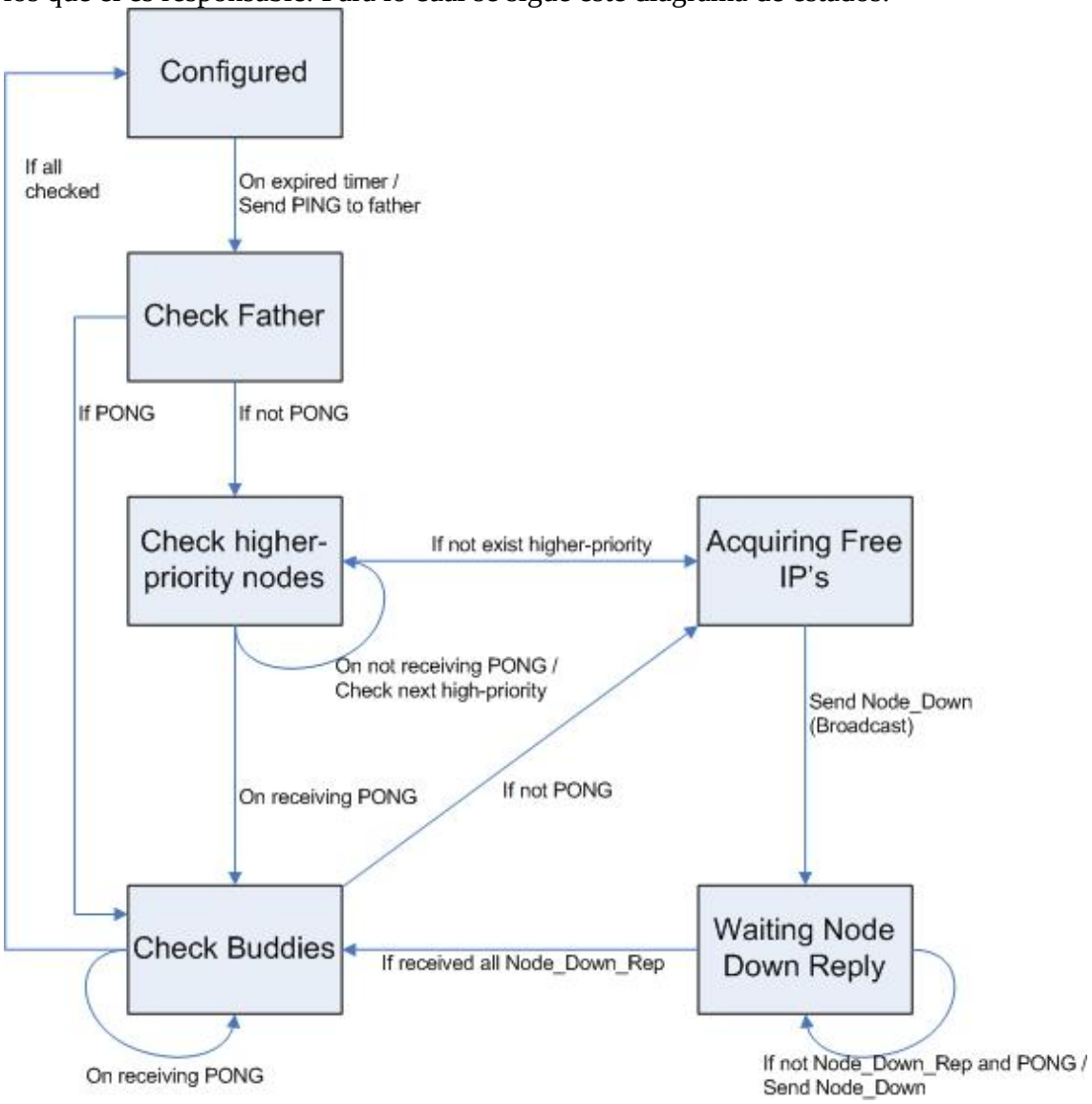


Figura 4.7 – Estados del nodo servidor en la salida brusca.

Estado Check Father.

Cada determinado tiempo, fijado por el temporizador de sincronización, los nodos configurados pasarán por un proceso de recuperación de direcciones. El primer paso es comprobar, en caso de que exista, el estado activo o no del nodo padre en la red. Si el nodo padre responde a un PING pasará a comprobar que los hijos siguen activos en la red en el estado Check Buddies. En caso contrario pasará al estado Check Higher-priority nodes.

Estado Check Higher-priority nodes.

El padre ha caído de la red. Por tanto el nodo deberá comprobar si existe un nodo con más prioridad que él para hacerse cargo de las direcciones libres del padre. Checkea mediante PING, basándose en los datos de su tabla allocated, si están activos nodos con menor timestamp. En caso negativo se hará el nodo responsable y adquirirá las direcciones en Acquiring Free IP's. En caso positivo no hará nada ya que supondrá que el nodo que le corresponda se hará responsable, y pasará al estado Check Buddies.

Estado Check Buddies.

Estado de comprobación de los nodos hijos de un nodo. Tras comprobar que todos siguen en la red pasa al estado Configured. Si algún nodo hijo ha caído tendrá que responsabilizarse de sus direcciones y pasa al estado Acquiring Free IP's.

Estado Acquiring Free IP's.

El nodo copiará las direcciones que quedan liberadas en su free_ip_block. Una vez actualizadas sus tablas notificará al resto de nodos la caída de un nodo mediante node_down. Pasará entonces al estado Waiting Node Down Reply

Estado Waiting Node Down Reply.

Aquí el nodo deberá esperar la respuesta de todos los nodos integrantes de la red. Para lo cual se establecerán temporizadores. Si en un determinado tiempo algún nodo no responde se comprobará mediante PING su estado activo en la red y en caso afirmativo se reenviará de nuevo un node_down. Una vez recibida la confirmación de todos los nodos pasará a estado Check buddies para comprobar los hijos.

5 – CONCLUSIONES Y SUGERENCIAS FUTURAS

5.1 – CONCLUSIONES

Para la realización de este trabajo se han estudiado los principales protocolos de autoconfiguración en redes móviles ad hoc. Se puso especial atención en las operaciones básicas de los protocolos de autoconfiguración de este tipo de redes como: asociación de nuevas direcciones, salida de nodos, caída de nodos, perdida de mensajes, fusión y separación de redes.

La infraestructura proyectada e implementada en este trabajo demostró que es posible el desarrollo de protocolos de comunicación de datos utilizando herramientas gratuitas en Internet con bajo costo y recursos limitados.

La implementación del protocolo fue de suma importancia para validar los conceptos abordados en su especificación.

5.2 – SUGERENCIAS PARA FUTURAS INVESTIGACIONES

Por ser una tecnología relativamente nueva, las redes móviles ad hoc aún necesitan de soluciones para varias cuestiones. El grupo de trabajo MANET ha juntado esfuerzos para estandarizar los protocolos desarrollados tanto en enrutamiento como en seguridad y autoconfiguración. Sin embargo, aún no se vio una aplicación que demande el uso comercial a gran escala de las redes móviles ad hoc. Por lo tanto, las investigaciones futuras aún son necesarias para el desarrollo de la tecnología.

Como este es el primer proyecto del protocolo de autoconfiguración, aún posee algunas funcionalidades que deben ser mejoradas.

Versiones futuras de este protocolo necesitan de investigación adicional para la mejoría de la distribución de las direcciones, disminución del tiempo necesario para la realización del proceso de autoconfiguración, así como la implementación de funcionalidades extras.

La seguridad, es un elemento muy importante que no se ha desarrollado en este protocolo. Y es uno de los problemas más importantes con los que se encuentra una red inalámbrica, teniendo en cuenta los usos a los que se aplica esta tecnología. Con lo cual crear un modulo de seguridad sería indispensable para un uso empresarial o militar de la red.

El enrutado es otro elemento no desarrollado en este protocolo y muy importante a tener en cuenta por la característica de la red de topología cambiante.

Desarrollo de un módulo adicional para el funcionamiento en redes que utilizan el protocolo de comunicación IPv6.

Por ello aun quedarían muchos apartados que desarrollar para realizar un protocolo completo y seguro. Siendo un tema para desarrollos futuros en el área de la autoconfiguración de direcciones en las redes móviles ad hoc.

BIBLIOGRAFÍA

[1] http://www.uta.fi/EGEDL/outline/packetradiotechnology.html

[2] Robert E. Kahn, Steven A. Gronemeyer, Jerry Burchfiel, and Ronald C. Kunzelman. *Advances in Packet Radio Technology*. Proceedings of the IEEE, vol 66, no 11, November 1978.

[3] John Jubin and Janet D. Tornow. *The DARPA Packet Radio Network Protocols*. Proceedings of the IEEE, vol 75, no 1, January 1987.

[4] http://www.ietf.org/html.charters/manet-charter.html

[5] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance. Issues and Evaluation Considerations. RFC 2501, Junio 1999.

[6] Swetha Narayanaswamy, Vikas Kawadia, R. S. Sreenivas and P. R. Kumar. *The COMPOW protocol for power control in ad hoc networks: Theory, architecture, algorithm, implementation, and experimentation*. May 16, 2001.

[7] Javier Gmez, Andrew T. Campbell, Mahmoud Naghshineh and Chatschik Bisdikian. *PARO: Conserving Transmission Power in Wireless ad hoc Networks*. IEEE 9th International Conference on Network Protocols (ICNP'01), Riverside, California. November 2001.

[8] C.K. Toh. *Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks*. IEEE Communications, June 2001.

[9] J. Lundberg. *Routing Security in Ad Hoc Networks*. Helsinki University of Technology, 2000.

[10] L. Joe and Philip M. Feldman. *Fundamental Research Policy for the Digital Battlefield*. RAND (http://www.rand.org), 1998.

[11] C. C. Cheng, et. al. *A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect*. ACM Sigcomm 1989, pp. 224-236.

[12] J.J. Garcia-Luna-Aceves. *A Fail-Safe Routing Algorithm for Multihop Packet-Radio Networks*. In INFOCOM. IEEE, Miami, Florida, April 1986.

[13] J. Hagouel. *Issues in Routing for Large and Dynamic Networks*. PhD thesis, Columbia University, May 1983.

[14] P. Humblet, *Another Adaptive Shortest-Path Algorithm*. IEEE Trans. Commun. (June 1991). Helsinki University of Technology. Networking Laboratory.

[15] B. Rajagopalan and M. Faiman. *A Responsive Distributed Shortest-Path Routing Algorithm with Autonomous Systems*. Internetworking: Research and Experience, 2:51–69, March 1991.

[16] M.S. Corson and J. Macker. *Mobile Ad hoc Networking (MANET):*

Routing Protocol Performance Issues and Evaluation Considerations. Request For Comments 2501, Internet Engineering Task Force, January 1999.
[17] http://www.europa.eu.int/comm/energy transport/en/gal en.html
[18] Tanenbaum, Andrew S., *Computer Networks*, Prentice Hall Inc, USA, 1996.
[19] K. Obraczka and G. Tsudik. *Multicast routing issues in ad hoc networks*.
[20] Sung-Ju Lee. *Routing and Multicasting Strategies in Wireless Mobile Ad hoc Networks*. University of California, Los Angeles, Computer Science Department, September 2000.
[21] Feeney L.M. *A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks*. SICS Technical Report T99:07. Swedish Institute of Computer Science, Kista, Sweden, October 1999.
[22] Charles E. Perkins, Elizabeth M. Royer, and Samir Das. *Ad Hoc On Demand Distance Vector (AODV) Routing*. IETF Internet draft, draftietf-manet-aodv-09.txt, November 2001 (work in progress).
[23] Johnson, D.B., and Maltz, D.A. *Dynamic Source Routing in Ad-Hoc Wireless Networks*. Mobile Computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer, 1996.
[24] Haas, Z.J. *A Routing Protocol for the Reconfigurable Wireless Networks*. IEEE ICUPC'97, San Diego, CA, October 12-16, 1997.
[25] M. Jiang, J. Li, and Y.C. Tay. *Cluster-Based Routing Protocol (CBRP)*. draft-ietf-manet-cbrpspec-01.txt, Internet Draft, IETF, Aug. 1999.
[26] P. Jacquet and T. Clausen. *Optimized Link State Routing Protocol*. IETF Internet draft, draft-ietf-manet-olsr-06.txt, September 2001 (work in progress)
[27] S. Murthy and J.J Garcia-Luna-Aceves. *An Efficient Routing Protocol for Wireless Networks*. ACM Mobile Networks and Applications Journal, Oct. 1996.
[28] T.-W.Chen and M.Gerla. *Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks*. In Proc. of IEEE ICC'98, Atlanta, GA, pages 171-175, Jun. 1998.
[29] G. Pei, M. Gerla and T.-W. Chen. *Fisheye State Routing in Mobile Ad Hoc Networks*. In Proc. of the 2000 ICDCS Workshops, Taipei, Taiwan, Apr. 2000, pp. D71-D78.
[30] B. A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. *Scalable Routing Strategies for Ad Hoc Wireless Networks*. IEEE Journal on Selected Areas in Communications (JSAC). Issue on Wireless Ad Hoc Networks, pages 1369 -1379, Vol. 17, No. 8, August 1999.
[31] M. Joa-Ng and I.-T. Lu, A Peer-to-Peer zone-based two-level link state

routing for mobile Ad Hoc Networks, IEEE Journal on Selected Areas in Communications, Vol. 17(8), Aug. 1999, pp. 1415-1425.

[32] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and M. Gerla. *Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel*. In Proc. of IEEE Singapore International Conference on Networks (SICON'97).

[33] M.S. Corson and A. Ephremides. *A Distributed Routing Algorithm for Mobile Wireless Networks*. ACM/Baltzer Wireless Networks, vol.1, no.1, pp.61-81, February 1995.

[34] V. Park and S. Corson. *Temporally-Ordered Routing Algorithms (TORA) Version 1 Functional Specification*. IETF Internet Draft, draft-ietf-manet-tora-spec-01.txt, Aug. 1998 (work in progress).

[35] C.K. Toh. *Associativity-Based Routing for Ad-Hoc Mobile Networks*. Wireless Personal Communications, Vol. 4, No. 2, pp. 1-36, Mar. 1997.

[36] R. Dube, C.D. Rais, K.-Y. Wang, S.K. Tripathi. *Signal stability-based adaptive routing (SSA) for ad hoc mobile networks*. IEEE Personal Communications, vol.4, no.1, pp.36-45, February 1997.

[37] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad hoc Networks", IEEE WCNC, 2003.

[38] N. H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad hoc Networks," in Proc. of ACM MobiHoc 2002, Lausanne, Switzerland, June 2002, pp. 206-216.

[39] M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad hoc Network", IEEE Milcom 2002.

[40] A. Shamir – How to Share a Secret. Communications of the ACM, 22(11):612-613, 1979.

[41] Archan Misra, Subir Das, Anthony McAuley, and Sajal K. Das, "Autoconfiguration, Registration and Mobility Management for Pervasive Computing", IEEE Personal Communication, August 2001, pp 24-31

[42] Wireless Tools for Linux. Actualizado el 14/02/2006. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

[43] The Libnet Packet Construction Library. Actualizado el 19/01/2005. http://www.packetfactory.net/projects/libnet/

[44] Libpcap. Actualizado en 16/12/2005. http://www.tcpdump.org/pcap.htm

Los abajo firmantes: Adam Ameziane y Miguel Angel Tolosa Diosdado, autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, y mencionando expresamente a sus autores, tanto la presente memoria como el código, la documentación y/o el prototipo desarrollado.

Adam Ameziane Miguel A. Tolosa Diosdado