ESCUELA SUPERIOR POLITÉCNICA DEL CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

"ESTUDIO COMPARATIVO DE PROTOCOLOS DE RUTEO EN REDES AD HOC APLICADO A REDES MOVILES"

TESIS DE GRADO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN

Presentado por:

PABLO FELIPE ANDINO JURADO

Riobamba – Ecuador

2010

Expreso mi agradecimiento a la Escuela Superior Politécnica del Chimborazo, a la Facultad de Informática y Electrónica, a la Escuela de Ingeniería Electrónica por haberme brindado la oportunidad de optar por una carrera actual y reconocida, la cual me permitirá continuar con mi formación profesional.

De igual manera al Ing. Alberto Arellano e Ing. Marcelo Donoso; quienes colaboraron como asesores en este trabajo brindándome su apoyo.

Quiero expresar mis agradecimientos de todo corazón a mi familia, por ser el apoyo fundamental en todos los aspectos de mi vida.

A MI FAMILIA:

Le dedico este trabajo a mi familia por todo su amor, apoyo y paciencia en todo momento de mi vida.

NOMBRE	FIRMA	FECHA
Ing. Iván Menes **DECANO FACULTAD INFORMÁTICA** **Y ELECTRÓNICA**	……	…….....
Ing. José Guerra **DIRECTOR ESCUELA** **INGENIERÍA ELECTRÓNICA**	……	…….....
Ing. Alberto Arellano **DIRECTOR DE TESIS**	……	…….....
Ing. Marcelo Donoso **MIEMBRO DE TESIS**	……	…….....
Lcdo. Carlos Rodríguez **DIRECTOR CENTRO DOCUMENTACIÓN**	……	…….....

NOTA DE LA TESIS ESCRITA: ……

Yo, PABLO FELIPE ANDINO JURADO, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica del Chimborazo.

_____

Pablo Andino

ÍNDICE DE ABREVIATURAS

AES	Advanced Encryption Standard (Estándar de enciptación avanzada)
AODV	Ad hoc On-Demand Distance Vector
AOMDV	Ad hoc on-demand multipath distance vector routing
ARP	Address Resolution Protocol (Protocolo de resolución de direcciones)
CBP	Contention Base Protocol (Protocolo base de contención)
CBR	Constant Bit Rate (Taza constante de bits)
CCK	Complementary Code Keying (Encriptacion de Codigo Complementario)
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance (acceso múltiple por detección de portadora con evasión de colisiones)
DARPA	Defense Advanced Research Projects Agency (Agencia de Investigación de Proyectos Avanzados de Defensa)
DCF	Hybrid Coordination Function
DFS	Dynamic Frequency Selection (Selección de Frecuencia Dinámica)
DSDV	Destination-Sequenced Distance-Vector
DSE	Dependent Station Enablement (Activacion de estación dependiente)
DSR	Dynamic Source Routing
DSSS	direct sequence spread spectrum (Espectro Ensanchado por Secuencia Directa)
ECSA	Extended Channel Switch Announcement (Anuncio extendido de cambio de canal)
EDCA	Enhanced Distributed Channel Access (Distribucion de canal de acceso mejorado)
ELF	Extremely low frequency (Frecuencia Extremadamente Baja)
ERO	European Radiocomunications Office (Oficina de radiocomunicaciones Europeas)
FTP	File Transfer Protocol (Protoclo de transferencia de archivos)
GNU	GNU is Not Unix (GNU No es Unix)
GUI	graphical user interface (interfaz gráfica de usuario)
HCCA	Enhanced Distributed Channel Access
HCF	Hybrid Coordination Function (Función de coordinación híbrida)
IAPP	HCF Controlled Access, equivalente a
ICANN	Corporación de Internet para la Asignación de Nombres y Números
IEEE	Institute of Electrical and Electronics Engineers (Instituto de ingenieros eléctricos y electrónicos)
IETF	Internet Engineering Task Force (Grupo de Trabajo en Ingeniería de Internet)

IP	Internet Protocol (Protocolo de Internet)
ISM	Industrial, Scientific and Medical (Industrial, Científico y Médico)
ITU	Unión Internacional de Telecomunicaciones
LAN	Local Area Network (Red de Área Local)
LL	Link Layer (nivel de enlace)
MAC	Media Access Control (Control de Acceso al Medio)
MAN	Metropolitan Area Network (Red de Área Metropolitana)
MANET	Mobile AdHoc Network (Red AdHoc móvil)
MIMO	Multiple Input – Multiple Output (Múltiple Entrada - Múltiple salida)
MPR	Multipoint Relays
NAM	Network animator
NS	Network Simulator
NS2	Network Simulator 2
OFDM	Orthogonal Frequency-Division Multiplexing (Multiplexación por División de Frecuencias Ortogonales)
OLSR	Optimized Link State Routing
OSI	Open System Interconnection (modelo de referencia de Interconexión de Sistemas Abiertos)
OSPF	Open Shortest Path First (Primero el camino abierto mas corto)
Otcl	TCL orientado a objetos
PBCC	Packet Binary Convolutional Coding (Codigo Convolucional de Paquetes Binarios)
PCF	Point Coordinated Function (Función de Punto Coordinado)
PDAs	Personal Digital Assistant (Asistente digital personal)
PHY	Physical (Físico)
PIC	Peripheral Interface Controller (controlador de interfaz periférico)
QoS	Quality of Service (Calidad de Servicio)
RED	Random Early Detection (Detección Randómica temprana)
RERR	Route Errors
RFC	Request For Comments
RIP	Routing Information Protocol (Protocolo de información de ruteo)
RREP	Route Replies
RREQ	Route Request
TC	Topology Control (Control de Topología)
Tcl	Tool Command Language (Lenguaje de herramientas de comando)
TCP	Transmission Control Protocol (Protocolo de Control de Transmisión)
Tk	Tool Kit
TKIP	Temporal Key Integrity Protocol (Protocolo de integridad de clave temporal)
TPC	Transmitter Power Control (Control de Potencia de Transmición)
UCB	Unit Control Block (Bloque de control de unidad)
UDP	User Datagram Protocol (Protocolo de Datagramas del Usuario)
UHF	Ultra High Frequency (frecuencia ultra alta)

USC/ISI	University of Southern California/Information Sciences Institute
VANET	Vehicular Ad-Hoc Network (Red Ad-Hoc vehicular)
VoIP	Voz sobre Protocolo de Internet
WIFI	Wireless Fidelity (Fidelidad inalámbrica)
WIMAX	Worldwide Interoperability for Microwave Access (interoperabilidad mundial para acceso por microondas)
WLAN	Wireless Local Area Network (Red de Área local inalámbrica)
WPA	Wi-Fi Protected Access (Acceso protegido de Wi-Fi)
YUM	Yellow Dog Updater Modified

ÍNDICE GENERAL

ÍNDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCIÓN

La comunicación inalámbrica con su gran desempeño y penetración en las redes de comunicaciones ha creado diferentes desafíos en cuanto a sus implementaciones y aplicaciones en múltiples ambientes. El concepto de movilidad para una red obliga a encontrar nuevas soluciones para las tareas principales de esta.

En este trabajo se realizo un estudio comparativo entre protocolos de enrutamiento en redes móviles AdHoc para mejorar el funcionamiento de las mismas.

El estudio se ejecutó mediante la programación de simulaciones en NS2 sobre Linux CentOs, creando escenarios en los cuales se sometió a distintas pruebas cada protocolo, los enlaces entre los nodos trabajan en una red inalámbrica WiFi simulada que busca enviar información a través de saltos en la red analizando el consumo de energía.

Al tener que gestionar enlaces en una red móvil de tipo AdHoc, se debe resolver el problema principal que es el enrutamiento. Para ello cada nodo debe contar con la capacidad de gestionar sus tablas de ruteo. De esta manera este trabajo busca estudiar y comparar los principales protocolos que buscan dar solución al enrutamiento en redes inalámbricas móviles.

Para realizar esta comparación de una manera práctica y objetiva, se realiza las pruebas dentro de un ambiente de simulación realizado en NS2 sobre Linux. En este simulador se puede realizar múltiples simulaciones para los protocolos AODV, AOMDV, DSR Y OLSR. Cada uno de estos protocolos enfoca la red desde un punto distinto brindando diferentes soluciones al encaminamiento.

Como resultado se obtiene datos respecto a los paquetes enviados junto con la energía consumida para la acción, que varían de acuerdo a los principios básicos de cada protocolo, proporcionando de esta manera información sobre los puntos fuertes y debilidades de cada uno.

Finalmente, se tiene el comportamiento de cada protocolo en distintos escenarios, de manera que puede apreciarse las soluciones más adecuadas que pueden implementarse en distintas condiciones.

CAPÍTULO I

MARCO REFERENCIAL

FORMULACION GENERAL DEL PROYECTO DE TESIS

1.1. ANTECEDENTES

La utilización de una red inalámbrica provee movilidad y cobertura en acceso a la red, estas redes han tenido una amplia implementación dentro de las ciudades con diferentes tecnologías como WiFi que es una solución trabajando con IP para dar una solución a las redes inalámbricas que utiliza ondas de radio en lugar de cables.

Las redes Ad Hoc tienen su inicio en los años 70, inicialmente desarrolladas por la agencia DARPA del departamento de defensa de los Estados Unidos. Una red Ad Hoc es una red inalámbrica descentralizada para casos en los que la red se sujeta un escenario en el que los

usuarios no pueden referirse a un punto central para acceder a la red, esto puede ser por factores geográficos o por la naturaleza misma de la red, en este tipo de red los puntos tienen la capacidad de enviar sus datos de forma dinámica en función de la conectividad de la red por ello no requiere de un punto central que gestione la conexión.

El hecho de que este tipo de red sea descentraliza le provee ciertas características y aplicaciones en situaciones en las que no se puede confiar en un nodo central y se necesita una gran escalabilidad, así también son muy útiles en situaciones de emergencia en los que se requiere un rápido despliegue del sistema de comunicación.

La movilidad dentro de una red inalámbrica es fundamental, pero en el caso de una red Ad Hoc, al hablar de movilidad, podemos hablar de a mayor escala manteniendo la comunicación entre los distintos puntos, así viene al caso el término MANET's, es decir, una red móvil Ad Hoc, ideada para que cada punto tenga movilidad en cualquier dirección de manera independiente, cambiando sus enlaces para adaptarse a la red de manera frecuente, de esta manera el enrutamiento para mantener los enlaces es fundamental.

Al tener una red constantemente cambiante, las rutas para alcanzar los distintos puntos buscando un alcance de extremo a extremo serán muy volátiles, aquí aparecen los protocolos de ruteo en las redes ad hoc móviles, lo que buscan es mantener la conectividad encontrando continuamente las rutas más aptas para alcanzar los distintos nodos. Las redes móviles se ven sujetas a cambios dinámicos en la topología de la red esta causa conflictos al establecer las rutas por ello existe la necesidad de protocolos que permitan una rápida convergencia de la red encontrando las rutas optimas.

La solución al ruteo en redes ad hoc móviles existe gracias a una variedad de protocolos proactivos y reactivos que buscan la manera de mantener los enlaces activos entre los distintos nodos.

1.2. JUSTIFICACION DEL PROYECTO DE TESIS

El estudio comparativo entre los distintos protocolos de ruteo dentro de una red móvil ad hoc busca explorar a fondo las diferentes opciones al seleccionar uno de estos protocolos, junto con todo su funcionamiento, configuración y posibilidad que brinda al usuario dentro de un entorno inalámbrico simulado diseñado para tener movilidad y múltiples puntos que buscan la manera de comunicarse entre ellos.

La necesidad de conocer el funcionamiento de una red ad hoc móvil junto con las herramientas que permiten su funcionamiento nos lleva a analizar las diferentes soluciones existentes estudiando su comportamiento en un entorno estable para de esta manera poder establecer los parámetros que se deben considerar al momento de trabajar con una red de características únicas expuesta a múltiples conexiones variables y el constantes movimiento de los clientes de la red.

Cada protocolo está pensado para satisfacer distintas necesidades en distintos entornos por ello se busca evaluar los diferentes métodos que utilizan los protocolos para llevar a cabo la tarea de encontrar una ruta al momento de llevar a cabo la comunicación entre puntos distantes de manera simultánea, por ello se deberá encontrar los resultados de su comportamiento y

eficiencia esto considerando y evaluando en consumo de energía de los dispositivos que formaran parte de la red. Debido a las distancias y modos de conexión cada nodo tendrá un determinado consumo de energía establecido en gran parte por el protocolo que gestiona la conexión, lo que buscamos es comparar ese consumo y la eficiencia del protocolo encontrando de esta manera los protocolos a escoger en un momento determinado.

El presente proyecto de tesis busca hallar el rendimiento de los protocolos de ruteo en redes ad hoc móviles realizando pruebas simuladas y sometiendo las diferentes técnicas en diferentes escenarios posibles en los que una red de este tipo podría encontrarse, de esta manera lo que se busca es evaluar las distintas capacidades que poseen y cómo funcionan bajo distintas condiciones tratando de encontrar de esta manera su potencial y capacidades junto con sus falencias.

Con esto se lograra discernir entre las distintas opciones y con ello aplicar de mejor manera un criterio al momento de trabajar en el entorno deseado al momento de seleccionar el mejor protocolo para un escenario específico.

De esta manera el estudio servirá a los estudiantes de la carrera de ingeniería en electrónica como un material de apoyo y complemento para asignaturas como redes inalámbricas aplicado en ambientes móviles con el fin de tener un conocimiento avanzado en este ámbito, ya que será un completo análisis de los protocolos, funcionamiento y aplicaciones dentro de este campo poco explotado en el país como lo son las redes inalámbricas ad hoc.

1.3. OBJETIVOS

1.3.1. OBJETIVOS GENERALES:

Comparar el funcionamiento y eficiencia de los distintos protocolos de ruteo aplicados en redes AdHoc móviles

1.3.2. OBJETIVOS ESPECIFICOS:

- Estudiar las necesidades de una red móvil ad hoc que requiera protocolos de enrutamiento
- Analizar los distintos protocolos de ruteo en redes móviles y su funcionamiento
- Crear un ambiente simulado que permita el estudio de los distintos protocolos de enrutamiento con el software libre NS2.
- Comparar la eficiencia de los distintos protocolos
- Estudiar el consumo de energía de los distintos nodos en una red ad hoc dependiendo del protocolo utilizado.
- Analizar las diferentes técnicas al utilizar un protocolo de ruteo y sus diferentes algoritmos de operación

1.4. HIPOTESIS

Con el estudio entre los distintos protocolos de enrutamiento para redes móviles ad hoc se podrá comparar las capacidades de cada protocolo para encontrar el que mejor se adapta en un ambiente propuesto.

1.5. Recursos necesarios

1.5.1. Equipos a utilizar

CANTIDAD	NOMBRE
1	PC
1	Cable de red

Tabla I.I Equipos

1.5.2. Otros

CANTIDAD	NOMBRE
	Simulador NS2
	Internet
	Libros

Tabla I.II Herramientas

1.6. Métodos y Técnicas

1.6.1. Estudio comparativo

El método comparativo suele ser popular en un estado temprano de la evolución de un campo de investigación, cuando se intenta salir del nivel inicial de los estudios de caso exploratorios a un nivel más avanzado de estructuras teóricas generales o leyes, como invariantes, causalidad o evolución.

El diseño de la investigación comparativa es simple. Se estudia ejemplares que pertenecen al mismo grupo pero que difieren en algunos aspectos. Estas diferencias llegan a ser el foco de la exanimación. La meta es descubrir por qué los casos son diferentes, para revelar la estructura subyacente general que genera o permite tal variación.

a) Comparación descriptiva

En el estudio descriptivo de productos hay muchas situaciones donde comparación es un método adecuado. Se podría, por ejemplo, estudiar los productos comparables que han sido diseñados por diversos diseñadores, o hechos por diversos productores. O se puede estudiar el mismo producto que se utiliza en países diferentes.

La comparación puede ser útil también cuando el investigador no está interesado en diferencias sino en un caso singular. Si el objeto que interesa pertenece al entorno cultural normal del investigador, no siempre son fáciles de percibir sus características especiales.

En estudio exploratorio sucede a menudo que se debe agregar gradualmente nuevos aspectos de la comparación, o los definir nuevamente cuando su conocimiento del objeto aumenta. Es también común que en las fases iniciales del estudio que solamente se alcance respuestas

descriptivas a las preguntas cuál y cómo el objeto es, y de esta base se puede entonces intentar *explicar* o contestar a la pregunta por qué el objeto es como es.

b) Comparación normativa

La diferencia entre los estilos descriptivos y normativos de la comparación es que en el análisis normativo uno de los criterios principales son evaluativo como la "satisfacción", la "utilidad" etc., y la puntería del estudio es precisar el mejor (en este respecto) entre las alternativas que se estudian. Además, la puntería final quizás es encontrar no sólo el mejor objeto existente, sino también mejorar los objetos similares más tarde. Es decir se espera que el análisis comparativo dé argumentos para el planeamiento de mejoras en circunstancias o productos existentes.

Dado que toda evaluación es subjetiva es importante considerar y definir exactamente cuyo punto de vista se utiliza en la evaluación. Las opiniones más interesantes vienen a menudo de la gente que ha usado el producto; es a veces el grupo de blanco de los clientes futuros cuyos puntos de vista son esenciales. Si las sugerencias normativas que usted prepara son utilizadas por una organización, por ejemplo en una empresa del negocio, la elección del punto de vista depende también del grado de autonomía que prevalece en los varios niveles de la organización.

CAPÍTULO II

MARCO TEÓRICO

2.1. Introducción a las redes inalámbricas Ad-Hoc

2.1.1. ¿Por qué Ad-Hoc?

Las tecnologías inalámbricas en los últimos años han tenido un desarrollo considerable debido a su concepto de movilidad en todos los campos, por ello su utilización en la vida cotidiana se ha visto incrementada gracias a la utilización de dispositivos móviles que explotan están tecnologías y para ello se han implementado estructuras de comunicación principalmente centralizadas para proveer un sistema de comunicación. En su gran mayoría las estructura centralizadas con tecnologías como wi-fi o wi-max brindan una solución al usuario al momento de proveer acceso a la red, pero, en el caso en el que no se pueda depender de una red de centralizada sea por un problema geográfico en el que los nodos no puedan hacer referencia a un punto central o por

necesidad de flexibilidad en los enlaces se recurre a una estructura descentralizada en la que la capacidad de comunicación no recaiga sobre un punto central.

Para proveer una comunicación descentralizada contamos con enlaces inalámbricos de igual a igual conocidos como “ad-hoc” en los cuales los nodos que desean comunicarse se enlazan entre sí, creando múltiples enlaces para alcanzar otros nodos, de esta manera, al no tener un punto central de referencia, la comunicación puede continuar entre los nodos que tengan la capacidad de comunicarse entre sí.

Al tener la capacidad de movilidad en una red inalámbrica contando con un estructura descentralizada ad-hoc hablamos de una “red móvil ad hoc” a sus siglas en inglés “MANET” que propone una solución al problema de dependencia de una red centralizada común, por lo tanto una MANET es una red autónoma en la que los puntos se logran a comunicar a través de múltiples saltos entre sí para alcanzar el punto deseado, permitiendo de esta manera una comunicación que no recae sobre un punto central, sino, a través de los mismos miembros de la red.

Este sistema nos presenta un concepto de adaptación de la red en un ambiente de movilidad constantemente variable, buscando una organización automática que responda a cambios en la infraestructura y esquema de la red ya que los puntos de este sistema nos son controlados por un ente externo, por ello, los nodos para converger en la red deben interactuar entre sí, de esta manera no es un solo punto el responsable de la comunicación sino que cada punto contribuirá a la conectividad de la red.

Al ser los nodos libres en la red se organizan entre ellos mismos, por ello son aptos para responder una manera rápida, espontanea y adaptable a los cambios continuos como en aplicaciones militares, operaciones de emergencia, dispositivos de red electrónicos personales, o

sistemas de comunicación de alta movilidad como comunicaciones entre vehículos en una ciudad, como taxis o buses, también son aplicables en puntos de difícil acceso como zonas montañosas y áreas geográficas variables.

Las Redes Móviles ad-hoc fueron diseñadas para proveer comunicación y ser implementadas de una manera rápida y eficiente, en sitios carentes de una infraestructura de red. Pero para que esto sea posible se necesita contar con protocolos de enrutamiento específicos para esta estructura de red que lleven a cabo la tarea de organizar los enlaces, debido a que los protocolos tradicionales propios de redes centralizadas no se adaptan a este tipo de ambientes móviles.

2.1.2. Redes ad-hoc

Ad hoc es una locución latina que significa literalmente «para esto». Generalmente se refiere a una solución elaborada específicamente para un problema o fin preciso y, por tanto, no es generalizable ni utilizable para otros propósitos. Se usa pues para referirse a algo que es adecuado sólo para un determinado fin. En sentido amplio, ad hoc puede traducirse como «específico» o «específicamente». [1]

Dentro del ámbito de las redes inalámbricas una red ad-hoc implica una estructura sin un nodo central sino varios nodos con iguales condiciones con una característica de auto organización que busca lograr la comunicación entre los entes de igual condición de la red

Por el propio concepto de red ad-hoc, esta, presenta ciertas dificultades en su funcionamiento a gran escala, ya que los nodos deberán gestionar las rutas para comunicarse entre ellos sin contar con un equipo que desempeñe el papel de enrutador para gestionar los enlaces, por ello, cada nodo actuara como un enrutador y decidirá como transmitir sus datos.

2.1.3. Transmisión inalámbrica

La comunicación inalámbrica es aquella en la que los emisores y receptores no se encuentran unidos por un medio de propagación físico como el cobre o la fibra óptica, sino que en su lugar se utiliza ondas electromagnéticas a través del espacio para propagar una señal, es decir el aire.

Una comunicación a través del medio libre requiere una organización del espectro electromagnético que permita una separación de frecuencias para diferentes aplicaciones y con diferentes regulaciones, en la figura II.1 podemos observar una organización básica del espectro, lo lineamientos para su control son especificados por las autoridades nacionales mediante organismos de control del estado.

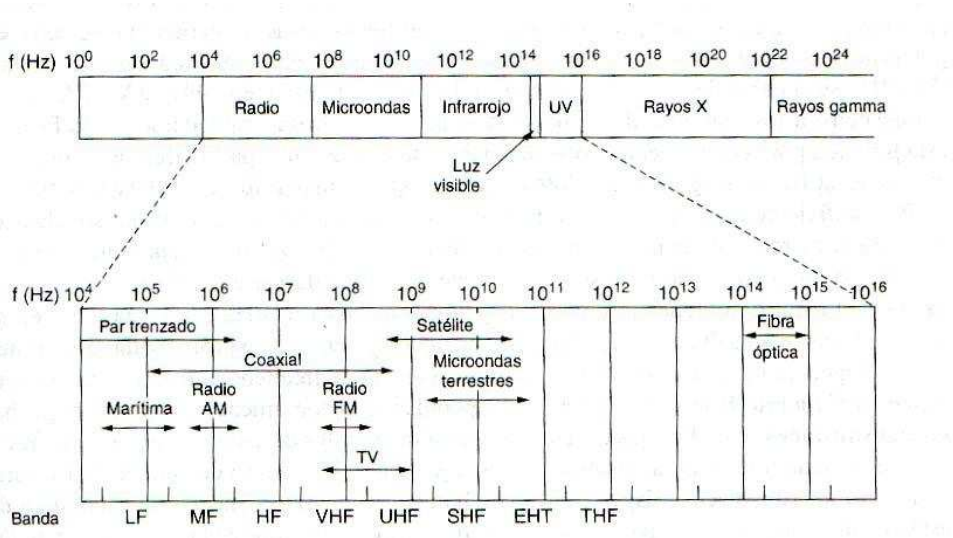


Figura II.1 Espectro electromagnético

Entre las ventajas de una red inalámbrica están los costos, ya que se elimina todo el cableado de las redes convencionales Ethernet y conexiones físicas entre nodos, pero su desventaja frente a redes cableadas esta la seguridad ya que por su construcción, una red inalámbrica se basa en broadcasting para cubrir su movilidad lo que la hace extremadamente vulnerable a

ataques, pero gracias a sus capacidades y beneficios esta tecnología es de las más prometedoras.

Según el rango de frecuencias que utilice un sistema para transmitir, los medios pueden ser las ondas de radio, las microondas terrestres o transmisiones satelitales, así como, los infrarrojos, o cualquier onda dentro del espectro radio eléctrico. Dependiendo del medio, una red inalámbrica tendrá ciertas características específicas que la definirán, entre estas tenemos:

- Ondas de radio: las ondas electromagnéticas son omnidireccionales, así que no son necesarias las antenas parabólicas. La transmisión no es sensible a las atenuaciones producidas por la lluvia ya que se opera en frecuencias no demasiado elevadas. En este rango se encuentran las bandas desde la ELF que va de 3 a 30 Hz, hasta la banda UHF que va de los 300 a los 3000 MHz, es decir, comprende el espectro radio eléctrico de 30 - 3000000 Hz.

- Microondas terrestres: se utilizan antenas parabólicas con un diámetro aproximado de unos tres metros. Tienen una cobertura de kilómetros, pero con el inconveniente de que el emisor y el receptor deben estar perfectamente alineados. Por eso, se acostumbran a utilizar en enlaces punto a punto en distancias cortas. En este caso, la atenuación producida por la lluvia es más importante ya que se opera a una frecuencia más elevada. Las microondas comprenden las frecuencias desde 1 hasta 300 GHz

- Infrarrojos: se enlazan transmisores y receptores que modulan la luz infrarroja no coherente. Deben estar alineados directamente o con una reflexión en una superficie. No pueden atravesar las paredes. Los infrarrojos van desde 300 GHz hasta 384 THz.

2.1.4. Estándares y su Relevancia

Para el correcto funcionamiento e implementación de redes inalámbricas se crearon estándares de la IEEE para las especificaciones de comunicación de los protocolos, para ello se utilizo el estándar 802.11 que define los dos niveles inferiores del modelo OSI para definir el funcionamiento de una WLAN, dentro de los que es 802.x se define el funcionamiento de redes área loca y metropolitana.

En la figura II.2 podemos observar la evolución de los estándares al pasar los años, en la que cada protocolo ha brindado una solución diferente a las necesidades pertinentes de la red.

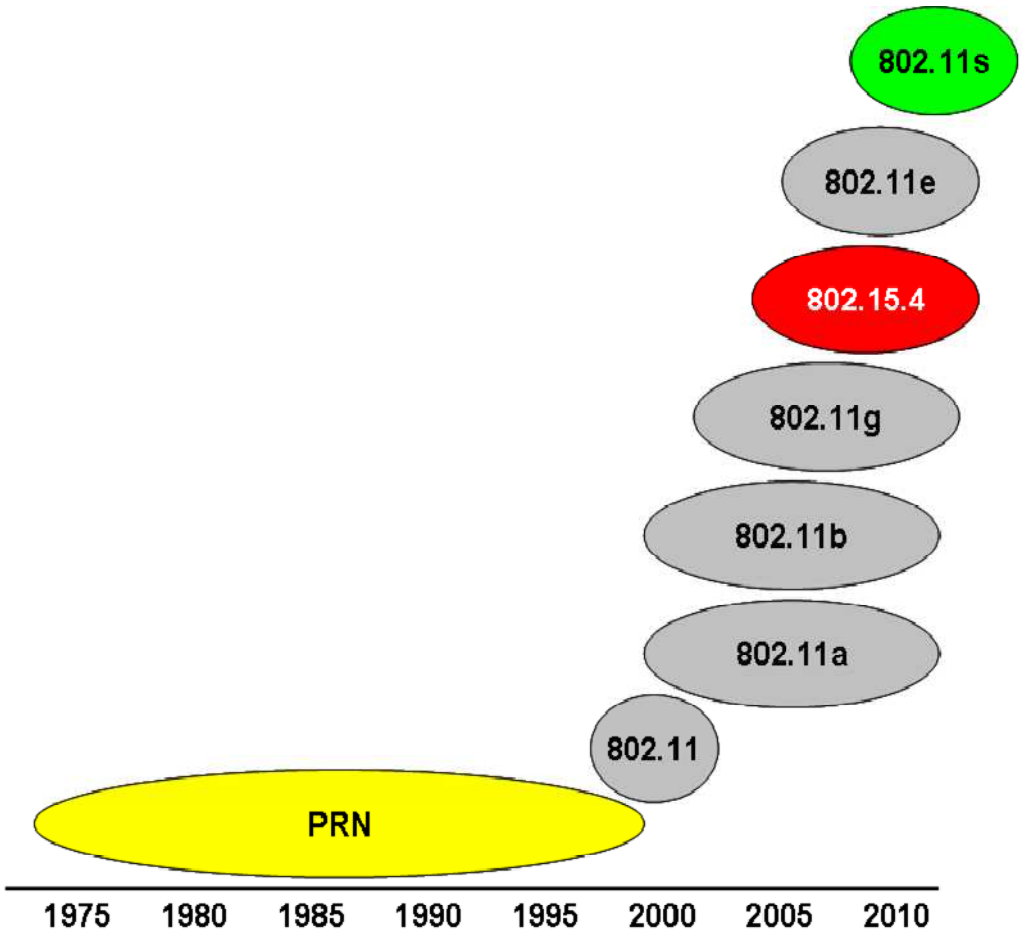


Figura II.2 Estándares en redes inalámbricas

802.11

La versión original del estándar IEEE 802.11 publicada en 1997 especifica dos velocidades de transmisión teóricas de 1 y 2 megabits por segundo (Mbit/s) que se transmiten por señales infrarrojas (IR). IR sigue siendo parte del estándar, si bien no hay implementaciones disponibles.

El estándar original también define el protocolo CSMA/CA (Múltiple acceso por detección de portadora evitando colisiones) como método de acceso. Una parte importante de la velocidad de transmisión teórica se utiliza en las necesidades de esta codificación para mejorar la calidad de la transmisión bajo condiciones ambientales diversas, lo cual se tradujo en dificultades de interoperabilidad entre equipos de diferentes marcas. Estas y otras debilidades fueron corregidas en el estándar 802.11b, que fue el primero de esta familia en alcanzar amplia aceptación entre los consumidores. [2][3][4]

2.2. MANET's

Las Manet's [6] son sistemas autónomos constituidos por nodos móviles que se comunican a través de enlaces inalámbricos de múltiples saltos. En términos más sencillos esto quiere decir que una red móvil ad hoc permite que una red se pueda establecer sin la necesidad de una administración central o de infraestructura preestablecida, ya que la red se conforma sólo de usuarios móviles capaces de transmitir y recibir información entre sí.

2.2.1. Redes AdHoc móviles

Las Redes Móviles AdHoc (MANET) fueron creadas para proporcionar comunicación y ser implementadas de una manera rápida y eficiente, en lugares carentes de una infraestructura de

red, puesto que son redes descentralizadas1. Sin embargo, para que esto sea posible se hace necesaria la introducción en la red de protocolos de enrutamiento específicos, debido a que los protocolos tradicionales propios de redes fijas no se adaptan a este tipo de ambientes móviles [7].

Puntos a considerar en una manet:

- Enrutamiento
- Redes ad hoc móviles inalámbricas distribuidas de múltiples saltos
- Total movilidad de los nodos
- Topología dinámica sin estructura previa
- Administración distribuida
- Funcionan gracias a la cooperación entre nodos
- Se realiza enrutamiento en cada nodo (problema no solucionado)

2.2.2. Efectos de la movilidad

La movilidad de los nodos tiene claras consecuencias en las comunicaciones establecidas a través de los enlaces en la topología. Cuando el usuario del terminal móvil cambia su posición, puede cambiar la forma en la que se establece el enlace entre los nodos, definiendo nuevas rutas para alcanzar los puntos deseados. [6]

Durante el intervalo de tiempo que dura este proceso, los paquetes destinados a un nodo y aquellos transmitidos por éste no pueden ser enrutados. Esto provoca pausas o latencias en la comunicación. [7]

Lo que se busca en una red móvil es satisfacer la necesidad de conexión a pesar de la topología contantemente cambiante, por ello el objetivo es permitir una auto-organización de lar de AdHoc. La auto-organización es un fenómeno que se puede apreciar claramente en muchos procesos de la naturaleza. En un cardumen de peces, por ejemplo, cada pez individual establece su comportamiento basado en la observación de la posición y la velocidad de sus vecinos más próximos. No existe una entidad central que dirija todo el cardumen. El mismo caso se presenta en un grupo de aves. Todas tratan de viajar a una misma velocidad sin colisionar entre sí y siguen un movimiento coordinado sin la presencia de un control central. De esta manera, se puede apreciar que en los sistemas que presentan auto-organización, el comportamiento sencillo de las entidades individuales conlleva a una organización sofisticada del sistema general. Las MANET se basan en este comportamiento y por esta razón se constituyen en una tecnología ideal para establecer comunicación en aplicaciones donde los usuarios son móviles. [8]

2.2.3. Aplicaciones prácticas

- Vanet

Las VANET (Vehicular Ad-Hoc Network) [9] son redes ad-hoc móviles capaces de comunicar información entre diversos vehículos colindantes y el sistema de tráfico.

El objetivo principal de estos sistemas es proporcionar un mejor conocimiento de las condiciones de las carreteras a los conductores para reducir así el número de accidentes y que la conducción sea más cómoda y fluida. Asimismo, estas redes permiten el acceso a contenidos multimedia e Internet, como la compartición de archivos entre diferentes vehículos.

Este tipo de redes, al ser una extensión de las redes ad-hoc móviles (MANET), supone los mismos desafíos aunque con diferencias sutiles.

- Aplicaciones militares

En caso de requerirse una red inalámbrica en la que no se puede depositar el funcionamiento de la red sobre un punto central como un punto de acceso inalámbrico, se requiere que cada nodo sea autosuficiente para comunicarse con los demás, al permitir que cada nodo actué como un enrutador se provee la capacidad de autosuficiencia, convirtiendo esta estructura descentralizada en una ventaja para aplicación militar en la que cada móvil, sea este un vehículo o persona, tiene la capacidad de comunicarse con la red siempre que esté al alcance de un nodo cercano que tenga capacidad de acceso a la red

- Operaciones de emergencia

En caso de necesitarse un red con la capacidad de desplegarse rápidamente sin tiempo para instalar un dispositivo central que brinde cobertura a la red o que no pueda cubrir la zona requerida por condiciones geográficas se vuelve una necesidad contar con una manet que permita que cada nodo actué independientemente a demás de brindar conectividad entre nodos distantes

- Otras aplicaciones

Gracias a la característica de red descentralizada se pueden encontrar numerosas aplicaciones para una red con la capacidad de brindar independencia a los nodos en especial en casos que se requiera un despliegue rápido de la red o que geográficamente no sea posible o eficiente instalar un dispositivo central.

2.3. Introducción a los protocolos de ruteo

2.3.1. ¿Por qué la necesidad de un protocolo de ruteo?

El encaminamiento de redes ad hoc [10] es una tarea primordial. Al tener nodos independientes con la capacidad de gestionar sus enlaces debe buscarse alcanzar la conectividad de extremo a extremo, evitando los lazos y buscando la mejor ruta.

Tomando en cuenta la movilidad de una red inalámbrica esta tarea puede volverse muy compleja para los nodos, por ello la necesidad y variedad de protocolos de enrutamiento que tienen la tarea de gestionar los enlaces de la red a través de los mismos nodos que estarán en constante movimiento en el tiempo de manera aleatoria lo que conlleva un cambio constante en las tablas de enrutamiento.

Los protocolos de ruteo comunes como OSPF o RIP no están diseñados para llevar a cabo estos cambios excesivos en las tablas de ruteo, ya que su eficiencia se debe a su comprensión de la red, pero cuando nos encontramos en una red ad hoc móvil, se requiere de protocolos diseñados con el fin de buscar la conexión de extremo a extremo buscando la mejor ruta del escenario móvil.

Las variaciones en la red darán como resultado que una ruta que se consideraba optima, en pocos segundos sea obsoleta, por ello se busca que el protocolo tenga una rápida capacidad de respuesta ante los cambios.

2.3.2. Algoritmos de enrutamiento

Como se observa al interior de una red de computadoras, para transferir información de un punto a otro se quiere de un protocolo que gestione la ruta entre los puntos. En el caso de la

arquitectura TCP/IP, esta tarea está confiada a la capa de Internet, en la cual se implementan los denominados algoritmos de enrutamiento, responsables de la determinación del camino seguido por cada paquete hasta alcanzar al destinatario. Igualmente, la responsabilidad de la red de conexión de internet (internetworking) [11] es delegada al protocolo IP. En Internet cada nodo es individualizado mediante una dirección IP, única en toda la red. Las direcciones IP se generan bajo la autoridad de la Internet Coorporation for Assigned Names and Numbers (ICANN), en base a las directivas impuestas por la RFC (Request for Comments) 2050 [25], que se organizan en muchas clases jerárquicas.

En particular, en Internet, cada datagrama IP transmitido lleva al interno la dirección IP del servidor remitente y del receptor: es, pues, tarea de los enrutadores hacer llegar el paquete al terminal de destino. La operación de tramitación de cada paquete viene llevada a cabo consultando la llamada tabla de enrutamiento. Una tabla de enrutamiento puede verse como una lista en la cual, a cada dirección de destinatario, le corresponde una puerta de salida hacia la que se transmite las informaciones. Tal lista se construye y se actualiza mediante un algoritmo de enrutamiento que implica el uso de protocolos y algoritmos entre más enrutadores.

Existen diversas tipologías de algoritmo de enrutamiento; aquellas más usadas en las redes cableadas tradicionales son [12]: Link State, Distance Vector, Source Routing,
Random e Flooding.

- Con el Link State se asigna un costo a cada link o conexión. Cada nodo administra un mapa completo de la topología de la red. Periódicamente cada nodo manda en broadcast (difusión) el costo de los enlaces a los cuales está conectado, y los restantes actualizan el mapa de la red y la tabla de enrutamiento aplicando un algoritmo que tiene en cuenta el camino a menor costo.

- En el Distance Vector cada nodo conoce ya el costo de los enlaces a los que está conectado. Cada nodo comunica con su vecino a que otros nodos pueden alcanzar y a qué costo. Así cada nodo re calcula la propia tabla de enrutamiento siguiendo las informaciones que ha recibido, y usando un algoritmo que tiene en cuenta, por ejemplo, el camino a menor costo.
- Con el Source Routing, las decisiones pertinentes al router vienen tomadas de la fuente y los paquetes de información siguen un camino ya establecido.
- El direccionamiento Random es de tipo casual ya que la rama de salida del nodo, a menos que el servidor destinatario del paquete no esté directamente conectado al nodo en cuestión, viene elegida casualmente. De este modo, sin embargo, el algoritmo garantiza una utilización óptima de los recursos de la red, ya que goza de la simplicidad de implementación y gestión.
- Finalmente, en el Flooding sucede que cada paquete de información recibido viene transmitido y replicado sobre todos los enlaces salientes, a menos que la dirección de destino no sea un servidor directamente conectado al mismo nodo.

2.3.3. Protocolos de ruteo

Los protocolos de ruteo para las redes inalámbricas como podemos ver en la figura II.3 se clasifican en varias categorías dentro de las cuales tenemos protocolos estandarizados de propósitos generales y otros más específicos para cada necesidad de acuerdo a propósitos específicos, de manera general se pueden clasificar en tres categorías[13]:

- Proactivos

- Reactivos
- Híbridos

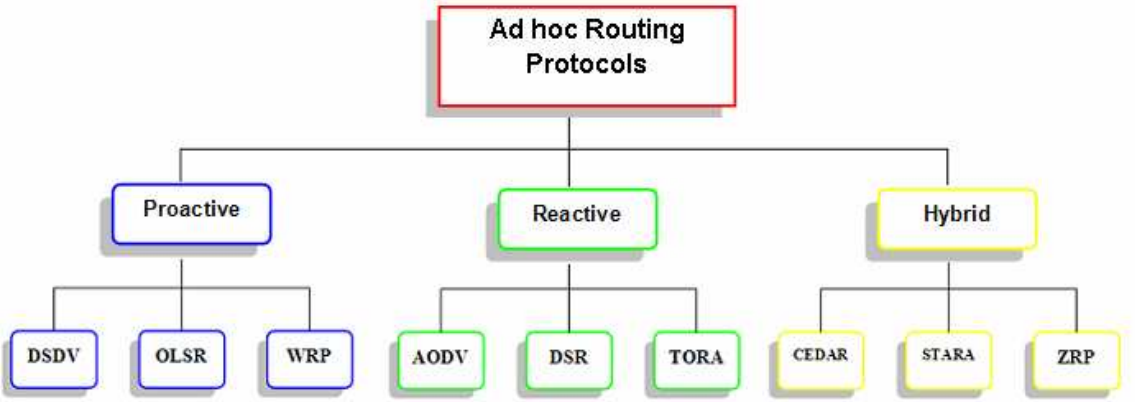


Figura II.3 Protocolos de ruteo para comunicación inalámbrica

Aquellas de tipo proactivos mantienen constantemente actualizados la información de direccionamiento a través de intercambios de paquetes a intervalos temporales fijos. Esto permite tener un direccionamiento disponible a cada petición de enrutamiento, pero está la desventaja de producir tráfico de señalización incluso cuando no se transmite ningún paquete de datos; esto puede provocar sobrecarga en la red.

En los protocolos de tipo reactivos viene invocado un procedimiento para determinar el correcto direccionamiento sólo en el momento en el que el paquete deba efectivamente transmitirse. De este modo, se reduce el tráfico de señalización en detrimento de un aumento de los tiempos de entrega.

El tercer tipo de protocolos, híbridos, busca, como dice su nombre, de unir las ventajas de ambos protocolos precedentes, limitando la aplicación de algoritmos proactivos sólo a los nodos adyacentes del que quiere transmitir.

En el caso en el que se encuentren redes no dotadas de una infraestructura como por ejemplo, las redes ad-hoc, se hacen necesarios los algoritmos de enrutamiento adecuados en los que hay que tener en cuenta la característica dinámica de tales sistemas (en las redes inalámbricas ad-hoc, los nodos podrían moverse modificando así la topología de la red). En este tipo de redes, cada nodo debe tener la capacidad de un router.

A continuación se describen algunos de los algoritmos de direccionamiento más usados en las redes inalámbricas ad-hoc:

AODV (Ad-hoc On Demand Vector),
AOMDV (Ad-hoc On Demand Multipath Distance Vector)
DSR (Dynamic Source Routing),
OLSR (Optimized Link State Routing)

- AODV (Ad-hoc On Demand Vector) [14]

El protocolo AODV es un protocolo de enrutamiento de tipo reactivos basado en el algoritmo Distance Vector.
Una característica fundamental del protocolo es que los nodos destino de un trayecto, antes de proporcionar información de direccionamiento, crean un número de secuencia de destino (destination sequence number), que proporciona a los nodos un instrumento para evaluar cuanto se ha actualizado un determinado recorrido evitando la formación de lazos (loop) en el camino de enrutamiento. Un terminal que deba elegir entre varios caminos hacia un cierto destino, elegirá

aquel caracterizado por el número de secuencia mayor, correspondiente a una información de routing mas reciente.

Además, el protocolo soporta el enrutamiento multidifusión (multicast) [14]. Este protocolo usa mensajes particulares llamados RREQ (Route Request), RREP (Route Replies) y RERR (Route Errors) que son enviados y recibidos mediante el protocolo UDP.

Cuando un nodo quiere encontrar un camino hacia otro nodo de la red:

1) envía en broadcast un mensaje del tipo RREQ

2) atiende una respuesta del destinatario o, de otro nodo, que posee un camino de enrutamiento bastante reciente hacia aquel destino. Esta respuesta llegará con un mensaje de tipo RREP confirmando incluso que el camino buscado está disponible.

Por nodo que posea un camino de direccionamiento bastante reciente se entiende un nodo que conozca un trayecto asociado a un número de secuencia destino que sea de grande, al menos, como aquel contenido en el mensaje RREQ.

Además, los nodos de la red que forman parte de trayectos activos pueden transmitir periódicamente mensajes especiales de RREP, llamados mensajes “Hello”, a sus nodos más cercanos. La falta de mensajes “Hello” por parte de los nodos vecinos viene interpretada como pérdida de la conexión con ese nodo y hace que el nodo que debería haber recibido tal mensaje proceda a corregir su tabla de enrutamiento, eliminando aquel trayecto.

En la fase de extracción de la tabla de enrutamiento de una puerta de acceso a un nodo vecino, con motivo de que ya no es alcanzable, el nodo se preocupa de mandar un mensaje RERR a los nodos adyacentes que usaban el trayecto, informándoles del acontecimiento. Todo esto sucede

sin dificultad en cuanto a que cada nodo conserva una lista de los nodos cercanos que están activos en cualquier comunicación.

El procedimiento del mensaje RERR viene, por tanto, repetido por los nodos intermedios determinando así la actualización de las tablas de direccionamiento de todos los nodos de la red.

- AOMDV (Ad-hoc On Demand Multipath Distance Vector)

AOMDV comparte muchas características en su funcionamiento básico con AODV [15]. Se basa en el concepto de vector distancia y utiliza aproximación de salto en salto. Este protocolo a demás descubre rutas bajo demanda utilizando un procedimiento de descubrimiento de ruta. La principal diferencia es el número de rutas encontradas para cada descubrimiento de ruta. En AOMDV la propagación de RREQ de la fuente hacia el destino establece múltiples caminos en reversa junto con los nodos intermedios al igual que el nodo destino

Varios RREPS atraviesan estos caminos de retorno utilizando varias rutas. AOMDV también provee nodos intermedios con caminos alternos, de esta manera se puede reducir la frecuencia de descubrimiento de rutas.

El núcleo de AOMDV busca asegurar que se descubran múltiples caminos libres de lazos, además de la eficiencia al encontrar rutas mediante inundación de la red. Este protocolo actualiza las reglas aplicadas localmente a un nodo, juega un papel fundamental al mantener la red libre de bucles.

AOMDV confía en la información de ruteo disponible bajo los lineamientos de AODV, en general no emplea ningún control de paquetes, de hecho el envió de RREPS y RERRS para descubrir múltiples rutas y mantenerlas, junto con campos extra en los campos de ruteo constituyen la única sobrecarga para el protocolo

- DSR (Dynamic Source Routing)

El protocolo DSR [16], de tipo reactivos, se caracteriza por el uso del Source Routing y del mecanismo de tipo “On Demand”. En tal sistema el source routing hace que los nodos fuente conozcan “paso a paso” (hop by hop) el camino que deben efectuar para alcanzar al destinatario. Esto se lleva a cabo gracias a una memoria de enrutamiento (route cache) que memoriza todos los caminos a efectuar.

Si el nodo que quiere enviar un paquete informativo pertenece a una red inalámbrica ad-hoc, se inicia un proceso de Routing Discovery. Tal proceso consiste en el envío, por parte del nodo, de mensajes RREQ en Flooding sobre la red, mensajes que todos los otros nodos receptores enviarán a su vez en Flooding. En cambio, en el caso en el que el nodo sea el nodo destinatario o son nodos que tienen, en la propia memoria de enrutamiento, un trayecto válido, responden al mensaje RREQ, transmitiendo al nodo solicitante un paquete RREP. Habitualmente, este último sigue un camino inverso respecto al del RREQ y mantendrá toda la información de direccionamiento que se memorizará desde el nodo solicitante.

Por último, si una conexión se interrumpe, vienen notificados una serie de paquetes RERR de modo que todos los nodos actualicen su memoria de direccionamiento y no usen más ese

enlace. El protocolo DSR hace un uso intenso de la memoria de direccionamiento y de la fuente de direccionamiento para evitar los lazos (loop).

- OLSR (Optimized Link State Routing)

El protocolo Optimized Link State Routing (OLSR) [17] es un mecanismo estándar de enrutamiento pro-activo, que trabaja en forma distribuida para establecer las conexiones entre los nodos en una red inalámbrica ad hoc (mobile ad hoc networks, MANETs). Este protocolo fue diseñado en un principio por investigadores del Instituto Nacional francés de Investigación en Informática y Automática (INRIA, por sus siglas en francés), y ha sido posteriormente estandarizado por el Internet Engineering Task Force (IETF).

La diseminación directa de información por toda la red (flooding) es ineficiente y muy costosa en una red inalámbrica y móvil, debido a las limitaciones de ancho de banda y la escasa calidad del canal radio. Por ello, OLSR prové un mecanismo eficiente de diseminación de información basado en el esquema de los Multipoint Relays (MPR).

Bajo este esquema, en lugar de permitir que cada nodo retransmita cualquier mensaje que reciba (flooding clásico), todos los nodos de la red seleccionan entre sus vecinos un conjunto de multipoint relays (retransmisores), encargados de retransmitir los mensajes que envía el nodo en cuestión. Los demás vecinos del nodo no pueden retransmitir, lo que reduce el tráfico generado por una operación de flooding.

Hay varias formas de escoger los multipoint relays de un nodo, pero independientemente de la forma de elección, el conjunto de MPRs de un nodo debe verificar que son capaces de alcanzar

a todos los vecinos situados a una distancia de 2 saltos del nodo que los calcula (criterio de cobertura de MPR).

Una red enrutada con OLSR utiliza básicamente dos tipos de mensajes de control:

- Los mensajes HELLO son enviados periódicamente por cada nodo de la red a sus nodos vecinos, pero nunca son retransmitidos más allá del primer salto (1 hop) desde su emisor (alcance local). Estos mensajes contienen la lista de vecinos conocidos por el nodo emisor así como la identidad de los multipoint relays seleccionados por transmisor. Su intercambio permite a cada nodo de la red conocer los nodos situados a 1 y 2 saltos de distancia (es decir, aquellos a los que se puede hacer llegar un mensaje con una transmisión directa o con una transmisión y una retransmisión) y saber si ha sido seleccionado como MPR por alguno de sus vecinos.

- Los mensajes TC (Topology Control) son enviados periódicamente y de forma asíncrona. A través de ellos, los nodos informan al conjunto de la red acerca de su topología cercana. Al contrario que los HELLO, los mensajes TC son de alcance global y deben llegar a todos los nodos de la red. El conjunto de los mensajes TC recibidos por un nodo inalámbrico le permite reconstruir su base de datos topológica, computar el árbol de caminos mínimos (mediante el algoritmo de Dijkstra) y calcular así la tabla de enrutamiento hacia todas las posibles destinaciones. La diseminación de mensajes TC se hace de acuerdo con el mecanismo de flooding basado en MPR. [18]

2.3.4. Perspectiva global del encaminamiento

El objetivo principal del encaminamiento en Redes Ad Hoc Inalámbricas es encontrar rutas óptimas en relación con un parámetro o conjunto de parámetros determinado. El nivel más básico de estos parámetros lo ocupa la capacidad para construir un camino lo más cercano a la línea recta que une fuente y destino. Esta capacidad es denominada en la presente tesis eficiencia de encaminamiento y engloba las contribuciones de los distintos factores que intervienen en el mismo.

Esta eficiencia de encaminamiento no debe circunscribirse exclusivamente al protocolo que gobierna la operación de los nodos en lo concerniente a la búsqueda y transporte de la información mediante distintos saltos. Para lograr una descripción completa del encaminamiento, es necesario un modelo que tenga en consideración factores que no siempre están encuadrados en la capa de red (RED) sino que pueden encontrarse también en las capas PHY y MAC. El hecho de obviar estas variables genera dos efectos indeseados: por una parte, la representación resultante es incompleta ya que no incluye las contribuciones de todos los parámetros que intervienen en el mismo. Además, no se permite la extracción de conclusiones separables sobre la influencia del conjunto de factores incluidos, ya que sus efectos se presentan confundidos entre otros procedentes de variables no contempladas por el modelo. Por estos motivos, el estudio del encaminamiento toma como base el protocolo de encaminamiento y su objetivo principal de encontrar una ruta lo más cercana posible a la línea recta que une fuente y destino.

No obstante, debe considerar como una parte integrante del mismo las restricciones impuestas por el resto de variables involucradas en el proceso global del encaminamiento. [19]

		PHY						MAC		RED					
		Pot. TX	Topología	Desvanecim.	Movilidad	BER	Cons. Energ.	Interferencia	Nº ReTX	Multisalto	Tasa transf.	Retardo	Equidad	Seguridad	Encaminam
PHY	Pot. TX														
	Topología														
	Desvanec.														
	Movilidad		x												
	BER	x	x	x	x										
	Con. Ener.	x			x										
MAC	Interfer.	x	x		x	x									
	Nº ReTX	x			x	x	x	x							
RED	Multisalto	x		x	x	x	x	x							
	Tasa trans		x		x	x		x	x						
	Retardo		x		x				x	x					
	Equidad										x	x			
	Seguridad	x								x					
	Encamin.	x	x	x	x	x	x	x	x	x	x	x	x	x	

Tabla II.I Interrelaciones entre parámetros de las capas PHY, MAC y RED

2.3.5. Consumo de energía en redes inalámbricas

Los dispositivos móviles tienen muchas restricciones y limitaciones en relación al consumo de la energía en comparación con los dispositivos de una red cableada. La conservación de la energía de ambientes inalámbricos es un asunto importante que debe tomarse en cuenta. Las redes de este tipo están expuestas a muchos factores que contrarrestan el uso óptimo de la energía, tales como la continúa comunicación entre los dispositivos, alojamiento de recursos y memoria, el uso eficiente de la batería de alimentación, el tráfico, etc. Todos estos factores juntos disminuyen la energía necesaria para el buen desempeño y comunicación en una red inalámbrica.

El uso eficiente de recursos computacionales no es un tema reciente, sus orígenes datan a inicios de 1992 cuando la Agencia de Protección Ambiental (www.epa.gov) y el Departamento de Energía (www.energy.gov) de los Estados Unidos (EUA), conjuntamente, promulgaron reconocer los esfuerzos en el uso óptimo de energía de diferentes dispositivos electrónicos a través del programa conocido como Energy Star (Estrella de Energía). El uso eficiente de recursos

energéticos tiene como objetivos la viabilidad económica, es decir que la tecnología sea económica; responsabilidad social mediante la construcción de tecnología que contribuya a minimizar los problemas de consumo irracional de energía y minimizar el impacto en el ambiente.

En el contexto del cómputo móvil, redes inalámbricas (Wi-Fi, bluetooth) y telefonía inalámbrica, se han realizado múltiples esfuerzos por optimizar el uso de recursos, tales como el caudal de comunicación y la energía. De mayor interés ha sido la optimización en dispositivos con alimentación limitada o nula de energía, dado que sus aplicaciones se extienden al campo militar, doméstico y comercial. Ejemplo de esto han sido la telefonía celular, PDAs (Personal Digital Assistant), Manet's y las redes Ad Hoc, cuya utilidad en sus inicios ha sido restringida por la capacidad limitada de almacenamiento de energía y mínimas optimizaciones para su administración. En este artículo nos enfocaremos a describir el problema de consumo de energía en medios inalámbricos. Primero iniciamos ilustrando que tipo de eventos son los responsables del mayor consumo energético en un proceso de comunicación, específicamente en la interfaz de red. Posteriormente describimos algunas heurísticas o propuestas que se han desarrollado en la capa de transporte, del modelo de referencia OSI, con el objetivo de administrar adecuadamente la utilización de la energía.

Una de las primeras investigaciones en el campo de la conservación de la energía fue liderada por los científicos Mark Stemm y Randy Katz en 1997. Stemm y Katz pertenecientes al departamento de Ingeniería Eléctrica y Ciencias Computacionales de la Universidad de Berkeley, a través de su publicación "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices" encontraron que interfaces de comunicación consumen una proporción considerable de energía al estar en un estado de ocio. En particular, mostraron que la proporción de consumo de energía es de 1:1.05:1.4 en términos de transmisión recepción ocio

correspondientemente. En otras palabras, una interfaz de comunicación consume un número menor de miliWatts (mW, 1x10-3 watts) durante la recepción de información en comparación con el consumo realizado durante el estado de ocio e inicialización de la interfaz de red. Otras investigaciones similares soportan las mismas conclusiones, sin embargo, Stemm y Katz han sido ampliamente referenciados y reconocidos en el área. Probablemente la mayor contribución de las anteriores investigaciones, yace en el hecho que el consumo de inicialización es mucho mayor en comparación con el realizado durante el estado de ocio. Por ejemplo, la interfaz de comunicación de un dispositivo WLAN operando en la banda de 915 MHz consume 177.3 mW en ocio y 1318 mW en inicialización, el mismo dispositivo operando en 2.4 GHz consume 143 mW en ocio y 1148.6 mW en inicialización. Los dispositivos tipo PDA, requieren aproximadamente 164 mW en ocio y 1187 mW en inicialización. Una laptop requiere en términos generales aproximadamente 8000 mW para la inicialización de su interfaz de comunicación.

A partir de estas observaciones, fabricantes e investigadores iniciaron estudios buscando determinar formas de minimizar el número de veces que una interfaz de comunicación sea inicializada y extender el tiempo durante el estado de ocio, esto cuando el dispositivo de comunicación presente baja utilización. [20]

Para el cálculo de la potencia de transmisión en NS2 se procede de la siguiente manera:

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

Donde Gt y Gr son so las ganancias de las antenas del transmisor y receptor respectivamente, ht y hr son las alturas de las antenas.

RXThresh es el valor mínimo de potencia que debe llegar al receptor para poder recibir el paquete, y coverage es la variable que contiene la cobertura deseada.

De esta manera se calcula Pt que será la potencia utilizada para la transmisión.

2.3.6. Heurísticas para el uso eficiente de los recursos.

No solo las redes inalámbricas (MANET's y Ad Hoc) se ven limitadas por la disparidad de consumo de energía en la interfaz de comunicación, sino también porque la heurística de acceso al medio es por inundación (CSMA/CA, ALOHA, 802.11, etc.), lo cual genera administración innecesaria de paquetes en dispositivos no destinatarios o que no se encuentren en una ruta de comunicación hacia un nodo destinatario. Por tal motivo, en el contexto de las redes inalámbricas, las optimizaciones han sido enfocadas a minimizar el número de dispositivos activos y de mensajes transmitidos por difusión, tomando como criterios de calidad el mantenimiento de conectividad y QoS en los flujos de comunicación.

Minimizar el número de dispositivos activos se logra a través de la elección de un conjunto dominante de comunicación, es decir, un subconjunto mínimo de dispositivos que actúen como enrutadores los cuales cuenten con suficiente capacidad de almacenamiento de mensajes y energía. Es importante que un dispositivo enrutador cuente con suficiente capacidad de almacenamiento temporal, ya que en ocasiones un flujo de información debe ser almacenado temporalmente dado que el dispositivo destinatario puede encontrarse en estado de ocio o el

flujo de comunicación es interrumpido por cambios en la topología de comunicación. La limitación de mensajes transmitidos tiene como consecuencia reducir el número de mensajes retransmitidos por dispositivos no destinatarios, minimizando así la aparición de mensajes duplicados y la necesidad de implementar estrategias para eliminar información redundante. Los protocolos de comunicación que se han visto impactados por las anteriores metas son los de enrutamiento. Tal como el Destination Séquense Distance Vector (DSDV), Ad Hoc On-Demand Distance Vector (AODV), Dynamic Source Routing (DSR), por mencionar algunos. No solo los protocolos de enrutamiento requieren emplear esquemas para la optimización de energía, si no, cualquier protocolo que requiera operaciones constantes de diseminación; por ejemplo, para mantener coherente tablas de enrutamiento, mantenimiento de un estado global, QoS, o comunicación en grupo.

Las redes inalámbricas tienen múltiples aplicaciones prácticas en medios comerciales, escenarios de emergencia, monitoreo y el hogar. Si tal infraestructura no cuenta con fuentes continuas de suministro de energía los protocolos de conservación de energía son de vital importancia ya que permitirán extender el uso de la red. Los retos actuales para el diseño de protocolos de conservación de energía incluyen: proveer heurísticas que extiendan su funcionalidad a través de la pila de protocolos, tal que generen una transición al estado modo pasivo MAC 802.11. Entre otros retos se tienen: QoS, preservar la fidelidad de enrutamiento y regular la potencia de transmisión con el objetivo de maximizar el uso del canal de comunicación.

Como previamente se ha discutido, múltiples esfuerzos se han realizado para optimizar el uso de energía en la capa MAC, estableciendo mecanismos rígidos de administración a conciencia de aspectos físicos, tal como la utilización óptima del medio de comunicación, minimización de contención, por mencionar algunos parámetros a optimizar. Sin embargo, nuevas heurísticas están siendo desarrolladas en niveles superiores de la jerarquía de protocolos, tales, toman en

consideración reducir flujos redundantes de comunicación y elección de un conjunto mínimo de comunicación. Podemos esperar, que nuevas heurísticas se acerquen más a la capa de usuario y que los patrones y necesidades de comunicación sean los criterios determinantes para administrar los recursos energéticos de los dispositivos de comunicación. [20][21]

2.4. Simulador NS2

2.4.1. Principios básicos.

Probablemente Network Simulator 2 sea el simulador de redes en software libre más extendido para propósitos de investigación y docentes. Es un software para simulaciones de red pilotado por acontecimientos y orientado a objetos.

El simulador Network Simulator (NS) [22], está concebido para el estudio del comportamiento de una red cualquiera de telecomunicaciones. Se basa en el protocolo IP, implementando protocolos de transporte como el TCP y el UDP, protocolos para la gestión de la congestión, mecanismos y procedimientos para la gestión de las colas como DropTail, RED (Random Early Detection), además de algoritmos de enrutamiento, de multicasting y finalmente, algunos protocolos MAC para la simulación de redes LAN, WLAN y módulos para probar arquitecturas DiffServ.

Al ser código abierto, este, está disponible íntegramente en todas sus versiones, que puede ser modificado en agrado de la comunidad científica que hace empleo de ello, siempre que se respetan las cláusulas de la licencia que acompaña el código. El software puede ser descargado libremente por Internet y tiene que ser compilado antes de poder usarlo; el programa gira

principalmente sobre plataformas respaldadas por Unix y, en particular, por Linux, aunque puede ser ejecutado en plataformas Windows mediante Cygwin.

NS ha sido desarrollado principalmente para dos diversas categorías de usuarios: aquellos interesados sencillamente en las simulaciones y a los usuarios interesados en el desarrollo de nuevos códigos y objetos. De aquí que trabaje con dos distinguidos lenguajes de programación: OTcl y el C++.

OTcl es un lenguaje interpretado, simple e intuitivo, que deriva del Tcl (Tool Command Language) y con la extensión Orientado a Objeto. Generalmente es utilizado en la redacción de los script por la programación de los acontecimientos que definen los escenarios de simulación. Por programación de los acontecimientos se entienden, por ejemplo, mandos para el encendido y el apagado de los manantiales de tráfico, y mandos para la definición de la topología de la red simulada y para la conexión de los varios objetos.

2.4.1.1. NS2

- Diseño

NS fue construido en C++ y proporciona una interfaz de simulación a través de OTcl, un dialecto orientado a objetos de Tcl. El usuario describe una topología de red escribiendo scripts Otcl y, a continuación, el programa principal de NS simula la topología con los parámetros especificados.

- Historia

NS comenzó a desarrollarse en 1989 como una variante del simulador de red REAL. En 1995, el simulador había ganado el apoyo de DARPA(Defense Advanced Research Projects Agency), el proyecto Vint de LBL, Xerox PARC, UCB y USC/ISI.

NS ahora es desarrollado en colaboración entre una serie de investigadores e instituciones, incluida la SAMAN (con el apoyo de DARPA), CONSER (a través de la NSF), y ICIR (antes ACIRI). Sun Microsystems y la UCB Daedelus y Carnegie Mellon (citado por la página de inicio de ns por la adicción de código wireless), también han aportado grandes contribuciones. [23]

El C++ es el lenguaje compilado con el que se implementa en NS la mayor parte de los modelos de los objetos de red utilizados durante las simulaciones. Siendo un lenguaje compilado, tiene la ventaja de ser más veloz y eficiente, aunque precisa de una mayor complejidad de escritura.

Para utilizar el simulador se define un script OTcl que define el escenario de red a simular. Cuando es encaminada la simulación, los comandos en OTcl inicializan los objetos internos en C++ que hacen partir la acción requerida. Durante una simulación se inicializan y trabajan, al mismo tiempo, dos jerarquías de objetos, el OTcl y el C++. Entre los objetos de las dos jerarquías existe una correspondencia uno a uno y son visibles a ambos lenguajes. Esto permite separar la implementación de las funciones relativas a la simulación de los datos en la red de la implementación de la lógica de control y configuración de la red. La interfaz OTcl/C++ es realizada por el tclcl como podemos observar en la figura5.

El usuario observa este proceso como se muestra en la figura II.4 Desde una perspectiva en la que se genera el script llamando a la simulación de ns2 así como al animador de la simulación. [22]

Figura II.4 Arquitectura estructural de NS2

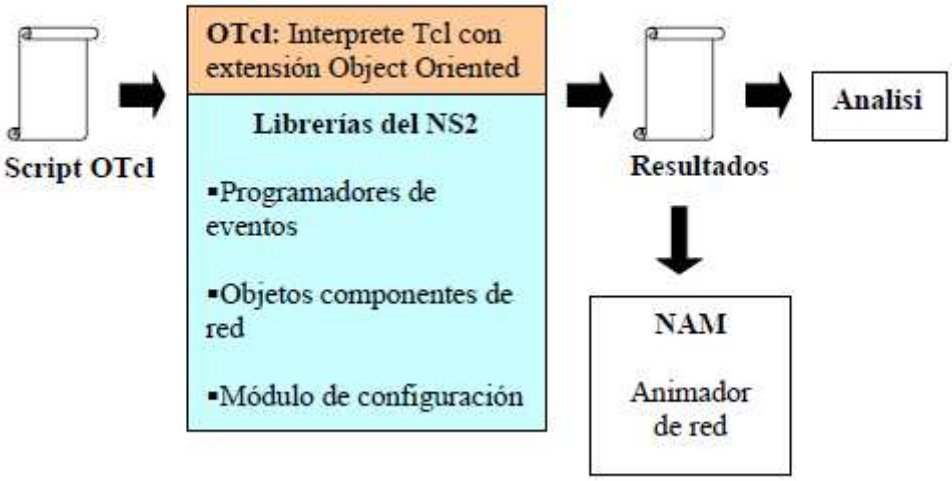


Figura II.5 Ejecución vista por el usuario del simulador en su conjunto

Entre los usos más habituales que les puedes dar a este tipo de simuladores se encuentran:

- Simular estructuras y protocolos de redes de todo tipo (satélite, inalámbricas, cableadas, etc.)
- Desarrollar nuevos protocolos y algoritmos y comprobar su funcionamiento.
- Comparar distintos protocolos en cuanto a prestaciones.

El código de Ns se ofrece bajo la versión 2 de la GNU General Public License [24][25]

2.4.1.2. Tcl

Tcl es un lenguaje de script creado por John Ousterhout, que ha sido concebido con una sintaxis sencilla para facilitarse su aprendizaje, sin ir en desmedro de la funcionalidad y expresividad.

Se utiliza principalmente para el desarrollo rápido de prototipos, aplicaciones "script", interfaces gráficas y pruebas. La combinación de Tcl con Tk (del inglés Tool Kit) es conocida como Tcl/Tk, y se utiliza para la creación de interfaces gráficas.

Tcl es un lenguaje interpretado, y su código puede ser creado y modificado dinámicamente. Sus reglas sintácticas son extremadamente simples y posee reglas de alcance dinámico. Permite escribir código fácil de mantener. Los "scripts" Tcl son a menudo más compactos y legibles que los programas funcionalmente equivalentes en otros lenguajes de programación. Es un lenguaje multiplataforma, con intérpretes que se ejecutan sobre Windows, Linux, UNIX, MacOS y OSX e incluso microprocesadores PIC.

Todos los elementos de un programa son comandos, incluyendo las estructuras del lenguaje. Dichos comandos se escriben en notación polaca y pueden ser redefinidos o sobrescritos de manera dinámica.

Una característica notable es que los datos son manejados como cadenas de caracteres Unicode, incluyendo el código fuente, soportando Unicode desde el lanzamiento de la versión 8.1, en el año 1999.

Una de las características más usadas de Tcl es su extensibilidad. Por ejemplo, si una aplicación requiere algo de funcionalidad no ofrecida por el Tcl estándar, los nuevos comandos de Tcl pueden ser implementados usando el lenguaje C, un integrado sumamente fácil. Tcl es

"extensible" a través de C, C++ y Java. Mediante una extensión, permite la programación orientada a objetos. Puede extenderse también a entornos gráficos, a través de una interfaz denominada Tk.

La programación orientada a eventos se realiza sobre "sockets" y archivos, además son posibles los eventos basados en tiempo y los definidos por el usuario.

El lenguaje Tcl fue originalmente proyectado para ser un lenguaje de comando reutilizable. Quienes desarrollaron Tcl estaban creando una serie de herramientas interactivas, y cada una constaba de su propio lenguaje de comando. Desde que comenzaron a interesarse más en estas herramientas que en los lenguajes de comandos que utilizarían, estos lenguajes comenzaron a construirse rápidamente sin considerar el diseño apropiado, sin mucha importancia.

Después de implementar varios lenguajes de comandos creados de esta forma y experimentar problemas con cada uno de ellos, decidieron concentrar su atención en la implementación de un objetivo general: un lenguaje de comando eficaz que pudiera ser integrado fácilmente en nuevas aplicaciones. Es de esta manera que nace el lenguaje Tcl (Tool Command Language), cuyas siglas en Inglés significan Lenguaje de Comando de Herramientas.

Desde ese entonces, el lenguaje Tcl ha sido utilizado como lenguaje de código. En muchos casos, Tcl es usado en combinación con la biblioteca Tk ("Tool Kit"), un conjunto de comandos y procedimientos que hacen relativamente fácil para programar interfaces de usuario gráficas.

Desde que Tcl comenzó a ser un lenguaje fácilmente extensible, se han escrito muchas extensiones para tareas determinadas, y están generalmente disponibles libremente en Internet. [26]

2.4.1.3. Tk

Tk es una aplicación libre multiplataforma y un conjunto de controles (widget toolkit). Es una biblioteca de elementos básicos para construir una interfaz gráfica de usuario (GUI).

Tk fue desarrollado por John Ousterhout como una extensión para el lenguaje de guiones (script) Tcl. También llamado "bindings", Tk puede ser usado por otros lenguajes como Perl, Python, y Ruby. Hay dos formas de usar Tk desde Perl: el módulo Tcl/Tk Perl que usa Tcl como un puente (este acercamiento proporciona más flexibilidad), y Perl/Tk tiene solo (solamente las extensiones adoptadas de Tcl/Tk disponibles). Python y Ruby también usan Tcl como puente para Tk.

Tk ha sido portado para correr en la mayoría de las variantes de Linux, Apple Macintosh, Unix, y Windows. Desde el Tcl/Tk 8, ofrece "native look and feel" (por ejemplo, los menús y botones son mostrados de forma “nativa” en cualquier plataforma). También, hay varias extensiones que proveen externamente arrastrar y soltar (drag-and-drop), ventanas no-rectangulares y controles originales.

La más inusual característica de Tk son sus controles canvas y texto, los cuales proveen capacidades no halladas en casi ningún conjunto de controles similares.

Al igual que Tcl, Tk soporta Unicode dentro del Plano Multilenguaje Básico pero este todavía no ha sido extendido para manejar Unicode de 32-bit. [27]

2.4.2. Modelos de WLAN y MANET's en NS2

Característica fundamental de las redes WLAN y en general de las LAN, que comparten el medio de transmisión, es que todos los nodos de la red tienen que poder escuchar el canal para poder recibir los datos a ellos destinados. Para obviar esto, NS2 pone a disposición un particular tipo de nodo llamado LanNode [43] y usa la pila de red de la figura 5.3. La estructura de cada nodo viene constituida por diferentes objetos que simulan de manera independiente los tres niveles más bajos del protocolo de red, es decir, el LL (Link Layer, nivel de enlace), el MAC y el PHY (PHYsical, nivel físico). Un genérico paquete generado por los niveles superiores, a través de la pila, se manda a la capa de enlace formado por los objetos Colas y LL, y sucesivamente se pasa al MAC que sigue las reglas apropiadas para hacer de interfaz sobre el canal físico. Este último, finalmente, se encarga de simular la propagación y hacer llegar, según los tiempos oportunos, los paquetes a todos los nodos en escucha sobre el mismo medio. Llegado a destino, el paquete recorre desde el nodo receptor el camino inverso hasta remontar al nivel de aplicación.
A nivel MAC se ejecuta la filtración de los paquetes, que deben ser presentados a los niveles superiores, entre todos los que transitan en el canal. Además, el intercambio de los paquetes de un nivel a otro es consentido por las funciones públicas que utiliza cada objeto implementado por una clase C++, para poder hacer de interfaz con los otros objetos. A la creación de un nuevo nodo viene asociada una nueva instancia de cada una de las clases necesarias; solamente el nivel físico es representado por una sola instancia común a todos los nodos.

Para el modelado de las redes inalámbricas multinodos ad-hoc se utiliza el nodo MobileNode [43], un modelo desarrollado en el ámbito del proyecto MONARCH de la universidad Carnegie Mellon [11] y sucesivamente englobado en NS2 como una extensión de NS. La implementación

del modelo es visualizado en la figura 5.4, dónde el objeto Src/Sink se ocupa de generar o consumir los paquetes de datos. [28]

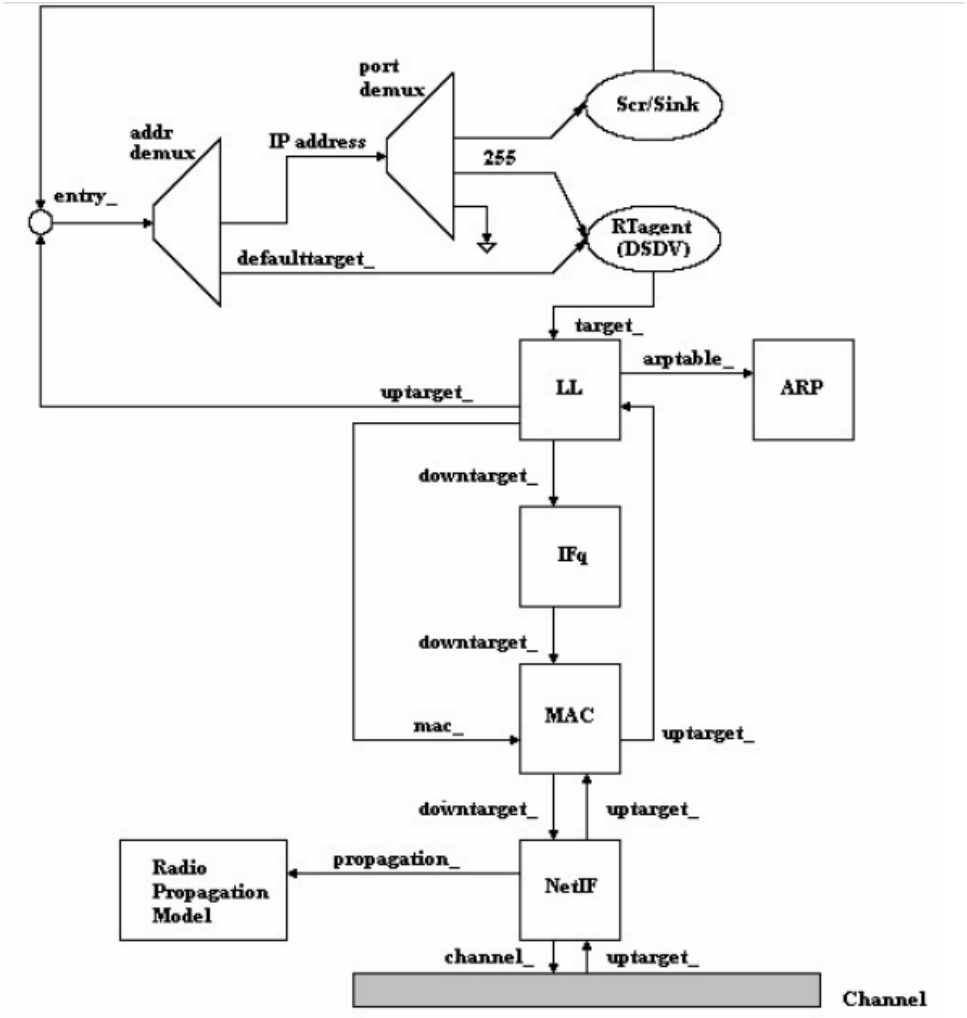


Figura II.6 Comunicación inalámbrica en NS2

Como se puede observar en la figura II.6 El objeto LL usado por los nodos móviles es muy parecido a aquel usado por los nodos no móviles a excepción del módulo ARP (Address Resolution Protocol) que convierte las direcciones IP en direcciones MAC, escribiéndolas en la

cabecera de la trama MAC. El objeto ARP puede generar ARP query (preguntas ARP) que son enviadas en difusión en el caso en que se tengan direcciones desconocidas. El objeto LL recibe los paquetes del objeto RTagent que implementa uno de los protocolos de enrutamiento, y los envía al objeto IFq, que constituye una particular cola a prioridad (priority queue) de interfaz entre el MAC y el LL, que privilegia los paquetes generados por los protocolos de routing poniéndolos en cabeza a la cola. El objeto MAC tiene la tarea de simular el estándar IEEE 802.11. En particular es simulado el protocolo DCF de las 802.11, usando tanto el physical como el virtual carrier sensing. Como estándar se utiliza el modelo RTS-CTS-FECHA-ACK para todos los paquetes unidireccionales, y se mandan directamente los datos para todos los paquetes de tipo broadcast (difusión). El objeto NetIF (Network InterFaces) actúa de interfaz hardware para el acceso al canal aproximando una interfaz radio DSSS, colaborando con el Radio Propagation Model que hace uso de la fórmula de atenuación de Friss (1/d2 donde d es la distancia) para pequeñas distancias, y del modelo a dos rayos de propagación (considerando la tierra como un plano completamente reflectante) para grandes distancias. Gracias al uso de estos dos últimos conceptos, que se ocupan de insertar datos en cada paquete, como la potencia de transmisión, la longitud de onda, etc.; el objeto Radio Propagation Model, conectado a las Network Interfaces que están recibiendo algo, puede determinar si el paquete posee la potencia mínima para ser recibido, capturado o reconocido (carrier sense) por el nodo receptor [28].

2.4.3. Nam: Network Animator

Nam es una herramienta para visualización de trazas basadas en Tcl/Tk y trazas del mundo real, soporta topologías, animación a nivel paquetes y varias herramientas de inspección de datos. Permite crear un entorno grafico para las simulaciones realizadas en NS, además de

crear simulaciones mediante un entorno más sencillo y visual. Nam dispone de un editor grafico para que no sea necesario crear los esquemas de red y definir la simulación mediante líneas de código para simulaciones sencillas. [29]

El diseño detrás de nam fue concebido para permitir una animación capaz de mostrar grandes cantidades de datos y para ser escalable de manera que se lo pueda utilizar para representar distintos escenarios de red. Con este concepto nam fue diseñado para leer comandos de eventos de simples animaciones a partir de un gran archivo generado de trazas. Para manejar grandes animaciones el manejo de datos requiere de un monto de información alojado en memoria.

Esta herramienta permite representar gráficamente la red diseñada como se muestra en la figura II.7, además permite visualizar dinámicamente los resultados de la simulación realizada.

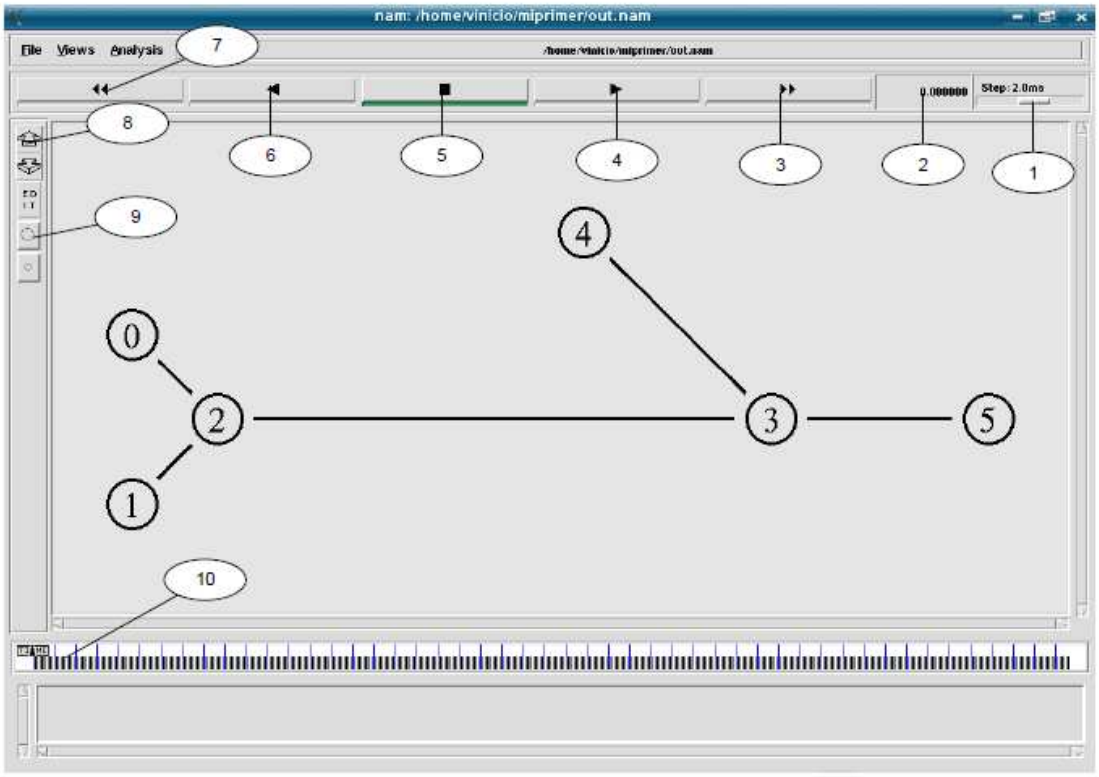


Figura II.7 NAM

2.4.3.1. Elementos de NAM:

Escala de tiempo (1): Muestra la equivalencia del tiempo del simulador con el tiempo real. En el ejemplo se muestra que 1 segundo en la realidad, representa apenas 2ms dentro de la simulación.

Tiempo (2): Indica el instante en el que se encuentra la simulación, se encuentra expresado en segundos.

Avance rápido (3): Aumenta la escala de tiempo en un factor de 25, lo cual genera un incremento de la velocidad en que se ejecutan los eventos dentro de la simulación.

Avance normal (4): Se inicia la animación o la hace continuar si estaba detenida de acuerdo a la escala de tiempo elegida.

Stop (5): Detiene la simulación.

Retroceso normal (6): Hace que los eventos dentro de la simulación retrocedan de acuerdo a la escala de tiempo elegida.

Retroceso rápido (7): Hace que los eventos dentro de la simulación retrocedan con una escala multiplicada por 25, de acuerdo a la escala de tiempo elegida.

Zoom (8): Permite alejar o acercar las imágenes de la simulación para tener una mejor visualización.

Tamaño de los nodos (9): permite variar el tamaño de los nodos que se encuentran dentro de la simulación.

Indicador de tiempo (10): Está compuesta por una barra de tiempo lineal que muestra el momento en que se encuentran ejecutándose los eventos durante el periodo de simulación. [30]

El primer paso para usar nam es crear el archivo trace. Este contiene la información de la topología como son los nodos, enlaces y paquetes. Usualmente, el archivo trace se genera al ejecutar la simulación en Ns. Durante una simulación Ns, el usuario puede crear la configuración de la topología, la información de capas y el comportamiento de paquetes mediante eventos, sin embargo otras aplicaciones pueden generar un archivo trace para nam

Cuando el archivo trace se genera, está listo para ser animado en nam, Al iniciarse, nam leerá este archivo, creara la topología, iniciara una ventana, creara capas de ser necesario, y pausará el tiempo en la simulación en 0 s. A través de la interface del usuario, nam provee el control de varios aspectos de la animación. Estas funcionalidades serán descritas mas tarde.

Existen errores en nam sin embargo cada versión se ha vuelto más estable que la anterior

Opciones de línea de comandos

Nam [-g \<geometry\>] [-t \<graphInput\>] [-i \<interval\>] [-j \<startup time\>]
[-k \<intial socket port number\>] [-N \<application name\>] [-c \<cache size\>]

[-f \<configuration file\>] [-r initial animation rate]

[-a] [-p] [-S]

[\<tracefile(s)\>]

-g Especifica la geometría de la ventana al inicio del programa.

-t Da la instrucción a nam de usar tkgraph, y especifica que la entrada sea tkgraph.

-N Especifica que el nombre de la aplicación en esa instancia. Este nombre de aplicación puede ser utilizado mas tarde en sincronización de puntos.

-c Tamaño máximo de cache usada para guardar objetos activos cuando se realiza una animación en reversa.

-f Nombra los archivos de inicialización para ser cargados durante el inicio. En este archivo el usuario define funciones las cuales serán llamadas en el archivo trace.

-a Crea una instancia separada de nam.

-p Imprime el formato del archivo nam trace.

-S Activa el comportamiento síncrono X así es más fácil depurar el programa. Para un sistema UNIX que corra únicamente X.

<tracefile> Es el nombre del archivo que contiene la información trace que será animada. Si este no puede ser leído, nam intentara abrir <tracefile>.nam. [29]

2.4.3.2. Objetos de animación

Nam realiza animaciones usando los siguientes bloques de construcción los cuales se definen a continuación.

- **Nodo**

  Nodos son creados por eventos en el archivo trace. Representa una fuente, host o enrutador. Nam no tomara en cuenta definiciones duplicadas para el mismo nodo. Un nodo puede tener 3 diferentes formas geométricas (círculo, cuadrado o hexágono), pero una vez creado no puede cambiar su forma.

  Los nodos pueden cambiar su color durante la animación y pueden ser etiquetados

- **Enlaces**

  Los enlaces se crean entre dos nodos de una topología de red. Internamente los enlaces nam están compuestos por dos link tipo simplex. El evento crea dos enlaces simplex y realiza la configuración. Sin embargo, para la perspectiva de ciertos usuarios todos los enlaces serán dúplex. Los enlaces pueden ser etiquetados y también puede cambiar su color durante la animación.

- **Cola**

  Las colas en nam se construyen a partir de dos nodos. Una cola en nam se asocia únicamente a un extremo de un enlace dúplex. Las colas se muestran como paquetes apilados. Estos se apilan en una línea, el ángulo entre la línea y el enlace puede ser especificado en el evento de generación de la cola.

- **Paquete**

Los paquetes son visualizados como un bloque con una flecha. La dirección de la flecha muestra la dirección del flujo del paquete. Paquetes a la cola se muestran como pequeños cuadrados. Un paquete puede ser desechado de una cola o un enlace. Paquetes desechados se muestran como cuadrados que caen mientras rotan, y desaparecen al final de la pantalla. Desafortunadamente, debido al diseño de nam los paquetes desechados no se visualizan durante una animación en retroceso.

- **Agente**

  Los agentes son usados para separar estados de protocolo de nodos. Siempre están asociados con nodos. Un agente tiene un nombre, el cual es un identificado único de este agente. Se muestra como un cuadrado con su nombre dentro, y se coloca junto al nodo asociado. [29]

2.4.4. Xgraph

Xgraph es una herramienta generadora de gráficos proporcionada por el ns-2. Además permite crear archivos PostScript, Tgif, y otros. Puede ser invocado dentro de los comandos Tcl para desplegar inmediatamente después de que la simulación haya concluido.

El comando Xgraph espera como entradas uno o más archivos ASCII que contengan datos en forma de pares ordenados x-y en cada línea. Por ejemplo, Xgraph f1 f2 imprimirá en la misma figura los archivos f1 y f2.

Al igual que NAM es un visualizador, pero a diferencia de NAM Xgraph muestra gráficas de monitoreo como se muestra en la figura9. Esta Herramienta muestra las graficas de tamaños de paquetes de distintos tipos como TCP, UDP, etc.

Algunas opciones que posee el Xgraph son:

Título: -t “título”

Tamaño: -geometry xsize x ysize.

Títulos de ejes: -x “xtitle” (para el título del eje x) y -y “ytitle” (para el título del eje y).

Color de texto y grilla: con la bandera -v. [31][32]

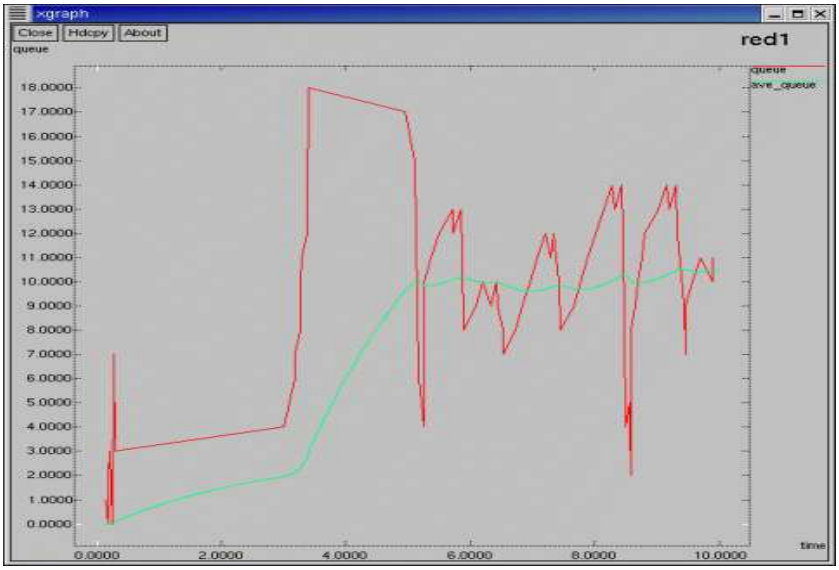


Figura II.8 Ejemplo de uso de Xgraph.

2.4.5. AWK

AWK es un lenguaje de programación diseñado para procesar datos basados en texto, ya sean ficheros o flujos de datos. El nombre AWK deriva de los apellidos de los autores: Alfred Aho, Peter Weinberger, y Brian Kernighan. awk, cuando está escrito todo en minúsculas, hace referencia al programa de Unix o Plan 9 que interpreta programas escritos en el lenguaje de programación AWK.

AWK es ejemplo de un lenguaje de programación que usa ampliamente el tipo de datos de listas asociativas (es decir, listas indexadas por cadenas clave), y expresiones regulares. El poder, brevedad y limitaciones de los programas de AWK y los guiones de sed inspiraron a Larry Wall a

escribir Perl. Debido a su densa notación, todos estos lenguajes son frecuentemente usados para escribir programas de una línea.

AWK fue una de las primeras herramientas en aparecer en Unix (en la versión 3) y ganó popularidad como una manera de añadir funcionalidad a las tuberías de Unix. La implementación de alguna versión del lenguaje AWK es estándar en casi todo sistema operativo tipo unix moderno. AWK es mencionado en las Single UNIX Specification (especificaciones básicas de unix) como una de las utilidades necesarias de todo sistema operativo Unix. Se pueden instalar implementaciones de AWK en casi todos los demás sistemas operativos.

CAPITULO III

IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS

3.1. Estructura de la simulación

La simulación de la red ad hoc móvil inalámbrica se estructuró de la siguiente manera como se observa en la figura III.1:

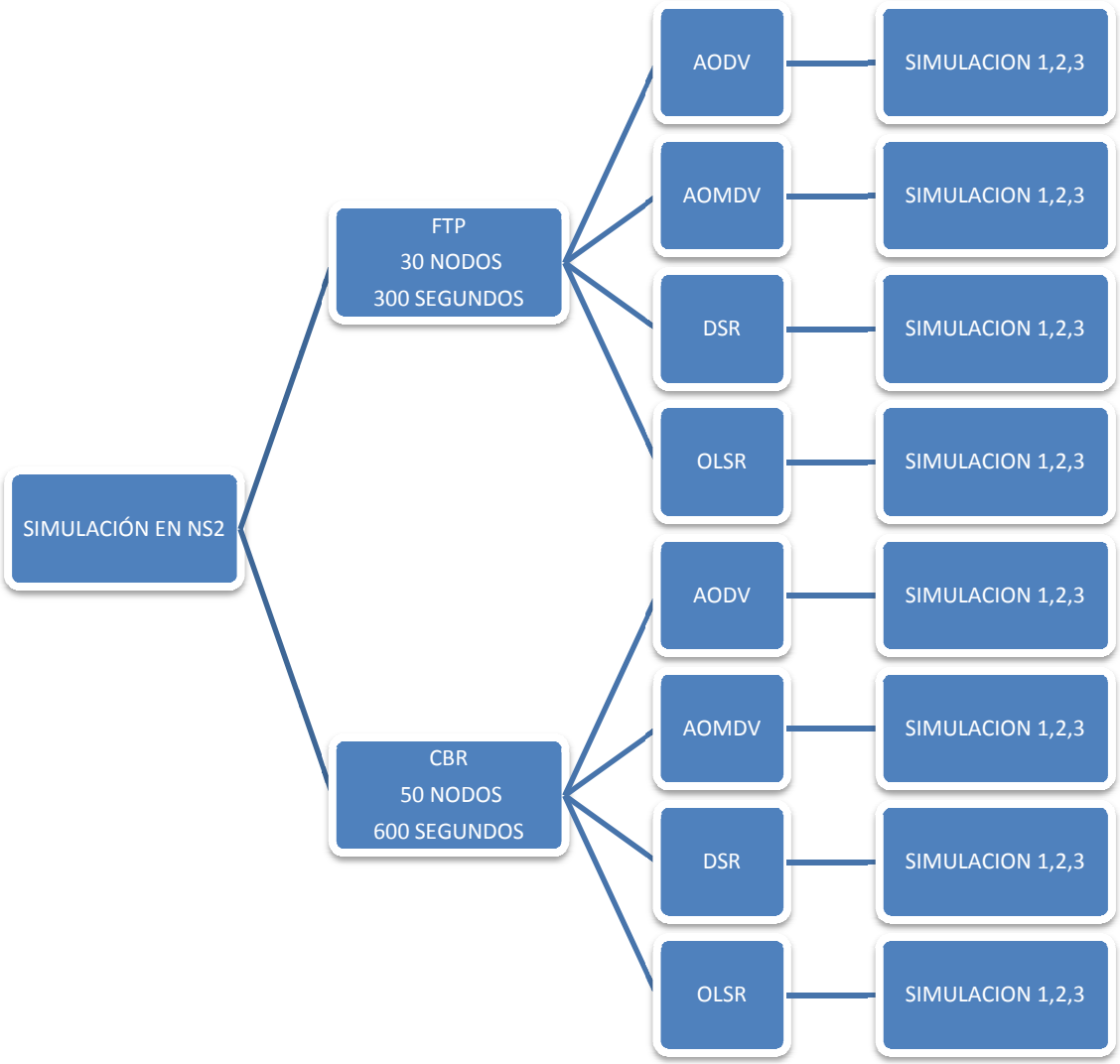


Figura III.1 Estructura de simulación

Al crear las combinaciones entre los protocolos de ruteo con los protocolos de aplicación CBR y FTP obtenemos un total de ocho simulaciones, y en cada una de estas se realizan tres simulaciones para más tarde como se observa en la figura III.2 computar los datos de las mismas y obtener finalmente 8 conjuntos de datos que serán analizados.

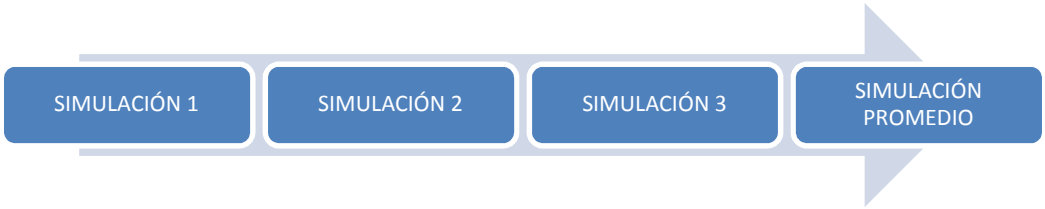


Figura III.2 Promedio de simulación

3.1.1. Parámetros de simulación

Cada simulación cuenta básicamente con los parámetros que se observan en la figura III.3, de esta manera tenemos los cuatro protocolos de ruteo a ser analizados, los protocolos de aplicación que utilizamos para manejar el trafico de la red, el numero de nodos que buscan explotar las capacidades de los protocolos de ruteo y el tiempo de simulación que permitirá que se generen eventos como cambios de la topología de la red gracias a los movimientos y tráfico aleatorio

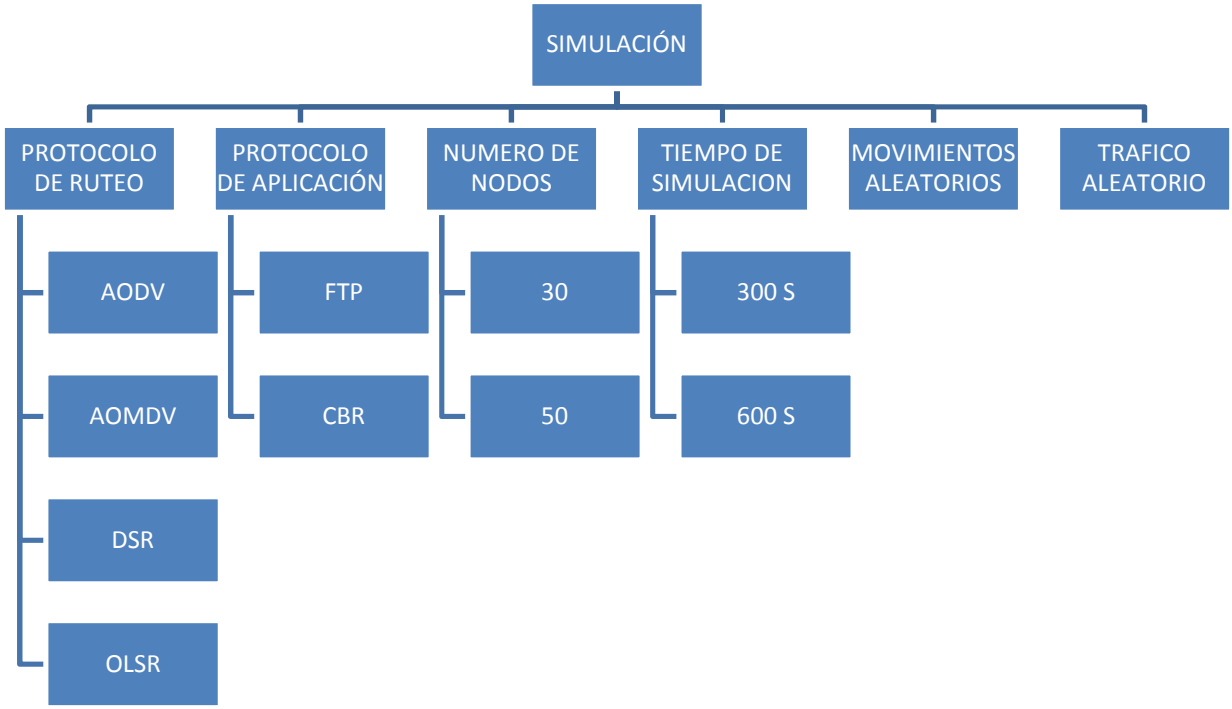


Figura III.3 Parámetros de simulación

En la figura III.4 tenemos nuestros parámetros constantes y variables en las tres simulaciones realizadas para cada protocolo de ruteo, así podemos observar que las únicas variables son las relativas al momento de la simulación como son los movimientos aleatorios y el tráfico generado aleatoriamente, mientras que parámetros como el tiempo de simulación, número de nodos y protocolos permanecen constantes. De esta manera podemos obtener resultados más confiables evitando caer en resultados que podrían darse por casualidades y excepciones esporádicas.

Figura III.4 Parámetros de simulación 2

Entre los distintos parámetros de simulación como se puede observar en la figura III.4 se debe tomar en cuenta los más representativos de la red como:

- Numero de nodos

  La simulación se realizo en primera instancia con una cantidad de treinta nodos para trafico FTP y con una cantidad de cincuenta nodos para trafico CBR, esto dado que el trafico CBR al ser más simple nos permite crear simular escenarios a mayor escala ya

que al funcionar dentro de UDP la conexión es más rápida y simple al comparar con FTP que trabaja sobre TCP y el hecho de ser un protocolo orientado a la conexión provoca que se envié grandes cantidades de trafico de control, esto provoca que al simular se genere grandes cantidades de información a ser analizada llegando a alcanzar varias decenas de gigabytes, lo que conlleva a una extracción de información que puede llevar días de procesamiento.

- Velocidad de movimiento

  La velocidad de movimiento es fundamental dentro del manejo de las rutas por partes de los protocolos, la velocidad promedio de la red determina la movilidad de la red y provoca cambios continuos en la elección de rutas, para la simulación se determino una máxima velocidad de los nodos de diez metros por segundo, que son treinta y seis kilómetros por hora y una velocidad mínima de cero metros por segundo que cambian aleatoriamente durante el tiempo de simulación.

- Tiempo de simulación

  El tiempo de simulación nos permite ver la evolución de la red desde su inicio en el que los nodos buscan llegar a establecer todas las rutas posibles que son relativas al tiempo ya que es una red móvil, luego en el tiempo de simulación se observa el comportamiento de la red respecto al cambio de rutas y conexiones aleatorias que se producen durante la simulación. Para este efecto se simulo en el caso de trafico TCP durante cinco minutos y para el trafico UDP diez minutos por las razones explicadas en el numero de nodos.

- Tipo de trafico

  El tipo de trafico es fundamental en una simulación, se puede utilizar un tráfico lo más cercano posible a la realidad en la que se transmite de manera confiable utilizando un

protocolo orientado a la conexión, y para tener otra perspectiva del funcionamiento de la red se utiliza trafico simple para estudiar el desempeño de los protocolos en la red como es el trafico CBR que funciona sobre UDP, CBR envía bits a una rata constante sin variaciones de ningún tipo, llegando a explotar la conexión y sin confirmaciones de entrega.,

Por ello para la simulación se trabajó con los dos tipos de enlaces para poder apreciar de mejor manera un tráfico real y un tráfico que busque simplemente explotar la red.

- Cobertura de los nodos

  Al tener una red dispersa y móvil, para alcanzar nodos extremos y mantener la conectividad, la cobertura es indispensable, por ello se asigno una cobertura deseada de doscientos metros para cada nodo, de esta manera aseguramos que la mayor parte del tiempo exista un enlace disponible entre los nodos, para obtener la mayor cantidad de rutas. La cobertura depende de la potencia asignada a la transmisión de cada nodo que cuenta con una antena omnidireccional.

- Tamaño del escenario

  El tamaño del escenario se relaciona directamente a la cantidad de nodos que poseamos en la red como se puede ver en la figura III.5, por ello para la simulación con treinta nodos se selecciono un escenario de un kilometro por un kilometro y para el de cincuenta nodos de un kilometro y medio por un kilometro y medio.

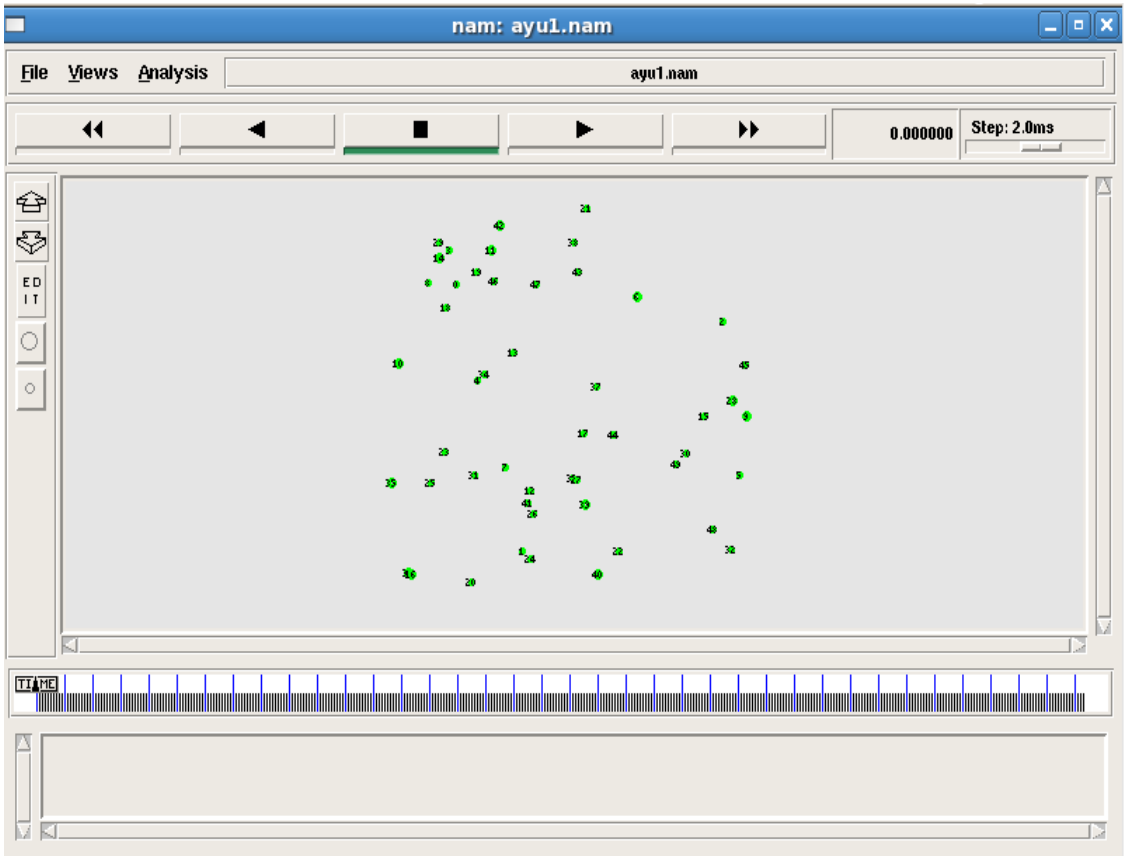


Figura III.5 Red de 50 nodos

3.1.2. Variables de análisis

Para el estudio se toma en cuenta las variables que se consideren más relevantes en análisis de eficiencia y consumo de energía de los protocolos en la red. Para dicho efecto se analizan los siguientes puntos:

- Cantidad de paquetes enviados, recibidos, desechados y reenviados a nivel de aplicación por nodo y total

- Cantidad de paquetes enviados, recibidos, desechados y reenviados a nivel de protocolo de ruteo por nodo y total
- Energía consumida en envió, recepción, espera por cada nodo y total
- Numero de nodos, velocidad de los nodos, tipo de tráfico, cobertura, número de conexiones

En la figura III.6 podemos observar la información que será analizada en los resultados obtenidos de la simulación

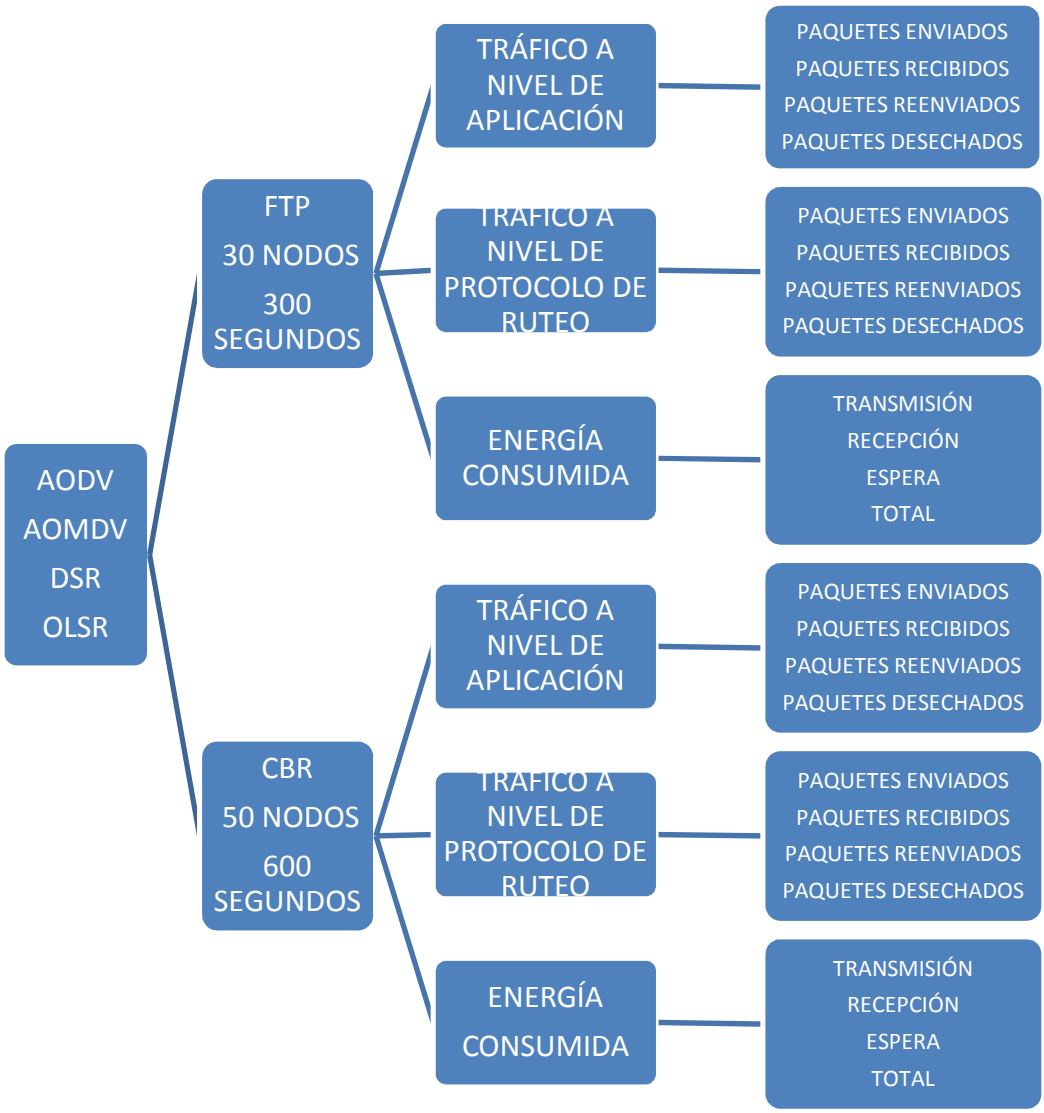


Figura III.6 Datos de análisis

Como se observa en la figura III.6 los resultados se clasifican en tres grupos principales que son:

- Paquetes a nivel de aplicación
- Paquetes a nivel de protocolo
- Energía consumida

3.1.2.1 Paquetes a nivel de aplicación

Dentro de la simulación se obtuvieron datos generando trafico CBR mediante una conexión UDP y trafico FTP mediante una conexión TCP

La generación de trafico FTP permite analizar la red mediante un protocolo orientado a la conexión, de esta manera se tiene un transporte más confiable de la información pero que genera mayor carga para la red, de esta manera antes de transportar la información se procede a crear una conexión entre los nodos lo que permite que una mayor parte de paquetes enviados lleguen a su destino.

Este tráfico es el que en una red es generado por los usuarios para intercambiar información y dentro de este tenemos:

a) Paquetes enviados

   Son los paquetes generados por el nodo emisor que es seleccionado aleatoriamente para crear un enlace de comunicación. Estos paquetes son de tipo FTP sobre una conexión TCP o de tipo CBR sobre una conexión UDP

   En el caso de CBR se utilizó los siguientes parámetros:

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set sink0 [new Agent/LossMonitor]
$ns_ attach-agent $node_(2) $sink0
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.5
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $sink0
```

En este caso tenemos por ejemplo un enlace entre el nodo 1 y 2 con paquetes de 512 bytes en un intervalo de 0,5 segundos y con un máximo de 10000 paquetes.

El número de paquetes enviados puede variar entre las distintas simulaciones y más aun en conexiones TCP dado que en este caso primero se establece el enlace para enviar los paquetes.

En el caso de FTP tenemos los siguientes parámetros:

```
set tcp_(0) [$ns_ create-connection  TCP $node_(1) TCPSink $node_(2) 0]
$tcp_(0) set window_ 32
$tcp_(0) set packetSize_ 512
set ftp_(0) [$tcp_(0) attach-source FTP]
```

Aquí se puede observar un enlace sobre TCP con una ventana de 32 bits, y paquetes de 512 bytes.

b) Paquetes recibidos

Respecto a los paquetes recibidos estos son aquellos que lograron transmitirse exitosamente entre los nodos saltando en ciertos casos varios puntos para alcanzar su destino, de esta manera en el destino tendremos el paquete enviado por el emisor.

Cabe especificar que el número de paquetes recibidos es proporcional al número de paquetes enviados tomando en cuenta la diferencia con los paquetes que se desechan en la red.

c) Paquetes reenviados

Los paquetes reenviados son aquellos que para alcanzar su destino realizaron saltos a través de la red, de esta manera los nodos intermedios reenvían paquetes que en este caso se contabilizan como reenviados teniendo la misma estructura que un paquete emitido por el emisor de tipo FTP o UDP. De este número de paquetes podemos concluir el número de saltos que utiliza cada conexión para enviar la información ya que mientras más paquetes se reenvíen significa que el número de saltos es mayor.

d) Paquetes desechados

Los paquetes desechados o caídos en la red son aquellos que por distintas razones como cambios en la topología de la red, destinos inalcanzables u otros no lograron llegar a su destino y en algún punto de la red fueron desechados.

3.1.2.2. Paquetes a nivel de protocolo de ruteo

Cada protocolo posee su técnica para encontrar las diferentes rutas de la red, por ello el trafico generado por el mismo es considerable en el desempeño de la red, por ello se analiza tanto los paquetes enviados y recibidos así como los reenviados y desechados, ligados directamente a la creación y mantenimiento de las tablas de ruteo

Este tráfico es generado por el protocolo de ruteo empleado para establecer las tablas de ruteo, de esta manera el numero de paquetes dependiendo de la topología de la red puede volverse una verdadera carga para el funcionamiento adecuado de los enlaces. Entre estos paquetes tenemos los siguientes:

a) Paquetes enviados

El número de paquetes enviados viene dado por las solicitudes de conocimiento de la red de los distintos nodos manejado de manera diferente por cada protocolo, estos paquetes en la mayoría de casos son de broadcast o a los nodos adyacentes para el intercambio de información

b) Paquetes recibidos

El número de paquetes recibidos es elevado en la mayoría de casos ya que al ser enviados mediante broadcast la cantidad de paquetes recibidos por los nodos se eleva consideradamente esto provoca por lo general tormentas de broadcast al inicializarse la red.

c) Paquetes reenviados

De igual manera el número de paquetes reenviados se debe a aquellos que atravesaron por distintos nodos para alcanzar su destino.

d) Paquetes desechados

Dado que el envío de paquetes por parte de los nodos que buscan establecer sus rutas tenemos varios paquetes de ruteo que no alcanzaron su destino sea por la topología o por problemas en el medio.

3.1.2.3. Energía consumida

Se extrae la información del consumo de energía que se encuentra directamente relacionado con la cobertura de la red por parte de cada nodo para la transmisión de paquetes y un valor nominal para la recepción y modo espera de los nodos, variando el consumo de energía de la red en función de la cantidad de trafico generada e influenciada directamente por el protocolo de ruteo.

El resultado de la energía consumida en la red viene dado por dos parámetros fundamentales que son la cobertura de cada nodo y el número de paquetes transmitidos en la red.

a) Energía consumida en transmisión

La energía que consume la red para transmitir la información es la más elevada dado que la cobertura depende de una antena con una potencia asignada de transmisión, y esto se multiplica por la cantidad de paquetes enviados en la red, así como cada paquete que se retransmite por los nodos, de esta manera la energía consumida en transmisión de paquetes es la más representativa y determinante en el análisis. Esta potencia se encuentra expresada en Watts, y la energía que se consume se expresa en Joules.

b) Energía consumida en recepción

La energía que se consume en recepción se obtiene de acuerdo a la capacidad del receptor y la energía que consume debido a su construcción variando en las distintas marcas y capacidad para amplificar la señal recibida de esta manera se ha establecido un valor general de potencia para la recepción de 0,053 W, y de igual manera la energía consumida se expresa en Joules.

c) Energía consumida en Espera

Cuando un nodo no se encuentra transmitiendo o recibiendo paquetes, decimos que se encuentra en un estado de espera, de esta manera existe una potencia asignada de 0.000014W con un consumo de energía expresado en Joules

d) Energía consumida total

Al hablar de energía se precisa que cada punto móvil en su inicio cuenta con una capacidad inicial, para la simulación si asignó un valor inicial de energía expresado en Joules de 400J, de este valor se procede a restar la energía que se va consumiendo en el transcurso de la simulación, de esta manera se puede obtener un valor de la energía consumida expresado en Joules.

3.2. Instalación del simulador

3.2.1. Sistema operativo

La simulación de la red ad hoc móvil para el estudio comparativo de los protocolos de enrutamiento será realizada sobre el simulador de redes NS2 diseñado para trabajar en sistemas UNIX, LINUX o Windows bajo Cygwin, pero para un mejor funcionamiento de los módulos del simulador y un mejor uso de memoria se recomienda instalar el simulador en un sistema Linux, ya que su diseño basado en c++ está pensado para ser ejecutado en esta plataforma.

El sistema operativo en el que se desarrolla esta tesis es Linux CentOs 5.4, la última versión de esta distribución basada en rpm's que tiene una total compatibilidad para el correcto funcionamiento de los distintos módulos de NS2 así como es compatible el animador de red NAM y el generador de gráficos estadísticos Xgraph.

3.2.2. Requerimientos de hardware

El simulador NS2 desarrollado para ejecutarse principalmente en sistemas UNIX no demanda funciona de acuerdo a la topología de red que se desee simular, por ejemplo para un simple

sistema de comunicación entre nodos con una topología simple de red cableada, no requerirá de gran capacidad de procesamiento ni de almacenamiento en memoria, sin embargo, en una simulación más compleja como en nuestro caso en la que se tienen nodos con enlaces inalámbricos, que se basan en estándares y utilizan enlaces inalámbricos en los que se realizan cálculos de potencia y tienen movilidad, los montos de memoria aumentaran exponencialmente mientras se incremente el número de nodos que no solo generan trafico sino que también generan información de acuerdo al protocolo de ruteo que se basa en el algoritmo que debe ser calculado por el computador. Los procesos que realiza NS2 para el manejo de eventos ocupan un espacio en memoria a demás de sus necesidades de alojar instrucciones en memoria cache. De esta manera podemos ver como una simulación extensa con la topología descrita puede volverse una gran carga para el procesador y memoria en base a la cantidad de nodos que se desea simular.

De esta manera NS2 puede ser instalado prácticamente en cualquier computador recomendando un mínimo de:

- 4GB de espacio en disco
- 128Mb de memoria RAM
- Procesador Pentium

Pero para simulaciones complejas se requerirá de una alta capacidad de procesador dependiendo del tiempo que se desee simular, ya que se debe tomar en cuenta que NS2 genera un archivo de trazas que dentro de sí crea una línea de código por cada paquete que viaja entre nodos, dado que el archivo guarda todos los parámetros del paquete, incluyendo su tamaño, momento de inicio, tiempo de vida, nodo que lo genero y nodo de llegada, se genera un gran archivo con mucha información. Por ejemplo si se simula una red móvil durante 60 segundos

entre dos nodos, el archivo de traza podría llegar a pesar unos 50 MB. Si lo ejecutamos en un computador con un procesador core2duo con una velocidad de 3Ghz podría llevar unos 10 min.

3.2.3. Instalación de NS2

Para la instalación de NS2 en un sistema Linux o Windows bajo cygwin se requiere contar con un compilador c++.

NS2 es un programa extenso que cuenta con varios módulos y aplicaciones complementarias. Puede encontrarse un paquete todo en uno con las aplicaciones típicas necesarias para el funcionamiento del simulador que requiere alrededor de unos 320MB de espacio en disco para compilarse. La construcción de NS2 instalando cada parte del simulador independientemente puede ahorrar un poco de espacio.

Para la descarga de NS2 podemos encontrar los archivos en SourceForge.

El simulador requiere básicamente dos lenguajes de programación para funcionar, preferiblemente actualizados, estos son: Tcl / Tk, y dos paquetes adicionales: tclcl y Otcl.

Para la construcción de NS2 tenemos dos opciones, hacerlo con cada paquete de manera independiente o utilizar un paquete todo en uno junto con un script que realice la instalación.

El paquete todo en uno posee un script de instalación que controla la instalación de Tcl/Tk, Otcl, tclcl, ns-2, nam-1 y otros varios paquetes.

Para la instalación en un sistema basado en rpm's como Red Hat, Fedora, CentOs, etc. Los pasos para la instalación por paquetes sería la siguiente:

3.2.3.1. Instalación de dependencias

Abrimos un Shell bash y escribimos en el prompt.

```
su
yum install gcc-c++
yum install libX11-devel
yum install xorg-x11-proto-devel
yum install libXt-devel

Yum install libXmu-devel
```

En estas instrucciones mediante el comando yum (Yellow Dog Updater Modified) actualizamos las dependencias e instalamos los compiladores y librerías necesarias de C++ y graficas necesarias para el funcionamiento de NAM.

3.2.3.2. Instalación de Tcl

```
su
cd
wget http://prdownloads.sourceforge.net/tcl/tcl8.4.18-src.tar.gz
cd /usr/share/
tar -zxvf /root/tcl8.4.18-src.tar.gz
cd /usr/share/tcl8.4.18/unix/
```

```
./configure
make
make install
rm -f /root/tcl8.4.18-src.tar.gz
```

En este paso descargamos el paquete de Tcl desde SourceForge, a continuación lo descomprimimos y lo instalamos en el sistema, de igual manera, a realizamos la instalación de Tk, Otcl y tclcl.

3.2.3.3. Instalación de Tk

```
su
cd
wget http://prdownloads.sourceforge.net/tcl/tk8.4.14-src.tar.gz
cd /usr/share/
tar -zxvf /root/tk8.4.14-src.tar.gz
cd /usr/share/tk8.4.14/unix/
./configure
make
make install
rm -f /root/tk8.4.14-src.tar.gz
```

3.2.3.4. Instalación de Otcl

su

cd

wget http://downloads.sourceforge.net/otcl-tclcl/otcl-src-1.12.tar.gz

cd /usr/share

tar -zxvf /root/otcl-src-1.12.tar.gz

cd /usr/share/otcl-1.12/

./configure --with-tcl=/usr/share/tcl8.4.14/

make

kwrite Makefile

INST_OLIBSH= NONE/lib

Lo modificamos:

INST_OLIBSH= /usr/local/lib

make install

rm -f /root/otcl-src-1.12.tar.gz

3.2.3.5. Instalación de tclcl

su

cd

wget http://downloads.sourceforge.net/otcl-tclcl/tclcl-src-1.18.tar.gz

cd /usr/share

tar -zxvf /root/tclcl-src-1.18.tar.gz

```
cd /usr/share/tclcl-1.18/
./configure --with-tcl=/usr/share/tcl8.4.14/
make
make install
rm -f /root/tclcl-src-1.18.tar.gz
```

3.2.3.6. Instalación de NS2

```
su
cd
wget http://downloads.sourceforge.net/nsnam/ns-src-2.30.tar.gz
cd /usr/share
tar -zxvf /root/ns-src-2.30.tar.gz
cd /usr/share/ns-2.30/
./configure
make
make install
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:' > /etc/profile.d/ns.sh
chmod 0733 /etc/profile.d/ns.sh
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:' > $HOME/.bashrc
rm -f /root/ns-src-2.30.tar.gz
```

En la instalación de los módulos de NS2 realizamos la instalación del paquete y configuramos las variables de entorno para permitir la ejecución de la aplicación en diferentes instancias.

Para comprobar la instalación del simulador podemos ejecutar el comando “ns” en un prompt y deberíamos poder ver el símbolo “%” que indica que nos encontramos dentro del simulador listo para ejecutar comandos Tcl

Como por ejemplo:

% set ns [new Simulator]

Que crea una nueva instancia del simulador

3.2.3.7. Instalación de NAM

```
su
cd
wget http://downloads.sourceforge.net/nsnam/nam-src-1.12.tar.gz
cd /usr/share
tar -zxvf /root/nam-src-1.12.tar.gz
cd /usr/share/nam-1.12/
./configure
make
make install
rm -f /root/nam-src-1.12.tar.gz
```

Para comprobar la instalación de NAM luego de asignar las variables de entorno podemos ejecutar el comando “nam” en un prompt y se debería abrir la ventana del animador de simulaciones.

3.2.3.8. Instalación de Xgraph

```
su
cd
wget http://downloads.sourceforge.net/nsnam/xgraph-12.1.tar.gz
cd /usr/share
tar -zxvf /root/xgraph-12.1.tar.gz
cd /usr/share/xgraph-12.1/
./configure
make
make install
rm -f /root/xgraph-12.1.tar.gz
```

Para comprobar la instalación de XGRAPH podemos ejecutar el comando “xgraph archivo” en donde archivo contiene coordenadas de puntos para generar gráficos de datos.

En caso de instalar el paquete ns-allinone, debemos comprobar las dependencias de los archivos, descomprimir el paquete y proceder a instalar de la siguiente manera:

```
tar –zxvf ns-allinone-2.34.tar
cd ns-allinone-2.34
./install
```

A continuación debemos configurar las variables de entorno del sistema para acceder directamente al simulador de la siguiente manera como vemos en la figura III.7

```
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin:/home/ns-allinone-2.34/bin:/home/ns-allinone-2.34/tcl8.4.18/unix:/home/ns-a
llinone-2.34/tk8.4.18/unix:/home/ns-allinone-2.34/ns-2.34/indep-utils/cmu-scen-gen/setdest/:/hom
e/ns-allinone-2.34/ns-2.34/indep-utils/cmu-scen-gen

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/ns-allinone-2.34/otcl-1.13:/home/ns-allinone-2.34/lib

TCL_LIBRARY=$TCL_LIBRARY:/home/ns-allinone-2.34/tcl8.4.18/library

unset LC_ALL

export PATH
export LD_LIBRARY_PATH
export TCL_LIBRARY
export LC_NUMERIC=C
unset USERNAME
```

Figura III.7 Variables de entorno

Una vez configurado el sistema podemos ejecutar el script de validación que viene con el paquete para ver las simulaciones que podrían funcionar con el simulador, este proceso pude llevar desde unos pocos minutos hasta una hora dependiendo del computador.

3.3. Creación del escenario

Para la simulación de la red inalámbrica Ad Hoc se crea un escenario que permita obtener los datos de una red móvil para posteriormente poder analizarlos y compararlos de acuerdo a cada protocolo, para ello se crean múltiples simulaciones con la misma configuración variando el protocolo encargado del ruteo en la red

3.3.1. Aspectos Importantes

Al momento de simular, como se muestra en la figura III.8, se toma en cuenta los elementos que conformaran nuestro sistema entre ellos tendremos la red inalámbrica, el área de cobertura, la movilidad, la generación de tráfico y la configuración de los nodos.

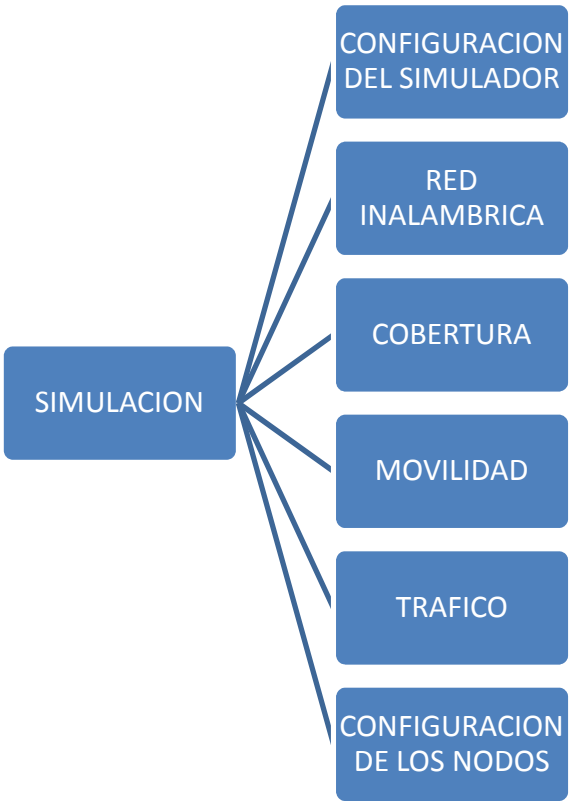


Figura III.8 Aspectos importantes en la simulación

3.3.2. Configuración del simulador

Primeramente creamos una nueva instancia en el simulador

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

Abrimos un archivo en el cual se genera la traza de los paquetes, es decir el resultado de la simulación

```
set tracefd     [open ayu1.tr w]
```

A continuación dirigimos toda la salida de las trazas al archive que abrimos para el efecto

```
$ns_ trace-all $tracefd
```

De igual manera creamos un archivo para guardar las trazas que utilizamos en la animación de la simulación

```
set namtrace [open ayu1.nam w]
```

Especificamos que la salida de datos de traza para nam se guarden en el archive abierto en un plano x y

```
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

Indicamos al simulador que use el nuevo formato de trazas

```
$ns_ use-newtrace
```

Creamos un nuevo objeto en la topología

```
set topo       [new Topography]
```

Asignamos a esta topología un plano x y en donde se ejecutaran los eventos

```
$topo load_flatgrid $val(x) $val(y)
```

Creamos el objeto god encargado de gestionar los enlaces entre nodos

```
set god_ [create-god $val(nn)]
```

En esta instrucción permitimos que se utilice la nueva traza para comunicación wireless

```
$ns_ set WirelessNewTrace_ ON
```

Les indicamos a los nodos participantes cuando termina la simulación

```
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(fin).0 "$node_($i) reset";
}
```

Detenemos la simulación llamando al proceso necesario

```
$ns_ at $val(fin).01 "stop"
$ns_ at $val(fin).02 "puts \"Finalizando...\" ; $ns_ halt"
```

En el proceso de finalización cerramos los archives y terminamos las trazas de datos

```
proc stop {} {
    global  ftemp ns_ tracefd
    $ns_ flush-trace
```

```
    $ns_ nam-end-wireless [$ns_ now]
    close $tracefd
}
```

Iniciamos de esta manera todo el script de simulación

```
puts "Iniciando simulación..."
$ns_ run
```

3.3.3. Red inalámbrica

Para la simulación de una red inalámbrica se deben establecer los siguientes parámetros básicos a utilizar mediante la creación de variables.

```
Tipo de canal
set val(chan)           Channel/WirelessChannel
Modelo de propagación
set val(prop)           Propagation/TwoRayGround
Tipo de interfaz de red
set val(netif)          Phy/WirelessPhy
Control de acceso al medio
set val(mac)            Mac/802_11
Manejo de colas en la interfaz
set val(ifq)            Queue/DropTail/PriQueue
Tipo de capa de enlace
set val(ll)             LL
```

Tipo de antena

set val(ant) Antenna/OmniAntenna

Cantidad máxima de paquetes en la interfaz

set val(ifqlen) 50

Cantidad de nodos móviles

set val(nn) 30

Protocolo de ruteo

set val(rp) AODV

Tamaño del plano en el eje x en metros

set val(x) 1000

Tamaño del plano en el eje y en metros

set val(y) 1000

Archivo con información de movilidad

set val(sc) "/root/Desktop/ejems/simu12/simuAODV/movs1.tcl"

Archivos con información de tráfico

set val(cp) "/root/Desktop/ejems/simu12/simuAODV/traf3.tcl"

Duración de la simulación

set val(fin) 300.0

Cobertura de los nodos

set val(MNcoverage) 200.0

Configuración del estándar 802.11

Mac/802_11 set dataRate_ 11Mb

Mac/802_11 set basicRate_ 1Mb

```
Mac/802_11 set PLCPDataRate_      1Mb
Mac/802_11 set RTSThreshold_       0
Mac/802_11 set PreambleLength_     72
Mac/802_11 set PLCPHeaderLength_   24
```

3.3.4. Configuración de los nodos

Para la creación de los nodos especificamos una configuración general en la que asignamos los valores de las variables a la configuración del nodo

```
$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) \
                 -topoInstance $topo \
```

Especificamos los eventos que generaran trazas, estos pueden ser del agente, de ruteo, de acceso al medio y de movimiento

```
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace ON \
```

Configuramos un modelo de energía en donde especificamos el consumo de energía en trasmisión en recepción y en espera asi como la energía inicial con la que cuentan los nodos

```
-energyModel "EnergyModel" \
-idlePower 0.000014 \
-rxPower 0.053 \
-txPower $pottrans \
-initialEnergy 400 \
```

Asignamos esta configuración a los nodos de la red

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

Especificamos el tamaño de los nodos dentro del escenario

```
for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 15
}
```

3.3.5. Cobertura

Se crea una función para el cálculo de la potencia en función 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
}
```

Se configura los parámetros básicos de las antenas

```
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
```

Se asigna la potencia calculada a la interfaz

```
set pottrans [SetPt $val(MNcoverage)]
puts "potencia de transmicion para cubrir $val(MNcoverage) metros es: $pottrans"
```

3.3.6. Movilidad

Se incluye el código de movilidad de la red dentro del script de la simulación

source $val(sc)

El contenido de este script incluye las coordenadas los nodos junto con los movimientos y la velocidad del mismo.

Para ello contamos con un programa generador de movimientos llamado setdest que forma parte de la aplicación de ns2 que establece los parámetros especificados en la tabla III.I. Al momento de utilizar el comando setdest tenemos la posibilidad de utilizar dos versiones de las aplicaciones con distintos parámetros, entre estos tenemos:

	Función	Rango	Default	Ver 1	Ver 2
-M	Velocidad máxima	≥ 0	0.0	Si	Si
-m	Velocidad mínima	[0, velocidad máxima]	0.0	No, siempre 0	Si
-n	Numero de nodos	1,2,...	0	Si	Si
-P	Tipo de pause	1 (constante), 2 (uniforme)	1	No	Si
-p	Tiempo de pausa	≥ 0	0.0	Sí, pero siempre constante	Si
-s	Tipo de velocidad	1 (uniforme), 2 (normal)	1	No	Si
-t	Tiempo máximo	≥ 0	0.0	Si	Si
-v	versión	1 (1999 CMU), 2 (2003 UM)	1	Si	Si
-x	Coordenada en x	≥ 0	0.0	Si	Si
-y	Coordenada en y	≥ 0	0.0	Si	Si

Tabla III.I Opciones setdest

El comando de ejecución es el siguiente:

```
<original 1999 CMU version (version 1)>
 setdest         -v <1> -n <nodes> -p <pause time> -M <max speed>
                 -t <simulation time> -x <max X> -y <max Y>


<modified 2003 U.Michigan version (version 2)>
 setdest        -v <2> -n <nodes> -s <speed type> -m <min speed> -M <max speed>
                -t <simulation time> -P <pause type> -p <pause time> -x <max X> -y <max Y>
                (Refer to the script files make-scen.csh and make-scen-steadystate.csh for detai
l.)
```

De esta manera el programa nos genera un script de este tipo:

Se especifica las coordenadas iniciales

$node_(0) set X_ 731.826497293057

$node_(0) set Y_ 942.399944648889

$node_(0) set Z_ 0.000000000000

Generamos movimiento especificando el momento de la acción, las coordenadas de destino y la velocidad de movimiento

$ns_ at 0.000000000000 "$node_(0) setdest 182.652531655442 607.361608494822 9.512758615579"

3.3.7. Tráfico

Se incluye el código de tráfico de la red dentro del script de la simulación

source $val(cp)

Para generar este script de tráfico contamos con otra herramienta incluida en NS2, este es un generador de tráfico de tipo CBR y FTP llamado cbrgen.tcl.

ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-rate rate]

Al ser un script Tcl lo ejecutamos con el comando “ns” de esta manera se redirige la salida hacia el archivo fuente que se utilizara en la simulación. Para ello debemos especificar en el comando el tipo de tráfico, la cantidad de nodos, un número de inicio aleatoria, la cantidad máxima de conexiones y la rata de datos.

El resultado de esta operación nos genera una secuencia de acciones como la siguiente:

```
set tcp_(0) [$ns_ create-connection  TCP $node_(1) TCPSink $node_(2) 0]
$tcp_(0) set window_ 32
$tcp_(0) set packetSize_ 512
set ftp_(0) [$tcp_(0) attach-source FTP]
$ns_ at 2.5568388786897245 "$ftp_(0) start"
```

De esta manera podemos observar cómo se crea una conexión en este caso tcp, entre los nodos 1 y 2 a los 2.55 segundos con una ventana de 32 bits y paquetes de 512 bits.

3.4. Valores por protocolo

3.4.1. Filtros AWK

Para la obtención de datos se utilizo el lenguaje AWK creando los siguientes programas para extraer la información de los archivos de simulación:

Paquetes recibidos y enviados

Paquetes caídos y reenviados

Paquetes por protocolo de ruteo

Consumo de energía

El código de estos programas puede ser encontrado en los anexos.

3.4.2. AODV

De esta manera generando tráfico FTP en una red móvil de treinta nodos con una velocidad máxima de diez metros por segundo con una simulación de cinco minutos obtuvimos la siguiente información que podemos ver en la tabla III.II

AODV FTP				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	77431	83433	67698	76187,333
trafico recibido	76562	82609	67129	75433,333
trafico reenviado	38708	48515	53015	46746,000
trafico desechado	658	599	455	570,667
protocolo enviado	13911	17148	14915	15324,667
protocolo recibido	56140	72804	52092	60345,333
protocolo reenviado	555	612	534	567,000
protocolo desechado	5010	6847	5115	5657,333
energía total consumida	956,277	1.037,9440	974,737	989,653
energía consumida en espera	0,05	0,056	0,051	0,052
energía consumida en transmisión	696,706	787,247	725,197	736,383
energía consumida en recepción	259,521	250,641	249,489	253,217

Tabla III.II Resultados AODV

En la figura III.9 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación

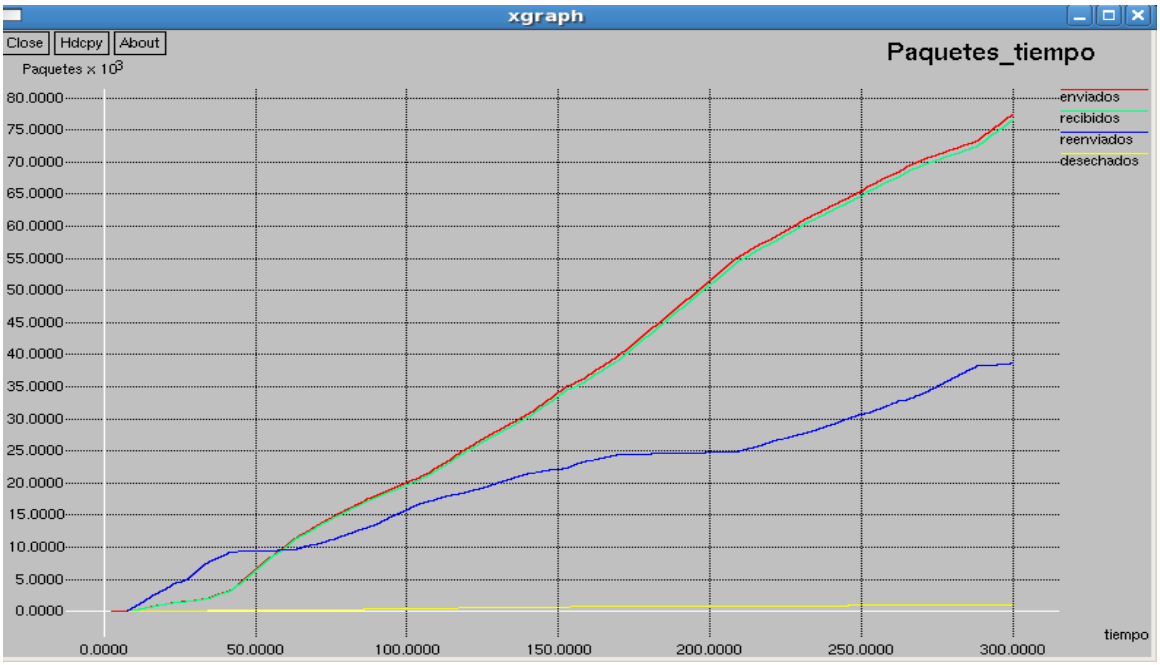


Figura III.9 Resultado AODV2

En la figura III.10 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación

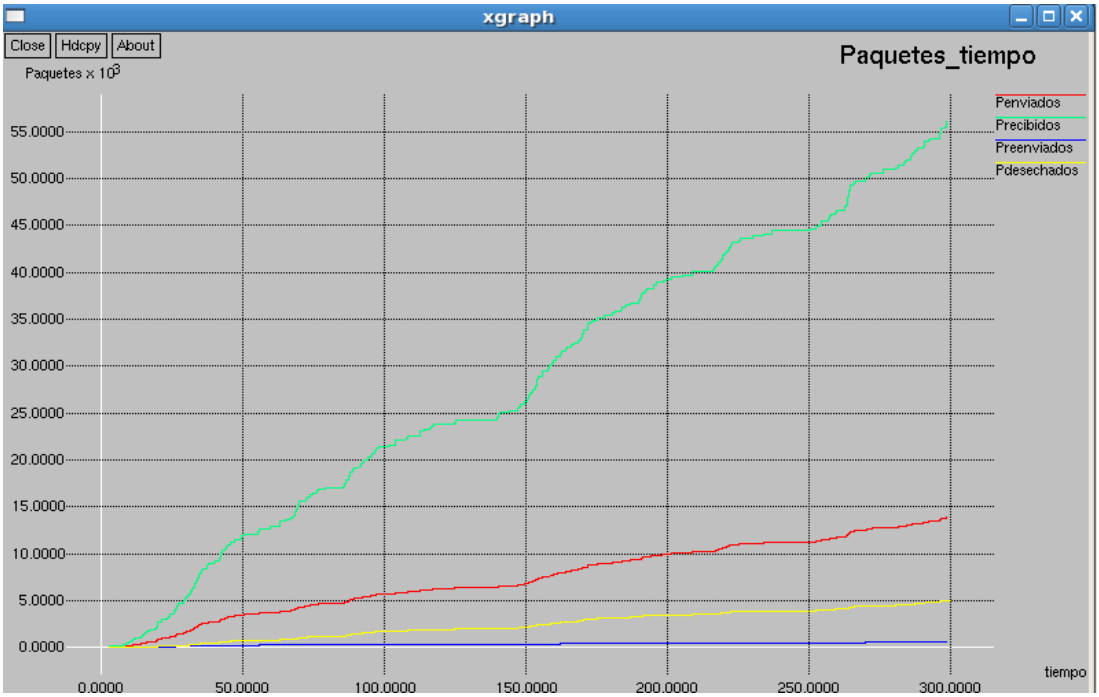


Figura III.10 Resultado AODV3

Generando tráfico CBR en una red móvil de cincuenta nodos con una velocidad máxima de diez metros por segundo con una simulación de diez minutos obtuvimos la siguiente información que podemos ver en la tabla III.III

AODV CBR				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	31511	31083	29862	30818,667
trafico recibido	16317	15248	9423	13662,667
trafico reenviado	73078	61926	33846	56283,333
trafico desechado	1645	1516	1443	1534,667
protocolo enviado	143764	129112	88727	120534,333
protocolo recibido	522338	476598	286500	428478,667
protocolo reenviado	3853	3425	8269	5182,333
protocolo desechado	22544	18585	1621	14250,000
energía total consumida	642,013	549,596	329,925	507,178
energía consumida en espera	0,372	0,385	0,400	0,386
energía consumida en transmisión	482,356	420,291	255,645	386,097
energía consumida en recepción	159,285	128,92	73,880	120,695

Tabla III.III Resultados AODV4

En la figura III.11 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación

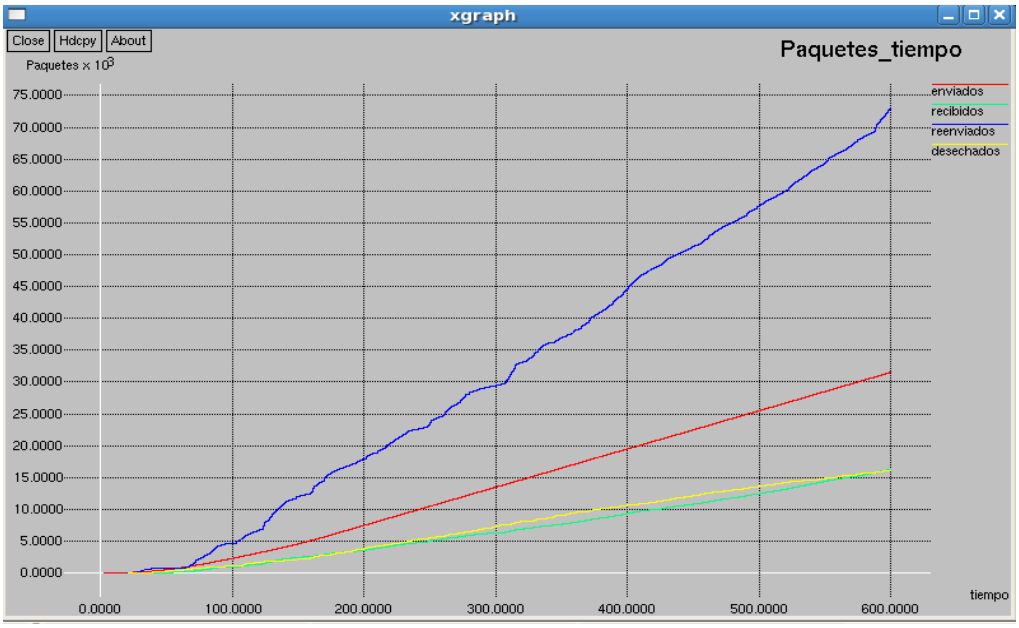


Figura III.11 Resultados AODV5

En la figura III.12 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación

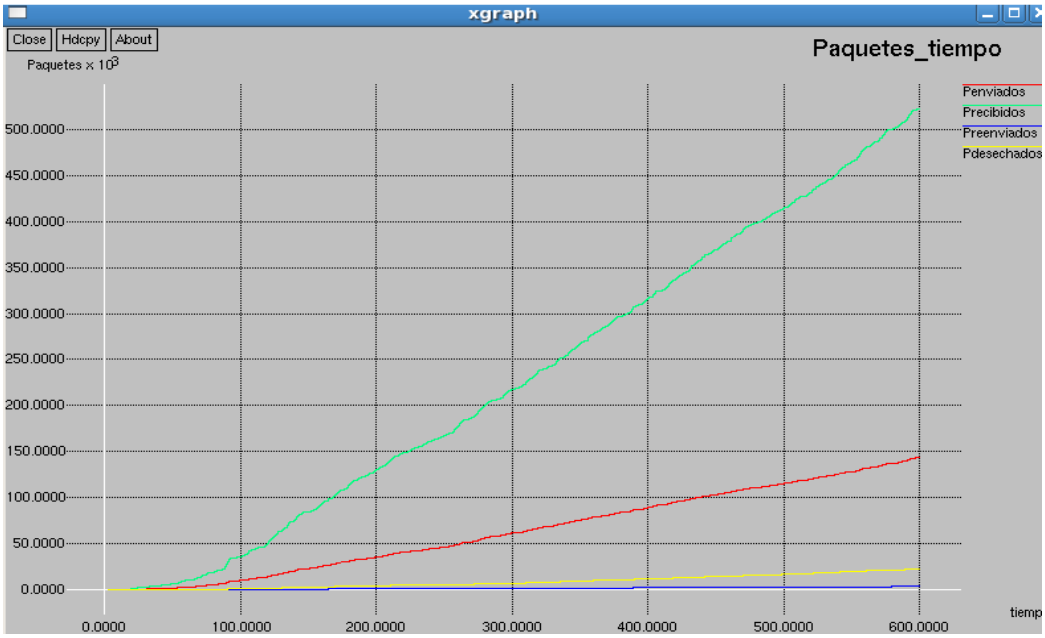


Figura III.12 Resultados AODV6

3.4.3. Protocolo AOMDV.

En la tabla III.IV podemos encontrar los resultados de la simulación para AOMDV con FTP

AOMDV FTP				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	76113	86240	59876	74076,333
trafico recibido	75446	85435	59141	73340,667
trafico reenviado	33797	29889	44582	36089,333
trafico desechado	539	657	464	553,333
protocolo enviado	24458	22493	22985	23312,000
protocolo recibido	103084	101652	85308	96681,333
protocolo reenviado	203	225	329	252,333
protocolo desechado	7275	6822	5842	6646,333
energía total consumida	939,994	964,5620	891,033	931,863
energía consumida en espera	0,052	0,064	0,057	0,058
energía consumida en transmisión	658,873	732,253	660,365	683,830
energía consumida en recepción	254,069	232,245	230,611	238,975

Tabla III.IV Resultados AOMDV

En la figura III.13 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación

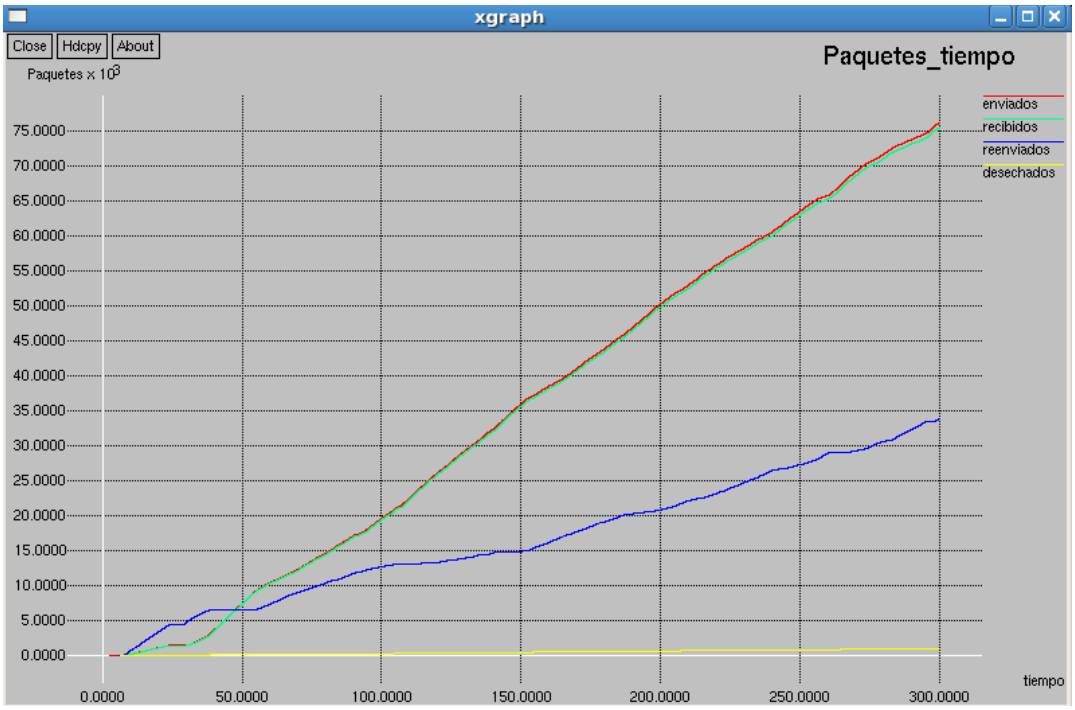


Figura III.13 Resultados AOMDV2

En la figura III.14 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

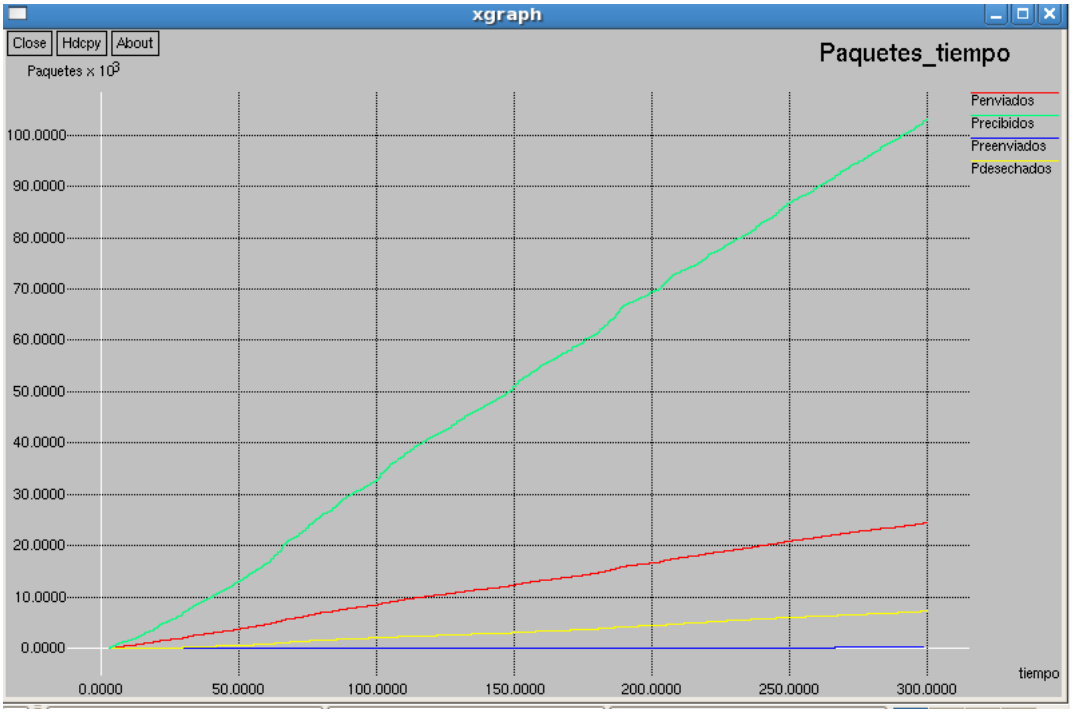


Figura III.14 Resultados AOMDV3

En la tabla III.V podemos encontrar los resultados de la simulación para AOMDV con CBR

AOMDV CBR				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	31578	31138	29896	30870,667
trafico recibido	11908	11240	7263	10137,000
trafico reenviado	58338	49375	25013	44242,000
trafico desechado	5696	5140	3577	4804,333
protocolo enviado	259575	264727	226515	250272,333
protocolo recibido	919084	948458	699512	855684,667
protocolo reenviado	2562	2178	1014	1918,000
protocolo desechado	47856	45844	29345	41015,000
energía total consumida	746,427	690,547	481,801	639,592
energía consumida en espera	0,365	0,376	0,399	0,380
energía consumida en transmisión	569,463	536,941	380,668	495,691
energía consumida en recepción	176,599	153,23	100,734	143,521

Tabla III.V Resultados AOMDV4

En la figura III.15 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

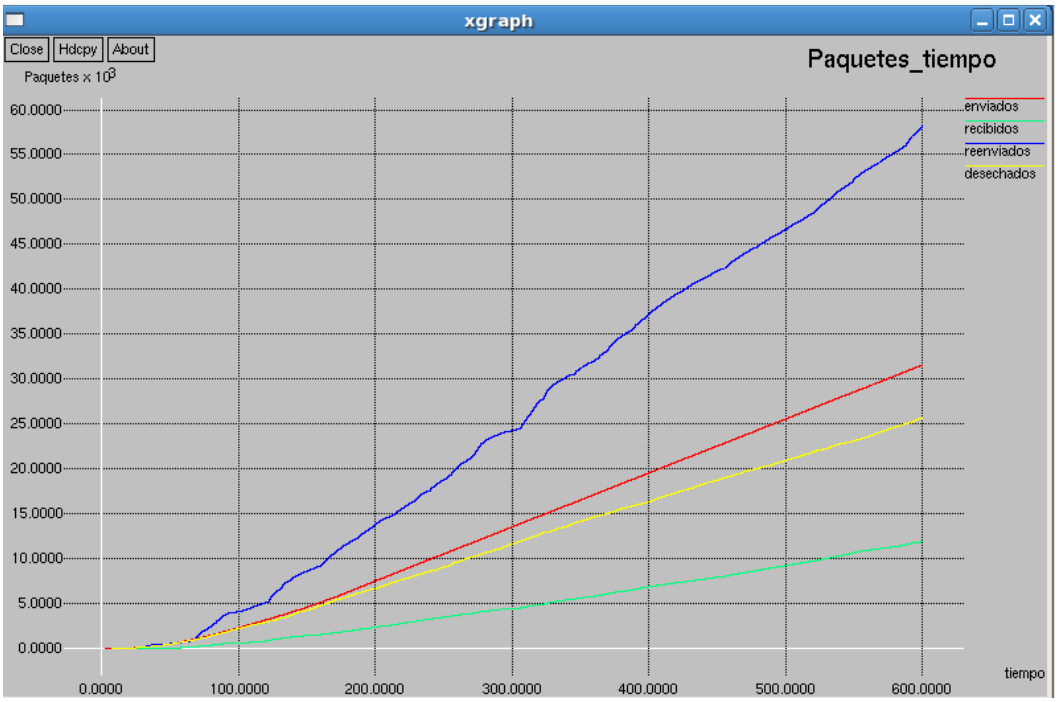


Figura III.15 Resultados AOMDV5

En la figura III.16 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

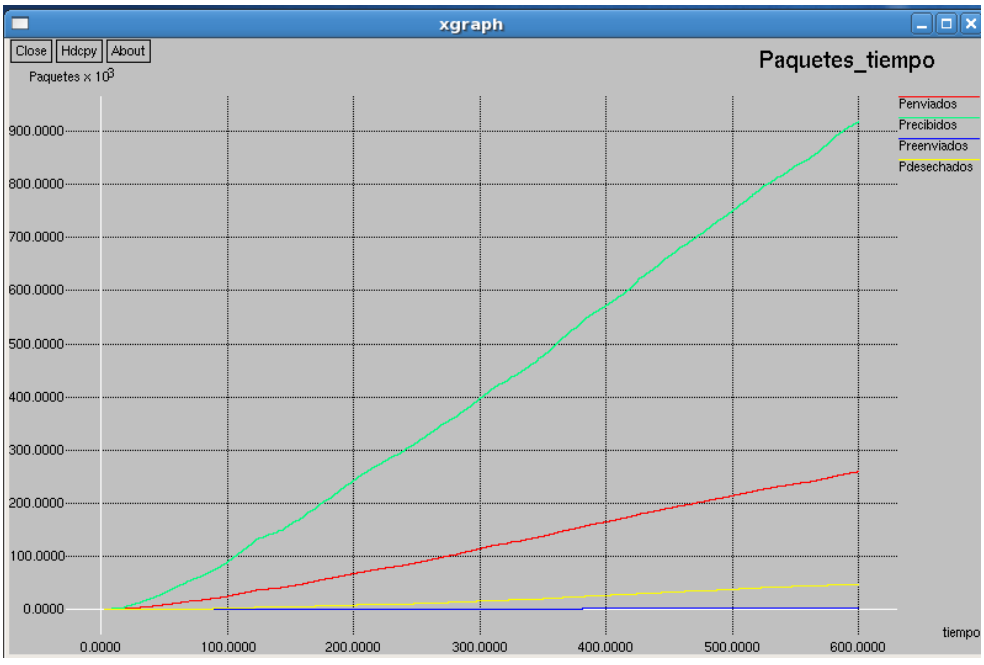


Figura III.16 Resultados AOMDV6

3.4.4. Protocolo DSR.

En la tabla III.VI podemos encontrar los resultados de la simulación para DSR

DSR FTP				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	69296	88649	64559	74168,000
trafico recibido	68900	88297	64012	73736,333
trafico reenviado	45548	41707	51476	46243,667
trafico desechado	314	373	386	357,667
protocolo enviado	9265	9060	9812	9379,000
protocolo recibido	22280	19076	22186	21180,667
protocolo reenviado	2952	3069	2994	3005,000
protocolo desechado	1692	1120	1956	1589,333
energía total consumida	969,343	1.053,2200	965,314	995,959
energía consumida en espera	0,05	0,053	0,052	0,052
energía consumida en transmisión	707,46	799,395	716,038	740,964
energía consumida en recepción	261,833	253,772	249,224	254,943

Tabla III.VI Resultados DSR

En la figura III.17 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

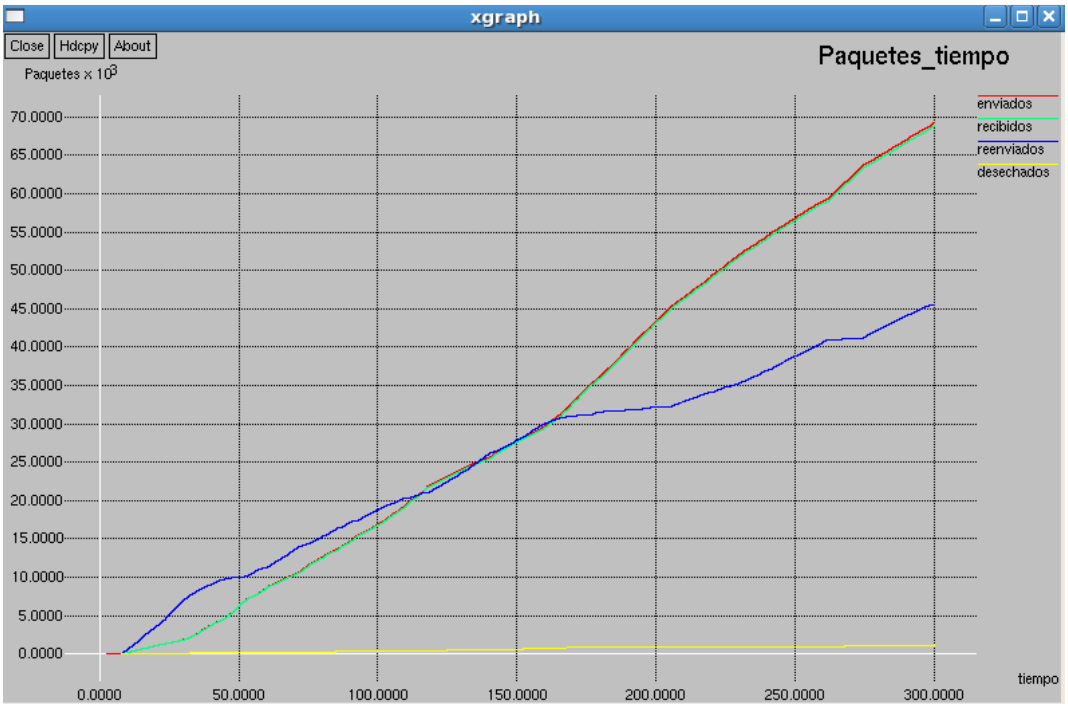


Figura III.17 Resultados DSR2

En la figura III.18 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

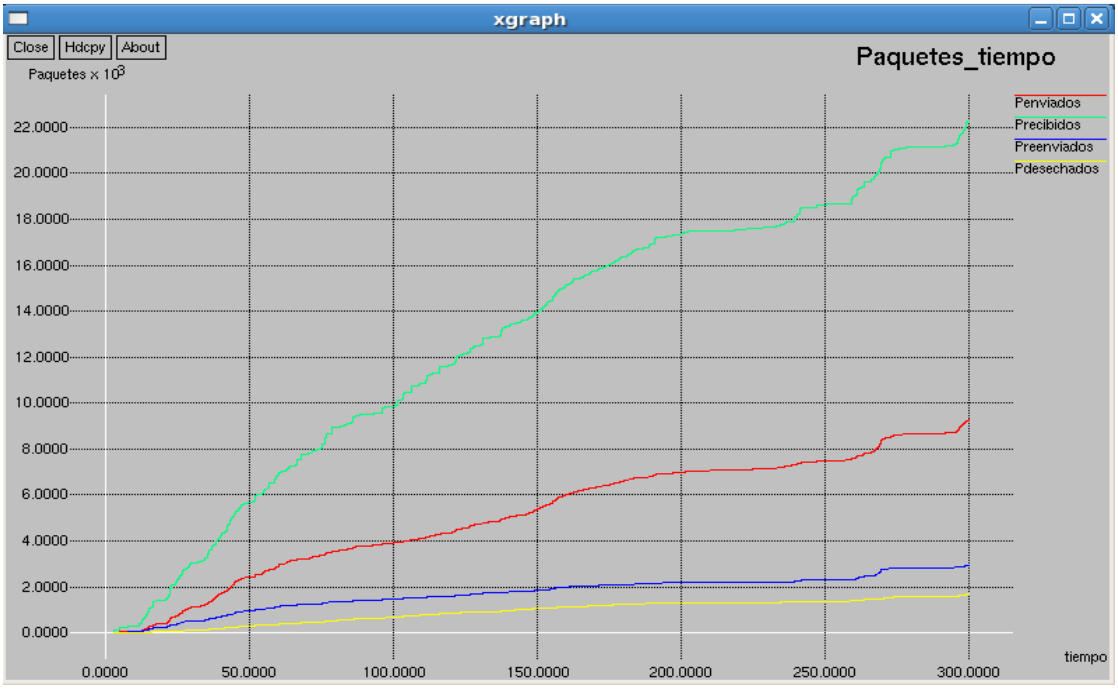


Figura III.18 Resultados DSR3

En la tabla III.VII podemos encontrar los resultados de la simulación para DSR con CBR

DSR CBR				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	31508	31205	29961	30891,333
trafico recibido	19703	17441	11725	16289,667
trafico reenviado	115158	86773	77226	93052,333
trafico desechado	955	798	478	743,667
protocolo enviado	71902	58785	55666	62117,667
protocolo recibido	304112	257070	208396	256526,000
protocolo reenviado	46675	37749	35634	40019,333
protocolo desechado	12544	9566	6300	9470,000
energía total consumida	901,627	695,905	570,959	722,830
energía consumida en espera	0,354	0,375	0,386	0,372
energía consumida en transmisión	683,237	534,436	442,736	553,470
energía consumida en recepción	218,036	161,094	127,837	168,989

Tabla III.VII Resultados DSR4

En la figura III.19 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

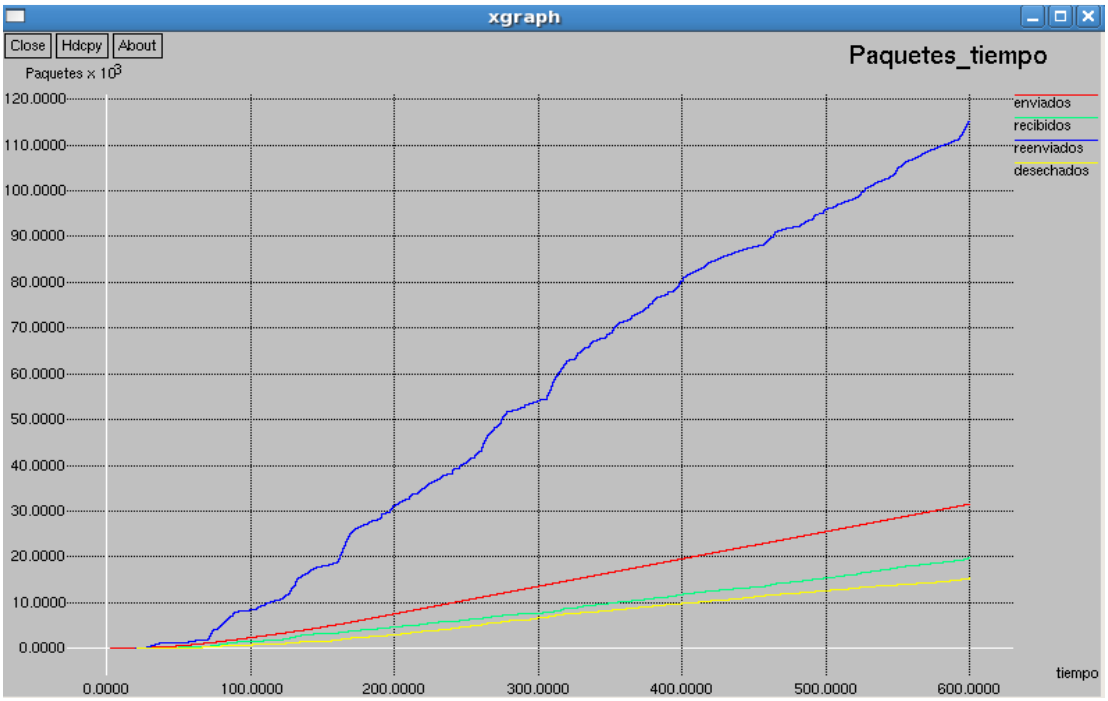


Figura III.19 Resultados DSR5

En la figura III.20 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

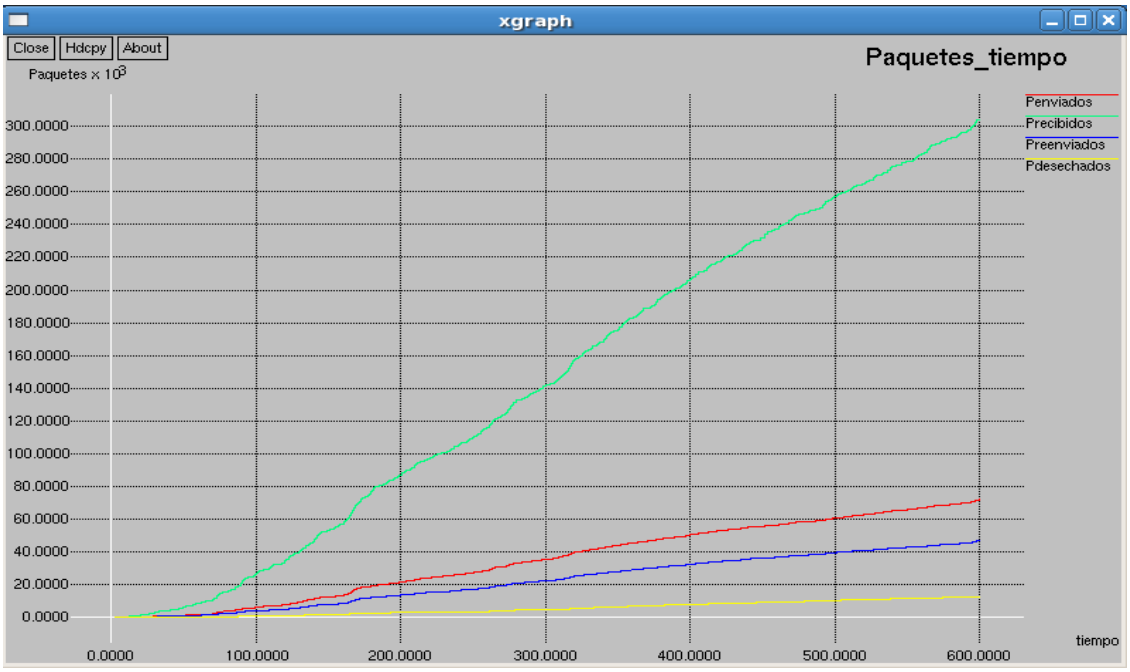


Figura III.20 Resultados DSR6

3.4.5. Protocolo OLSR.

En la tabla III.VIII podemos encontrar los resultados de la simulación para OLSR con FTP

OLSR FTP				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	83984	96849	69710	83514,333
trafico recibido	83309	96456	69182	82982,333
trafico reenviado	19802	23393	27789	23661,333
trafico desechado	503	248	273	341,333
protocolo enviado	21474	22066	20771	21437,000
protocolo recibido	100292	111504	86670	99488,667
protocolo reenviado	0	0	0	0,000
protocolo desechado	5449	5126	4337	4970,667
energía total consumida	899,864	993,87	833,230	908,988
energía consumida en espera	0,055	0,064	0,063	0,061
energía consumida en transmisión	647,56	758,076	616,520	674,052
energía consumida en recepción	252,249	235,73	216,647	234,875

Tabla III.VIII Resultados OLSR

En la figura III.21 podemos observar el número de paquetes a nivel aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

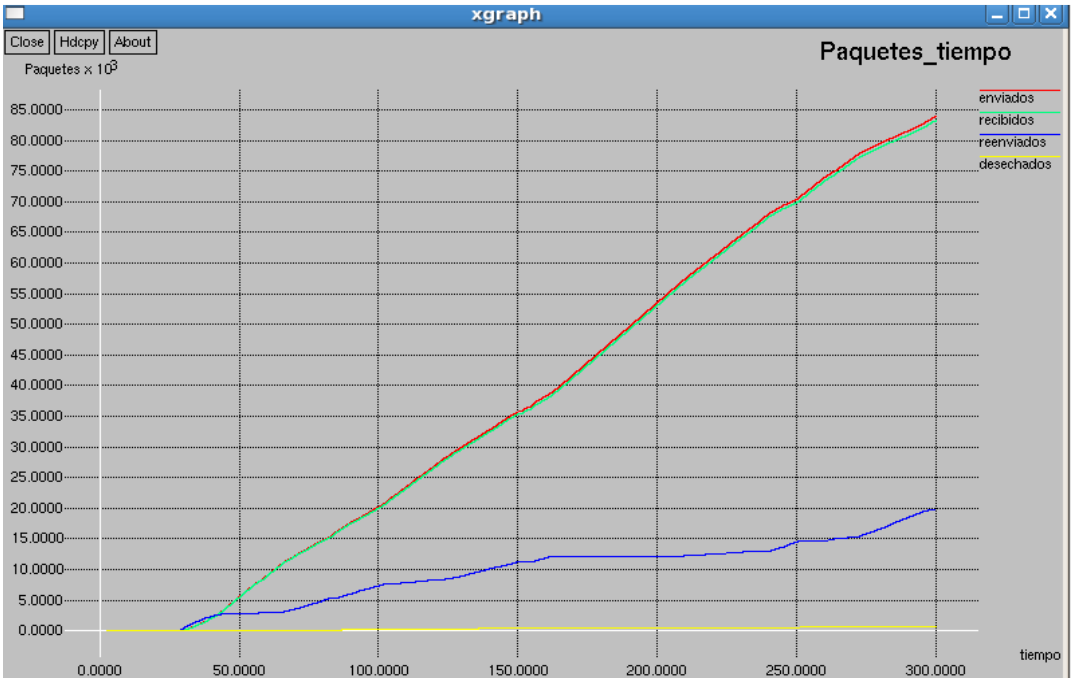


Figura III.21 Resultados OSLR2

En la figura III.22 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

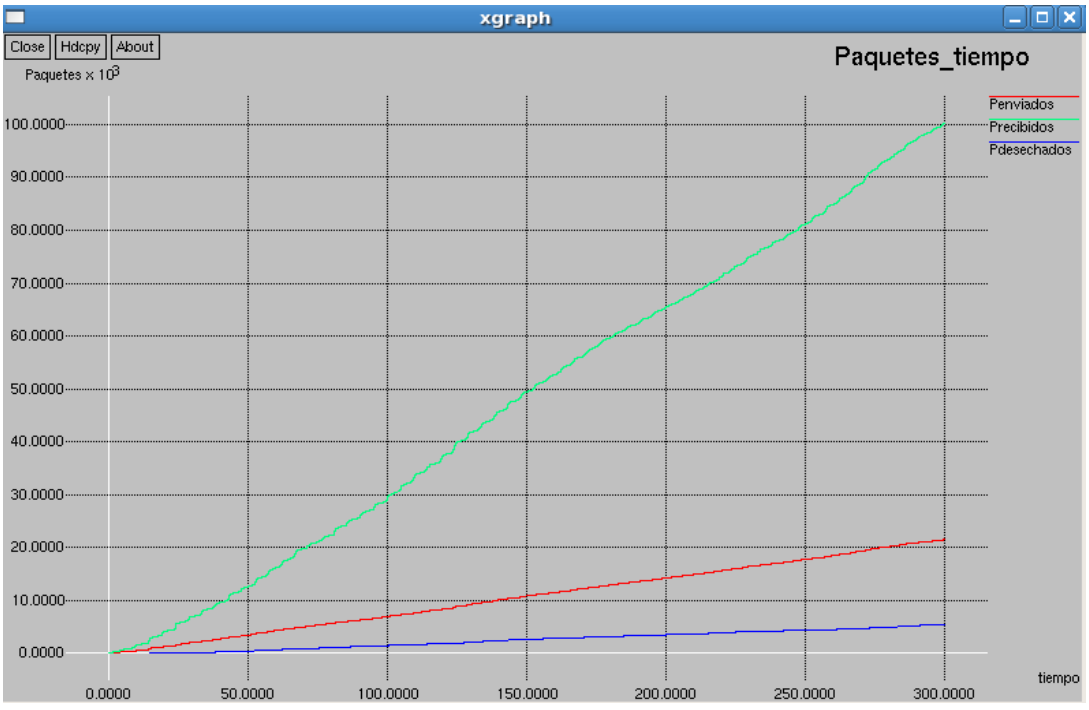


Figura III.22 Resultados OLSR3

En la tabla III.IX podemos encontrar los resultados de la simulación para OLSR con CBR

OLSR CBR				
	simulacion1	simulacion2	simulacion3	promedio
trafico enviado	31426	31198	29980	30868,000
trafico recibido	7344	7862	4681	6629,000
trafico reenviado	34528	30250	12901	25893,000
trafico desechado	19589	19380	23102	20690,333
protocolo enviado	84712	77938	64484	75711,333
protocolo recibido	348956	316162	219316	294811,333
protocolo reenviado	0	0	0	0,000
protocolo desechado	2832	2188	891	1970,333
energía total consumida	412,461	366,3500	218,480	332,430
energía consumida en espera	0,398	0,4	0,400	0,399
energía consumida en transmisión	307,11	277,94	169,221	251,424
energía consumida en recepción	104,953	88,01	48,859	80,607

Tabla III.IX Resultados OLSR4

En la figura III.23 podemos observar el número de paquetes a nivel de aplicación en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

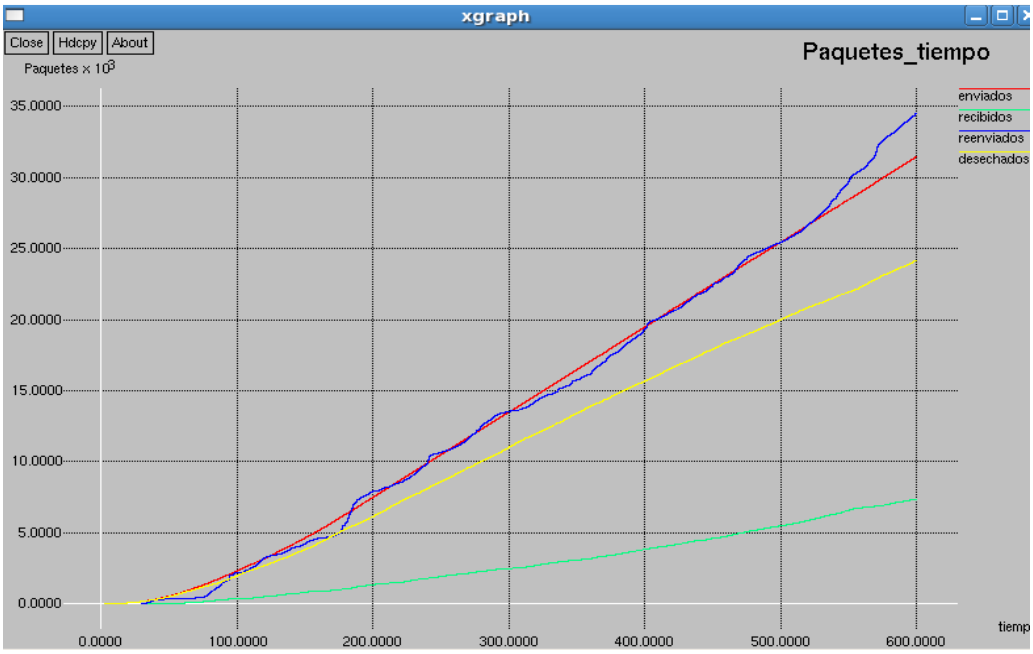


Figura III.23 Resultados OLSR5

En la figura III.24 podemos observar el número de paquetes a nivel de protocolo de ruteo en donde se puede observar la cantidad de paquetes enviados, recibidos, reenviados y desechados a lo largo del tiempo de simulación.

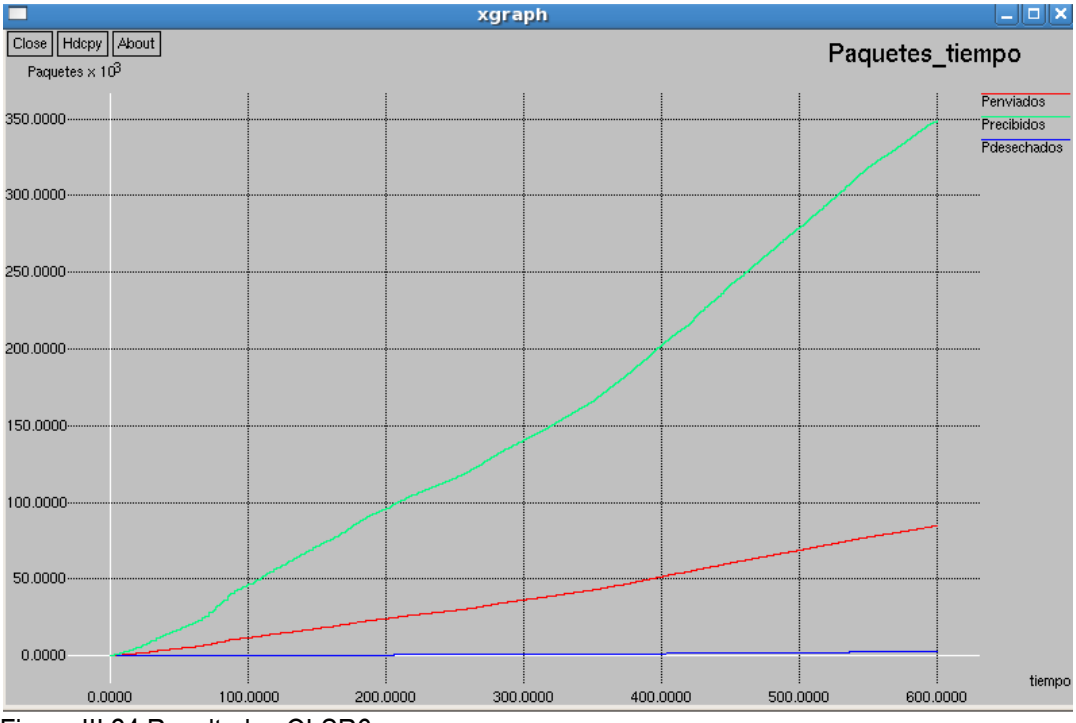


Figura III.24 Resultados OLSR6

3.5. Comparación de resultados

Dentro de la simulación se realizaron distintas pruebas de los distintos protocolos de ruteo utilizando distinto tráfico para obtener resultados. Para obtener resultados más precisos se realizo las simulaciones tres veces utilizando los mismos parámetros en cuanto a cantidad de nodos, tipo de protocolo, tipo de tráfico y demás parámetros básicos, y se varió únicamente los movimientos de los nodos junto con los enlaces aleatorios. De esta manera se realizó la misma simulación pero bajo distintas condiciones, con lo que se obtuvo resultados similares en proporción, a continuación se obtuvo un resultado sacando un promedio de estos valores, de esta manera se tienen datos más precisos en cuanto al funcionamiento de la red.

Para cada protocolo de ruteo se simulo con tráfico FTP y CBR para explotar la red de distintos puntos de vista que pueden presentarse bajo distintas condiciones de aplicación.

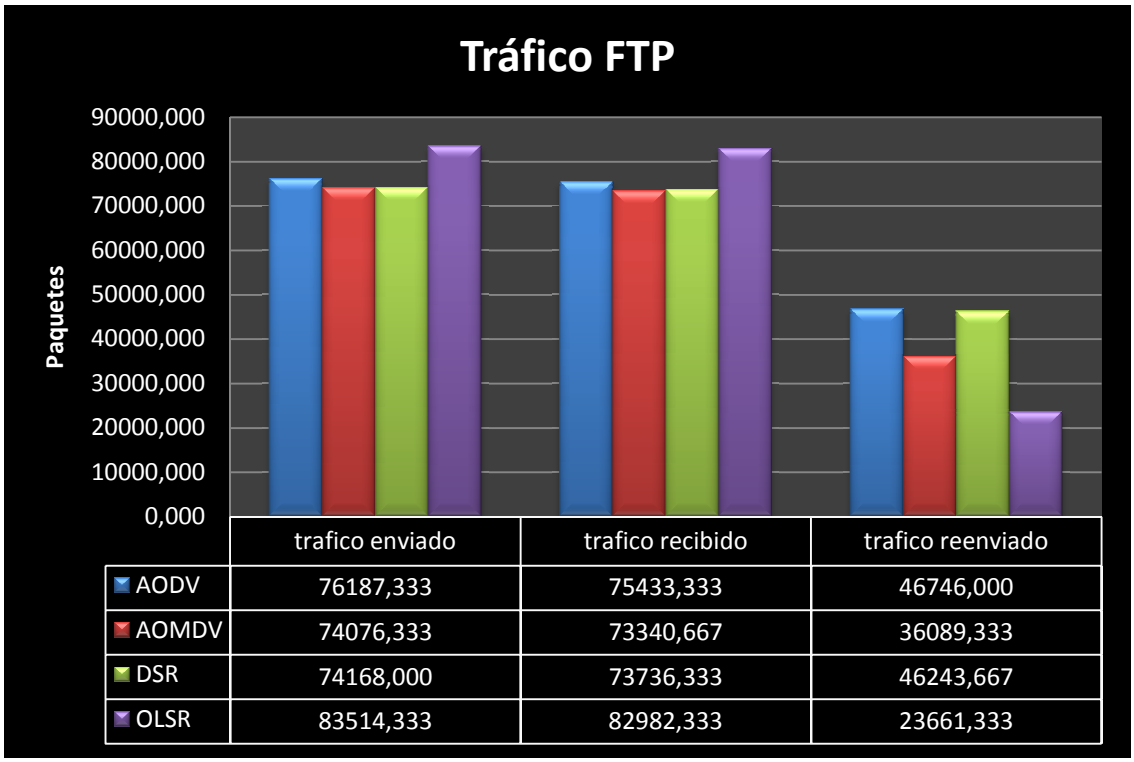
De esta manera se obtuvieron los siguientes resultados:

3.5.1. Tráfico FTP

3.5.1.1. Valores a nivel de paquetes de aplicación

FTP				
	AODV	AOMDV	DSR	OLSR
trafico enviado	76187,333	74076,333	74168,000	83514,333
trafico recibido	75433,333	73340,667	73736,333	82982,333
trafico reenviado	46746,000	36089,333	46243,667	23661,333
trafico desechado	570,667	553,333	357,667	341,333

Tabla III.X. Comparación trafico FTP



	trafico enviado	trafico recibido	trafico reenviado
AODV	76187,333	75433,333	46746,000
AOMDV	74076,333	73340,667	36089,333
DSR	74168,000	73736,333	46243,667
OLSR	83514,333	82982,333	23661,333

Figura III.25 Cuadro comparativo 1

En el primer cuadro comparativo podemos ver la capacidad del protocolo para establecer los enlaces, permitiendo de esta manera enviar distinto numero de paquetes, de esta manera podemos ver el protocolo OLSR tiene una mayor capacidad para establecer un enlace ya que logra enviar más tráfico en la red en el caso de FTP y vemos que el protocolo con menos éxito en el envió de paquetes fue el protocolo AOMDV.

Existe una estrecha relación entre los paquetes enviados y recibidos, el hecho de recibir un paquete implica estabilidad del enlace y capacidad de cambio de rutas ante variaciones de la red. De esta manera vemos que OLSR ampliamente supera a los demás protocolos en recepción de paquetes debido a su capacidad de envío.

También podemos observar en la cantidad de paquetes reenviados en la red que el protocolo AODV es el que más maquetes posee, de esta manera podemos ver la consecuencia al crear rutas en las tablas de ruteo por parte de los protocolos, y asa percibir que protocolo escoge rutas

más cortas para la comunicación de extremo a extremo. Así, como resultado tenemos que OLSR realiza su tarea con menos reenvío de paquetes lo que significa que tiene rutas con menos saltos mientas que DSR y AOMDV escogen rutas de mayores saltos en la red.

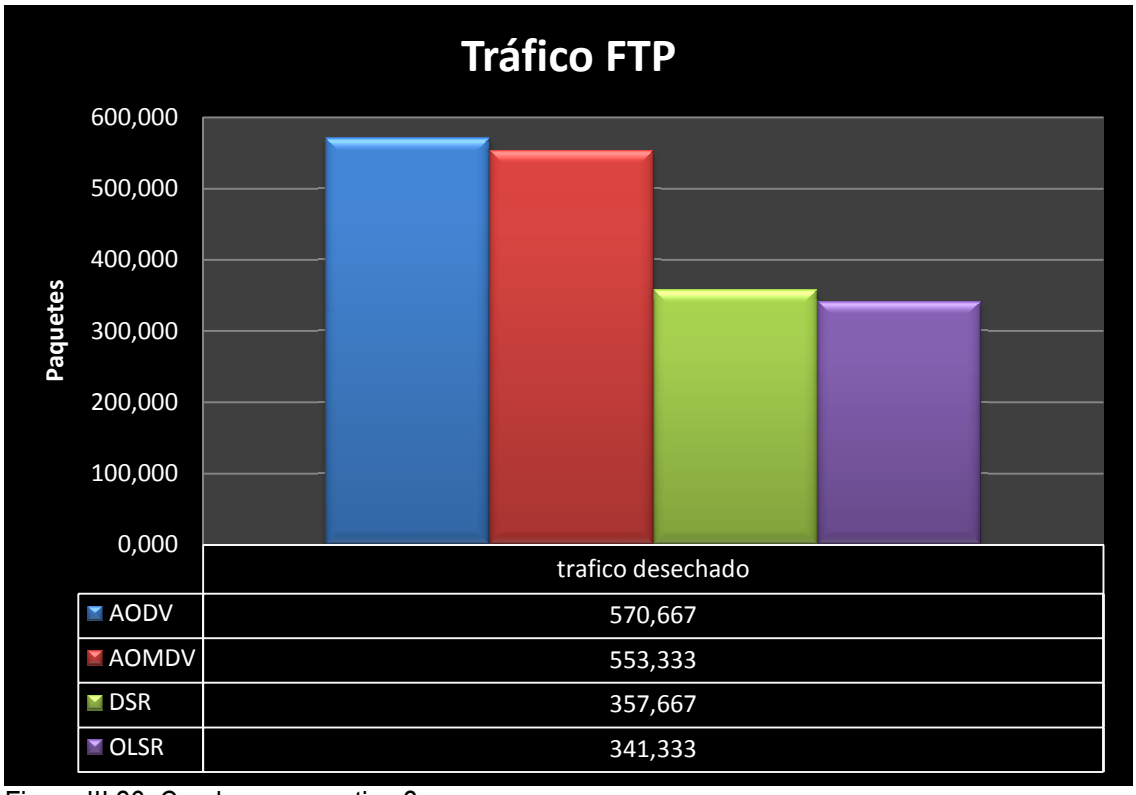


Figura III.26. Cuadro comparativo 2

La cantidad de paquetes caídos nos muestra la estabilidad que brinda cada protocolo a la red, así como su capacidad de respuesta ante cambios en la topología. Podemos observar un mejor desempeño de los protocolos DSR y OLSR respecto a los protocolos AOMDV y AODV.

3.5.1.2. Tráfico a nivel de protocolo de ruteo

	AODV	AOMDV	DSR	OLSR
protocolo enviado	15324,667	23312,000	9379,000	21437,000
protocolo recibido	60345,333	96681,333	21180,667	99488,667
protocolo reenviado	567,000	252,333	3005,000	0,000
protocolo desechado	5657,333	6646,333	1589,333	4970,667

Tabla III.XI. Comparación a nivel de protocolo en FTP

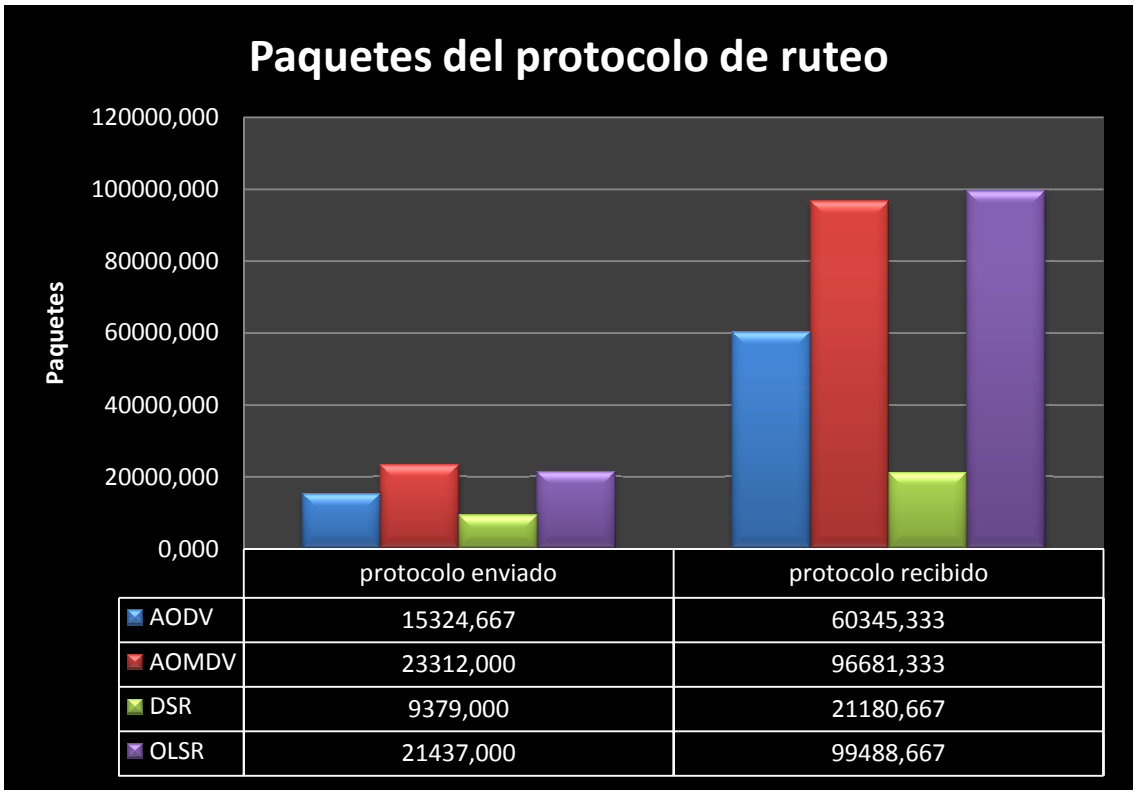
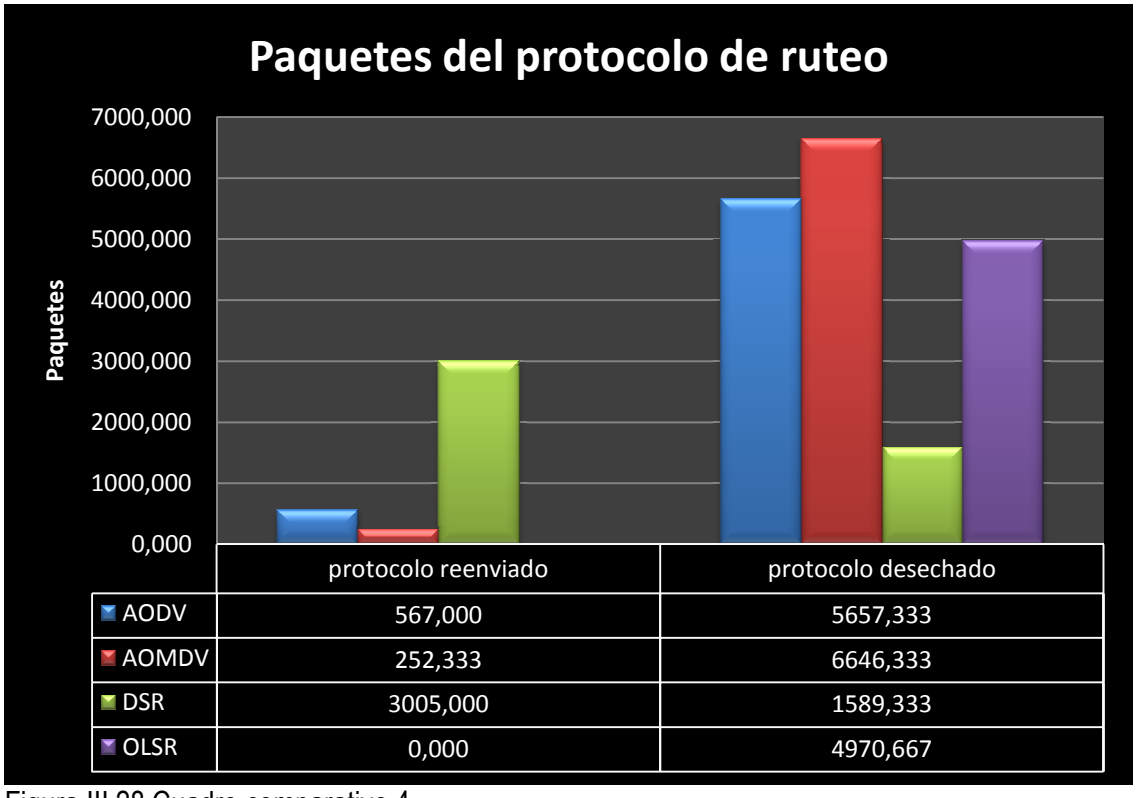


Figura III.27 Cuadro comparativo 3

La cantidad de tráfico generado por parte de los protocolos de ruteo es fundamental en un análisis de desempeño del mismo dado que la cantidad de tráfico que pueden producir puede llegar a afectar considerablemente el ancho de banda de una red inalámbrica. La cantidad de trafico generada está ligada directamente al algoritmo que dirige el funcionamiento del protocolo, mientras menos trafico genere el protocolo indica que es más simple y no realiza actualizaciones

constantes, mientras que si tenemos un protocolo as complejo buscando que sea más confiable podemos llegar a inundar la red constante con paquetes generados en busca de mantener tablas de ruteo lo más confiables posible.

Como resultado se obtuvo que el protocolo DSR por gran diferencia genera menos tráfico en la red, como consecuencia el numero de paquetes recibidos es mucho más bajo que el de los otros protocolos, también podemos observar que la complejidad de protocolos como OLSR y AOMDV provoca que generen un alto trafico de configuración de la red. Finalmente podemos ver que la cantidad de paquetes recibidos es mucho más alta que la de paquetes enviados, esto tiene mucha lógica al recordar que muchas de las peticiones para encontrar las rutas se realizan mediante broadcast lo que implica un envío pero muchas recepciones de los paquetes



	protocolo reenviado	protocolo desechado
AODV	567,000	5657,333
AOMDV	252,333	6646,333
DSR	3005,000	1589,333
OLSR	0,000	4970,667

Figura III.28 Cuadro comparativo 4

En el cuadro comparativo 4 tenemos la cantidad de paquetes reenviados y desechados de protocolo de ruteo, estos vienen dados por la estructura misma del protocolo, podemos ver que

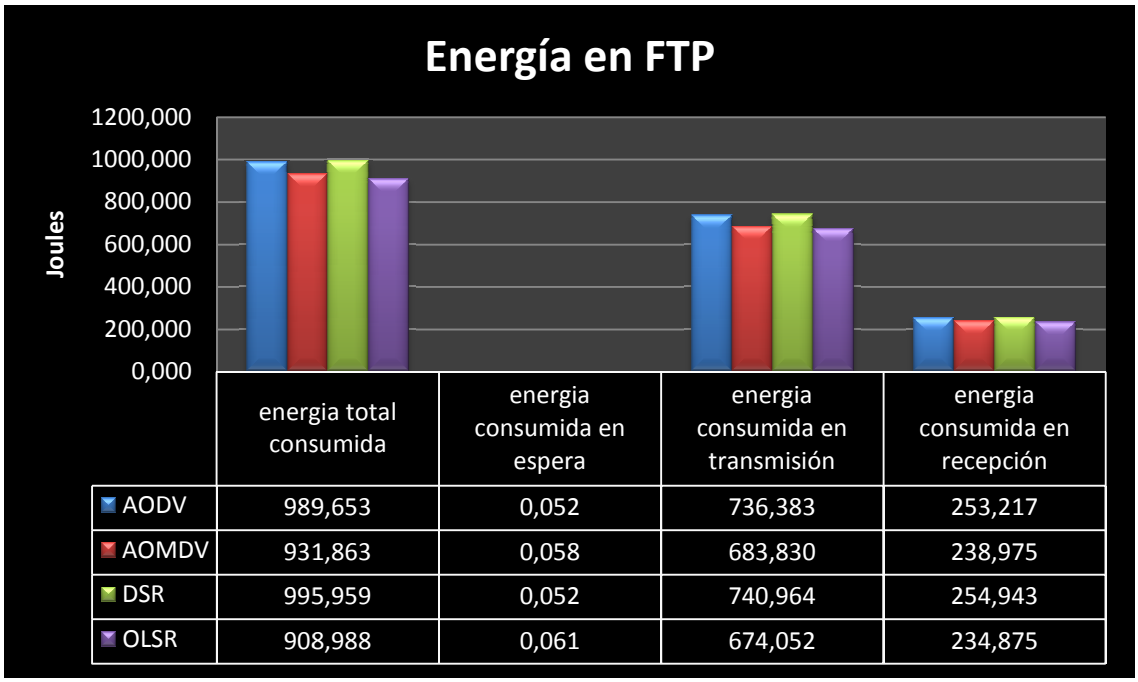
DSR es el más paquetes reenviados posee esto se debe a como crea sus tablas de ruteo, su comportamiento de reenviar viene dado por su algoritmo que reenvía la información para actualizar sus rutas, mientras que OLSR no posee paquetes reenviados ya que solo se comunica con sus nodos adyacentes.

En lo que respecta a paquetes caídos viene dado por los paquetes broadcast que se enviaron y no lograron llegar a su destino.

3.5.1.3. Consumo de energía

	AODV	AOMDV	DSR	OLSR
energía total consumida	989,653	931,863	995,959	908,988
energía consumida en espera	0,052	0,058	0,052	0,061
energía consumida en transmisión	736,383	683,830	740,964	674,052
energía consumida en recepción	253,217	238,975	254,943	234,875

Tabla III.XII valores de energía



	energia total consumida	energia consumida en espera	energia consumida en transmisión	energia consumida en recepción
AODV	989,653	0,052	736,383	253,217
AOMDV	931,863	0,058	683,830	238,975
DSR	995,959	0,052	740,964	254,943
OLSR	908,988	0,061	674,052	234,875

Figura III.29 Cuadro comparativo 5

Dado que el consumo de energía está ligado directamente a la cantidad de información que se transmite en la red ya que el consumo es resultado de la potencia necesaria para satisfacer la cobertura, vemos que la transmisión de cada paquete sea de información o del protocolo implica un aumento en el consumo. Por ello la energía consumida es el resultado de la cantidad de información generada no solo por información transportada sino por las demandas del protocolo.

Como resultado tenemos que el protocolo de mayor consumo en la red es DSR debido a su alto índice de reenvío de paquetes seguido de cerca por AODV y el protocolo de menor consumo es OLSR como consecuencia de que no reenvía paquetes en la creación y actualización de sus tablas de ruteo seguido por AOMDV.

3.5.2. Tráfico CBR

3.5.2.1. Valores a nivel de paquetes de tráfico

	AODV	AOMDV	CBR	OLSR
trafico enviado	30818,667	30870,667	30891,333	30868,000
trafico recibido	13662,667	10137,000	16289,667	6629,000
trafico reenviado	56283,333	44242,000	93052,333	25893,000
trafico desechado	1534,667	4804,333	743,667	20690,333

Tabla III.XIII. Valores a nivel de paquetes de trafico CBR

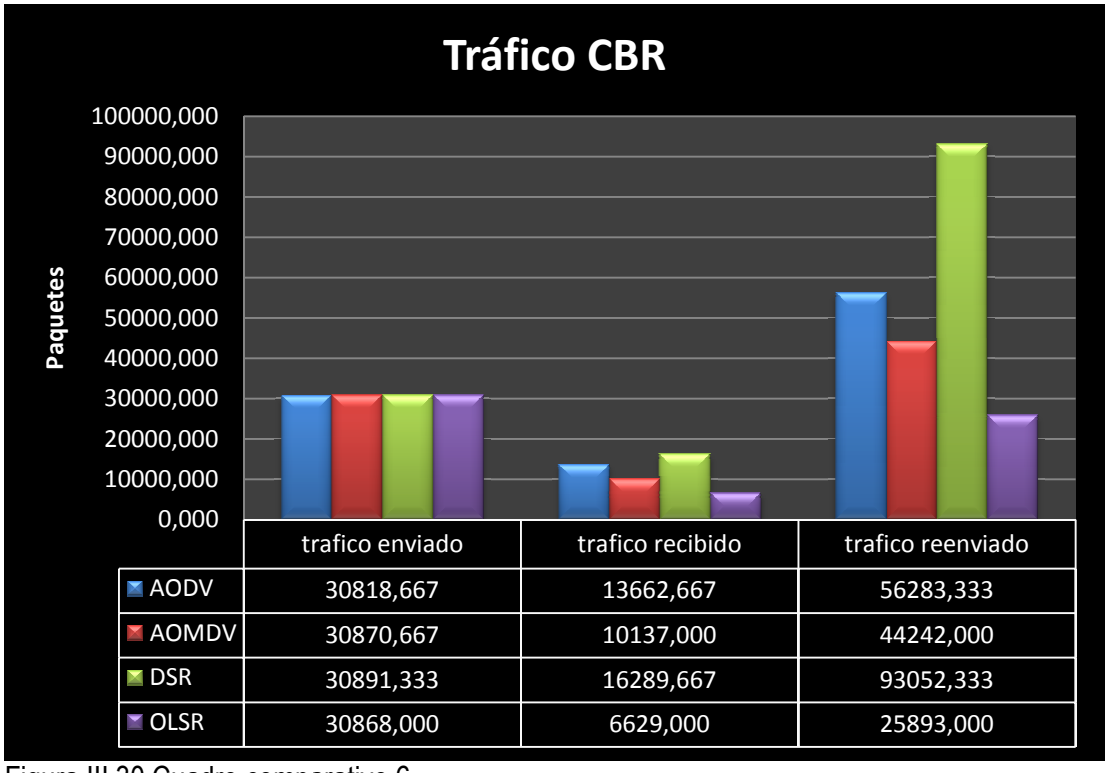


Figura III.30 Cuadro comparativo 6

A diferencia del trafico FTP, el trafico CBR que funciona bajo UDP es mucho más sencillo por lo que se realizo la simulación durante diez minutos con cincuenta nodos, de esta manera al no tener que establecer un enlace orientado a la conexión la cantidad de paquetes enviados es

prácticamente la misma, lo que permite apreciar de mejor de mejor manera el desempeño de la red durante el tiempo de simulación ya que no existe una relación directa entre paquetes enviados y recibidos, así podemos observar claramente la fiabilidad del protocolo respecto al éxito del transporte de la información en el medio.

De esta manera tenemos como resultado que el protocolo con mayor existo en la recepción de paquetes es DSR con una gran diferencia con el que tuvo menor éxito que fue OLSR que no supo responder de gran manera en una red más extensa con un protocolo no orientado a la conexión.

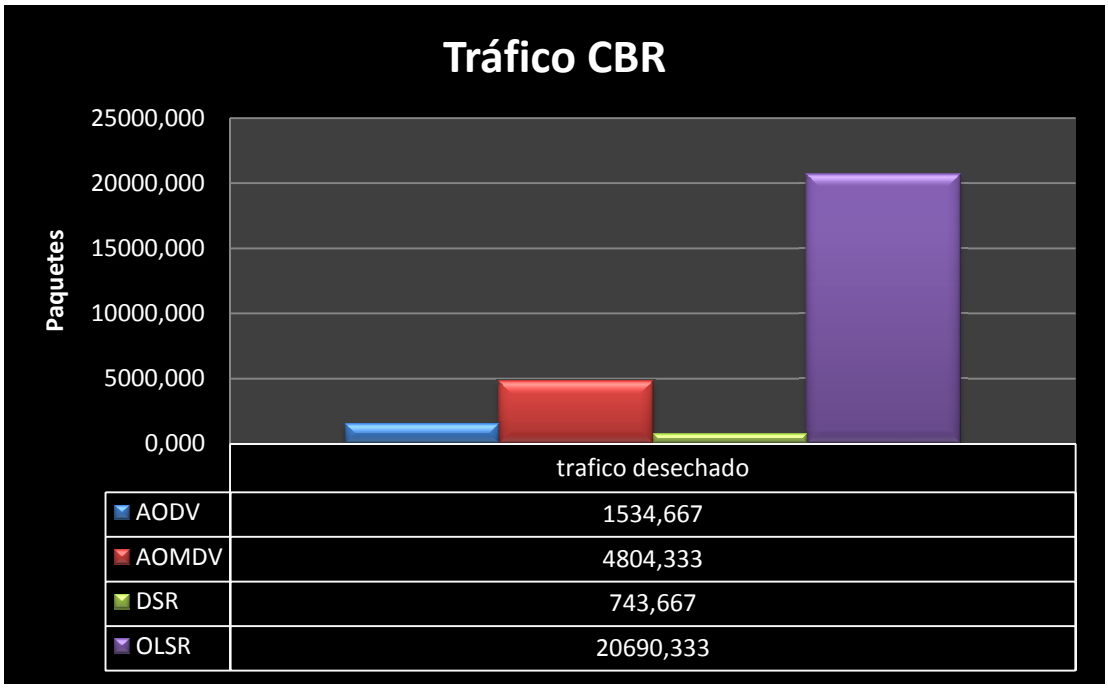


Figura III.31 Cuadro comparativo 7

La cantidad de paquetes caídos es resultado del análisis anterior, consecuencia del desempeño de los protocolos tenemos como resultado pocos paquetes caídos con DSR y un alto índice de paquetes caídos con OLSR.

3.5.2.2. Tráfico a nivel de protocolos de ruteo

	AODV	AOMDV	CBR	OLSR
protocolo enviado	120534,333	250272,333	62117,667	75711,333
protocolo recibido	428478,667	855684,667	256526,000	294811,333
protocolo reenviado	5182,333	1918,000	40019,333	0,000
protocolo desechado	14250,000	41015,000	9470,000	1970,333

Tabla III.XIV. Valores a nivel de protocolo en CBR

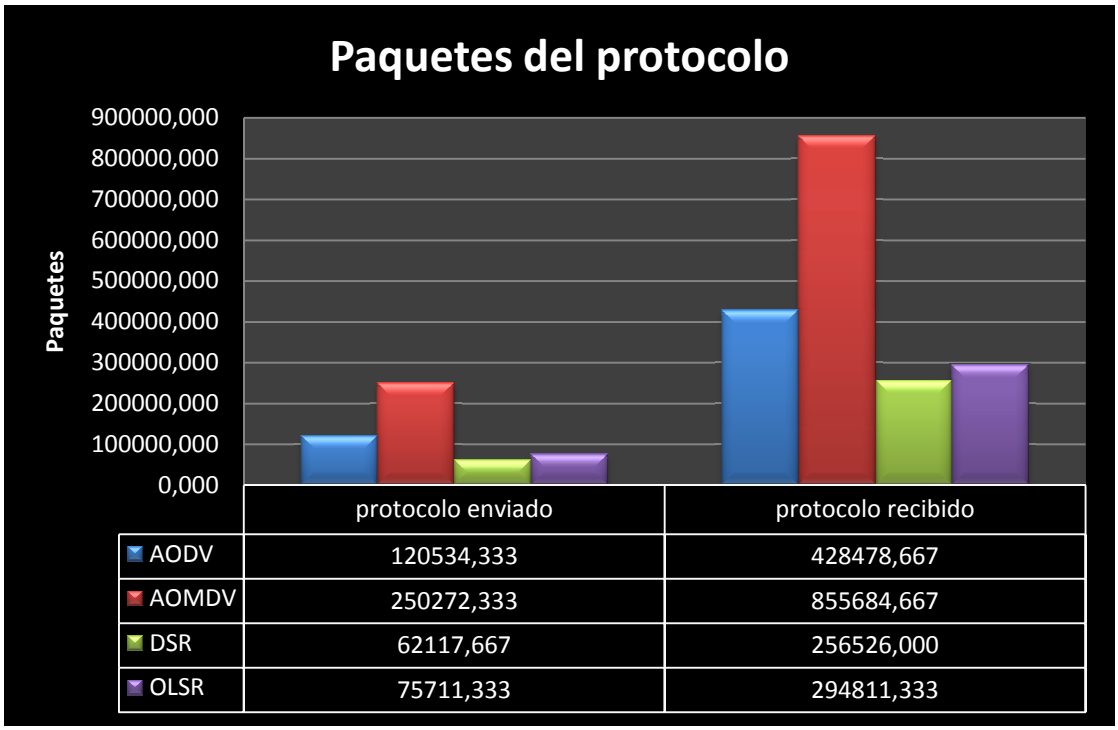
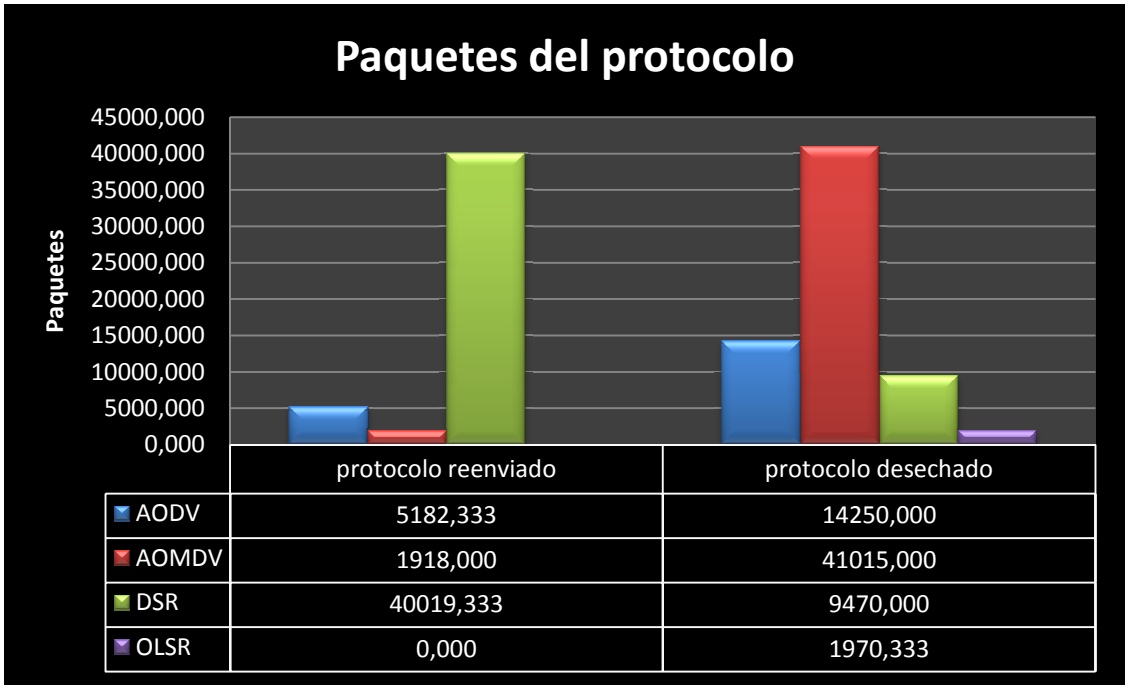


Figura III.32 Cuadro comparativo 8

A nivel de tráfico de protocolo podemos observar que para esta red más extensa AOMDV supera ampliamente a los otros protocolos en la generación de tráfico, lo que puede llegar a ser un inconveniente para el mismo. De la misma manera que en la red de treinta nodo podemos observar la gran mayor cantidad de paquetes recibidos debido a que se envía mucho de los

paquetes mediante broadcast. También podemos notar que DSR genera menos trafico que los demás protocolos, consecuentemente tenemos menos tráfico en la red.

Los protocolos DSR y OLSR tienen valores similares en cuanto a la generación de paquetes de configuración de la red, de esta manera en el numero de paquetes recibidos no existe gran diferencia pero como consecuencia del broadcast mientras aumenta el tiempo de simulación esta diferencia se hará más grande



	protocolo reenviado	protocolo desechado
AODV	5182,333	14250,000
AOMDV	1918,000	41015,000
DSR	40019,333	9470,000
OLSR	0,000	1970,333

Figura III.33 Cuadro comparativo 9

De igual manera a la simulación de trafico FTP podemos observar un resultado similar, en el que DSR por su estructura supera ampliamente la cantidad de paquetes reenviados en la red, y por la manera de crear tablas, tenemos gran cantidad de paquetes caídos en la simulación de AOMDV. También podemos notar que OLSR a pesar de ser una red de mayor tamaño continúa sin reenviar paquetes por su estructura.

3.5.2.3. Consumo de energía

	AODV	AOMDV	CBR	OLSR
energía total consumida	507,178	639,592	722,830	332,430
energía consumida en espera	0,386	0,380	0,372	0,399
energía consumida en transmisión	386,097	495,691	553,470	251,424
energía consumida en recepción	120,695	143,521	168,989	80,607

Tabla III.XV. Valores de energía en CBR

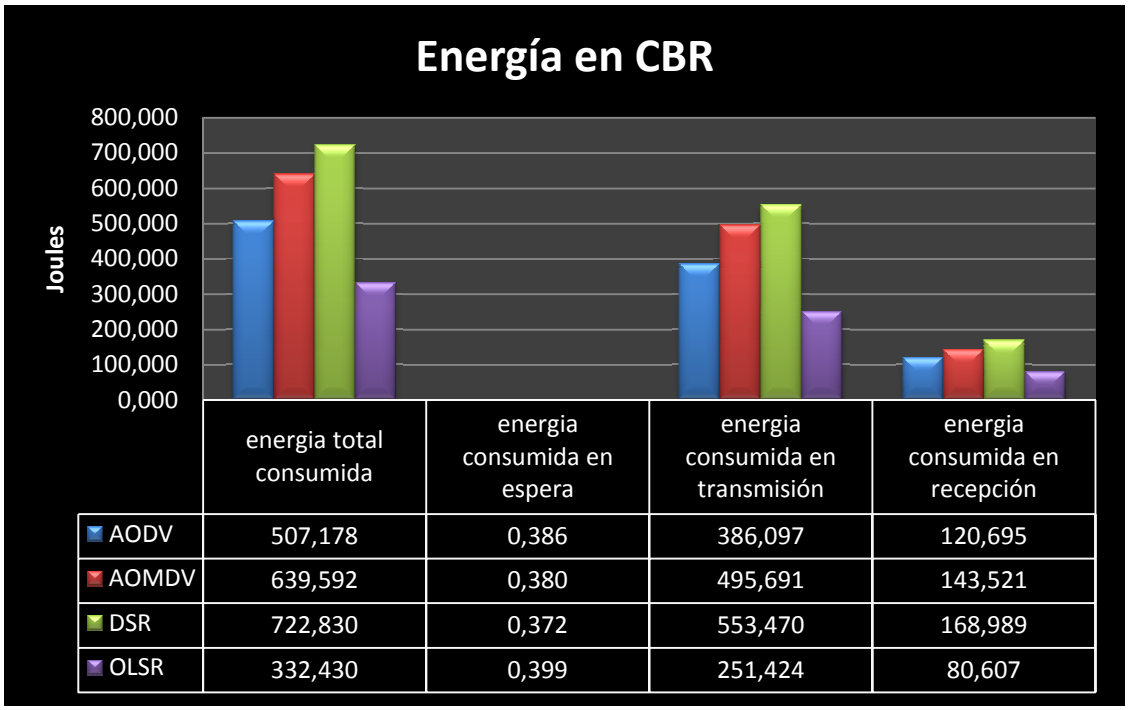


Figura III.34 Cuadro comparativo 10

Finalmente en lo que respecta al consumo de la energía, hemos visto que viene ligado a la cantidad de paquetes en la red sean estos generados por el protocolo de aplicación o por el de ruteo.

Como resultado tenemos que DSR continua siendo el de mayor consumo debido a su constante reenvío de paquetes en la red y un menor consumo por parte de OLSR gracias a que no reenvía paquetes pero también debido a su poco éxito en la recepción de paquetes.

CAPITULO IV

ESTUDIO COMPARATIVO DE PROTOCOLOS DE RUTEO EN REDES AD HOC APLICADO A REDES MOVILES Y COMPROBACION DE LA HIPÓTESIS

El estudio comparativo se realiza entre los protocolos AODV, AOMDV, DSR y OLSR, de esta manera se busca comparar sus capacidades, para ello se realizó una simulación en NS2 en la que se pudo obtener valores cuantitativos en base a varias pruebas de la simulación, de esta manera en esta sección se valorará de manera cualitativa a los protocolos para más tarde comprobar la hipótesis mediante el método de chi cuadrado en base a la variable independiente, dependiente y a sus respectivos indicadores.

4.1. SISTEMA HIPOTÉTICO

4.1.1. HIPÓTESIS DE LA INVESTIGACIÓN

Con el estudio entre los distintos protocolos de enrutamiento para redes móviles ad hoc se podrá comparar las capacidades de cada protocolo para encontrar el que mejor se adapta en un ambiente propuesto.

4.1.2. OPERACIONALIZACIÓN DE LAS VARIABLES

En las tablas se presentan la operacionalización conceptual y metodológica de las variables, las mismas que se han identificado de acuerdo a la hipótesis:

VARIABLE	TIPO	DEFINICIÓN
V1. Análisis de los protocolos de enrutamiento para redes móviles AdHoc	Independiente	Estudio de los diferentes protocolos: AODV AOMDV DSR OLSR
V2. Fiabilidad	Dependiente	Se refiere al grado de confianza que se puede tener en un protocolo al momento de enviar y recibir paquetes en la red evitando la perdida de información
V3. Carga del protocolo	Dependiente	Se refiere a la cantidad de información que envía el protocolo para establecer los

		enlaces que permiten la comunicación entre los nodos.
V4. Consumo de energía	Dependiente	Consiste en la cantidad de energía que se consume en la red durante su funcionamiento.

Tabla IV.I: Operacionalización conceptual de las variables.

4.1.3. OPERACIONALIZACIÓN METODOLÓGICA

Variables	Categoría	Indicadores	Técnicas	Fuente de Verificación
V1. Independiente Análisis de los protocolos de ruteo	Compleja	I1. Capacidad de adaptación I2. Manejo de rutas I3. Ancho de banda	Observación Razonamiento Recopilación de información Análisis Lectura científica	Información bibliográfica (Libros, Internet, Tesis)

Tabla IV.II: Operacionalización Metodológica de la variable independiente

Variable	Categoría	Indicadores	Técnica	Fuente de Verificación
V2. Dependiente Fiabilidad	Compleja	I4.Capacidad de envío I5.Capacidad de recepción I6. Paquetes perdidos	Pruebas Resultados simulación Conclusiones	Simulación NS2

Tabla IV.III: Operacionalización Metodológica de la variable dependiente FIABILIDAD

Variable	Categoría	Indicadores	Técnica	Fuente de Verificación
V3. Dependiente Carga del protocolo	Compleja	I7. Cantidad de paquetes generados I8. Cantidad de paquetes recibidos	Pruebas Conclusiones	Simulación NS2

Tabla IV.IV: Operacionalización Metodológica de la variable dependiente CARGA DEL PROTOCOLO

Variable	Categoría	Indicadores	Técnica	Fuente de Verificación
V4. Dependiente Consumo de energía	Compleja	I9. Energía consumida en envío I10. Energía consumida en recepción I11. Energía consumida en espera	Pruebas Conclusiones	Simulación NS2

Tabla IV.V: Operacionalización Metodológica de la variable dependiente CONSUMO DE ENERGÍA

4.1.4. DESCRIPCIÓN DE LAS VARIABLES Y SUS RESPECTIVOS INDICADORES

Para el estudio comparativo de protocolos de ruteo para redes móviles AdHoc se determinaron varios indicadores que servirán de base para comparar las distintas capacidades de los mismos.

4.1.4.1. V1. VARIABLE INDEPENDIENTE: Análisis de los protocolos de ruteo

4.1.4.1.1. INDICADORES

I1. Capacidad de adaptación

Cuando la red presenta movilidad, se debe tomar en cuenta que se trabaja sobre una topología en constante cambio por ello el protocolo debe ser capaz de adaptarse a los cambios constantes en sus tablas de ruteo. Esto puede apreciarse en la cantidad de trafico que el escenario fue capaz de manejar.

I2. Manejo de rutas

Cada protocolo tiene la capacidad de determinar los caminos a seguir por los paquetes de información. Al ser una red tipo mesh, se tiene múltiples caminos para la comunicación. El protocolo debe seleccionar un camino a seguir. Esto se puede determinar en base al número de saltos que existe entre cada enlace al momento de seleccionar rutas.

I3. Ancho de banda

Al tener un manejo diferente de tráfico en la red por parte de cada protocolo, se puede constatar que existe una variación en el ancho de banda real que existe en cada caso. Este ancho de banda se expresa en Kbps o Mbps

4.1.4.2. V2. VARIABLE DEPENDIENTE: FIABILIDAD

4.1.4.2.1. INDICADORES

I4.Capacidad de envío

Cada protocolo crea enlaces de diferente manera por ello solo envía la información si existe un enlace disponible. De esta manera se contabiliza el número de paquetes que se lograron generar para el envío.

I5.Capacidad de recepción

Se refiere al número de paquetes que lograron llegar a su destino, de esta manera se los contabiliza al llegar al nodo receptor

I6. Paquetes perdidos

Por diferentes motivos como rutas inexistentes, colas, retrasos, etc. Los paquetes que no llegan a su destino se desechan o se caen en la red, de esta manera se los puede contabilizar.

4.1.4.3. V2. VARIABLE DEPENDIENTE: CARGA DEL PROTOCOLO

4.1.4.3.1. INDICADORES

I7. Cantidad de paquetes generados

Para descubrir la topología de la red y generar las tablas de enrutamiento en cada nodo, el protocolo genera su propio tráfico en la red, de esta manera se generan paquetes en la mayoría de los casos de tipo broadcast.

I8. Cantidad de paquetes recibidos

Debido a una generación de paquetes en su mayoría de tipo broadcast se obtiene una alta tasa de recepción de paquetes para establecer la topología y enlaces de la red por parte de los nodos

4.1.4.4. V2. VARIABLE DEPENDIENTE: CONSUMO DE ENERGÍA

4.1.4.4.1. INDICADORES

I9. Energía consumida en envío

Para enviar paquetes a través de la red, el nodo inalámbrico requiere de energía que le permita brindar una cobertura para establecer la comunicación con un nodo cercano, de esta manera mediante una antena, se logra enviar la información. Esta energía se expresa en Joules(W.s).

I10. Energía consumida en recepción

Cuando un nodo detecta que otro está enviándole información, requiere de un monto de energía para procesar el paquete recibido, de esta manera existe un consumo de energía ligado a la recepción de paquetes. Esta energía se expresa en Joules(W.s).

I11. Energía consumida en espera

Cuando un nodo no se encuentra enviando o recibiendo paquetes, permanece en un estado de espera, escuchando el medio, de esta manera existe un consumo pasivo mientras no existe una comunicación explicita entre los nodos. Esta energía se expresa en Joules(W.s).

4.2. POBLACIÓN Y MUESTRA

La población en este estudio está compuesta por todos protocolos de ruteo para redes móviles AdHoc

MUESTRA

- Para el estudio de protocolos de enrutamiento para redes móviles AdHoc, la muestra estuvo constituido por los protocolos AODV, AOMDV, DSR, OLSR.

- Para determinar la fiabilidad, la carga del protocolo y el consumo de energía de los protocolos, las pruebas se las realizaron mediante la simulación de escenarios en NS2 en los cuales se implementó los protocolos obteniendo datos cuantitativos en base a número de paquetes.

4.3. ESTUDIO COMPARATIVO

Los protocolos AODV, AOMDV, DSR y OLSR serán comparados en base a sus características mediante cuadros comparativos, de esta manera se los califica cualitativamente en base a criterio del autor basándose en los resultados obtenidos en la simulación en NS2 junto con la información teórica de los protocolos con sus algoritmos, de esta manera se consigue interpretar objetivamente los resultados que se pueden extraer en base a la simulación.

4.3.1. ESTUDIO COMPARATIVO DE LA VARIABLE INDEPENDIENTE

Para la valoración cualitativa de los indicadores de la variable independiente se utilizará la siguiente escala:

0	Muy deficiente
1	deficiente
2	regular
3	medio
4	bueno
5 en adelante	Muy bueno

Tabla IV.VI: Escala de calificación

INDICADOR 1: Capacidad de adaptación

La capacidad de adaptación se refiere al comportamiento del protocolo frente a cambio en la topología debido al movimiento de los nodos, la comunicación en la red puede cortarse mientras el protocolo intenta encontrar una ruta que permita restablecer un enlace. Como consecuencia la red se encuentra activa durante un menor periodo de tiempo transmitiendo un menor número de paquetes de datos y mas paquetes de configuración de protocolo.

	AODV	AOMDV	DSR	OLSR
TCP	3	3	5	3

UDP	2	1	6	0

Tabla IV.VII: Capacidad de adaptación

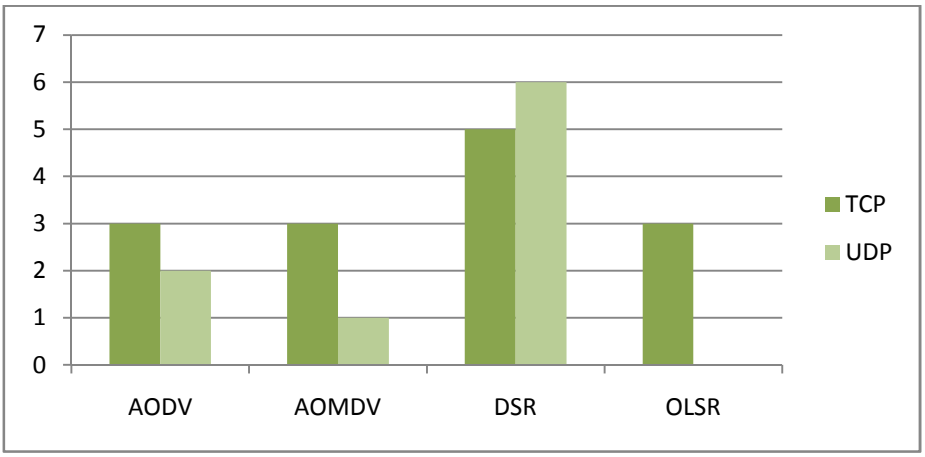


Figura IV.1: Capacidad de adaptación

Interpretación:

La Figura IV.1 nos muestra que el protocolo DSR tiene una buena adaptación al trabajar con tráfico TCP o UDP mientras que AODV y AOMDV tienen un funcionamiento similar con un desempeño ligeramente superior por parte de AODV al no ser multi camino, lo que le permite una adaptación más rápida ante los cambios. OLSR tiene una buena adaptación al trabajar con TCP pero al momento de trabajar con UDP tiene una baja respuesta ante los cambios rápidos que representa.

INDICADOR 2: MANEJO DE RUTAS

El manejo de rutas se evalúa en función al número de saltos que utiliza el protocolo para transportar un paquete, al tener enlaces del mismo tipo, tenemos que la ruta más corta es aquella que realiza el menor número de saltos. Esta información se obtiene en base a los paquetes reenviados en la red, siendo menos eficiente aquel protocolo que reenvió más paquetes, de esta manera tenemos los siguientes resultados:

	TCP(Paquetes)	UDP(Paquetes)
0	>50 mil	>80 mil
1	Entre 45 y 50mil	Entre 70 y 80mil
2	Entre 40 y 45mil	Entre 60 y 70mil
3	Entre 35 y 40mil	Entre 50 y 60mil
4	Entre 30 y 35mil	Entre 40 y 50mil
5 en adelante	<30000	<40000

Tabla IV.VIII: Equivalencias1

	AODV	AOMDV	DSR	OLSR
TCP	1	3	1	6
UDP	3	4	0	6

Tabla IV.IX: Manejo de rutas

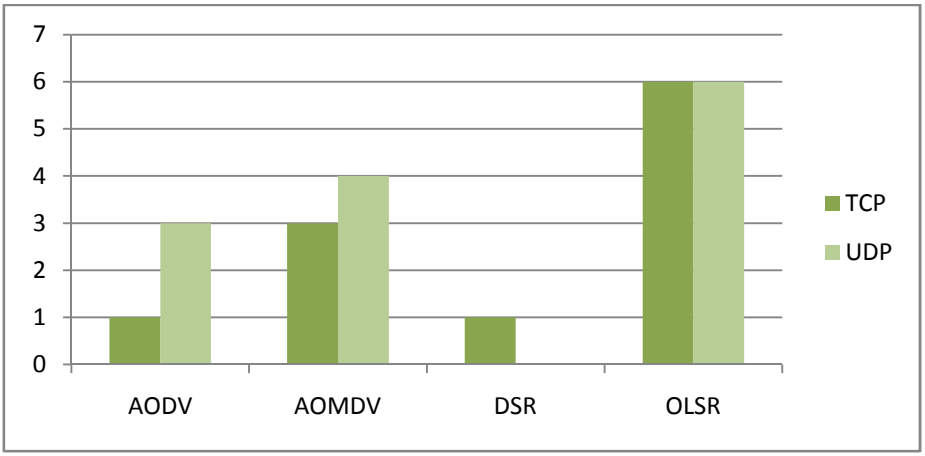


Figura IV.2: Manejo de rutas

Interpretación:

Como podemos observar en la Figura IV.2 el protocolo OLSR tiene una gran capacidad para determinar rutas eficientes al igual que AOMDV por su característica de seleccionar caminos múltiples, mientras que DSR o AODV tienden a seleccionar rutas menos eficientes debido a sus algoritmos de enrutamiento.

INDICADOR 3: ANCHO DE BANDA

El ancho de banda analizado en esta sección corresponde al ancho de banda con el que operan los nodos a partir de un ancho de banda teórico asignado a cada enlace inalámbrico de 11Mbps.

Debe tomarse en cuenta los factores que reducen el ancho de banda considerable como la distancia entre los nodos, así como la movilidad constante de la red.

	TCP(Kbps)	UDP(Kbps)
0	<1000	<20
1	Entre 1000 y 1025	Entre 20 y 40
2	Entre 1025 y 1050	Entre 40 y 60
3	Entre 1050 y 1075	Entre 60 y 80
4	Entre 1075 y 1100	Entre 80 y 100
5 en adelante	>1100	>100

Tabla IV.X: Equivalencias2

	AODV	AOMDV	DSR	OLSR
TCP	2	1	1	7
UDP	4	3	7	2

Tabla IV.XI: Tamaño del escenario

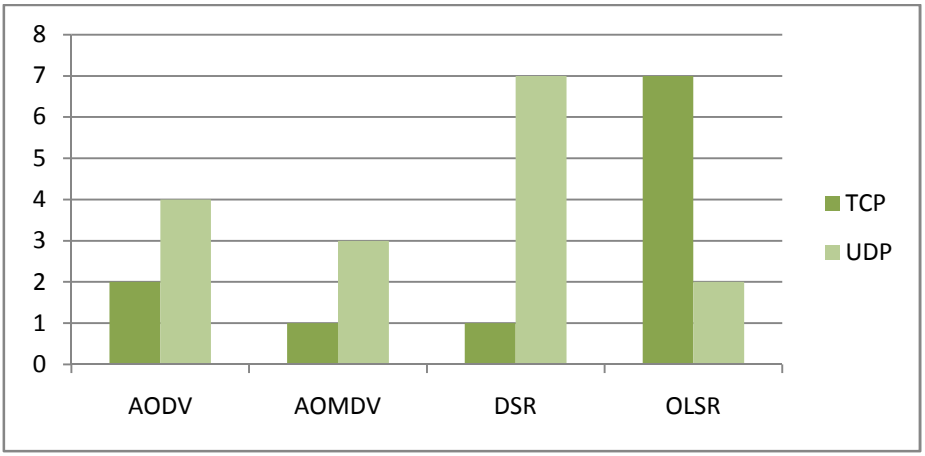


Figura IV.3: Tamaño del escenario

Interpretación:

En este grafico se puede constatar que el protocolo con mejor manejo de ancho de banda en TCP es el protocolo OLSR, mientras que DSR se desempeña mejor en UDP, los protocolos AODV y AOMDV tienen un manejo del ancho de banda similar.

4.3.1.1. TABLA DE RESUMEN DE LA VARIABLE INDEPENDIENTE

V1. Análisis de los protocolos de ruteo para redes móviles AdHoc	PARAMETROS	AODV	AOMDV	DSR	OLSR
I1. Capacidad de adaptación	TCP	3	3	5	3
	UDP	2	1	6	0
I2. Manejo de rutas	TCP	1	3	1	6
	UDP	3	4	0	6
I3.Ancho de banda	TCP	2	1	1	7
	UDP	4	3	7	2
TOTAL		15	15	20	24

Tabla IV.XII: Resumen de la variable independiente

Como podemos observar en la Tabla IV.7 El protocolo que tiene un desempeño en general mejor en base a la variable independiente es DSR seguido por OLSR. AODV y AOMDV tienen igual puntaje inferior a los otros protocolos.

4.3.2. ESTUDIO COMPARATIVO DE LAS VARIABLES DEPENDIENTES

Para el análisis de las variables independientes se obtuvo los datos mediante la simulación y pruebas de los protocolos en el simulador NS2 y mediante la extracción de resultados mediante filtros AWK.

4.3.2.1. V2: FIABILIDAD

INDICADOR 4: Capacidad de envío

La capacidad de envío se refiere al número de paquetes que un nodo logro enviar (sin tomar en cuenta si se receptaron o no). Esto permite determinar que protocolo configura la red de manera más rápida y tiene más tiempo para establecer la comunicación.

	TCP(Paquetes)	UDP(Paquetes)
0	<70 mil	<30800
1	Entre 70 y 72mil	Entre 30800 y 30820
2	Entre 72 y 74mil	Entre 30820 y 30840
3	Entre 74 y 76mil	Entre 30840 y 30860
4	Entre 76 y 78mil	Entre 30860 y 30880
5 en adelante	>78000	>30880

Tabla IV.XIII: Equivalencias3

	AODV	AOMDV	DSR	OLSR
TCP	4	3	3	7
UDP	1	4	6	4

Tabla IV.XIV: Capacidad de envío

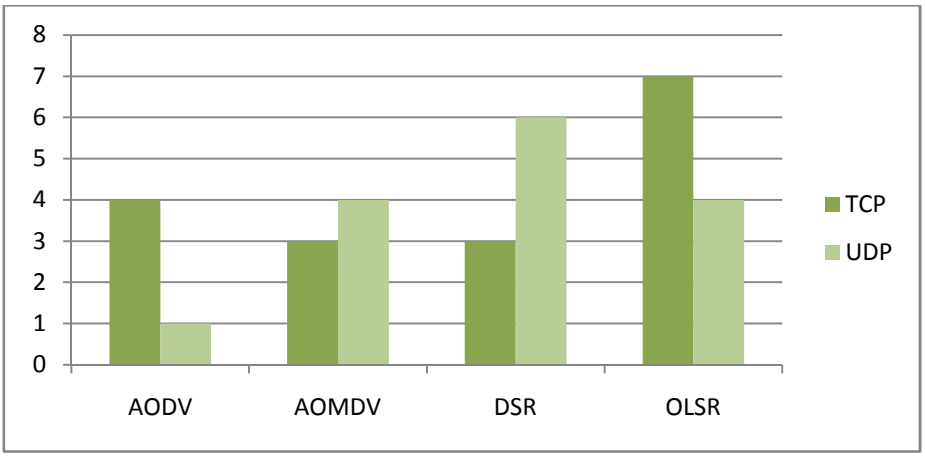


Figura IV.4: Instalación y Configuración

Interpretación:

En la Figura IV.4 se puede observar que al tratarse de tráfico TCP los protocolos OLSR y AODV tienen mayor capacidad para establecer los enlaces y de esta manera logran enviar más

información en el mismo período de tiempo, mientras que al tratarse de tráfico UDP, el protocolo dominante es DSR por su relativa simplicidad al establecer enlaces.

INDICADOR 5: CAPACIDAD DE RECEPCIÓN

Se refiere al número de paquetes recibidos correctamente por cada nodo durante el transcurso de la simulación del escenario.

	TCP(Paquetes)	UDP(Paquetes)
0	<70 mil	<6000
1	Entre 70 y 72mil	Entre 6000 y 8000
2	Entre 72 y 74mil	Entre 8000 y 10000
3	Entre 74 y 76mil	Entre 10000 y 12000
4	Entre 76 y 78mil	Entre 12000 y 14000
5 en adelante	>78000	>14000

Tabla IV.XV: Equivalencias4

	AODV	AOMDV	DSR	OLSR
TCP	3	2	2	7
UDP	4	3	6	1

Tabla IV.XVI: Capacidad de recepción.

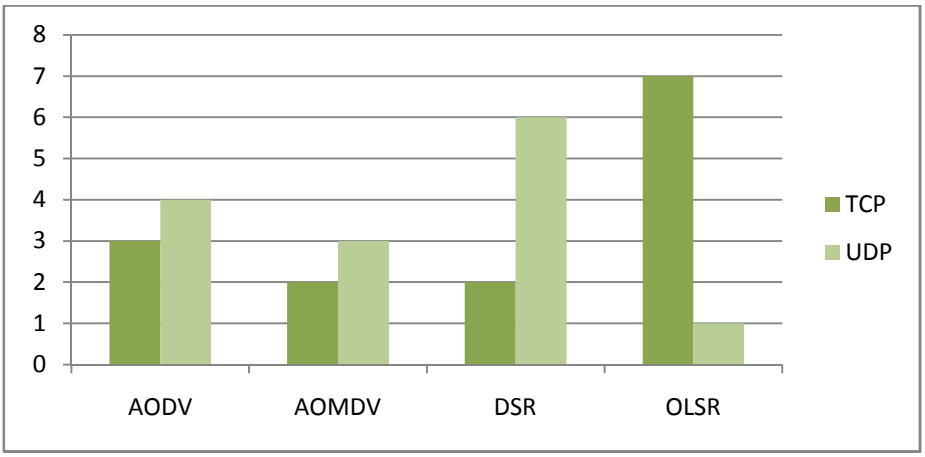


Figura IV.5: Capacidad de recepción.

Interpretación:

En la Figura IV.5 se observa que al tratarse de tráfico TCP el protocolo OLSR es el que posee un mejor desempeño mientras que los demás poseen una tasa de recepción menor pero equivalente entre ellos. Al tratarse de tráfico UDP se observa que DSR por su consistencia se desempeña de mejor manera contra un bajo desempeño de OLSR.

INDICADOR 6: PAQUETES PERDIDOS

Número de paquetes que se perdieron en la red.

	TCP(Paquetes)	UDP(Paquetes)
0	>500	>16000
1	Entre 450 y 500	Entre 12000 y 16000
2	Entre 400 y 450	Entre 8000 y 12000

3	Entre 350 y 400	Entre 4000 y 8000
4	Entre 300 y 350	Entre 1000 y 4000
5 en adelante	<300	<1000

Tabla IV.XVII: Equivalencias5

	AODV	AOMDV	DSR	OLSR
TCP	0	0	3	4
UDP	4	3	5	0

Tabla IV.XVIII: Paquetes perdidos

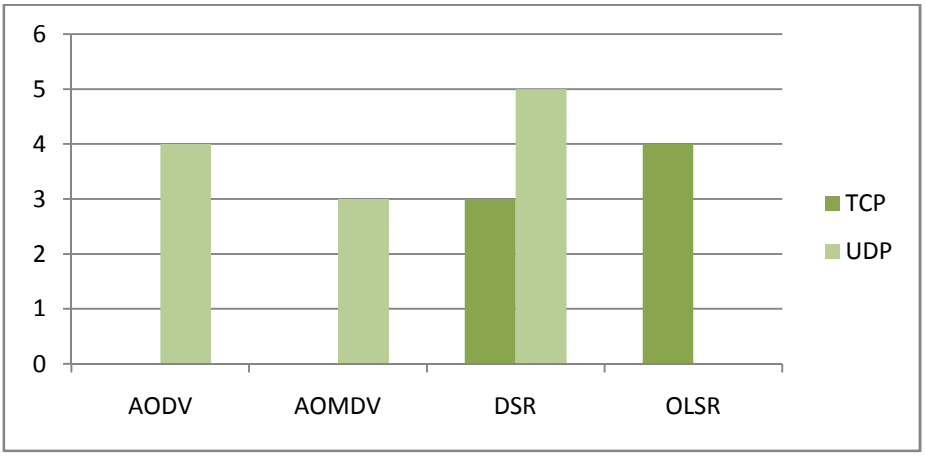


Figura IV.6: Tasa de paquetes perdidos

Interpretación:

Podemos observar que al tratarse de tráfico TCP los protocolos DSR y OLSR pierden una menor cantidad de paquetes teniendo un mejor desempeño. Cuando los protocolos deben manejar tráfico UDP los protocolos multi camino como OLSR y AOMDV tienen un bajo desempeño y tenemos un mejor rendimiento por parte de AODV y DSR.

4.3.2.2. V3: CARGA DEL PROTOCOLO

INDICADOR 7: CANTIDAD DE PAQUETES GENERADOS

Se refiere a los paquetes que genera el protocolo para su funcionamiento, de esta manera establece las rutas a utilizarse en la red. El tráfico que genere el protocolo puede afectar el funcionamiento de la red.

	TCP(Paquetes)	UDP(Paquetes)
0	>22 mil	>110 mil
1	Entre 19 y 22mil	Entre 100 y 110mil
2	Entre 16 y 19mil	Entre 90 y 100mil
3	Entre 13 y 16mil	Entre 80 y 90mil
4	Entre 10 y 13mil	Entre 70 y 80mil
5 en adelante	<10 mil	<70mil

Tabla IV.XIX: Equivalencias6

	AODV	AOMDV	DSR	OLSR
TCP	3	0	5	1
UDP	0	0	6	3

Tabla IV.XX: Cantidad de paquetes generados

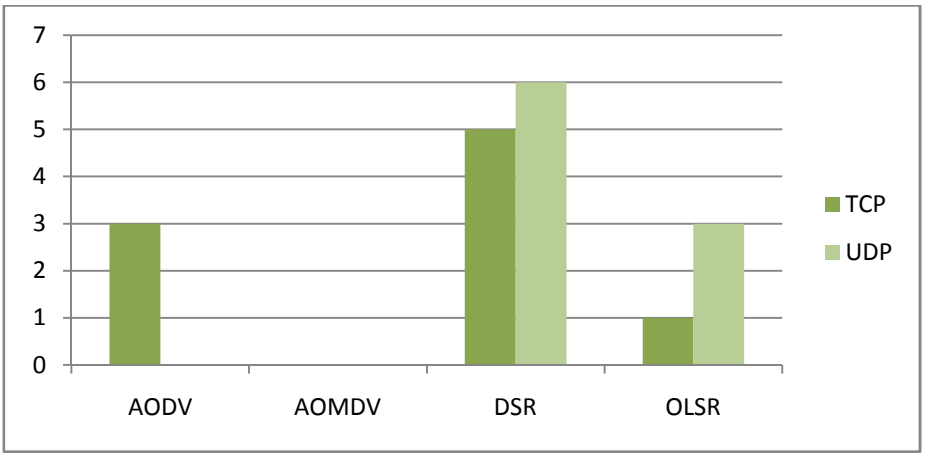


Figura IV.7: Cantidad de paquetes generados

Interpretación:

En el gráfico podemos observar que el protocolo que menos tráfico de protocolo envía es DSR, siendo de esta manera el mejor, mientas que el protocolo AOMDV posee la menor calificación debido a la cantidad de tráfico que genera en ambos casos.

INDICADOR 8: CANTIDAD DE PAQUETES RECIBIDOS

Los paquetes recibidos por protocolo dependen del funcionamiento del mismo, ya que en varios casos los paquetes a nivel de protocolo se envía en forma de broadcast y en otros casos solo a los nodos vecinos.

	TCP(Paquetes)	UDP(Paquetes)
0	>100 mil	>600 mil
1	Entre 80 y 100mil	Entre 500 y 600mil
2	Entre 60 y 80mil	Entre 400 y 500mil
3	Entre 40 y 60mil	Entre 300 y 400mil
4	Entre 20 y 40mil	Entre 200 y 300mil
5 en adelante	<20 mil	<200mil

Tabla IV.XXI: Equivalencias7

	AODV	AOMDV	DSR	OLSR
TCP	2	1	4	1
UDP	2	0	5	3

Tabla IV.XXII: Cantidad de paquetes recibidos

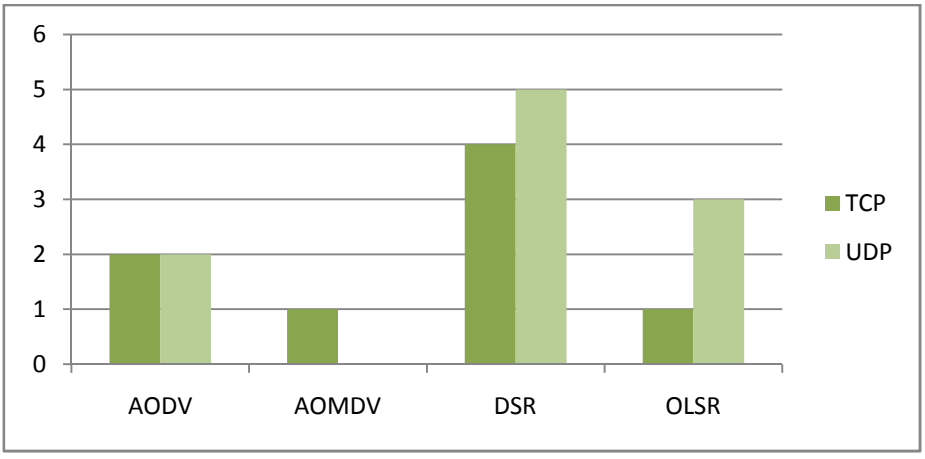


Figura IV.8: Cantidad de paquetes recibidos

Interpretación:

Los paquetes recibidos son consecuencia directa de la cantidad de paquetes generados por el protocolo de enrutamiento. Por ello, se refleja los resultados de igual manera, es decir DSR maneja mejor el tráfico generado para su funcionamiento mientras que AOMDV tiene el menor desempeño en este campo.

4.3.2.3. V4: CONSUMO DE ENERGÍA

INDICADOR 9: ENERGÍA CONSUMIDA EN TRANSMISIÓN

La energía contabilizada se obtiene incrementando el consumo total en base a los paquetes enviados, puede observarse un consumo mayor en envió debido a la potencia necesaria para enviar la señal en una cobertura deseada. Esta energía se contabiliza en vatios por segundo (W.s) y la potencia en Watts.

	TCP(W.s)	UDP(W.s)
0	>800	>500
1	Entre 750 y 800	Entre 450 y 500
2	Entre 700 y 750	Entre 400 y 450
3	Entre 650 y 700	Entre 350 y 400
4	Entre 600 y 650	Entre 300 y 350

5 en adelante	<600	<300

Tabla IV.XXIII: Equivalencias8

	AODV	AOMDV	DSR	OLSR
TCP	2	3	1	3
UDP	3	1	0	5

Tabla IV.XXIV: Consumo de energía en envío

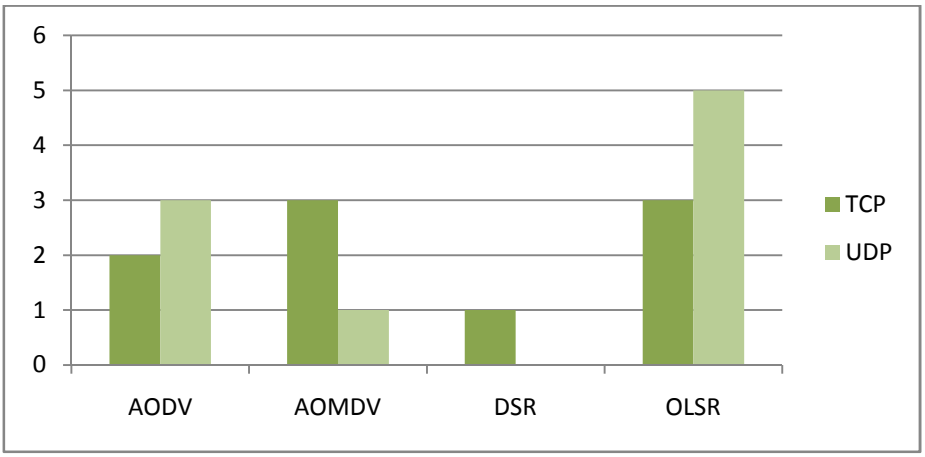


Figura IV.9: Consumo de energía en envío

Interpretación:

Como se puede observar, el protocolo OLSR es el más eficiente en cuanto a consumo, esto se debe a su mejor elección de rutas, por lo que se realizan menos envíos de paquetes lo que reduce el consumo de energía. El protocolo DSR es el que tiene mayor consumo de energía, a

pesar de su eficiencia en el manejo de paquetes, lo realiza de una manera más general, utilizando rutas menos eficientes lo que produce un mayor de consumo de energía en la red.

INDICADOR 10: ENERGÍA CONSUMIDA EN RECEPCIÓN

La energía consumida en recepción se contabiliza en base a los paquetes que alcanzaron su destino, esta energía se contabiliza en W.s.

	TCP(W.s)	UDP(W.s)
0	>270	>160
1	Entre 260 y 270	Entre 140 y 160
2	Entre 250 y 260	Entre 120 y 140
3	Entre 240 y 250	Entre 100 y 120
4	Entre 230 y 240	Entre 80 y 100
5 en adelante	<230	<80

Tabla IV.XXV: Equivalencias9

	AODV	AOMDV	DSR	OLSR

TCP	2	4	1	4
UDP	2	1	0	4

Tabla IV.XXVI: Energía consumida en recepción

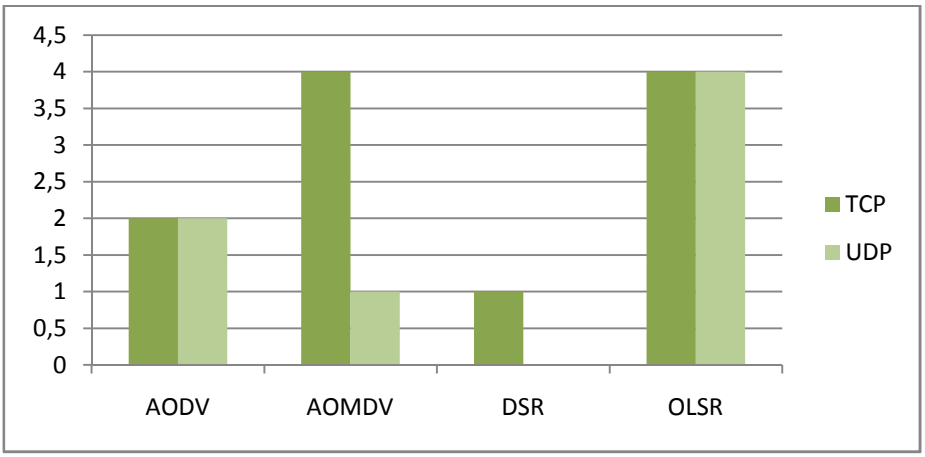


Figura IV.10: Energía consumida en recepción

Interpretación:

La energía consumida en recepción responde a la generación de paquetes por parte de los nodos, por ello, el resultado de la energía consumida en envío es reflejo de la cantidad de paquetes generados, de esta manera, el protocolo más eficiente en consumo sigue siendo OLSR y el protocolo con mayor consumo de energía es DSR, mientras que AODV y AOMDV mantienen una relación cercana, con un consumo mayor por parte de AOMDV en el caso de tráfico FTP.

INDICADOR 11: ENERGÍA CONSUMIDA EN ESPERA

La energía que se consume espera se incrementa cuando un nodo no se encuentra transmitiendo o receptado paquetes, tiene un pequeño consumo mientras escucha la red y envía paquetes de enrutamiento.

	TCP(W.s)	UDP(W.s)
0	>0,07	>0,38
1	Entre 0,06 y 0,07	Entre 0,36 y 0,38
2	Entre 0,05 y 0,06	Entre 0,34 y 0,36
3	Entre 0,04 y 0,05	Entre 0,32 y 0,34
4	Entre 0,03 y 0,04	Entre 0,3 y 0,32
5 en adelante	<0,03	<0,3

Tabla IV.XXVII: Equivalencias10

	AODV	AOMDV	DSR	OLSR
TCP	2	2	2	2
UDP	1	1	1	1

Tabla IV.XXVIII: Energía consumida en espera

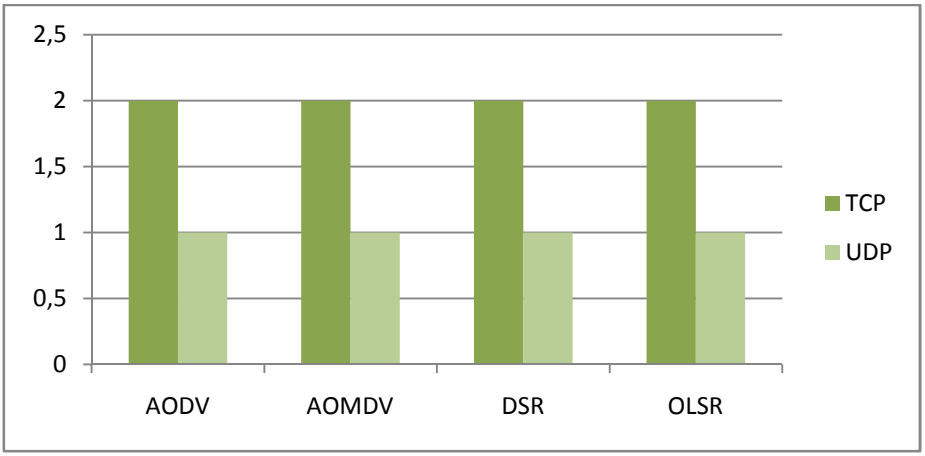


Figura IV.11: Energía consumida en espera

Interpretación:

Al ser un consumo tan bajo el del estado de espera, se considera que el consumo en estado para todos los protocolos es equivalente, por lo que se puede ver que el consumo de energía dependerá en su gran mayoría de la generación de paquetes.

4.4. PUNTAJES TOTALES

A continuación tenemos los resultados generales de las variables dependientes de los valores obtenidos para el estudio comparativo de los protocolo de enrutamiento para redes móviles AODV, AOMDV, DSR y OLSR

Variables	Indicadores	AODV	AOMDV	DSR	OLSR
Fiabilidad	**I4**	5	7	9	11
	I5	7	5	8	8
	I6	4	3	8	4
Total V2		**16**	**15**	**25**	**23**
Carga del protocolo	**I7**	3	0	11	4
	I8	4	1	9	4
Total V3		**7**	**1**	**20**	**8**
Consumo de energía	**I9**	5	4	1	8
	I10	4	5	1	8
	I11	3	3	3	3
Total V4		**12**	**12**	**5**	**19**
Total		***35***	***28***	***50***	***50***

Tabla IV.XXIX: Tabla General de Resultados

Podemos observar en los resultados que los protocolos OLSR y DSR tienen un puntaje igual de 50, mientras que el protocolo con menor calificación es AOMDV.

4.5. RESULTADOS DEL ESTUDIO COMPARATIVO

Se realizó el estudio comparativo entre los protocolos de ruteo para redes inalámbricas móviles AODV, AOMDV, DSR y OLSR obteniendo los siguientes puntos relevantes:

- En cuanto a fiabilidad de los protocolos se puede observar que DSR posee una gran capacidad de respuesta en cuanto al manejo de paquetes, también podemos observar que el protocolo con el menor es AOMDV que no tuvo una capacidad inferior en cuanto al manejo de paquetes. En cuanto a fiabilidad OLSR y AODV tienen un manejo similar de paquetes en cuanto a su fiabilidad.
- Respecto a la carga de los protocolos en la red podemos observar que el protocolo DSR es el más ligero, tiene un funcionamiento más sencillo. Podemos observar que los protocolos AOMDV y DSR son los que más carga aportan a la red, eso se debe a su complejidad y a su característica de generar rutas válidas.
- En cuanto al consumo de energía en la red podemos observar que OLSR es el que mejor características presenta, esto se debe en gran parte al número de paquetes que genera en la red, gracias a su mejor elección de rutas. El protocolo DSR se muestra como el protocolo con el mayor consumo en la red, esto como resultado de su funcionamiento en general. Finalmente los protocolos AODV y AOMDV tienen un consumo similar de energía

.

4.6. COMPROBACIÓN DE LA HIPÓTESIS

Para la comprobación de la hipótesis planteada se debe calcular el estadístico Chi Cuadrado utilizando los datos que se obtuvo en el estudio comparativo de los protocolos de enrutamiento para redes inalámbricas AdHoc que se obtuvieron mediante análisis cuantitativos y cualitativos, mediante una simulación en NS2 y análisis de información teórica.

Hipótesis para chi cuadrado:

Hi: hipótesis de la investigación

Ho: hipótesis nula

Hi: A través del estudio comparativo de protocolos de ruteo para redes inalámbricas móviles AdHoc se podrá comparar las capacidades de cada protocolo para encontrar el que mejor se adapta en un escenario posible.

Ho: A través del estudio comparativo de protocolos de ruteo para redes inalámbricas móviles AdHoc no se podrá comparar las capacidades de cada protocolo para encontrar el que mejor se adapta en un escenario posible.

En el siguiente cuadro se puede observar los valores de las variables estimadas en el estudio de los protocolos AODV, AOMDV, DSR, OLSR.

Variable independiente: Análisis de los protocolos de enrutamiento de redes móviles AdHoc

Variables independientes: Fiabilidad, Carga del protocolo, Consumo de energía

Análisis de los protocolos de	**Indicadores**	**AODV**	**AOMDV**	**DSR**	

enrutamiento de redes móviles AdHoc					**OLSR**
Fiabilidad	**I4**	5	7	9	11
	I5	7	5	8	8
	I6	4	3	8	4
Carga del protocolo	**I7**	3	0	11	4
	I8	4	1	9	4
Consumo de energía	**I9**	5	4	1	8
	I10	4	5	1	8
	I11	3	3	3	3
Total		***35***	***28***	***50***	***50***

Tabla IV.XXX: Resultados obtenidos para variables dependientes

Frecuencias Observadas:

Las frecuencias observadas se encuentran sumando los indicadores de cada variable dependiente.

	AODV	AOMDV	DSR	OLSR	Sumatoria de cada variable
Fiabilidad	**16**	**15**	**25**	**23**	79
Carga del P	**7**	**1**	**20**	**8**	36
Consumo de E	**12**	**12**	**5**	**19**	48
Total	35	28	50	50	163

Tabla IV.XXXI: Frecuencias observadas

Frecuencias Esperadas:

Las frecuencias esperadas de cada celda, se calcula mediante la siguiente

$$fe = \frac{(total\_de\_fila)(total\_de\_columna)}{N}$$

Donde *N* es el número total de frecuencias observadas

	AODV	AOMDV	DSR	OLSR	Sumatoria de cada variable
Fiabilidad	16,9631902	13,5705521	24,2331288	24,2331288	79
Carga del P	7,73006135	6,18404908	11,0429448	11,0429448	36
Consumo de E	10,3067485	8,24539877	14,7239264	14,7239264	48
Total	35	28	50	50	163

Tabla IV.XXXII: Frecuencias esperadas

Calculo de X^2:

Para encontrar el valor de chi cuadrado se utiliza la siguiente fórmula

$$\mathrm{X}^2 = \sum \frac{(\mathrm{O} - \mathrm{E})^2}{\mathrm{E}}$$

O: Frecuencia observada en cada celda

E: Frecuencia esperada en cada celda

Observado(O)	Esperado(E)	(O-E)	(O-E)^2	{(O-E)^2/E}
17	16,9631902	-0,96319018	0,92773533	0,05469109
15	13,5705521	1,42944785	2,04332116	0,15057023
18	24,2331288	0,76687117	0,58809138	0,02426807
17	24,2331288	-1,23312883	1,52060672	0,06274909
10	7,73006135	-0,73006135	0,53298957	0,06895024
4	6,18404908	-5,18404908	26,8743649	4,34575543
15	11,0429448	8,95705521	80,2288381	7,26516701
5	11,0429448	-3,04294479	9,25951297	0,83850034
14	10,3067485	1,69325153	2,86710076	0,27817704
16	8,24539877	3,75460123	14,0970304	1,70968449
9	14,7239264	-9,72392638	94,5547443	6,42184305
22	14,7239264	4,27607362	18,2848056	1,24184305
				X^2 =22,462

Tabla IV.XXIII: Calculo de Chi Cuadrado

Grados de libertad:

Para poder realizar la comparación del resultado de chi cuadrado debemos encontrar los grados de libertad de nuestra tabla, para ello tenemos la siguiente fórmula:

$$Gl = (f - 1)(c - 1)$$

Donde:

r: es el número de filas de la tabla de contingencia

c: es el número de columnas de la tabla de contingencia

Gl = (3-1) (4-1) → Gl = 6

De la tabla de distribución de X^2 que se encuentra en los anexos, podemos decir que con un 95% de seguridad y con 6 grados de libertad el valor de crítico de chi cuadrado es 12,59.

Criterio de decisión

- SI X^2 calculado es mayor a X_{α}^2 (Valor crítico) de la tabla de distribución se rechaza la hipótesis nula Ho y por lo tanto se acepta la hipótesis de Investigación.

- SI X^2 calculado es menor a X_{α}^2 (Valor crítico) de la tabla de distribución se acepta la hipótesis nula Ho y por lo tanto se rechaza la hipótesis de Investigación.

Por lo tanto tenemos el siguiente resultado:

X^2 Calculado = 22,4621991

X_{α}^2 (Valor crítico) = 12,59

22,46 > 12,59

Por lo tanto: $X^2 > X_{\alpha}^2$

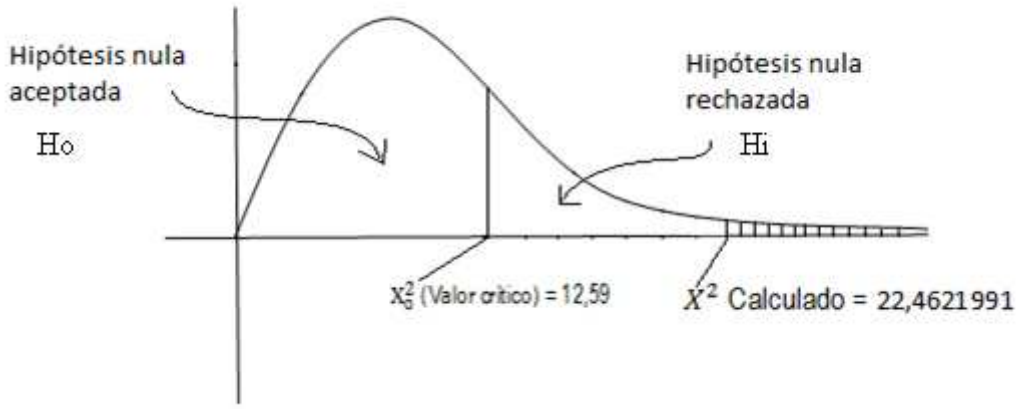


Figura IV.12: Gráfico chi cuadrado

De esta manera aceptamos la hipótesis de la investigación (Hi) que indica que a través del estudio comparativo de protocolos de ruteo para redes inalámbricas móviles AdHoc se podrá comparar las capacidades de cada protocolo para encontrar el que mejor se adapta en un escenario posible.

CONCLUSIONES

- Las redes inalámbricas móviles son una tecnología que día a día demuestran su utilidad en diferentes ámbitos para dar solución en diferentes aspectos a los requerimientos de la comunicación IP.

- Una red descentralizada es una solución real en aspectos de confiabilidad en caso de no poder confiar el funcionamiento de la red a un punto central susceptible a fallos.

- La auto adaptabilidad de una red es un concepto muy útil al momento de necesitar comunicación IP inalámbrica en un ambiente que no prevé las facilidades de una topología regular en la que se cuenta con la estabilidad de los enlaces o ambientes controlados en lo que puede preverse el comportamiento de los integrantes de la red.

- La investigación para el desarrollo de aplicaciones y tecnologías es fundamental, de esta manera se logra explorar los diferentes parámetros permitiendo gracias al estudio profundo del conocimiento actual crear nuevas ideas a ser implementadas o adaptaciones para mejorar lo existente.

- La simulación como herramienta de investigación y desarrollo permite explorar una realidad dentro de parámetros determinados llegando explotar las capacidades de una

tecnología, de esta manera, se consigue determinar alcances, funcionamiento, capacidades y posibles soluciones a nuevos retos.

- La movilidad en una red crea una nueva ramificación en el concepto de enrutamiento de redes, de esta manera se exploran puntos que no se consideraron al crear redes regulares, por ello, en base a los estándares ya desarrollados para redes cableadas se busca dar solución a los infinitos retos que pueden presentarse gracias a la movilidad.

- Los protocolos diseñados para dar soporte a las redes móviles inalámbricas brindan una solución para los requerimientos de estas, por ello, constantemente se crean nuevos protocolos que brinden una mejor solución a casos específicos, de esta manera podemos encontrar gran cantidad de protocolos generados a partir de los más generalizados.

- El estudio comparativo permitió determinar que el protocolo OLSR y DSR son los más adecuados al momento de trabajar con una red móvil, OLSR gracias a su desarrollo y complejidad permite un análisis más preciso de la red, mientras que DSR por su simplicidad obtiene buenos resultados al trabajar en redes extensas y propensas a cambios.

RECOMENDACIONES

- Al momento de simular un escenario no se puede alcanzar una perfección ya que por definición es una simulación de la realidad, por ello se recomienda estudiar los parámetros que se consideran determinantes para los cambios en la red, de esta manera se consigue que la simulación sea más precisa respecto a la realidad

- Se recomienda que para tener una vista más amplia respecto a los sucesos dados en una simulación se analicen situaciones reales de funcionamiento así como un análisis que permita explotar la red en sus máxima capacidad

- Para estudiar distintas tecnologías debe de antemano conocerse su funcionamiento así como la razón de su existencia, ya que cada solución viene dada gracias a la existencia de una necesidad, por ello se recomienda entender el fin que busca cada a solución de manera que los análisis sean objetivos

- Para este trabajo se recomienda explotar desde diferentes ángulos los protocolos analizados, de esta manera se consigue entender sus funciones y de qué manera podemos utilizarlos al momento de realizar una implementación.

BIBLIOGRAFÍA

LIBROS

1.- OZAN, K. Tonguz and GIANLUIGI, Ferrari. Ad Hoc Wireless Networks: a communication theoretic perspective. New York: John Wiley and Sons, 2006. 330 p.

2. - CHAI, K. Toh. Ad Hoc Mobile Wireless Networks: protocols and systems. Cambridge: Prentice Hall, 2002. 336 p.

BIBLIOGRAFÍA INTERNET

[1] Redes AdHoc

http://es.wikipedia.org/wiki/Ad_hoc

2010-02-10

[2] Estándares en redes inalámbricas

http://standards.ieee.org/inalámbricas/

2010-03-27

[3] Wireless LAN

http://www.eveliux.com/mx/estandares-wlan.php

2010-02-20

[4] Redes de área local

http://www.hiperlan2. */files/asb.html*

2010-02-22

[5] Estándares en redes

http://standards.ieee.org/getieee802/802.11.html

2010-02-12

[6] Redes móviles

http://www.hiperlan2.com/

2010-02-11

[7] Movilidad en redes IP

eav.upb.edu.co/banco/files/03INTRODUCCION

2010-02-02

[8] Espectro radioeléctrico

http://www.homerf.org/

2010-02-07

[9] Redes vehiculares

http://es.wikipedia.org/wiki/VANET

2010-02-10

[10] Protocolos en redes móviles

http://www.arqhys.com/construccion/protocolos-introduccion.html

2010-02-02

[11] Enrutamientos en redes móviles

www.redhucyt.oas.org/webesp/PRESENTATIONS/.../Routing.pp

2010-03-09

[12] Simulación de redes

http://bibing.us.es/proyectos/abreproy/11306/fichero/TEORIA%252F08+-+Capitulo+3.pdf

2010-01-14

[13] Encaminamiento en redes

es.wikipedia.org/wiki/Encaminamiento

2010-04-02

[14] Protocolo AODV

moment.cs.ucsb.edu/AODV/aodv.html

2010-04-18

[15] Protocolo DSR

en.wikipedia.org/wiki/Dynamic_Source_Routing

2010-05-10

[17] Protocolo OLSR

www.olsr.org/

2010-01-13

[18] Protocolo OLSR

http://es.wikipedia.org/wiki/Optimized_Link_State_Routing

2010-05-20

[19] Redes móviles

http://e-archivo.uc3m.es/bitstream/10016/2462/1/_JJ_Vinagre_Diaz

2010-05-20

[20] Consumo de energía en redes inalámbricas

http://www.eveliux.com/mx/conservacion-de-energia-en-medios-inalambricos.php

2010-05-20

[21] Modelos de energía

http://eav.upb.edu.co/banco/?q=node/472

2010-02-14

[22] La energía en redes móviles

http://bibing.us.es/proyectos/abreproy/11306/fichero/TEORIA%252F10+-+Capitulo+5

2010-02-14

[23] Simulador NS2

es.wikipedia.org/wiki/Ns_(simulador)

2010-03-11

[24] Simulador NS2

http://www.isi.edu/nsnam/ns/

2010-03-22

[25] Simulador NS2

http://mural.uv.es/jovapin/pg1

2010-03-21

[26] Lenguaje TCL

http://es.wikipedia.org/wiki/Tcl

2010-02-10

[27] Lenguaje TK

es.wikipedia.org/wiki/Tk

2010-02-10

[28] NS2 en simulaciones móviles

http://www3.euitt.upm.es/taee/Congresosv2/2006/papers/2006S1G01

2010-02-10

[29] Animador de redes

www.isi.edu/nsnam/nam/

2010-03-11

[30] Xgraph

www.xgraph.org/

2010-04-02

[31] Xgraph

en.wikipedia.org/wiki/Xgraph

2010-01-12

RESUMEN

Se realizó un estudio comparativo entre protocolos de enrutamiento en redes móviles AdHoc para mejorar el funcionamiento de las mismas.

El estudio fue programado en NS2 sobre Linux CentOs creando enlaces entre nodos que trabajan en una red WiFi y envían información mediante saltos analizando el consumo de energía. Los protocolos fueron evaluados en dos escenarios: El primero FTP con treinta nodos y el segundo con tráfico exhaustivo CBR y cincuenta nodos.

De esta manera se encontró que OLSR es capaz en el primer escenario de enviar y recibir 10% mas paquetes que AODV que le precede, y 13% más que DSR que se encuentra al final, a demás consume aproximadamente 8% menos de energía que los demás protocolos, siendo de esta manera el más eficiente y confiable de ellos. Por otra parte en el segundo escenario que explota las capacidades de los protocolos tenemos que DSR se desempeña de mejor manera con un 16% más en recepción de paquetes en relación a AOMDV, y un 59% más en relación a OLSR aunque este último continúa siendo el de menor consumo de energía. También podemos comprobar que AOMDV tiene un desempeño ligeramente superior a AODV.

Finalmente se consiguió mediante la comparación entre los protocolos encontrar el más eficiente en el manejo de la red y consumo de energía que es OLSR.

Se recomienda estudiar las características de cada escenario antes de optar por la utilización de un protocolo.

SUMMARY

A comparative study between routing protocols in ad hoc mobile networks was carried out to improve their functioning. The study was programmed in NS2 on Linux CentOs creating links between nodes working in a network WIFI and sending information through gaps analyzing the power consumption. The protocols were evaluated in two sceneries: The first with thirty nodes and the second one with exhaustive traffic CBR and fifty nodes. Thus it was found out that OLSR is capable of, in the first scenery, sending and receiving 10% more packages tan the AODV which precedes it and 13% more than the DSR which is at the end; moreover it consumes approximately 8% less energy than the other protocols, this way being the most efficient and reliable. On the other hand, in the second scenery which exploits the capabilities of the protocols, DSR performs in a better way with a 16% more in the package reception as related to the AOMDV and 59% more as related to an OLSR although this last one keeps on being the one with the least energy consumption, it is also possible to test that AOMDV has a slightly higher performance tan the AODV. Finally it was possible to find, through comparison between protocols, the more efficient one in handling the network and energy consumption tan the OLSR. It is recommended to study the characteristics of each scenery before choosing the use a protocol.

ANEXOS

Código de simulación

```
set ini [clock seconds]


set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            Queue/DropTail/PriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(ifqlen)         50
set val(nn)             50
set val(rp)             AODV
set val(x)       1500
set val(y)       1500
set val(sc)
"/root/Desktop/ejems/simu13/simuAODV/movs1.tcl"
set val(cp)
"/root/Desktop/ejems/simu13/simuAODV/traf3.tcl"
set val(fin)            600.0
set val(icbr)          0.005
set val(MNcoverage)    200.0

Mac/802_11 set syncFlag_ 1

Mac/802_11 set dutyCycle_ 10


Mac/802_11 set dataRate_          11Mb
Mac/802_11 set basicRate_          1Mb
Mac/802_11 set PLCPDataRate_       1Mb
Mac/802_11 set RTSThreshold_         0
Mac/802_11 set PreambleLength_      72
Mac/802_11 set PLCPHeaderLength_    24
Agent/UDP set  packetSize_        1500

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

set ns_          [new Simulator]
set tracefd     [open ayu1.tr w]
set f1 [open out1.tr w]
```

```
set f2 [open out2.tr w]
set f3 [open out3.tr w]
set f4 [open out4.tr w]
set ftemp [open ftemp w]
$ns_ trace-all $tracefd

set namtrace [open ayu1.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
$ns_ use-newtrace


set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

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
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

set pottrans [SetPt $val(MNcoverage)]
puts "potencia de transmicion para cubrir $val(MNcoverage) metros
es: $pottrans"
set god_ [create-god $val(nn)]


set chan_1_ [new $val(chan)]


$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) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace ON \
                  -energyModel "EnergyModel" \
                  -idlePower 0.000014 \
                  -rxPower 0.053 \
                  -txPower $pottrans \
                  -sleepPower 0.1 \
```

```
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy 400 \
            -channel $chan_1_

$ns_ set WirelessNewTrace_ ON

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

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

source $val(sc)
for {set i 0} {$i < $val(nn)} {incr i} {
     $ns_ initial_node_pos $node_($i) 15
}


source $val(cp)
puts "cargando trafico"


for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(fin).0 "$node_($i) reset";
}


$ns_ at $val(fin).01 "stop"
$ns_ at $val(fin).02 "puts \"NS EXITING...\" ; $ns_ halt"


proc stop {} {
    global  ftemp ns_ tracefd
    $ns_ flush-trace
    $ns_ nam-end-wireless [$ns_ now]
    close $tracefd

}

puts "Starting Simulation..."
$ns_ run
set fini [clock seconds]
set tiempo [expr $fini - $ini]
puts "tiempo: $tiempo segundos"
```

CBRGEN

```
#
#  Copyright (c) 1999 by the University of Southern California
#  All rights reserved.
```

```
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License,
#  version 2, as published by the Free Software Foundation.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public
License along
#  with this program; if not, write to the Free Software
Foundation, Inc.,
#  59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.
#
#  The copyright of this module includes the following
#  linking-with-specific-other-licenses addition:
#
#  In addition, as a special exception, the copyright holders of
#  this module give you permission to combine (via static or
#  dynamic linking) this module with free software programs or
#  libraries that are released under the GNU LGPL and with code
#  included in the standard release of ns-2 under the Apache 2.0
#  license or under otherwise-compatible licenses with advertising
#  requirements (or modified versions of such code, with unchanged
#  license).  You may copy and distribute such a system following
the
#  terms of the GNU GPL for this module and the licenses of the
#  other code concerned, provided that you include the source code
of
#  that other code when and as the GNU GPL requires distribution of
#  source code.
#
#  Note that people who make modified versions of this module
#  are not obligated to grant this special exception for their
#  modified versions; it is their choice whether to do so.  The GNU
#  General Public License gives permission to release a modified
#  version without this exception; this exception also makes it
#  possible to release a modified version which carries forward
this
#  exception.

# Traffic source generator from CMU's mobile code.
#
# $Header: /cvsroot/nsnam/ns-2/indep-utils/cmu-scen-
gen/cbrgen.tcl,v 1.4 2005/09/16 03:05:39 tomh Exp $

#
=================================================================
===
# Default Script Options
#
=================================================================
===
set opt(nn)      0          ;# Number of Nodes
```

```
set opt(seed)         0.0
set opt(mc)      0
set opt(pktsize) 512

set opt(rate)         0
set opt(interval)     0.0        ;# inverse of rate
set opt(type)           ""

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

proc usage {} {
    global argv0

    puts "\nusage: $argv0 \[-type cbr|tcp\] \[-nn nodes\] \[-seed
seed\] \[-mc connections\] \[-rate rate\]\n"
}

proc getopt {argc argv} {
      global opt
      lappend optlist nn seed mc rate type

      for {set i 0} {$i < $argc} {incr i} {
            set arg [lindex $argv $i]
            if {[string range $arg 0 0] != "-"} continue

            set name [string range $arg 1 end]
            set opt($name) [lindex $argv [expr $i+1]]
      }
}

proc create-cbr-connection { src dst } {
      global rng cbr_cnt opt

      set stime [$rng uniform 0.0 180.0]
      set stime2 [expr $stime + 20]
      puts "#\n# $src connecting to $dst at time $stime\n#"

      ##puts "set cbr_($cbr_cnt) \[\$ns_ create-connection \
            ##CBR \$node_($src) CBR \$node_($dst) 0\]";
      puts "set udp_($cbr_cnt) \[new Agent/UDP\]"
      puts "\$ns_ attach-agent \$node_($src) \$udp_($cbr_cnt)"
      puts "set sink$cbr_cnt \[new Agent/LossMonitor\]"
      puts "\$ns_ attach-agent \$node_($dst) \$sink$cbr_cnt"
      puts "set cbr_($cbr_cnt) \[new Application/Traffic/CBR\]"
      puts "\$cbr_($cbr_cnt) set packetSize_ $opt(pktsize)"
      puts "\$cbr_($cbr_cnt) set interval_ $opt(interval)"
      puts "\$cbr_($cbr_cnt) set random_ 1"
      puts "\$cbr_($cbr_cnt) set maxpkts_ 10000"
      puts "\$cbr_($cbr_cnt) attach-agent \$udp_($cbr_cnt)"
      puts "\$ns_ connect \$udp_($cbr_cnt) \$sink$cbr_cnt"

      puts "\$ns_ at $stime \"\$cbr_($cbr_cnt) start\""
```

```
      #puts "\$ns_ at $stime \"\$node_($src) add-mark $src red
circle\""
        #puts "\$ns_ at $stime \"\$node_($dst) add-mark $dst blue
circle\""
      #puts "\$ns_ at $stime2 \"\$node_($src) delete-mark $src \""
        #puts "\$ns_ at $stime2 \"\$node_($dst) delete-mark $dst
\""

      incr cbr_cnt
}

proc create-tcp-connection { src dst } {
      global rng cbr_cnt opt

      set stime [$rng uniform 0.0 180.0]
      set stime2 [expr $stime + 20]
      puts "#\n# $src connecting to $dst at time $stime\n#"

      puts "set tcp_($cbr_cnt) \[\$ns_ create-connection \
            TCP \$node_($src) TCPSink \$node_($dst) 0\]";
      puts "\$tcp_($cbr_cnt) set window_ 32"
      puts "\$tcp_($cbr_cnt) set packetSize_ $opt(pktsize)"

      puts "set ftp_($cbr_cnt) \[\$tcp_($cbr_cnt) attach-source
FTP\]"


      puts "\$ns_ at $stime \"\$ftp_($cbr_cnt) start\""

      #puts "\$ns_ at $stime \"\$node_($src) add-mark $src red
circle\""
        #puts "\$ns_ at $stime \"\$node_($dst) add-mark $dst blue
circle\""
        #puts "\$ns_ at $stime2 \"\$node_($src) delete-mark $src
\""
        #puts "\$ns_ at $stime2 \"\$node_($dst) delete-mark $dst
\""

      incr cbr_cnt
}

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

getopt $argc $argv

if { $opt(type) == "" } {
    usage
    exit
} elseif { $opt(type) == "cbr" } {
    if { $opt(nn) == 0 || $opt(seed) == 0.0 || $opt(mc) == 0 ||
$opt(rate) == 0 } {
      usage
      exit
    }
```

```
    set opt(interval) [expr 1 / $opt(rate)]
    if { $opt(interval) <= 0.0 } {
      puts "\ninvalid sending rate $opt(rate)\n"
      exit
    }
}

puts "#\n# nodes: $opt(nn), max conn: $opt(mc), send rate:
$opt(interval), seed: $opt(seed)\n#"

set rng [new RNG]
$rng seed $opt(seed)

set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ 100
$u use-rng $rng

set cbr_cnt 0
set src_cnt 0

for {set i 0} {$i < $opt(nn) } {incr i} {

      set x [$u value]

      if {$x < 50} {continue;}

      incr src_cnt

      set dst [expr ($i+1) % [expr $opt(nn) + 1] ]
      #if { $dst == 0 } {
          #set dst [expr $dst + 1]
          #}

      if { $opt(type) == "cbr" } {
            create-cbr-connection $i $dst
      } else {
            create-tcp-connection $i $dst
      }

      if { $cbr_cnt == $opt(mc) } {
            break
      }

      if {$x < 75} {continue;}

      set dst [expr ($i+2) % [expr $opt(nn) + 1] ]
      #if { $dst == 0 } {
            #set dst [expr $dst + 1]
      #}

      if { $opt(type) == "cbr" } {
            create-cbr-connection $i $dst
      } else {
            create-tcp-connection $i $dst
```

```
        }

        if { $cbr_cnt == $opt(mc) } {
                break
        }
}

puts "#\n#Total sources/connections: $src_cnt/$cbr_cnt\n#"
```

FILTROS

```
BEGIN {
vez
salida2="filtro3.txt"
mayor_id=-1
bytes_env_0=0
rec_1=0
env_0=0
fow_1=0
bytes_rec_1=0
bytes_fow_1=0
var1=0
totalPenv=0
totalPrec=0
totalBenv=0
totalBrec=0
totalPfow=0
totalBfow=0
}
{
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$35
trace_level=$19
nodo_fuente=$5


if(accion!="SFESTs" && accion!="SFs" && trace_level=="AGT"){
if ( accion == "s"){
    if (id_paquete > mayor_id         mayor_id=id_paquete
        if ( prot_mes == "cbr"){ # Sólo interesan los paquetes CBR
                if ( nodo_actual==nodo){
                   bytes_env_0 = bytes_env_0 + bytes_paquete
                                       env_0++

                                }
           totalPenv++
           totalBenv = totalBenv + bytes_paquete

        }
    }
}
if ( accion == "r" && prot_mes == "cbr"){
    if (nodo_actual == nodo_destino){
        if (nodo_actual==nodo){
             bytes_rec_1 = bytes_rec_1 + bytes_paquete - 20

             rec_1++
        }
      totalPrec++
```

```
        totalBrec = totalBrec + bytes_paquete - 20


    }
}

}

if ( accion == "f" && prot_mes == "cbr" && trace_level == "RTR"){

        if (nodo_actual==nodo){
             bytes_fow_1 = bytes_fow_1 + bytes_paquete - 20
                          # la cabecera IP de 20 bytes
             fow_1++
        }
        totalPfow++
        totalBfow = totalBfow + bytes_paquete - 20



}


}
END{

if (ban == 0){
printf(":\n") >> salida2
printf("NODO%i\n",nodo) > salida2
printf("Paquetes recibidos por el nodo%i: %i \n", nodo, rec_1) >
salida2
printf("Paquetes enviados por el nodo%i: %i \n", nodo, env_0) >
salida2
printf("Paquetes reenviados por el nodo%i: %i \n", nodo, fow_1) >
salida2
#printf("Throughput1: %f \n", rec_1/env_0) > salida2
printf("Bytes recibidos por el nodo%i: %i \n", nodo, bytes_rec_1) >
salida2
printf("Bytes enviados por el nodo%i: %i \n", nodo, bytes_env_0) >
salida2
#printf("Throughput2:  %f \n", bytes_rec_1/bytes_env_0) > salida2
printf("Bytes reenviados por el nodo%i: %i \n", nodo, bytes_fow_1)
> salida2

}
else{
printf("\n\n\n") >> salida2
printf("TOTALES:\n\n") > salida2
printf("total paquetes enviados:%i \n",totalPenv) > salida2
printf("total bytes enviados:%i \n\n",totalBenv) > salida2
printf("total paquetes recibidos:%i \n",totalPrec) > salida2
printf("total bytes recibidos:%i \n\n",totalBrec) > salida2
printf("total paquetes reenviados:%i \n",totalPfow) > salida2
printf("total bytes reenviados:%i \n\n\n",totalBfow) > salida2

}
```

```
}



BEGIN {
salida2="filtro4.txt"
mayor_id=-1
caidos=0
reenviados=0
}
{
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$35
trace_level=$19
nodo_fuente=$5


if ( accion == "d"){
    if (id_paquete > mayor_id){
mayor_id=id_paquete
        if ( prot_mes == "cbr"){
caidos++

      }
        }
    }

if ( accion == "f"){
        if ( prot_mes == "cbr"){
                reenviados++
         }
    }

}

END{
printf ("\nPaquetes caidos: %i\n",caidos)>>salida2
printf ("\nPaquetes reenviados: %i\n\n",reenviados)>salida2
close(salida2)
}




BEGIN {
salida2="filtro5.txt"
mayor_id=-1
protoenv=0
protorec=0
```

```
protodrop=0
protofow=0

bprotoenv=0
bprotorec=0
bprotodrop=0
bprotofow=0

}
{
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$45
trace_level=$19
nodo_fuente=$5


if ( accion == "s"){

        if ( prot_mes == "aodv"){
           protoenv++
      bprotoenv = bprotoenv + bytes_paquete
        }
    }


if ( accion == "r"){

        if ( prot_mes == "aodv" && $35=="AODV" && trace_level="RTR"
){
                protorec++
        bprotorec = bprotorec + bytes_paquete
      }


    }

if ( accion == "d"){

        if ( prot_mes == "aodv"){ # Sólo interesan los paquetes CBR
                protodrop++
      bprotodrop = bprotodrop + bytes_paquete
        }
     }

if ( accion == "f"){

        if ( prot_mes == "aodv"){ # Sólo interesan los paquetes CBR
                protofow++
        bprotofow = bprotofow + bytes_paquete
        }
     }
```

```
}

END{
printf ("\nPaquetes de protocolo enviados: %i\n",protoenv)>>salida2
printf ("Bytes de protocolo enviados: %i\n",bprotoenv)>salida2
printf ("\nPaquetes de protocolo recibidos: %i\n",protorec)>salida2
printf ("Bytes de protocolo recibidos: %i\n",bprotorec)>salida2
printf ("\nPaquetes de protocolo desechados:
%i\n",protodrop)>salida2
printf ("Bytes de protocolo desechados: %i\n",bprotodrop)>salida2
printf ("\nPaquetes de protocolo reenviados:
%i\n",protofow)>salida2
printf ("Bytes de protocolo reenviados: %i\n\n",bprotofow)>salida2

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




BEGIN {
salida2="energia2.txt"
salida1="energia2.sh"
mayor_id=-1
bytes_env_0=0
rec_1=0
env_0=0
bytes_rec_11=0
var1=0
enerfactual=0.0000
enerfidle=0.0000
enerftrans=0.0000
enerfrec=0.0000
totalactual=0.00000
totalidle=0
totaltrans=0
totalrec=0
consumototal=0
energiaprotocoloidle=0
energiaprotocolotrans=0
energiaprotocolorec=0
energiaprotocolototal=0
}
{

accion=$1
nodo_actual=$3
prot_mes=$7
trace_level=$4
eneractual=$14
eneridle=$16
enertrans=$20
enerrec=$22
```

```
                if ( nodo_actual==nodo ){
                   enerfactual=eneractual
                 enerfidle=eneridle
                 enerftrans=enertrans
                 enerfrec=enerrec
             }


}
END{
printf (":")>>salida2
printf ("Energia final del nodo %s es:  %s\n
",nodo,enerfactual)>salida2
printf ("Energia en idle del nodo %s es:  %s\n
",nodo,enerfidle)>salida2
printf ("Energia en transmision del nodo %s es:  %s\n
",nodo,enerftrans)>salida2
enerfrec = substr(enerfrec,1,5)
printf ("Energia en recepcion del nodo %s es:  %s\n
",nodo,enerfrec)>salida2
consumototal=enerfidle+enerftrans+enerfrec
printf ("Consumo total del nodo %s es: %s\n\n
",nodo,consumototal)>salida2


if (nodo=="_0_" ||nodo=="_1_" ||nodo=="_2_" ||nodo=="_3_"
||nodo=="_4_" ||nodo=="_5_" ||nodo=="_6_" ||nodo=="_7_"
||nodo=="_8_" ||nodo=="_9_"){
nodo=substr(nodo,2,1)
}
else{
nodo=substr(nodo,2,2)
}

printf ("var1%s= %s idle \n",nodo,enerfidle)>>salida1
printf ("var2%s= %s trans \n",nodo,enerftrans)>salida1
printf ("var3%s= %s rec \n",nodo,enerfrec)>salida1
printf ("total%s= %s total \n",nodo,consumototal)>salida1


close(salida2)
}
```

EJECUCION SETDEST CBRGEN

```
setdest -v 1 -n 50 -p 0 -M 10 -t 600 -x 1500 -y 1500 >
/root/Desktop/ejems/simu13/simuAODV/movs1.tcl
ns cbrgen.tcl -type cbr -nn 49 -seed 1 -mc 30 -rate 2.0 >
/root/Desktop/ejems/simu13/simuAODV/traf3.tcl
```

```
#ns cbrgen.tcl -type tcp -nn 29 -seed 1 -mc 10 -rate 2.0 >
/root/Desktop/ejems/simu13/simuAODV/traf3.tcl
```

GENERACION DE TRAFICO

```
#
# nodes: 49, max conn: 30, send rate: 0.5, seed: 1
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set sink0 [new Agent/LossMonitor]
$ns_ attach-agent $node_(2) $sink0
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.5
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $sink0
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
# 4 connecting to 5 at time 56.333118917575632
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(1)
set sink1 [new Agent/LossMonitor]
$ns_ attach-agent $node_(5) $sink1
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 512
$cbr_(1) set interval_ 0.5
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $sink1
$ns_ at 56.333118917575632 "$cbr_(1) start"
#
# 4 connecting to 6 at time 146.96568928983328
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set sink2 [new Agent/LossMonitor]
$ns_ attach-agent $node_(6) $sink2
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 512
$cbr_(2) set interval_ 0.5
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $sink2
$ns_ at 146.96568928983328 "$cbr_(2) start"
#
# 6 connecting to 7 at time 55.634230382570173
```

```
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(3)
set sink3 [new Agent/LossMonitor]
$ns_ attach-agent $node_(7) $sink3
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 512
$cbr_(3) set interval_ 0.5
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $sink3
$ns_ at 55.634230382570173 "$cbr_(3) start"
#
# 7 connecting to 8 at time 29.546173154165118
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(4)
set sink4 [new Agent/LossMonitor]
$ns_ attach-agent $node_(8) $sink4
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 512
$cbr_(4) set interval_ 0.5
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $sink4
$ns_ at 29.546173154165118 "$cbr_(4) start"
#
# 7 connecting to 9 at time 7.7030203154790309
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(5)
set sink5 [new Agent/LossMonitor]
$ns_ attach-agent $node_(9) $sink5
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 512
$cbr_(5) set interval_ 0.5
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $sink5
$ns_ at 7.7030203154790309 "$cbr_(5) start"
#
# 8 connecting to 9 at time 20.48548468411224
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(6)
set sink6 [new Agent/LossMonitor]
$ns_ attach-agent $node_(9) $sink6
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 512
$cbr_(6) set interval_ 0.5
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
```

```
$ns_ connect $udp_(6) $sink6
$ns_ at 20.48548468411224 "$cbr_(6) start"
#
# 9 connecting to 10 at time 76.258212521792487
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(7)
set sink7 [new Agent/LossMonitor]
$ns_ attach-agent $node_(10) $sink7
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 512
$cbr_(7) set interval_ 0.5
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $sink7
$ns_ at 76.258212521792487 "$cbr_(7) start"
#
# 9 connecting to 11 at time 31.464945688594575
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(8)
set sink8 [new Agent/LossMonitor]
$ns_ attach-agent $node_(11) $sink8
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 512
$cbr_(8) set interval_ 0.5
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $sink8
$ns_ at 31.464945688594575 "$cbr_(8) start"
#
# 11 connecting to 12 at time 62.77338456491632
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(9)
set sink9 [new Agent/LossMonitor]
$ns_ attach-agent $node_(12) $sink9
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 512
$cbr_(9) set interval_ 0.5
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $sink9
$ns_ at 62.77338456491632 "$cbr_(9) start"
#
# 11 connecting to 13 at time 46.455830739092008
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(10)
set sink10 [new Agent/LossMonitor]
$ns_ attach-agent $node_(13) $sink10
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 512
```

```
$cbr_(10) set interval_ 0.5
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $sink10
$ns_ at 46.455830739092008 "$cbr_(10) start"
#
# 13 connecting to 14 at time 83.900868549896813
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(11)
set sink11 [new Agent/LossMonitor]
$ns_ attach-agent $node_(14) $sink11
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 512
$cbr_(11) set interval_ 0.5
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $sink11
$ns_ at 83.900868549896813 "$cbr_(11) start"
#
# 14 connecting to 15 at time 155.17211061677529
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(12)
set sink12 [new Agent/LossMonitor]
$ns_ attach-agent $node_(15) $sink12
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 512
$cbr_(12) set interval_ 0.5
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $sink12
$ns_ at 155.17211061677529 "$cbr_(12) start"
#
# 15 connecting to 16 at time 39.088702704333095
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(13)
set sink13 [new Agent/LossMonitor]
$ns_ attach-agent $node_(16) $sink13
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 512
$cbr_(13) set interval_ 0.5
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $sink13
$ns_ at 39.088702704333095 "$cbr_(13) start"
#
# 15 connecting to 17 at time 43.420613009212822
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(14)
```

```
set sink14 [new Agent/LossMonitor]
$ns_ attach-agent $node_(17) $sink14
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 512
$cbr_(14) set interval_ 0.5
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $sink14
$ns_ at 43.420613009212822 "$cbr_(14) start"
#
# 16 connecting to 17 at time 121.92280978985261
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(16) $udp_(15)
set sink15 [new Agent/LossMonitor]
$ns_ attach-agent $node_(17) $sink15
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 512
$cbr_(15) set interval_ 0.5
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $sink15
$ns_ at 121.92280978985261 "$cbr_(15) start"
#
# 16 connecting to 18 at time 137.20174070317378
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(16) $udp_(16)
set sink16 [new Agent/LossMonitor]
$ns_ attach-agent $node_(18) $sink16
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 512
$cbr_(16) set interval_ 0.5
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $sink16
$ns_ at 137.20174070317378 "$cbr_(16) start"
#
# 17 connecting to 18 at time 72.99343390995331
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(17)
set sink17 [new Agent/LossMonitor]
$ns_ attach-agent $node_(18) $sink17
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 512
$cbr_(17) set interval_ 0.5
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $sink17
$ns_ at 72.99343390995331 "$cbr_(17) start"
#
```

```
# 17 connecting to 19 at time 19.655724884781858
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(18)
set sink18 [new Agent/LossMonitor]
$ns_ attach-agent $node_(19) $sink18
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 512
$cbr_(18) set interval_ 0.5
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $sink18
$ns_ at 19.655724884781858 "$cbr_(18) start"
#
# 20 connecting to 21 at time 170.32769159894795
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(20) $udp_(19)
set sink19 [new Agent/LossMonitor]
$ns_ attach-agent $node_(21) $sink19
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 512
$cbr_(19) set interval_ 0.5
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $sink19
$ns_ at 170.32769159894795 "$cbr_(19) start"
#
# 20 connecting to 22 at time 160.44260791523504
#
set udp_(20) [new Agent/UDP]
$ns_ attach-agent $node_(20) $udp_(20)
set sink20 [new Agent/LossMonitor]
$ns_ attach-agent $node_(22) $sink20
set cbr_(20) [new Application/Traffic/CBR]
$cbr_(20) set packetSize_ 512
$cbr_(20) set interval_ 0.5
$cbr_(20) set random_ 1
$cbr_(20) set maxpkts_ 10000
$cbr_(20) attach-agent $udp_(20)
$ns_ connect $udp_(20) $sink20
$ns_ at 160.44260791523504 "$cbr_(20) start"
#
# 24 connecting to 25 at time 60.419296464146719
#
set udp_(21) [new Agent/UDP]
$ns_ attach-agent $node_(24) $udp_(21)
set sink21 [new Agent/LossMonitor]
$ns_ attach-agent $node_(25) $sink21
set cbr_(21) [new Application/Traffic/CBR]
$cbr_(21) set packetSize_ 512
$cbr_(21) set interval_ 0.5
$cbr_(21) set random_ 1
$cbr_(21) set maxpkts_ 10000
```

```
$cbr_(21) attach-agent $udp_(21)
$ns_ connect $udp_(21) $sink21
$ns_ at 60.419296464146719 "$cbr_(21) start"
#
# 26 connecting to 27 at time 46.258873029732555
#
set udp_(22) [new Agent/UDP]
$ns_ attach-agent $node_(26) $udp_(22)
set sink22 [new Agent/LossMonitor]
$ns_ attach-agent $node_(27) $sink22
set cbr_(22) [new Application/Traffic/CBR]
$cbr_(22) set packetSize_ 512
$cbr_(22) set interval_ 0.5
$cbr_(22) set random_ 1
$cbr_(22) set maxpkts_ 10000
$cbr_(22) attach-agent $udp_(22)
$ns_ connect $udp_(22) $sink22
$ns_ at 46.258873029732555 "$cbr_(22) start"
#
# 27 connecting to 28 at time 98.067954088592884
#
set udp_(23) [new Agent/UDP]
$ns_ attach-agent $node_(27) $udp_(23)
set sink23 [new Agent/LossMonitor]
$ns_ attach-agent $node_(28) $sink23
set cbr_(23) [new Application/Traffic/CBR]
$cbr_(23) set packetSize_ 512
$cbr_(23) set interval_ 0.5
$cbr_(23) set random_ 1
$cbr_(23) set maxpkts_ 10000
$cbr_(23) attach-agent $udp_(23)
$ns_ connect $udp_(23) $sink23
$ns_ at 98.067954088592884 "$cbr_(23) start"
#
# 28 connecting to 29 at time 47.128346453946243
#
set udp_(24) [new Agent/UDP]
$ns_ attach-agent $node_(28) $udp_(24)
set sink24 [new Agent/LossMonitor]
$ns_ attach-agent $node_(29) $sink24
set cbr_(24) [new Application/Traffic/CBR]
$cbr_(24) set packetSize_ 512
$cbr_(24) set interval_ 0.5
$cbr_(24) set random_ 1
$cbr_(24) set maxpkts_ 10000
$cbr_(24) attach-agent $udp_(24)
$ns_ connect $udp_(24) $sink24
$ns_ at 47.128346453946243 "$cbr_(24) start"
#
# 28 connecting to 30 at time 99.87114126788039
#
set udp_(25) [new Agent/UDP]
$ns_ attach-agent $node_(28) $udp_(25)
set sink25 [new Agent/LossMonitor]
$ns_ attach-agent $node_(30) $sink25
set cbr_(25) [new Application/Traffic/CBR]
```

```
$cbr_(25) set packetSize_ 512
$cbr_(25) set interval_ 0.5
$cbr_(25) set random_ 1
$cbr_(25) set maxpkts_ 10000
$cbr_(25) attach-agent $udp_(25)
$ns_ connect $udp_(25) $sink25
$ns_ at 99.87114126788039 "$cbr_(25) start"
#
# 33 connecting to 34 at time 170.4269140541679
#
set udp_(26) [new Agent/UDP]
$ns_ attach-agent $node_(33) $udp_(26)
set sink26 [new Agent/LossMonitor]
$ns_ attach-agent $node_(34) $sink26
set cbr_(26) [new Application/Traffic/CBR]
$cbr_(26) set packetSize_ 512
$cbr_(26) set interval_ 0.5
$cbr_(26) set random_ 1
$cbr_(26) set maxpkts_ 10000
$cbr_(26) attach-agent $udp_(26)
$ns_ connect $udp_(26) $sink26
$ns_ at 170.4269140541679 "$cbr_(26) start"
#
# 33 connecting to 35 at time 139.08523271749041
#
set udp_(27) [new Agent/UDP]
$ns_ attach-agent $node_(33) $udp_(27)
set sink27 [new Agent/LossMonitor]
$ns_ attach-agent $node_(35) $sink27
set cbr_(27) [new Application/Traffic/CBR]
$cbr_(27) set packetSize_ 512
$cbr_(27) set interval_ 0.5
$cbr_(27) set random_ 1
$cbr_(27) set maxpkts_ 10000
$cbr_(27) attach-agent $udp_(27)
$ns_ connect $udp_(27) $sink27
$ns_ at 139.08523271749041 "$cbr_(27) start"
#
# 35 connecting to 36 at time 32.17101298839367
#
set udp_(28) [new Agent/UDP]
$ns_ attach-agent $node_(35) $udp_(28)
set sink28 [new Agent/LossMonitor]
$ns_ attach-agent $node_(36) $sink28
set cbr_(28) [new Application/Traffic/CBR]
$cbr_(28) set packetSize_ 512
$cbr_(28) set interval_ 0.5
$cbr_(28) set random_ 1
$cbr_(28) set maxpkts_ 10000
$cbr_(28) attach-agent $udp_(28)
$ns_ connect $udp_(28) $sink28
$ns_ at 32.17101298839367 "$cbr_(28) start"
#
# 36 connecting to 37 at time 16.444244867397586
#
set udp_(29) [new Agent/UDP]
```

```
$ns_ attach-agent $node_(36) $udp_(29)
set sink29 [new Agent/LossMonitor]
$ns_ attach-agent $node_(37) $sink29
set cbr_(29) [new Application/Traffic/CBR]
$cbr_(29) set packetSize_ 512
$cbr_(29) set interval_ 0.5
$cbr_(29) set random_ 1
$cbr_(29) set maxpkts_ 10000
$cbr_(29) attach-agent $udp_(29)
$ns_ connect $udp_(29) $sink29
$ns_ at 16.444244867397586 "$cbr_(29) start"
#
#Total sources/connections: 20/30
#
```

RESULTADOS ESTADISTICOS

```
BEGIN {
salida1="enviados"
salida2="recibidos"
salida3="reenviados"
salida4="desechados"
mayor_id=-1
bytes_env_0=0
rec_1=0
env_0=0
fow_1=0
dro_1=0
bytes_rec_1=0
bytes_fow_1=0
bytes_dro_1=0
var1=0
totalPenv=0
toralPrec=0
totalBenv=0
totalBrec=0
totalPfow=0
totalPdro=0
totalBfow=0
totalBdro=0
}
{
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$35
trace_level=$19
nodo_fuente=$5


if(accion!="SFESTs" && accion!="SFs" && trace_level=="AGT"){
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==nodo){
                   fuente
                   bytes_env_0 = bytes_env_0 + bytes_paquete
                                        env_0++

                                   }
           totalPenv++
           totalBenv = totalBenv + bytes_paquete
           printf("%6f %6f\n",$3,totalPenv)>>salida1
        }
    }
}
if ( accion == "r" && prot_mes == "cbr"){
    if (nodo_actual == nodo_destino){
        if (nodo_actual==nodo){
             bytes_rec_1 = bytes_rec_1 + bytes_paquete - 20
                         # la cabecera IP de 20 bytes
             rec_1++
        }
      totalPrec++
        totalBrec = totalBrec + bytes_paquete - 20
      printf("%6f %6f\n",$3,totalPrec)>>salida2

    }
}
}

if ( accion == "d" && prot_mes == "cbr"){

        if (nodo_actual==nodo){
             bytes_dro_1 = bytes_dro_1 + bytes_paquete - 20
                         # la cabecera IP de 20 bytes
             dro_1++
                      }
        totalPdro++
        totalBdro = totalBdro + bytes_paquete - 20
        printf("%6f %6f\n",$3,totalPdro)>>salida4


}

if ( accion == "f" && prot_mes == "cbr"){

        if (nodo_actual==nodo){
             bytes_fow_1 = bytes_fow_1 + bytes_paquete - 20
                         # la cabecera IP de 20 bytes
             fow_1++
                      }
        totalPfow++
        totalBfow = totalBfow + bytes_paquete - 20
        printf("%6f %6f\n",$3,totalPfow)>>salida3
```

```
}



}
END{
close(salida2)
}
```

```
BEGIN {
salida1="Penviados"
salida2="Precibidos"
salida3="Preenviados"
salida4="Pdesechados"
mayor_id=-1
bytes_env_0=0
rec_1=0
env_0=0
fow_1=0
dro_1=0
bytes_rec_1=0
bytes_fow_1=0
bytes_dro_1=0
var1=0
totalPenv=0
toralPrec=0
totalBenv=0
totalBrec=0
totalPfow=0
totalPdro=0
totalBfow=0
totalBdro=0
}
{
accion=$1
nodo_actual=$9
bytes_paquete=$37
id_paquete=$41
nodo_destino=$7
prot_mes=$35
trace_level=$19
nodo_fuente=$5



if ( accion == "s"){
        if ( prot_mes == "AODV"){ # Sólo interesan los paquetes CBR
                  if ( nodo_actual==nodo){
                                          bytes_env_0 = bytes_env_0 +
bytes_paquete
                                          env_0++

                                  }
           totalPenv++
           totalBenv = totalBenv + bytes_paquete
           printf("%6f %6f\n",$3,totalPenv)>>salida1
        }
    }

if ( accion == "r" && prot_mes == "AODV"){

        if (nodo_actual==nodo){
             bytes_rec_1 = bytes_rec_1 + bytes_paquete - 20
             rec_1++
        }
```

```
      totalPrec++
        totalBrec = totalBrec + bytes_paquete - 20
      printf("%6f %6f\n",$3,totalPrec)>>salida2


}


if ( accion == "d" && prot_mes == "AODV"){

        if (nodo_actual==nodo){
             bytes_dro_1 = bytes_dro_1 + bytes_paquete - 20

             dro_1++
                     }
        totalPdro++
        totalBdro = totalBdro + bytes_paquete - 20
        printf("%6f %6f\n",$3,totalPdro)>>salida4


}

if ( accion == "f" && prot_mes == "AODV"){

        if (nodo_actual==nodo){
             bytes_fow_1 = bytes_fow_1 + bytes_paquete - 20
             fow_1++
                     }
        totalPfow++
        totalBfow = totalBfow + bytes_paquete - 20
        printf("%6f %6f\n",$3,totalPfow)>>salida3


}



}
END{
close(salida2)
}




rm -f enviados
rm -f recibidos
rm -f reenviados
rm -f desechados

#for i in `seq 0 14`
#do
#echo "nodo $i..... ok"
#awk -v nodo=$i -v ban=0 -f estad1.awk
/root/Desktop/ejems/simu11/ayu1.tr
```

```
#done
awk -v nodo=$i -v ban=1 -f estad1.awk
/root/Desktop/ejems/simu13/simuAODV/ayu1.tr
xgraph -t Paquetes_tiempo -x tiempo -y Paquetes enviados recibidos
reenviados desechados -geometry 900x500 &



rm -f Penviados
rm -f Precibidos
rm -f Preenviados
rm -f Pdesechados

#for i in `seq 0 14`
#do
#echo "nodo $i..... ok"
#awk -v nodo=$i -v ban=0 -f estad1.awk
/root/Desktop/ejems/simu11/ayu1.tr
#done
awk -v nodo=$i -v ban=1 -f estad2.awk
/root/Desktop/ejems/simu13/simuAODV/ayu1.tr
xgraph -t Paquetes_tiempo -x tiempo -y Paquetes Penviados
Precibidos Preenviados Pdesechados -geometry 900x500 &
```

EJECUCION DE ANALISIS

```
cd /root/Desktop/ejems/simu13/simuAODV
./exfilt3.sh
./exfilt4.sh
./exfilt5.sh
./exfilt7.awk
./exfilt8.sh
./exfilt9.sh

cd /root/Desktop/ejems/simu13/simuAODV/estadistico
./exestad1.sh
./exestad2.sh

cd /root/Desktop/ejems/simu13/simuAOMDV
./exfilt3.sh
./exfilt4.sh
./exfilt5.sh
./exfilt7.awk
./exfilt8.sh
./exfilt9.sh

cd /root/Desktop/ejems/simu13/simuAOMDV/estadistico
./exestad1.sh
./exestad2.sh

cd /root/Desktop/ejems/simu13/simuDSR
```

```
./exfilt3.sh
./exfilt4.sh
./exfilt5.sh
./exfilt7.awk
./exfilt8.sh
./exfilt9.sh

cd /root/Desktop/ejems/simu13/simuDSR/estadistico
./exestad1.sh
./exestad2.sh

cd /root/Desktop/ejems/simu13/simuOLSR
./exfilt3.sh
./exfilt4.sh
./exfilt5.sh
./exfilt7.awk
./exfilt8.sh
./exfilt9.sh

cd /root/Desktop/ejems/simu13/simuOLSR/estadistico
./exestad1.sh
./exestad2.sh
```

EJECUCION DE SIMULACION

```
cd /root/Desktop/ejems/simu13/simuAODV
ns ayu1.tcl
ns ayu1ener.tcl
cd /root/Desktop/ejems/simu13/simuAOMDV
ns ayu1.tcl
ns ayu1ener.tcl
cd /root/Desktop/ejems/simu13/simuDSR
ns ayu1.tcl
ns ayu1ener.tcl
cd /root/Desktop/ejems/simu13/simuOLSR
ns ayu1.tcl
ns ayu1ener.tcl
```