Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Estudio y análisis de prestaciones de redes móviles Ad Hoc mediante simulaciones NS-2 para validar modelos analíticos

Autor: Jordi Chalmeta Ugas

Directora: Mónica Aguilar Igartua

Noviembre 2009

Agradecimientos

Quiero agradecer a las siguientes personas por la ayuda que me han proporcionado durante la realización de este Proyecto Final de Carrera:

Aída Zavala Ayala

Por su ayuda en la realización de este Proyecto Final de Carrera en el que hemos trabajado conjuntamente durante la realización de su tesis doctoral. También por haber co-dirigido y supervisado el proyecto.

Mónica Aguilar Igartua

Por dirigir este proyecto eficazmente y por su ayuda siempre que la he necesitado. También por su supervisión del proyecto y por haberme brindado la ocasión de realizar este proyecto.

Ricard Sauch Muray

Por su ayuda y apoyo moral, así como sus consejos y explicaciones sobre todos los temas burocráticos que conoce muy bien.

A mi familia

Por haberme ayudado y apoyado durante el tiempo que he estado realizando este proyecto.

A mis amigos

Por haberse interesado por mi proyecto, haberme ayudado cuando los he necesitado y por su apoyo moral durante este tiempo.

Índice

Índice de figuras

Índice de tablas

Glosario de acrónimos

ABR	Associativity-Based Routing
ACK	Acknowledge
ADV	Adaptative Distance Vector
AODV	Ad hoc On Demand Distance Vector
ARP	Address Resolution Protocol
AWK	A. Aho, P. Weinberger y B. Kernighan (autores)
CBR	Constant Bit Rate
CBRP	Cluster-Based Routing Protocol
CDMA	Code Division Multiple Access
CEDAR	Core-Extraction Distributed Ad Hoc Routing
CGSR	Cluster Gateway Switch Routing
CSMA/CA	Carrier Sense Multiple Acces Collision Avoidance
CTS	Clear To Send
DARPA	Defence Advanced Research Projects Agency
DREAM	Distance Routing Effect Algorithm for Mobility
DSR	Dynamic Source Routing Protocol
DSDV	Destination-Sequenced Distance Vector
DYMO	Dynamic MANET On demand
FDMA	Frequency Division Multiple Access
FQMM	Flexible QoS Model for MANET
FTP	File Transfer Protocol
HSR	Hierarchical State Routing
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IR	Infrared Light
LAR	Location-Aided Routing
LBNL	Lawrence Berkeley National Laboratory
LL	Link Layer
MAC	Medium Access Control
MANET	Mobile Ad hoc NETworks
MANET-WG	MANET Work Group
MIT	Massachusetts Institute of Technology
MMWN	Multimedia support in Mobile Wireless Networks
MRTP	Multiflow Realtime Transport Protocol

NAM	Network AniMator
NS-2	Network Simutalor 2
OLSR	Optimised Link State Routing
OSI	Open Systems Interconnection
OTCL	Object Tool Command Language
PAN	Personal Area Networks
PC	Personal Computer
PDA	Personal Digital Assistant
PLCP	Physical Layer Convergence Procedure
QOLSR	QoS Object Tool Command Language
QoS	Quality of Service
RERR	Route Error
RF	Radio Frequence
RFC	Request For Comments
RREP	Route Replay
RREQ	Route Request
RTS	Request To Send
SCTP	Stream Control Transport Protocol
SLURP	Scalable Location Update Routing Protocol
STAR	Source-Tree Adaptative Routing
TBP	Ticket-Based Probing
TBRPF	Topology dissemination Based on Reverse-Path Forwarding
TCL	Tool Command Language
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TORA	Temporally Ordered Routing Algorithm
TPA	Transport Protocol for Ad Hoc Network
TTL	Time-to-Live
UCB	University of California, Berkeley
UDP	User Datagram Protocol
USC/ISI	University of Southern California's Information Sciences Institute
VINT	Virtual InterNetwork Testbed
WRP	Wireless Routing Protocol
Xerox PARC	Xerox Palo Alto Research Center
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

Capítulo 1. Introducción

Hoy en día es difícil entender la sociedad sin Internet. Desde finales del siglo pasado la popularidad de Internet no ha parado de crecer y su expansión es tal que en el año 2006 Internet alcanzó los 1.100 millones de usuarios en el mundo. Su uso no es sólo por interés personal sino que abarca todos los ámbitos de la vida desde el trabajo o los ámbitos académicos hasta simplemente el ocio personal. En la sociedad actual, existe pues una gran necesidad de acceder rápidamente a la información que ofrece Internet, es por eso que el acceso a Internet ya no se limita a la conexión telefónica de un PC. En los últimos años el acceso a Internet se ha diversificado apareciendo, entre otras, las redes inalámbricas, las cuales pueden ser con infraestructura fija o sin ella. Las más conocidas son las que cuentan con una infraestructura fija, pero se dan casos en los cuales es imposible o muy difícil disponer de dicha infraestructura. Es en este punto dónde cobran gran importancia las redes Ad Hoc, en las que los terminales actúan no sólo como receptores o emisores sino también como encaminadores.

En este sentido, las redes inalámbricas móviles, o MANET (*Mobile Ad hoc NETworks*), están creciendo en importancia recientemente. Pero en este campo todavía hay muchos temas de estudio que se encuentran en evolución hoy en día. Los estudios se centran en campos como la calidad de servicio ofrecida a los usuarios, la seguridad de las redes, la autonomía de las baterías de los terminales, etc.

Dentro de este amplio campo de estudios se está realizando mucha investigación para mejorar el funcionamiento de las MANET`s. El Grupo de Servicios Telemáticos de la UPC [9, 10, 11] está enfocando sus esfuerzos en dotar a las redes Ad Hoc con Calidad de Servicio (QoS, *Quality of Service*). Este Proyecto Final de Carrera nace dentro del Grupo de Servicios Telemáticos y se une al trabajo de investigación que está realizando dicho grupo.

El objetivo principal de este Proyecto Final de Carrera es validar mediante simulaciones un modelo matemático desarrollado dentro del Grupo de Servicios Telemáticos de esta Universidad, en concreto los modelos pretenden constituir una herramienta de soporte al diseño, implementación y evaluación de arquitecturas de red basadas en redes inalámbricas Ad Hoc. Así, el ingeniero o ingeniera dispondrá de una herramienta flexible, económica y fácilmente modificable sin tener que realizar simulaciones costosas en tiempo. Además, le será muy útil en la fase de diseño, antes de proceder a extender implementaciones reales en *testbeds*, que además son costosos económicamente. Es objetivo de este PFC el validar dicho modelos analíticos, mediante simulaciones. Este paso es necesario para poder utilizar la herramienta analítica con total confianza.

Para alcanzar el objetivo principal este proyecto está compuesto por una serie de objetivos secundarios igualmente importantes. Para empezar, uno de los objetivos secundarios es analizar y entender perfectamente el funcionamiento del protocolo de encaminamiento DSR [8] (*Dynamic Source Routing*) para poder interpretar los resultados de las simulaciones. El DSR no es el único protocolo de encaminamiento para las MANET, pero es el que se ha escogido para realizar este proyecto por ser fácilmente extensible con nuevas aportaciones, como en [9]. Este protocolo utiliza unos mecanismos de descubrimiento y mantenimiento de rutas complejo, el cual hay que comprender previamente para poder obtener los resultados deseados.

Para poder alcanzar el objetivo principal de este proyecto es indispensable conocer y aprender a manejar correctamente las herramientas necesarias para realizar las simulaciones e interpretar los datos arrogados por éstas. En este sentido, un objetivo de gran importancia es estudiar ampliamente el funcionamiento del programa NS-2 [15] (*Network Simulator 2*) que ha sido el escogido para realizar las simulaciones de este proyecto.

El NS-2 es un simulador de redes ampliamente usado y muy potente, sin embargo, su uso y entendimiento no son triviales. De aquí se deriva otro de los objetivos de este Proyecto de Fin de Carrera, los resultados que se obtienen con el NS-2 son ficheros de datos [19] difíciles de interpretar. Para interpretar los resultados de las simulaciones hace falta otra herramienta de tratamiento de datos, que en este caso, es el AWK [20], un programa que permite extraer la información de los resultados de las simulaciones. Así que otro objetivo es conocer y familiarizarse con el lenguaje AWK para poder analizar los resultados.

Dicho esto se puede entender como se ha estructurado por capítulos este proyecto.

En el segundo capítulo se explica que son las redes MANET y sus características más importantes. El tercer capítulo explica ampliamente el protocolo de encaminamiento DSR, el cual es fundamental para entender este proyecto.

En el cuarto capítulo se explica el modelo matemático a validar, el quinto capítulo enumera las características de los escenarios simulados.

A continuación, en los capítulos sexto y séptimo, se ven en profundidad las herramientas utilizadas en este proyecto. Empezando por entender el funcionamiento del NS-2 y siguiendo por como interpretar los resultados obtenidos de las simulaciones con la ayuda del lenguaje AWK.

En el capítulo octavo se presentan los resultados más significativos obtenidos mediante simulaciones en este proyecto y se comparan con los obtenidos mediante el modelo matemático. En el noveno y último capítulo se presentan las conclusiones que se extraen de los resultados previamente citados y las líneas futuras de estudio.

Capítulo 2. Redes MANET (*Mobile Ad hoc NETworks*)

Las redes Ad Hoc consisten en un conjunto de nodos que se comunican mediante enlaces inalámbricos y que no tienen una infraestructura fija. Esto implica que no tienen ningún tipo de control centralizado y que por lo tanto son flexibles y fácilmente desplegables. Dentro de las redes Ad Hoc existen varios tipos: las redes de sensores, las redes mesh, las redes vehiculares (VANET, *Vehicular Ad Hoc Network*) y las redes móviles Ad Hoc (MANET, *Mobile Ad Hoc Network*). Este proyecto se centrará básicamente en el último de los tipos mencionados, las MANETs.

Las MANET, como su nombre indica, añaden la característica de movilidad de los nodos, eso conlleva que los enlaces entre nodos se rompan y que haya enlaces que se incorporen o abandonen la red continuamente, por lo tanto la topología de la red es altamente cambiante y aleatoria. Todos los nodos pueden actuar tanto como emisores, receptores y *routers* (o encaminadores), esto es necesario ya que las rutas para llegar a un destino pueden tener varios saltos. Los nodos pueden ser dispositivos tales como ordenadores portátiles, PDAs (*Personal Digital Assistant*), teléfonos móviles, etc. Veamos un ejemplo de una red MANET:

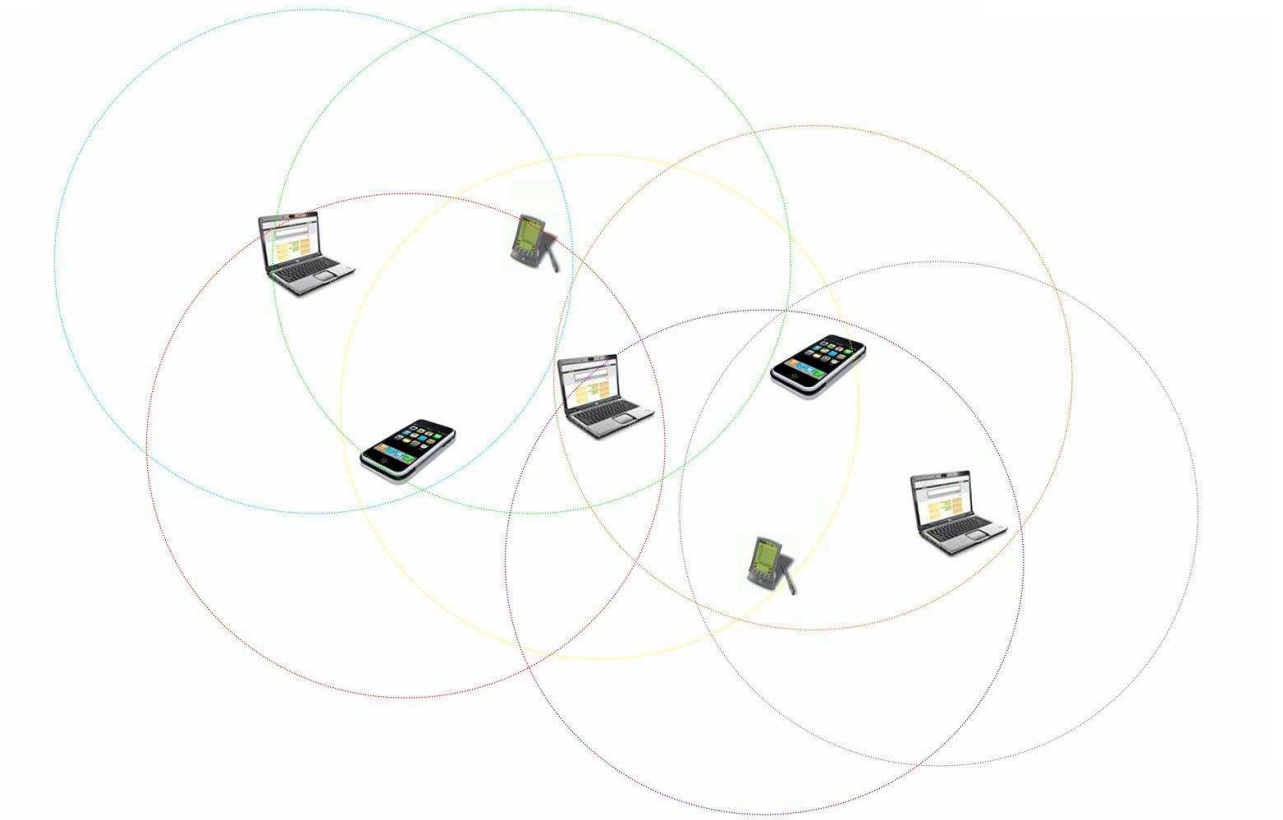

Figura 1-1 Ejemplo de una MANET

2.1. Principales características de las redes Ad Hoc

Las características principales de las MANET son las siguientes:

- Terminales autónomos: Cada Terminal se comporta como un nodo autónomo que puede funcionar como emisor, receptor o encaminador.
- Conexiones inalámbricas: No existe ningún tipo de infraestructura fija, los terminales usan el aire como canal de comunicación.
- Funcionamiento distribuido: No existe ningún elemento central que se encargue de la gestión y el control de la red, todos los nodos son iguales y por lo tanto la gestión está distribuida.
- Topología dinámica: Como no existe ninguna infraestructura fija y además los nodos son móviles, la topología de la red puede ser altamente cambiante. Las redes MANET deben adaptarse rápidamente a los cambios de tráfico generado por los nodos, a los distintos patrones de movimientos y a las condiciones de propagación.
- Capacidad variable de los enlaces: Al tratarse de un medio de transmisión compartido el canal de transmisión cambia constantemente los niveles de ruido, atenuación e interferencias. Además, en una transmisión extremo a extremo puede participar varios

enlaces distintos y la ruta puede cambiar varias veces en una misma transmisión.

- Consumo de energía: Los nodos son móviles y por lo tanto es de suponer que funcionan con baterías de vida limitada, por esa razón es muy importante que el consumo de energía se reduzca lo máximo posible.

2.2. Aplicaciones de las redes MANET

Al tratarse de redes que no requieren de una infraestructura fija, las redes MANET son fácilmente desplegables. Es por eso que son muy útiles en entornos dónde resulte muy costoso instalar una infraestructura fija, dónde las características físicas del entorno no lo permitan o en dónde se requiera de un despliegue rápido. Algunas de las aplicaciones que pueden tener las redes MANET son las siguientes:

- Entornos militares: En muchos avances tecnológicos las aplicaciones militares son de las primeras en aparecer y las redes MANET no son una excepción. En entornos militares las redes MANET permiten establecer comunicación entre distintas unidades, vehículos o centros de mando sin necesidad de establecer una infraestructura fija, lo cual puede ser muy difícil o imposible.
- Situaciones de emergencia: Ya sea por desastres naturales o por otras razones, cuando los equipos de emergencia, rescate o salvamento tienen que actuar rápidamente no existe la posibilidad de instalar una infraestructura fija si es que no existe ninguna previamente o la que existe ha quedado inutilizada. Desplegar una red MANET es una solución rápida y eficaz para muchas situaciones de emergencia.
- Entornos civiles: Las posibles aplicaciones en entornos civiles son muy amplias. Se pueden crear redes de sensores por ejemplo en entornos agrícolas, más económico que instalar una infraestructura. Pero también se pueden crear redes MANET para compartir información entre los participantes en un congreso, una conferencia, una clase, etc. Otros ejemplos dónde existen muchas aplicaciones son en estadios deportivos, en aeropuertos, cafeterías, museos, centros comerciales, etc.
- Redes de área personal: Conocidas como PAN (*Personal Area Network*) se tratan de redes formadas por dispositivos de uso personal como un ordenador portátil, un teléfono móvil, una PDA, etc. Usar una red MANET nos puede permitir comunicar estos dispositivos entre ellos fácilmente.

2.3. Estado del arte

Las redes inalámbricas así como los dispositivos móviles se han hecho muy populares, eso es porque facilitan el acceso a la información en cualquier momento y lugar. En general, las redes inalámbricas convencionales necesitan una infraestructura fija y una gestión centralizada. En cambio las MANETs no precisan de esta infraestructura fija ni de elementos centralizados de gestión.

Como punto de inicio de la investigación de las redes Ad Hoc se puede establecer el año 1995, concretamente en una conferencia de la IETF (*Internet Engineering Task Force*). Estas primeras discusiones se centraron en redes inalámbricas satelitales, redes gubernamentales y redes autoconfigurables. La idea principal fue la de adaptar los protocolos existentes en Internet a las redes inalámbricas y altamente dinámicas. Posteriormente, en el año 1997, se creo el grupo de trabajo sobre MANETs de la IETF: el MANET-WG (*MANET Work Group*). Las principales tareas de este grupo de trabajo consisten en especificar interfaces y protocolos que soporten Internet basado en IP sobres las MANET. La mayor parte de las investigaciones y avances que se han realizado sobre estos temas lo ha hecho este grupo de trabajo, que además actúa como principal organismo de normalización. Además del MANET-WG se han creado otros grupos de investigación, como por ejemplo el creado en Japón en 2003 denominado *Ad Hoc Consortium*. El objetivo de este grupo de investigación es el de unir intereses y esfuerzos de la industria, el mundo académico y el gobierno para usar la tecnología de las redes Ad Hoc en todo tipo de aplicaciones.

En la actualidad el área de provisión de QoS (*Quality of Service*) en Internet tiene una gran importancia y crecimiento, es por ello que hay mucho trabajo realizado en esta área. Pero estos mecanismos desarrollados para Internet no son aplicables directamente a las MANETs debido a las diferencias considerables entre ambas tecnologías. Por lo tanto uno de los desafíos más importantes actualmente en las MANETs es el de dotar de QoS a unas redes que son altamente dinámicas, no disponen de infraestructura fija y tienen unos recursos muy limitados.

A continuación veremos más detalladamente el funcionamiento de las redes MANET y los principales protocolos que se usan en los distintos niveles de estas redes.

2.3.1. Capa Física

En las redes MANET los nodos se comunican entre ellos usando como medio de transmisión el espacio libre. Lo pueden hacer a través de canales de radiofrecuencia (RF) o a través de rayos infrarrojos (IR). Aunque realmente las redes basadas en rayos infrarrojos se usan muy poco y sólo en aplicaciones muy concretas. Esto se debe a que los rayos infrarrojos tienen muchas

desventajas respecto a los canales de radiofrecuencia, las principales serían que no pueden atravesar paredes ni otros obstáculos y que requieren más potencia.

Existen estándares del IEEE 802.11X que definen interfaces para los canales de RF en las bandas de 2'4GHz y de 5GHz. La banda de 2'4GHz está más saturada ya que se utiliza para otro tipo de redes como Bluetooth y para dispositivos como por ejemplo hornos microondas y juguetes de radio control. Además en ésta banda de frecuencias se permiten usos abiertos y experimentales, de manera que no hace falta tener licencias especiales para trabajar en ella, siempre que no se sobrepasen los niveles de potencia permitidos. Por lo tanto, las redes de RF que funcionan es esta banda de frecuencias sufren muchas más interferencias que las que funcionan en la banda de 5GHz. Pero la ventaja que tiene la banda de 2'4GHz respecto a la banda de 5GHz es que las señales sufren muchas menos perdidas de transmisión en el espacio libre.

En los canales de RF las señales se comportan de manera dispersa, eso permite que se usen antenas omnidireccionales, las cuales son muy económicas y permiten que los dispositivos móviles se comuniquen con cualquier otro nodo que se encuentre lo suficientemente cerca en cualquier dirección. Esto es una gran ventaja pero implica otro inconveniente y es que dos nodos distintos pueden estar transmitiendo al mismo tiempo usando el mismo canal de RF, por eso hace falta que el protocolo MAC (*Medium Access Control*) se encargue de evitarlo.

Al no poder transmitir libremente, aunque las velocidades de transmisión teóricas pueden alcanzar hasta los 54Mbps en los estándares IEEE 802.11a y 802.11g (incluso más en los nuevos estándares que están siendo revisados por el comité IEEE 802.11), éstas no se reflejan en la velocidad efectiva de transmisión vista desde las capas superiores de la red. Además, los mecanismos usados por la capa MAC para controlar el acceso a los canales de RF compartidos y maximizar su utilización, evitando colisiones, impiden que se logre una utilización del 100% de los recursos de la red. Existen posibles soluciones a este problema como se puede ver en [1], pero ésta pasa por usar antenas direccionales lo cual no permite que los nodos sean móviles ya que al moverse la antena dejaría de apuntar al siguiente nodo.

Como conclusión podemos decir que el problema principal de la capa física de las redes MANET se debe al medio de transmisión utilizado, los canales de RF. Éstos son limitados y deben ser compartidos por varios nodos, eso conlleva que se reduce la capacidad. Además, las señales de RF al ser transmitidas por el espacio libre sufren atenuaciones debidas a distintos fenómenos como la absorción, la distorsión, el multitrayecto, etc. Por lo tanto, nos encontramos frente a un canal de transmisión dinámico y poco fiable.

2.3.2. Capa MAC

Existen dos familias principales de protocolos de control de acceso al medio (MAC): protocolos de acceso aleatorio y protocolos de acceso controlado. En los protocolos de acceso aleatorio los nodos compiten entre ellos para conseguir el canal de transmisión, mientras que en los de acceso controlado existe un dispositivo que decide cuál de los nodos puede acceder al medio en cada momento.

Hay muchos protocolos MAC de acceso controlado entre los cuales destacan los siguientes: FDMA (*Frequency Division Multiple Access*), TDMA (*Time Division Multiple Access*), CDMA (*Code Division Multiple Access*). Estos protocolos se aplican principalmente en las redes que cuentan con una infraestructura y con un nodo maestro que se encarga de administrar el acceso al medio de transmisión por parte de los nodos de la red. En estos protocolos no existen colisiones. Pero como ya hemos explicado anteriormente las redes MANET no cuentan con una infraestructura fija y todos los nodos son iguales por la cual cosa, en general, no se pueden usar estos protocolos.

Por lo tanto, las redes MANET usan protocolos MAC de acceso aleatorio. Concretamente, el protocolo estandarizado por el comité IEEE 802.11 fue el CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) debido a su inherente flexibilidad y porque resuelve el problema de los terminales ocultos a través del sencillo mecanismo de paquetes de control RTS/CTS-DATA/ACK [2,3]. El estándar IEEE 802.11 define el uso de los dos niveles más bajos de la arquitectura OSI (*Open Systems Interconnection*), que son la capa física y la capa de enlace, y se ha convertido en uno de los estándares más populares.

Para mejorar el comportamiento de los protocolos MAC para las redes MANET se incluyen varias propuestas: algoritmos para reducir el consumo de energía de los nodos móviles que permiten que los nodos entren en un estado de bajo consumo de energía cuando éstos están ociosos; algoritmos de gestión de paquetes; reducción de los radios de transmisión y de recepción; diferentes esquemas de codificación y modulación de la señal, etc.

2.3.3. Capa de red

Es en esta capa dónde más se distinguen las redes MANET de las redes más conocidas y por lo tanto el estudio esta capa es el más destacable. A este nivel los protocolos de encaminamiento para las redes MANET es necesario que se adapten rápidamente a los cambios constantes de la topología de la red para poder mantener una ruta para la comunicación entre los nodos. Existen una gran cantidad de protocolos de encaminamiento para las redes Ad Hoc, los cuales pueden ser más o menos adecuados en cada escenario en concreto. Estos protocolos se pueden dividir en tres grandes grupos [4, 5] en función del método que utilizan para determinar las rutas:

- Protocolos reactivos
- Protocolos proactivos
- Protocolos híbridos

Además de estos tres grandes grupos también existe un pequeño grupo que además de realizar la función de encaminamiento, intenta dotar de los mecanismos necesarios para garantizar QoS en las comunicaciones.

En la figura siguiente se pueden ver distintos protocolos de encaminamientos ya clasificados según las categorías anteriormente mencionadas:

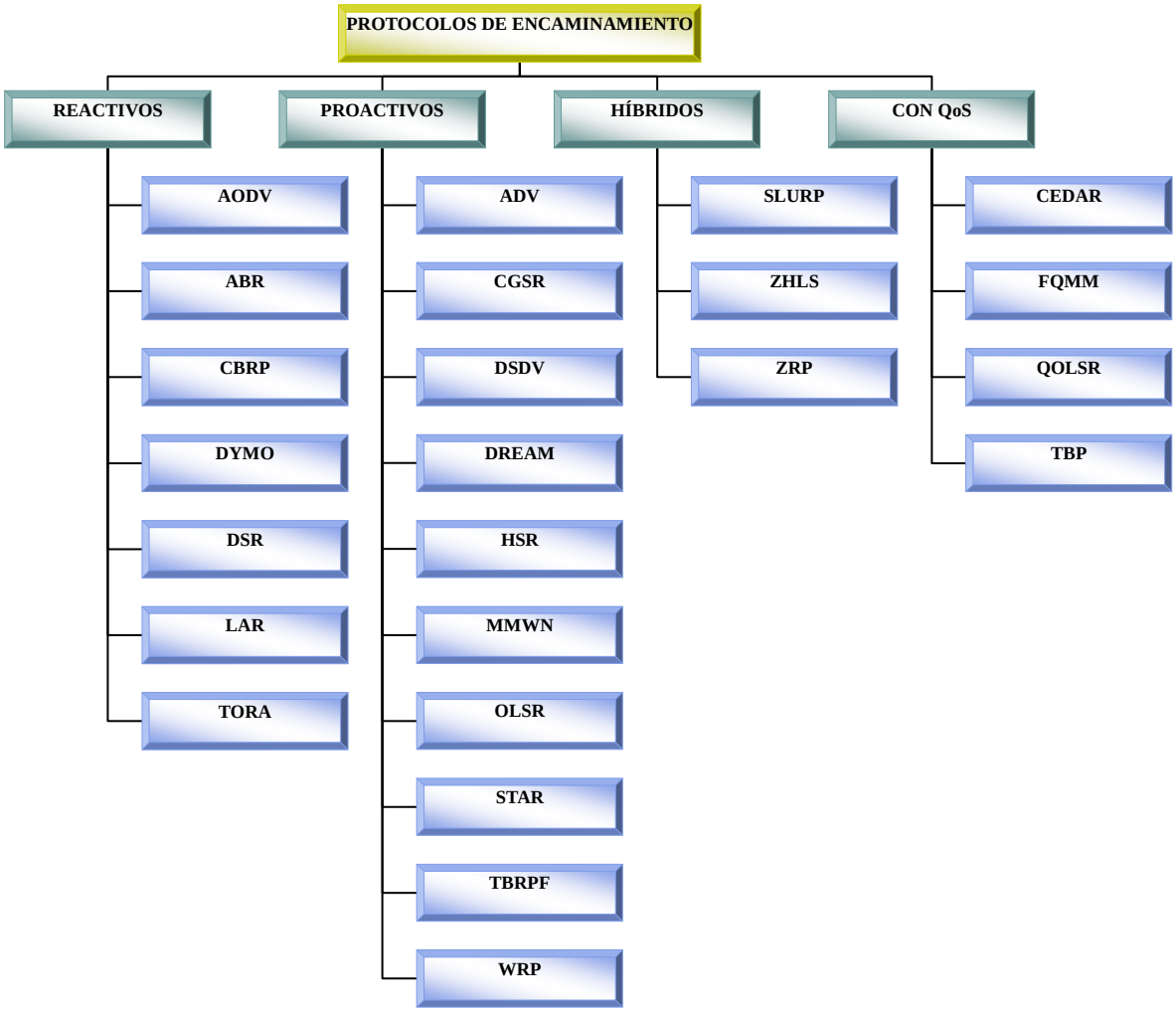

Figura 1-2 Clasificación de los protocolos de encaminamiento de las redes Ad Hoc

2.3.3.1. Protocolos reactivos

Los protocolos de encaminamiento reactivo o bajo demanda son aquellos en los cuales se hallan las rutas entre un nodo origen y un nodo destino bajo demanda de la fuente. Es decir, que sólo cuando sea necesario iniciar una transmisión se buscará una ruta para realizarla. Una vez establecida la ruta, los nodos que participen en la transmisión se encargarán de su mantenimiento. Las ventajas de este tipo de protocolos es que no necesitan demasiada señalización, lo cual reduce el *overhead* y optimiza el uso de las baterías, al contrario de lo que sucede con los protocolos proactivos. Sin embargo, el tiempo de establecimiento de las rutas es mayor, ya que cuando se necesita la ruta se inicia el mecanismo de descubrimiento de ruta y hasta que éste no termina no se puede iniciar la transmisión.

Hay una gran cantidad de protocolos de encaminamiento reactivos, las diferencias entre ellos se encuentran en la implementación del mecanismo de descubrimiento de ruta y en las optimizaciones del mismo. Los protocolos reactivos más importantes son los siguientes:

- *Ad Hoc On Demand Distance Vector Routing* (AODV), 2003
- *Associativity-Based Routing* (ABR), 1996
- *Cluster-Based Routing Protocol* (CBRP), 1999
- *Dynamic MANET On demand* (DYMO), 2005
- *Dynamic Source Routing* (DSR), 2004
- *Location-Aided Routing* (LAR), 1998
- *Temporally Ordered Routing Algorithm* (TORA), 2001

De ellos los protocolos AODV y DSR han sido presentados como RFC (*Request for Coments*) experimentales, mientras que del resto se han presentado como *Draft* por el IETF (*Internet Engineering Task Force*) los protocolos ABR, CBRP, DYMO y TORA. Aunque el DYMO está siendo considerado por el MANET-WG para convertirse en RFC.

2.3.3.2. Protocolos proactivos

Los protocolos de encaminamiento proactivos intentan mantener tablas de las rutas actualizadas constantemente. Eso significa que cada nodo debe mantener actualizada una tabla con todas las rutas hacia los otros nodos. La información que contienen las tablas debe actualizarse periódicamente y ante cualquier cambio de la tipología de red. Esta actualización constante provoca que estos protocolos generen una gran cantidad de paquetes de señalización (*overhead*) lo cual afecta a la utilización del ancho de banda, el *throughput* y el consumo de energía entre otras cosas. La ventaja principal que aportan estos protocolos es que el establecimiento de una

nueva ruta para iniciar una transmisión precisa de un tiempo muy pequeño al tener todos los nodos las tablas de rutas actualizadas. Además, el *overhead* aunque sea elevado también es bastante estable en el tiempo. El principal problema con el que se encuentran estos protocolos es cuando las redes son muy densas y los nodos tienen una movilidad alta ya que en estos casos el *overhead* crece muy rápidamente y las tablas pueden llegar a ser demasiado grandes.

Los distintos protocolos de encaminamiento proactivos de distinguen entre ellos por el número de tablas, la información que contienen las tablas y en como se actualizan éstas. Los protocolos proactivos más destacables son los siguientes:

- *Adaptative Distance Vector* (ADV), 2000
- *Cluster Gateway Switch Routing* (CGSR), 1999
- *Destination-Sequenced Distance Vector* (DSDV), 1994
- *Distance Routing Effect Algorithm for Mobility* (DREAM), 1998
- *Hierarchical State Routing* (HSR), 2000
- *Multimedia support in Mobile Wireless Networks* (MMWN), 1998
- *Optimised Link State Routing* (OLSR), 2003
- *Source-Tree Adaptaive Routing* (STAR), 1999
- *Topology Dissemination Based on Reverse-Path Forwarding* (TBRPF), 2004
- *Wireless Routing Protocol* (WRP), 1996

De los cuales, los protocolos OLSR y TBRPF han sido presentados como RFC experimentales, mientras que del resto se han presentado como *Draft* por el IETF los protocolos ADV, CGSR, HSR y STAR.

2.3.3.3. Protocolos híbridos

Este grupo de protocolos híbridos se basan en combinar las características de los dos grupos anteriores para intentar aprovechar las ventajas de ambos. En general su funcionamiento se basa en agrupar los nodos en grupos o zonas, de esta manera cuando necesitan descubrir rutas hacia otro nodo de su zona utilizan un encaminamiento proactivo y para descubrir rutas en nodos lejanos utilizan un encaminamiento reactivo.

Dentro de este grupo de protocolos existen algunos ejemplos como los siguientes:

- Scalable Location Update Routing Protocol (SLURP), 2004
- Zone-based Hierarchical Link State (ZHLS), 1999

- Zone Routing Protocol (ZRP), 2002

De estos tres protocolos sólo se ha presentado como *Draft* por el IETF el protocolo ZRP.

2.3.3.4. Protocolos con QoS

Este grupo de protocolos se basa en buscar rutas con los recursos suficientes para satisfacer los requerimientos de QoS establecidos. Su función no es otra que la de intentar garantizar la QoS de una comunicación dentro de una red MANET. Para ello deben intentar que la comunicación extremo a extremo cumpla una serie de restricciones como el retardo, la variación del retardo, el ancho de banda mínimo necesario o una combinación de estos parámetros. Esta misión es muy difícil de conseguir en las redes MANET debido al *overhead* introducido, así que tienen que intentar encontrar un compromiso entre el *overhead* introducido y los beneficios obtenidos. También deben combatir contra la dificultad de mantener información precisa sobre el estado de los enlaces debido a la naturaleza móvil de los nodos. Además, durante la comunicación la QoS tampoco se puede garantizar que se mantenga estable sino más bien todo lo contrario, así que los protocolos deben buscar rápidamente una nueva ruta en caso de romperse un enlace.

Existen algunos protocolos dentro de este grupo, algunos de los más conocidos son los siguientes:

- *Core-Extraction Distributed Ad hoc Routing* (CEDAR), 1999
- *Flexible QoS Model for MANET* (FQMM), 2000
- *QoS Optimized Link State Rounting* (QOLSR), 2006
- *Ticket-based Probing* (TBP), 1999

De ellos sólo el QOLSR ha sido presentado como *Draft* por el IETF.

2.3.3.5. DSR (Dynamic Source Routing)

De todos los protocolos de encaminamiento vistos, el DSR es el que se ha escogido para usar en todas las simulaciones de este Proyecto Final de Carrera. Esta elección se basa fundamentalmente en el hecho de hacerlo compatible con los trabajos realizados por el Grupo de Servicios Telemáticos que han usado este protocolo. Esto es debido a que el DSR es uno de los protocolos más conocidos, además, debido a sus características es el que mejor se adapta a los escenarios que se estudian es este Proyecto Final de Carrera y uno de los cuales están implementados en el simulador NS-2. Al tratarse de una parte importante de este proyecto, este

protocolo se explica detalladamente en un capítulo a parte, concretamente en el siguiente capítulo.

2.3.4. Capa de transporte

En las redes MANET se busca que sean al máximo compatibles con los protocolos utilizados por Internet, que son los más populares. Esto es porque se ve como objetivo final de las redes MANET el poder ofrecer Internet en todo lugar y momento, por lo tanto, las redes Ad Hoc están basadas, principalmente, en la pila de protocolos TCP/IP (*Transmission Control Protocol/Internet Protocol*). Es por esta razón que prácticamente todos los esfuerzos de investigación del MANET-WG están dirigidos al desarrollo de algoritmos de encaminamiento específicos para este tipo de redes, intentando que el resto de protocolos de la pila no se vean afectados. A pesar de ello, si que se han presentado algunas propuestas con el fin de mejorar las prestaciones de los principales protocolos de transporte de Internet, TCP y UDP (*User Datagram Protocol*), en entornos Ad Hoc.

Por ejemplo, podemos ver en [6] una mejora de TCP para redes Ad Hoc. Ésta utiliza mecanismos de señalización implícita que permiten tener un TCP que no active, de manera equivocada, el mecanismo de control de congestión cuando se pierde una ruta entre el nodo fuente y el nodo destino debido a un fallo en un enlace. También se han hecho propuestas de protocolos de transporte específicos para redes Ad Hoc, un ejemplo es el TPA (*Transport Protocol for Ad Hoc Network*) [7]. El protocolo TPA incluye mecanismos que detectan cuando se pierde un enlace y se recupera la ruta, además establece mecanismos de control de congestión distintos a los de TCP. Por otro lado, se han desarrollado otros protocolos de transporte para otras aplicaciones de tiempo real y *media-streaming* como pueden ser el SCTP (*Stream Control Transport Protocol*) y el MRTP (*Multiflow Realtime Transport Protocol*). Estos protocolos tienen como característica principal, el hecho de que establecen múltiples flujos de datos entre dos nodos para una misma comunicación y funcionan junto a protocolos de encaminamiento multicamino.

2.3.5. Capa de Aplicación

Dentro del MANET-WG uno de los objetivos principales es que las aplicaciones diseñadas para Internet se pueden soportar también en las redes MANET. En general, las aplicaciones de transmisión de datos, que no precisen de grandes anchos de banda ni altas limitaciones en cuanto al tiempo de entrega de los paquetes, se pueden soportar con la filosofía *Best Effort*. Pero hoy en día en Internet existe una gran cantidad de aplicaciones, algunas de las cuales son multimedia. Estas aplicaciones son mucho más sensibles al retardo y a la variación de este retardo, así que

para funcionar correctamente necesitan mucho más ancho de banda y otras garantías. La parte positiva es que ofrecen una mayor flexibilidad por lo que a pérdidas de paquetes se refiere.

Por todas las limitaciones y características intrínsecas de las redes MANET aún quedan muchos retos por superar antes que éstas puedan ofrecer aplicaciones multimedia. Para conseguirlo, es necesario que todos los niveles de red trabajen en el mismo sentido para lograr el objetivo de poder garantizar una cierta QoS.

Existen algunas propuestas a nivel de aplicación para adaptarse a las redes MANET. Estas propuestas pasan por intentar disminuir el ancho de banda, básicamente con nuevas técnicas de codificación que sacrifiquen un poco la calidad para aumentar la compresión. Aunque también hay propuestas encaminadas a modificar la calidad de la información en función del estado en el que se encuentre la red.

Capítulo 3. DSR (*Dynamic Source Routing*)

El DSR [8] pertenece al grupo de protocolos de encaminamiento reactivos, es decir, busca una ruta cuando ésta es requerida. Es un sencillo y eficiente protocolo de encaminamiento diseñado específicamente para usarlo en MANETs multisalto. Su uso hace que la red sea completamente autoorganizable y autoconfigurable, sin necesidad de añadirle infraestructura ni administración centralizada. Está diseñado para funcionar bien con topologías complejas y altamente cambiantes, DSR puede funcionar adecuadamente hasta con una red de 200 nodos, que es más que otros protocolos. A partir de 200 nodos el protocolo empieza a no funcionar correctamente debido a que las cabeceras de los paquetes crecen demasiado, ya que a más nodos mayor tamaño de cabecera.

El DSR está compuesto de dos mecanismos principales que trabajan juntos para permitir descubrir y mantener las rutas desde la fuente hacia el destino. Estos mecanismos son: *Route Discovery* y *Route Maintenance*. Los nodos intermedios cooperan retransmitiendo los paquetes necesarios cuando no hay una comunicación directa entre fuente y destino. Un nodo fuente usa el proceso *Route Discovery* cuando necesita transmitir paquetes hacia un nodo destino y no conoce ninguna ruta para hacerlo. En cambio, el proceso *Route Maintenance* lo usa para verificar si alguna de las rutas que ya conoce sigue siendo válida.

A continuación veremos más detalladamente el funcionamiento de estos dos mecanismos:

3.1. Route Discovery

Cuando un nodo fuente tiene un paquete para enviar a un destino primero busca en su memoria caché si tiene una ruta para ese destino, si es así, crea un nuevo paquete añadiendo a la cabecera la ruta en donde se indican los saltos que debe seguir para llegar al destino y lo envía. Normalmente, la fuente obtiene la ruta buscando en su memoria caché de rutas donde están las rutas que ha descubierto previamente. Si no encuentra ninguna ruta en su caché es cuando debe iniciar el mecanismo *Route Discovery* para encontrar dinámicamente una nueva ruta hacia el nodo destino.

Lo primero que hace el nodo fuente es enviar un paquete de *broadcast* denominado *Route Request* (RREQ), el cual es recibido por todos los nodos que se encuentren dentro del rango de transmisión del nodo fuente. El paquete RREQ contiene una identificación del nodo fuente y el nodo destino de la ruta a descubrir, y también contiene un identificador único para cada RREQ, este identificador lo determina la fuente. Cada nodo intermedio que reenvía el RREQ hace una copia de éste y le añade también su identificador.

Cuando un nodo recibe un RREQ y comprueba que no es el destino de la ruta, mira si ya ha recibido recientemente otro RREQ con la misma fuente, destino e identificación o si su propia dirección ya aparece en el paquete, en cualquiera de los dos casos el nodo descarta el paquete. Si no se cumple ninguna de las dos condiciones anteriores, el nodo primero mira en su caché si tiene una ruta hacia el nodo destino, y si la tiene, el nodo responde con un *Route Reply* (RREP) hacia el origen en lugar de reenviar el RREQ. En ese RREP escribe los nodos por los que ha pasado el RREQ recibido más los nodos que ha encontrado en su caché. Si el nodo no encuentra ninguna ruta en su caché, añade su dirección en el paquete y lo reenvía mediante *broadcast*.

Si el nodo destino recibe un RREQ, contesta con un *Route Reply* (RREP) al nodo fuente. Este RREP contiene una copia de las direcciones acumuladas en el RREQ recibido. Si los enlaces son bidireccionales, el nodo envía el RREP invirtiendo la ruta por la que ha recibido el RREQ. En caso contrario (que no es el de este proyecto ya que el estándar 802.11 implica canales bidireccionales puesto que se basa en ACK para funcionar) el nodo fuente debe iniciar un *Route Ddiscovery* para hallar una ruta para enviar el RREP. Para evitar entrar en un bucle infinito este RREQ que se inicia, debe contener el RREP que el nodo quiere enviar. Este procedimiento se conoce como *piggybacking* del *Route Reply*.

Una vez el nodo origen recibe el RREP, éste guarda dicha ruta en su caché. Esta ruta será incluida en la cabecera de cada paquete que el nodo envíe, de esta manera todos los nodos que reciban el paquete sabrán hacia que nodo deben reenviarlo.

3.2. Route Maintenance

Cuando se envía un paquete a través de una ruta, cada nodo es responsable de confirmar que el paquete se ha recibido en el siguiente nodo. Este reconocimiento suele ser a nivel de enlace. Después de haber retransmitido un paquete un número máximo de veces, si el paquete no se ha podido enviar, entonces el nodo considera que el enlace hacia el siguiente nodo está roto. En ese caso el nodo debe eliminar ese enlace de su caché y enviar un *Route Error* (RERR) a cada nodo que haya enviado un paquete con una ruta que usa el enlace roto.

Si el nodo origen de la transmisión recibe un RERR debe eliminar esa ruta de su caché. Entonces para seguir transmitiendo primero debe mirar si en su caché tiene otra ruta hacia el nodo destino (por ejemplo, si ha recibido varios RREP de algún *Route Discovery* anterior). En el caso que no tenga ninguna otra ruta en su caché, el nodo origen tiene que iniciar un nuevo *Route Discovery* para encontrar una nueva ruta hacia el nodo destino. Este nuevo *Route Discovery* se envía junto con el mensaje de error recibido (*piggybacking* del mensaje de error) para informar a los demás nodos del enlace roto y así evitar que respondan con un RREP usando la ruta antigua que ya no es válida.

3.3. Características adicionales

Un nodo intermedio u otro nodo que recibe cualquier paquete puede añadir toda la información de rutas que recibe en su propia caché. Por ejemplo, si tenemos una transmisión entre el nodo A y el nodo E que pasa por los nodos B, C y D, cuando el nodo A inicia el *Route Discovery*, los nodos intermedios aprovechan los paquetes recibidos para actualizar sus cachés. Así el nodo C conocerá la ruta para llegar al nodo E y el nodo B para llegar al nodo D, etc. Esto ayuda a mantener las rutas de las cachés actualizadas, lo cual es importante para agilizar las comunicaciones y colaborar en lo posible contestando a los RREQ de otros nodos.

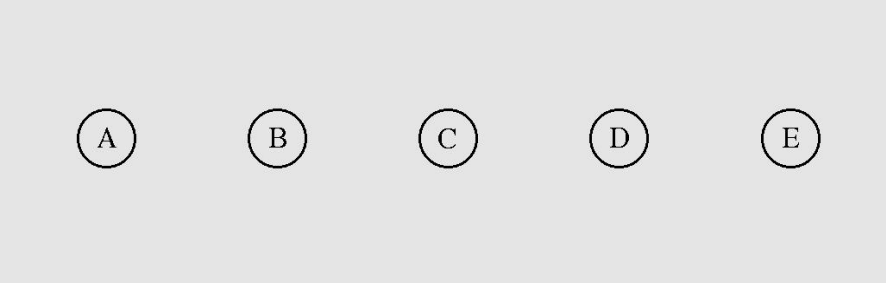

Figura 3-3 Ejemplo de MANET para explicar características del DSR

Cada paquete RREQ contiene un límite de saltos que se usa como el número máximo de nodos intermedios que pueden reenviar una copia de dicho RREQ. Se limita el número de saltos para

evitar que los RREQ se propaguen demasiado lejos y que llegue una gran cantidad de RREP innecesarios, lo cual implicaría aumentar la congestión de la red. Este límite de saltos se implementa usando el campo *Time-to-Live* (TTL) en la cabecera IP del paquete que contiene el RREQ. Cada vez que un nodo reenvía el paquete el TTL se decrementa y si llega a cero el paquete se descarta. Este límite puede usarse para implementar varios algoritmos para controlar el alcance de un RREQ. Lo más normal es establecer un límite de saltos para cada paquete inicialmente bajo. Si no es posible encontrar una ruta, el límite de saltos se va aumentando hasta que se encuentra alguna ruta. Por ejemplo, un nodo podría usar este límite para enviar un RREQ que no se propague. Un nodo que usa esta técnica envía su primer intento de RREQ con un límite de saltos de 1, de manera que cualquier nodo que recibe este RREQ no puede propagarlo. Esto permite determinar si el nodo destino es un vecino del nodo origen o si un vecino del nodo origen tiene una ruta hacia el destino en su caché. Si después de un pequeño lapso de tiempo no ha recibido ningún RREP, entonces el nodo envía un nuevo RREQ con un límite de saltos definido por una variable.

Para evitar la colisión cuando varios nodos envían sus RREP al nodo origen, cada nodo espera un tiempo de *back-off* aleatorio antes de enviar un paquete.

Existe un mecanismo denominado *Packet Salvaging* que se usa para evitar que se pierdan paquetes. Concretamente, cuando un nodo intermedio debe reenviar un paquete y detecta, gracias al *Route Maintenance*, que el siguiente enlace establecido en la ruta está roto. En dicho caso el nodo mira en su memoria caché si tiene otra ruta para ir al siguiente nodo, y si la tiene, en lugar de descartar el paquete, lo “salva” enviándolo por esa nueva ruta.

Una ruta en uso puede ser automáticamente acortada si uno o más nodos intermedios en la ruta pasan a ser innecesarios. Este mecanismo, denominado *Automatic Route Shortening*, funciona de la siguiente manera: si un nodo intermedio recibe paquetes dirigidos a otro nodo intermedio y a la vez éste se encuentra en la ruta en un salto posterior, este nodo puede enviar un RREP gratuito para avisar que hay una nueva ruta más corta disponible. De todas formas el nodo intermedio que envía el RREP no lo hace hasta que recibe el paquete con una relación señal a ruido suficientemente buena, por encima de un umbral predeterminado.

3.4. Extensión *flowstate*

Existe una extensión del protocolo DSR denominada *Flowstate Extension*, la cual es importante conocer ya que el NS-2 usa el DSR con esta extensión incorporada.

La extensión *flowstate* permite enviar la mayoría de los paquetes sin necesidad de incluir en la cabecera la ruta explicita que debe seguir el paquete. Por esta razón, esta extensión permite reducir el *overhead* del protocolo DSR pero sin cambiar las propiedades fundamentales de su funcionamiento.

El *flowstate* lo inicializa automáticamente el primer paquete de un flujo de datos. De esta manera, cada flujo de datos se identifica con tres parámetros: la dirección del nodo origen, la dirección del nodo destino y un identificador de flujo denominado *flow ID*. Este identificador lo escoge el nodo origen y es un número de 16 bits.

Los paquetes enviados deben contener una cabecera DSR con la información completa de la ruta a seguir o con el *flow ID* del flujo al que pertenece dicho paquete. Si un nodo recibe un paquete en cuya cabecera no aparece la ruta completa ni el *flow ID* del flujo al que pertenece, el nodo debe establecer una ruta por defecto por donde encaminará los paquetes con el origen y el destino indicados en ese paquete.

Usar la extensión *flowstate* permite reducir el tamaño de las cabeceras y eso conlleva poder utilizar el protocolo DSR en redes con un número de nodos mayor y con más saltos a realizar entre los nodos origen y destino.

3.5. Ventajas e inconvenientes de DSR

Una vez visto el funcionamiento de DSR podemos ver que ventajas e inconvenientes presenta respecto a otros protocolos:

Ventajas:

- Simple y eficiente.
- Pocos paquetes de control.
- No requiere *broadcasts* periódicos de información de encaminamiento.
- *Overhead* reducido, se gestionan sólo las rutas entre nodos que desean comunicarse.

Inconvenientes:

- Alta latencia en la búsqueda de nuevas rutas.
- Poco escalable, la cabecera aumenta de manera directamente proporcional con el número de saltos.
- Alta probabilidad de colisión de los RREQ.

- Un nodo intermedio puede corromper la información de encaminamiento si envía un RREP usando una ruta de su caché obsoleta.

Podemos concluir entonces que el DSR funcionará bien en redes no muy cambiantes, lentas o no muy dinámicas y en redes no muy grandes. No funcionará bien en redes muy cambiantes ya que al tener cada paquete la ruta a seguir desde que es enviado por el nodo origen, no es capaz de adaptarse a los cambios de la red. Por otro lado, si la red es muy grande y el número de nodos que tiene que atravesar cada paquete para llegar a su destino es muy grande, las cabeceras de los paquetes aumentan demasiado y se reduce la eficiencia del protocolo.

3.6. Ejemplo de DSR

Vamos a ver ahora en un ejemplo como funciona el mecanismo *Route Discovery* del DSR. En la Figura 1 vemos que el nodo fuente (número 1) quiere iniciar una transmisión hacia el nodo destino (número 0).

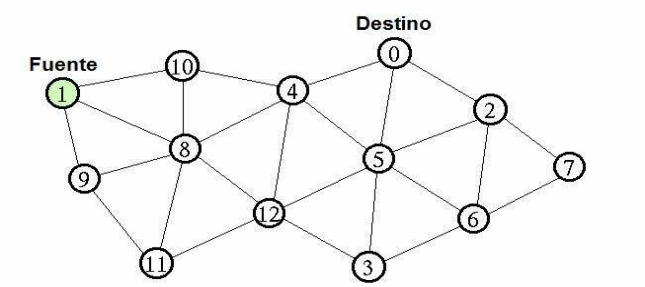

Figura 3-4 Ejemplo de funcionamiento del DSR I

Si el nodo 1 tuviese una ruta en su caché hacia el nodo 0, usaría dicha ruta, en caso contrario, inicia un *Route Discovery* enviando un RREQ a sus nodos vecinos como vemos en la Figura 2.

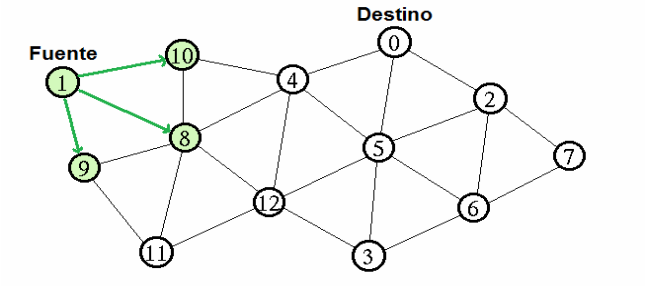

Figura 3-5 Ejemplo de funcionamiento del DSR II

Reciben el RREQ los nodos 8, 9 y 10, estos deben en primer lugar mirar si tienen almacenada en su caché una ruta hacia el nodo 0 que es el destino. Si tuviesen dicha ruta en su caché enviarían al nodo 1 un RREP con la ruta. Al no tenerla, añaden su dirección al paquete RREQ recibido y lo reenvían a sus nodos vecinos como vemos en la Figura 3.

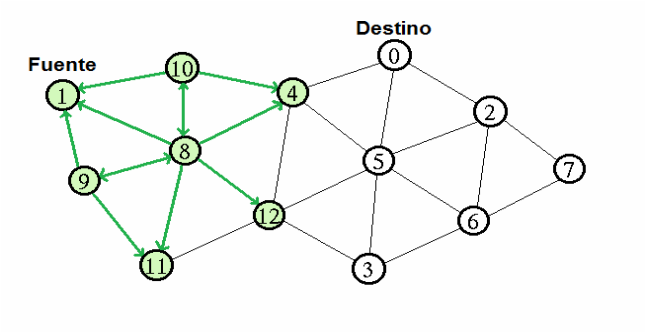

Figura 3-6 Ejemplo de funcionamiento del DSR III

Ahora reciben RREQ los nodos 4, 11 y 12, además del 1, 8, 9 y 10. El nodo 1 es el que ha iniciado el *Route Discovery* por lo tanto descarta el RREQ que le reenvían. También lo descartan los nodos 8, 9 y 10 ya que previamente ya han recibido ese mismo RREQ. Los nodos 4, 11 y 12 miran si tienen la ruta en su caché y si no la tienen reenvían el RREQ, añadiendo su dirección, a sus nodos vecinos como vemos en la Figura 4.

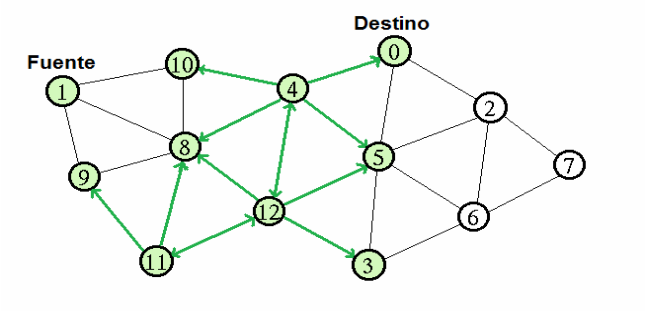

Figura 3-7 Ejemplo de funcionamiento del DSR IV

El RREQ llega finalmente al nodo destino. Al tratarse de enlaces bidireccionales, el nodo destino envía un RREP hacia la fuente usando la ruta por la que ha recibido el RREQ. En este caso la ruta podría ser tanto (1, 10, 4, 0) como (1, 8, 4, 0), la ruta válida sería la primera que hubiese llegado al nodo 4, ya que el segundo RREQ recibido por el nodo 4 se habría descartado. En el ejemplo se supone que la ruta es (1, 10, 4, 0) como vemos en la Figura 5.

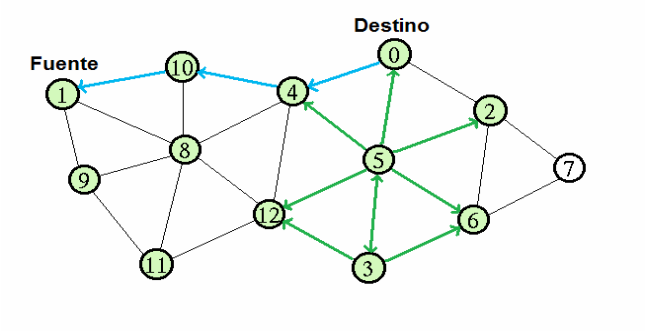

Figura 3-8 Ejemplo de funcionamiento del DSR V

Por último es importante destacar que los otros nodos siguen propagando RREQ y que de hecho vemos como el nodo destino recibe otros RREQ como, por ejemplo, el que le podría llegar por la ruta (1, 8, 12, 5, 0). El nodo destino también responderá con un RREP a este RREQ y a todos los que le lleguen. De esta manera el nodo fuente acabará recibiendo varias rutas hacia el nodo 0 y las almacenará en su caché. Éstas serán de utilidad por si en un futuro se rompe algún enlace de la primera ruta encontrada, ya que sin necesidad de iniciar otro *Route Discovery*, el nodo 1 podría usar otra de las rutas descubiertas.

Capítulo 4. Modelos Matemáticos

Antes de empezar con la explicación de los modelos matemáticos contenidos en este capítulo, cabe destacar que dichos modelos analíticos se empezaron a desarrollar durante la tesis doctoral de Víctor Carrascal [9] y han continuado su desarrollo durante la tesis doctoral de Aída Zavala Ayala [11], con quien he colaborado estrechamente en la realización de mi Proyecto Final de Carrera.

Uno de los principales objetivos de este proyecto final de carrera es validar, mediante simulaciones hechas en el NS-2, uno de los modelos matemáticos que se han desarrollado en el Grupo de Servicios Telemáticos de la Universidad Politécnica de Cataluña [9, 10, 11]. Este grupo de investigación sigue trabajando en el desarrollo de modelos matemáticos que caractericen a una red MANET. Los resultados obtenidos en este Proyecto Final de Carrera han sido útiles para validar uno de los modelos ya desarrollados, así como también, han sido útiles para el desarrollo y validación de nuevos modelos matemáticos.

El modelo matemático que se validará en este proyecto tiene como objetivos obtener el tiempo medio que una ruta está disponible en una MANET, el máximo número posible de saltos en la red y la probabilidad de que haya un número de saltos *i* en una comunicación.

A continuación mostramos el desarrollo que hizo el Grupo de Servicios Telemáticos para llegar al modelo matemático que se validará con las simulaciones realizadas en este proyecto. Las consideraciones que se hacen son las siguientes: los nodos de la red son homogéneos y los nodos se mueven siguiendo el modelo de movimiento *Random Waypoint* [14].

El desarrollo del modelo matemático se hizo en tres pasos. Primero se calculó la distancia media entre dos nodos de la red, después se calculó la probabilidad de error de un enlace y por último se calculó la probabilidad de error en una ruta completa.

4.1. Distancia media entre dos nodos

La función de distribución de probabilidad (pdf) de la distancia entre dos nodos, $f_d(d)$, que se encuentran dentro de un área de dimensiones *W X W* puede ser calculada usando la ecuación (1), cuyo comportamiento para una red de 1m^2 se muestra en la Figura 4-9 [12].

$$f_d(d) = \begin{cases} \frac{2d}{W^2}\left(\pi - \frac{4d}{W} + \frac{d^2}{W^2}\right) & 0 \le d \le W; \\ \frac{2d}{W^2}\left(2arcsen\frac{W}{d} - 2arccos\frac{W}{d} + \frac{4\sqrt{d^2-W^2}-W}{W} - \frac{W^2+d^2}{W^2}\right) & W < d \le \sqrt{2}W; \\ 0 & d > \sqrt{2}W. \end{cases} \quad (1)$$

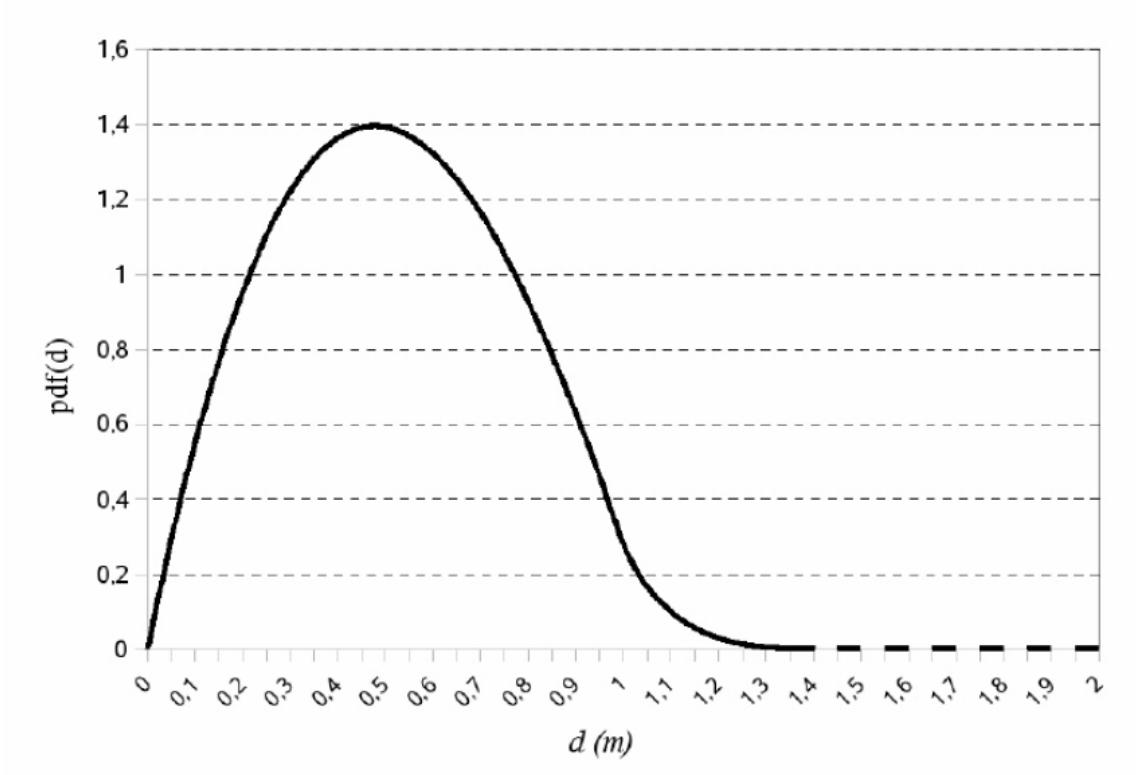

Figura 4-9 Pdf de la distancia entre dos nodos

Para conocer la distancia media entre dos nodos es necesario integrar la ecuación anterior, de lo cual se obtuvo:

$$\begin{aligned} \bar{l} &= E[d] = \int_a^b x f_d(x) dx \\ &= p(0 \le x \le W) \int_0^W x f_d(x) dx + p(W \le x \le \sqrt{2}W) \int_W^{\sqrt{2}W} x f_d(x) dx \end{aligned} \quad (2)$$

Resolviendo la integral finalmente se obtuvo que la expresión para estimar la distancia media entre dos nodos que es:

$$\bar{l}(W) \approx 0.598779W \qquad (3)$$

En este modelo se asumió que un nodo se mueve hacia otro siguiendo una línea recta.

4.2. Probabilidad de error en un enlace

Una ruta se compone de varios enlaces, sabiendo esto, se puede estimar fácilmente la probabilidad de error en una ruta a partir de la probabilidad de error de los enlaces que la componen.

Hay un excelente trabajo de McDonald y Znati [13] donde se deduce analíticamente la probabilidad de que una ruta esté disponible en una MANET. En ese trabajo usaron una distribución de Raleigh para representar la distancia movida por un nodo. A partir de la distribución de Raleigh obtuvieron la Función de Distribución Acumulada (CDF, *Cumulative Distribution Function*) de la distancia movida por un nodo necesaria para calcular la probabilidad de que una ruta esté disponible.

El Grupo de Servicios Telemáticos de la UPC tomó como punto de partida el modelo matemático desarrollado por McDonald y Znati. Con el objetivo de simplificar la expresión y su solución, en el modelo analítico desarrollado por el Grupo de Servicios Telemáticos se asume que los nodos se mueven en línea recta desde una posición inicial hacia una posición final, en la figura 4-10 se muestra el esquema de movimiento. El movimiento de los nodos se aproximó con una distribución lineal.

$$p(d \leq x) \approx linear_{appr}(x, r) = \frac{x}{r},\ 0 \leq x \leq r \qquad (4)$$

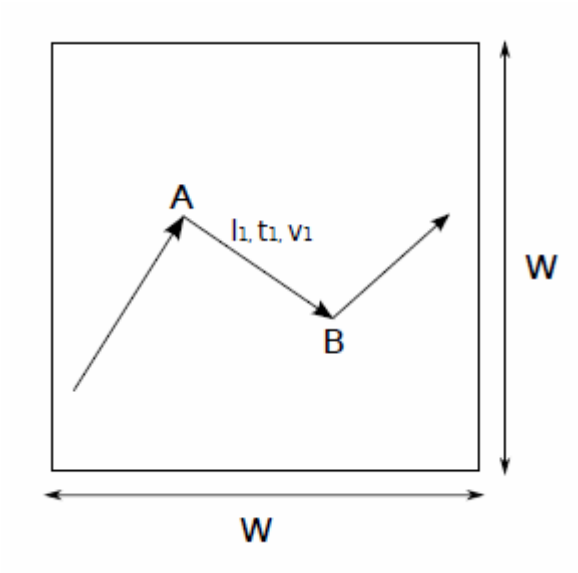

Figura 4-10 Esquema de movimiento de los nodos

En [13] también se dedujo la pdf de la distancia que un nodo debe viajar antes de alcanzar el límite de la celda de cobertura de otro nodo. En el modelo desarrollado por el Grupo de Servicios Telemáticos el nodos está inicialmente en cualquier lugar dentro de la celda de cobertura y se mueve siempre alejándose del centro de la celda con una dirección uniformemente distribuida en (0, 2π). Con base en el trabajo realizado en [13] y las consideraciones arriba mencionadas el Grupo de Servicios Telemáticos obtuvo la función de distribución de probabilidad de la distancia que un nodo debe viajar antes de alcanzar el límite de la celda de cobertura de otro nodo (5).

$$f_x(x) = \frac{4}{\pi r^2}\sqrt{r^2 - x^2}\ ;\ 0 \le x \le r \qquad (5)$$

Finalmente, se obtuvo la expresión (6) que calcula la probabilidad de error en un enlace como la probabilidad de que el nodo *m* deje la celda de cobertura del nodo *n* en un tiempo *t*.

$$\begin{aligned} P_{link\ error}(x,r) &= \int_0^x (1 - CDF_d(y))pdf_d(y)dy \\ &\approx \int_0^x (1 - \frac{y}{r})\frac{4}{\pi r^2}\sqrt{r^2 - y^2}dy \\ &= \frac{4}{\pi r^2}[xr - 0.22x^2 - \frac{r^2}{3} + \frac{\sqrt{(r^2 - x^2)^3}}{3r}],\ 0 \le x \le r, \end{aligned} \qquad (6)$$

En la expresión anterior *(1 – CDF_d (y))* es igual a *p(d > y)* donde *d* es la variable aleatoria de la distancia movida por un nodo e *y* es un umbral de distancia, *pdf_d (y)* es la función de distribución de probabilidad de la distancia que un nodo viaja antes de alcanzar el límite de la celda de cobertura de otro nodo.

La ecuación (4) fue utilizada para aproximar la *CDF* de la distancia movida por un nodo.

Analizando la ecuación (6) y con el fin de simplificar la expresión de la probabilidad de error en un enlace se hizo la siguiente aproximación.

$$p_{link\ error}^{linear} \approx 1 - e^{-\frac{x}{r}},\ 0 \leq x \leq r \qquad (7)$$

Recordando que *x=vt* la ecuación (9) se puede poner de la siguiente manera,

$$p_{link\ error}^{linear} \approx 1 - e^{-\frac{vt}{r}},\ 0 \leq vt \leq r \qquad (8)$$

La figura 4-11 compara las gráficas de las ecuaciones (6) y (7).

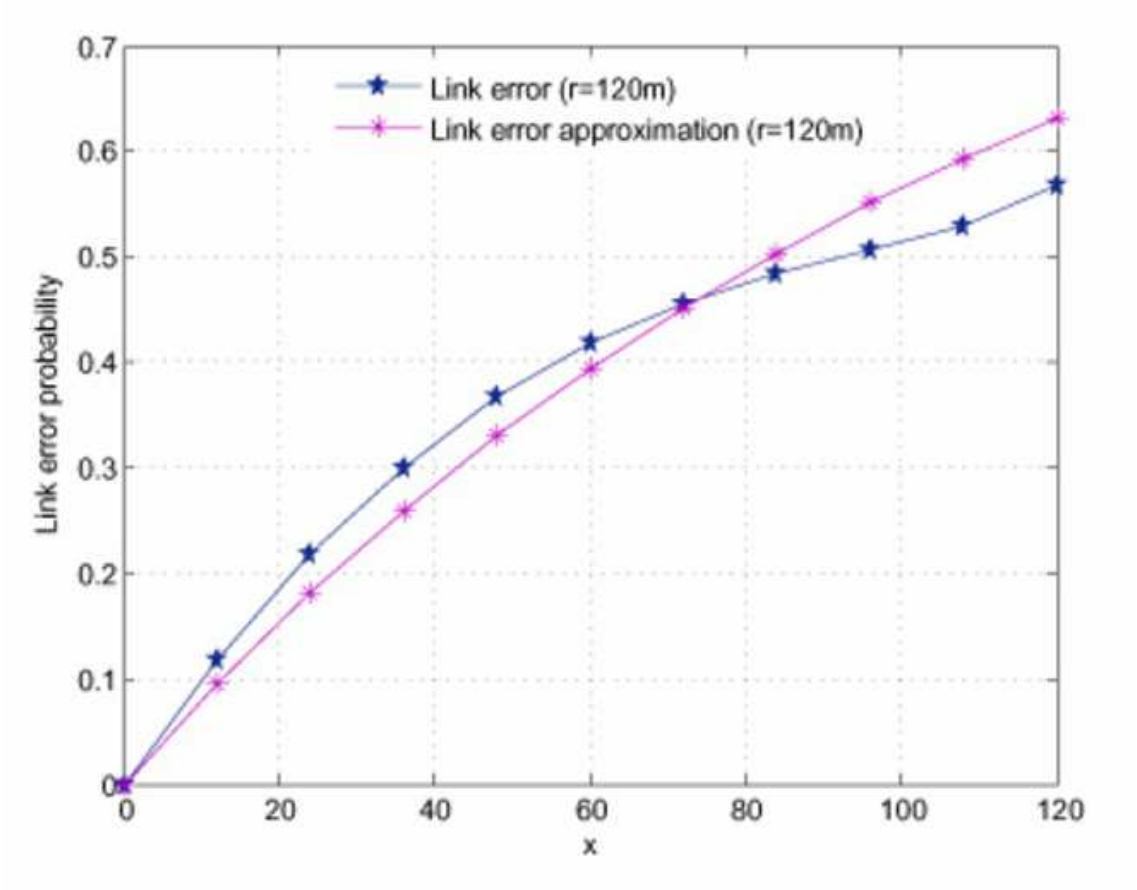

Figura 4-11 Probabilidad de error en un enlace con y sin aproximación

4.3. Probabilidad de error en una ruta

Como se mencionó con anterioridad, una ruta está compuesta de varios enlaces. La probabilidad de error de una ruta se calculó como el promedio de la probabilidad de error de cada enlace por el número de enlaces que forman la ruta. Entonces:

$$p_{path\ error} = 1 - \sum_{i=1}^{H} (1 - p_{link\ error})^i p(ihops) \qquad (9)$$

Donde *H* es el número máximo de saltos y *p(hops)* es la probabilidad de que haya *i* saltos,

$$H = min(N - 1, \frac{\sqrt{2}W}{r}) \qquad (10)$$

La probabilidad de tener *i* saltos se obtuvo a partir de la ecuación (1):

$$p(ihops) = \int_{(i-1)r}^{ir} f_d(x)dx \qquad (11)$$

$$\int_{(i-1)r}^{ir} f_d(x)\,dx = \begin{cases} \frac{\pi}{W^2}x^2 - \frac{8}{3W^3}x^3 + \frac{1}{W^4}x^4 & 0 \le d \le W \\ \frac{2x^2}{W^2}\left(\operatorname{arcsen}\frac{W}{x} - \operatorname{arccos}\frac{W}{x} + \frac{4x}{W}\sqrt{1-\left(\frac{W}{x}\right)^2} - \frac{\left(W^2-x^2\right)\left(8\sqrt{x^2-W^2}-3W\right)}{3W^3} - \frac{x^4+2W^2x^2}{2W^4}\right) & W < d \le \sqrt{2}W \\ 0 & d > \sqrt{2}W \end{cases} \qquad (12)$$

Este proyecto se centra en el tiempo que una ruta está disponible con una probabilidad dada. El Grupo de Servicios Telemáticos, con el objetivo de conocer el tiempo que una ruta está disponible desarrolló la ecuación (9) y obtuvo un polinomio donde la variable es el tiempo que transcurre antes de que un error ocurra en una ruta con una probabilidad fija.

4.4. Ejemplo numérico

En este apartado se va a considerar un ejemplo numérico de estudio particular para evaluar la probabilidad que se rompa una ruta, calculada como se observa en la fórmula (9). De esta manera, se podrá comparar este modelo analítico con los valores obtenidos mediante simulaciones, con tal de poder validar dicho modelo. En este ejemplo se consideran los siguientes valores: N=100 nodos, W=500 metros de área de simulación cuadrada, r=120 metros de cobertura de los nodos y s=0 de desviación de la velocidad de los nodos, mientras que la velocidad de los nodos *m* se considera como parámetro. A continuación, se calculan todos los parámetros que aparecen en la ecuación (9).

Primero se calcula la inversa de la duración media de los intervalos (λ):

$$\lambda(W = 500, \mu = 20) = \frac{20}{0'598779 \cdot 500} \approx 0{,}0668026 \qquad (12)$$

Esto significa que cada nodo cambiaría de dirección después de cada 1/ λ≈15 segundos en media.

También hay que calcular el número máximo de saltos (*hops*) que puede tener una ruta a partir de la ecuación (10). En este caso H=5.

Finalmente, después de calcular todos los parámetros necesarios y hacer todas las operaciones pertinentes se puede expresar la probabilidad de que se rompa una ruta como:

$$p_{patherror}(N=100nodes, W=500m, r=120m, \mu, t) = \\ =1-\left[e^{-\frac{\mu\cdot t}{120}}\cdot 0{,}1457+e^{-\frac{\mu\cdot t}{60}}\cdot 0{,}3097+e^{-\frac{\mu\cdot t}{40}}\cdot 0{,}3122+e^{-\frac{\mu\cdot t}{30}}\cdot 0{,}1930+e^{-\frac{\mu\cdot t}{24}}\cdot 0{,}03781\right] \quad , \; 0\le t\le 120/\mu \tag{13}$$

Tomando como parámetros el tiempo (*t*) y la velocidad media de los nodos (*m*), a partir de la ecuación (13) se ha obtenido la gráfica siguiente:

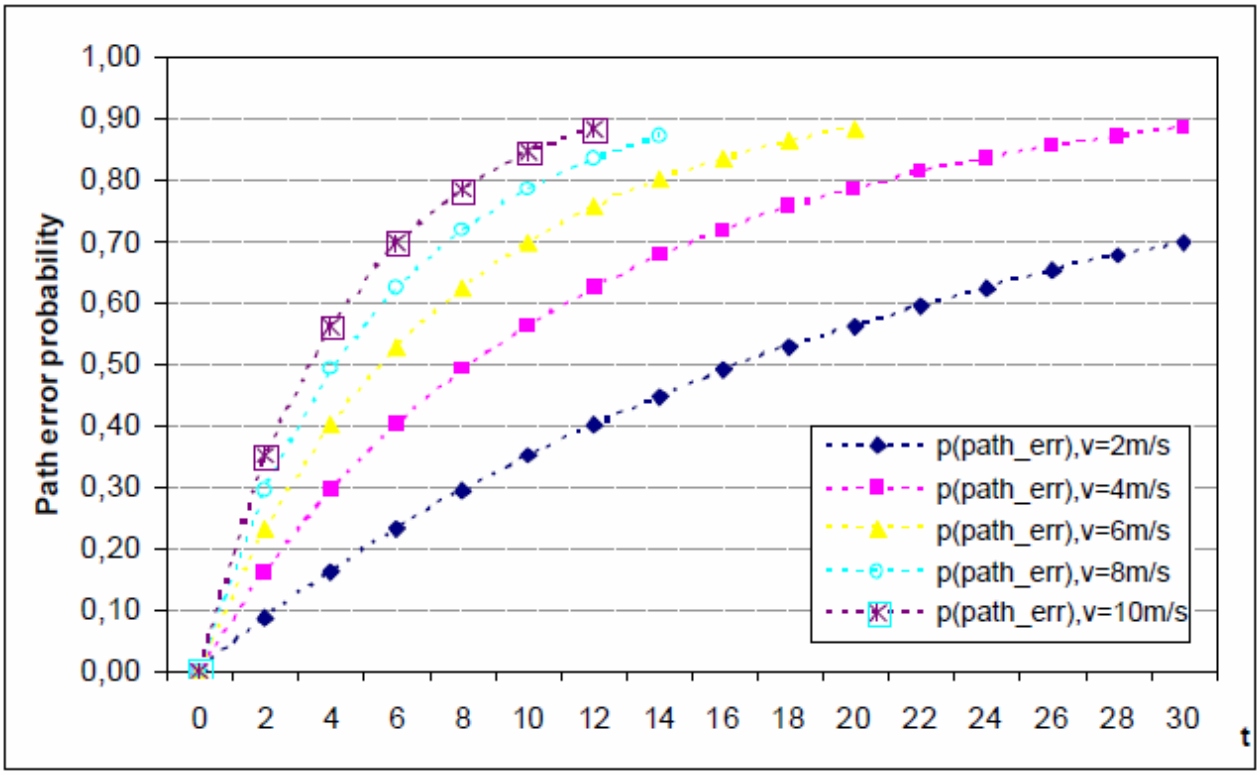

Figura 4-12 Probabilidad de error de ruta en función del tiempo y de la velocidad de los nodos. r=120m, W=500m, H=5 hops.

Como era de esperar, cuanto mayor es la velocidad de los nodos, antes se rompen las rutas y por lo tanto, mayor es la probabilidad de error de las rutas. Por ejemplo, después de 10 segundos, la probabilidad de error en la ruta es de 35%, 56%, 70%, 78% y 85% para una velocidad media de 2m/s, 4m/s, 6m/s, 8 m/s y 10m/s respectivamente.

De acuerdo con la gráfica anterior, para una máxima probabilidad de error en la ruta del 60%, es decir, $P_{threshold}$=0.6, el nodo fuente necesitaría encontrar una ruta alternativa antes de llegar a $T_{path\ life}$. De esta manera, el nodo fuente podría decidir el periodo de tiempo óptimo del algoritmo de encaminamiento (llamado $T_{routing}$) para encontrar nuevas rutas. En este caso, $T_{routing}$ sería igual a $T_{path\ life}$.

Se puede obtener el $T_{path\ life}$ analíticamente a partir de la ecuación (13) sólo sustituyendo $t=T_{path\ life}$ cuando $p_{path\ error}$ es igual a $P_{threshold}$=0.6. Para obtener dicho valor, hace falta aplicar análisis numérico para resolver la ecuación no lineal resultante. En nuestro caso, hemos utilizado el método de la bisección [21], que es un algoritmo de búsqueda de raíces que trabaja dividiendo el intervalo a la mitad y seleccionando el subintervalo que tiene la raíz.

$$P_{th} = 1 - \left[e^{-\frac{\mu \cdot T_{path\ life}}{120}} \cdot 0{,}1457 + e^{-\frac{\mu \cdot T_{path\ life}}{60}} \cdot 0{,}3097 + e^{-\frac{\mu \cdot T_{path\ life}}{40}} \cdot 0{,}3122 + e^{-\frac{\mu \cdot T_{path\ life}}{30}} \cdot 0{,}1930 + e^{-\frac{\mu \cdot T_{path\ life}}{24}} \cdot 0{,}03781 \right] \quad (14)$$

Resolviendo esta ecuación se obtienen los resultados que se presentan en la tabla siguiente:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)
	Modelo analítico con P=0,6
2	22,32
4	11,19
6	7,44
8	5,56
10	4,51

Tabla 4-1 Resultados duración media de una ruta

Capítulo 5. Escenarios de trabajo

Las simulaciones de este proyecto se han realizado en base a distintos escenarios con la idea de simular situaciones reales en las que puede existir una red MANET, en concreto se han dividido en los dos escenarios siguientes:

- Escenario pequeño
- Escenario mediano

Vamos a ver con más detalles las características de cada uno de ellos.

5.1. Escenario pequeño

Este tipo de escenario nos permite simular gran cantidad de situaciones de redes MANET. Por ejemplo podrían ser redes creadas dentro de cafeterías, museos, salas de exposiciones o conferencias o incluso una escuela o universidad. Este escenario se caracteriza por tener pocos nodos y con una baja movilidad ya que se tratan de personas que van andando o están quietas. En la tabla del final vemos los parámetros simulados en este escenario.

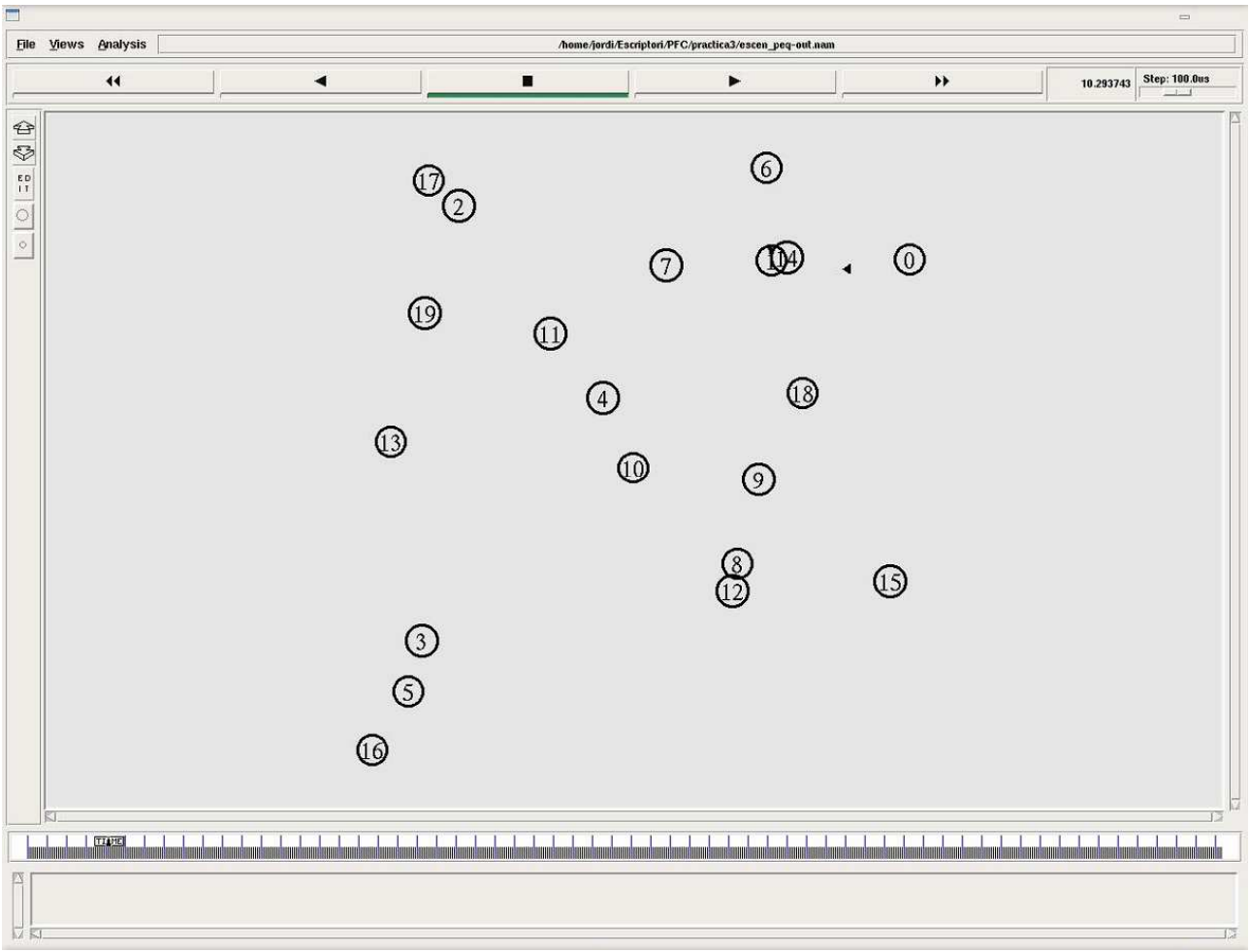

Figura 5-13 Ejemplo captura NAM de escenario pequeño

5.2. Escenario mediano

Este segundo escenario tiene unas dimensiones bastante mayores al anterior. Como ejemplos de redes MANET que puedan encajar con este modelo de escenario, pueden ser: aeropuertos, campus universitarios, congresos, etc. Estos entornos ya son mayores y con mucha más gente, por lo tanto aumentará el número de nodos. La movilidad será parecida al del caso anterior, personas andando o paradas. Podemos ver los parámetros usados en las simulaciones en la tabla del final.

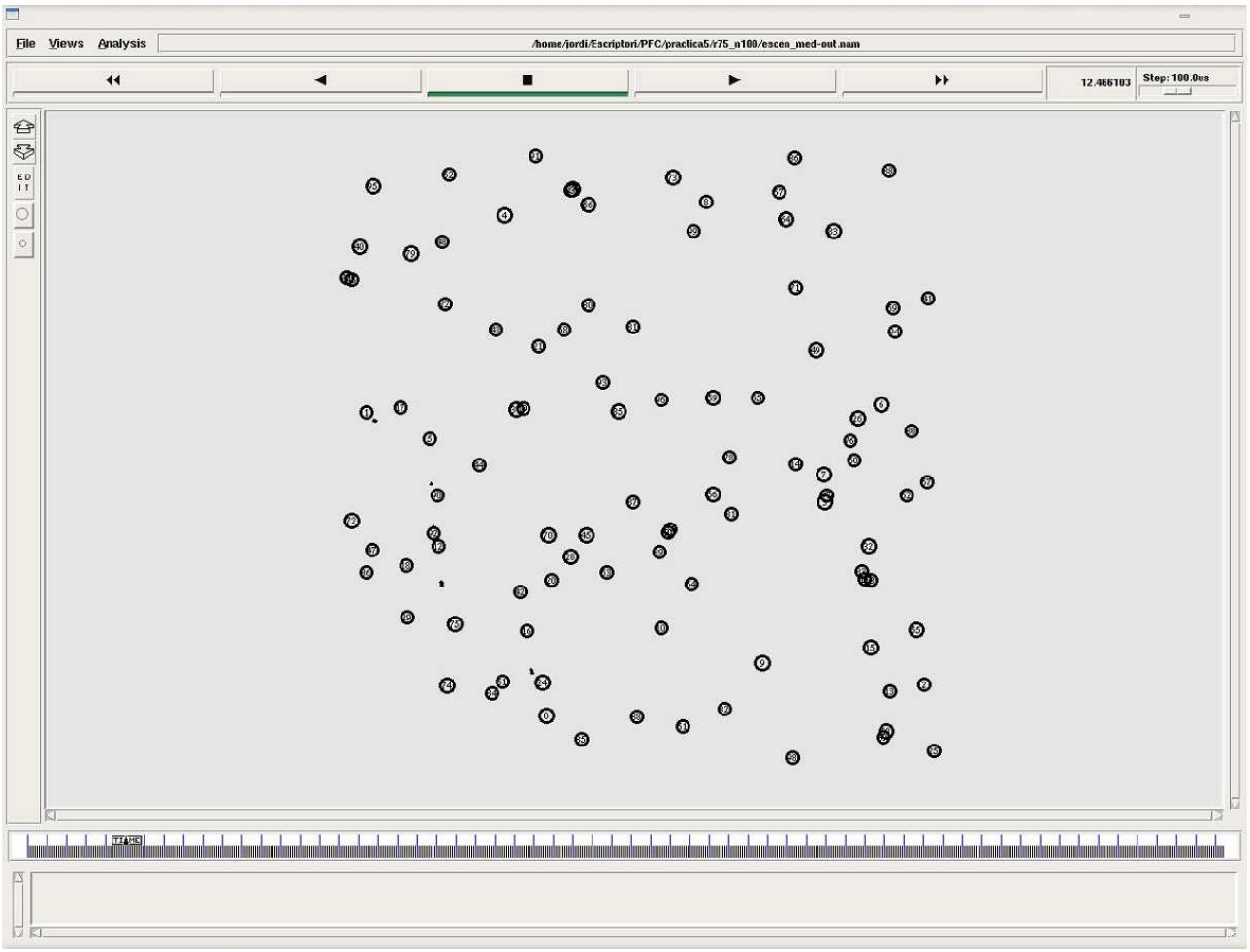

Figura 5-14 Ejemplo captura NAM de escenario mediano

5.3. Parámetros utilizados en las simulaciones

En la tabla siguiente podemos observar los parámetros usados en las simulaciones para cada tipo de escenario. Los parámetros aquí descritos son los que se modifican para crear los distintos escenarios y casos a simular, estos parámetros se modifican algunos en los ficheros de simulación NS-2, otros en los parámetros que se introducen en el *setdest* en el momento de generar los patrones de movimiento y otros parámetros se modifican en ambos casos. De cada escenario se han realizado diez simulaciones con los mismos parámetros cada una cambiando las posiciones iniciales, las velocidades y direcciones de los nodos de forma aleatoria. Es decir, introduciendo los mismos parámetros al *setdest* y obteniendo así una simulación nueva cada vez aunque con las mismas características. Una vez hechas las simulaciones se han obtenido como resultados los datos relevantes a través del filtrado *awk* de los ficheros de trazas obtenidos. Al tener diez simulaciones distintas con las mismas características, se han promediado los resultados obtenidos. Estos resultados los veremos en el capítulo correspondiente. El patrón de movimiento de los nodos en todos los escenarios es el *Random Waypoint* [14], que es el mismo que se ha supuesto en el modelo analítico, que consiste en un movimiento aleatorio. Este patrón

es el que se suele usar aunque luego en la realidad los nodos siguen otros patrones de movimiento más realistas (siguen calles, van en grupos, etc.). El funcionamiento de dicho patrón está explicado más detalladamente en el apartado 6.8 dedicado al *setdest* del capítulo 6.

	Escenario pequeño	**Escenario mediano**
Dimensiones	200x200	500x500
Nº Nodos	20	100, 200
Nº fuentes interferentes	0, 2	0, 2
Protocolos Encaminamiento	DSR	DSR
Patrones de Movilidad	*Random Waypoint*	*Random Waypoint*
Tamaño Paquete (*Bytes*)	1500	1500
Cobertura (m)	30, 50	75, 150
Velocidad de los Nodos (m/s)	0’5, 0,8, 1, 1’5, 2	0’5, 0,8, 1, 1’5, 2
Tráfico Conexión Principal (Mbps)	1	1

Tabla 5-2 Características de los escenarios de simulación

Capítulo 6. *Network Simulator* NS-2

6.1. Introducción

Con el objetivo de analizar distintos parámetros de las redes MANET es básico disponer de un potente simulador, en este caso el simulador escogido ha sido el *Network Simulator* [15] en su versión 2 (NS-2 a partir de ahora) por su potencial y por su amplio uso en este tipo de redes.

NS-2 es un simulador de redes dirigido por eventos. El uso de este simulador está ampliamente extendido en ambientes académicos debido a que está escrito en código abierto y que hay mucha documentación sobre éste en Internet [16]. Se pueden simular gran variedad de protocolos tanto *unicast* como *multicast* y sobretodo muchos de los protocolos propios de las redes MANET.

NS-2 está encapsulado dentro del lenguaje Tcl (*Tool Command Language*). El motor del simulador está implementado en C++ y está configurado y controlado mediante la interfaz Tcl. El simulador se invoca vía el intérprete de comandos del NS, que es una extensión del shell otclsh.

Las simulaciones se realizan mediante un programa Tcl. Pero en la versión 2 en lugar de encontrarse encapsulado en el lenguaje Tcl, se sustituye por el *Object Tool Command Language* de MIT (*Massachusetts Institute of Technology*), OTCL (una versión de Tcl orientada a objetos). Utilizando las instrucciones de NS-2 se define la topología de la red, se configuran las fuentes de tráfico, se recogen los resultados en un fichero de salida y se invoca el simulador. Las instrucciones de NS-2 permiten invocar los procedimientos Tcl desde puntos arbitrarios de la simulación, ofreciendo un mecanismo flexible que permite modificar desde la topología de la simulación, hasta registrar acontecimientos que de forma estándar no se registran o se hacen en formatos diferentes.

Una topología de red se define mediante tres primitivas que construyen los bloques: nodos (*nodes*), enlaces (*links*) y agentes (*agents*).

Los nodos se crean mediante la instrucción “ns node” y se enlazan en una topología de red con la instrucción “ns link”. Los nodos son objetos pasivos, que son dirigidos por los agentes, los cuales son los objetos que activamente conducen la simulación. Ejemplos de agentes son las fuentes de tráfico constante o módulos de encaminamiento dinámico, y se crean mediante la instrucción “ns agent”.

6.1.1. Apunte histórico y versión

Network Simulator se empezó a desarrollar como una variante del simulador de red REAL en 1989. En 1995 el desarrollo se ganó el apoyo de DARPA (*Defence Advanced Research Projects Agency*) a través del proyecto VINT (*Virtual Internetwork Testbed*). Este proyecto es una colaboración entre USC/ISI (*University of Southern California’s Information Sciences Institute*), Xerox PARC (*Xerox Palo Alto Research Center*), LBNL (*Lawrence Berkeley National Laboratory*) y UCB (*University of California, Berkeley*), y tiene como objetivo crear un simulador de redes que permita estudiar la escalabilidad y la interacción entre actuales y futuros protocolos de red. Desde 1996 se encuentra en la versión 2 (NS-2). La última versión estable es la NS-2.33, esta versión es la que ha sido usada en este proyecto y data del 31 de Marzo de 2008, aunque ya ha aparecido la versión 2.34 en 2009. A parte de la versión 2, desde el 2006 se está trabajando en desarrollar la tercera versión que se prevé que aparecerá en 2010.

6.2. Instalación del NS-2

NS-2 se creo para ser usado por sistemas operativos Linux. De la página web del proyecto, incluida en la bibliografía, se puede descargar el programa gratuitamente. Para instalarlo lo único necesario es descomprimirlo y ejecutar el comando para instalar. Pasados unos minutos si todo ha ido bien el programa ya estará instalado, pero antes de empezar a usarlo hay que hacer dos pasos más. Primero actualizar las variables de entorno y caminos del *path* y por último es recomendable validar la instalación, esto último puede tardar alrededor de media hora. Los comandos necesarios para este proceso son los siguientes:

```
tar -xzf ns-allinone-2.33.3.tar.gz
cd ns-allinone-2.33
./install
cd ns-2.33
./validate
```

Al instalar NS-2 también se instalan otros complementos, algunos necesarios y otros opcionales, como son los siguientes:

- Componentes obligatorios:
  a) Tcl/Tk (versiones 8.4.18)
  b) Otcl (versión 1.13)
  c) TclCL (versión 1.19)
  d) NS-2 (versión 2.33)

- Componentes opcionales:
  a) nam-1 (versión 1.13)
  b) xgraph (versión 12.1)
  c) Cweb (versión 3.4g)
  d) SGB (versión 1.0)
  e) Zlib (versión 1.2.3)

6.3. Funcionamiento básico del NS-2

Hay dos maneras distintas de ejecutar el simulador, la primera consiste en ejecutar el comando *ns* seguido de un argumento (en general un fichero tcl a ejecutar) y la segunda ejecutando sólo el comando *ns* y entrando en el modo interactivo.

- Con argumentos:

NS-2 abre y ejecuta el fichero que se pasa como argumento, al acabar la simulación vuelve a la línea de comandos. El archivo pasado como argumento tiene que tener extensión .tcl y en él deben estar todas las instrucciones necesarias para realizar la simulación. Usando esta manera de ejecución deberíamos obtener algo así:

```
bash$ ns ejemplo.tcl
bash$
```

- Sin argumentos:

NS-2 entra en modo interactivo y se comporta como cualquier intérprete de comandos, es decir, analiza y ejecuta cada instrucción una a una. En este modo aparece el símbolo % en la consola de comandos, para salir de éste hay que introducir *exit*. Veamos un ejemplo:

```
bash$ ns
% set ns [new simulator]
% …
% exit
bash$
```

Como es lógico, las simulaciones realizadas en este proyecto se han hecho pasando el fichero .tcl como argumento, ya que introducir línea a línea los programas no sería muy eficiente. Además de pasar el fichero como argumento también se pueden pasar más argumentos definiéndolos en el fichero. De esta manera se pueden cambiar algunos parámetros de las simulaciones sin necesidad de cambiar el código del programa. Ejemplo:

```
bash$ ns ejemplo.tcl arg1 arg2 arg3 ...
```

Los resultados obtenidos de las simulaciones se almacenan en archivos de trazas. Estos ficheros se pueden configurar para que contengan distinta información pero en este proyecto se ha dejado la que contiene por defecto.

Es muy importante interpretar los ficheros de trazas, pero éstos son difíciles de entender y digerir, debido a que se presentan como archivos de gran tamaño y multitud de datos sin explicación alguna. Es por ello que se precisa de otros programas para entender e interpretar los resultados obtenidos, algunos son de representación gráfica, como NAM (*network animator*) [17], *gnuplot* o *xgraph*, y otros son programas de filtrado como *awk*. En este proyecto se han usado tanto NAM para visualizar el comportamiento de la red como *awk* para entender y analizar los resultados.

- NAM (*Network AniMator*)

NAM es una de las aplicaciones que se instalan junto al NS-2. Se puede obtener un fichero de salida (.nam) añadiendo unas instrucciones al programa .tcl. Este fichero lo interpreta y ejecuta el NAM presentando una reproducción visual bastante agradable de la simulación realizada. Lo malo es que no se pueden analizar muchos parámetros con este programa.

- Ficheros AWK

AWK es un programa de UNIX que cogiendo como entrada el fichero de trazas (.tr) nos permite operar con los datos o mostrar solo la información deseada para una mejor comprensión. Más adelante se detallan y explican más cosas acerca de los ficheros AWK.

6.4. Fichero de Simulación

Para realizar la simulación de una red MANET con NS-2 ya hemos visto que hace falta crear un archivo en el que se definen una serie de parámetros y elementos. Veamos lo que deben contener los ficheros .tcl para poder hacer una correcta simulación de la red MANET deseada. La estructura es la siguiente:

- Definición de las variables globales
- Definición de los ficheros de trazas (NS-2 y NAM)
- Configuración de los nodos
- Creación de los nodos
- Definición de la posición inicial y generación de los movimientos de los nodos
- Creación y asociación de los agentes de tráfico
- Planificación de sucesos
- Definición del proceso de final de simulación
- Inicio de la simulación

6.4.1. Definición de variables globales

Para empezar la simulación lo primero es crear una nueva simulación, para definir variables se utiliza el comando *set* seguido del nombre de la variable y el valor inicial. Si se trata de definir un objeto de una clase hay que usar el método *new*. Entonces para crear una nueva simulación debemos introducir, por ejemplo:

```
set ns_ [new Simulator]
```

Una vez definida la simulación, cualquier prodecimiento referido a este objeto se realizará empezando con *$ns_*.

El siguiente paso consiste en crear un espacio en el que se desarrolle la simulación, esto se consigue adquiriendo una intancia a la clase *Topography*. Para ello el procedimiento es muy parecido al anterior, por ejemplo:

```
set topo [new Topography]
$topo load_flatgrid 200 200
```

Como en el caso anterior, una vez definida la topología cualquier prodecidimiento que se refiera a ella debe empezar con *$topo*, vemos el ejemplo justo en la línea siguiente en la que se define el tamaño donde se desarrolla la simulación, en este caso 200 por 200 metros.

Por último, también se deben definir otras variables globales, éstas pueden ser variables creadas por nosotros mismos para usar posteriormente en el programa o variables globales ya existentes en librerias, a éstas últimas se le pueden asignar valores diferentes a los que vienen por defecto. En nuestro caso, algunas variables que hay que redefinir pueden ser de configuración de los nodos, de configuración de protocolos en la capa MAC, etc. Veamos algunos ejemplos:

```
set val(chan)  Channel/WirelessChannel;     # Parámetros de configuración de los
set val(prop)  Propagation/TwoRayGround;    # nodos
set val(netif)        Phy/WirelessPhy

Mac/802_11 set dataRate_ 11Mb       ;# Parámetros de la capa MAC
Mac/802_11 set basicRate_ 1Mb
Mac/802_11 set PLCPDataRate_ 1Mb
```

6.4.2. Definición ficheros de trazas (NS-2 y NAM)

Una vez creadas las variables globales, el siguiente paso consiste en definir los archivos donde se almacenarán los resultados de la simulación, estos archivos de trazas pueden ser los de NS-2 (.tr) o los de NAM (.nam). No es necesario crear estos archivos de trazas, almenos no los dos, si no se van a usar, aunque si no se creasen la simulación no nos proporcionaría información alguna. Veamos un ejemplo de definición:

```
set tracefd [open ejemplo.tr w]                ;# Definición para el fichero de
$ns_ trace-all $tracefd                        ;# trazas ns
$ns_ use-newtrace                              ;# Usar el nuevo formato de trazas

set namtrace [open ejemplo.nam w]               ;# Definición para el fichero de
$ns_ namtrace-all-wireless $namtrace 200 200    ;# trazas nam
```

Estas instrucciones generan dos archivos al terminar la simulación: *ejemplo.tr* y *ejemplo.nam*, uno para analizarlo con AWK y el otro para que lo reproduzca el NAM. Los nombres de estos archivos

son iguales por casualidad, pero obviamente no es necesario que lo sean. Para el archivo de NAM es necesario especificar el tamaño de la simulación y en el caso de no tratarse de una red Ad Hoc habría que usar una instrucción distinta. Es importante destacar que aquí se ha incluido la instrucción necesaria para usar el nuevo formato de trazas de NS-2, si no introdujéramos esta instrucción se usaría el formato de trazas antiguo, que es el que viene por defecto. El nuevo formato contiene más información y por lo tanto es preferible usarlo, es el que se ha usado en este Proyecto Final de Carrera.

6.4.3. Configuración de los nodos

Antes de crear los nodos que intervendrán en la simulación, hay que definir primero las características que éstos van a tener. En primer lugar, como ya hemos visto anteriormente, se tiene que crear un procedimiento denominado *god* indicándole el número de nodos, veamos un ejemplo:

```
set god [create-god 10]
```

Una vez creado, se pueden definir algunas opciones como definir el número mínimo de saltos necesarios para ir de un nodo a otro, veamos un ejemplo en el que se define la distancia mínima entre el nodo 2 y el nodo 3 igual a un salto:

```
$god setdist 2 3 1
```

Por último hay una serie de características importante que hay que definir. Para hacerlo se usa el proceso *node-config*. Veamos un ejemplo:

```
$ns_ node-config -adhocRouting DSR                  \
                 -llType LL                         \
                 -macType Mac/802_11                \
                 -ifqType CMUPriQueue               \
                 -ifqLen 50                         \
                 -antType Antenna/OmniAntenna       \
                 -propType Propagation/TwoRayGround \
                 -phyType Phy/WirelessPhy           \
                 -channelType Channel/WirelessChannel \
                 -topoInstance $topo                \
                 -agentTrace ON                     \
                 -routerTrace ON                    \
                 -macTrace OFF                      \
                 -movementTrace OFF
```

Estos son los parámetros que se han definido en el ejemplo:

- Protocolo de encaminamiento: Se define *DSR* que es el que se ha usado en este proyecto, se pueden elegir otras opciones como*: DSDV, AODV, TORA*, etc.
- Capa de enlace: Las opciones son *LL* o *LL/Sat*.
- Capa Mac: En este proyecto se ha usado solo *MAC/802.11*, aunque existen otras opciones como *Mac/Csma/Ca*, *Mac/Sat*, *Mac/Sat/unslottedAloha*, *Mac/Tdma* y *SMAC*.
- Tipo de cola: Para DSR la única válida es *CMUPriQueue*, con otra no funciona. También existen *Queue/DropTail* y *Queue/DropTail/PriQueue*.
- Capacidad de la cola: Número máximo de paquetes que puede almacenar un nodo en su cola.
- Tipo de antena: *Antenna/OmniAntenna*, antena omnidireccional.
- Tipo de propagación: *Propagation/TwoRayGround* (considera no solo el camino directo de las ondas de antena a antena, sino que también considera el efecto de la reflexión de esas ondas en el suelo), *Propagation/FreeSpace* (considera tan solo el camino directo de las ondas, suponiendo que no hay ningún tipo de obstáculo), *Propagation/Shadowing* (considera el efecto de la propagación multicamino).
- Capa física: *Phy/WirelessPhy* o *Phy/Sat.*
- Tipo de canal: *Channel/WirelessChannel* y *Channel/Sat.*
- Topología de red: Definida anteriormente.
- Información de las trazas: Activación o no de distinta información que aparecerá en los ficheros de trazas acerca de la capa MAC, los agentes, el encaminamiento y los movimientos en los ficheros de salida.

6.4.4. Creación de los nodos

Una vez que ya se han definido y configurado las características de los nodos, éstos pueden ser creados. Todos ellos tendrán las características definidas anteriormente. Para crear los nodos se usa la siguiente instrucción:

```
set node_(0) [$ns_ node]
```

En este ejemplo se crea un nodo 0, denominado node_(0). Pero en la mayoría de casos no querremos crear un solo nodo sino que crearemos 20, 50, 100 o cualquier otro número elevado, por lo tanto podremos usar un bucle como puede ser un for, veamos un ejemplo:

```
for {set i 0} {$i < 10 } { incr i } {
set node_($i) [$ns_ node]
$node_($i) random-motion 0
}
```

En este ejemplo se crean 10 nodos, el valor del número de nodos igual que otros muchos valores vistos en el apartado anterior pueden definirse como variable global para evitar problemas en cambiar dichos valores en el futuro. Al definirlos como variable global también se pueden pasar como parámetro al ejecutar la simulación, asi es más cómodo. Por último, es importante destacar que la instrucción $node_($i) random-motion 0 sirve para desactivar que el nodo tenga un movimiento aleatorio. Es aconsejable hacerlo si el movimiento de los nodos se va a establecer mediante otras instrucciones más adelante.

6.4.5. Definición de la posición inicial y generación de movimientos de los nodos

Ahora que ya están creados los nodos hay que situarlos y definir el movimiento que seguirán. Para definir la posición inicial hay que indicar mediante una sencilla instrucción que coordenadas cartesianas (x, y, z) va a ocupar cada nodo. Veamos un ejemplo:

```
$node_(1) set X_ 40.0
$node_(1) set Y_ 40.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 100.0
$node_(2) set Y_ 40.0
$node_(2) set Z_ 0.0
```

En el ejemplo podemos ver como la coordenada z no suele usarse y por ello la inicializamos a 0.

Para crear el movimiento se puede usar la instrucción *setdest* que se contruye de la siguiente forma:

```
$ns_ at $time "$node_(i) setdest <x> <y> <velocidad>"
```

Esto significa que a partir del momento *time* el nodo *i* se desplaza hacia el punto (*x,y*) a una cierta *velocidad* (expresada en m/s). Veamos algunos ejemplos numéricos:

```
$ns_ at 1.000000000000 "$node_(1) setdest 424.729252430537 47.568711543989 0.113710754004"
$ns_ at 1.000000000000 "$node_(2) setdest 163.864354156839 366.730268062068 0.803904537161"
```

Establecer estos movimientos manualmente uno por uno es inviable por lo tanto hace falta ayudarse de alguna otra herramienta. Existen varias herramientas que se pueden compatibilizar con el NS-2, pero la más simple de usar es la instrucción setdest a través de una aplicación que

se incorpora en la instalación de NS-2. Para utilizar el *setdest* es necesario acceder a una carpeta concreta dentro del directorio donde se ha instalado el NS-2, concretamente:

```
.../ns2/ns-allinone-2.33/ns-2.33/indep-utils/cmu-scen-gen/setdest$
```

Una vez nos encontremos en este directorio ya podemos ejecutar la aplicación que nos genera de forma aleatoria la posición inicial y el movimiento de todos los nodos de la simulación, veamos un ejemplo:

```
./setdest -v <versión> -n <numero_nodos> -p <tiempo_pausa> -M <velocidad_maxima> -t
<tiempo_simulacion> -x <x_max> -y <y_max> >> ".../.../movimientos"
```

La versión del *setdest* puede ser la 1 o la 2, en este proyecto se ha usado la versión 1, en la versión 2 hay parámetros diferentes. El tiempo de simulación, el número de nodos, la *x* máxima y la *y* máxima deben coincidir con los de la simulación. El tiempo de pausa es para indicar el tiempo que estarán parados los nodos cuando lleguen a su destino antes de tomar una nueva dirección. Por último los nodos tomarán velocidades aleatorias entre 0 y la velocidad máxima. Esto nos proporciona una posición inicial y unos movimientos completamente aleatorios para todos los nodos de la simulación, para ello genera una gran cantidad de instrucciones que devuelve por pantalla o, en el ejemplo, los almacena en un archivo llamado *movimientos*. Si no hay demasiados nodos no es necesario guardarlo directamente en un archivo pero para simulaciones de 100 nodos o más es imprescindible. Una vez creado el archivo de movimientos mediante la instrucción ./setdest, este archivo se puede pasar como argumento o definirlo como variable global en el fichero de la simulación.

6.4.6. Creación y asociación de los agentes de tráfico

Los nodos por si solos no generan ni reciben tráfico, para hacerlo hay que crear los agentes. Una vez creados estos agentes se deben asociar a distintos nodos. De la implementación de estos agentes dependerán los protocolos de las capas de transporte y superiores. El NS-2 permite implementar una gran cantidad de agentes, veamos algunos de los más usuales:

- Agentes TCP (*Transport Control Protocol*):

Algunos ejmplos de emisores de tráfico TCP son *TCP/Reno*, *TCP/Sack1* o *TCP/Asym*. Ejemplos de receptores son *TCPSink*, *TCPSink/Sack1* o *TCPSink/Asym*.

- Agentes UDP (*User Datagram Protocol*):

El transmisor es *UDP* y el receptor puede ser simplemente un agente que descarta los paquetes *Null*.

A los agentes también les podemos definir el protocolo de la capa de aplicación, los más comúnes son:

- FTP (*File Transfer Protocol*)
- CBR (*Constant Bit Rate*)

Aunque este proyecto se centra en tráfico CBR.

6.4.6.1. CBR sobre UDP

Veamos un ejemplo de cómo se crean los agentes y como se asignan a los nodos. En este ejemplo usamos CBR sobre UDP.

```
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0
set des1 [new Agent/Null]

$ns_ attach-agent $node_(1) $des1
$ns_ connect $udp0 $des1

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 1500
$cbr0 set interval_ 0.005
$cbr0 set rate_ 1mbps

$cbr0 attach-agent $udp0
```

Vemos como primero se crea el agente UDP, luego el destino *Null* y después se establece la conexión entre ambos. A continuación se establece el protocolo a nivel de aplicación y se asigna al agente UDP en este caso. Hay una serie de características del tráfico a nivel de aplicación que se pueden dejar por defecto o asignarles un valor, en el ejemplo vemos como se ha asignado el valor del tamaño de los paquetes, el intervalo de tiempo entre paquetes y la tasa de transmisión de los paquetes, también existen otras posibilidades como añadir ruido.

6.4.7. Planificación de sucesos

Una vez creados y configurados todos los elementos de la simulación, el siguiente paso consiste en planificar cuando debe empezar y cuando acabar su participación. Es decir, hay que planificar cuando tendrá lugar cada suceso. Todos los sucesos se indicarán con una estructura como la siguiente:

```
$ns_ at <time> <event>
```

Dónde se indica en que instante de tiempo (*time*) sucederá un determinado suceso (*event*).

Veamos un ejemplo simple:

```
$ns_ at 10.0 “$cbr0 start”

$ns_ at 150.0 “$cbr0 stop”

for {set i 0} {$i < 10 } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
```

A los 10 segundos se inicia el agente de tráfico CBR y se inicia la transmisión, es mejor siempre dejar un tiempo de seguridad prudencial al inicio de la transmisión. Una vez se acaba la transmisión hay que indicárselo al agente (o agentes si hubiese otros) y también indicárselo a todos los nodos (por eso se usa un bucle). Por último se llama al procedimiento *finish* para que termine la simulación.

6.4.8. Definición del proceso de final de simulación

Para terminar la simulación se usa un procedimiento que se encarga de cerrar los ficheros de salida que se han abierto al inicio de la simulación y ejecutar, si es necesario, alguna aplicación para el analisis de los resultados. Veamos un ejemplo dónde se ejecuta la aplicación NAM para visualizar la simulación:

```
proc finish {}{
        global ns tracefd namtrace
        $ns flush-trace
        close $tracefd
        close $namtrace
        exec nam ejemplo.nam &
        exit 0
```

Se puede observar como primero se cierran los ficheros de trazas del NAM y de NS-2 y luego se ejecuta el fichero de trazas del NAM. Por último se ejecuta exit para salir de la aplicación.

6.4.9. Inicio de la simulación

Finalmente, hay que indicarle al NS-2 que inicie la simulación. Esta instrucción debe estar al final de todo del archivo dónde se ha descrito la simulación ya que antes de inicarse debe estar todo bien definido. Para iniciar la simulación solo hay que poner la instrucción:

```
$ns_ run
```

Una vez incluida esta instrucción el fichero está completo, ya se puede ejecutar para obtener los resultados en forma de ficheros de trazas que luego se deben analizar.

6.5. Fichero de trazas

El fichero de trazas es el resultado fundamental que obtenemos de cada simulación. Éste contiene gran cantidad de datos al respecto de ésta que nos permiten calcular la eficiencia, adaptabilidad, tiempos de transmisión, número de ruta, etc. de cada simulación. A partir de los parámetros proporcionados y los calculos aplicados debemos extraer todas las conclusiones de este proyecto. Por lo tanto, dada la gran importancia de este fichero de trazas, se ha dedicado un capítulo entero al estudio y la utilitzación del fichero de trazas.

6.6. Visualización NAM

El NAM (Network AniMator) es una herramienta de soporte para poder interpretar y entender las simulaciones de un modo visual. De hecho la herramienta no es imprescidible y no nos aporta ningún tipo de datos o información con la cual poder extraer conclusiones directamente. Pero es imprescindible el uso de este programa, sobre todo al principio, para entender que es lo que está sucediendo en la simulación y así saber como tratar los resultados obtenidos.

Así que una vez aclarada la utilidad del NAM, comentaremos algunas de las opciones de simulación que nos proporciona que nos pueden ayudar a interpretar rápidamente lo que está sucediendo en la simulación. Como no son sumamente importantes sólo comentaremos un poco las opciones más importantes:

- Color de los nodos: Para diferenciar los nodos más importantes de la simulación, por ejemplo la fuente y el destino del resto, podemos utilizar la instrucción: **`$node_(0) color red`**

- Forma de los nodos: También podemos diferenciar los distintos nodos con formas diferentes: **`$node_(0) shape [box, hexagon, circle…]`**
- Marcar los nodos: Otra manera de diferenciar los nodos es añadirles una marca, ésta puede ser sólo durante un tiempo y luego quitarsela. Si por ejemplo quisieramos marcar un nodo desde el segundo 20 hasta el 50, lo haríamos con las siguientes insctrucciones:

  `$ns_ at 20.0 "$node_(0) add-mark m3 green box"`

  `$ns_ at 50.0 "$node_(0) delete-mark m3"`
- Poner etiquetas: A los nodos les podemos también añadir etiquetas de la misma forma que se ponen marcas pero en este caso con un texto añadido, por ejemplo: **`$ns_ at 10.0 "node_(0) label \"nodo origen\""`**
- Crear texto informativo: a medida que avanza la simulación podemos añadir un texto en la parte inferior de la pantalla que nos indique cada uno de los acontecimientos programados, por ejemplo: **`$ns_ at 10.0 "$ns_ trace-anotate \"empieza la transmisión\""`**

Para concluir este apartado vemos una captura de pantalla del programa NAM dónde se observa la gran cantidad de herramientas que proporciona para la visualización:

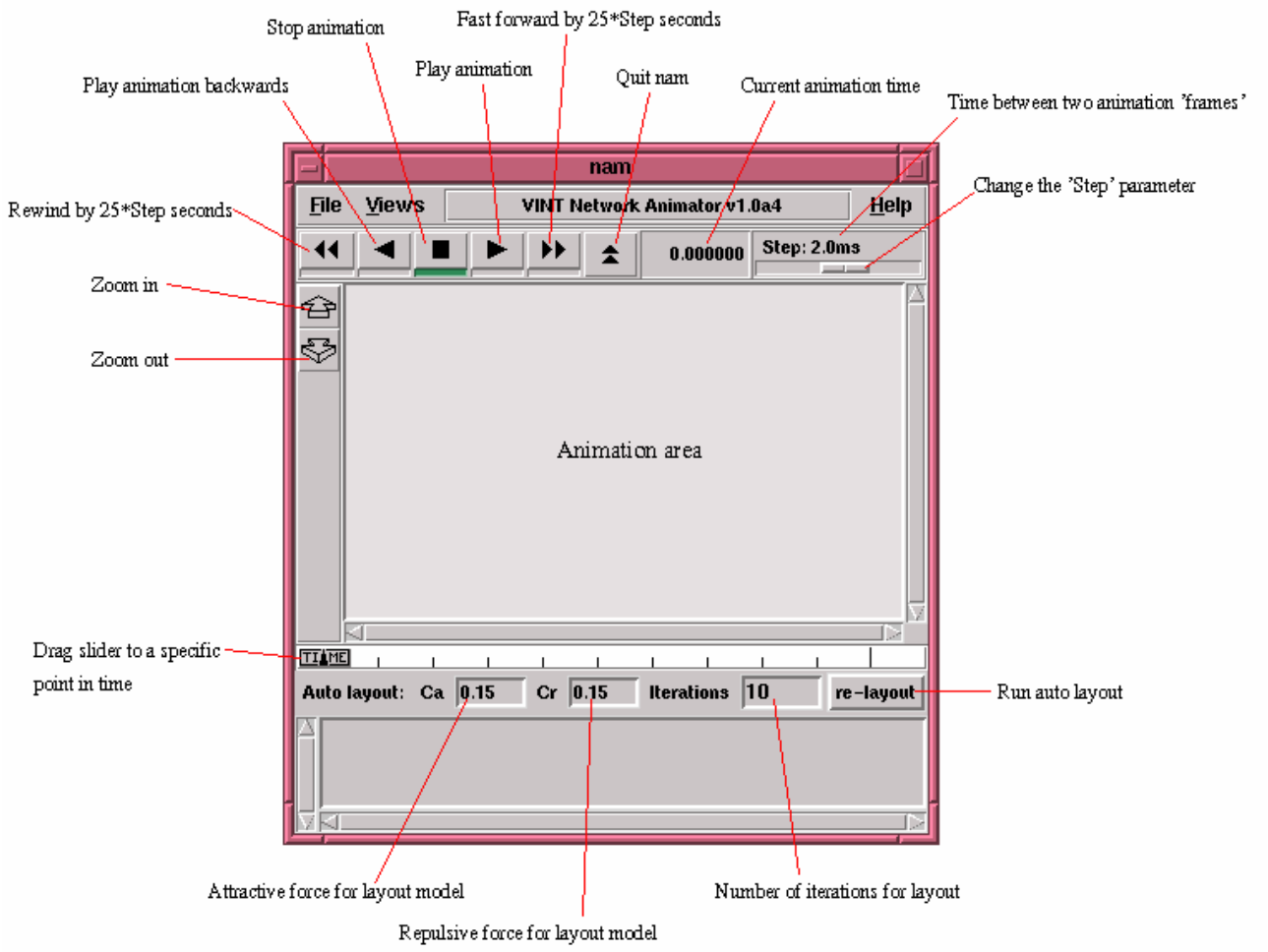

Figura 6-15 Captura del NAM con explicación de sus botones e indicadores

6.7. Establecer cobertura de los nodos

Una opción imprescindible para la realización de este Proyecto Final de Carrera pero no explicada todavía consiste en poder establecer el radio de cobertura de los nodos. Establecer esta cobertura no es imprescindible para hacer una simulación con NS-2 pero si para obtener resultados satisfactorios. Por defecto NS-2 nos asigna gran cantidad de parametros en la configuración de los nodos, enlaces, etc. Pero estos parámetros se pueden configurar según nuestras necesidades. No es objecto de este proyecto enumerar la gran cantidad de variables a configurar que existen en el NS-2, así que sólo comentaremos en este caso como establecer la cobertura de los nodos.

Lo primero que hay que decir al respecto de la cobertura es que no existe un parámetro configurable que establezca directamente la cobertura, pero si que podemos configurar la potencia emitida por los nodos, así que vamos a ver que relación hay entre ambas.

Partimos de la fórmula de la potencia recibida por una antena:

$$\mathrm{Pr} = \frac{Pt \cdot Gt \cdot Gr \cdot ht^2 \cdot hr^2}{d^4 \cdot L} \quad (1)$$

Los parámetros que aparecen en esta fórmula corresponden a:

- Pt es la potencia transmitida por la antena transmisora
- Pr es la potencia recibida en la antena receptora
- Gt es la ganancia de la antena transmisora
- Gr es la ganancia de la antena receptora
- ht es la altura de la antena transmisora
- hr es la altura de la antena receptora
- d es la distancia entre ambas antenas
- L son las pérdidas

Aislando la potencia transmitida de la fórmula anterior obtenemos:

$$\mathrm{Pt} = \frac{Pr \cdot d^4 \cdot L}{Gt \cdot Gr \cdot ht^2 \cdot hr^2} \quad (2)$$

Como vemos la potencia transmitida se puede resumir como una función de la distancia entre antenas, además de otros valores que se comportarán como constantes. De esta forma, sabiendo cual es la cobertura máxima que deben tener los nodos, ajustaremos la potencia máxima que

podrán transmitir. Pero todo esto debemos programarlo en el NS-2 como una función, ésta quedará de la forma siguiente:

```
proc SetPt { coverage } {
set Gt [Antenna/OmniAntenna set Gt_]
set Gr [Antenna/OmniAntenna set Gr_]
set ht [Antenna/OmniAntenna set Z_]
set hr [Antenna/OmniAntenna set Z_]
set RXThresh [Phy/WirelessPhy set RXThresh_]
set d4 [expr pow($coverage,4)]
set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
return $Pt
}
```

En esta función tenemos por argumento la cobertura de la antena expresada en metros, asigna al resto de variables los parámetros que tienen las antenas de los nodos y nos devuelve como resultado la potencia que debemos asignarle a las antenas para que tengan la cobertura indicada.
Los parámetros de las antenas pueden ser los que vienen dados por defecto o los podemos modificar de la siguiente manera:

```
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 0.2
Antenna/OmniAntenna set Gr_ 0.2
```

También podemos ajustar las potencias y pérdidas mediante las siguientes instrucciones:

```
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set Pt_ [SetPt $opt(MNcoverage)]
```

La última sentencia es imprescindible ya que es dónde se llama a la función *SetPt* que nos devuelve el valor de la potencia transmitida que debemos asignar a las antenas.

6.8. Setdest

Una herramienta fundamental para poder realizar las simulaciones de este Proyecto Final de Carrera es el *Setdest* [18]. El *setdest* se instala junto al NS-2 y nos permite generar movimientos para los nodos de un escenario. Esto es imprescindible para las simulaciones ya que debemos tener nodos móviles. A parte del *setdest* hay otros programas que también permiten generar movimientos, incluso más complejos, pero el *setdest* nos permite generar movimientos aleatorios y además es plenamente compatible con el NS-2, de esta manera el fichero generado por el *setdest* está listo para que lo ejecute el NS-2.

El modelo de movimiento que nos genera el *setdest* es el más utilizado, el *random waypoint*. Este modelo incluye pausas entre los cambios de velocidad y dirección de los nodos. Los nodos permanecen inmóviles en una posición durante un tiempo asignado, tiempo de pausa, y luego se dirigen a una nueva posición aleatoria con una velocidad que está distribuida uniformemente entre un mínimo y un máximo, la velocidad es constante durante todo el trayecto hacia el nuevo punto. Cuando llega al destino, vuelve a esperar un tiempo de pausa y entonces escoge un nuevo destino y velocidad aleatorios.

Este modelo describe una gran cantidad de movimientos de la vida real, como podría ser gente que pasea en una conferencia, por un museo, etc.

Una vez explicado que es y para que sirve el *setdest*, vamos a ver como funciona [18]. Primero hay que ir a una carpeta concreta dentro de la carpeta de NS-2, exactamente a la siguiente:

```
…/ns2/ns-allinone-2.33/ns-2.33/indep-utils/cmu-scen-gen/setdest/
```

Una vez ya nos encontramos en dicha carpeta ya se puede ejecutar el *setdest* para que genere el escenario que deseemos. Para ejecutar el *setdest* primero debemos conocer que campos están disponibles, podemos verlos en la siguiente tabla:

Campo	Descripción	Rango de valores	Valor por defecto	Versión 1	Versión 2
-M	Velocidad máxima	≥0	0	Disponible	Disponible
-m	Velocidad mínima	[0, velocidad máxima]	0	No disponible	Disponible
-n	Número de nodos	1,2,…	0	Disponible	Disponible
-P	Tipo de pausa	1 (constante) 2 (uniforme)	1	No disponible	Disponible
-p	Tiempo de pausa	≥0	0	Disponible	Disponible
-s	Tipo de velocidad	1 (uniforme) 2 (normal)	1	No disponible	Disponible
-t	Tiempo máximo	≥0	0	Disponible	Disponible
-v	Versión	1 ó 2	1	Disponible	Disponible
-x	Tamaño del escenario	≥0	0	Disponible	Disponible
-y	Tamaño del escenario	≥0	0	Disponible	Disponible

Tabla 6-3 Opciones y valores disponibles en el *setdest*

Vistos todos los campos posibles con cada versión, veamos un ejemplo de cómo ejecutar el *setdest* con cada versión:

```
./setdest -v 1 -n 10 -p 1 -M 5 -t 150 -x 200 -y 200 > escenario_v1.tcl
./setdest -v 2 -n 10 –s 1 –P 2 -p 1 – m 1 -M 5 -t 150 -x 200 -y 200 > escenario_v2.tcl
```

Es importante, aunque no necesario para simulaciones con pocos nodos, el comando final para que nos guarde la salida del programa en un archivo, si no lo indicásemos nos aparecería volcado en línea de comandos. Del resto de opciones destacar que tanto el número de nodos, como el tiempo de simulación y el tamaño del escenario deben coincidir con nuestro escenario, sino no podremos simular correctamente. También destacar que la versión 2 sirve para añadir algunas posibilidades extra a la versión 1 pero que en este Proyecto Final de Carrera se ha utilizado la versión 1 porque los campos extra de la segunda versión no son acordes con los escenarios simulados.

Debido al carácter aleatorio de este programa, aún introduciendo los mismos parámetros cada vez nos generará unos movimientos aleatorios y por lo tanto distintos a los previamente generados.

El *setdest* nos proporciona un archivo con los movimientos de los nodos pero no simula nada, para hacer una simulación con estos movimientos debemos introducir el archivo a nuestro fichero de simulación. Para ello podríamos copiar el contenido del archivo en el fichero de simulación o mejor, abrirlo directamente desde el fichero de simulación. El archivo dónde se encuentra el escenario generado por el *setdest* lo debemos definir como una variable al inicio del fichero y luego abrirlo una vez ya generados los nodos. El archivo generado por el *setdest* también puede ser introducido como argumento. Estos son los comandos usados en este Proyecto Final de Carrera:

```
set val(cp)             …/escenario_v1.tcl"
…
source $val(cp)
```

Por último y para concluir con este apartado veamos un ejemplo, con 10 nodos, del código generado por el *setdest*. El comando introducido es el siguiente:

```
./setdest -v 1 -n 10 -p 1 -M 0.5 -t 150 -x 200 -y 200 >> escenario_ejemplo.tcl
```

Y una parte del fichero de salida generado por el *setdest* es el siguiente:

```
#
# nodes: 10, pause: 1.00, max speed: 0.50, max x: 200.00, max y: 200.00
#
$node_(0) set X_ 13.475967478706
$node_(0) set Y_ 53.380967933508
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 171.166256582965
$node_(1) set Y_ 71.256061508614
$node_(1) set Z_ 0.000000000000
…
$node_(9) set X_ 37.083892224694
$node_(9) set Y_ 189.391540734433
$node_(9) set Z_ 0.000000000000
$god_ set-dist 0 1 1
$god_ set-dist 0 2 1
$god_ set-dist 0 3 1
$god_ set-dist 0 4 1
$god_ set-dist 0 5 1
$god_ set-dist 0 6 1
…
$god_ set-dist 6 9 1
$god_ set-dist 7 8 1
$god_ set-dist 7 9 1
$god_ set-dist 8 9 1
$ns_ at 1.000000000000 "$node_(0) setdest 69.723469974120 170.285245827243 0.306745480351"
$ns_ at 1.000000000000 "$node_(1) setdest 28.822111570043 189.423510199434 0.003596447815"
$ns_ at 1.000000000000 "$node_(2) setdest 123.209593172091 80.906616157986 0.287348021117"
$ns_ at 1.000000000000 "$node_(3) setdest 136.916633899943 32.404385725994 0.400041099919"
```

```
$ns_ at 1.000000000000 "$node_(4) setdest 147.468525560911 184.857275162431 0.027273446944"
$ns_ at 1.000000000000 "$node_(5) setdest 5.557729852388 83.465974070347 0.326040830188"
$ns_ at 1.000000000000 "$node_(6) setdest 101.549718930000 1.940108370861 0.414746953935"
$ns_ at 1.000000000000 "$node_(7) setdest 180.198665447841 183.881349639809 0.228720497823"
$ns_ at 1.000000000000 "$node_(8) setdest 38.288944671885 105.587277925143 0.010590730631"
$ns_ at 1.000000000000 "$node_(9) setdest 15.886492911817 89.358220153141 0.412721414898"
#
# Destination Unreachables: 0
#
# Route Changes: 0
#
# Link Changes: 0
#
# Node | Route Changes | Link Changes
#    0 |             0 |            0
#    1 |             0 |            0
#    2 |             0 |            0
#    3 |             0 |            0
#    4 |             0 |            0
#    5 |             0 |            0
#    6 |             0 |            0
#    7 |             0 |            0
#    8 |             0 |            0
#    9 |             0 |            0
#
```

6.9. Ejemplo completo

Una vez ya hemos visto parte por parte que debe contener un fichero de simulación para obtener unos resultados útiles, ya podemos ver un fichero de simulación completo. Se pueden ver distintos ejemplos en el Anexo A de este proyecto.

Capítulo 7. Análisis de los ficheros de trazas

7.1. Ficheros de trazas del NS-2

Como ya se ha comentado anteriormente, el resultado básico que nos proporcionan las simulaciones son los ficheros de trazas [19]. También se ha comentado que estos ficheros de trazas son poco digeribles y difíciles de entender. Así que en este capítulo vamos a analizar como son estos ficheros.

Lo primero que hay que conocer es como se estructura el fichero de trazas del NS-2. Cada fila del fichero contiene un evento diferente, puede ser un paquete enviado, un paquete recibido, un paquete perdido, etc. Dentro de cada fila hay una gran cantidad de campos separados por espacios en blanco, el número de campos puede ser distinto en función del tipo de evento. En general, los primeros campos de cada evento son los mismos y son los últimos los que más difieren.

Antes de empezar a ver como son las trazas, hay que destacar que el NS-2 nos ofrece distintos formatos de trazas. Por defecto viene un formato de traza más antiguo con menos información, pero en este proyecto se ha usado el nuevo formato de traza, así que todo lo explicado a partir de aquí sólo será aplicable para este nuevo formato de traza.

Dicho esto, empezaremos viendo un evento que nos servirá de ejemplo para ver que información nos proporcionan los distintos campos:

1 *2* *3* *4* *5* *6* *7* *8* *9* *10* *11* *12* *13* *14* *15* *16* *17* *18* *19* *20* *21* *22* *23* *24* *25* *26*
s -t 10.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 370.37 -Ny 325.73 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -It cbr -Il 1500 -If 0 -Ii 0 -Iv 32 -Pn cbr -Pi 0 -Pf 0 -Po 2

Campo	Marca	Descripción
1	**-**	Indica el tipo de evento. Puede tener cinco valores distintos: • s → send, indica que se trata de un paquete enviado • r → receive, es un paquete recibido • f → forward, es un paquete reenviado por un nodo intermedio • d → drop, se trata de un paquete perdido
2	**-t**	Tiempo en qué ocurre el evento.
-	**-H…**	Los campos que empiecen con esta marca se refieren a información acerca del siguiente salto.
3	**-Hs**	Número de nodo dónde acontece el evento.
4	**-Hd**	Número del siguiente nodo hacia el destino.
-	**-N…**	Los campos que empiecen con esta marca se refieren a información acerca de las propiedades del nodo.
5	**-Ni**	Identificador del nodo.
6	**-Nx**	Coordenada x en la que se encuentra el nodo.
7	**-Ny**	Coordenada y en la que se encuentra el nodo.
8	**-Nz**	Coordenada z en la que se encuentra el nodo.
9	**-Ne**	Nivel de energía del nodo.
10	**-Nl**	El valor de este campo depende de qué capa estamos teniendo en cuenta: • MAC → paquete correspondiente a la capa MAC • AGT → capa de transporte (tcp ó udp) • RTR → paquete enrutado • IFQ → referente a la cola salida del nodo PHY → capa física
11	**-Nw**	Motivo por el cual se descarta el paquete: • --- → el paquete no se descarta • END → final de la simulación • COL → colisión a nivel MAC Existen muchas más posibilidades, no hace falta enumerarlas todas.
-	**-M…**	Los campos que empiecen con esta marca se refieren a información acerca de los paquetes a nivel de capa MAC.
12	**-Ma**	Tiempo esperado en segundos para enviar el paquete a través del canal *wireless.*
13	**-Md**	Dirección MAC del nodo destino.
14	**-Ms**	Dirección MAC del nodo fuente.
15	**-Mt**	Tipo de capa MAC.
-	**-I…**	Los campos que empiecen con esta marca se refieren a información acerca de los paquetes a nivel de IP.
16	**-Is**	Dirección IP del nodo fuente seguido de "." Seguido de el número de puerto del nodo fuente.

17	**-Id**	Dirección IP del nodo destino seguido de "." Seguido del número de puerto del nodo destino.
18	**-It**	Tipo de paquete. Puede ser: cbr, DSR, ack, udp, tcp, ARP…
19	**-Il**	Tamaño del paquete en *bytes*.
20	**-If**	Identificador flowstate.
21	**-Ii**	Identificador de secuencia global del paquete. Identifica el paquete en el fichero de trazas.
22	**-Iv**	Valor del TTL (Time To Live).
-	**-P…**	Los campos que empiecen con esta marca se refieren a información acerca de los paquetes a nivel de "aplicación".
23	**-P**	En este campo se indica el tipo de "aplicación" que puede ser: • arp → Address Resolution Protocol • dsr → Dynamic Source Routing • cbr → Constant Bit Rate • tcp → Transport Control Protocol En función de este campo, el número y tipo de campos siguientes serán distintos. Este ejemplo está hecho a partir de un paquete cbr así que seguiremos con los campos de este tipo de paquete.
24	**-Pi**	Número de secuencia cbr.
25	**-Pf**	Número de veces que el paquete ha sido reenviado.
26	**-Po**	Número óptimo de reenvíos necesarios.

Tabla 7-4 Explicación de los campos de las trazas de NS-2

7.1.1. Trazas del DSR

7.1.1.1. Campos de las trazas del DSR

Vamos a analizas las trazas correspondientes a paquetes del DSR. Para empezar vamos a diferenciar los dos eventos que más observaremos como son el *Route Request* (RREQ) y el *Route Reply* (RREP). Los dos tipos de paquetes contienen campos muy similares y prácticamente sólo se observan cambios en los *flags* correspondientes, los de RREQ y los de RREP. Veamos pues unos ejemplos extraídos de una simulación con 12 nodos, de los cuales el 0 es la fuente y el 11 el destino. Empecemos por un ejemplo de RREQ:

Evento	***1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34***
(1)	s -t 2.007500299 -Hs 0 -Hd -1 -Ni 0 -Nx 25.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.255 –Id 11.255 -It DSR -Il 32 -If 0 -Ii 1 -Iv 32 -P dsr -Ph 1 -Pq 1 -Ps 1 -Pp 0 -Pn 1 -Pl 0 -Pe 0->0 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(2)	r -t 2.008704466 -Hs 4 -Hd -1 -Ni 4 -Nx 25.00 -Ny 100.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md ffffffff -Ms 0 -Mt 800 -Is 0.255 -Id 11.255 -It DSR -Il 32 -If 0 -Ii 1 -Iv 32 -P dsr -Ph 1 -Pq 1 -Ps 1 -Pp 0 -Pn 1 -Pl 0 -Pe 0->0 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0-Id

En este ejemplo vemos como en (1) el nodo 0 envía un paquete de RREQ para encontrar una ruta hacia el nodo 11. En (2) vemos como el nodo 4, podría ser otro nodo intermedio, recibe el RREQ. Podemos ver a quién va dirigido el RREQ en el campo 17 (-Id) que corresponde a la dirección IP del nodo destino, el 11. Veamos un ejemplo de RREP:

Evento	*1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34*
(1)	s -t 2.058927790 -Hs 11 -Hd 10 -Ni 11 -Nx 175.00 -Ny 50.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 11.255 -Id 0.255 -It DSR -Il 60 -If 0 -Ii 17 -Iv 254 -P dsr -Ph 4 -Pq 0 -Ps 2 -Pp 1 -Pn 2 -Pl 4 -Pe 0->11 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(2)	r -t 2.066774363 -Hs 10 -Hd 10 -Ni 10 -Nx 125.00 -Ny 50.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma da -Md a -Ms b -Mt 800 -Is 11.255 -Id 0.255 -It DSR -Il 60 -If 0 -Ii 17 -Iv 254 -P dsr -Ph 4 -Pq 0 -Ps 2 -Pp 1 -Pn 2 -Pl 4 -Pe 0->11 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0

Aquí podemos observar como en (1) el nodo 11 que era el destino del RREQ responde enviando un RREP al nodo 10, ya que este nodo le he enviado previamente un RREQ. A continuación vemos en (2) que el nodo 10 recibe dicho RREP. Aquí podemos ver también como ahora el RREP va dirigido al nodo 0, que es la fuente, en el campo 17 (-Id).

Una vez vistos los ejemplos de paquetes del DSR en los que los campos son los mismos que ya se han comentado anteriormente hasta el 23, veamos los nuevos campos a partir del 24:

Campo	Marca	Descripción
24	**-Ph**	-
25	**-Pn**	Número de nodos atravesados.
26	**-Pq**	*Route Request flag*: 1 si es un RREQ y 0 si no lo es.
27	**-Pi**	Número de secuencia del RREQ.
28	**-Pp**	*Route Reply flag*: 1 si es un RREP y 0 si no lo es.
29	**-Pl**	Longitud del RREP en *bytes*.
30	**-Pe**	Origen -> destino del *source routing*.
31	**-Pw**	*Error report flag*: 1 si hay error y 0 si no lo hay.
32	**-Pm**	Número de errores.
33	**-Pc**	A quién se reporta el error.
34	**-Pb**	Origen y destino del enlace que ha provocado el error.

Tabla 7-5 Explicación de los campos de las trazas del DSR

Por último cabe destacar del ejemplo que otra diferencia entre los paquetes de RREQ y RREP es el tamaño de estos, ya que como se ha comentado anteriormente el paquete de RREP contiene la ruta encontrada y eso conlleva que tenga *bytes* adicionales. En este ejemplo vemos en el campo 19 (-Il) como los paquetes de RREQ contienen 32 *bytes* y en cambio los paquetes RREP contienen 60 *bytes*.

7.1.1.2. Ejemplo de trazas en un *Route Discovery*

Una vez comprendidos los campos de los paquetes DSR vamos a ver un ejemplo completo de descubrimiento de rutas en DSR, desde que el nodo fuente envía el RREQ hasta que este mismo nodo recibe el RREP.

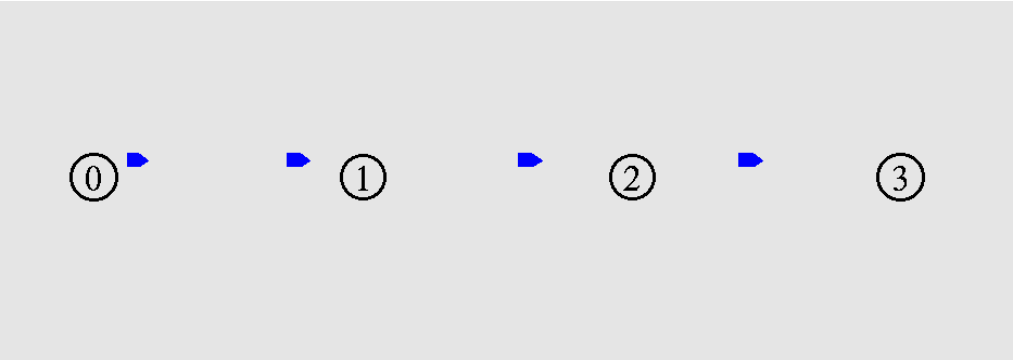

Figura 7-16 Ejemplo de trazas en un *Route Discovery*

Como vemos en la figura anterior el ejemplo consiste en 4 nodos, del 0 al 3. En este ejemplo el nodo fuente será el nodo 0 y el nodo destino el nodo 3. Se ha especificado una cobertura de los nodos de tal manera que los nodos sólo alcancen a comunicarse directamente con los nodos más cercanos, es decir, el 0 con el 1, el 1 con el 0 y el 2, el 2 con el 1 y el 3 y finalmente el 3 con el 2. Por lo tanto para transmitir del nodo 0 al nodo 3, los paquetes deben pasar por todos los nodos por una única ruta. Veamos a continuación las trazas obtenidas del DSR que se envían para establecer la ruta:

Evento	Campos del 1 al 34
(1)	**s** -t 1.000000000 **-Hs 0 -Hd -2** -Ni 0 -Nx 50.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl AGT** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 **-Is 0.0 -Id 3.0 -It cbr -Il 1500** -If 0 -Ii 0 -Iv 32 -Pn cbr -Pi 0 -Pf 0 -Po 0
(2)	**r** -t 1.000000000 **-Hs 0 -Hd -2** -Ni 0 -Nx 50.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 **-Is 0.0 -Id 3.0 -It cbr -Il 1500** -If 0 -Ii 0 -Iv 32 -Pn cbr -Pi 0 -Pf 0 -Po 0

(3)	**s** -t 1.032033503 **-Hs 0 -Hd -1** -Ni 0 -Nx 50.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 **-Is 0.255 -Id 3.255 -It DSR -Il 32** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 1 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(4)	**r** -t 1.032757836 **-Hs 1 -Hd -1** -Ni 1 -Nx 150.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 0 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 32** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 1 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(5)	**f** -t 1.041557792 **-Hs 1 -Hd -1** -Ni 1 -Nx 150.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 0 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 48** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 2 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(6)	**r** -t 1.042730126 **-Hs 0 -Hd -1** -Ni 0 -Nx 50.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 1 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 48** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 2 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(7)	**r** -t 1.042730126 **-Hs 2 -Hd -1** -Ni 2 -Nx 250.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 1 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 48** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 2 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(8)	**f** -t 1.043049540 **-Hs 2 -Hd -1** -Ni 2 -Nx 250.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 1 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 68** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 3 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(9)	**r** -t 1.044381873 **-Hs 1 -Hd -1** -Ni 1 -Nx 150.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 2 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 68** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 3 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(10)	**r** -t 1.044381873 **-Hs 3 -Hd -1** -Ni 3 -Nx 350.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 2 -Mt 800 **-Is 0.255 -Id 3.255 -It DSR -Il 68** -If 0 -Ii 11 -Iv 32 -P dsr -Ph 3 **-Pq 1** -Ps 2 **-Pp 0** -Pn 2 -Pl 0 -Pe 0->16 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(11)	**s** -t 1.049589342 **-Hs 3 -Hd 2** -Ni 3 -Nx 350.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 254 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(12)	**r** -t 1.052540130 **-Hs 2 -Hd 2** -Ni 2 -Nx 250.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma da -Md 2 -Ms 3 -Mt 800 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 254 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(13)	**f** -t 1.052540130 **-Hs 2 -Hd 1** -Ni 2 -Nx 250.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma da -Md 2 -Ms 3 -Mt 800 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 253 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0

(14)	**r** -t 1.055178918 **-Hs 1 -Hd 1** -Ni 1 -Nx 150.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma da -Md 1 -Ms 2 -Mt 800 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 253 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(15)	**f** -t 1.055178918 **-Hs 1 -Hd 0** -Ni 1 -Nx 150.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma da -Md 1 -Ms 2 -Mt 800 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 252 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(16)	**r** -t 1.058777706 **-Hs 0 -Hd 0** -Ni 0 -Nx 50.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 **-Nl RTR** -Nw --- -Ma da -Md 0 -Ms 1 -Mt 800 **-Is 3.255 -Id 0.255 -It DSR -Il 60** -If 0 -Ii 14 -Iv 252 -P dsr -Ph 4 **-Pq 0** -Ps 2 **-Pp 1** -Pn 2 -Pl 4 -Pe 0->3 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0

A continuación se explican los datos más relevantes que se pueden extraer de las trazas, se han resaltado los campos de mayor interés, para entender cómo se establece la ruta a partir de los paquetes RREQ y RREP:

- Vemos en los eventos **(1)** y **(2)** como el agente de tráfico CBR envía un paquete de 1500 *bytes* al agente enrutador del mismo nodo 0. Una vez el agente enrutador lo recibe, inicia el mecanismo *Route Discovery* para hallar una ruta hacia el nodo 3.
- En el evento **(3)** vemos como el nodo 0 envía un RREQ a la red, sin definir quien lo debe recibir, para descubrir la ruta hacia el nodo 3. El tamaño de este paquete es de 32 *bytes*.
- Vemos en el evento **(4)** como el nodo 1 recibe el RREQ enviado por el nodo 0. Este paquete tiene exactamente las mismas características que el anterior, por razones obvias.
- En el evento **(5)** observamos que el nodo 1 reenvía el RREQ recibido previamente. Destacar que ahora el RREQ contiene un total de 48 *bytes*, esto se debe a que el nodo 1 ha añadido su dirección a la lista de nodos atravesados por el RREQ.
- Los eventos **(6)** y **(7)** corresponden a la recepción del RREQ reenviado en **(5)**. Este RREQ lo reciben los nodos 0 y 2 que son con los que puede comunicarse directamente el nodo 1.
- Vemos en el evento **(8)** como ahora es el nodo 2 el que reenvía el RREQ recibido anteriormente, lo hace, como ya hemos visto, añadiendo su dirección en la lista de nodos atravesados, lo cual provoca que el paquete ahora contenga 68 *bytes*. Hay que destacar también que el nodo 0 no ha reenviado el paquete recibido en **(6)** eso es porque el RREQ ya contiene su dirección y por lo tanto en lugar de reenviarlo lo descarta.
- En los eventos **(9)** y **(10)** vemos como ahora reciben el RREQ los nodos 1 y 3. El nodo 1 descartará el paquete igual que ha sucedido anteriormente con el nodo 0. Por otro lado, el RREQ alcanza su destino al recibirlo en nodo 3.
- Vemos en **(11)** como ahora el nodo 3 responde con un RREP. Ahora el nodo 3 ya conoce la ruta de regreso hacia el nodo 0, por eso envía el paquete con dirección al nodo 2 en

lugar de hacer un *broadcast* como sucedía con los RREQ. El paquete RREP tiene un tamaño de 60 *bytes* que ya no cambiará al ser reenviado ya que ahora la ruta completa ya está incluida.

- En los eventos **(12)** y **(13)** el nodo 2 recibe el RREP y lo reenvía hacia el nodo 1.
- En los eventos **(14)** y **(15)** vemos como el nodo 1 vuelve a hacer lo mismo, recibe el RREP y lo reenvía hacia el nodo 0.
- Finalmente en **(16)** el nodo 0 recibe el RREP con la ruta para ir hacia el nodo destino y ya puede iniciar la transmisión del tráfico CBR.

7.1.1.3. Cabeceras en DSR

Como ya se ha comentado anteriormente, la versión más simple del DSR tiene sólo una cabecera para los paquetes y ésta contiene la información de todos los nodos por los que deben pasar los paquetes hacia su destino. Pero el simulador NS-2 no usa la versión más simple, sino que usa la extensión *flowstate*, eso conlleva que existan dos tipos de cabeceras distintas:

- La cabecera normal del DSR que contiene la información de los nodos que debe atravesar el paquete.
- La cabecera del *flowstate* que indica a que flujo de datos pertenece el paquete. El flujo de datos es único en el caso de tener un único nodo fuente y un único nodo destino, como son los ejemplos vistos. En este caso la cabecera del paquete es el identificador de este único flujo de datos.

Para empezar veamos un ejemplo de cabecera normal:

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
SFESTs	2.080008582	_0_	0	[0 -> 11]	1(1)	to	5 [0 |5 10 11]
SFESTs	2.080008582	_0_	3	[0 -> 11]	1(1)	to	5 [0 |5 10 11]
SFESTs	2.080008582	_0_	5	[0 -> 11]	1(1)	to	5 [0 |5 10 11]

Destacar que esta cabecera a parte de encaminar el paquete, también tiene como misión establecer el flujo, o *flowstate*, de datos para próximas transmisiones. Vemos en el ejemplo tres cabeceras que corresponden a tres paquetes distintos. Podemos observar que contienen los nodos por los que deben pasar los paquetes, en este caso la ruta va del nodo 0 al 11 pasando por los nodos 5 y 10. Veamos exactamente qué significa cada campo:

Campo	Descripción
1	Indica el tipo de cabecera, en éste caso normal *ó Established Flowstate.*
2	Tiempo en qué ocurre el evento.
3	Identificador del nodo.
4	Identificador de cabecera, coincide con el número de paquete al que hace referencia.
5	Origen y destino del paquete.
6	Identificador del flujo y cabecera del flujo.
7	Siguiente salto a realizar por el paquete.
8	Contiene las direcciones por las que ha de pasar el nodo. Es la ruta.

Tabla 7-6 Explicación de los campos de las trazas de cabeceras normales del DSR

A continuación vamos a ver un ejemplo de cabeceras *flowstate* que se utilizan después de haberse establecido un flujo de datos:

1	*2*	*3*	*4*	*5*	*6*	*7*
SFs	2.080008582	_0_	7	[0 -> 11]	1(0)	to 5
SFs	2.080008582	_0_	9	[0 -> 11]	1(0)	to 5

En este ejemplo vemos como la cabecera ser ha reducido al haber eliminado el campo que contenía toda la ruta. Veamos que significan los campos en éste caso:

Campo	Descripción
1	Indica el tipo de cabecera, en éste caso *Flowstate.*
2	Tiempo en qué ocurre el evento.
3	Identificador del nodo.
4	Identificador de cabecera, coincide con el número de paquete al que hace referencia.
5	Origen y destino del flujo de datos.
6	Identificador del flujo y cabecera del flujo.
7	Siguiente salto a realizar por el paquete.

Tabla 7-7 Explicación de los campos de las trazas de cabeceras flowstate del DSR

7.1.1.4. Datos en DSR

Aún no hemos visto como aparecen las trazas del flujo de datos. Una vez ya hemos visto los distintos campos y las cabeceras, observaremos como funciona la comunicación de los agentes de tráfico con los enrutadores. Veamos un ejemplo del recorrido de un paquete desde que lo envía el agente de tráfico cbr hasta que llega al agente del destino:

Evento	*1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26*
(1)	s -t 2.088000000 -Hs 0 -Hd -2 -Ni 0 -Nx 25.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 11.0 -It cbr -Il 1500 -If 0 -Ii 33 -Iv 32 -Pn cbr -Pi 11 -Pf 0 -Po 0
(2)	r -t 2.088000000 -Hs 0 -Hd -2 -Ni 0 -Nx 25.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 11.0 -It cbr -Il 1500 -If 0 -Ii 33 -Iv 32 -Pn cbr -Pi 11 -Pf 0 -Po 0
(3)	SFs 2.088000000 _0_ 33 [0 -> 11] 1(0) to 5
(4)	s -t 2.088000000 -Hs 0 -Hd 5 -Ni 0 -Nx 25.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 11.0 -It cbr -Il 1520 -If 0 -Ii 33 -Iv 32 -Pn cbr -Pi 11 -Pf 0 -Po 0
	…
(5)	r -t 2.097276769 -Hs 11 -Hd 11 -Ni 11 -Nx 175.00 -Ny 50.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma da -Md b -Ms a -Mt 800 -Is 0.0 -Id 11.0 -It cbr -Il 1520 -If 0 -Ii 33 -Iv 30 -Pn cbr -Pi 11 -Pf 3 -Po 0
(6)	r -t 2.097276769 -Hs 11 -Hd 11 -Ni 11 -Nx 175.00 -Ny 50.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma da -Md b -Ms a -Mt 800 -Is 0.0 -Id 11.0 -It cbr -Il 1500 -If 0 -Ii 33 -Iv 30 -Pn cbr -Pi 11 -Pf 3 -Po 0

En el evento (1) vemos como el agente cbr del nodo origen, el nodo 0, envía el paquete de datos al agente enrutador del propio nodo y en (2) vemos como el paquete es recibido por éste agente enrutador.

Seguidamente vemos la cabecera *flowstate* en (3) y en (4) como el agente enrutador del nodo 0 envía el paquete al siguiente nodo añadiéndole 20 *bytes* de cabecera IP pero sin cabecera de DSR. Los eventos de los nodos intermedios no se han incluido porque no aportan nada adicional.

Por último vemos en (5) como el paquete es recibido por el agente enrutador del nodo destino, el nodo 11, y luego por el agente cbr del nodo destino en (6) pero ya sin la cabecera IP.

7.1.1.5. Paquetes perdidos

Para terminar con los ficheros de trazas vamos a ver como aparecen en éstos los paquetes perdidos. Veamos el siguiente ejemplo:

	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
(1)	r -t 2.076232698 -Hs 1 -Hd 1 -Ni 1 -Nx 75.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma da -Md 1 -Ms 6 -Mt 800 -Is 11.255 -Id 0.255 -It DSR -Il 60 -If 0 -Ii 16 -Iv 253 -P dsr -Ph 4 -Pq 0 -Ps 2 -Pp 1 -Pn 2 -Pl 4 -Pe 0->11 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(2)	f -t 2.076232698 -Hs 1 -Hd 0 -Ni 1 -Nx 75.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma da -Md 1 -Ms 6 -Mt 800 -Is 11.255 -Id 0.255 -It DSR -Il 60 -If 0 -Ii 16 -Iv 252 -P dsr -Ph 4 -Pq 0 -Ps 2 -Pp 1 -Pn 2 -Pl 4 -Pe 0->11 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0
(3)	d -t 2.077315547 -Hs 1 -Hd 0 -Ni 1 -Nx 75.00 -Ny 150.00 -Nz 0.00 -Ne -1.000000 -Nl IFQ -Nw ARP -Ma da -Md 1 -Ms 1 -Mt 800 -Is 11.255 -Id 0.255 -It DSR -Il 60 -If 0 -Ii 16 -Iv 252 -P dsr -Ph 4 -Pq 0 -Ps 2 -Pp 1 -Pn 2 -Pl 4 -Pe 0->11 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0

Podemos ver en este ejemplo como en (1) el nodo 1 recibe un paquete, en este caso se trata de un paquete de DSR. A continuación el nodo 1 reenvía en (2) el paquete hacia el nodo 0. Finalmente en (3) vemos como el paquete se ha perdido, como nos indica el campo 1 con una *d* de *drop* que significa que el paquete se ha descartado. El motivo en este caso lo encontramos en el campo 10 (-Nl) que nos indica IFQ, esto significa que el paquete ha sido descartado porque la cola, o buffer, estaba llena.

7.1.2. AWK

Una vez visto el formato de las trazas de los paquetes, con todos sus campos y los distintos tipos de eventos que se pueden encontrar, vamos a ver el problema principal: ¿cómo analizamos las trazas?

Como se ha visto cada evento tiene una gran cantidad de campos, pero ese no es el único problema, el principal problema es que en cada fichero de simulación puede haber miles de eventos. En función del número de nodos, la tasa de tráfico, del tiempo de simulación, etc. el tamaño del fichero de trazas puede variar entre unos *kbytes* hasta alcanzar cerca de 1 *Gbyte*. En general, los ficheros de trazas tratados en este Proyecto Final de Carrera has estado alrededor de unos 50 *Mbytes*. Por lo tanto en cada fichero de trazas debemos analizar bastantes campos de cada evento y teniendo que analizar miles o hasta millones de eventos, es decir, que sin ayuda de alguna herramienta esto es completamente inviable. Es aquí dónde aparece la gran importancia del lenguaje AWK.

El AWK [20] es un lenguaje de programación diseñado para procesar datos basados en texto, ya sean ficheros o flujos de datos. El nombre de AWK viene dado por las iniciales de sus tres autores: A. Aho, P. Weinberger y B. Kernighan. Cuando se habla del lenguaje AWK se debe escribir en mayúsculas ya que si se hace en minúsculas se refiere al programa de Unix awk. Este lenguaje es idóneo para trabajar con nuestros ficheros de trazas que contienen información estructurada en campos, ya que sus características permiten descomponer líneas de entradas en campos y comparar estos campos con patrones que se especifiquen.

El lenguaje AWK interpreta cada fila por separado, hasta que aparece un salto de línea, y cada fila la interpreta como una serie de campos separados por espacios, es decir, exactamente como se componen los ficheros de trazas. Para acceder a un campo en el AWK sólo hace falta escribir $ seguido del número del campo deseado. Así escribiendo $1 se accede al primer campo o $19 al campo número 19.

Así pues, para analizar los ficheros de trazas debemos confeccionar un archivo .awk con los comandos que creamos convenientes. Este fichero es como un filtro que extrae la información deseada del gran fichero de trazas. Por lo tanto, una vez tenemos el fichero de trazas y el filtro, solamente hace falta ejecutar un comando para que el fichero de trazas sea filtrado según hemos indicado en el archivo .awk. El comando necesario para filtrar un fichero de trazas es el siguiente:

```
> awk –f filtro.awk trazas.tr
```

La estructura del filtro AWK debe seguir un partón determinado que podemos ver a continuación:

```
BEGIN {
#instrucciones que sólo se ejecutaran una vez
}
{
#instrucciones que se ejecutaran para cada evento
}
END {
#instrucciones que sólo se ejecutaran una vez
}
```

Vemos tres partes claramente diferenciadas. En la parte inicial (BEGIN) se suelen definir e inicializar las variables necesarias para el programa, esta primera parte sólo se ejecuta al inicio y una vez. En la parte intermedia es dónde se programan todas las instrucciones necesarias para filtrar el contenido de los eventos. Esta parte intermedia se ejecuta, a modo de bucle, para cada evento del fichero de trazas por separado, es decir, es dónde se encuentra el cuerpo del filtro. Finalmente, en la tercera parte (END) se ejecutan las instrucciones necesarias para sintetizar los datos obtenidos en el resto del programa y normalmente para imprimir los resultados obtenidos por pantalla (con la sentencia *print*). Esta parte final también se ejecuta una sola vez.

En este Proyecto Final de Carrera se han programado dos filtros awk, uno para calcular el *throughput* y el otro para calcular todo lo referente a las rutas: número de rutas encontradas, número de rutas usadas, tiempo medio de duración de las rutas, etc. Se podía haber implementado en un solo filtro, pero así queda todo un poco más claro al no mezclar conceptos. El filtro encargado del cálculo de *throughput* no reviste de gran complejidad, todo lo contrario que el segundo filtro.

Probablemente programar el filtro que se encarga de obtener los datos necesarios acerca de las rutas del DSR ha sido una de las partes más complejas y laboriosas de este Proyecto Final de

Carrera. La razón es que para programar el filtro es necesario comprender ampliamente el funcionamiento de la red que estamos simulando, en este caso, además es imprescindible conocer la forma de trabajar del protocolo de encaminamiento que estamos analizando, el DSR. Este filtro está programado para interpretar el funcionamiento de la búsqueda y mantenimiento de las rutas del protocolo DSR que se ha explicado anteriormente. Para unos pocos nodos el problema no es demasiado complejo pero para tratar simulaciones con cientos de nodos el filtro debe estar muy finamente programado para tener en cuenta todos los cambios de rutas. Para ello se han usado simulaciones más o menos simples y luego se ha comprobado con sumo cuidado el funcionamiento para simulaciones más grandes, de manera que el filtro ha ido evolucionando hasta el punto de poder extraer resultados altamente precisos que han servido para extraer las conclusiones. Los filtros utilizados se pueden encontrar en el anexo B.

Capítulo 8. Resultados

En este capítulo se presentan y analizan los resultados obtenidos mediante las simulaciones de NS-2 [15]. Estos resultados se han obtenido realizando, en cada caso, 10 simulaciones con las mismas características y promediando luego los resultados obtenidos. Los resultados se presentan en función de la velocidad máxima de los nodos que va de 0’5 m/s a 2 m/s (de 1’8 km/h a 7’2 km/h). Que emulan distintos escenarios como se ha explicado anteriormente en el capítulo 5.

En cuanto a la evaluación de prestaciones realizada, el parámetro estudiado más profundamente ha sido el **tiempo medio de duración de una ruta**, el cual ha sido comparado en cada caso con los valores proporcionados por el modelo analítico explicado en el capítulo 4. No obstante, también se ha estudiado el **número de rutas encontradas**, el **número de rutas encontradas** por un nodo intermedio cualquiera, escogido previamente, y el **número de rutas que realmente se han usado** en cada simulación. Por último, también se ha estudiado el ***throughput***, entendido como el porcentaje de paquetes que llegan al nodo destino respecto a los que envía el nodo fuente.

Finalmente, se han obtenido nuevos resultados de los mismos parámetros, pero cambiando sustancialmente el escenario. Inicialmente sólo existía un nodo fuente y un nodo destino, así que el último caso de estudio se ha hecho con tres nodos fuentes y tres nodos destino y analizando qué le sucede a la transmisión inicial, es decir, comportándose los dos nodos fuente nuevos como interferentes de la transmisión principal.

8.1. Estudio del tiempo medio de duración de una ruta

Se entiende como tiempo medio de duración de una ruta, lo que dura la transmisión de paquetes sin cambiar de ruta. Es decir, que se cuenta el tiempo desde que se encuentra una ruta y se inicia una transmisión hasta que la transmisión por esta ruta se interrumpe, bien sea por un cambio de ruta o bien porque se rompe la ruta y no existe ninguna otra ruta disponible en ese momento. Finalmente se calcula la duración media como el promedio de la duración de todas las rutas usadas en la simulación. Para calcular estos tiempos de duración de las rutas se han hecho simulaciones con una duración de 150 segundos, lo cual es más que suficiente teniendo en cuenta los resultados obtenidos, así se garantiza que lo que se está midiendo es la duración real de las rutas y el final de la transmisión no interviene en los resultados.

8.1.1. Evaluación de prestaciones. Caso I

En la tabla siguiente se pueden ver los parámetros escogidos para realizar estas simulaciones:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x 200	20	500	30	0	0'5, 0’8, 1, 1'5, 2

Los resultados obtenidos se comparan con los del modelo analítico explicado en el capítulo 4, concretamente a partir de la ecuación 9 y son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,85
0,5	31,05	35,22
0,8	28,15	22,01
1	25,77	17,61
1,5	9,87	11,74
2	10,57	8,80

Tabla 8-8 Resultados duración media de una ruta, caso I

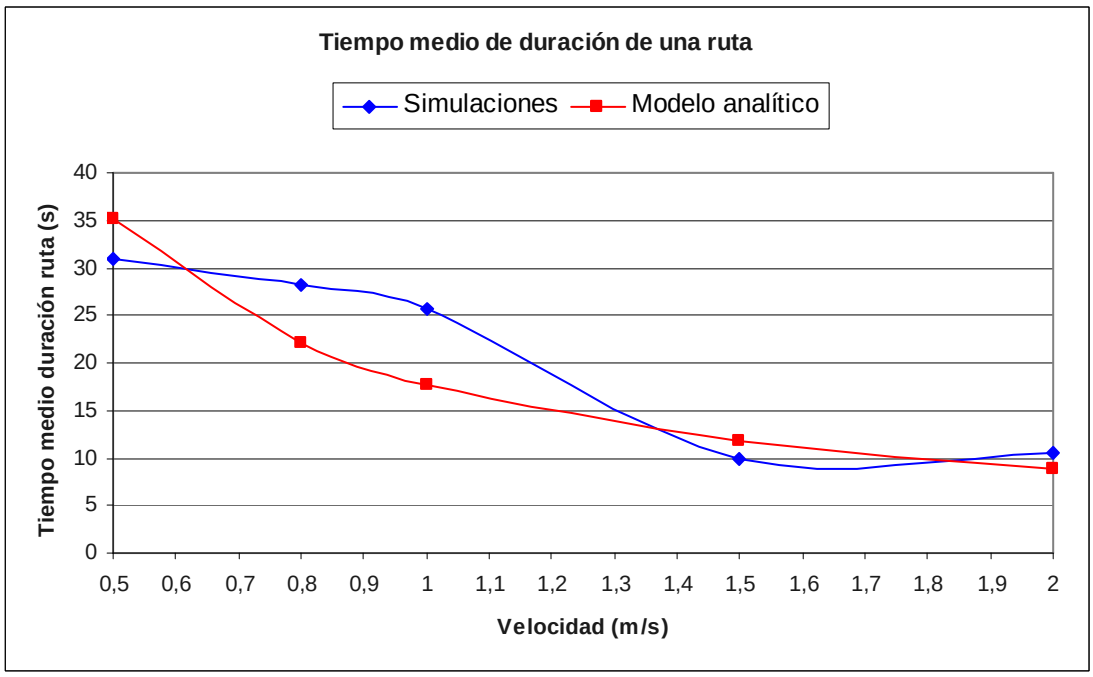

Figura 8-17 Duración media de una ruta, caso I

Como se observa en la gráfica anterior, según el modelo analítico a medida que aumenta la velocidad de los nodos disminuye el tiempo medio de duración de las rutas. Esto es lógico ya que cuanto mayor sea la movilidad de los nodos, más rápidamente se romperá la ruta que se esté usando. Las simulaciones no dan un resultado idéntico al modelo analítico debido a que el número de simulaciones realizadas no es demasiado alto al igual que el tiempo de duración de dichas simulaciones, pero si observamos que la tendencia es la misma, a medida que aumenta la velocidad de los nodos disminuye el tiempo de duración de las rutas. En este caso, es normal que los resultados difieran un poco, ya que al tratarse de una cobertura reducida, en muchas simulaciones no se ha llegado a utilizar ninguna ruta, mientras que en las que se ha usado alguna, ha sido sólo una o muy pocas, así que los resultados obtenidos se han promediado mucho menos que en el resto de casos. De todas formas, aumentando mucho el número de simulaciones, probablemente las gráficas se parecerían aún más.

8.1.2. Evaluación de prestaciones. Caso II

En la siguiente tabla se pueden ver los parámetros elegidos para el siguiente caso, la única variación es que ahora la cobertura de las antenas es mayor:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x 200	20	500	50	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,65
0,5	37,08	44,51
0,8	28,18	27,82
1	23,81	22,26
1,5	14,46	14,84
2	11,15	11,13

Tabla 8-9 Resultados duración media de una ruta, caso II

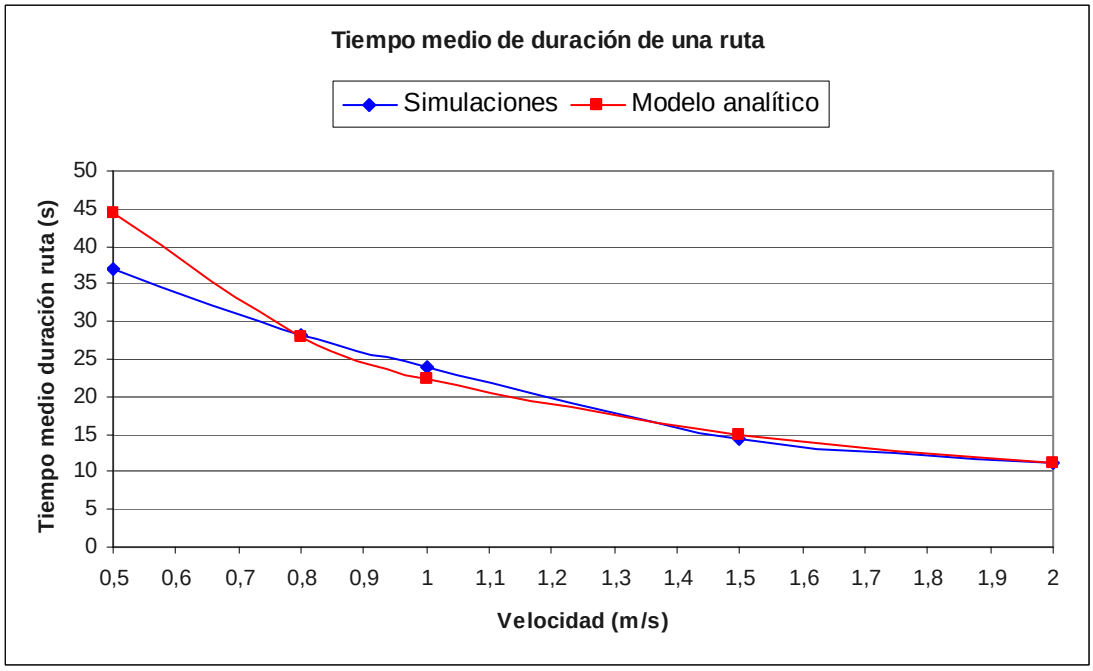

Figura 8-18 Duración media de una ruta, caso II

En la gráfica anterior se observa como la tendencia de los resultados es la misma que en el caso previo, a medida que aumenta la velocidad de los nodos el tiempo de duración de las rutas disminuye. En este caso podemos ver como el resultado de las simulaciones es casi idéntico al resultado dado por el modelo analítico. Sólo hay un valor que difiere un poco que es el primero, eso se debe a que en el primer caso el número de rutas es menor, ya que los nodos se mueven menos, y además hay un factor que distorsiona un poco los resultados que es la duración limitada de las simulaciones. Esto afecta en mayor medida al primer valor que es dónde más duran las rutas, ya que al ser la simulación de 150 segundos (de los cuales 140 transmitiendo), las rutas no pueden durar más que eso. Por lo tanto, en este caso que en general ya se encuentran rutas fácilmente, se puede decir que el comportamiento del modelo analítico se aproxima mucho a los resultados obtenidos mediante simulaciones.

8.1.3. Evaluación de prestaciones. Caso III

En la siguiente tabla se pueden ver los parámetros escogidos para el siguiente caso, que corresponde al escenario mediano, con dimensiones superiores, con más nodos y mayor cobertura de las antenas:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	75	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,3
0,5	13,98	13,87
0,8	9,39	8,67
1	5,69	6,93
1,5	3,68	4,62
2	3,47	3,47

Tabla 8-10 Resultados duración media de una ruta, caso III

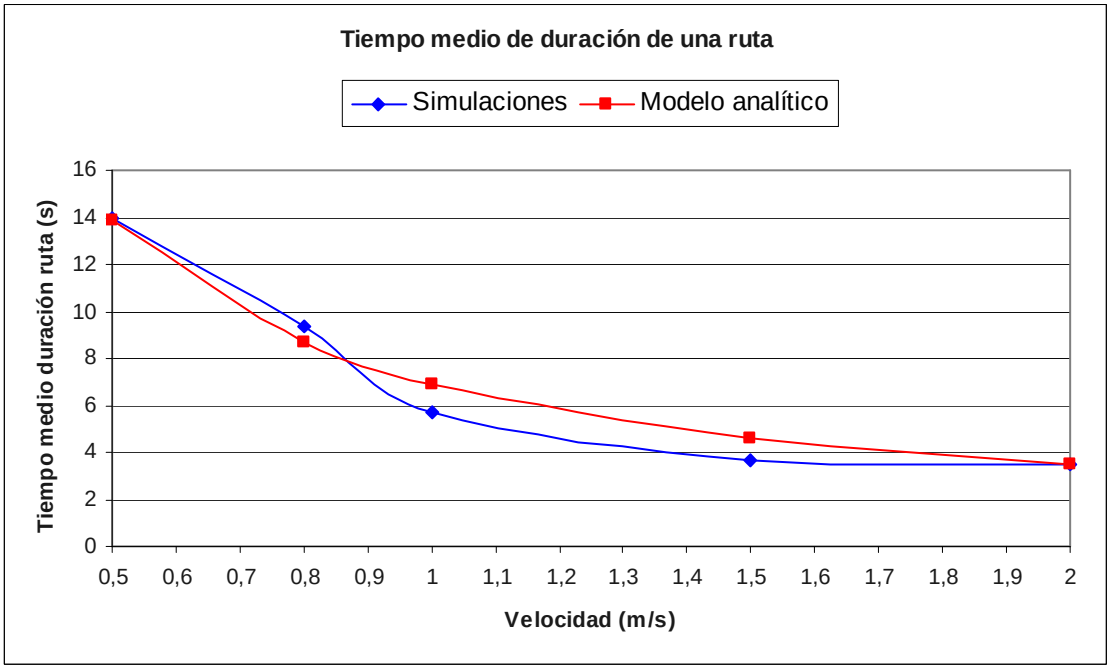

Figura 8-19 Duración media de una ruta, caso III

A partir de la gráfica anterior se puede observar el mismo patrón en los resultados analíticos, así como una gran coincidencia de los resultados de las simulaciones con el modelo analítico. Sólo algunos parámetros difieren ligeramente, lo cual se podría subsanar realizando un mayor número de simulaciones. En este caso nos encontramos en un escenario mucho mayor y con una gran cantidad de nodos, respecto a los casos anteriores vemos que el tiempo de duración de las rutas se ha reducido considerablemente. Esto se debe, en gran parte, a que ahora las rutas tienen más saltos y por lo tanto la probabilidad de rotura de un enlace aumenta sustancialmente.

8.1.4. Evaluación de prestaciones. Caso IV

En la siguiente tabla se pueden ver los parámetros escogidos para el siguiente caso:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	150	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,65
0,5	116,56	-
0,8	99,96	95,89
1	77,73	76,72
1,5	53,74	51,14
2	41,12	38,36

Tabla 8-11 Resultados duración media de una ruta, caso IV

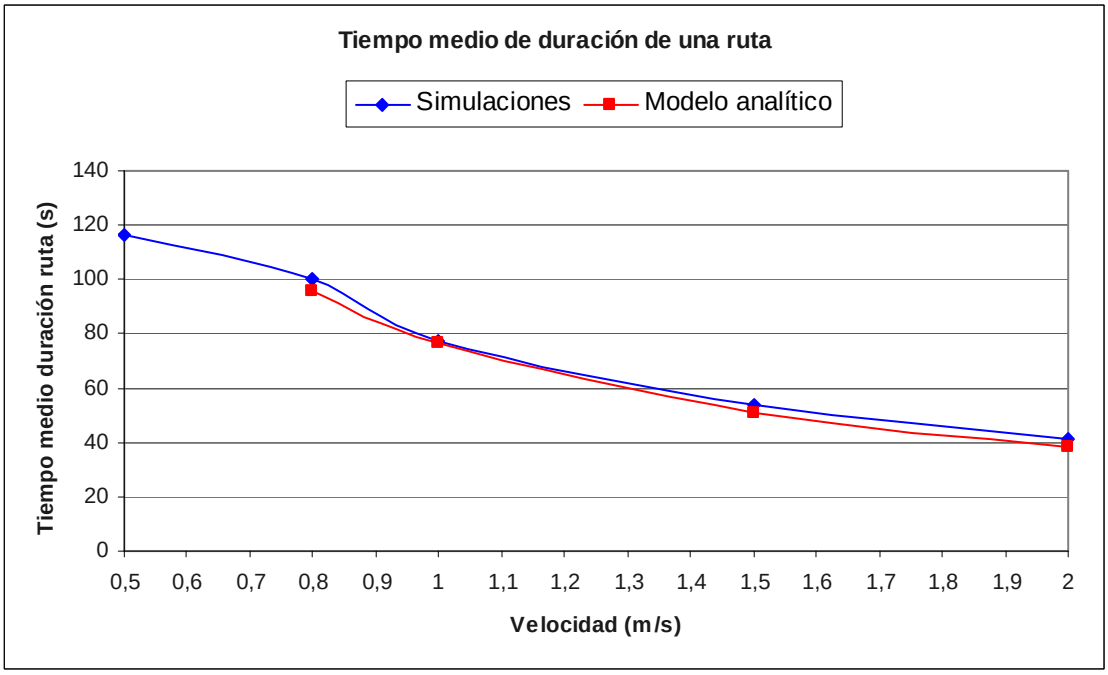

Figura 8-20 Duración media de una ruta, caso IV

Se puede observar en la gráfica anterior que en este caso las simulaciones también aportan unos resultados casi calcados a los proporcionados por el modelo analítico. Pero hay que destacar en este caso un déficit importante, y es que la implementación en MATLAB que se ha hecho del modelo analítico no proporciona resultado alguno para el escenario con velocidad máxima de los nodos de 0'5 m/s. Esto se debe a que el tiempo de duración de las rutas es demasiado elevado, de hecho se observa según las simulaciones que este tiempo es de unos 116 segundos (sobre un máximo de 140). Justamente el primer valor en las simulaciones tampoco se ciñe demasiado a la forma de la gráfica, esto ocurre como ya se ha dicho en un caso anterior a que las simulaciones tienen una duración limitada lo cual distorsiona este valor. Destacar que en este caso la cobertura

es muy grande (150 metros, siendo el espacio de simulación de 500 por 500 metros) lo que junto a un número elevado de nodos proporciona comunicación entre el nodo fuente y el nodo destino en todo momento y con pocos saltos. Esto provoca que las rutas raramente se rompan y que el tiempo medio de duración de las rutas sea en todos los casos muy elevado.

8.1.5. Evaluación de prestaciones. Caso V

En la siguiente tabla se pueden ver los parámetros escogidos para el siguiente caso:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	200	800	75	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,3
0,5	15,94	13,87
0,8	8,77	8,67
1	7,58	6,93
1,5	5,41	4,62
2	3,26	3,47

Tabla 8-12 Resultados duración media de una ruta, caso V

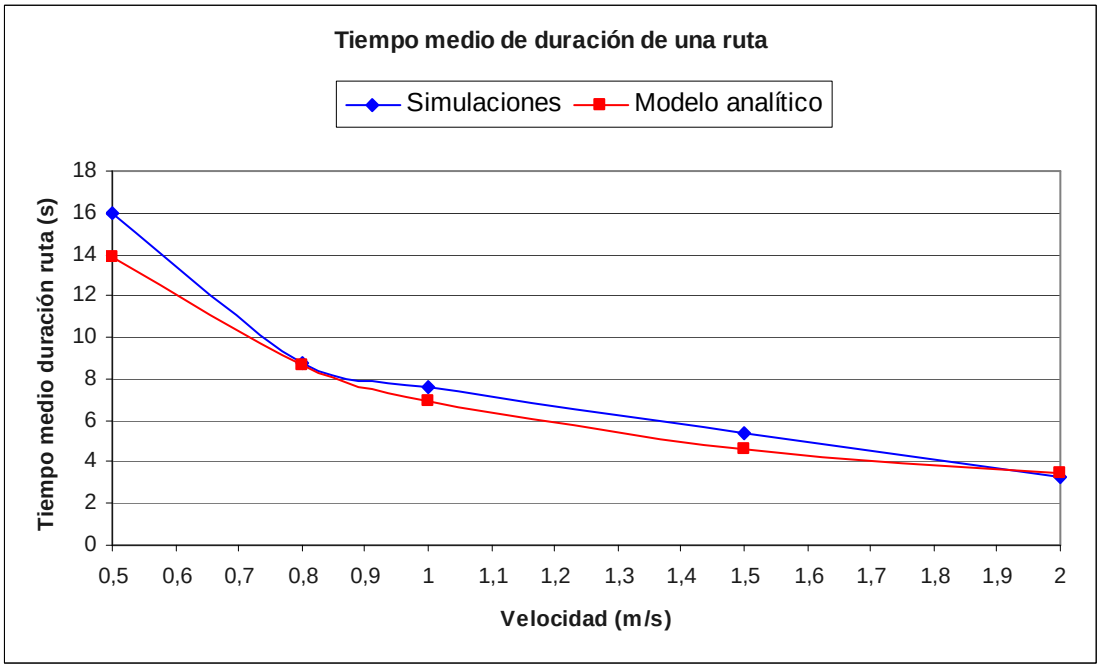

Figura 8-21 Duración media de una ruta, caso V

Antes que nada, en este ejemplo hay que destacar que el modelo analítico nos aporta unos resultados idénticos a los ya vistos dos casos atrás. En ese caso los parámetros usados eran los mismos que en éste pero con 100 nodos en lugar de los 200 nodos de este caso. Esto significa que, en determinados escenarios (como éste), el número de nodos no afecta en absoluto a los

resultados. Se puede observar como los resultados de las simulaciones también refuerzan esta teoría ya que los resultados obtenidos se asimilan en gran medida a los resultados dados por el modelo analítico.

8.1.6. Evaluación de prestaciones. Caso VI

En la siguiente tabla se pueden ver los parámetros escogidos para el siguiente caso:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	200	800	150	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Simulaciones	Modelo analítico con P=0,65
0,5	139,79	-
0,8	99,87	95,89
1	77,06	76,72
1,5	58,25	51,14
2	58,30	38,36

Tabla 8-13 Resultados duración media de una ruta, caso VI

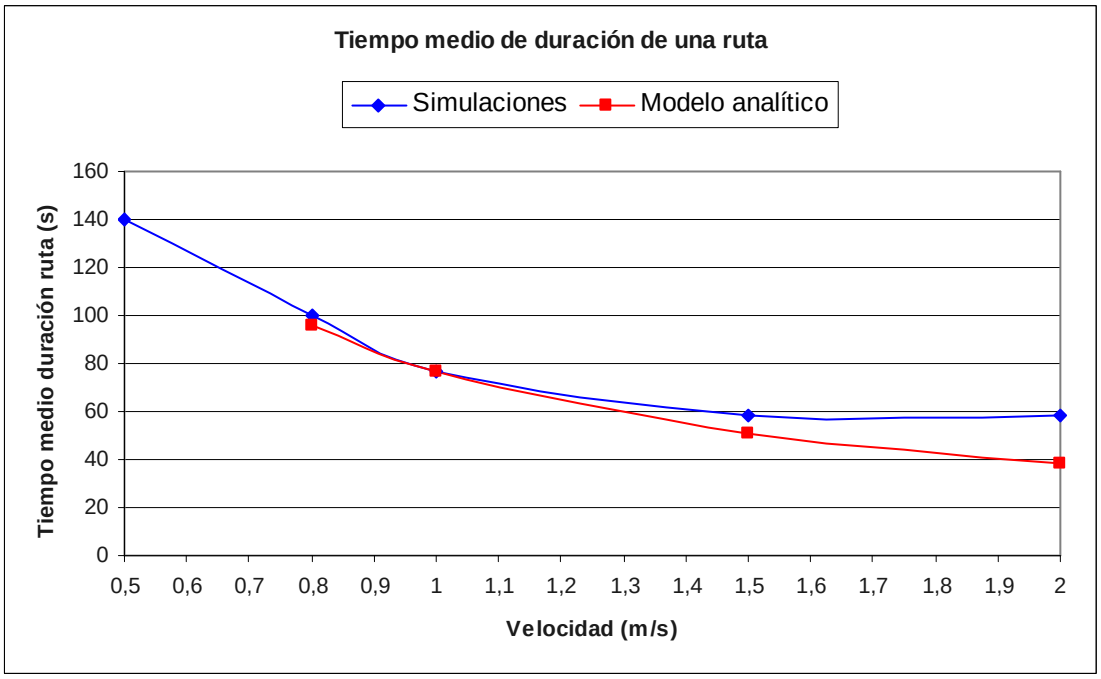

Figura 8-22 Duración media de una ruta, caso VI

En este caso, igual que en el anterior, nos encontramos con unos resultados del modelo analítico idénticos a los de un caso previo ya que la única diferencia radica en el número de nodos que ha pasado de 100 a 200. Podemos ver que en general los resultados de las simulaciones se ajustan bastantes a los del modelo analítico. También podemos observar como la implementación del

modelo analítico carece también en este caso de una referencia para una velocidad máxima de los nodos de 0’5 m/s, que como ya se ha comentado anteriormente se debe a que el tiempo de duración de las rutas en dicho escenario es demasiado elevado para dicha implementación. De hecho se puede observar que el tiempo de duración de las rutas en el caso de velocidad mínima es de aproximadamente 140 segundos, lo que coincide con el tiempo de simulación, es decir, que si aumentásemos el tiempo de simulación, también aumentaría este tiempo.

8.2. Estudio del número medio de rutas encontradas y usadas

Junto con el tiempo de duración de las rutas, también se ha contabilizado el número medio de rutas encontradas por el nodo fuente, las rutas que se han usado para la transmisión y además el número medio de rutas encontradas, o que pasan por un nodo cualquiera que hará función de nodo intermedio. No existe un modelo analítico conocido con el cual comparar estos resultados, pero estos datos son útiles para entender qué sucede en las simulaciones y también de cara a poder proponer nuevos modelos analíticos o perfeccionar los ya existentes. De hecho el número medio de rutas encontradas y usadas por el nodo fuente es un dato relativamente sencillo de obtener una vez se ha obtenido el tiempo medio de duración de las rutas a partir del mismo filtro AWK utilizado. Para contabilizar el número medio de rutas que pasan por un nodo intermedio se ha tenido que añadir código más específico al filtro de AWK.

Como estos resultados no son el objetivo principal de este proyecto, no se comentarán todos los resultados obtenidos, sólo algunos casos más significativos.

A continuación vemos en la tabla los parámetros escogidos para el primer caso:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	30	0	0'5, 0’8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Número medio de rutas encontradas	Número medio de rutas encontradas nodo intermedio	Número medio de rutas usadas
0,5	1,3	0,0	1,0
0,8	2,5	0,4	1,0
1	2,2	0,0	1,3
1,5	3,5	0,2	1,7
2	3,5	0,2	2,0

Tabla 8-14 Resultados número medio de rutas, caso I

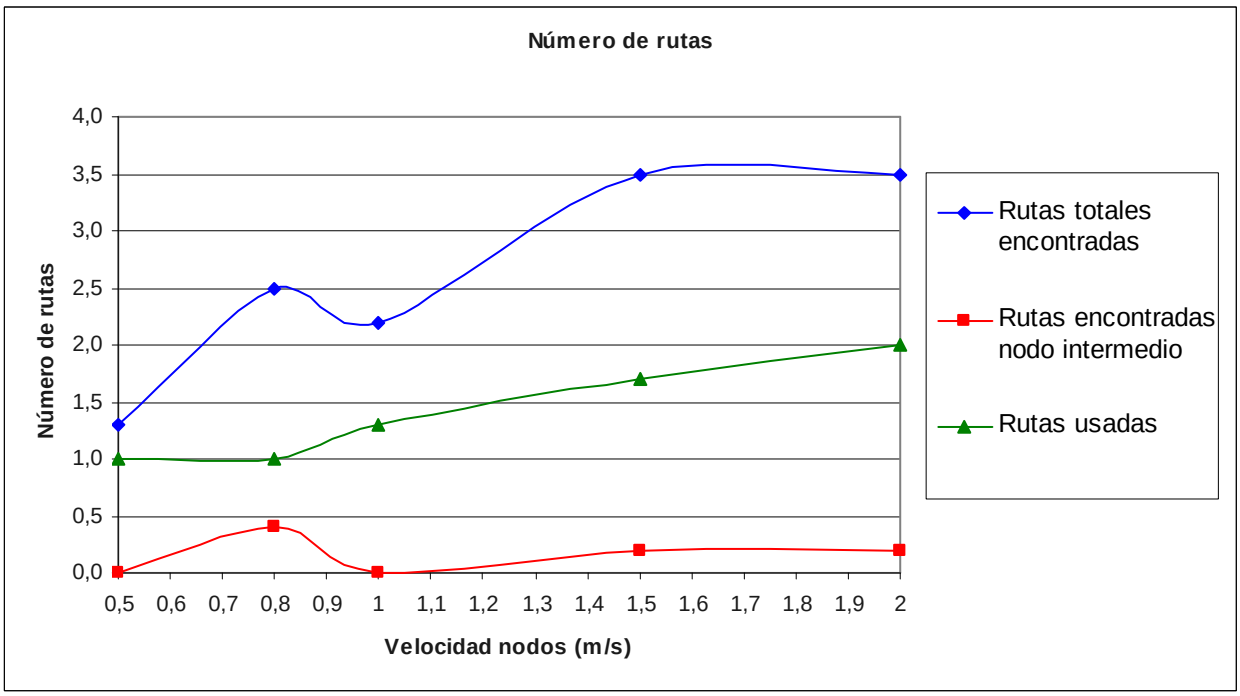

Figura 8-23 Número medio de rutas, caso I

En la gráfica anterior podemos observar distintos fenómenos. Para empezar, en cuánto al número medio de rutas totales vemos que en cada simulación se encuentran pocas rutas y que a medida que aumenta la velocidad aumenta el número medio de rutas encontradas. Esto se debe a que en este caso hay pocos nodos, baja densidad de nodos, y la cobertura es pequeña por lo tanto en general es difícil encontrar una ruta, por lo tanto cuanto más se muevan los nodos más posibilidades que se encuentre alguna ruta.

Por lo que se refiere al número medio de rutas encontradas que pasen por un nodo intermedio dado, vemos como la cantidad es prácticamente cero. Este resultado es muy lógico ya que este número debe ser un pequeño porcentaje del número de rutas totales, aunque al tener un número limitado de simulaciones este porcentaje puede ser bastante aleatorio y por lo tanto es lógico no apreciar mucha variación.

Las rutas usadas para transmitir son muy pocas e inferiores al número medio de rutas encontradas como es lógico. También la tendencia es a aumentar a medida que aumenta la velocidad y es que estos dos datos, número de rutas encontradas y número de rutas usadas, están muy relacionados.

En la tabla siguiente se presentan los parámetros usados en el siguiente caso:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	75	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Número medio de rutas encontradas	Número medio de rutas encontradas nodo intermedio	Número medio de rutas usadas
0,5	41,6	0,2	8,4
0,8	72,8	1,0	13,0
1	203,8	10,2	23,4
1,5	372,6	38,2	36,0
2	323,0	12,6	35,4

Tabla 8-15 Resultados número medio de rutas, caso II

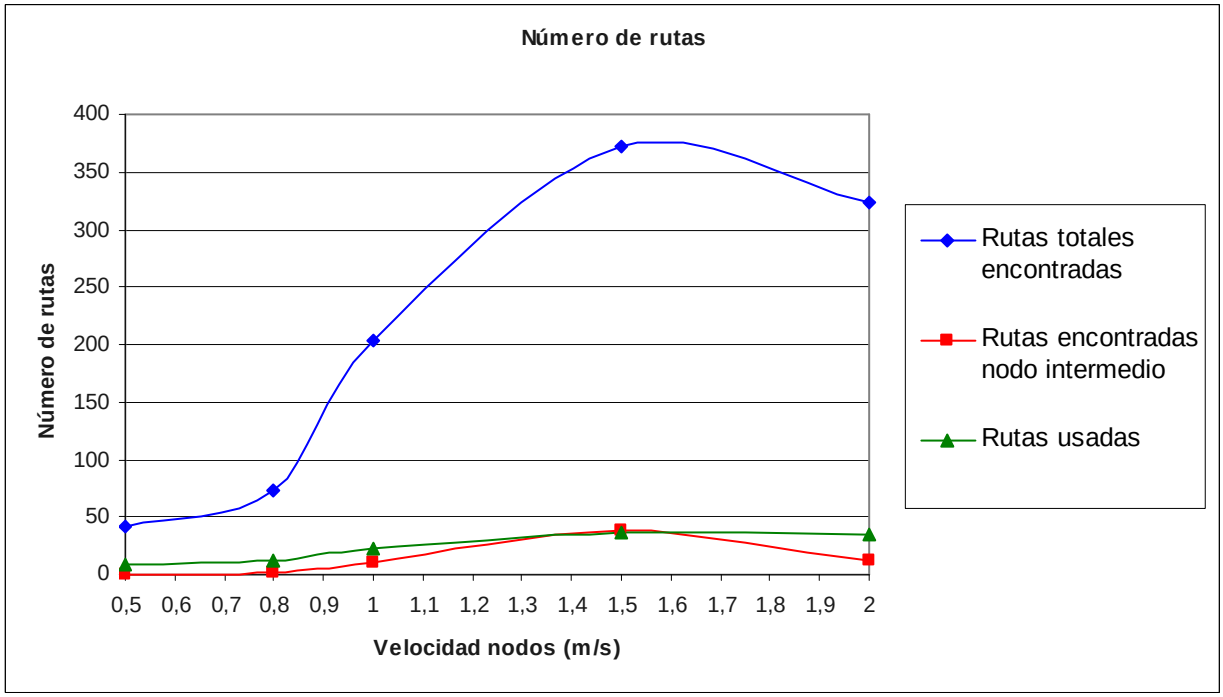

Figura 8-24 Número medio de rutas, caso II

El primer dato a destacar de la gráfica anterior es que el número medio de rutas encontradas ha aumentado mucho respecto al caso anterior. Este aumento considerable se debe a que ahora el número de nodos simulados es de 100 mientras que antes era de 20, así como la cobertura que ha pasado de 30 a 75 metros. Se puede observar que la tendencia del número medio de rutas es a aumentar a medida que aumenta la velocidad de los nodos lo cual coincide con lo explicado en el caso anterior. De todas formas, tampoco es un crecimiento claro y es que como ya se ha dicho el número de simulaciones hechas para obtener estos resultados es limitado.

En cuanto al número medio de rutas que pasan por un nodo intermedio dado, observamos una tendencia también a aumentar aunque como ya se ha dicho antes esta tendencia tiene una componente bastante aleatoria ya que la posición del nodo en cuestión es importante para obtener este resultado y esta posición es completamente aleatoria.

En último lugar, se observa como el número medio de rutas usadas crece de manera similar al número medio de rutas totales, así como que la diferencia entre estos dos parámetros ha crecido de manera considerable, lo cual se debe fundamentalmente al crecimiento del número de nodos.

En la siguiente tabla se recogen los parámetros utilizados en el siguiente caso:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	200	800	75	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Número medio de rutas encontradas	Número medio de rutas encontradas nodo intermedio	Número medio de rutas usadas
0,5	73,0	2,4	10,6
0,8	168,8	16,0	15,6
1	156,4	6,4	18,0
1,5	327,8	8,6	25,0
2	574,2	40,2	37,2

Tabla 8-16 Resultados número medio de rutas, caso III

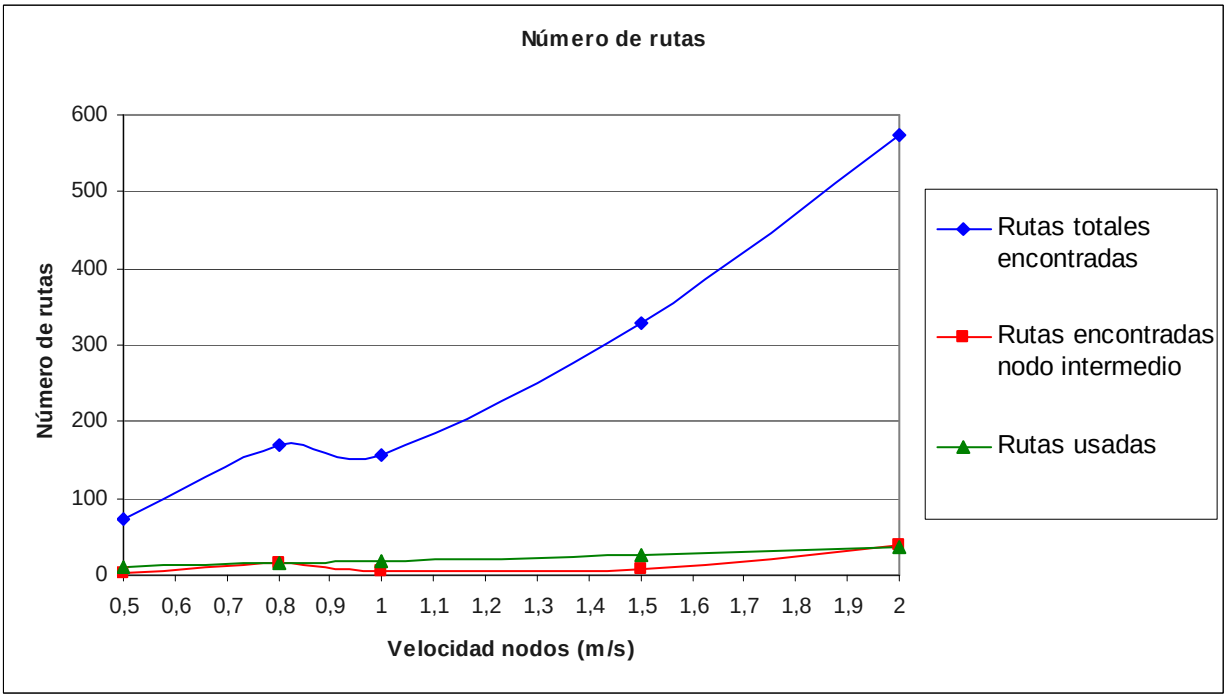

Figura 8-25 Número medio de rutas, caso III

En la gráfica anterior vemos como el número medio de rutas totales ha aumentado ligeramente respecto al caso anterior, esto se debe a que el número de nodos es aún mayor. Pero de hecho este crecimiento no es comparable al del número de nodos que se ha doblado. Vemos también como se mantiene la tendencia a aumentar a medida que aumenta la velocidad.

Comprobamos, una vez más, como el número medio de rutas que pasan por un nodo intermedio dado tiene una tendencia creciente, respecto a la velocidad de los nodos, y que mantiene una fuerte componente aleatoria.

Por lo que se refiere al número medio de rutas vemos que se asimilan mucho a las del caso anterior, esto coincide con los resultados del apartado anterior, dónde se comenta que al aumentar el número de nodos, en este escenario, el tiempo medio de duración de las rutas no aumenta. La explicación es clara ya que el tiempo medio de duración de las rutas y el número medio de rutas usadas son datos íntimamente relacionados.

En último lugar, se observa como el número medio de rutas usadas crece de manera similar al número de rutas totales, así como que la diferencia entre estos dos parámetros ha crecido de manera

8.3. Estudio del *throughput*

Como en el apartado anterior, los resultados de este apartado no son el objetivo principal de este proyecto, por lo tanto sólo se citan los resultados obtenidos en algunos casos. Antes de proseguir es importante definir que se entiende como *throughput* ya que éste se usa en muchas ocasiones con distintos significados. En este proyecto se entiende el *throughput* como la cantidad de información (paquetes) recibida en el nodo destino entre la cantidad de información (paquetes) enviados por el nodo fuente. Como el nodo fuente transmite 1 Mbps, el *throughput* será como máximo de 1 Mbps. De esta manera se puede evaluar en que grado o con que eficiencia funciona la transmisión entre el nodo fuente y el nodo destino.

En el primer caso, los parámetros utilizados han sido los que se muestran en la tabla siguiente:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	30	0	0'5, 0’8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)
0,5	0,223
0,8	0,202
1	0,241
1,5	0,122
2	0,154

Tabla 8-17 Resultados *throughput*, caso I

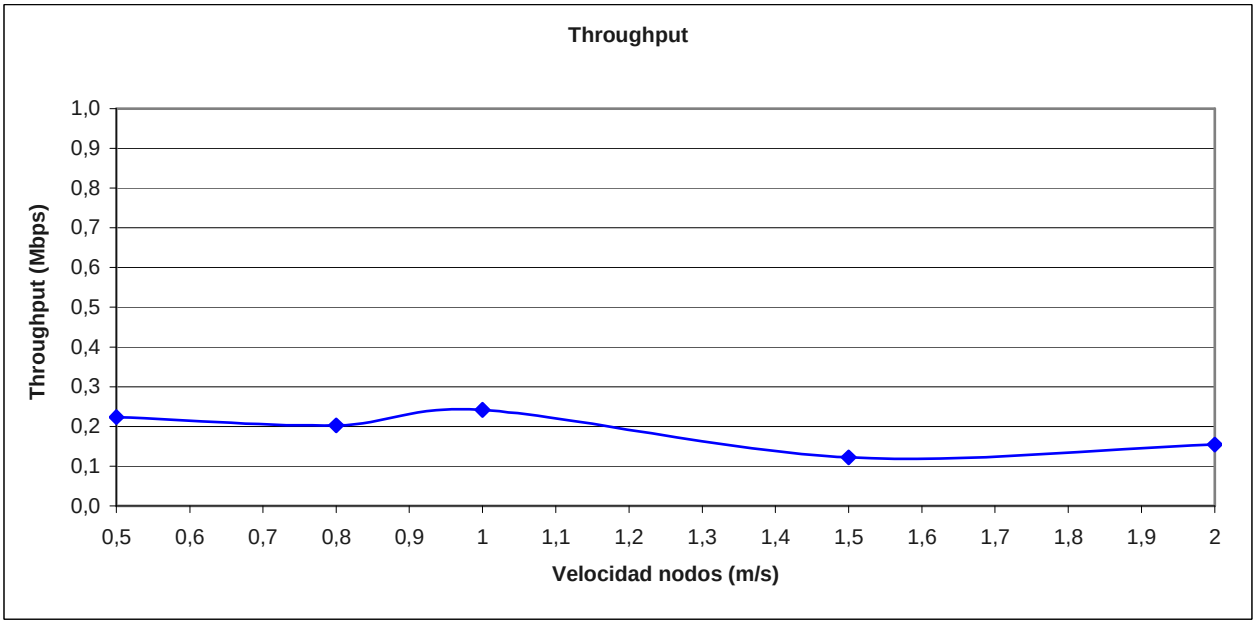

Figura 8-26 *Throughput*, caso I

En la gráfica anterior se puede observar como el *throughput* se comporta de una forma más bien estable en función de la velocidad de los nodos. Esto parece lógico ya que el hecho de que los nodos tengan poca o mucha movilidad afecta a la transmisión provocando cambios de rutas pero en global el tiempo de transmisión acaba siendo el mismo. Seguramente aumentando el número de simulaciones obtendríamos una gráfica más plana. Por lo tanto, lo más destacable de estos resultados es que el *throughput* en este escenario se encuentra alrededor de 0'2 Mbps lo cual reafirma los resultados anteriores de este caso en los que se ha comentado que el número de rutas era muy bajo. En este caso la pérdida de paquetes se debe a que no se pueden encontrar rutas para enviar toda la información, es por eso que el *throughput* tiene un valor tan bajo.

En el segundo caso, los parámetros escogidos son los que se indican en la tabla siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	75	0	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)
0,5	0,843
0,8	0,805
1	0,842
1,5	0,595
2	0,766

Tabla 8-18 Resultados *throughput*, caso II

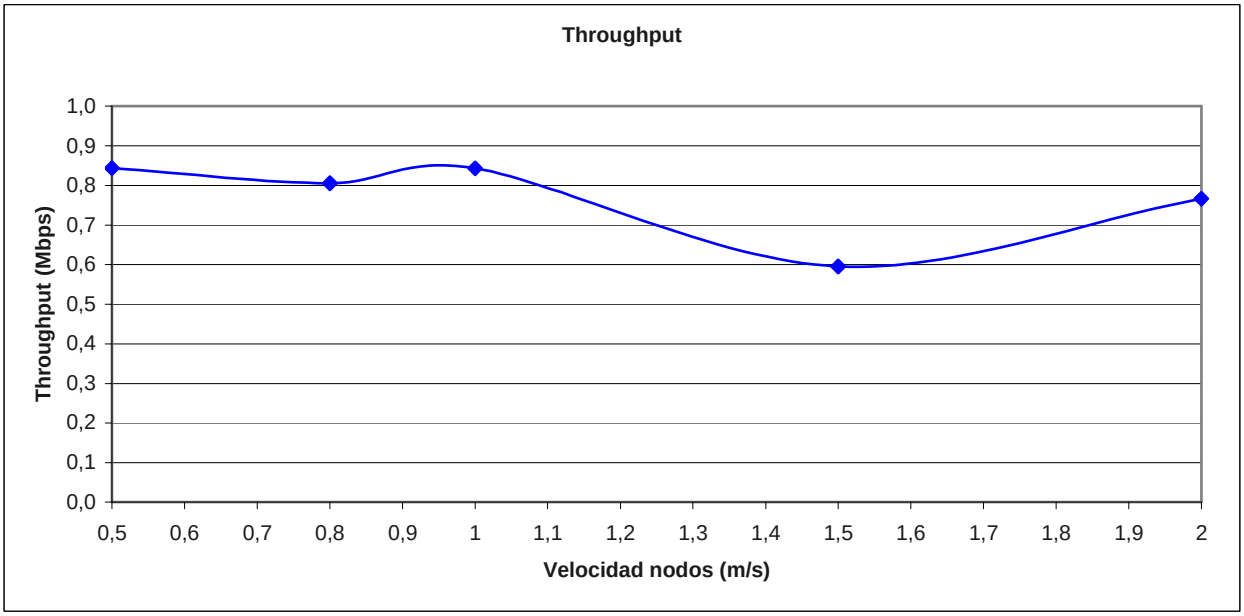

Figura 8-27 *Throughput*, caso II

En la gráfica anterior volvemos a ver un comportamiento más o menos estable del *throughput* en función de la velocidad de los nodos. De todas formas, en este caso se puede observar una ligera tendencia a la baja del *throughput*, este fenómeno se puede explicar debido al número de rutas usadas ya que a medida que aumenta el número de rutas, es más fácil que algunos cambios sean provocados por pérdidas las cuales repercuten en el *throughput*. Pero para ver una gran influencia de este fenómeno en el *throughput*, el número de rutas usadas debería variar muchísimo en función de la velocidad y no es el caso. En este caso, como ya se ha comentado anteriormente, hay muchas rutas disponibles lo que da un *throughput* medio aproximado de 0'75 Mbps. Se puede observar en este escenario con mayor número de nodos y mayor radio de cobertura que el *throughput* incrementa notablemente.

8.4. Resultados con dos fuentes interferentes

Otro escenario interesante para extraer resultados es cuando hay transmisiones que interfieren el tráfico que estamos analizando. Es por ello que se dedica este apartado a estudiar el caso en que además del nodo fuente y el nodo destino hay otros dos nodos fuente que envían tráfico interferente a otros dos nodos destino. En este caso el comportamiento de la transmisión estudiada sufrirá ciertas variaciones en función de la naturaleza de cada escenario. Se podría aumentar todavía más el número de interferentes, pero con dos interferentes es suficiente para observar cambios significativos.

Los resultados que se presentan a continuación corresponden a los mismos parámetros estudiados en el caso en que no había ningún tipo de tráfico interferente.

8.4.1. Estudio del tiempo medio de duración de una ruta

Igual que en el caso en el que no había fuentes interferentes, empezamos por ver el tiempo de duración de las rutas usadas en la transmisión principal. Cabe destacar que no se dispone de un modelo analítico para comparar estos resultados, si bien, éstos pueden servir para establecer un nuevo modelo analítico más adelante. Los resultados obtenidos mediante las simulaciones, se comparan en este apartado con los resultados obtenidos sin tráfico interferente.

8.4.1.1. Evaluación de prestaciones. Caso I

En la tabla siguiente se observan los parámetros escogidos para el caso siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/m²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	30	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Con dos interferentes	Sin interferentes
0,5	34,65	31,05
0,8	23,43	28,15
1	20,46	25,77
1,5	12,97	9,87
2	8,47	10,57

Tabla 8-19 Resultados duración media de una ruta con tráfico interferente y sin él, caso I

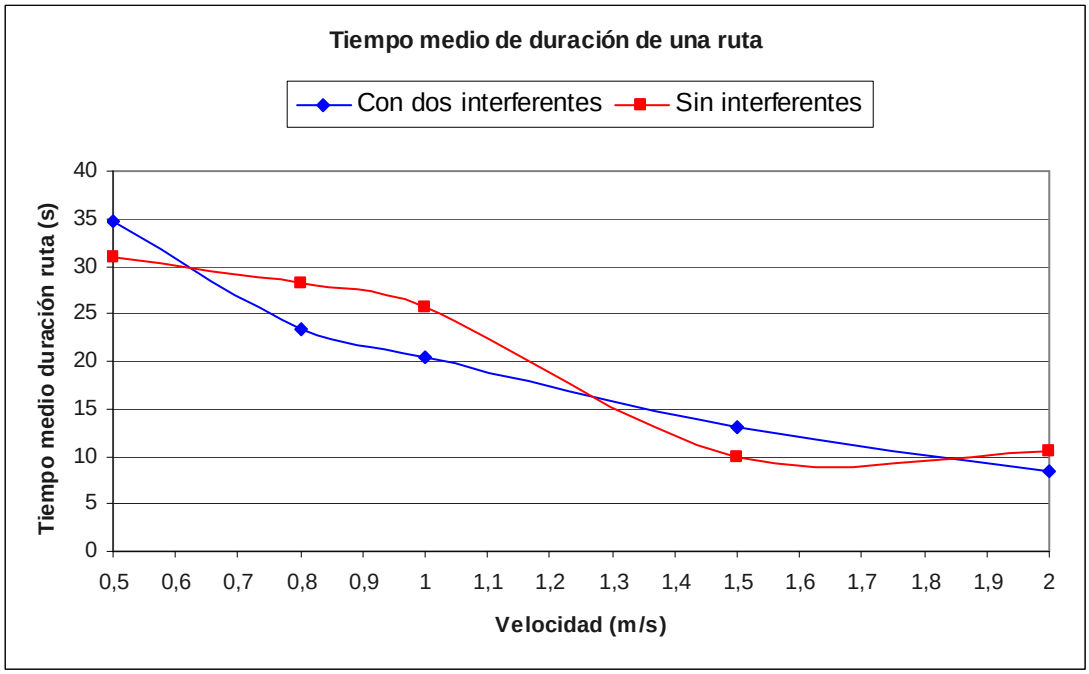

Figura 8-28 Comparación duración media de una ruta con tráfico interferente y sin él, caso I

A partir de la gráfica anterior podemos ver como en este primer caso la diferencia entre tener o no tener tráfico interferente es más bien pequeña por no decir casi inexistente. Las dos gráficas tienen la misma tendencia y valores bastante próximos, más teniendo en cuenta que se trata de resultados obtenidos de un número limitado de simulaciones. La razón por la cual el tráfico interferente provoca poca variación en la duración de las rutas es sencilla, y es que en este caso el número de rutas es muy bajo debido a la poca cobertura y los pocos nodos que hay, por lo tanto como ya es difícil la comunicación entre dos nodos, también lo es que las interferencias molesten demasiado.

8.4.1.2. Evaluación de prestaciones. Caso II

Los parámetros usados en el caso siguiente son los que se pueden ver en la tabla siguiente:

Escenario	Dimensiones (m^2)	Número de nodos	Densidad (nº nodos/km^2)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	50	2	0'5, 0’8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Con dos interferentes	Sin interferentes
0,5	34,00	37,08
0,8	11,91	28,18
1	10,18	23,81
1,5	11,48	14,46
2	18,30	11,15

Tabla 8-20 Resultados duración media de una ruta con tráfico interferente y sin él, caso II

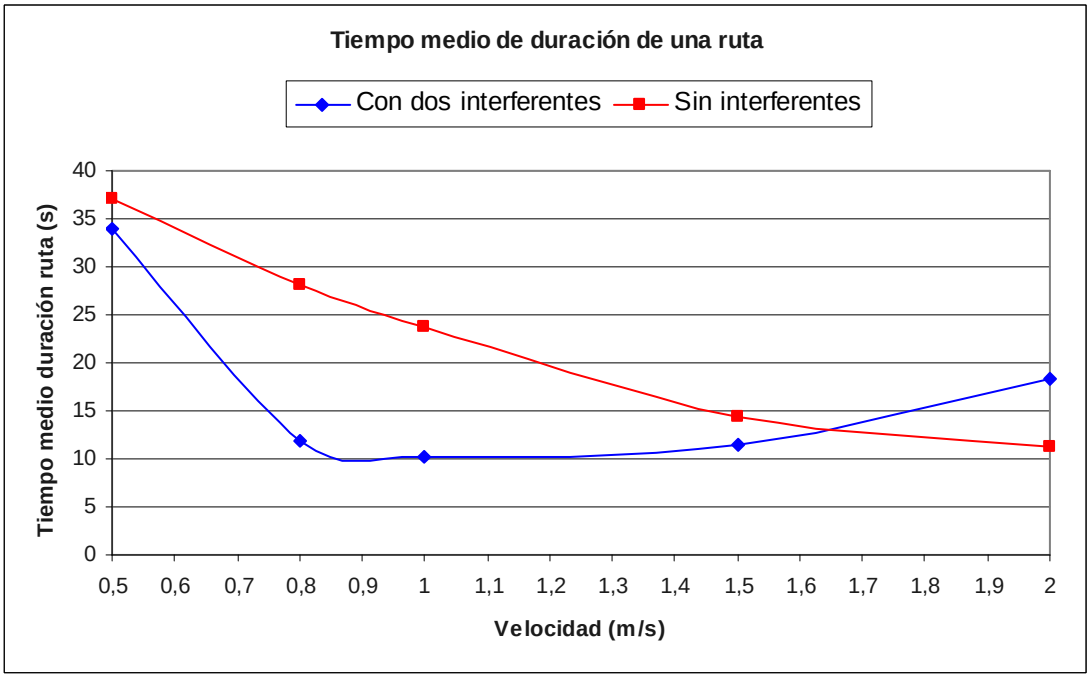

Figura 8-29 Comparación duración media de una ruta con tráfico interferente y sin él, caso II

En este caso, como vemos en la gráfica anterior, las interferencias juegan un papel bastante importante. En general se puede intuir una ligera tendencia a parecerse entre ambas gráficas, aunque esta claro que las interferencias distorsionan en gran medida esta tendencia. Podemos concluir por lo tanto que en este caso las interferencias provocan más cambios de rutas y esos cambios conllevan que disminuya la duración de las rutas, pero estas interferencias no afectan siempre sino de una manera bastante aleatoria.

8.4.1.3. Evaluación de prestaciones. Caso III

En la siguiente tabla se pueden ver los parámetros usados en el caso siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	150	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Con dos interferentes	Sin interferentes
0,5	4,79	13,98
0,8	4,10	9,39
1	4,04	5,69
1,5	3,67	3,68
2	4,14	3,47

Tabla 8-21 Resultados duración media de una ruta con tráfico interferente y sin él, caso III

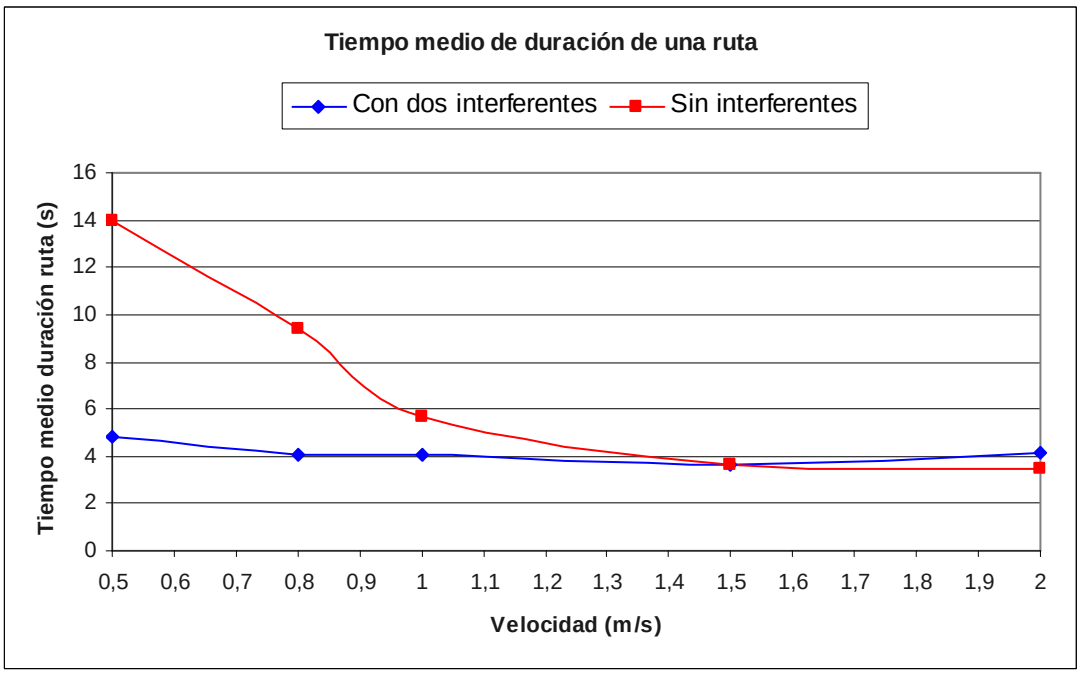

Figura 8-30 Comparación duración media de una ruta con tráfico interferente y sin él, caso III

Según se observa en la gráfica anterior, las interferencias cobran un papel crucial en este caso. Ya no se observa ningún tipo de semejanza entre ambas gráficas, al haber mucho tráfico interferente, la duración de las rutas es aproximadamente el mismo para cualquier velocidad de los nodos. Aunque probablemente sigue existiendo una tendencia a la baja de la duración de las rutas, ésta queda completamente enmascarada por el tráfico interferente. El hecho de que influya tanto el tráfico interferente se debe a que en este caso hay muchos nodos y las rutas suelen tener varios saltos, y cuanto más saltos tengan las rutas, mayor es la probabilidad de que alguno se vea afectado por el tráfico interferente, que también cuenta con rutas con más saltos.

8.4.1.4. Evaluación de prestaciones. Caso IV

En la tabla siguiente se presentan los parámetros usados en el caso siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	150	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Tiempo medio de duración de una ruta (s)	
	Con dos interferentes	Sin interferentes
0,5	8,19	116,56
0,8	10,50	99,96
1	7,37	77,73
1,5	10,04	53,74
2	7,02	41,12

Tabla 8-22 Resultados duración media de una ruta con tráfico interferente y sin él, caso IV

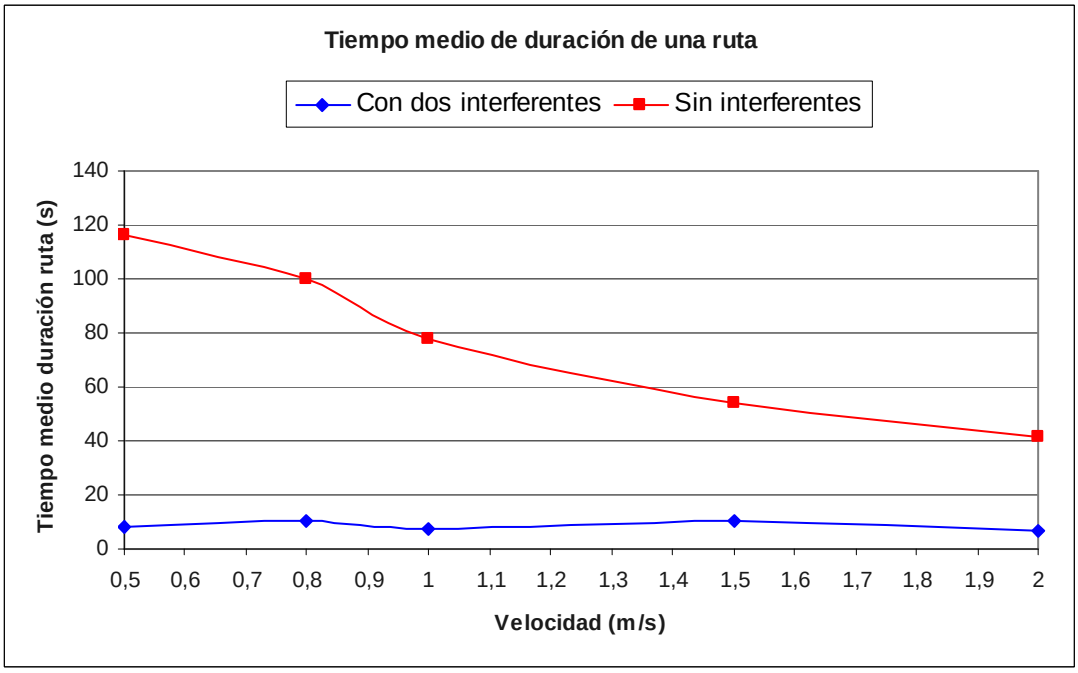

Figura 8-31 Comparación duración media de una ruta con tráfico interferente y sin él, caso IV

En este caso, igual que en el anterior, los resultados se ven claramente afectados por las interferencias. En este caso el número de saltos de las rutas no es tan grande, pero la cobertura si, eso implica que la interferencia de un nodo afecte a muchos más nodos de su alrededor. Como se puede observar la duración de las rutas se mantiene prácticamente estable en función de la velocidad de los nodos.

8.4.2. Estudio del número medio de rutas encontradas y usadas

En el caso de tener tráfico interferente conocer el número medio de rutas no es el objetivo principal, como no lo era cuando no había tráfico interferente. De todas formas, los datos obtenidos pueden ser útiles para su estudio y también para entender que está sucediendo en las simulaciones y los otros resultados. Así que vamos a ver sólo un caso para ayudar a entender lo explicado en el apartado anterior.

En la tabla siguiente se pueden ver los parámetros de este caso, que son los mismos que en el último caso visto:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	150	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	Con dos interferentes			Sin interferentes		
	Número medio de rutas encontradas	Número medio de rutas encontradas nodo intermedio	Número medio de rutas usadas	Número medio de rutas encontradas	Número medio de rutas encontradas nodo intermedio	Número medio de rutas usadas
0,5	200,2	0,2	16,2	8,6	0,0	1,2
0,8	130,2	0,1	13,0	8,2	0,4	1,4
1	136,1	1,7	18,7	8,2	0,0	1,8
1,5	133,5	1,0	13,6	17,2	3,0	2,6
2	221,7	2,9	19,2	15,6	0,8	3,4

Tabla 8-23 Resultados número medio de rutas con tráfico interferente y sin él, caso I

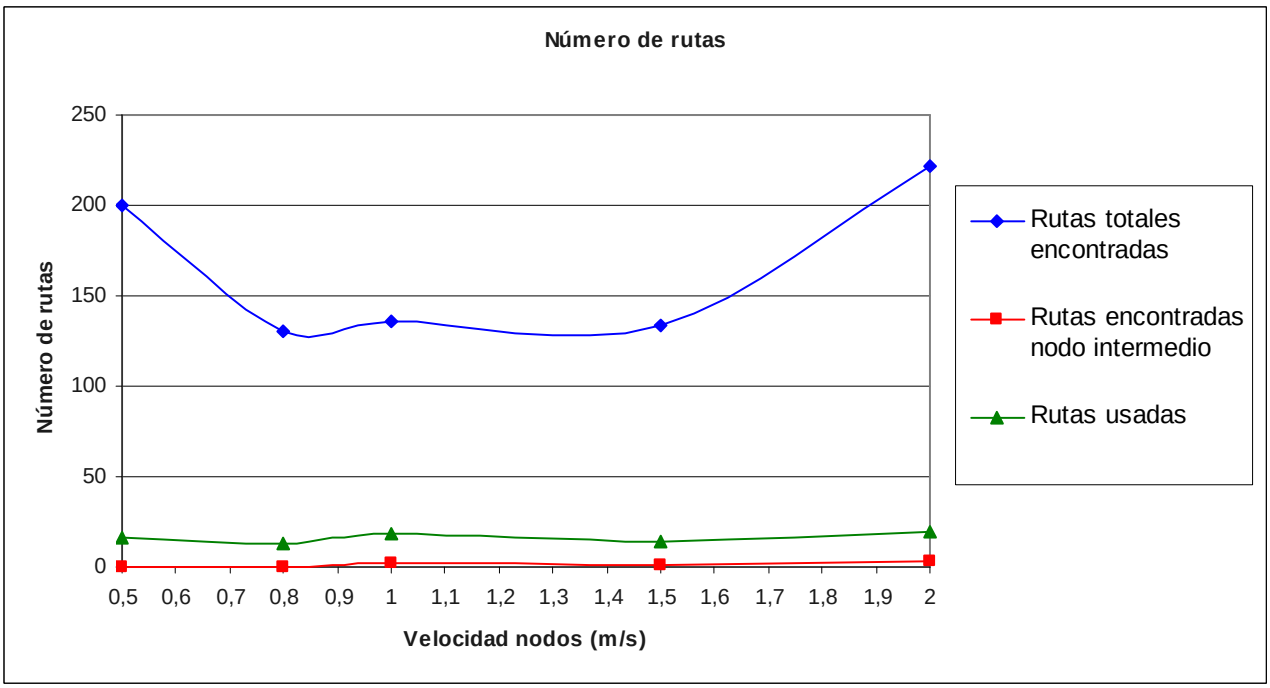

Figura 8-32 Número medio de rutas con tráfico interferente, caso I

Como se observa fácilmente los resultados no parecen seguir ningún patrón en función de la velocidad. Esto es debido, como ya se ha dicho anteriormente, a que las interferencias afectan de manera sustancial. Además en este caso se incluye en la tabla de resultados los valores obtenidos en el mismo escenario pero sin interferentes para compararlos. Es aquí dónde se observa claramente la diferencia entre que haya o no tráfico interferente, el tráfico interferente provoca que el número medio de rutas usadas y encontradas se multiplique aproximadamente por

un factor 10, lo cual provoca que la duración de las rutas disminuya drásticamente como ya se ha comentado.

8.4.3. Estudio del *throughput*

Por último también se analizan los resultados obtenidos en cuanto al *throughput* se refiere. Estos resultados no son el objetivo principal pero tienen una gran utilidad entre otras cosas de cara a ayudar a entender lo que sucede con la duración de las rutas. Por eso en este apartado se compara el *throughput* obtenido en los 4 casos realizados con tráfico interferente, con los mismos casos sin interferencias.

Se observa en la tabla siguiente los parámetros usados en el primer caso:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	30	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)	
	Con dos interferentes	Sin interferentes
0,5	0,194	0,223
0,8	0,404	0,202
1	0,288	0,241
1,5	0,221	0,122
2	0,168	0,154

Tabla 8-24 Resultados *throughput* con tráfico interferente y sin él, caso I

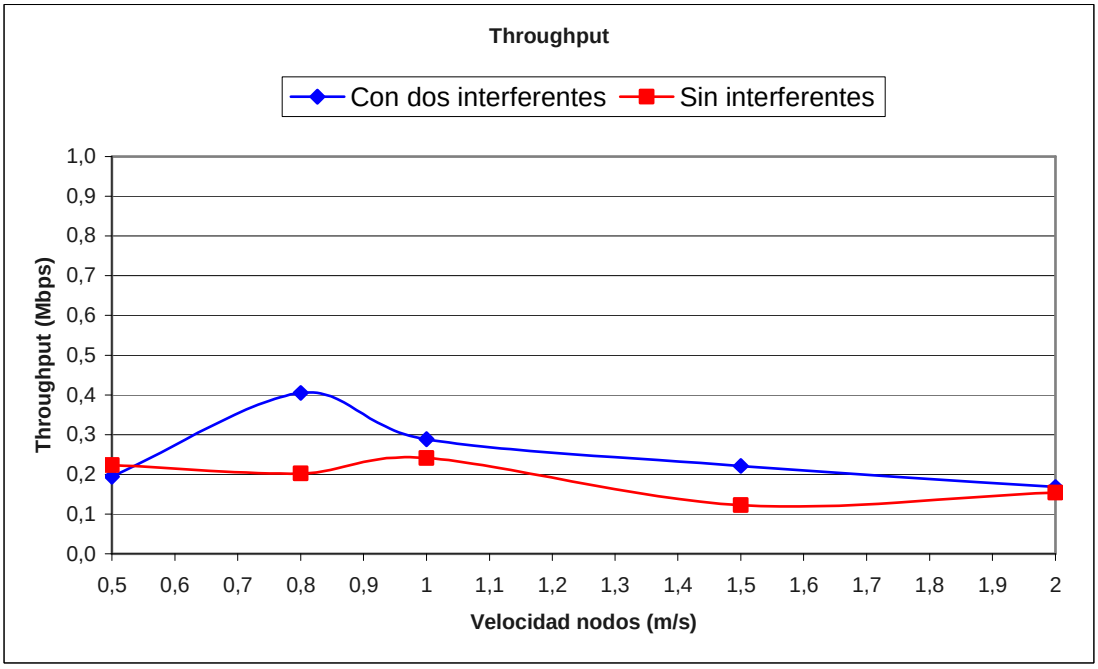

Figura 8-33 Comparación *throughput* con tráfico interferente y sin él, caso I

A raíz de la gráfica anterior se observa que el *throughput* en este caso no ha sufrido grandes variaciones respecto a cuando no había tráfico interferente. Si bien es cierto, que existe un cierto aumento del *throughput* en algún caso, éste se debe simplemente al carácter aleatorio de las simulaciones, la tónica general es que el *throughput* se mantiene alrededor de 0'2 o 0'25 Mbps. Esto reafirma los resultados obtenidos en cuanto a duración de las rutas dónde se ha comentado que en este caso las interferencias no afectan demasiado.

El segundo caso se ha hecho con los parámetros presentados en la tabla siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Pequeño	200x200	20	500	50	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)	
	Con dos interferentes	Sin interferentes
0,5	0,654	0,677
0,8	0,494	0,688
1	0,381	0,888
1,5	0,629	0,728
2	0,752	0,923

Tabla 8-25 Resultados *throughput* con tráfico interferente y sin él, caso II

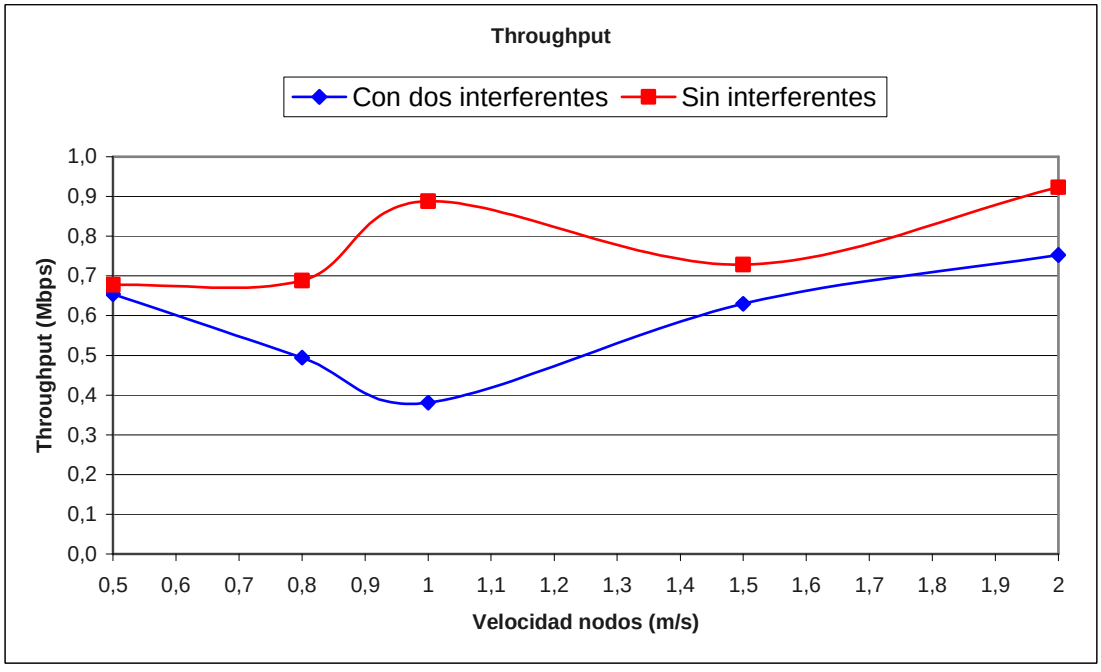

Figura 8-34 Comparación *throughput* con tráfico interferente y sin él, caso II

A partir de la gráfica anterior se puede observar como en este caso las interferencias ya afectan en cierto grado. La diferencia no es abismal pero si sustancial, pasando el *throughput* de estar alrededor de 0'8 Mbps sin interferencias a estar sobre 0'6 Mbps con tráfico interferente. También

se puede observar como la tónica del *throughput* sigue siendo más o menos estable en función de la velocidad igual como sucedía cuando no había interferencias.

En la tabla siguiente se presentan los parámetros escogidos para el caso siguiente:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	75	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)	
	Con dos interferentes	Sin interferentes
0,5	0,539	0,843
0,8	0,344	0,805
1	0,295	0,842
1,5	0,403	0,595
2	0,404	0,766

Tabla 8-26 Resultados *throughput* con tráfico interferente y sin él, caso III

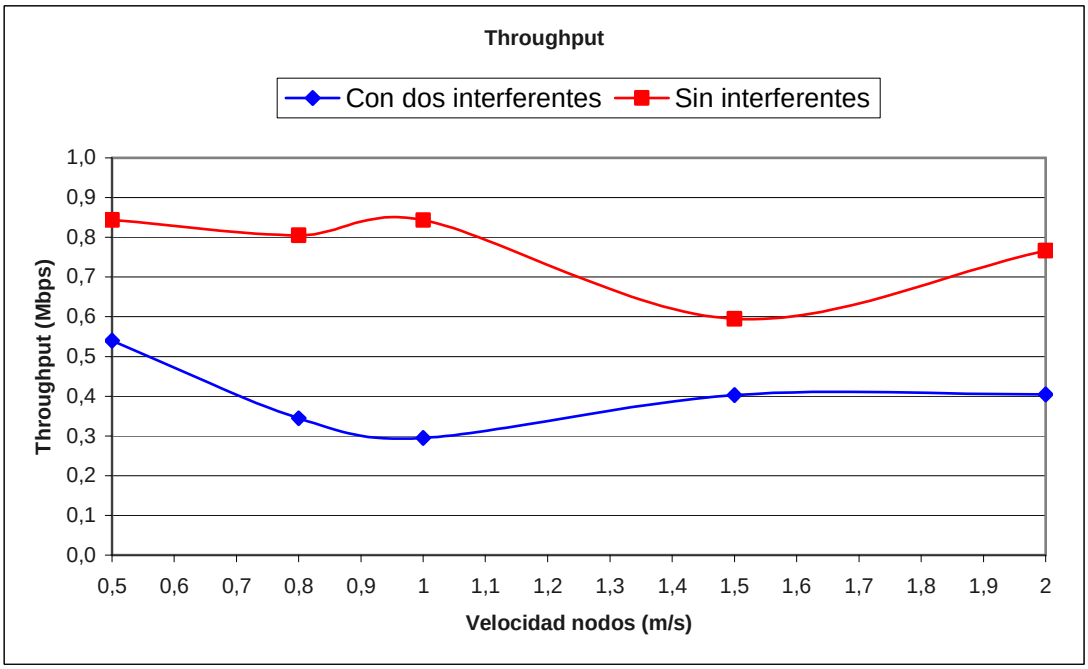

Figura 8-35 Comparación *throughput* con tráfico interferente y sin él, caso III

En este caso, observando la gráfica anterior, se deduce que las interferencias tienen un efecto muy importante en las simulaciones. El *throughput* ha pasado de aproximadamente 0'75 Mbps a tan sólo 0'4 Mbps. También se observa una vez más la tónica de independencia entre el *throughput* y la velocidad de los nodos.

El último caso se ha realizado con los parámetros que se presentan en la siguiente tabla:

Escenario	Dimensiones (m²)	Número de nodos	Densidad (nº nodos/km²)	Cobertura antenas (m)	Nodos interferentes	Velocidad máxima de los nodos
Mediano	500x500	100	400	150	2	0'5, 0'8, 1, 1'5, 2

Los resultados obtenidos son los siguientes:

Velocidad (m/s)	*Throughput* (Mbps)	
	Con dos interferentes	Sin interferentes
0,5	0,730	0,999
0,8	0,701	1,000
1	0,702	0,999
1,5	0,814	0,999
2	0,744	0,999

Tabla 8-27 Resultados *throughput* con tráfico interferente y sin él, caso IV

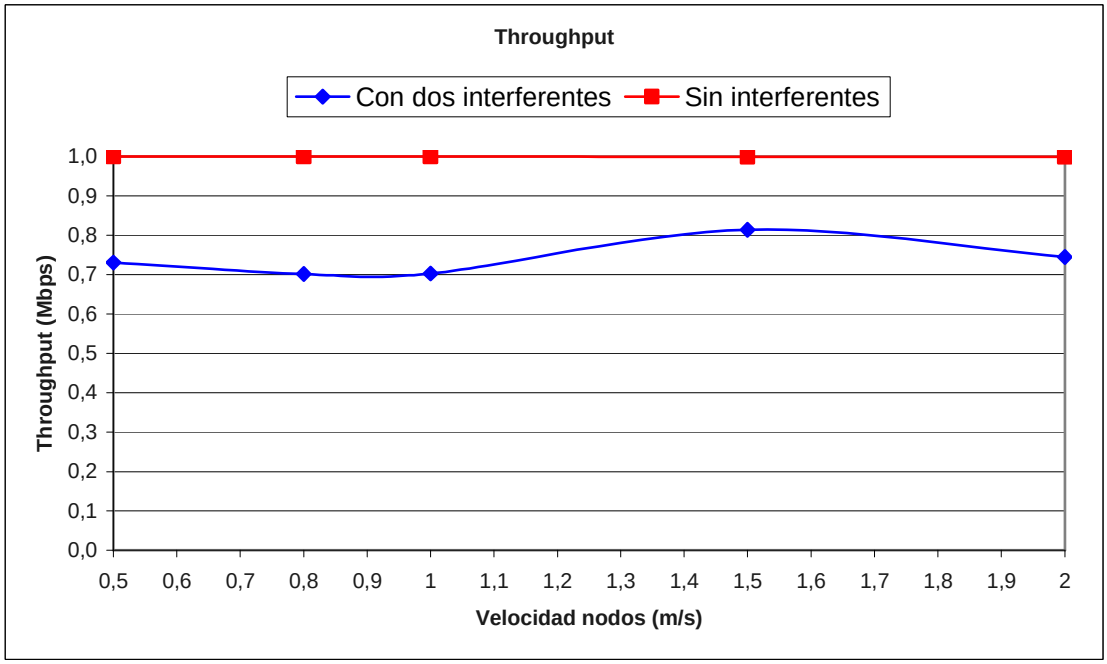

Figura 8-36 Comparación *throughput* con tráfico interferente y sin él, caso IV

En la gráfica anterior se puede observar una vez más la clara influencia de las interferencias en los resultados de las simulaciones. En este caso se ha pasado a un *throughput* prácticamente 1 Mbps a tan solo 0'75 Mbps sólo por el efecto del tráfico interferente.

Capítulo 9. Conclusiones y líneas futuras

En este Proyecto Final de Carrera (PFC) se han estudiado las principales características de las redes MANET (*Mobile Ad Hoc Networks*) y de los protocolos de encaminamiento más usados en estas redes. Concretamente, se ha estudiado en profundidad el protocolo DSR (*Dynamic Source Route*) [8].

También se han estudiado unos modelos analíticos para evaluar la probabilidad de error en los caminos de transmisión. Dichos modelos analíticos han sido desarrollados previamente en el marco de dos tesis doctorales, en una de las cuales [11] he colaborado activamente durante el desarrollo de mi PFC. Dichos modelos analíticos han podido ser validados a partir de su comparación con resultados de simulación obtenidos en este PFC.

A continuación, se ha explicado el funcionamiento de las herramientas utilizadas para realizar las simulaciones y extraer los resultados. El simulador utilizado es NS-2 [15], por ser uno de los más ampliamente utilizados en temas de investigación sobre redes de comunicaciones.

Finalmente en el capítulo 8 se han presentado los resultados obtenidos mediante simulaciones y se han comparado con los obtenidos mediante modelos matemáticos, presentando un buen ajuste y validando por lo tanto la herramienta analítica desarrollada en [11].

9.1. Conclusiones generales

Después de haber estudiado con profundidad la teoría asociada al PFC (el funcionamiento de las redes MANET, el protocolo de encaminamiento DSR, los escenarios a simular), y estudiar el funcionamiento de las herramientas de simulación utilizadas en el proyecto, hemos obtenido los resultados mediante simulaciones y han sido comparados con los resultados proporcionados por el modelo analítico. A continuación vamos a extraer unas conclusiones generales.

Como se ha comentado en el capítulo de resultados (capítulo 8) el parámetro estudiado más extensamente ha sido la duración media de las rutas, el cual se ha obtenido mediante simulaciones y también se ha comparado con los resultados obtenidos a partir del modelo analítico. Además también se han estudiado otros parámetros, como el número medio de rutas encontradas por el protocolo DSR y aquellas que han sido efectivamente usadas, así como las que se encuentran que pasan por un nodo cualquiera. Estos resultados nos han ayudado a comprender mejor lo que está pasando en cada caso y sirven también para en un futuro poder desarrollar nuevos modelos matemáticos.

Después de haber visto los resultados en muchos escenarios distintos, variando: la velocidad de los nodos, la densidad de los nodos (el número de nodos y las dimensiones) y el radio de cobertura de la antenas, el tráfico interferente; podemos llegar a dos conclusiones importantes:

- A mayor velocidad de los nodos mayor número de rutas y menor duración de éstas.
- El radio de cobertura de los nodos es el parámetro que más afecta a la duración de las rutas, siendo poco relevante el número de nodos.

La primera conclusión se puede observar rápidamente en el comportamiento de la mayoría de las gráficas del capítulo 8. En la duración de las rutas se observa como a medida que la velocidad de los nodos aumenta, la duración de las rutas, en media, disminuye. Esto se debe a que cuanto mayor es la velocidad de los nodos, mayor es la probabilidad de que la ruta activa se rompa. Por este mismo motivo, cuanto mayor es la velocidad de los nodos mayor es el número de rutas usadas y encontradas, puesto que cada vez que se rompe una ruta, si no hay ninguna en caché disponible, se debe iniciar otra vez el mecanismo de descubrimiento de rutas (*route discovery*) y por lo tanto más rutas se acaban encontrando.

Centrándonos en la segunda conclusión, podemos ver como en las simulaciones realizadas con exactamente las mismas condiciones pero variando el número de nodos de 100 a 200 y por lo tanto doblando la densidad de nodos, los resultados son prácticamente los mismos. En cambio, variando el radio de cobertura de 75 a 150 metros y manteniendo el resto de parámetros, la duración de las rutas aumenta considerablemente. Esto significa que si se quiere conseguir un

buen funcionamiento y por lo tanto garantizar una calidad de servicio (QoS), los esfuerzos no se deben centrar en aumentar el número de nodos de la red sino en aumentar la cobertura de las antenas.

En cuanto a la afectación que tiene el tráfico interferente en el funcionamiento de las redes MANET podemos decir que es variable. En general, cuantos más saltos tengan las rutas (alta densidad de nodos y baja cobertura) más afectaran las interferencias, puesto que la probabilidad de que un nodo deba retransmitir dos tráficos distintos es mayor. En el caso que eso ocurra es fácil que el nodo se sature y que la calidad se la transmisión empeore considerablemente: pérdida de paquetes, cambios de ruta constantes, etc. En cambio, en casos en que la cobertura sea mayor y el número de saltos menor, es más improbable que se llegue a situaciones de mal funcionamiento. En el caso que la cobertura sea muy baja tampoco afectan demasiado las interferencias aunque en este caso es porqué la transmisión ya es muy complicada de por sí.

Los resultados que se han comparado entre el modelo analítico y los obtenidos mediante simulaciones arrojan otras conclusiones importantes. Para empezar hay que resaltar que los resultados del modelo analítico se asimilan mucho a los obtenidos mediante simulaciones, en concreto, en la mayoría de casos el error en la duración de las rutas es en media entre un 5% y un 10% en cada caso. La única excepción sería el caso con cobertura muy pequeña, de 30 metros, que como ya se ha comentado los resultados no están lo suficientemente promediados al existir pocas rutas, en este caso el error es de un 20%. Además los resultados extraídos mediante el modelo analítico concuerdan con las conclusiones que se han explicado previamente.

9.2. Líneas futuras de trabajo

Una vez dicho todo esto se puede decir que existen muchas líneas en las que seguir trabajando.

Para empezar, todo el trabajo de simulaciones realizado se podría ampliar a otros protocolos, el DSR es uno de los más usados pero no el único, y en este sentido podría ser muy interesante escoger alguno de los protocolos presentados en el capítulo 2 y realizar simulaciones en distintos escenarios para ver qué prestaciones tiene. Junto a esto también se podría realizar un modelado analítico del funcionamiento de otro protocolo y compararlo con las simulaciones.

Por otro lado, sería interesante desarrollar modelos analíticos que caractericen otros parámetros de las redes MANET además del que ya se ha visto de duración de las rutas. Así, se podría estudiar el número de peticiones de rutas, número de rutas encontradas, el tiempo que tarda en encontrarse una nueva ruta, etc.

En general hay que continuar trabajando en el estudio de las redes MANET ya sea mediante modelos analíticos o mediante simulaciones con un objetivo claro que es el de poder mejorar y garantizar la calidad de servicio (QoS) de las redes MANET.

Referencias

[1] R. Ramanathan, On the performance of ad hoc networks with beamforming antenas. ACM MobiHoc, 2001. ISBN:1-58113-428-2.

[2] S. Xu, T. Saadawi. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?. Communications Magazine, IEEE, vol. 39, no. 6, pp. 130-132. June 2001.

[3] S. Xu, T. Saadawi. Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks. The International Journal of Computer and elecommunications Networking, vol. 38, no. 4. March 2002. SSN: 1389-1286.

[4] E.M. Royer, C.K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Wireless Communications, vol. 6, no. 2. April 1999.

[5] K. Petteri. Classification of ad hoc routing protocols. Finish Defence Forces, Naval Academy, Finland 2002.

[6] G. D. Holland. Adaptive models for Mobile Ad Hoc Networks. Department of Computer Science. Collage Station, TX: Texas A&M University. 2004.

[7] G. Anastasi, E. Ancillotti, M. Conti, A. Passarella. TPA: A Transport Protocol for Ad Hoc Networks. Computers and Communications, ISCC 2005. June 2005. ISBN: 0-7695-2373-038G

[8] The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for Ipv4, D. Johnson, Y. Hu , D. Maltz. Internet Draft: http://tools.ietf.org/html/rfc4728, February 2007.

[9] Víctor Carrascal Frías, "Contribution to provide QoS over Mobile Ad Hoc Networks for Video-Streaming Services based on Adaptive Cross-Layer Architecture", PhD Disertation, Advisor: Dra. Mónica Aguilar Igartua, 2nd March 2009, Department of Telematic Engineering, Technical University of Catalonia, Barcelona, Spain. Available in http://sertel.upc.es/tesis.php. NS-2 contributed code available in http://globus.upc.es/~vcarrascal/ns2/.

[10] http://gridnet.upc.es/~maguilar/

[11] Aída Zavala Ayala, “Modelos Analíticos para el diseño y optimización de servicios multimedia sobre redes MANET con QoS” http://gridnet.upc.es/~azavala

[12] L.E. Miller, “Distribution of link distances in a wireless network”, Journal of research of the National Institute of Standards and Technology, Vol. 106, No. 2, pp. 401-212, March-April 2001.

[13] A. B. McDonald, T. Znati, “A path availability model for wireless ad-hoc networks”, IEEE Wireless Communications and Networking Conference (WCNC’99) New Orleans, USA, pp. 35-40, 1999. ISBN: 0-7803-5668-3.

[14] Bai, Fan; Helmy, Ahmed (2006), A Survey of Mobility Models in Wireless Adhoc Networks (Chapter 1 in Wireless Ad-Hoc Networks. Kluwer Academic. 2006).

[15] Network Simulator (NS-2) http://www.isi.edu/nsnam/ns/

[16] Manual de NS-2 http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

[17] NAM (Network AniMator) http://www.isi.edu/nsnam/nam/

[18] Generador de movimientos *Setdest* http://winet.ece.ufl.edu/~wen/setdest_para.html

[19] Formato trazas NS-2 http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats

[20] Manual awk http://h1.ripway.com/chube/Manual_Awk/Menus.htm

[21] Método de bisección http://es.wikipedia.org/wiki/M%C3%A9todo_de_bisecci%C3%B3n

Anexo A. Ficheros utilizados en las simulaciones

En este anexo se presentan algunos de los ficheros de simulación utilizados en este proyecto. No se presentan todos ya que en general el código es bastante similar y no aportaría demasiado, así que se han recogido tres de los más representativos. Junto al código aparecen unos extensos comentarios para facilitar su comprensión.

```
                              # escenario_pequeño.tcl #

# ======================================================================
# DEFINICIÓN DE LA CONFIGURACIÓN DE LOS NODOS
# ======================================================================
set val(chan)           Channel/WirelessChannel    ;# tipo de canal
set val(prop)           Propagation/TwoRayGround   ;# modelo de propagación
set val(netif)          Phy/WirelessPhy            ;# interfaz de red
set val(mac)            Mac/802_11                 ;# tipo MAC
set val(ifq)            CMUPriQueue                ;# tipo interfaz de cola
set val(ll)             LL                         ;# tipo capa de enlace
set val(ant)            Antenna/OmniAntenna        ;# modelo de antena
set val(ifqlen)         50                         ;# maximo de paquetes en ifq
set val(nn)             20                         ;# número de nodos móviles
set val(rp)             DSR                        ;# protocolo de encaminamiento
set val(pck)            1500                       ;# Tamaño de los paquetes
set val(icbr)           0.005                      ;# Intervalo entre los paquetes
set val(x)              200                        ;# dimensión X de la topografía
set val(y)              200                        ;# dimensión Y de la topografía
set val(seed)           0.0
set val(cp)             "/home/jordi/Escriptori/PFC/r30/setdest1_05.tcl"
# archivo setdest con la posición inicial y movimientos de los nodos
set val(stop)           150.0                    ;# tiempo de simulación en segundos

set val(MNcoverage) 30.0;          #cobertura de los nodos
# Este valor es distinto según el caso a simular, dentro del escenario pequeño puede
# ser de 30 metros o 50 metros.

Mac/802_11 set dataRate_          11Mb ;# Configuración de los parámetros de la capa
Mac/802_11 set basicRate_          1Mb ;# MAC IEEE 802.11 para que funcione como el
Mac/802_11 set PLCPDataRate_       1Mb ;# IEEE 802.11b con el preÃ¡mbulo corto
Mac/802_11 set RTSThreshold_         0
Mac/802_11 set PreambleLength_      72
Mac/802_11 set PLCPHeaderLength_    24
Agent/UDP set  packetSize_        1500 ;# Tamaño de los paquetes UDP
```

```
# Definición de la función para el cálculo de la potencia transmitida a partir de la
# cobertura deseada
proc SetPt { coverage } {
  set Gt [Antenna/OmniAntenna set Gt_]
  set Gr [Antenna/OmniAntenna set Gr_]
  set ht [Antenna/OmniAntenna set Z_]
  set hr [Antenna/OmniAntenna set Z_]
  set RXThresh [Phy/WirelessPhy set RXThresh_]
  set d4 [expr pow($coverage,4)]
  set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
  return $Pt
}
Phy/WirelessPhy set Pt_ [SetPt $val(MNcoverage)] ;# asigna la potencia calculada

# ======================================================================
# PROGRAMA PRINCIPAL
# ======================================================================

# Inicialización de las variables globales
set ns_      [new Simulator]
set tracefd  [open escen_peq-out.tr w]
set namtrace [open escen_peq-out.nam w]

# Escritura de las trazas NS y NAM
$ns_ use-newtrace ;# usamos el nuevo formato de traza
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Creación la topografía del escenario
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Asignación parámetros básicos de las antenas de los nodos
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 0.2
Antenna/OmniAntenna set Gr_ 0.2

# Inicialización de los parámetros del interfaz radio de los nodos
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
```

```
#Phy/WirelessPhy set RXThresh_ 3.652e-2
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set Pt_ [SetPt $val(MNcoverage)] ;# asigna la potencia trnsmitida a
# partir de la función previamente definida

# Creación y configuración de los nodos con los parámetros determinados previamente
set god_ [create-god $val(nn)]
$ns_ node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
             -propType $val(prop) \
             -phyType $val(netif) \
             -channelType $val(chan) \
             -topoInstance $topo \
             -agentTrace ON \
             -routerTrace ON \
             -macTrace OFF \
             -movementTrace OFF

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

# ================================================================
# DEFINICIÓN DE LA POSICIÓN INICIAL Y DEL MOVIMIENTO DE LOS NODOS
# ================================================================

source $val(cp)          ;# cargamos setdest creado previamente

# Se define la posición inicial de los nodos en el NAM
for {set i 0} {$i<$val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 10
}

# ================================================================
# DEFINICIÓN MODELO DE TRÁFICO
# ================================================================

# Creación agente de tráfico UDP y asignación a un nodo
```

```
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0

# Creación agente de tráfico Null y asignación a un nodo
set des1 [new Agent/Null]
$ns_ attach-agent $node_(1) $des1

# Conexión de los agentes creados
$ns_ connect $udp0 $des1

# Creación fuente de tráfico CBR
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ $val(pck)
$cbr0 set interval_ $val(icbr)
$cbr0 set rate_ 1mbps

# Asignación fuente de tráfico con su agente correspondiente
$cbr0 attach-agent $udp0

# ======================================================================
# PLANIFICACIÓN DE SUCESOS
# ======================================================================

$ns_ at 10.0 "$cbr0 start" ;# Se inicia la fuente de tráfico

# Se indica a los nodos que ha finalizado la simulación
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop) "$node_($i) reset";
}

# Se finaliza la simulación
$ns_ at $val(stop) "stop"
$ns_ at $val(stop).01 "puts \"FIN DE LA SIMULACIÓN\" ; $ns_ halt"

proc stop {} {
       global ns_ tracefd namtrace
       $ns_ flush-trace
       close $tracefd
       close $namtrace
       exec nam escen_peq-out.nam &
       exit 0
}

$ns_ run
```

```
                              # escenario_mediano.tcl #

# ===================================================================
# DEFINICIÓN DE LA CONFIGURACIÓN DE LOS NODOS
# ===================================================================
set val(chan)           Channel/WirelessChannel    ;# tipo de canal
set val(prop)           Propagation/TwoRayGround   ;# modelo de propagación
set val(netif)          Phy/WirelessPhy            ;# interfaz de red
set val(mac)            Mac/802_11                 ;# tipo MAC
set val(ifq)            CMUPriQueue                ;# tipo interfaz de cola
set val(ll)             LL                         ;# tipo capa de enlace
set val(ant)            Antenna/OmniAntenna        ;# modelo de antena
set val(ifqlen)         50                         ;# maximo de paquetes en ifq
set val(nn)             100                        ;# número de nodos móviles
# también hay casos con escenario mediano y 200 nodos.
set val(rp)             DSR                        ;# protocolo de encaminamiento
set val(pck)            1500                       ;# Tamaño de los paquetes
set val(icbr)           0.005                      ;# Intervalo entre los paquetes
set val(x)              500                        ;# dimensión X de la topografía
set val(y)              500                        ;# dimensión Y de la topografía
set val(seed)           0.0
set val(cp)             "/home/jordi/Escriptori/PFC/r75_n100/setdest1_05.tcl"
# archivo setdest con la posición inicial y movimientos de los nodos
set val(stop)           150.0                    ;# tiempo de simulación en segundos

set val(MNcoverage) 75.0;          #cobertura de los nodos
# Este valor es distinto según el caso a simular, dentro del escenario mediano puede
# ser de 75 metros o 150 metros.

Mac/802_11 set dataRate_         11Mb ;# Configuración de los parámetros de la capa
Mac/802_11 set basicRate_         1Mb ;# MAC IEEE 802.11 para que funcione como el
Mac/802_11 set PLCPDataRate_      1Mb ;# IEEE 802.11b con el preÃ¡mbulo corto
Mac/802_11 set RTSThreshold_        0
Mac/802_11 set PreambleLength_     72
Mac/802_11 set PLCPHeaderLength_   24
Agent/UDP set  packetSize_       1500 ;# Tamaño de los paquetes UDP

# Definición de la función para el cálculo de la potencia transmitida a partir de la
# cobertura deseada
proc SetPt { coverage } {
  set Gt [Antenna/OmniAntenna set Gt_]
  set Gr [Antenna/OmniAntenna set Gr_]
  set ht [Antenna/OmniAntenna set Z_]
```

```
  set hr [Antenna/OmniAntenna set Z_]
  set RXThresh [Phy/WirelessPhy set RXThresh_]
  set d4 [expr pow($coverage,4)]
  set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
  return $Pt
}

# ======================================================================
# PROGRAMA PRINCIPAL
# ======================================================================

# Inicialización de las variables globales
set ns_      [new Simulator]
set tracefd  [open escen_peq-out.tr w]
set namtrace [open escen_peq-out.nam w]

# Escritura de las trazas NS y NAM
$ns_ use-newtrace ;# usamos el nuevo formato de traza
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Creación la topografía del escenario
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Asignación parámetros básicos de las antenas de los nodos
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 0.2
Antenna/OmniAntenna set Gr_ 0.2

# Inicialización de los parámetros del interfaz radio de los nodos
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
#Phy/WirelessPhy set RXThresh_ 3.652e-2
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set Pt_ [SetPt $val(MNcoverage)] ;# asigna la potencia trnsmitida a
# partir de la función previamente definida

# Creación y configuración de los nodos con los parámetros determinados previamente
set god_ [create-god $val(nn)]
```

```
$ns_ node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
             -propType $val(prop) \
             -phyType $val(netif) \
             -channelType $val(chan) \
             -topoInstance $topo \
             -agentTrace ON \
             -routerTrace ON \
             -macTrace OFF \
             -movementTrace OFF

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

# ==================================================================
# DEFINICIÓN DE LA POSICIÓN INICIAL Y DEL MOVIMIENTO DE LOS NODOS
# ==================================================================

source $val(cp)           ;# cargamos setdest creado previamente

# Se define la posición inicial de los nodos en el NAM
for {set i 0} {$i<$val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 10
}

# ==================================================================
# DEFINICIÓN MODELO DE TRÁFICO
# ==================================================================

# Creación agente de tráfico UDP y asignación a un nodo
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0

# Creación agente de tráfico Null y asignación a un nodo
set des1 [new Agent/Null]
$ns_ attach-agent $node_(1) $des1

# Conexión de los agentes creados
```

```
$ns_ connect $udp0 $des1

# Creación fuente de tráfico CBR
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ $val(pck)
$cbr0 set interval_ $val(icbr)
$cbr0 set rate_ 1mbps

# Asignación fuente de tráfico con su agente correspondiente
$cbr0 attach-agent $udp0

# ======================================================================
# PLANIFICACIÓN DE SUCESOS
# ======================================================================

# Se etiquetan los nodos fuente y destino
# Al exisitir en este escenario una gran cantidad de nodos es necesario marcar los
# nodos importantes para poder distinguirlos fácilmente en el NAM
$ns_ at 5.0 "$node_(0) label \"Origen\""
$ns_ at 5.0 "$node_(1) label \"Destino\""

$ns_ at 10.0 "$cbr0 start" ;# Se inicia la fuente de tráfico

# Se indica a los nodos que ha finalizado la simulación
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop) "$node_($i) reset";
}

# Se finaliza la simulación
$ns_ at $val(stop) "stop"
$ns_ at $val(stop).01 "puts \"FIN DE LA SIMULACIÓN\" ; $ns_ halt"

proc stop {} {
       global ns_ tracefd namtrace
       $ns_ flush-trace
       close $tracefd
       close $namtrace
       exec nam escen_peq-out.nam &
       exit 0
}

$ns_ run
```

```
                    # escenario_pequeño_2interferentes.tcl #

# ===================================================================
# DEFINICIÓN DE LA CONFIGURACIÓN DE LOS NODOS
# ===================================================================
set val(chan)           Channel/WirelessChannel    ;# tipo de canal
set val(prop)           Propagation/TwoRayGround   ;# modelo de propagación
set val(netif)          Phy/WirelessPhy            ;# interfaz de red
set val(mac)            Mac/802_11                 ;# tipo MAC
set val(ifq)            CMUPriQueue                ;# tipo interfaz de cola
set val(ll)             LL                         ;# tipo capa de enlace
set val(ant)            Antenna/OmniAntenna        ;# modelo de antena
set val(ifqlen)         50                         ;# maximo de paquetes en ifq
set val(nn)             20                         ;# número de nodos móviles
set val(rp)             DSR                        ;# protocolo de encaminamiento
set val(pck)            1500                       ;# Tamaño de los paquetes
set val(icbr)           0.005                      ;# Intervalo entre los paquetes
set val(x)              200                        ;# dimensión X de la topografía
set val(y)              200                        ;# dimensión Y de la topografía
set val(seed)           0.0
set val(cp)             "/home/jordi/Escriptori/PFC/2interferentes/r50/setdest1_05.tcl"
# archivo setdest con la posición inicial y movimientos de los nodos
set val(stop)           150.0                   ;# tiempo de simulación en segundos

set val(MNcoverage) 50.0;          #cobertura de los nodos
# Este valor es distinto según el caso a simular, dentro del escenario pequeño puede
# ser de 30 metros o 50 metros.

Mac/802_11 set dataRate_         11Mb ;# Configuración de los parámetros de la capa
Mac/802_11 set basicRate_         1Mb ;# MAC IEEE 802.11 para que funcione como el
Mac/802_11 set PLCPDataRate_      1Mb ;# IEEE 802.11b con el preÃ¡mbulo corto
Mac/802_11 set RTSThreshold_        0
Mac/802_11 set PreambleLength_     72
Mac/802_11 set PLCPHeaderLength_   24
Agent/UDP set  packetSize_       1500 ;# Tamaño de los paquetes UDP

# Definición de la función para el cálculo de la potencia transmitida a partir de la
# cobertura deseada
proc SetPt { coverage } {
  set Gt [Antenna/OmniAntenna set Gt_]
  set Gr [Antenna/OmniAntenna set Gr_]
  set ht [Antenna/OmniAntenna set Z_]
  set hr [Antenna/OmniAntenna set Z_]
```

```
  set RXThresh [Phy/WirelessPhy set RXThresh_]
  set d4 [expr pow($coverage,4)]
  set Pt [expr ($RXThresh*$d4)/($Gt*$Gr*$ht*$ht*$hr*$hr)]
  return $Pt
}

# ======================================================================
# PROGRAMA PRINCIPAL
# ======================================================================

# Inicialización de las variables globales
set ns_      [new Simulator]
set tracefd  [open escen_peq-out.tr w]
set namtrace [open escen_peq-out.nam w]

# Escritura de las trazas NS y NAM
$ns_ use-newtrace ;# usamos el nuevo formato de traza
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Creación la topografía del escenario
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

# Asignación parámetros básicos de las antenas de los nodos
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 0.2
Antenna/OmniAntenna set Gr_ 0.2

# Inicialización de los parámetros del interfaz radio de los nodos
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
#Phy/WirelessPhy set RXThresh_ 3.652e-2
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set Pt_ [SetPt $val(MNcoverage)] ;# asigna la potencia trnsmitida a
# partir de la función previamente definida

# Creación y configuración de los nodos con los parámetros determinados previamente
set god_ [create-god $val(nn)]
$ns_ node-config -adhocRouting $val(rp) \
```

```
              -llType $val(ll) \
              -macType $val(mac) \
              -ifqType $val(ifq) \
              -ifqLen $val(ifqlen) \
              -antType $val(ant) \
              -propType $val(prop) \
              -phyType $val(netif) \
              -channelType $val(chan) \
              -topoInstance $topo \
              -agentTrace ON \
              -routerTrace ON \
              -macTrace OFF \
              -movementTrace OFF

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

# ================================================================
# DEFINICIÓN DE LA POSICIÓN INICIAL Y DEL MOVIMIENTO DE LOS NODOS
# ================================================================

source $val(cp)           ;# cargamos setdest creado previamente

# Se define la posición inicial de los nodos en el NAM
for {set i 0} {$i<$val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 10
}

# ================================================================
# DEFINICIÓN MODELO DE TRÁFICO
# ================================================================

# Creación agente de tráfico UDP y asignación a un nodo
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0

# Creación agente de tráfico Null y asignación a un nodo
set des1 [new Agent/Null]
$ns_ attach-agent $node_(1) $des1

# Conexión de los agentes creados
$ns_ connect $udp0 $des1
```

```
# Creación fuente de tráfico CBR
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ $val(pck)
$cbr0 set interval_ $val(icbr)
$cbr0 set rate_ 1mbps

# Asignación fuente de tráfico con su agente correspondiente
$cbr0 attach-agent $udp0

# Se crean a continuación las 2 fuentes y los dos destinos interferentes, el proceso es
casi idéntico al descrito anteriormente

set udp4 [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp4

set des5 [new Agent/Null]
$ns_ attach-agent $node_(5) $des5

$ns_ connect $udp4 $des5
# Se usan los nodos 4 y 5 para esta conexión como podrían ser otros nodos cualesquiera.
# El nodo 3 no se usa ya que en el filtro que se explica en el anexo B, se usará con
# otro fin.

set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ $val(pck)
$cbr4 set interval_ $val(icbr)
$cbr4 set rate_ 1mbps

$cbr4 attach-agent $udp4

set udp6 [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp6

set des7 [new Agent/Null]
$ns_ attach-agent $node_(7) $des7

$ns_ connect $udp6 $des7

set cbr6 [new Application/Traffic/CBR]
$cbr6 set packetSize_ $val(pck)
$cbr6 set interval_ $val(icbr)
$cbr6 set rate_ 1mbps
```

```
$cbr6 attach-agent $udp6

# ======================================================================
# PLANIFICACIÓN DE SUCESOS
# ======================================================================

# Se etiquetan los nodos fuente y destino
# Al exisitir en este escenario varios nodos fuente y destino es conveniente marcar los
# nodos importantes para poder distinguirlos fácilmente en el NAM
$ns_ at 5.0 "$node_(0) label \"Origen\""
$ns_ at 5.0 "$node_(1) label \"Destino\""
$ns_ at 5.0 "$node_(4) label \"Interferente O1\""
$ns_ at 5.0 "$node_(5) label \"Interferente D1\""
$ns_ at 5.0 "$node_(6) label \"Interferente O2\""
$ns_ at 5.0 "$node_(7) label \"Interferente D2\""

# Se inician las fuentes de tráfico
$ns_ at 10.0 "$cbr0 start"
$ns_ at 10.0 "$cbr4 start"
$ns_ at 10.0 "$cbr6 start"

# Se indica a los nodos que ha finalizado la simulación
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop) "$node_($i) reset";
}

# Se finaliza la simulación
$ns_ at $val(stop) "stop"
$ns_ at $val(stop).01 "puts \"FIN DE LA SIMULACIÓN\" ; $ns_ halt"

proc stop {} {
       global ns_ tracefd namtrace
       $ns_ flush-trace
       close $tracefd
       close $namtrace
       exec nam escen_peq-out.nam &
       exit 0
}

$ns_ run
```

Anexo B. Filtros AWK utilizados para extraer los resultados

En este anexo se presentan los dos filtros AWK utilizados en este proyecto. Se incluyen con los comentarios oportunos para facilitar su comprensión.

```
                                # filtro_throughput.awk #

BEGIN {
# Las instrucciones contenidas dentro de BEGIN sólo se ejecutan una vez

# Se abre el fichero de salida
salida2="Graf_troughput.txt"

# Se inicializan las variables globales
mayor_id=-1
bytes_env_0=0
rec_1=0
env_0=0
bytes_rec_11=0
}

{
# Las instrucciones contenidas dentro de este apartado principal se ejecutan una vez
# para cada fila del fichero

# Se asignan variables a las columnas o campos a utilizar del fichero de trazas
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$35
trace_level=$19

if(accion!="SFESTs" && accion!="SFs" && trace_level=="AGT"){
# Se analizan los paquetes enviados
if ( accion == "s"){
    if (id_paquete > mayor_id){ # Se miran los paquetes no analizados previamente
        mayor_id=id_paquete
       if ( prot_mes == "cbr"){ # Sólo interesan los paquetes CBR
                if ( nodo_actual == 0){
                # Se comprueba si el paquete lo envía el nodo fuente
```

```
                    bytes_env_0 = bytes_env_0 + bytes_paquete
                    # Se suman los bytes CBR enviados por el nodo 0 o fuente
                    env_0++
                    # Se incrementa el número de paquetes CBR enviados por el nodo 0
                }
        }
    }
}

# Se analizan los paquetes recibidos
if ( accion == "r"){
    if (nodo_actual == nodo_destino){
        if (nodo_actual == 1){
             bytes_rec_1 = bytes_rec_1 + bytes_paquete - 20
             # Se suman los bytes CBR recibidos por el nodo 1 o destino, se descuenta
             # la cabecera IP de 20 bytes
             rec_1++
             # Se incrementa el número de paquetes CBR recibidos por el nodo 1
        }
    }
}

}
}

END{
# Las instrucciones contenidas dentro de END sólo se ejecutan una vez

# Se guardan los resultados obtenidos en el fichero de salida
printf(": ") >> salida2
printf("Paquetes recibidos  %i ", rec_1) > salida2
printf("Paquetes enviados  %i ", env_0) > salida2
printf("Throughput1 %f ", rec_1/env_0) > salida2
printf("Bytes recibidos  %i ", bytes_rec_1) > salida2
printf("Bytes enviados  %i ", bytes_env_0) > salida2
printf("Throughput2  %f \n", bytes_rec_1/bytes_env_0) > salida2

# Aunque se calculan dos throughputs, uno a partir de los paquetes y otro a partir de
# los bytes (descontando en este último caso los bytes de las cabeceras IP), en este
# proyecto todos los datos obtenidos del throughput usados son a partir del número de
# paquetes. El segundo throughput no se ha usado para extraer los resultados pero se
# mantiene aquí como dato extra.

# Se cierra el fichero de salida
```

```
close(salida2)
}
```

```
                                # filtro_rutas.awk #

BEGIN {
# Las instrucciones contenidas dentro de BEGIN sólo se ejecutan una vez

# Se abre el fichero de salida
salida2="Graf_rutas.txt"

# Se inicializan las variables globales
rutas_encontradas=0
nodo_fuente=0
nodo_destino=1
tiempo_inicial=0
tiempo_final=150
ruta_activa=0
solicitud_ruta_ok=0
transmitiendo=0
ruta_usada=0
rutas_validas=0
umbral=1
posible_nueva_ruta=0
cambios_rutas=0
tiempo_ultimo_dsr=0
rutas_encontradas_int=0
nodo_intermedio=3
# Se ha escogido el nodo 3 para ser usado como nodo intermedio del cual calcularemos
# las rutas. El nodo escogido podía haber sido cualquier otro que no fuese ni fuente ni
# destino.
duracion_rutas=0
paq_tr_id=0
ip_nodo_fuente_DSR=0.255
# ip . puerto, el puerto es 255 para los paquetes dsr y 0 para los paquetes cbr
ip_nodo_destino_DSR=1.255
ip_nodo_fuente_cbr=0.0
ip_nodo_destino_cbr=1.0

# En una primera versión se mostraban por pantalla la información de todos los paquetes
DSR, una vez evolucionado el filtro ya no ha sido necesario, de todas formas dejo las
instrucciones necesarias como información adicional.

#format = "%-10s %-10s %-10s %-10s %-10s %-10s %-10s %-10s %-10s %-10s %s\n"
#             printf format, "Accion",
"Node_id","Level","pkt_type","RREQ","RREP","pkt_id", "pkt_size","X","Y","Time" >>
salida2
```

```
#            printf format, "----", "------","------","------","------","------","----
--","------","------","------","------" >> salida2
}

{
# Las instrucciones contenidas dentro de este apartado principal se ejecutan una vez
# para cada fila del fichero

# Se asignan variables a las columnas o campos a utilizar del fichero de trazas
time=$3
accion=$1
nodo_id=$9
position_X=$11
position_y=$13
trace_level=$19
pkt_type=$35
pkt_size=$37
pkt_id=$41
rreq_id=$49
rrep_id=$53
src_ip=$31
dst_ip=$33

# Se comprueba que los paquetes DSR corresponden a tráfico de los nodos 0 y 1, que son
# el origen y el destino que nos interesan. Esto es imprescindible cuando hay tráfico
# interferente, en los casos que no lo hay, esto no afecta al filtro.
if((src_ip==ip_nodo_fuente_DSR && dst_ip==ip_nodo_destino_DSR) ||
(dst_ip==ip_nodo_fuente_DSR && src_ip==ip_nodo_destino_DSR) ||
dst_ip==ip_nodo_fuente_DSR){

# Se analizan sólo los paquetes DSR
if(accion!="SFESTs" && accion!="SFs" && pkt_type=="DSR"){

# Las líneas siguientes también corresponden a una primera versión en que se mostraban
por pantalla la información de todos los paquetes DSR, por eso está como comentario.
#printf
format,accion,nodo_id,trace_level,pkt_type,rreq_id,rrep_id,pkt_id,pkt_size,position_X,p
osition_y,time >> salida2
#}

# Nodo fuente envía un Route Request, es decir busca una nueva ruta
if(pkt_type=="DSR" && accion=="s"){
      if(rreq_id==1){
            tiempo_ultimo_dsr=time
            if(nodo_id==nodo_fuente && ruta_activa==1){
```

```
                  posible_nueva_ruta=0
                  tiempo_final=time
                  if(ruta_usada==1){
                        #printf("%f DuraciÃ³n ruta = %f \n", time, tiempo_final-
tiempo_inicial) >> salida2
                        rutas_validas++
                        duracion_rutas+=(tiempo_final-tiempo_inicial)
                  }
                  ruta_activa=0
                  transmitiendo=0
                  ruta_usada=0
                  tiempo_inicial=0
                  tiempo_final=150
            }
      # detecta un Route Error o un Route Reply que puede propiciar un cambio de ruta
sin que el nodo fuente envíe ni reciba ningun paquete de DSR
      }else{
            if(ruta_activa==1 && (time-tiempo_ultimo_dsr)>umbral){
                  tiempo_ultimo_dsr=time
                  posible_nueva_ruta=1
            }
      }
}

# Nodo fuente recibe un Route Reply, es decir una nueva ruta
if(pkt_type=="DSR" && accion=="r"){
      if(rrep_id==1){
            tiempo_ultimo_dsr=time
            if(nodo_id==nodo_fuente){
                  posible_nueva_ruta=0
                  if(ruta_activa==0){
                        solicitud_ruta_ok++
                        tiempo_inicial=time
                  }
                  ruta_activa=1
                  rutas_encontradas++
            }
      }
}

# Detecta un cambio de ruta transparente para el nodo fuente, pone posible ya que puede
no ser 100% infalible
if(pkt_type=="DSR" && accion=="r"){
      if(rrep_id==1 && ruta_activa==1 && posible_nueva_ruta==1){
```

```
                posible_nueva_ruta=0
                tiempo_final=time
                if(ruta_usada==1){
# La siguiente instrucción también correspondía a una primera versión
                        #printf("%f Duración Posible ruta = %f \n", time,
tiempo_final-tiempo_inicial) >> salida2
                        rutas_validas++
                        duracion_rutas+=(tiempo_final-tiempo_inicial)
                }
                transmitiendo=0
                ruta_usada=0
                tiempo_inicial=time
                tiempo_final=150
                cambios_rutas++
        }
}

# Nodo intermedio recibe un Route Reply, es decir una nueva ruta
if(pkt_type=="DSR" && accion=="r"){
        if(rrep_id==1){
                if(nodo_id==nodo_intermedio){
                        rutas_encontradas_int++
                }
        }
}

}
# Aquí se cierra la comparación de las direcciones ip, con puerto 255 para los paquetes
DSR

# Para los paquetes cbr también hay que diferenciar los paquetes que envía la fuente al
destino de los que se envían los nodos interferentes
if(src_ip==ip_nodo_fuente_cbr && dst_ip==ip_nodo_destino_cbr){

# Se comprueba que la ruta se llega a usar, descarta las rutas "falsas" o no
actualizadas
if(accion!="SFESTs" && accion!="SFs" && pkt_type=="cbr"){
        if(accion=="s" && nodo_id==nodo_fuente && ruta_activa==1 && transmitiendo==0){
                transmitiendo=1
                paq_tr_id=pkt_id
        }
        if(accion=="r" && nodo_id==nodo_destino && ruta_activa==1 && transmitiendo==1 &&
pkt_id>paq_tr_id){
                ruta_usada=1
```

```
	}
}

}
# Aquí se cierra la comparación de las direcciones ip, con puerto 0 para los paquetes cbr

}


END{
# Las instrucciones contenidas dentro de END sólo se ejecutan una vez

# Se comprueba si en el momento de finalizar la transmisión hay una ruta activa
if (transmitiendo==1){
	duracion_rutas+=(tiempo_final-tiempo_inicial)
	if(tiempo_final!=tiempo_inicial){
		rutas_validas++
	}
}
# Se guardan los resultados obtenidos en el fichero de salida
printf("############# RESUMEN DE DATOS ############ \n") >> salida2
printf("----Tiempo medio duración por ruta  =  %f \n", duracion_rutas/rutas_validas) > salida2
printf("Rutas encontradas totales = %i \n", rutas_encontradas) > salida2
printf("Rutas encontradas nodo intermedio = %i \n", rutas_encontradas_int) > salida2
printf("Solicitudes de Ruta con alguna ruta encontrada = %i \n", solicitud_ruta_ok) > salida2
printf("Rutas validas usadas = %i \n", rutas_validas) > salida2
printf("Cambios de ruta sin enterarse nodo fuente = %i \n", cambios_rutas) > salida2
printf("tiempo final = %f \n", tiempo_final) > salida2
printf("tiempo inicial = %f \n", tiempo_inicial) > salida2
printf("Duración ruta final = %f \n", tiempo_final-tiempo_inicial) > salida2
printf("------------------------------------------------------ \n") > salida2

# Se cierra el fichero de salida
close(salida2)
}
```